Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Mosawer Nurzai

Couch 2.0 - Eine kontextsensitive Installation für eine Smart-Home-Umgebung

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Mosawer Nurzai

Couch 2.0 - Eine kontextsensitive Installation für eine Smart-Home-Umgebung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Kai von Luck
Zweitgutachter: Prof. Dr. rer. nat. Gunter Klemke

Eingereicht am: 09. August 2013

Mosawer Nurzai

Thema der Arbeit

Couch 2.0 - Eine kontextsensitive Installation für eine Smart-Home-Umgebung

Stichworte

Couch, Kontext, kontextsensitiv, Smart Home, Context awareness, ubiquitäres Rechnen

Kurzzusammenfassung

Diese Arbeit stellt einen Beitrag zur Forschung von Smart Home bzw. Smart Home environment dar. In dieser Bachelorarbeit wird ein kontextsensitives System entwickelt, welches in einer Smart-Home-Umgebung eingesetzt werden soll. Dabei spielt der Wohnbereich und besonders die Couch eine besondere Rolle. Die Situation eines Benutzers auf der Couch wird ermittelt, hierfür wurden Szenarien aufgestellt. Es werden verschiedene Kontexte eines Benutzers analysiert und erfasst. Anhand der Kontexte und den Aktionen des Benutzers soll das System reaktiv handeln. In einem Designentwurf wird das System und das Verhalten des System spezifiziert. Wie das System realisiert wird und welche Hard- und Softwarekomponenten notwendig sind, wird ebenfalls erläutert. Die wichtigsten Elemente der verschiedenen Teilkomponenten werden nacheinander dargelegt.

Mosawer Nurzai

Title of the paper

Couch 2.0 - A context-sensitive installation for a Smart Home environment

Keywords

Couch, Context, Context-sensitive, Smart Home, Context awareness, Ubiquitous Computing

Abstract

This work represents a contribution to the research of Smart Home and Smart Home environment. In this thesis a context-sensitive system is developed, which will be applied in a smart-home-environment. The living room and especially the couch plays a special role. The situation of a user on the couch is to be determined, therefore use cases were set up. Different contexts of a user are analyzed and recorded. Based on the contexts and the actions of the user the system should act reactively. The system and the behavior of the system will be specified in a draft design. This is demonstrated in this work. How the system is implemented and which hardware and software components are needed, will also be explained. The important elements of the different subcomponents will presented one after another.

Danksagung

Zuallererst möchte ich Allah danken. Er hat mir die Kraft gegeben diese Arbeit fertig zu stellen.
Ein großes Dankeschön geht an meinen Betreuer Herr Prof. Dr. rer. nat. Kai von Luck. Danke für die Ratschläge, Unterstützung und die gute Betreuung.
Ich danke auch meinen Zweitgutachter Herr Prof. Dr. rer. nat. Gunter Klemke.
Mein tiefster Dank geht an meine Eltern, sie standen mir stets zur Seite.

Inhaltsverzeichnis

1 Einleitung

In der heutigen Gesellschaft dringen Computer, verteilte Systeme und jegliche andere Form von Rechnereinheiten in jeglichen Aspekten unseren Lebens ein. Wir sind beinahe gezwungen mit Computern umgehen zu können, damit wir in der heutigen Gesellschaft mitreden und mitwirken können. Der Mensch und die Technik nähern sich in vielen Bereichen, sei es in der Freizeit oder in der Arbeitswelt. Sei es der Kühlschrank, der per W-LAN die Wetterprognose für Morgen anzeigt oder die Selbstkassierung am Supermarkt mittels Computer-Touch-Oberfläche. Jedoch fällt es immer noch vielen Menschen schwer mit Technik umzugehen, da sich die Technik nicht dynamisch an den Menschen anpasst, sondern der Mensch gezwungen ist sich an die Technik anzupassen.
Die Forschung bezüglich intelligenter Umgebung (Ambient Intelligence) schafft neue Maßstäbe. Gemeinsam mit dem Aspekt des ubiquitären Computerwesen und der Beachtung des Kontext versucht man die Umgebung an den Menschen anzupassen. Der Mensch soll so wenig wie möglich von den Computern um ihn herum Bescheid wissen. Das einzige Ziel ist es seine Bedürfnisse zu erfüllen.
Diese Arbeit beschäftigt sich mit der Entwicklung eines solchen Systems bzw. Applikation unter der Beachtung der Anforderungen der oben genannten Forschungsgebiete.
Menschen, die wenig bis kaum Erfahrung mit Computern und auch diejenige die Fachwissen in diesem Gebiet haben, werden hier angesprochen.

1.1 Ziel der Arbeit

Ziel der Arbeit ist es ein System aufzubauen, welches den Anforderungen an intelligente Umgebung genügt. Dabei wird auf die Analyse, dem Design und die Realisierung des System eingegangen. Im Rahmen des langlebigen Forschungsprojekt „Living Place“ an der HAW Hamburg wird das System aufgebaut. Es bietet eine Umgebung um die Fragen und die oben genannte Problematik zu klären.
Im „Living Place“ wird die Couch-Region betrachtet und verschiedene Szenarien abgebildet.

Der Couch-Potato[1] soll Protagonist in diesen Szenarien sein. Es wird verschiedene Hardware genutzt, um den Couch-Potato und seine Umgebung zu analysieren. Später in der Arbeit wird darauf genauer eingegangen. Es soll ein reaktives kontextabhängiges System realisiert werden.

1.2 Gliederung

Die Arbeit ist in sieben Kapitel aufgeteilt. Jeder der Kapitel baut auf die Erarbeitungen und das Wissen des vorherigen Kapitels auf.
Das zweite Kapitel beschreibt kurz die Grundlagen, um die Hintergründe und das angewandte Fachwissen dieser Arbeit zu verstehen.
Im dritten Kapitel findet eine Analyse der Arbeit und des zu entwickelten Systems statt. Es werden Szenarien, Kontexte und Anforderungen formuliert.
Die Arbeit hat einen theoretischen und praktischen Teil. Der praktische Teil findet in einer speziellen Laborumgebung statt. Diese Laborumgebung wird im vierten Kapitel beschrieben.
Um das zu System realisieren zu können, müssen Entwürfe und Verhalten des Systems vordefiniert werden. Im fünften Kapitel werden diese Aspekte behandelt und das Design des System definiert.
Im sechsten Kapitel wird die Realisierung des Systems aufgezeigt. Das Design im fünften Kapitel dient als Grundlage für die Realisierung. Es werden Codefragmente aufgezeigt und erklärt. Am Ende des Kapitels wird ein Fazit gezogen.
Das siebte Kapitel enthält eine kurze Zusammenfassung der Arbeit und einen Ausblick.

[1] Jemand, der sich nicht sportlich betätigt, sondern vorwiegend (fernsehend) auf der Couch sitzt oder liegt.

2 Grundlagen

In diesem Kapitel sollen die Grundlagen für das bessere Verständnis dieser Arbeit erläutert werden. Die Forschungsgebiete Ubiquitous Computing und Smart Home werden aufgezeigt.

2.1 Ubiquitous Computing

Ubiquitous Computing ist das allgegenwärtige Dasein von Computern. Es beschreibt, dass sich Computer in allzu allen alltäglichen und sonstigen Gegenständen befinden. Jeder dieser Computer übernimmt eine Aufgabe. Interaktionen können durch aktive und bewusste Teilnahme oder auch ohne jegliche Wahrnehmung der Computer, vom Menschen eingeleitet werden. Mark Weiser ist einer der Pioniere in Ubiquitous Computing. Er formulierte 1997, dass Computer sich in allen Gegenständen befinden werden. Eine Person wird, laut seiner Aussage, mit 100 von Computern gleichzeitig interagieren. Jeder dieser Computer ist eingebettet in die Umgebung und kommuniziert drahtlos miteinander (Weiser und Brown [1997], Weiser [1991]). Laut William Buxton charakterisiert sich Ubiquitous Computing durch zwei Eigenschaften (Buxten [1995])

Allgegenwart (Ubiquity): Interaktionen werden nicht durch eine einzelne Station gelenkt. Die Berechnungen finden überall statt.

Tranzparenz (Transparency): Die Technologie ist unsichtbar und nicht zugänglich. Sie soll im Umfeld des Haus oder auf dem Arbeitsplatz integriert sein, beispielsweise im Schreibtisch, Stuhl, Buch oder in den Wänden.

> *„The third wave of computing is that of ubiquitous computing, whose cross-over point with personal computing will be around 2005-2020. The „UC“ era will have lots of computers sharing each of us. Some of these computers will be the hundreds we may access in the course of a few minutes of Internet browsing. Others will be imbedded in walls, chairs, clothing, light switches, cars - in everything. UC is fundamentally characterized by the connection of things in the world with computation.“-Weiser und Brown [1997]*

Ubiquitous computing ist eine dritte Welle des Computerseins. Die erste Welle beschreibt, dass mehrere Menschen ein Computer bedienen. Die zweite Wellte ist die Ära des PC (Personal Computer). Jeder Mensch hat ein Computer.

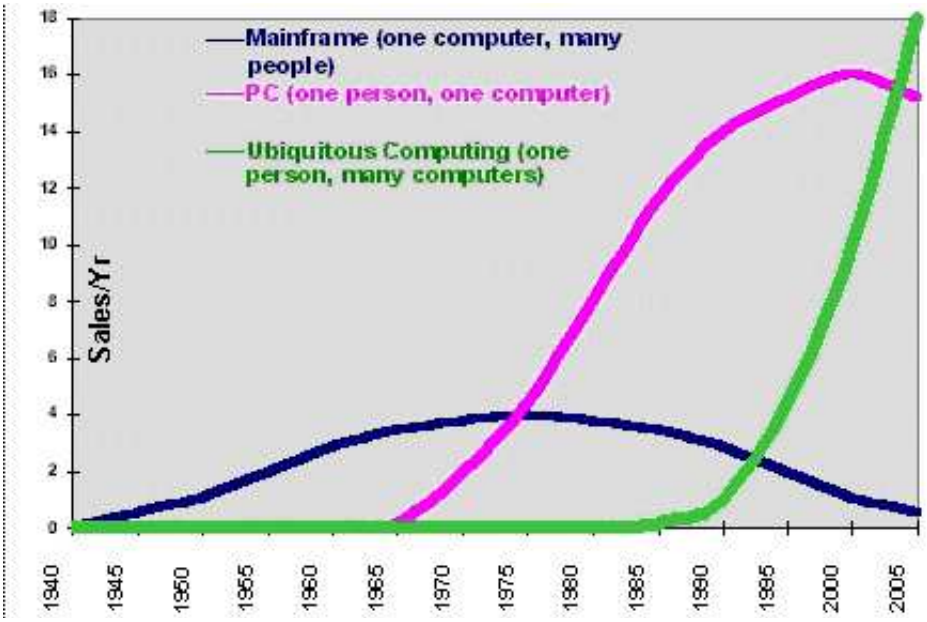

Abbildung 2.1: Strömungen des Computerwesens
Quelle: Koudounas und Iqbal [2013]

Die dritte Welle ist die Ära von mehreren Computern pro Person. Zur heutigen Zeit befinden sich Computer überall. Wir nehmen sie kaum noch wahr. Diese Computer sind keine Personal Computers, also PC, sie sind eingebettet und haben eine für sie definierte Aufgabe.

Ambient Intelligence

Ambient Intelligence ist ein Prozess, welche Technologie im Leben des Menschen in so einer Art einführt, dass der Mensch selber nicht aktiv etwas lernen muss um es zu bedienen. Eher wird er vom Ambient Intelligence bedient. Das System benötigt kein spezielles Interface, der Mensch selbst dient als „Manual“, damit das System entsprechend reagieren kann. Ambient Intelligence zu schaffen ist keine einfache Aufgabe, denn es erfordert die neuste Technologie und entsprechende Hardware für die intelligente Umgebung. Es reicht nicht aus bei Berührung eines Türknaufes, die Hausklingel läuten zu lassen: Die Umgebung muss sich ständig an den Personen, die sich in ihr befinden, anpassen. Dies soll ohne ständige manuelle Konfiguration der Personen statt finden. Die Bedürfnisse der Personen sollen durch das System berücksichtigt werden. Abstrakt gesagt, soll eine lernende Maschine erschaffen werden (Riva u. a. [2005]).

2.2 Smart Home

Die Idee des intelligenten Hauses, beschreibt eine Umgebung in der alle Bedürfnisse der Bewohner durch technologische Mittel gelöst werden. Darüber hinaus kann es zusätzliche Dienste, wie beispielsweise Unterhaltung leisten. In der Vorstellung eines Smart Homes denkt man oft an Szenarien, wie das automatisierte Kochen der Lieblingsspeise oder das selbstreinigende Bad. In der Wirklichkeit werden Anwendungen in diese Richtung versucht zu entwickelt, jedoch wird viel mehr versucht als diese Vorstellung zu erfüllen, denn das „Wie?“ spielt in den Laboren rund um Smart Homes eine entscheidende Rolle. Wie sollen solche Umgebungen aufgebaut werden und wie können die Bedürfnisse, Verlangen und Intentionen eines Menschen erfasst werden?
Die computergestützten Technologien eines Smart Homes müssen in die Umgebung eingebettet und unaufdringlich sein. Sie müssen mit dem Benutzer auf eine natürliche Art und Weise interagieren (Meyer und Rakotonirainy [2003]), denn sie könnten im alltäglichen Leben als störend empfunden werden oder die Bewohner des Hauses könnten die Technologie beschädigen.

> *„Das Smart Home ist ein privat genutztes Heim (z.B. Eigenheim, Mietwohnung), in dem die zahlreichen Geräte der Hausautomation (wie Heizung, Beleuchtung, Belüftung), Haushaltstechnik (wie z.B. Kühlschrank, Waschmaschine), Konsumelektronik und Kommunikationseinrichtungen zu intelligenten Gegenständen werden, die sich an den Bedürfnissen der Bewohner orientieren. Durch Vernetzung dieser Gegenstände untereinander können neue Assistenzfunktionen und Dienste zum Nutzen des Bewohners bereitgestellt werden und einen Mehrwert generieren, der über den einzelnen Nutzen der im Haus vorhandenen Anwendungen hinausgeht“ -Strese u. a. [2010]*

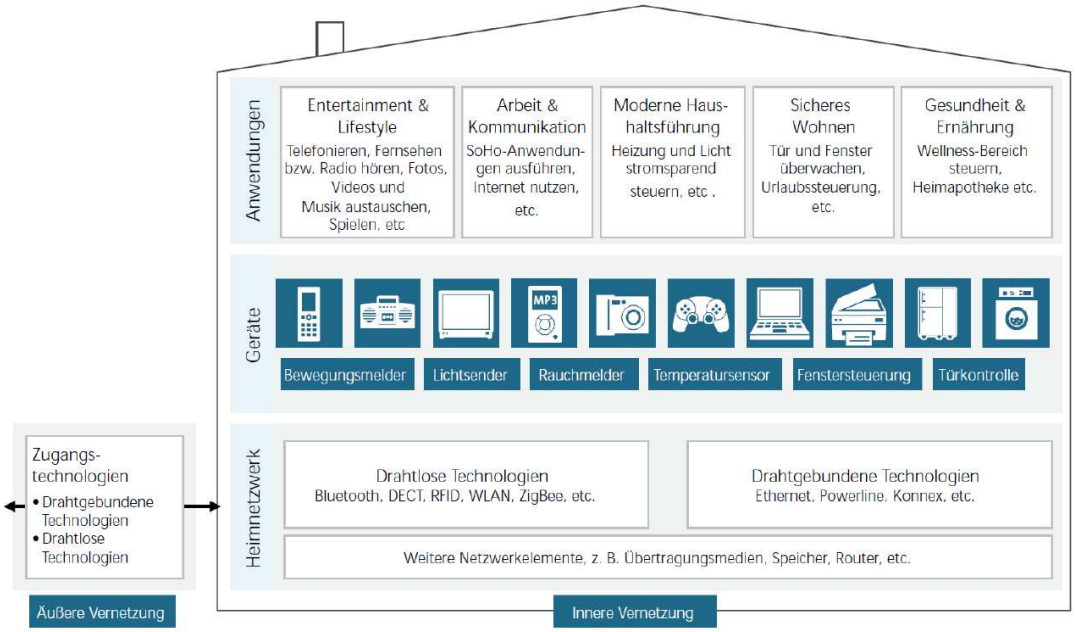

Abbildung 2.2: Visualisierung eines Smart Homes
Quelle: Glasberg und Feldner [2009]

In Abbildung 2.2 werden die wichtigen Elemente eines Smart Homes aufgezeigt. Dies soll ein exemplarisches Beispiel sein, jedoch entnommen aus einem Leitfaden. Für die allgemeine Kommunikation und die Kommunikation der Geräte wird ein Heimnetzwerk verwendet. So sind die Geräte untereinander durch das Heimnetzwerk verbunden. Durch verschiedene Anwendungen, aufbauend auf das Smart Home, können verschiedene Funktionalitäten und Intelligenz geschaffen werden. Benutzeranwendungen erweitern Funktionalitäten des Smart Homes.

3 Analyse

> *„Das Forschungsvorhaben folgt der Vision, dass technische Systeme der Zukunft Companion-Systeme sind - kognitive technische Systeme, die ihre Funktionalität vollkommen individuell auf den jeweiligen Nutzer abstimmen: Sie orientieren sich an seinen Fähigkeiten, Vorlieben, Anforderungen und aktuellen Bedürfnissen und stellen sich auf seine Situation und emotionale Befindlichkeit ein. Dabei sind sie stets verfügbar, kooperativ und vertrauenswürdig und treten ihrem Nutzer als kompetente und partnerschaftliche Dienstleister gegenüber."-Ulm u. a. [2010]*

Diese aktuelle Diskussion, die weit über diese Bachelor-Arbeit hinaus geht, zeigt eine Vision auf, die durch Forschung angestrebt wird. Es werden in dieser Analyse einige Punkte genannt, die in diese Richtung gehen. Die Kontextbetrachtung, die ein Teil dieser Vision ist wird in diesem Kapitel behandelt, genauso wie der, durch das System entstehende, Komfort.
Ein weiterer Punkt, der in der Analyse hervortritt ist Ubiquitous Computing. Mark Weiser hat dies in seinem Aufsatz „The Computer for the 21st Century" wie folgt beschrieben:

> *„The third wave of computing is that of ubiquitous computing, whose cross-over point with personal computing will be around 2005-2020. The „UC" era will have lots of computers sharing each of us. Some of these computers will be the hundreds we may access in the course of a few minutes of Internet browsing. Others will be imbedded in walls, chairs, clothing, light switches, cars - in everything. UC is fundamentally characterized by the connection of things in the world with computation." -Weiser und Brown [1997]*

Diese Arbeit soll eine ubiquitäre Schnittstelle schaffen. Es soll möglich viel vom Benutzer verborgen bleiben und die Konzentration auf sein Wohlbefinden gerichtet sein.
Das System welches verborgen wird, ist wie Weiser sagt, ein Netzwerk welches untereinander kommuniziert. Neben Sensoren die ein Netzwerk bilden, werden unterschiedliche Arten von Computern eingesetzt. Im weiteren Verlauf der Arbeit soll dies verdeutlicht werden.

3.1 Context aware computing

Der Mensch ist in der Lage seine Umgebung vollkommen wahrzunehmen und darauf entsprechend zu reagieren. Ihm ist es möglich auf die Einwirkungen, die ihm widerfahren einzugehen. Er betrachtet jeglichen Kontext, durch seine fünf Sinne, seinen Verstand und durch seine Emotionen, sprich er kann die Informationen die um ihn herum sind richtig verarbeiten und gleichzeitig im hohen Tempo darauf reagieren.
Diese Vorstellung wünscht man sich gerne auch für Computer - also die Betrachtung des Kontextes und die korrekte Interpretation, verbunden mit einer angemessenen, richtigen Reaktion. In Context aware computing betrachtet man diese Herausforderung. So soll der Umgang und die Interaktion mit Computern vereinfacht werden. Um einen besseren Einblick in Context aware computing zu bekommen, wird der Begriff Kontext erstmal erklärt und anschließend wird auf Context aware computing eingegangen:

Kontext

> *„Context is any information that can be used to characterize the situation of an entity. An entitiy is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.“*
> *-Dey und Abowd [2013]*

Ein Kontext wird also als jegliche Information betrachtet, die die Situation einer Entität charakterisiert. Diese Entität erzwingt oder verursacht eine Interaktion. Mit dieser abstrakten Definition kann der Entwickler die benötigen Kontexte einfacher spezifizieren.
Um den Begriff zu veranschaulichen, erklären die Autoren diesen anhand eines Beispiels: Es gibt eine Applikation, die die Aufgabe hat eine Liste von Gewichten in einer Tabelle zu verwalten. *Entitäten* sind hier der Benutzer und die Applikation selbst. Zwei Informationen werden betrachtet, Gegenwart von anderen Personen und der Standort. Nun muss entschieden werden sind diese Informationen, für diese Anwendung brauchbare, *Kontexte*. Da die Gegenwart von anderen Personen, den Benutzer und die Applikation nicht beeinflussen ist dies kein Kontext. Der Standort des Benutzers ist allerdings ein Kontext, denn wenn der Benutzer sich in den Vereinigten Staaten befindet, so wird die Summe der Gewichte in Pfund und Unzen angezeigt. In Kanada beispielsweise würde dies Kilogramm und Gramm sein. So wird der Standort als Kontext betrachtet, da er die Situation des Benutzern beeinflusst.

Kategorie von Kontext

Von einer kontextbewussten Applikation erwartet man, dass sie auf das wer, wo, wann und was der Entität schaut, um das warum zu identifizieren. Um den Entwickler bei der Suche nach Kontexten zu unterstützen, werden Kontexte in Kategorien klassifiziert. Typische Kategorien sind Standort (location), Identität (identity), Aktivität (activity) und Zeit (time). Der Entwickler kann nun die aufkommenden Informationen leichter zuordnen. Diese Kategorien werden *primary context* bezeichnet (Dey und Abowd [2013]). Sie lösen nicht alle auftauchenden Fragen, jedoch decken sie die Wichtigsten ab. Um möglichst alle Fragen zu beantworten gibt es die *secondary context* (Dey und Abowd [2013]). Die secondary context werden von den primary context indiziert. Ein einfaches Beispiel ist: die Telefonnummer einer Person ist ein secondary context und die Identität der primary context. Es kann eine Indizierung von Identiät auf Telefonnummer passieren.

Context aware computing

> *„A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.“ -Dey und Abowd [2013]*

Diese Definition veranschaulicht, dass ein System Kontexte benutzt um, in Abhängigkeit der Aufgabe des Benutzers, Informationen zu liefern oder einen Dienst zu erfüllen. Es wird bei der Definition nicht auf die Entdeckung und Interpretation der Kontexte eingegangen. Hauptinhalt der Definition ist die Antwort des Systems. Somit wird eine sehr allgemeine Definition präsentiert.
Dey und Abowd (Dey und Abowd [2013]) haben spezifiziert, was ein System leisten sollte, um die Anforderungen von einem Context aware computing System zu erfüllen:

1. Präsentation von Informationen und Dienstleistungen für ein Benutzer
2. Automatische Ausführung eines Dienstes und
3. Markierung von Kontext-Informationen für einen späteren Abruf.

Die Punkte stellen drei wichtige Aspekte von Context aware computing dar. Bei der Markierung von Kontext-Informationen, wird es ermöglicht Kontexte einzuordnen und zu speichern. Dadurch hat man eine Quelle für die spätere Interpretation des Kontextes und die darauffolgende Interaktion mit dem Benutzer. Die automatische Ausführung eines Dienstes sorgt dafür, dass der Benutzer so wenig wie möglich aktiv etwas machen muss. Weder muss ein Knopf

betätigt, noch eine Zustimmung eingeholt werden. Die Reaktion des Systems wird durch die Präsentation der Informationen und Dienstleistungen ausgedrückt. Spätestens hier erfährt der Benutzer, dass ein computergesteurtes System ihm Aufgaben des Lebens vereinfachen. So findet eine Interaktion mit dem Menschen und dem (Computer-)System statt.
Auf die obigen drei Punkte wird im Kapitel Realisierung eingegangen - welche Faktoren bzw. Komponenten diese Punkte erfüllen und was benötigt wurde um sie zu erfüllen.

3.2 Szenarien

Damit man ein besseres Bild über die Szenarien, die in diesem System „Couch 2.0“ auftauchen könnten, bekommt folgt eine kurze Geschichte über den Protagonisten Bob, in der die Anwendungsfälle abgebildet werden. Die Anwendungsfälle werden dann anschließend in einem Use-Case Diagramm dargestellt und erläutert.

3.2.1 Geschichte über Bob

Auf dem Heimweg freut sich Bob schon auf seine Couch, um endlich mal zu relaxen und den gestressten Arbeitstag zu vergessen. Zuhause angekommen muss er feststellen wie seine kleine Tochter Alice vergeblich versucht den Fernseher an zu bekommen. Sie weiß allerdings nicht wofür die Familie Bob eines Abends, einer nach dem anderen, vorm TV für eine gewisse Zeit still sitzen mussten. Die Gesichter der Familie wurden aufgezeichnet und Alice darf, laut System, nicht um diese Uhrzeit Fern schauen. Verärgert und traurig geht Alice auf ihr Zimmer. Bob betritt die vordefinierte Couch-Region, gibt sich zu erkennen und der TV schaltet sich von alleine, ohne Beteiligung von Bob, ein. Dies funktioniert nur wiederum, weil Bob die gewünschte Zeitspanne, in dem der TV sich anschalten soll, vorher eingestellt hat. Die Beleuchtung wird für ein entspanntes Fernseh-Erlebnis automatisch, in Abhängigkeit der herrschenden Lichtverhältnisse, angepasst.
Nach einiger Zeit kommt auch Bobs Frau, Christina, von der Arbeit nach Hause und will erst mal schnell, nach einem kurzen Gespräch mit Bob, unter die Dusche. Bobs Aufmerksamkeit auf den Fernseher wird für eine lange Zeit entzogen, da das kurze Gespräch zu einer ausführliche Unterhaltung über die Finanzen der Familie Bob wurde. Der TV wird zunächst leiser. Nach der Unterhaltung wendet sich Bob wieder dem TV zu und die Lautstärke wird wieder geregelt, hätte sich Bob dem TV nicht zugewendet, so wäre er automatisch aus gegangen.
Allmählich macht sich der Arbeitsstress bemerkbar und Bob fällt unbemerkt auf die Couch und schläft ein. Seine Aufmerksamkeit ist nicht mehr dem TV gewidmet. Dieses Nickerchen wird

gefördert, indem der Fernseher sich automatisch ausschaltet und die Beleuchtung abgeschaltet bzw. sehr schwach wird.

Bob

Bob ist eine Couch-Potato und der Protagonist der Geschichte. Neben seiner Arbeit und Familie widmet er sich hauptsächlich der Couch und dem Fernseher zu. Dabei möchte er so gut wie es geht vom System verwöhnt werden. Denn beim Fernsehen will er besten Komfort haben. Am Besten ohne Anstrengung und Bewegung seinerseits. Oft schläft er beim Fernsehen ein, dies treibt die Stromkosten der Familie beachtlich in die Höhe. Da Bob sich mit Technik nicht auskennt, erfreut er sich auf die nicht sichtbaren Rechnereinheiten und dass er so gut wie nichts Einstellung muss, um diesen Komfort zu erhalten.

Alice

Alice ist die kleine Tochter von Bob. Sie hat vom neuen Fernseher-System gehört und will es so schnell wie möglich austesten. Da sie noch zu jung ist müssen ihr Grenzen gesetzt werden, sie kann deshalb nicht zu jeder Zeit Fernsehen. Das System schafft da Abhilfe.

Christina

Die Frau von Bob ist Christina. Oft nach der Arbeit spricht sie mit ihrem Mann über ihren Alltag, dabei läuft der Fernseher und stört die Unterhaltung. Die Fernbedienung ist oft in solchen Momenten nicht aufzufinden. Christina wünscht sich in solchen Fällen das der Fernseher von allein leiser oder aus gehen sollte.

3.2.2 Anwendungsfälle

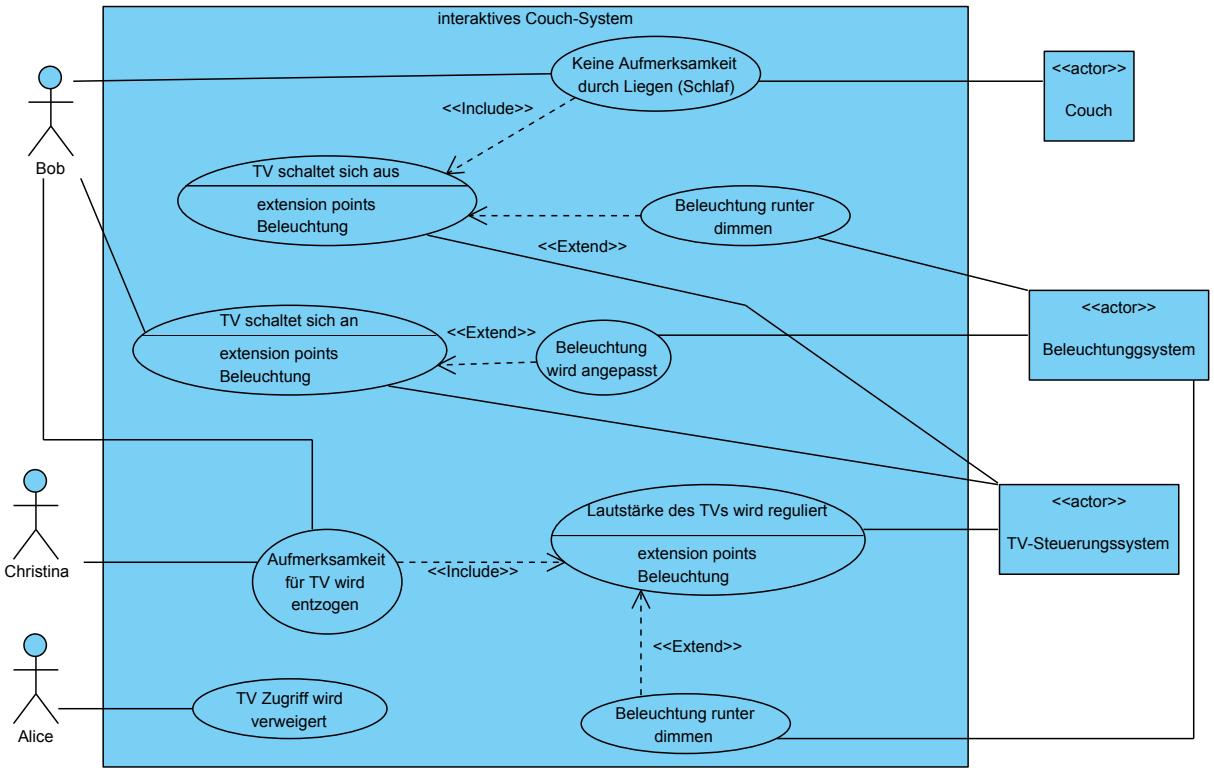

Abbildung 3.1: Use-Case Diagramm

In der Abbildung 3.1 ist das Use-Case Diagramm zu sehen. Wie in der Geschichte über Bob und dem Use-Case Diagramm zu entnehmen ist gibt es drei Charaktere.

Szenarien

Aus dem Use-Case Diagramm 3.1 resultieren folgende Szenarien:

Bob setzt sich auf die Couch

Bob kommt von der Arbeit nach Hause und setzt sich auf die Couch, nach dem das System ihn erkannt hat, schaltet sich der Fernseher automatisch an. Gegebenenfalls werden die Lichtverhältnisse für ein spektakuläres TV-Erlebnis angepasst, in dem die Rollos runter-/hochgefahren werden und/oder die Lichtumgebung reguliert wird.

Bob schenkt dem TV keine Aufmerksamkeit

Wird Bobs Aufmerksamkeit für eine kurze Weile nicht dem TV geschenkt, sei es durch eine Unterhaltung oder Beschäftigung mit dem mobilen Gerät, so wird der die Lautstärke des Fernsehers runter reguliert.

Bob schenkt dem TV keine Aufmerksamkeit (durch Schlaf)

Nach einem langen Arbeitstag entspannt sich Bob und legt sich auf die Couch um Fernzusehen. Nach einer Zeit wird die Aufmerksamkeit Bobs dem TV entzogen (Schlaf). Das System analysiert sein Umfeld und stellt fest, dass er auf der Couch liegt und für lange Zeit keinen Blickkontakt zum Fernseher hat. Der Fernseher schaltet sich automatisch aus und das Licht wird heruntergedimmt.

TV-Verbot für Alice

Alice versucht am nächsten Abend Fernzusehen, jedoch ist es Abends und laut Einstellung ist sie nicht befugt um diese Uhrzeit Fernzusehen. Der TV schaltet sich nicht an und die Beleuchtung blinkt zwei mal stark auf.

Komfort

Ein angestrebtes Ziel dieser Arbeit ist es, durch das entworfene System, dem Menschen einen erhöhten Komfort zu bieten. Um sich damit auseinander zu setzen muss der Begriff Komfort kurz erläutert werden. Laut Duden ist Komfort:

> *„auf technisch ausgereiften Einrichtungen beruhende Bequemlichkeiten, Annehmlichkeiten; einen bestimmten Luxus bietende Ausstattung“* (Duden:Komfort [2013])

In dieser Definition fallen drei Adjektive bzw. Eigenschaften besonders auf: Technisch, Bequemlichkeiten und Luxus. Es wird von vornherein ausgegangen, dass Komfort durch Technik hervorgeholt werden kann. Das in dieser Arbeit realisierte System verwendet qualitativ hohe Technik (4), um Komfort zu ermöglichen. Das Living Place bietet die geforderte Ausstattung sei es ein Sensornetzwerk, professionelle Beleuchtungssysteme oder high-end Rechner für eine hohe Rechenleistung. Neben umgebungsunterstütztes Leben (AAL: Ambient Assisted Living), die oft Ältere Menschen und Menschen mit Behinderung als Bezugsgruppe hat, ist Bequemlichkeit auch ein Ziel, welches im Living Place angestrebt wird und außerdem bietet es eine Plattform für innovative Technik. In dieser Arbeit soll dies auch erstrebt werden. Die gewünschte Technik bietet auch einen Luxus. Bequemlichkeit und Luxus sind oft stark mit einander verknüpft, deshalb kann das zu entwickelte System als eine Luxusgut angesehen werden. In der Arbeit jedoch soll nicht dieser Aspekt betrachtet werden. Eher werden die technischen Aspekte und der Komfort als Ganzes betrachtet.

Warum Komfort?

Der Mensch in der heutigen Gesellschaft klagt oft über zu wenig Zeit. Der Faktor Stress im Alltag, auf der Arbeit oder in sonstigen Situationen des Lebens bestärkt noch dieses Klagen. Um gegen dieses Wehklagen anzugehen, sollen Dinge und Angelegenheiten schnell und korrekt funktionieren. Komfort ist eines der Eigenschaften, mit der man versucht das Leben eines Menschen einfacher bzw. angenehmer zu gestalten.

Komforterhöhung

Die Szenarien in Kapitel 3.2.2 werden im Rahmen der Komforterhöhung erläutert. Es wird zunächst die übliche Situation vorgestellt und anschließend werden die Aspekte, die zur Komforterhöhung beitragen präsentiert. Die komfortableren Szenarien sind in Kapitel 3.2.2 zu finden.

Bob setzt sich auf die Couch

- Übliche Situation: Bob kommt erschöpft von der Arbeit nach Hause. Er macht das Licht an und will sofort Fernsehen, jedoch findet er erst nach einer langen Zeit die Fernbedienung. Das Licht ist für das Fernsehen viel zu grell. Es spiegelt am TV. Bob muss aufstehen und die Rollos runter ziehen.
- Komforterhöhung durch: Automatische Einschaltung des Fernseher bei Kontakt mit der Couch, Anpassung der Lichtverhältnisse für besseres TV-Erlebnis

Bob schenkt dem TV keine Aufmerksamkeit

- Übliche Situation: Bob ist für eine Weile abgelenkt und widmet dem TV keine Aufmerksamkeit. Er fühlt sich jedoch gestört von der Lautstärke des Fernsehers und muss seine momentane Aktivität abbrechen und den TV leiser machen.
- Komforterhöhung durch: Automatische Regulierung der Lautstärke des Fernsehers

Bob schenkt dem TV keine Aufmerksamkeit (durch Schlaf)

- Übliche Situation: Nach einem langen Fernseherabend fällt Bob auf der Couch in Schlaf und schenkt dem TV keine Aufmerksamkeit mehr. Er wird ständig von der Lautstärke des TVs und der Helligkeit der Beleuchtung aufgeweckt. An einigen Abenden schafft er es noch den Fernseher aus zu machen und die Beleuchtung aus zu schalten und ins Bett zu gehen und an anderen Abenden schläft er auf der Couch, ohne Fernseher und Beleuchtung aus zu schalten. Dies verursacht hohe Stromkosten.
- Komforterhöhung durch: Automatische Ausschaltung des Fernsehers, Herunterdimmen der Beleuchtung

TV-Verbot für Alice

- Übliche Situation: Bob kommt spät von der Arbeit nach Hause und sieht wie seine Tochter noch fernsieht. Nach einer Auseinandersetzung mit seiner Tochter, verlässt sie das Wohnzimmer und Bob kann sich dem TV widmen.
- Komforterhöhung durch: Automatische Erkennung der Person, Kindersicherung

3.3 Kontexte

Wie in Context aware computing 3.1 dargestellt, werden Kontexte benutzt um eine Interaktion mit dem Menschen zu realisieren. Die, für dieses System, relevanten Kontexte sollten vordefiniert sein, um die Menge an Informationen einzugrenzen und dem Entwickler die Realisierung eines Context aware computing System zu vereinfachen.

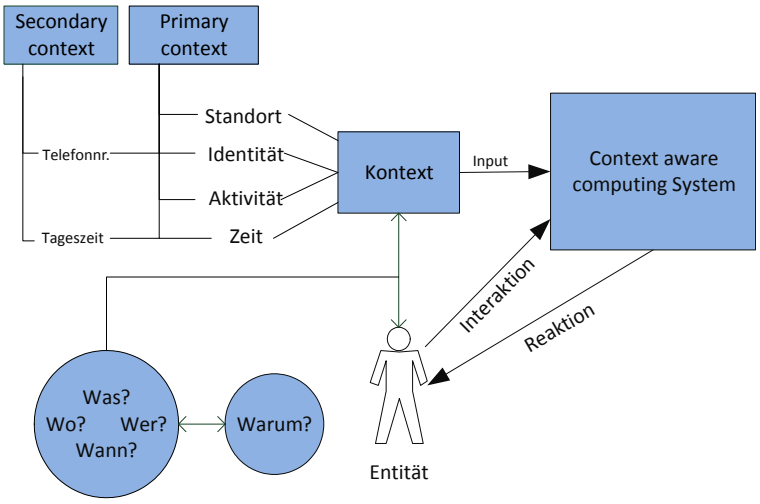

Abbildung 3.2: Kontextumgebung
Quelle: angelehnt an Dey und Abowd [2013]

In Abbildung 3.2 wird aufgezeigt wie ein Kontext aufgebaut ist und wie das Zusammenspiel mit der Entität ist, das Context aware computing System wird mit einbezogen und dient als Schnittstelle zwischen dem Kontext und der Entität. Hierbei ist zu beachten, dass die Entität den Kontext beeinflusst und gleichzeitig vom Kontext beeinflusst werden kann.
Es folgt nun eine Erläuterung und anschließend eine schematische Zuordnung der Kontexte im zu entwickelten System. Dabei werden die Begriffe, die in den Context aware computing 3.1 benutzt wurden, eingesetzt.

3.3.1 Sitz-/Liegeprofile

Um die Position und Situation des Benutzern einzuschätzen, müssen Sitz-/Liegeprofile definiert werden. Die Couch dient hier als Behandlungsobjekt und Kontext. Anhand der Position des Benutzers wird darauf geschlossen, ob er sitzt oder liegt. Mit Hilfe von eingebauten Sensoren in der Couch wird die Position des Benutzers festgestellt.

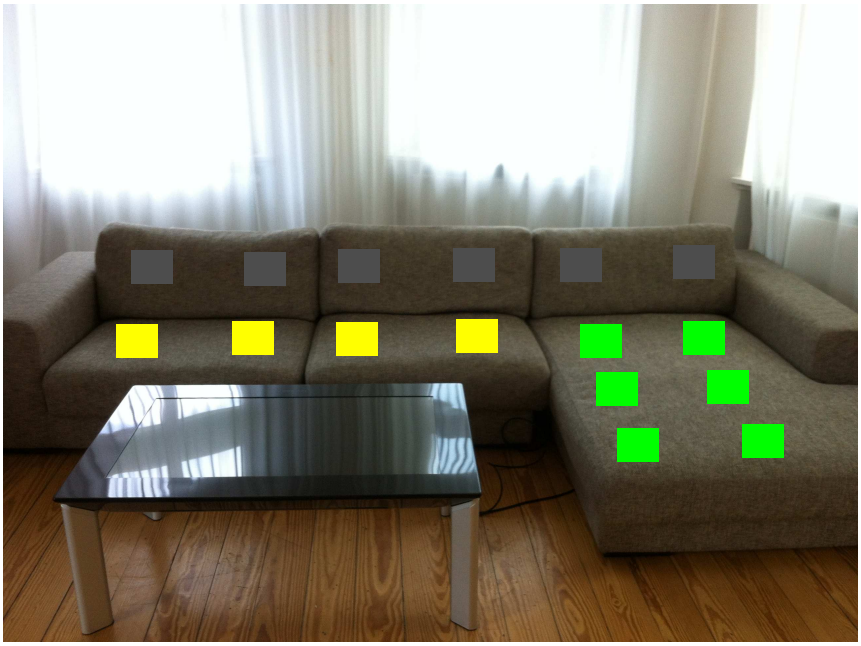

Abbildung 3.3: Couch: Sensoren farbig

Es wird von einem 1,78 m großen und 83,4 kg schwerem Mann ausgegangen, dies sind Durchschnittswerte der in Deutschland lebenden Männer (Bundesamt [2010]). In der Abbildung 3.3 ist die Einteilung der Couch zu sehen. Wenn eine Person sitzt, wird davon ausgegangen, dass sie nicht mehr als zwei gelbe/grüne Sensoren beeinflusst. Werden beispielsweise die Füße auf die Couch gelegt, gilt dies schon als Liegen bzw. eine Position, die dem Liegen nahe kommt, sprich werden mehr als zwei gelbe/grüne Sensoren beeinflusst, so gilt dies als Liegen. Die grauen Sensoren werden für die Beurteilung, ob eine Person liegt oder sitzt, nicht berücksichtigt, da ein Anlehnen im Sitzen und Liegen geschehen kann oder auch nicht, so haben sie für die Analyse keine Relevanz.

Sitz-/Liegeprofile bzw. Couch als Kontext

Der Begriff Kontext wird mit dieser Information (Sitz-/Liegeprofile) und dem dazugehörigen Objekt Couch verglichen.
Entität: eine Person, die Applikation
Sitz-/Liegeprofile Situation: Dieser Kontext kann dem Secondary context „Position auf der Couch“, wobei er vom Primary context Standort indiziert werden kann, zugeordnet werden. Je nach dem in welcher Position der Benutzer ist (Liegen oder Sitzen) kann das System Rückschlüße ziehen. In Kombination mit dem Kontext „Aufmerksamkeit“ kann das System den Fernseher

ausschalten und die Beleuchtung runterdimmen: liegt die Person und ist seine Aufmerksamkeit für eine lange Zeit nicht dem Fernseher gewidmet.
Zustände: Sitzen, Liegen, kein Kontakt mit der Couch

3.3.2 Aufmerksamkeit

Eines der Kontexte ist Aufmerksamkeit. Je nach dem wie der Benutzer auf den Fernseher schaut schaltet sich der Fernseher ab/an oder wird lauter und leiser. Dies wird als Aufmerksamkeit bezeichnet. Es wird nur auf den Blickkontakt geachtet, nicht die Aufmerksamkeit, die mit anderen Sinnen, wie zum Beispiel Ohren, Mund, Nase oder kinetische Äußerungen, hervorgeholt werden können. Mit Hilfe der xuuk eyebox2 - einem Eyetracker, soll dieser Kontext beobachtet werden. So kann präzise ermittelt werden, ob eine Person Richtung TV Blickkontakt hat.

Aufmerksamkeit als Kontext

In welche Kategorie fällt dieser Kontext und wie wird das System darauf reagieren, dies wird hier erklärt.
Entität: eine Person
Aufmerksamkeitssituation: Wird dem Fernseher, vom Benutzer Aufmerksamkeit geschenkt, so bleibt er an. Dies ist die Hauptaufgabe dieses Kontextes. Dieser Kontext wird dem Primary context Aktivität zugeordnet. Die Aufmerksamkeit wird als eine Aktivität angesehen, denn sie wird verursacht durch den Blickkontakt des Benutzers mit der eyebox2. Fehlt für lange Zeit der Blickkontakt so wird der Fernseher leiser. Kombiniert mit dem Kontext „Sitz-/Liegeprofile“ kann der Fernseher ausgeschaltet werden, falls der Benutzer für eine lange Zeit liegt und der Blickkontakt für eine lange Zeit fehlt.
Zustände: Blick auf TV, Blick nicht auf TV

3.3.3 Lichtsituation

Für die Beurteilung der Lichtsituation ist es erforderlich die Lichtumgebung zu messen. Die Messwerte werden anschließend interpretiert und daraus soll eine geeignete Reaktion erfolgen. Die Reaktion kann so aussehen, dass die Beleuchtung runter- oder hochgedimmt wird und/oder die Rollos hoch oder runter gefahren werden. Es ist äußerst wichtig, dass die Lichtsituation beim Fernsehen korrekt abgestimmt ist.

Lichteinfluss auf den Menschen

Um einen besseren Einblick über die Wichtigkeit von Licht zu erhalten, wird gezeigt wie der Mensch vom Licht beeinflusst werden kann. Dabei wird zuerst kurz auf den gesundheitliche Aspekt eingegangen und anschließend Bezug zum Wohlergehen in verschiedenen Lichtsituationen genommen.
Der Mensch ist eine tages-aktive Spezies (Schobersberger u. a. [2007]). Bedeutet, dass er die Hauptaufgaben des Lebens am Tag verrichtet. Gesteuert von seiner inneren Uhr werden Schlaf-Wach-Phasen reguliert. Das Licht spielt hierbei einen großen Faktor, es dient als ein Zeitgeber und synchronisiert die innere Uhr auf den 24-Stunden-Hell-/Dunkel-Rhythmus der Erde ein (Schobersberger u. a. [2007]).

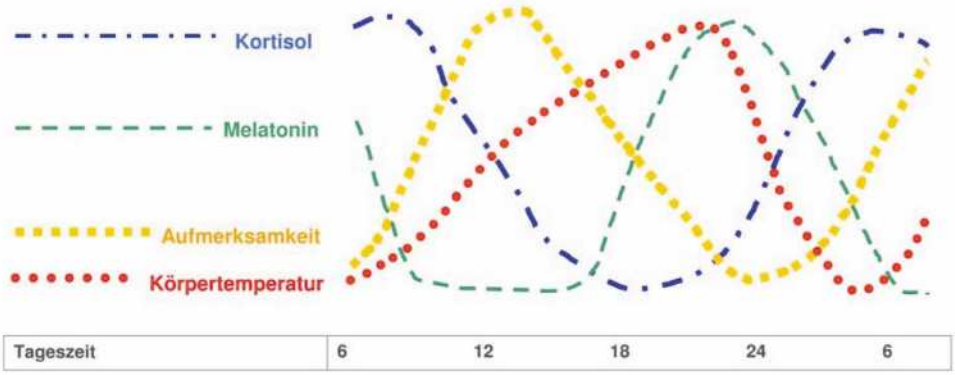

Abbildung 3.4: Zirkadiane Rhythmen
Quelle: Schobersberger u. a. [2007]

In Abbildung 3.4 sieht man, dass die körperlichen Hormone beeinflusst werden und dafür sorgen, dass die Aufmerksamkeit dementsprechend fällt oder sinkt.
Für das Wohlbefinden bzgl. Licht ist darauf zu achten, dass durch das Licht nicht am Bildschirm störende Blendungen auftauchen. Es sollten große Helligkeitunterschiede zwischen Bildschirm und Raumumgebung vermieden werden, da sonst die Augen stark beansprucht werden (Licht:Fernsehen [2013]). Es soll nun festgestellt werden wie viel Lichtstärke angenehm für einen Menschen ist. Normen sind im industriellen und Arbeitsbereich unumgänglich - wie viel Beleuchtungstärke, Arbeitsplätze und Räume im Allgemeinen, für effizientes Erfüllen der jeweiligen Aufgaben, haben müssen wird beispielsweise in der DIN5035 festgehalten (DIN5035 [2013]).
Laut Jürgen Waldorf dem Geschäftsführer von licht.de und vom Fachverband Licht im ZVEI, gibt es keine normativen Werte für Lichtstärke im Wohnbereich (Waldorf [2013]). Jedoch kann

ein Richtwert, also wie viel Lichtstärke für ein Wohnzimmer oder den Wohnbereich geeignet ist, aus DIN5035 (DIN5035 [2013]) abgeleitet werden.

Zweck des Raumes	Beleuchtungstärke E in Lux
Lagerräume für gleiches oder großes Lagergut:	50
Liegeräume und sonstige Pausenräume:	100
Empfangsräume:	100
Lagerräume mit Leseaufgabe:	200
Büroräume mit tageslichtorientierten Arbeitsplätzen:	300
Sitzungs- und Besprechungsräume:	300
Unterrichtsräume allgmein:	300
Sanitätsräume:	500
Räume für EDV und PC-Anwendung:	500

Quelle: DIN5035 [2013]

Tabelle 3.1: Normwerte in Lux in Räumen

In Tabelle 3.3.3 sieht man einige Normwerte in verschiedenen Räumen. In normalen Lagerräumen ist wenig Lux nötig, um die Erfüllung der Aufgaben zu bewerkstelligen. Die Hauptaufgabe dieser Räume ist das Lagern von Gegenständen. Menschen halten sich nicht die meiste Zeit in Lagerräumen auf. Eine Erhöhung der Beleuchtungsstärke ist erst notwendig, wenn eine Leseaufgabe hinzukommt. Sanitätsräume benötigen viel Lux, denn in solchen Räumen werden Operationen bzw. Behandlungen von Patienten durchgeführt. Die Sachlage muss also klar zu erkennen sein. Die Annahme, die in dieser Arbeit getroffen wird ist, dass der Luxwert im Wohnbereich zwischen den Werten, die für Lagerräume und Sanitätsraume nötig sind, liegen muss. Die Annahme ist wie folgt begründet: Es wird mehr Lichtstärke benötigt als in einem Lager, da man sich im Wohnbereich gut zurecht finden muss. Allerdings wird weniger Lichtstärke benötigt als in einem Sanitätsraum, da man keine präzisen Arbeiten im Wohnbereich machen muss. Der Wert liegt also zwischen 300 und 400 Lux.

Lichtsituation als Kontext

Es folgt nun eine Abhandlung darüber Licht als ein Kontext zu betrachten. Gemäß der Definition und den Merkmalen von Kontext wird analysiert was für eine Art Kontext die Lichtsituation ist.
Entität: eine Person, die Applikation
Lichtsituation: Nach den Kategorien von Kontext, kann es dem Secondary context zugeordnet

werden. Es kann von den Primary context der Aktivität und der Zeit indiziert werden. Je nach dem welche Tageszeit gerade vorherrscht, also Morgens, Mittags, Nachmittags, Abends oder Nachts, kann die Lichtsituation bzw. die Beleuchtungstärke in den jeweiligen Fällen variieren. Je nach Aktivität des Benutzer, also schaut er Fern, schaltet er den Fernseher aus oder aber auch liegt er auf der Couch und schenkt dem Fernseher für lange Zeit keine Beachtung, kann die Beleuchtungstärke variieren. Sie kann runter- oder hochgedimmt werden. Die Einheit ist Lumen pro Quadratmeter (Lexikon [2013]).
Zustände: Angenehmes Licht, Licht entspricht nicht den Anforderungen

3.3.4 Aufmerksamkeit und Sitz-/Liegeprofile

Es wird über die Kombination der beiden Kontexte eingegangen. Dabei werden die Situationen anhand von Bildern beschrieben. In Kombination der beiden Kontexte können Aktionen ausgelöst werden, diese werden jetzt betrachtet.

Abbildung 3.5: Sitzen und Aufmerksamkeit

In Abbildung 3.5 hat der Bewohner die Aufmerksamkeit auf den TV gerichtet und sitzt. Durch diese Kombination schaltet sich der TV automatisch ein.

Abbildung 3.6: Liegen und Aufmerksamkeit

In Abbildung 3.6 hat der Bewohner die Aufmerksamkeit auf den TV gerichtet und liegt. Der Fernseher ist dabei eingeschaltet (automatisch).

Abbildung 3.7: Liegen und keine Aufmerksamkeit

In Abbildung 3.7 hat der Bewohner durch Schlaf keine Aufmerksamkeit auf den TV. Der

Fernseher wird automatisch ausgeschaltet (Beleuchtung wird ebenfalls angepasst, nicht zu sehen auf den Bildern).

Abbildung 3.8: Sitzen und keine Aufmerksamkeit

Dem Fernseher wird keine Aufmerksamkeit geschenkt (3.8). Diese Situation kann vorkommen, wenn der Bewohner mit jemanden anders spricht oder andere Dinge tut (beispielsweise mit dem Handy beschäftigt sein). Der TV wird hier leiser, um die gegenwärtige Aktivität nicht zu stören.

3.4 Anforderungen

Nach der Zusammenstellung und Erläuterung der Szenarien erfolgt die Erfassung der Anforderungen. Es werden zunächst die Stakeholder ermittelt und kurz vorgestellt. Anschließend folgen die funktionalen Anforderungen. Sie werden in diesem Teil beschrieben und sollen im weiteren Verlauf der Arbeit umgesetzt werden.

3.4.1 Stakeholder

Die Stakeholder sind Personengruppen, die mit dem System in jeglicher Form in Kontakt stehen. Sie beeinflussen direkt oder indirekt das System. In diesem System gibt es zwei Stakeholder.

- **Benutzer:** Der Benutzer ist jede Person die direkt mit dem System in Kontakt treten kann. Sie beeinflusst das System und wird vom System beeinflusst. Der Benutzer weiß nicht wie das System funktioniert. Er wird nur vom System unterstützt, damit der Benutzer Komfort verspürt.
- **Administrator:** Der Administrator kann ein Benutzer mit Wissen über das System sein. Er hat da System aufgesetzt und kann Einstellungen am System vornehmen. Um Einstellungen am System vorzunehmen nutzt der Administrator eine Weboberfläche.

3.4.2 Funktionale Anforderungen

Die funktionale Anforderungen beschreiben die Funktionen die das System erfüllen soll. Ein- und Ausgabeverhalten werden auch beschrieben. Die Anforderungen sind gemäß ihrer Priorität aufgelistet, dabei steht die höchste Priorität oben. Jede funktionale Anforderung beginnt mit FA (für funktionale Anforderung) und anschließende Nummerierung.

FA1 - TV schaltet sich automatisch ein

- **Akteur:** Benutzer, TV-Steuerungssystem
- **Ereignisse:** 1. Benutzer sitzt oder liegt auf der Couch, 2. Benutzer schaut zum TV, 3. TV geht an
- **Vorbedingungen:** Benutzer wurde bereits erkannt und ist befugt TV zu schauen
- **Nachbedingungen:** TV schaltet sich automatisch ein
- **Zeitverhalten:** Unmittelbar nachdem Benutzer zum TV schaut, soll der Fernseher angehen.

FA2 - TV schaltet sich automatisch aus

- **Akteur:** Benutzer, TV-Steuerungssystem
- **Ereignisse:** 1. Benutzter liegt auf der Couch, 2. Benutzer schaut für eine lange Zeit nicht zum Fernseher, 3. TV geht aus
- **Nachbedingungen:** TV geht automatisch aus
- **Zeitverhalten:** Das Ausschalten des Fernseher kann sofort erfolgen, muss aber nicht.

FA3 - Beleuchtungsanpassung

- **Akteur:** Benutzer, Beleuchtungssystem
- **Ereignisse:** 1a. Lichtverhältnisse ändern sich, 2a. Beleuchtungssystem berechnet neue Daten für korrekte Lichtverhältnisse und schickt sie ans System, 3a. Lichtverhältnisse werden angepasst
  1b. Es wurde erkannt das Benutzer inaktiv ist und auf Couch liegt, 2b. Beleuchtung wird runtergedimmt
- **Vorbedingungen:** a & b: Benutzer befindet sich auf der Couch, b: Benutzer schenkt TV keine Aufmerksamkeit und liegt auf der Couch
- **Nachbedingungen:** Beleuchtung wird geregelt
- **Zeitverhalten:** a: Aktualisierung der Daten vom Beleuchtungssystem sollten in zehn Sekunden Intervallen statt finden, b: Das Runterdimmen der Beleuchtung kann sofort erfolgen, muss aber nicht, falls wichtige Berechnung Prioriät haben.

FA4 - Lautstärken-Anpassung des TVs

- **Akteur:** Benutzer, TV-Steuerungssystem
- **Ereignisse:** 1a. Benutzer schenkt dem TV keine Aufmerksamkeit, 2a. Lautstärke des TV wird runter reguliert, 1b. Benutzer schenkt dem TV Aufmerksamkeit, 2b. Lautstärke des TV wird lauter
- **Vorbedingungen:** a & b: Benutzer befindet sich auf der Couch, b: Benutzer hat dem TV keine Aufmerksamkeit geschenkt
- **Nachbedingungen:** Regulierung der Lautstärke des Fernsehens
- **Zeitverhalten:** Unmittelbar nach der Erkennung der Situation des Benutzers

FA5 - Gesichtserkennung mit Bestimmung zur Befugnis TV zu schauen

- **Akteur:** Benutzer
- **Ereignisse:** 1. Benutzer setzt sich auf die Couch, 2. Benutzer wird vom System erkannt, 3a. Befugnis für Benutzer TV anzumachen, 3b. Keine Befugnis für Benutzer TV anzumachen
- **Zeitverhalten:** Unmittelbar

FA6 - Einstellung wann TV automatisch angehen soll

- **Akteur:** Administrator
- **Ereignisse:** 1. Administrator geht auf die Webinterface-Seite, 2. Wählt Kategorie TV Steuerung 3. Stellt ein „von“ wann „bis“ wann der TV automatisch bei Erkennung des Benutzers an gehen soll, 4. Klickt auf „Bestätigen“
- **Nachbedingungen:** Einstellung wird im System gespeichert
- **Zeitverhalten:** Übernahme bis zur nächsten Nutzung

FA7 - Einstellung automatische Beleuchtungsanpassung

- **Akteur:** Administrator
- **Ereignisse:** 1. Administrator geht auf die Webinterface-Seite, 2. Wählt Kategorie Beleuchtungssteuerung, 3. Stellt ein „von“ wann „bis“ wann die Beleuchtung automatisch reguliert werden soll, 4. Klickt auf „Bestätigen“
- **Nachbedingungen:** Einstellung wird im System gespeichert
- **Zeitverhalten:** Übernahme bis zur nächsten Nutzung

FA8 - Einstellung TV Laustärkenanpassung

- **Akteur:** Administrator
- **Ereignisse:** 1. Administrator geht auf die Webinterface-Seite 2. Wählt Kategorie TV Steuerung, 3a. Setzt Haken für TV Lautstärkenanpassung, 3b. Setzt Haken raus für keine TV Lautstärkenanpassung, 4. Klickt auf „Bestätigen“
- **Nachbedingungen:** Einstellung wird im System gespeichert
- **Zeitverhalten:** Sollte schnell wie möglich übernommen werden, spätestens bis zur nächsten Nutzung

Ergänzend wird kurz auf den Faktor Zeit bzw. Zeitverhalten eingegangen: Das System muss nicht die Anforderungen eines Echtzeitsystems erfüllen, sprich das Einhalten von bestimmten Deadlines, da die Szenarien dies nicht bieten. Es sind eher zeitnahe Reaktionen zu erwarten. Da der Benutzer eine Reaktion auf seine Aktion erwartet, soll diese Reaktion zeitnahe geschehen, sprich es darf vom Benutzer nicht das Gefühl des Wartens entstehen (Dahm [2006]).

3.4.3 Hardwareanforderungen

Ubiquitäre Standalone-Komponente

Um die TV-Steuerung und die Durchführung der Analyse der Lichtsituation, die in der der bisherigen Analyse (3.4.2) aufgezeigt wurde, zu realisieren wird eine zusätzliche Komponente benötigt wird. Es werden folgende Anforderungen an die Komponente gestellt:

- Die TV-Steuerung und die Analyse der Lichtsituation soll mit einer Hardwarekomponente durchgeführt werden, es wird somit gewährleistet, dass eine Hardware für mehrere Aufgaben zuständig sein kann.
- Die Hardware sollte klein sein und unaufällig. Es sollte dem Prinzip des Ubiquitous Computing folgen, also ubiquitär sein. Dies ist notwendig, da die Hardware direkt in der Nähe des Fernsehers angebracht wird und der Benutzer sollte so gut wie gar nicht davon abgelenkt werden.
- Die Kosten der Hardware sollten gering wie möglich sein.
- Für zukünftige Arbeiten und Projekte sollte die Hardware leicht erweiterbar sein, sprich es sollten Schnittstellen zur Verfügung stehen, um weitere zusätzliche Komponenten einfach anzubinden.
- Ein weiterer wichtiger Punkt ist, dass die Einarbeitung bzgl. der Hardware keine große Hürde stellen darf, die allgemeine Entwicklung sollte nicht gebremst werden. Dies ist besonders wichtig, wenn neue Entwickler sich mit der Hardware beschäftigen und neue Funktionalität hervorbringen wollen.

3.5 Vergleichbare Projekte

CAMUS - Context-Aware TV Task

Im Rahman von Ubiquitous Computing und Context awareness computing ist das Projekt CAMUS entstanden. Es stammt aus der koreanischen Universität Soongsil University. CAMUS steht für Context-aware Middleware for Ubiquitous Computing Systems und ist eine Middleware. Diese Middleware erleichtert es kontextabhängige Systeme zu entwickeln. Es soll hier nicht die Middleware betrachtet werden, sondern das Projekt Context-Aware TV Task. Dieses Projekt wurde im Rahmen einer Evaluation von CAMUS realisiert.

Projektbeschreibung

Context-Aware TV Task ist ein Projekt welches in Abhängigkeit von der Position eines Benutzern den Fernseher mit bevorzugtem Kanal einschaltet oder ausschaltet. Eine Sprachsteuerung ist auch implementiert. Der Benutzer kann so den TV durch Spracheingaben ein- und ausschalten.

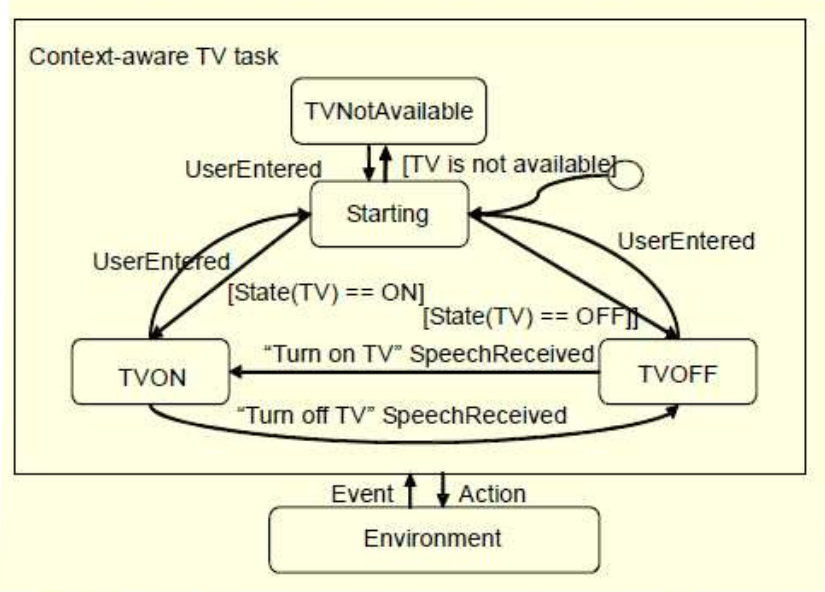

Abbildung 3.9: Zustandsautomat von Context-Aware TV Task
Quelle: Moon u. a. [2007]

In Abbildung 3.9 sieht man den Zustandsautomaten des Systems. „Environment“ wird als ein Kontext betrachtet. Daraus erfolgt ein Event und löst eine Aktion aus.
Für die Erkennung der Person, also ob sie ein Raum betritt oder verlässt werden RFID[1] Tags benutzt. Zusätzlich werden noch Kameras eingesetzt.

[1] radio-frequency identification: Technik um Gegenstände und Lebewesen automatisch zu identifizieren und lokalisieren.

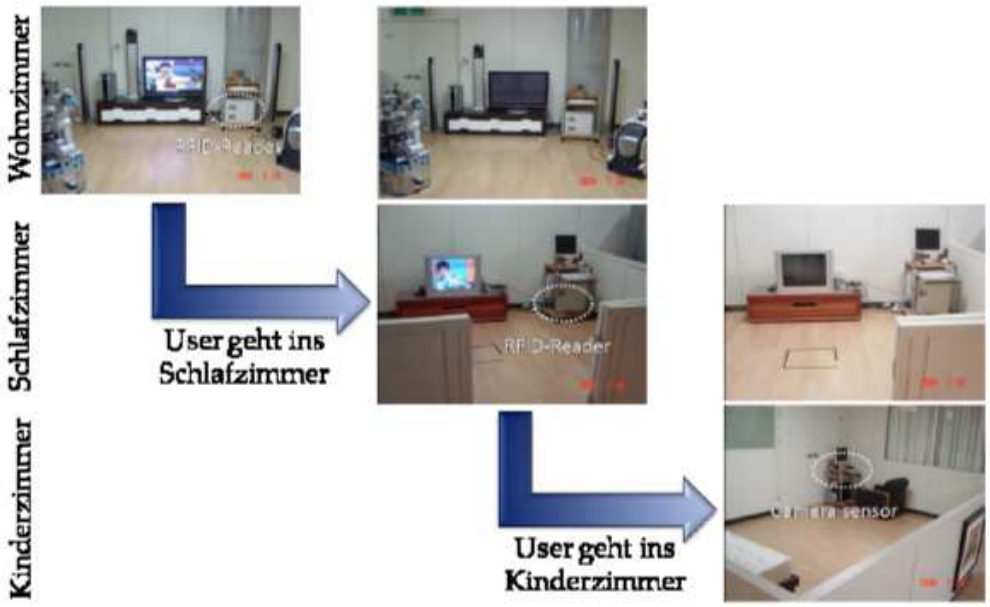

Abbildung 3.10: Context-Aware TV Task: das Experiment
Quelle: Urich [2008]

In Abbildung 3.10 ist ein exemplarischer Ablauf zu sehen. Der Benutzer geht von Wohnzimmer zum Schlafzimmer. Der Fernseher wird im Wohnzimmer automatisch ausgeschaltet und im Schlafzimmer eingeschaltet. Das selbe Ereignis erfolgt bei Betretung des Kinderzimmers, dort gibt es jedoch keinen Fernseher, es wird durch eine Kamera erkannt, dass eine Person das Kinderzimmer betreten hat. Aus dieser Information können andere hier nicht auftauchende Szenarien gebildet werden.

Vergleich zu dieser Arbeit

In Context-Aware TV Task wurde, wie in dieser Arbeit, ein System entwickelt, welches Context aware computing realisiert. Es wurde zusätzlich in der anderen Arbeit (Moon u. a. [2007]) die RFID-Technologie eingesetzt, um zu erkennen, ob eine Person den Raum verlässt. In dieser Arbeit werden auch Sensoren benutzt, die sich jedoch in der Couch befinden. Es wird der gleiche Kontext anders identifiziert und betrachtet.
In diesem System wird die Aufmerksamkeit und die Lichtsituation als zusätzliche Kontexte betrachtet, dies fehlt in dem mit CAMUS entwickelten System. Um diese Kontexte zu behandeln wird für die Aufmerksamkeit ein Eyetracker und für die Lichtsituation ein Lichtsensor ver-

wendet, zwei zusätzliche Sensoren, die es in der anderen Arbeit nicht gibt. Zusammenfassend ist zu sagen, dass in beiden Arbeiten Sensoren benutzt werden, jedoch spielt in dieser Arbeit neben Ubiquitous Computing und Context aware computing, der Komfort auch eine Rolle.

4 Laborumgebung

In diesem Kapitel soll die Laborumgebung, in dem das System realisiert, wird vorgestellt werden. Zunächst wird das „Living Place“ an der HAW Hamburg vorgestellt. Um das System zu realisieren müssen die im Kapitel 3.4.2 formulierten funktionalen Anforderungen mittels Hardware und Software implementiert werden. Diese sollen hier präsentiert werden.

4.1 Living Place Hamburg

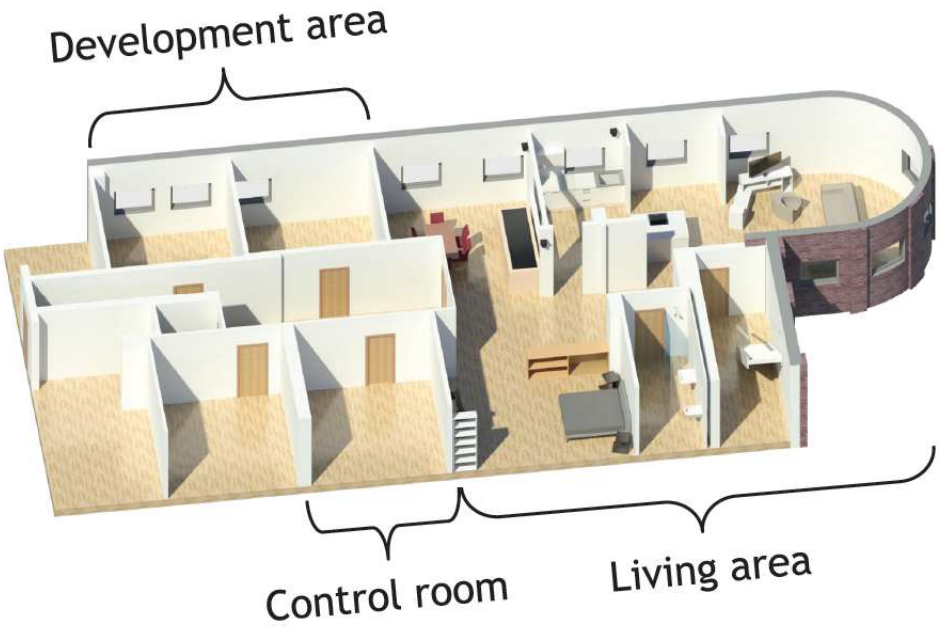

Abbildung 4.1: Living Place Hamburg

Das „Living Place“ ist ein Forschungsprojekt der HAW Hamburg, dass seit 2009 Fragen rund um Context awareness, Ubiquitous Computing und Ambient Intelligence analysiert. Das „Living Place“ ist eine bewohnbare Wohnung in Form eines Lofts, hat eine Fläche von 140m^2 und ist eine Realisierung eines Smart Homes.

Abbildung 4.2: Living Place Hamburg - Schlafzimmer, Wohnzimmer, Küche

Abbildung 4.3: Living Place Hamburg - TV, Gesichtserkennungskamera und xuuk eyebox2

Durch das „Living Place“ ist es möglich realitätsnahe Tests bzw. Szenarien abzubilden. Diese können dann sofort in einem richtigen Rahmen geprüft werden. Wie in Abbildung 4.2 dargestellt, ist das „Living Place“ in den klassischen Räume (Wohnzimmer, Schlafzimmer, Küche) eingeteilt wurden. Dies ermöglicht eine exemplarische Gestaltung der Szenarien. Einer der wichtigsten Ziele des Projekts „Living Place“ ist es Lösungen zu finden, die den Menschen

beim alltäglichen Leben unterstützen. Man findet die Ziele des „Living Place“ in einer Kurzbeschreibung (von Luck u. a. [2010]).
Laut Weiser sollen die Computer in den Hintergrund treten (Weiser [1991]). Der Benutzer soll nicht mitbekommen, dass diese Computer ihn helfen sein Leben einfacher zu gestalten. Im „Living Place“ wird diese Idee angestrebt. Also die Idee des Ubiquitous Computing - Systeme die allgegenwärtig, aber vom Benutzer unbemerkbar sind. Ein weiterer Aspekt des „Living Place“ ist es eine barrierefreie, nahtlose Interaktion des Bewohners mit der Wohnung zu schaffen, dadurch müssen ebenfalls die Grundsätze des Ubiquitous Computing umgesetzt werden. Um dies zu erreichen sind Sensoren, Kameras und Mikrofone in jedem Raum im „Living Place“ angebracht. So kann eine Instanz Informationen vom Benutzer erhalten. Diese Informationen werden in den jeweiligen entwickelten Applikationen verwertet um Systeme zu schaffen, die den obigen Anforderungen genügen.

4.2 Hardware

Die Hardware soll hier vorgestellt werden. Es wird auf die im Kapitel 3.3 genannten Kontexte eingegangen. Die Hardwareanforderungen werden analysiert und mit der vorhanden Hardware verglichen.

4.2.1 xuuk eyebox2

Abbildung 4.4: xuuk - eyebox2
Quelle: xuuk [2013]

In Abbildung 4.4 sieht man den Eyetracker eyebox2 von xuuk (xuuk [2013]). Ein Eyetracker ist eine Hardwarekomponente mit der man die Augen und auch oft Gesichter eines oder mehrerer Menschen erfassen kann. So kann ein Eyetracker Daten liefern, wie beispielsweise ob eine Person in Richtung der Kamera guckt oder welche Person in Richtung der Kamera guckt oder auch gerade wegguckt.

Abbildung 4.5: Links: Kamera, Rechts: Illuminator
Quelle: xuuk [2013]

Die eyebox2 besteht aus zwei Einheiten einen Illuminator und einer Kamera. In Abbildung 4.5 sieht man Links die Kamera und Rechts den Illuminator. An der Kamera- und Illuminatoreinheit befinden sich LEDs, die Infrarot ausstrahlen. Das Infrarot wird benötigt um die Augen einer Person zu erkennen. Die Pupillen werden dadurch in der Kamera hell verfärbt angezeigt. So kann eine Markierung statt finden, die bei der Bildverarbeitung nützlich sein kann, um die Pupillen leichter und genauer zu erkennen. Die eyebox2 kann in Abhängigkeit von der Kulisse (wie viel Licht/Infrarotlicht im Raum ist) zehn Personen gleichzeitig, die sich in einem Abstand (xuuk [2007])zwischen 3.05 m und 9.13 m befinden, erkennen. Laut Manual (xuuk [2007]) beträgt die Genauigkeit bei Gesichtserkennung 95 % oder besser und bei Blickerkennung 80 % oder besser. Dies kann bei verschiedenen Lichtverhältnissen variieren.
Mit der eyebox2 soll der Kontext Aufmerksamkeit behandelt werden. Durch die hohe Genauigkeit der eyebox2 und das Erkennen des Blickkontaktes einer Personen bietet die eyebox2 die passende Voraussetzung um diesen Kontext zu bestimmen und zu identifizieren. So ist es auch Möglich mit der eyebox2 falls eine Kontextveränderung, wie Weggucken der Person, stattfinden zu bemerken und aufzuzeichnen.

4.2.2 Couch

Abbildung 4.6: Couch im „Living place“ im Wohnbereich

In der Abbildung 4.6 ist die Couch im Wohnbereich zu sehen. Sie wurde im Rahmen eines

Projektes des „Living Place“ entworfen. In den Kissen und Sitzkissen befinden sich kapazitive Sensoren, die dafür sorgen, dass bei Berühung der Kissen ein Signal bzw. eine Nachricht an einem Message Broker (mehr dazu in Software 4.3) gesendet wird und man diesen auslesen kann.
Kapazitive Sensoren: Es muss zunächst das „kapazitive“ im Wort erklärt werden. Es kommt von der physikalischen Größe Kapazität. Wenn zwei Leiterplatten gegeneinander gestellt werden und diese Platten entgegengesetzt elektrisch (also eine Platte ist positiv und die andere negativ) geladen und isoliert sind, spricht man von Kondensatoren. Zwischen diesen Platten herrscht eine Spannung U. Die Kapazität C sagt nun aus, wie viel Ladungsmenge Q, bei einer Spannung U auf den Platten gespeichert werden kann. (Hering [2012])
Die Kapazität kann durch verschiedene Faktoren verändert werden. Es besteht eine Abhängigkeit der Kapazität zum Plattenabstand. Beim Verändern des Plattenabstand wird man eine Veränderung der Kapazität beobachten können. Dies kann nun genutzt werden, um zum Beispiel bei einem bestimmten Schwellwert auszusagen, dass jemand den Sensor berührt hat. (Nehrig [2003])

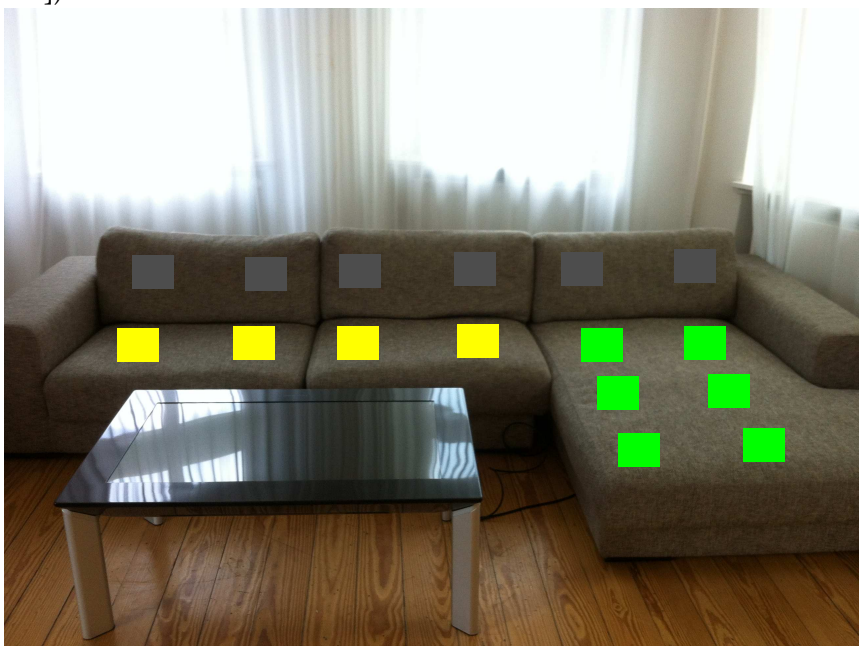

Abbildung 4.7: Couch: Sensoren farbig

Mit der Couch soll er Kontext Sitz-/Liegeprofile behandelt werden. In Abbildung 4.7 sieht man die Couch und die ungefähren Positionen der kapazitiven Sensoren. Die Farben sollen in dieser Erklärung keine Rolle spielen. Es soll der Sensoren die Lage der Person erfasst werden.

Die Couch bieten hier die nötige Ausstattung. Eine präzise Aussage kann durch die Couch nicht getätigt werden, da die Fläche der Kissen nicht komplett durch die Sensoren erfasst werden. Es wird pro Sitzkissen, abgesehen von der Liegefläche, nur zwei Sensoren benutzt. Es können deshalb nur grobe Aussagen getätigt werden, wie Liegen, Sitzen, aber nicht wie eine Person liegt oder sitzt. Für diese Arbeit reichen jedoch diese Aussagen und deshalb ist eine Nachrüstung von Sensoren nicht nötig.

4.2.3 Lichtsensor TSL2561 - Licht zu Digital Konverter

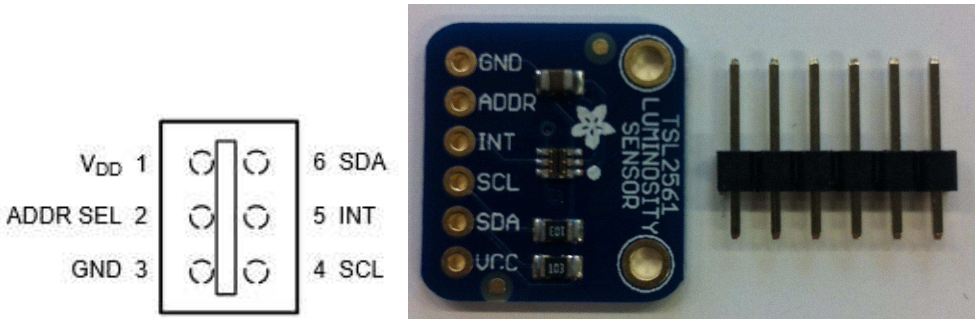

Abbildung 4.8: Lichtsensor: TSL2561 Licht zu Digital Konverter
Quelle: Texas Advanced Optoelectronic Solutions Inc. [2005]

In Abbildung 4.8 ist der Licht zu Digital Konverter TSL2561 abgebildet. Der TSL2561 ist ein 16-Bit digitaler Lichtsensor. Er konvertiert die Bestrahlungsstärke zu einem digitalen Ausgang. Die Daten werden über den Datenbus I^2C übertragen. Der Sensor besitzt zwei Photodioden, eine, die das Gesamtspektrum an Licht (unsichtbares und Infrarotlicht) und eine, die nur Infrarotlicht aufnehmen kann. Über zwei integrierte Analog-Licht-Wandler werden die Datenströme der Photodioden zu einem digitalen Ausgang konvertiert. Die Werte, die am Ausgang anliegen, können dann über empirische Gleichungen, zu Lux (Beleuchtungsstärke) umgerechnet werden.

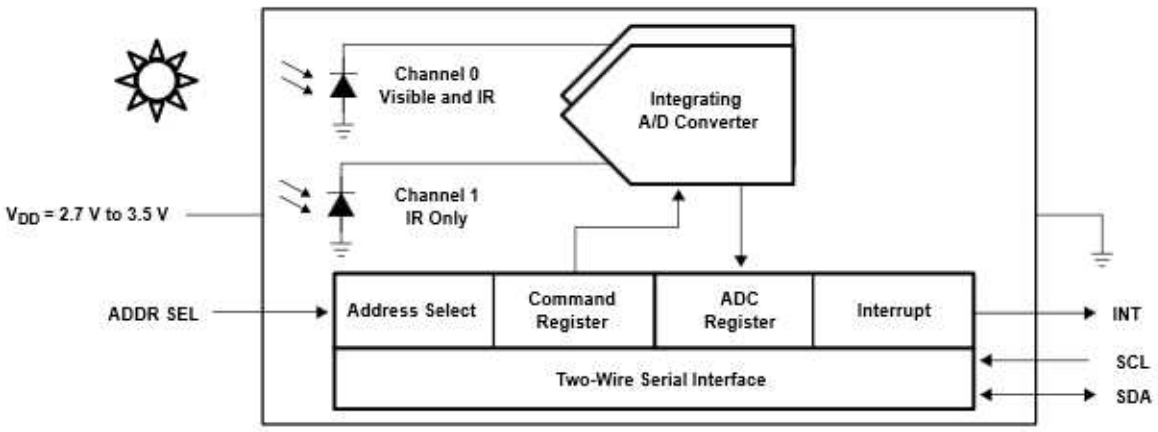

Abbildung 4.9: TSL2561 Licht zu Digital Konverter - Blockdiagramm
Quelle: Texas Advanced Optoelectronic Solutions Inc. [2005]

Um die Arbeitsweise des Sensors zu verstehen kann die Abbildung 4.9 betrachtet werden. Das Blockdiagramm des Lichtsensors enthält alle Eingänge und Ausgänge. Channel 0 und Channel 1 beschreiben die beiden Photodioden. Die Werte werden wie oben beschrieben, zu zwei A/D Konverter gegeben und umgewandelt, die Werte werden in einem ADC Register abgelegt. SCL ist die Taktleitung und über SDA werden die Kommandos vom Mikrocontroller an den Sensor gesendet. Außerdem dient die SDA als Datenleitung für die Übertragung der digitalen Werte vom Lichtsensor an den Mikrocontroller. Mit ADDR können verschiedene Sensoren angesprochen werden. Die INT-Leitung kann eine Interrupt-Funktionalität zur Verfügung stellen und kann nach einer vordefinierten Zeit oder wenn ein vordefinierter Luxwert über-/unterschritten wird aktiv werden.
Dieser Sensor wird verwendet um den Kontext Lichtsituation zu analysieren. Es werden die Werte, die der Sensor liefert gelesen und anhand von Schwellwerten wird entschieden, ob die Lichtsituation ideal für die entsprechende Situation ist. Die Schwellwerte werden in der Einheit Lux (Beleuchtungsstärke) sein.

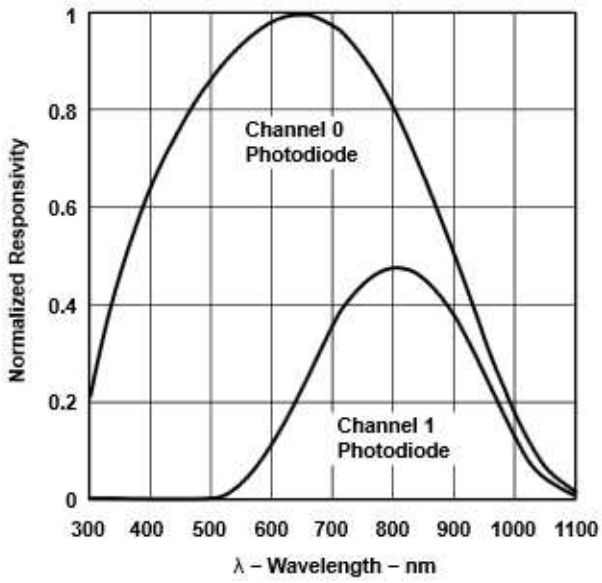

Abbildung 4.10: TSL2561 Licht zu Digital Konverter - Diagramm
Quelle: Texas Advanced Optoelectronic Solutions Inc. [2005]

Der obige Graph 4.10 zeigt die zwei Channels, die für die verschiedenen Spektren verantwortlich sind. Channel 0 kann das gesamte Spektrum aufnehmen und Channel 1 nur den Infrarotbereich.

4.3 Software

Es werden in diesem Kapitel die Software, die bei der Entwicklung und Entstehung des Systems eingesetzt werden, vorgestellt. Anforderungen, falls nötig, werden hier auch ausformuliert. Entwicklungsumgebungen und sonstige Software, die für die Erstellung des Systems gebraucht wurden sind, werden nicht vorgestellt.

4.3.1 JSON

Für die Kommunikation zwischen den einzelnen Applikationen bzw. Systemen im „Living Place“ wird das Nachrichtenformat JSON als Standard benutzt.
JSON ist ein leichtgewichtiges Datenformat, welches den Datenaustausch zwischen Anwendung in einer einfachen Art und Weise ermöglicht. Es ist für Menschen und Maschinen leicht zu verstehen und zu generieren. JSON ist Programmiersprachen unabhängig. JSON basiert auf einer Untermenge der JavaScript Programmiersprache. Es werden in JSON zwei grundlegende

Strukturen benutzt: Name/Wert Paare und eine geordnete Liste von Werten. Es werden grundlegende Datenstrukturen benutzt, die von jeder modernen Programmiersprache auf die ein oder andere Art unterstützt werden (JSON1 [2013], Crockford [2006]).
Folgende strukturierte Typen gibt es:

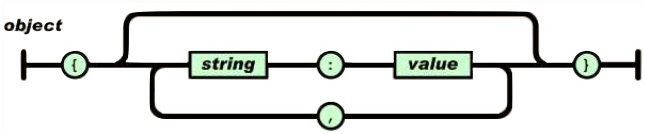

Abbildung 4.11: JSON - object
Quelle: JSON1 [2013]

Es hat eine Zeichenkette und einen zugehörigen Wert.

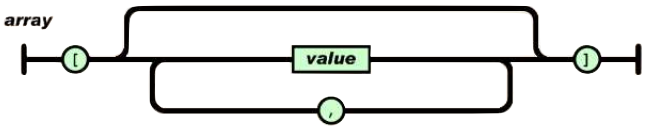

Abbildung 4.12: JSON - array
Quelle: JSON1 [2013]

Das Array kann wie gewohnt mehrere Werte haben.

Folgende primitive Typen gibt es:

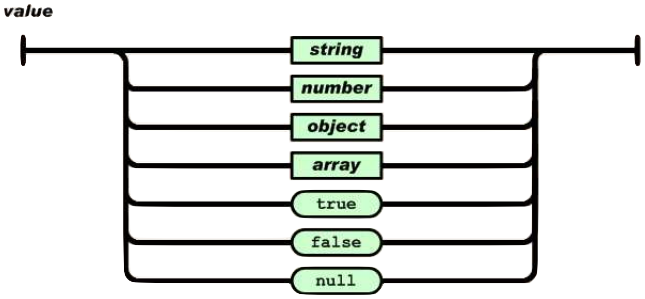

Abbildung 4.13: JSON - value
Quelle: JSON1 [2013]

Ein Wert kann eine Zeichenkette, Zahl, Objekt, Array oder einer der Ausdrücke true (wahr), false (falsch) oder null sein.

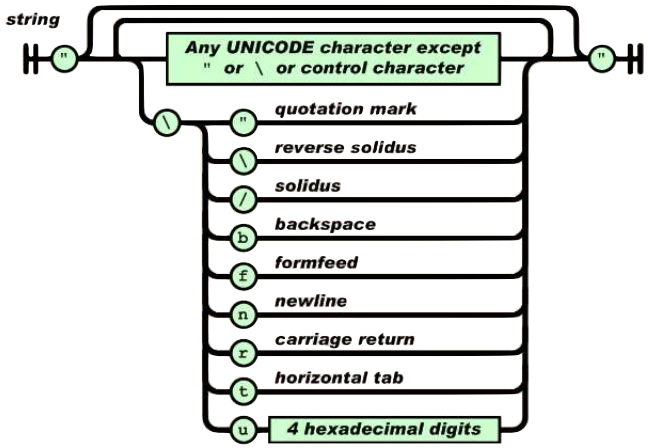

Abbildung 4.14: JSON - string
Quelle: JSON1 [2013]

Ein string ist eine Zeichenkette, die ähnlich aufgebaut ist wie in C oder Java.

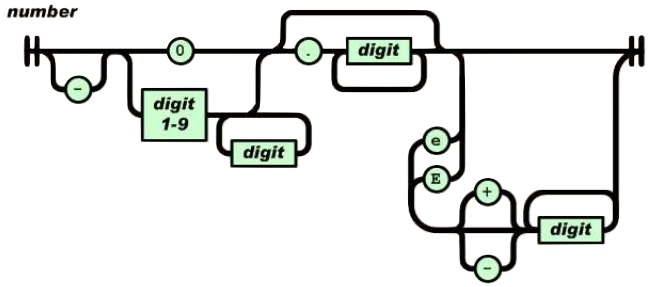

Abbildung 4.15: JSON - number
Quelle: JSON1 [2013]

Eine Zahl ist ähnlich aufgebaut wie in C oder Java, es fehlen allerdings oktale und hexadezimale Zahlen.

4.3.2 ActiveMQ

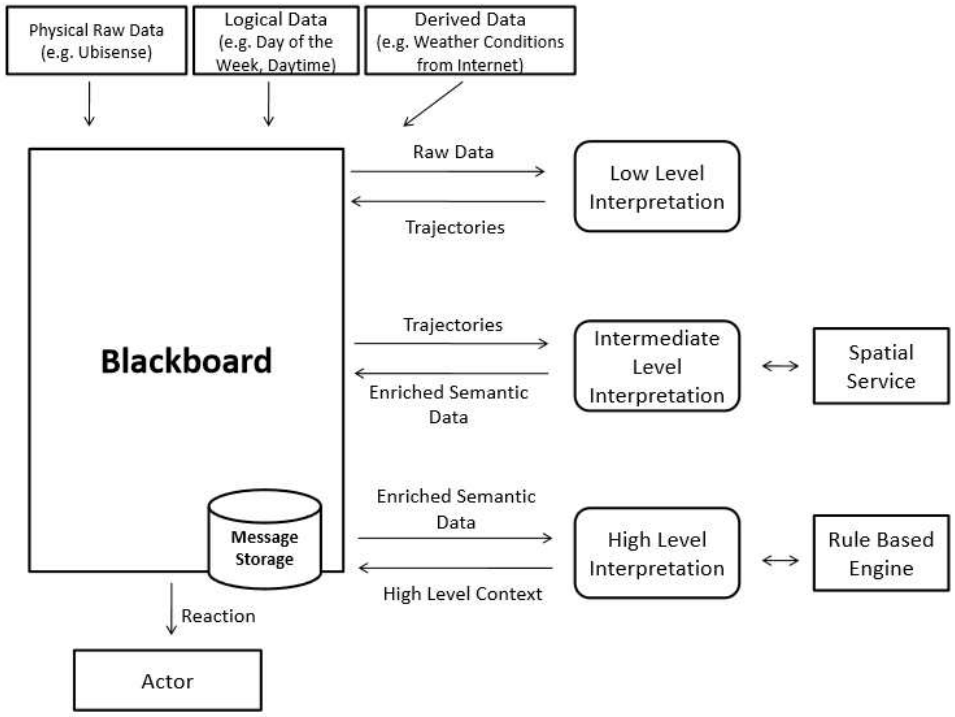

Abbildung 4.16: ActiveMQ-basierte Architektur des Living Place
Quelle: Ellenberg u. a. [2011]

ActiveMQ ist eine Open-Source Nachrichten-orientierte Middleware bzw. message-oriented middleware (MOM) von Apache Software Foundation. Es sorgt dafür, dass Nachrichten mit einer hohen Lieferbarkeit, Performanz, Skalierbarkeit, Vertrauenswürdigkeit (man weiß von wem man die Nachricht erhalten hat und die Person kann dies auch bestätigten) und Sicherheit ausgeliefert werden. Dadurch dass es Open-Source ist und die Apache Lizenz genutzt wird, kann ActiveMQ ohne rechtliche Auswirkungen benutzt und verändert werden. Eine MOM soll Events und Nachrichten durch ein verteiltes System vermittelt und dafür garantieren, dass sie ihr angestrebtes Ziel erreicht. ActiveMQ liefert diese Funktionalität, da es auch wie bereits erwähnt viele Sicherheitsfaktoren und die oben genannten Eigenschaft erfüllt, welche für eine MOM notwendig sind. Die Plattformunabhängigkeit wird von ActiveMQ unterstützt, es können verschiedene Nachrichten-orientierte Applikationen entwickelt mit verschiedenen Programmiersprachen, mit einander kommunizieren.
Es gibt Topics und Queues. Queues werden von sogenannten Producer und Consumer verwen-

det. Ein Producer fügt eine Nachricht in die Queue und ein Consumer holt sich diese Nachricht ab und liest sie. Wurde die Nachricht gelesen, steht sie nicht erneuert (für andere Consumer) zur Verfügung, da sie aus der Queue gelöscht wurde.
Topics werden von sogenannten Publisher und Subscriber verwendet. Publisher veröffentlicht eine Nachricht in einem Topic und Subscriber, die sich für diesen Topic angemeldet haben, erhalten die Nachricht. Anders als bei der Queue, können mehrere Subscriber die gleiche Nachricht erhalten (Snyder u. a. [2009], Foundation [2013a]).
In Abbildung 4.16 ist die Architektur des „Living Place“ in abstrakter Darstellung abgebildet, hierbei wird ActiveMQ als Mittelpunkt gesetzt. Aktionen werden durch Nachrichten die bei ActiveMQ eintreffen ausgelöst. Nachrichten können von verschiedenen Systemen geschickt werden. Die Reaktion erfolgt (oft) beim Benutzer. Dadurch, dass man sich im „Living Place“ über das Standard Nachrichtenformat (JSON) geeignet hat, sind Interaktionen zwischen verschiedenen Systemen möglich. Es muss nur vorab geklärt bzw. definiert werden welche Nachrichten, welche Bedeutungen haben. Somit ist eine Erweiterbarkeit des Systems unterstützbar, da jede Nachricht, die das System beeinflussen kann durch das ActiveMQ gesendet und empfangen wird. Dies ist, neben der Performance und Stabilität, einer der wichtigsten Aspekte für die Benutzung von ActiveMQ und trägt somit eine entscheidende Rolle für die Wahl von ActiveMQ in diesem System.

4.3.3 MongoDB

MongoDB (vom englischen „hu**mongo**us“ , was gigantisch bedeutet) ist eine in C++ geschriebene Open-Source NoSQL Datenbank. Sie ist skalierbar, hochperformant und dokumentenorientiert. Dies bedeutet, dass sie die Daten nicht in Zeilen und Spalten speichert, sondern in einer binären Form von JSON ähnlichen Dokumenten. So kann man einfach und verständlich auf die Datenbank zugreifen. MongoDB kann im Gegensatz zu relationalen Datenbanken, die flache und starre Schemas durch die ganze Tabellen haben, seine Schemas in den Dokumenten, abhängig wie sich die Applikation verändert, variieren. Es ist nicht unbedingt nötig Zeilen und Spalten zu definieren, dies macht die Handhabung von dynamischen Nachrichten schwer, da nicht immer von Anfang an klar ist was für ein Format die Nachrichten haben (10gen [2013]). Diesen Vorteil und andere wichtige Aspekte, wie die Performance, waren entscheidend für die Wahl dieses persistenten Elements für das „Living Place“ (Otto und Voskuhl [2011]).

4.3.4 Couch Webseite - ASP.NET SignalR

ASP.NET SignalR ist eine Bibliothek von ASP.NET. Die Bibliothek vereinfacht erheblich das Hinzufügen von Real-Time-Webfunktionalitäten in Anwendungen. Dies bedeutet, dass serverseitige Inhalte in Echtzeit an die verbundenen Clients gesendet werden. Für Anwendungen, die vom Server oft, also in einer hohen Taktrate Updates benötigen, ist SignalR eine gute Wahl (ASP.NET [2013]).
SignalR wird verwendet um die Sensordaten der Couch zu erhalten. Dieses Projekt ist außerhalb dieser Arbeit entstanden. Die Daten werden für eine Webanwendung benutzt. Da sich die Daten ständig ändern und dies in einem hohen Tempo, hat man sich für SignalR entschieden.

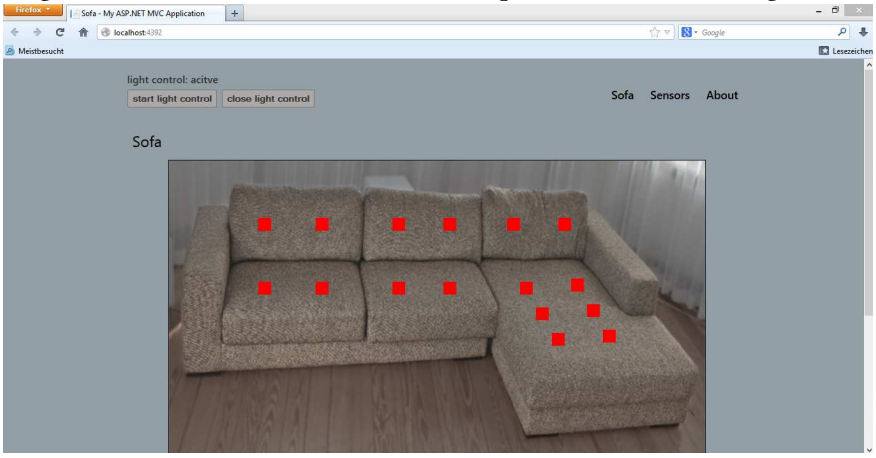

Abbildung 4.17: Couch - Webanwendung Startseite

In Abbildung 4.17 sieht man die Webanwendung, man sieht in Echtzeit die Informationen der Sensoren in Form von gefärbten Kästen. Die Bedeutung der Kästen erfolgt nach einigen Worten über den Aufbau der Webanwendung.
Oben links, gibt es zwei Buttons mit denen in Abhängigkeit wo ein Sensor anschlägt (also wo sich eine Person befindet), das Licht an der jeweiligen Position über der Couch eingeschaltet wird. Diese Funktionalität kann man aus- oder einschalten. Für dieses System ist diese Funktionalität nicht brauchbar, da in dieser Arbeit mehrere Kontexte betrachtet werden und nicht nur die Position der Person.

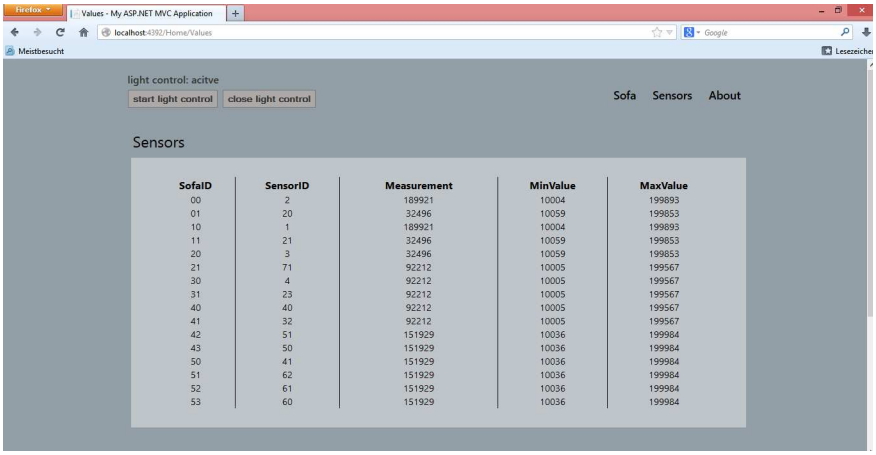

Abbildung 4.18: Couch - Webanwendung Sensordaten

In Abbildung 4.18 findet man die Sensordaten. Man gelangt zu diesem Punkt wenn man auf „Sensors“ klickt. Diese Tabelle beschreibt die internen SensorIDs und die, für die Webanwendung zugewiesenen, SofaIDs. Measurement beschreibt welcher Wert zurzeit gemessen wird. Die Werte können theoretisch zwischen MinValue und MaxValue liegen. Die Zahlen zeigen einen Wert, der die Kapazität des jeweiligen Sensors ausdrückt.

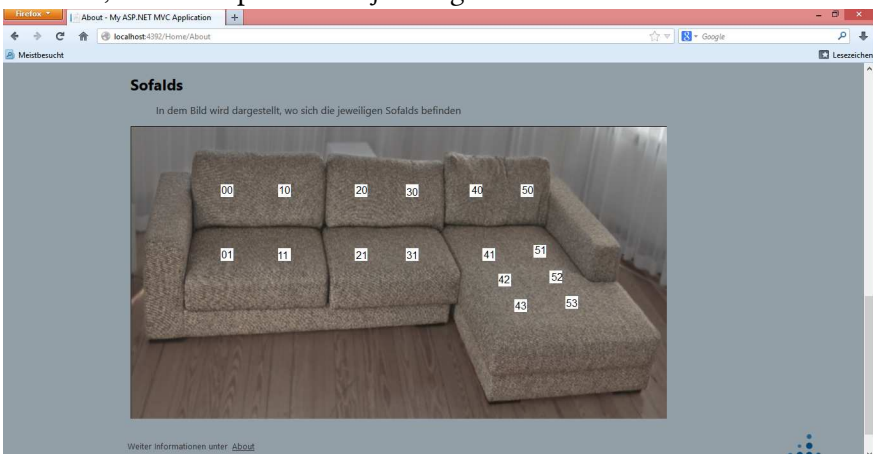

Abbildung 4.19: Couch - Webanwendung About

In der Abbildung 4.19 sieht man die Zuweisung der SofaIDs für die richtige Interpretation der Position eines Menschen. Diese Informationen erhält man wenn man auf den Punkt About klickt. Wie bereits in Abbildung 4.17 zu sehen ist, können die Sensoren verschiedene Farben annehmen. Diese Informationen bedeuten folgendes (entnommen aus der Webanwendung):

- Weiß: Der vorgesehene Sensor ist nicht gefunden wurden.

- Gelb: Sensor wird initialisiert.
- Grün: Sensor befindet sich im Normalbereich.
- Rot: Veränderung des kapazitiven Feldes - eine Detektion hat stattgefunden.
- Blau: Sensor ist im Fehlerzustand: Software beenden, Stromzufuhr ausschalten und System neu starten.

Laut dem About haben die gemessenen Werte folgende Bedeutung:

- Initialisierung: Die Sensoren laufen im unbelasteten Zustand im Wertebereich von 190.000 +- 5%.
- Erkennung: Wenn ein Objekt erkannt wird, stört dieser das kapazitive Feld, daher sinkt der Messwert bis zu 100.000.
- Fehlerwert: Wenn ein Sensor den Messwert 11 liefert, ist dieser Im Fehlerzustand.

5 Design

In diesem Kapitel soll das Design, des zu realisierenden System beschrieben werden. Dabei wird auf die Architektur eingegangen. Es werden verschiedene Entwurfsmuster, sogenannte Pattern, die in diesem System eingesetzt werden beschrieben. Abläufe werden anhand von Zustandsautomaten dargestellt. Das Design dient als Konzept für die spätere Realisierung des Systems, die ebenfalls im Laufe dieser Arbeit beschrieben wird.

5.1 Architektur

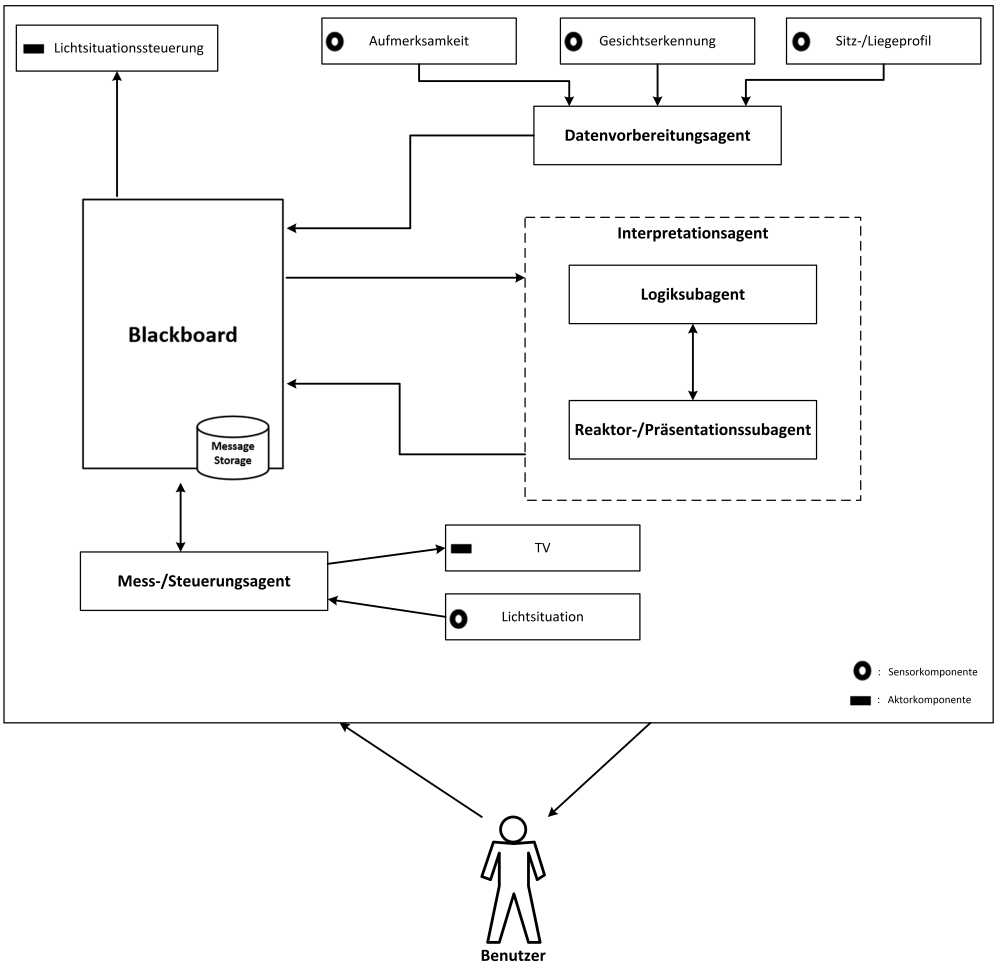

Abbildung 5.1: Architektur des Systems

In der Abbildung 5.1 sieht man die grobe Architektur des Systems. Das System ist anhand der Blackboard Architektur entworfen worden. Im nächsten Teilkapitel wird darauf eingegangen. Es besteht aus drei Teilsystemen, den Agenten, die die verschiedenen Kontexte behandeln und entsprechende Reaktionen durchführen.
Zwei Agenten (Datenvorbereitungsagent und Interpretationsagent) werden als eine Rechenkomponente, die als virtuelle Maschine im „Living Place“, welche sich in einem Rechenraum befindet, realisiert. Die Vernetzung befindet sich in den Decken und Wänden. Somit verschwindet die Rechenkomponente im Hintergrund, ist jedoch allgegenwärtig. Das Prinzip von Ubiquitous Computing wird hier umgesetzt. Diese Agenten behandelt die Kontexte Aufmerksamkeit und Sitz-/Liegeprofile. Das Teilsystem steuert das Licht im Wohnbereich.

Das dritte Teilsystem ist der Mess-/Steuerungsagent, der den Kontext Lichtsituation behandelt. Außerdem nimmt er Befehle vom Interpretationsagent entgegen, um den Fernseher zu steuern. Der Benutzer dient als weiterer Input und steht für die Veränderung der Kontexte, durch seine Aktionen.

Blackboard Architektur

Die Blackboard Architektur wird für dieses System eingesetzt. Zunächst wird die Blackboard Architektur im Allgemeinen vorgestellt und anschließend folgt eine Erläuterung des System in Zusammenhang mit der Blackboard Architektur.
Die Blackboard Architektur wurde ursprünglich im Rahmen eines Spracherkennungssystems, namens Hearsay-II eingesetzt ([Erman u. a., 1980]). Ziel des System war es natürlichsprachige Anfragen an eine Literaturdatenbank zu erkennen. Hier wurde der Begriff Blackboard Architektur zum ersten Mal verwendet.
Eine Blackboard Architektur besteht aus zwei Elementen. Zum einen hat eine Blackboard Architektur ein Blackboard und eine Wissensquelle (Knowledge Sources). Oftmals wird die Blackboard Architektur als Agentensystem dargestellt. Diese Agenten haben unterschiedliche Fähigkeiten. Sie können Probleme lösen, als Input, Ouput oder Sonstiges genutzt werden. Die Gesamtheit der Agenten ist die Wissensquelle. Die Kommunikation zwischen den Agenten erfolgt über das Blackboard. Es können beispielsweise Lese- und Schreiboperationen bei den jeweiligen Agenten veranlasst werden (Erman u. a. [1980], Nii und Nii [1986]).

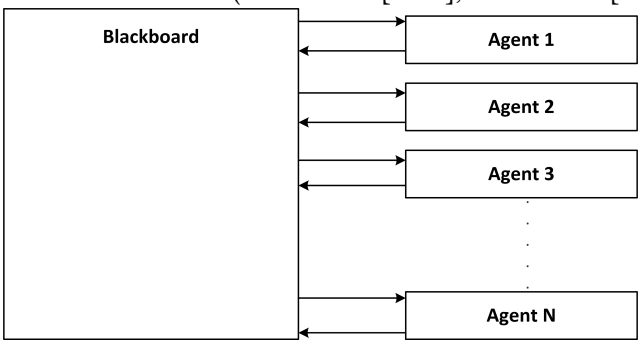

Abbildung 5.2: Blackboard Architektur

In Abbildung 5.2 sieht man die Blackboard Architektur. Die Menge der Agenten wird Wissensquelle bezeichnet. Die Wissensquelle sind logisch unabhängige „Experten“ oder „Spezialisten“ mit unterschiedlichen Aufgabengebieten. Sie haben spezielles Wissen und sind sensitiv auf für sie interessante („self-activating“) Veränderungen am Blackboard. Somit können die jeweiligen Agenten zur Lösung eines Problems etwas beitragen. Es ist beispielsweise möglich Proble-

me in Unterprobleme zu unterteilen und diese von den jeweiligen Agenten lösen zu lassen. Agenten können beliebig ausgetauscht, hinzugefügt oder entfernt werden. Das Blackboard ist die globale zentrale Datenquelle. Die Informationen die auf dem Blackboard liegen, werden unterschiedlich repräsentiert. Sie können unter anderem Fakten, Schlussfolgerungen, Befehle darstellen. Das Blackboard ist die einzige Kommunikationsmöglichkeit der Wissensquelle untereinander. Einige der Vorteile der Blackboard Architektur sind die Modularität, also das Aufteilen eines Ganzen zu Teile und diese Teile bilden für sich Teilsysteme, Robustheit, sprich das Funktionieren eines System unter schlechten Voraussetzungen und die Nebenläufigkeit, also das Ausführen von gleichzeitigen Berechnungen und Operationen. Aus den oben genannten Vorteilen wird dieser Architekturtyp verwendet.

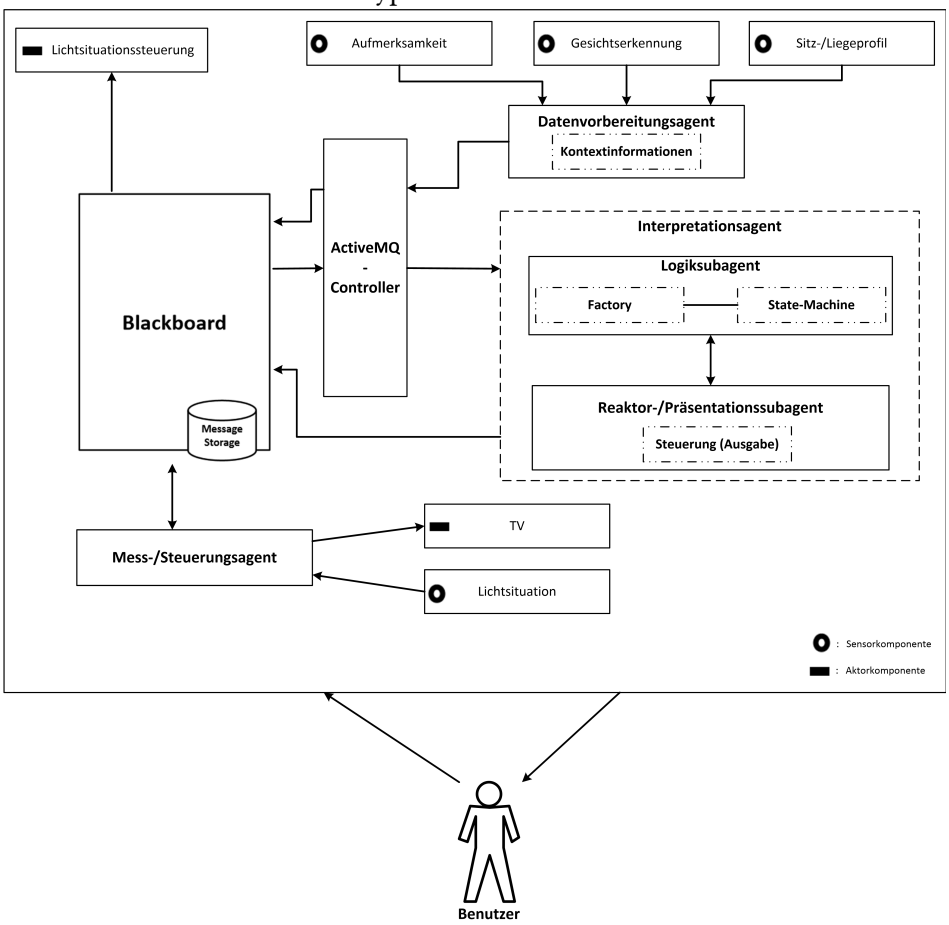

Abbildung 5.3: Blackboard Architektur des zu realisierenden Systems Couch 2.0

In Abbildung 5.3 sieht man die abstrakte Darstellung des Systems in Form der Blackboard Architektur. Es gibt keine direkte Eingabe vom Benutzer, somit entfällt eine GUI. Der Interpre-

tationsagent hat zwei Subagenten, ein Logiksubagent und ein Reaktor-/Präsentationssubagent. Der Reaktor-/Präsentationssubagent, dient als Ausgabeelement. Reaktionen aus dem Verhalten des Benutzers werden in Form von Steuerung des „Living Places“ dargestellt. Der Logiksubagent erhält, über den ActiveMQ-Controller per Blackboard, vom Datenvorbereitungsagent die Daten, um diese korrekt (in der Factory) zu interpretieren und entsprechende Steuerungen bzw. Ausgaben zum Reaktor-/Präsentationssubagent zu übermitteln. Außerdem enthällt der Logiksubagent einen Zustandsautomaten, um das Systemverhalten zu spezifizieren. Es werden die (indirekten) Eingaben des Benutzers, in Form von Kontextbetrachtungen vom Datenvorbereitungsagent entgegengenommen. Diese Daten stammen aus den jeweiligen Hardwarekomponenten. Der Interpretationsagent nimmt die Daten und interpretiert sie, um das System richtig reagieren zu lassen. Der Mess-/Steuerungsagent erfasst die Lichtsituation und schickt dies über das Blackboard an den Logiksubagent. Außerdem werden vom Reaktor-/Präsentationssubagent Befehle für die Steuerung des TV an den Mess-/Steuerungsagent über das Blackboard geschickt.
Das Blackboard wird durch das ActiveMQ (beschrieben in 4.3.2) realisiert. Es werden Schnittstellen (durch Topics) angeboten, um die Agenten gezielt anzusprechen, im Kapitel Realisierung werden diese Möglichkeiten aufgezeigt.
Damit die drei Agenten untereinander kommunizieren können gibt es einen ActiveMQ-Controller. Welcher als eine Komponente betrachtet werden kann. Die Daten werden vom Datenvorbereitungsagent erst über das ActiveMQ geschickt. Der ActiveMQ-Controller nimmt diese entgegen und leitet sie weiter an den Logiksubagenten. Dieser Schritt bzw. diese Entscheidung ist notwendig, da das „Living Place“ so aufgebaut ist, dass die Kommunikation über ActiveMQ (also das Blackboard) passiert. So können die Daten nicht nur intern, also für dieses System genutzt werden, sondern sind frei verfügbar für andere Anwendungen. Außerdem vereinfacht dies die Erstellung von Simulationsdaten, die von diesem System wie echte Daten aufgefasst werden können, um verschiedene Reaktionen herbeizuführen. Das Testen wird dadurch ebenfalls vereinfacht.

5.2 System Entwurf

In diesem Kapitel wird das System vorgestellt. Es erfolgt eine Darstellung der Komponenten anhand eines Komponentendiagramms. Das Verhalten des Systems wird mit Hilfe eines Zustandsautomaten dargestellt.

5.2.1 Komponentendiagramm

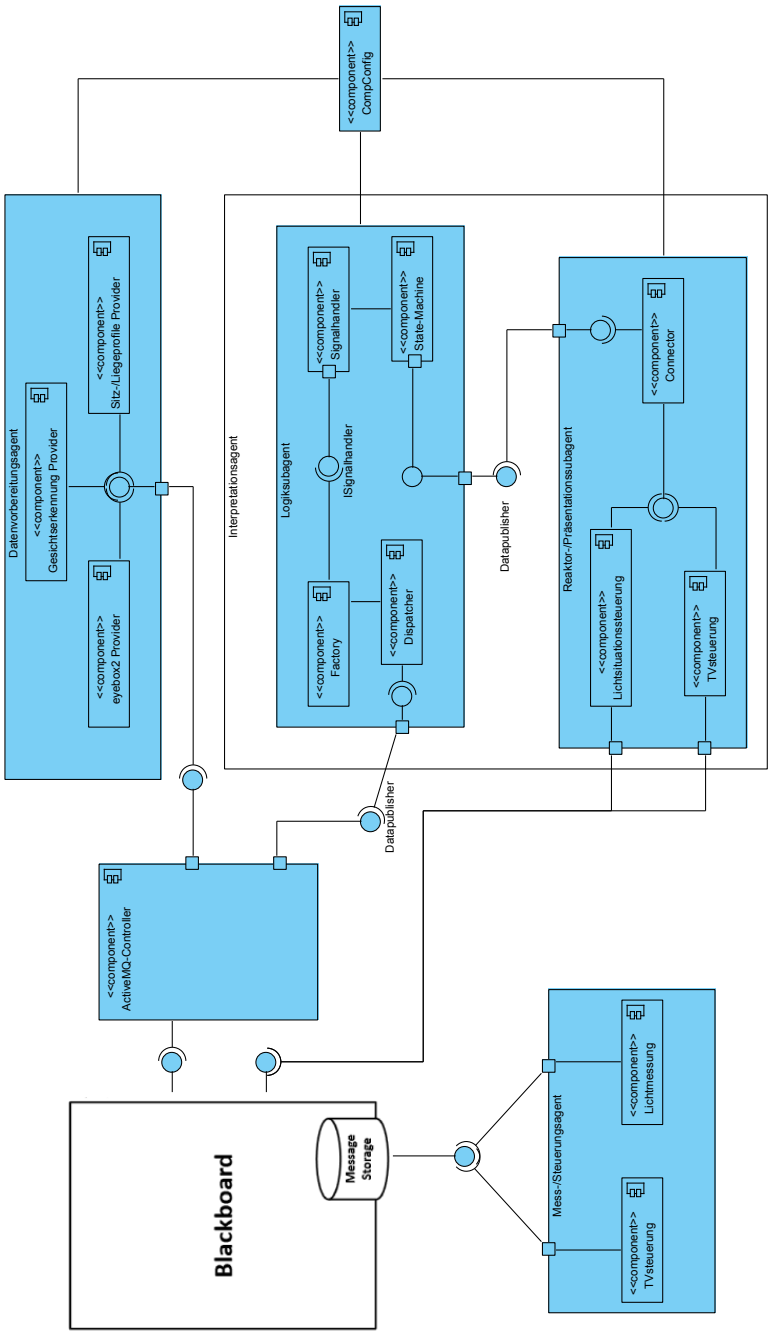

Abbildung 5.4: Komponentendiagramm

In Abbildung 5.4 sieht man das Komponentendiagramm des Systems. Die einzelnen Komponenten werden genauer beschrieben. Wie schon 5.1 beschrieben wird die Blackboard Architektur

verwendet. Das System besteht aus einen Datenvorbereitungsagent, Mess-/Steuerungsagent und einen Interpretationsagent, welcher aus einen Logiksubagent und Reaktor-/Präsentationsagent besteht. Die Agenten haben jeweils verschiedene Komponenten, welche hier vorgestellt werden sollen.

Blackboard Das Blackboard ist die zentrale Datenquelle. Daten können über sie empfangen und verschickt werden. Im „Living Place“ wird ActiveMQ für die Kommunikation eingesetzt und soll in diesem System als Blackboard dienen. ActiveMQ bietet bereits Schnittstellen für das Empfangen und Versenden von Nachrichten.

ActiveMQ-Controller Um die Kommunikationselemente Zentral zu halten, werden die Elemente im ActiveMQ-Controller gehalten. Der ActiveMQ-Controller verbindet sich mit dem Blackboard (also dem ActiveMQ) und registriert die gewünschten Kommunikationspartner, sei es Empfänger oder Sender.

CompConfig Der Komponentenkonfigurator ist ein bekanntes Entwurfsmuster, welches hier als Komponente dargestellt ist. Alle Komponenten und deren Abhängigkeiten werden hier konfiguriert. Eine nähere Erläuterung findet in Abschnitt 5.3 statt.

Datenvorbereitungsagent:

Gesichtserkennung Provider Diese Komponente sorgt für die Gesichtserkennung in diesem System. Auf Anfrage des Interpretationsagent kann eine Gesichtserkennung bzw. eine Gesichtswiedererkennung erfolgen. Die Daten werden über das ActiveMQ (Blackboard) dem Interpretationsagent (Logiksubagent) übermittelt.

eyebox2 Provider Der Kontext Aufmerksamkeit wird mit dieser Komponente behandelt. Es wird per xuuk eyebox2 die nötigen Informationen einer Person über das ActiveMQ an den Interpretationsagenten (Logiksubagent) geschickt. Konkretes wird im Kapitel Realisierung aufgezeigt.

Sitz-/Liegeprofile Provider Die Daten, die über die Couch erfasst werden, werden in dieser Komponente behandelt. Über das ActiveMQ werden die Daten den Interpretationsagenten (Logiksubagent) gesendet.

Interpretationsagent

Logiksubagent:

Dispatcher Der Dispatcher ist ein Verteiler von anstehenden Aufträgen, die vom Blackboard über den ActiveMQ-Controller eintreffen. Er nimmt die Daten, die vom Datenvorbereitungsagenten und Mess-/Steuerungsagenten kommen auf und verteilt sie an einen entsprechenden sogenannten „Worker“. Die Worker befinden sich in der Komponente Factory.

Factory Die eintreffenden Daten vom Dispatcher werden in der Factory aufgenommen. Je nach Zugehörigkeit wird ein bestimmter Worker beauftragt die Daten auszuwerten. Ein großer Teil der Logik wird somit in dieser Komponente behandelt.

Signalhandler In dieser Komponente treffen Signale ein, die von den Workern in der Factory ausgelöst wurden sind. Die Signale sind logische Entscheidungen, die in der Factory von den Workern, anhand von aktuellen Daten, getroffen wurden sind. Die Signale werden weitergeleitet an eine Zustandsmaschine, somit repräsentieren die Signale, Transitionen in der Zustandsmaschine.

State-Machine Um das System zu spezifizieren wird ein Zustandsautomat benötigt. Diese Komponente repräsentiert den Zustandsautomaten. Das System kann somit verschiedene Zustände annehmen, die benötigt werden um es zu spezifizieren bzw. klassifizieren. In den jeweiligen Zuständen erfolgen Befehle um die Aktoren des Systems zu steuern. Dies ist die zweite Komponente, die einen großen Anteil an Logik übernimmt.

Reaktor-/Präsentationssubagent:

Connector Der Connector ist das Bindeglied zwischen dem Logiksubagenten und den Reaktor-/Präsentationssubagenten. Befehle die von der Zustandsmaschine eintreffen werden vom Connector entgegengenommen und entsprechend delegiert. Es wird in dieser Komponente entschieden, welche Aktionen und Reaktionen der Benutzer erfahren soll.

Lichtsituationssteuerung Die Lichtsituation wird in dieser Komponente behandelt. Es werden Operationen bereitgestellt, die bestimmte Aktionen im „Living Place“ ausführen. Für die Lichtsituationssteuerung werden die Lichter im Wohnbereich gezielt gesteuert, sowie die Rollos. Die Steuerungsbefehle werden an das Blackboard gesendet. Es werden im „Living Place“ Module bereitgestellt, die die Nachrichten bezüglich der Steuerung

der Rollos und Lichter per Blackboard entgegennehmen und die daraus resultierenden Befehle ausführen.

TVsteuerung Um den TV zu steuern, werden, wie in der Komponente Lichtsituationssteuerung, Operationen bereitgestellt, die vom Connector aufgerufen werden. Diese Befehle werden an das Blackboard gesendet und vom Mess-/Steuerungsagenten entgegengenommen.

Mess-/Steuerungsagent:

TVsteuerung Die gesendeten Befehle vom Reaktor-/Präsentationssubagenten werden in dieser Komponente entgegengenommen und an den TV gesendet. Diese Komponente wird mit einem HDMI-CEC[1]-fähigen TV verbunden.

Lichtmessung Die Lichtmessung erfolgt mit Hilfe einer Sensorkomponente. Es werden in regelmäßigen Abständen Lichtmessungen im Wohnbereich durchgeführt. Die Ergebnisse der Messungen werden per Blackboard an den Logiksubagenten gesendet. Diese Komponente behandelt den Kontext Lichtsituation.

5.2.2 Sequenzdiagramm

Der Ablauf des System soll anhand eines exemplarischen Sequenzdiagramm vorgestellt werden. In Abbildung 5.5 sieht man das Sequenzdiagramm. Es erfolgt eine kurze Beschreibung der jeweiligen Punkte des Sequenzdiagramms.

[1] Consumer Electronics Control - Schnittstelle für Ansteuerungsfunktionen von elektronischen Konsumergeräten. CEC benutzt einen seriellen, einadrigen Datenbus für die Kommunikation.

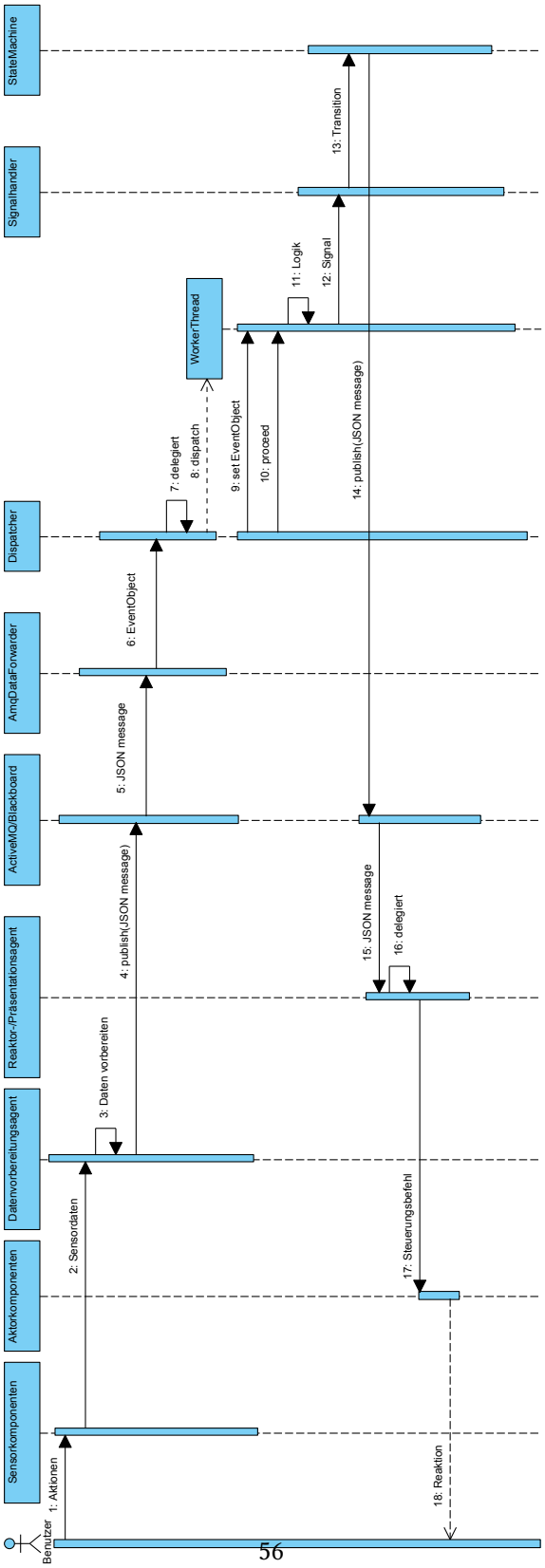

Abbildung 5.5: Sequenzdiagramm

1. Der Benutzer übt eine Aktion aus, die das System dazu veranlasst darauf zu reagieren. Die Sensorkomponenten nehmen diese Aktionen auf.

2. Die Sensordaten werden vom Datenvorbereitungsagenten entgegengenommen. Für die Kommunikation gibt es für jede der Sensorkomponenten eine jeweilige passende Schnittstelle.

3. Um die Daten in einem angemessenen Format weiterzusenden, müssen sie jeweils passend vorbereitet werden. Die Daten werden verschieden vorbereitet. Sie werden unter anderem mit alten Daten verglichen oder fehlerbehaftete Daten werden raus gefiltert, dies hängt jedoch von der jeweiligen Sensorkomponente ab.

4. Nach der Vorbereitung der Daten werden sie weitergeleitet an das Blackboard bzw. das ActiveMQ. Sie wird als passende JSON Nachricht versendet.

5. Man kann auf bestimmte Nachrichten sensitiv sein, so wird die Nachricht vom ActiveMQ-Controller entgegen genommen.

6. Es wird ein zur JSON-Nachricht passendes EventObject erstellt und an den Dispatcher weitergeleitet.

7. Im Dispatcher wird der passende Worker in der Factory ausgesucht.

8. Es wird dem entsprechenden Worker Informationen bzw. die Daten übermittelt. Die Daten werden verwendet und benötigt um eine geeignete Lösung bzw. Reaktion zu bestimmen.

9. Nach dem die benötigten Daten übermittelt wurden sind, kann der Worker seine Arbeit fortführen. Er wird von einem Ruhestand aufgeweckt.

10. Die passende Logik wird in Abhängigkeit der Daten (EventObject) verwendet, sprich wie hat sich das System bezüglich ihrer Kontexte geändert. Dies soll hier analysiert werden.

11. Ein Signal, welches eine Veränderung deutlich macht, wird dem Signalhandler gesendet.

12. Der Signalhandler nimmt das Signal auf und leitet es weiter an den Zustandsautomaten. Dabei spiegeln die Signale die Transitionen wider.

13. In den jeweiligen Zuständen wird in Abhänigkeit der eintreffenden Transitionen und zustätzlichen Daten, Logik ausgeführt. Es wird hier entschieden wie sich das System auf den Kontextwechsel verhalten soll.

14. Die Änderung des System wird in Form einer JSON-Nachricht an das Blackboard gesendet.

15. Die JSON-Nachricht wird vom Reaktor-/Präsentationsagenten entgegengenommen.

16. Der Reaktor-/Präsentationsagent delegiert die antreffende Nachricht und entscheidet welche Komponente verändert werden soll.

17. Diese Veränderung drückt sich in Form eines Steuerungsbefehl aus. Es wird an die zuständige Aktorkomponente übermittelt.

18. Der Benutzer erfährt nun aus seiner Aktion eine entsprechende Reaktion. Diese Reaktion kann der Benutzer bewusst oder unbewusst wahrnehmen. Es hängt von der Kontextänderung ab.

5.2.3 Zustandsautomat

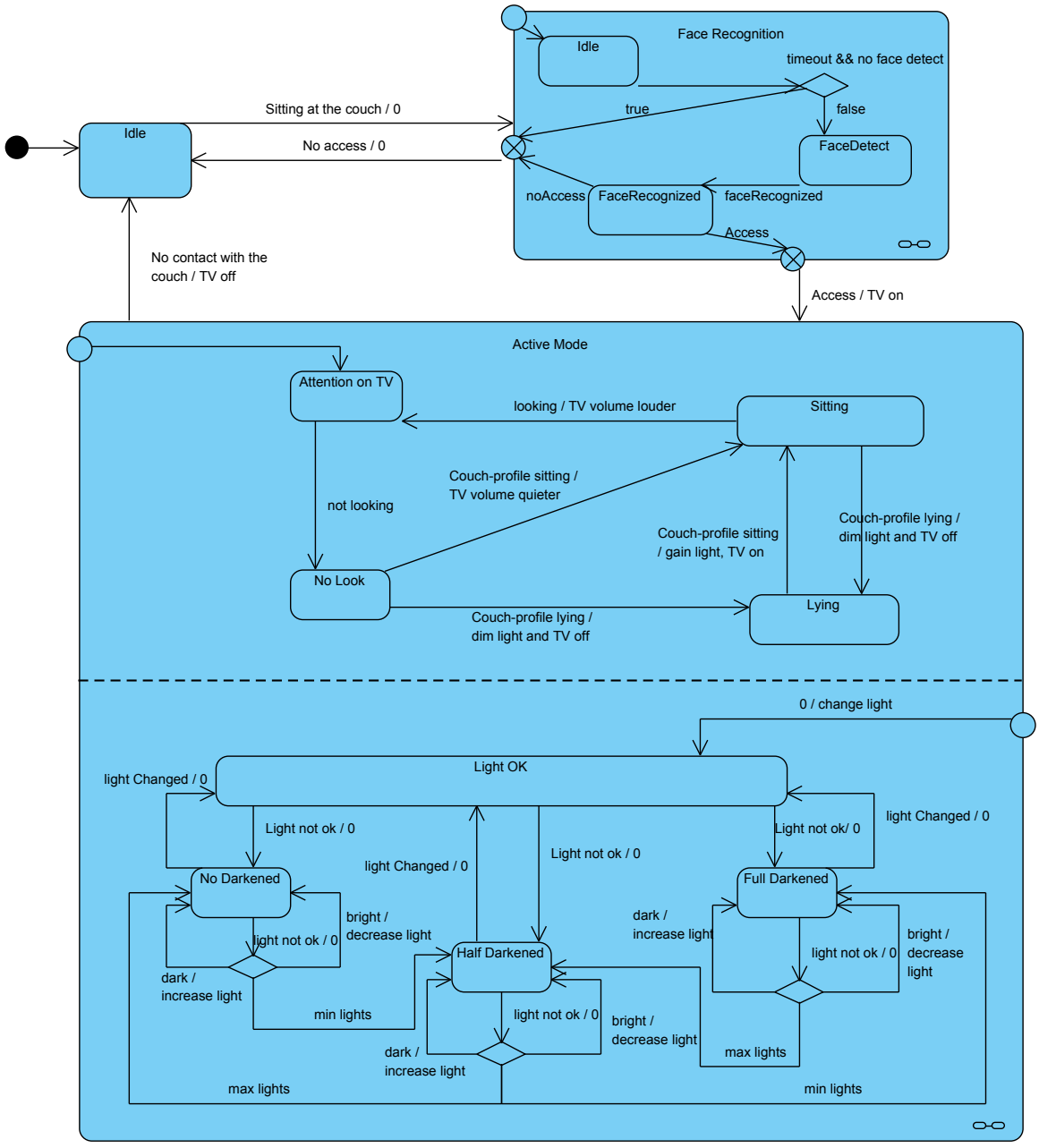

Abbildung 5.6: Zustandsautomat zur Spezifikation des Verhaltens des Systems

In Abbildung 5.6 sieht man den Zustandsautomaten, der das Verhalten des Systems beschreibt. Es gibt drei Zustände, wobei zwei davon Unterzustandsautomaten sind. Es erfolgt eine kurze

Erläuterung der drei Zustände, dabei ist zu erwähnen, dass die Zustände und Transitionen in englischer Sprache gehalten sind, da Bezeichnungen aller Entitäten in der Programmierung auf Englisch sind. Der Automat ist eine direkte Umsetzung.

Idle In diesem Zustand befindet sich das System, wenn er im Leerlaufbetrieb ist, sprich das System hat noch keinen Benutzer detektierten können. Die Detektion erfolgt bei Kontakt mit der Couch. In diesem Zustand gelangt man immer, wenn eine Person die Couch verlässt.

Face Recognition Um Entscheiden zu können welche Personen befugt sind TV zu gucken, erfolgt eine Gesichtswiedererkennung. Die Abläufe der Gesichtswiedererkennung werden in diesem Unterzustandsautomat dargestellt. Ist die Person befugt, so wechselt der Zustandsautomat in den Zustand Active Mode, ansonsten in den Zustand Idle.

Active Mode Dieser Zustand ist ebenfalls ein Unterzustandsautomat, der orthogonal ist. Es hat also zwei parallel laufende Zustandsautomaten. Der obere orthogonale Zustandsautomat beschreibt die Abläufe bei einer Kontextänderung der Aufmerksamkeit und Sitz-/Liegeprofile. Der Kontext Lichtsituation wird in dem unteren orthogonalen Zustandsautomaten behandelt. Die beiden Automaten können parallel laufen, da sie unabhängig von einander sind. Lediglich der Kontext Aufmerksamkeit und Sitz-/Liegeprofile sind einander abhängig, wie in 3.3.4 dargestellt wurde.

Um den unteren orthogonalen Zustandsautomaten zu verstehen erfolgt eine kurze Erläuterung. Entspricht die Lichtsituation den Anforderungen, so befindet sich der Zustandsautomat im Zustand „Light OK“. Je nach Situation der Rollos im Wohnbereich wechselt der Automat, bei unangemessener Lichtsituation, in den Zustand „No Darkened“, „Half Darkened“ oder „Full Darkened“. Die Zustände „No Darkened“ für nicht verdunkelt , „Half Darkened“ für halb verdunkelt und „Full Darkened“ für voll verdunkelt beschreiben jeweils die Zustände der Rollos im Wohnbereich, also offen, halb offen und zu. Die Sensorkomponente, die für die Messung des Lichts zuständig ist, schickt in zeitlich festgelegten Intervalllängen die Lichtwerte. Hat die Lichtstärke den geforderten Wert, so wechselt der Automat in den Zustand „Light OK“, ansonsten findet eine Regulierung der Lichtintensität der Lampen statt, bis die Lichtsituation den Anforderungen entspricht. Hat die Lichtintensität das Maximum oder das Minimum erreicht, so werden die Rollos passend entweder runter, halb-runter oder hoch gefahren und der Zustandsautomat wechselt in den entsprechenden Zustand. Ist beispielsweise das Maximum der Lichtintensität erreicht und die Rollos sind zu, jedoch ist es immer noch zu hell, so kann an

dieser Situation nichts geändert werden, da keine weiteren Ressourcen zur Verfügung stehen.

5.3 Entwurfsmuster - Pattern

In diesem Abschnitt werden die Entwurfsmuster, die in diesem System eingesetzt werden vorgestellt. Entwurfsmuster oder Pattern sind Lösungsmuster für bekannte Entwurfsprobleme. Sie werden in der Softwarearchitektur und -entwicklung eingesetzt. Um wiederkehrende Probleme zu lösen, werden die vordefinierten Muster als Vorlage benutzt und korrekt angewandt.
Nach der GoF werden die Entwurfsmuster in drei Arten von Mustern klassifiziert:

- Erzeugungsmuster: Diese Art von Muster beschäftigt sich mit der Erzeugung von Objekten. Hierbei wird die Konstruktion eines Objekts von der Repräsentation entkoppelt. Es wird versucht das Wissen einer Klasse und wie Instanzen erzeugt und verbunden werden zu verbergen (Gamma u. a. [1996]).
- Strukturmuster: Strukturmuster befassen sich mit der Strukturierung von Klassen und Objekten. Es werden hierbei Schnittstellen definiert, die bei Klassenmustern zur Übersetzungszeit festgelegt werden und bei Objektmustern zur Laufzeit festgelegt werden können (Gamma u. a. [1996]).
- Verhaltensmuster: Unter Verhaltensmustern wird jegliches Pattern gesehen, welches die Interaktionen zwischen Objekten beschreibt. Dabei wird Objekten ermöglicht untereinander zu kommunizieren, indem Schnittstellen identifiziert werden (Gamma u. a. [1996]).

5.3.1 Observer Pattern

Es erfolgt eine kurze Erläuterung des Oberserver Pattern, anschließend wird auf die Verwendung im System eingegangen. Das Observer Pattern gehört zu den Verhaltensmuster. Kurz ausgedrückt ist die Funktionsweise des Observer Pattern bei Änderung eines Objektes A, andere Objekte (Observer), die sich für dieses Objekt A registriert haben Bescheid zu geben. Durch eine Schnittstelle wird eine Aktualisierungsfunktion zur Verfügung gestellt, die vom Objekt A aufgerufen wird, um andere Objekte bei Änderungen zu informieren. Die Observer, die sich für ein Objekt interessieren registrieren sich bei diesem Objekt. Dieses Objekt enthält eine Datenstruktur mit den Objekten.

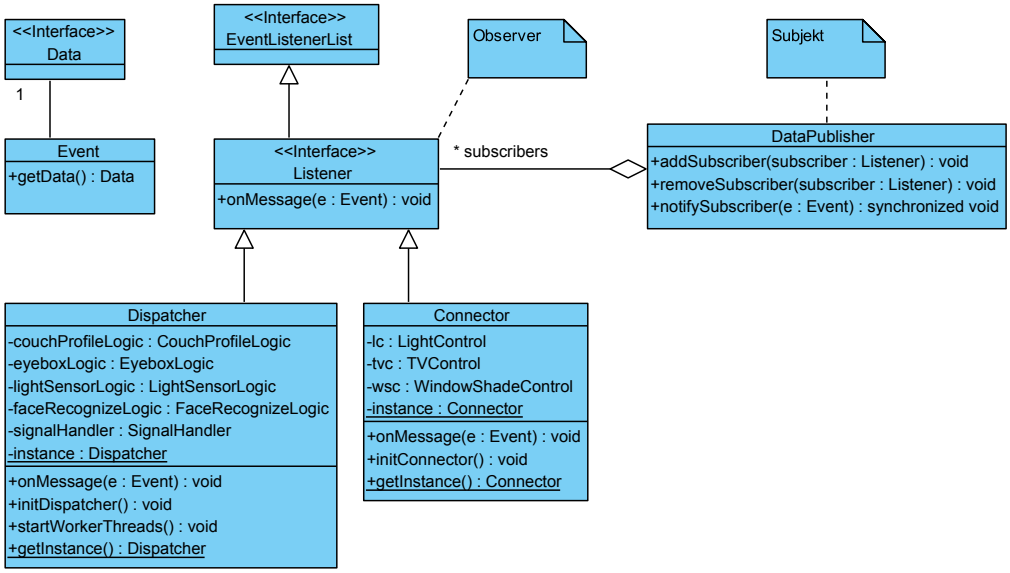

Abbildung 5.7: Observer Pattern

In Abbildung 5.7 sieht man das Klassenmodell des Observer Pattern und wie es konkret im System „Couch 2.0“ eingesetzt wird. Der DataPublisher ist das Subjekt, welches mit den Methoden addSubscriber und removeSubscriber, Subscriber bzw. Observer in eine Datenstruktur hinzufügt oder entfernt. Mit notifySubscriber werden die registrierten Observer in der Datenstruktur benachrichtigt. Bei der Benachrichtigung kann das Subjekt, dem Observer ein Event übermitteln. Der Listener ist ein Observer, der sich bei einem Subjekt registriert. Die onMessage-Methode wird bei der Benachrichtigung des Listeners vom Subjekt aufgerufen.
Das Observer Pattern wird eingesetzt, um zwischen den beiden Subagenten, Logiksubagent und Reaktor-/Präsentationssubagent, im Interpretationsagenten, die Befehle, Daten und Anweisungen zu übermitteln. Dafür muss sich der Reaktor-/Präsentationssubagent beim Logiksubagent registrieren, um auf Änderungen sensitiv zu sein. Außerdem wird das Observer Pattern eingesetzt um eingehende Daten vom ActiveMQ-Controller an den Dispatcher zu übermitteln. Der Dispatcher ist somit ein Bindeglied zwischen dem Datenvorbereitungsagenten (wobei der ActiveMQ-Controller dazwischen liegt) und dem Logiksubagenten und der Connector vermittelt zwischen Logiksubagent und Reaktor-/Präsentationssubagent.

5.3.2 Component Configurator

In diesem System wurde das Component Configurator Pattern, welches ein Strukturmuster ist, eingesetzt. Es erlaubt einer Applikation ihre Komponenten im System zur Laufzeit hinzu zufü-

gen und zu entfernen, ohne dabei die Applikation an sich zu modifizieren, rekompilieren oder statisch wieder zu verlinken. Des weiteren ist es möglich Komponenten neu zu konfigurieren ohne das System neu zu starten (Rosa und Silva [1997]).

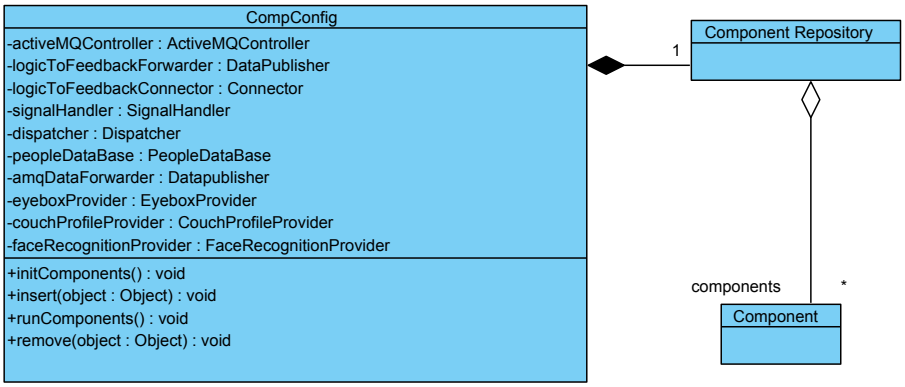

Abbildung 5.8: Componenten Configurator

In Abbildung 5.8 sieht man den Component Configurator des Systems. Da das System aus unterschiedlichen Komponenten besteht, bietet sich dieses Pattern an. Ein weiterer Vorteil ist, dass die Komponenten überschaubar und zentral in einer Komponente liegen. So kann man sich schnell Einblick über das System verschaffen.
Die Komponenten werden vom Component Configurator initialisiert und in ein Repository hinzugefügt. Nach dem Hinzufügen werden die Komponenten gestartet. Je nach Situation können Komponenten verändert werden, in dem sie beispielsweise vorher pausiert werden, damit das System nicht gestört wird. Soll das System herunterfahren kann der Component Configurator die Komponenten beliebig zerstören.

5.3.3 State Pattern

Um das Verhalten des Systems zu spezifizieren wird ein Zustandsautomat benötigt. In der Umsetzung wird dies durch das State Pattern realisiert. Es gehört zu den Verhaltensmustern und ist ein Gang Of Four-Pattern. Laut GOF kann das State Pattern wie folgt beschrieben werden:

> *„Ermögliche es einem Objekt, sein Verhalten zu ändern, wenn sein interner Zustand sich ändert. Es wird so aussehen, als ob das Objekt seine Klasse gewechselt hat.“ - Gamma u. a. [1996]*

Dieses Pattern realisiert einen zustandsabhängiges Verhalten eines Objekts. In Abhängigkeit des Zustands in welches es sich befindet, ändert das Objekt sein Verhalten. Zur Laufzeit ist es

also möglich verschiedene Aktionen auszuführen. Mit diesem Pattern wurde der Zustandsautomat, der in 5.2.3 vorgestellt wurde realisiert.

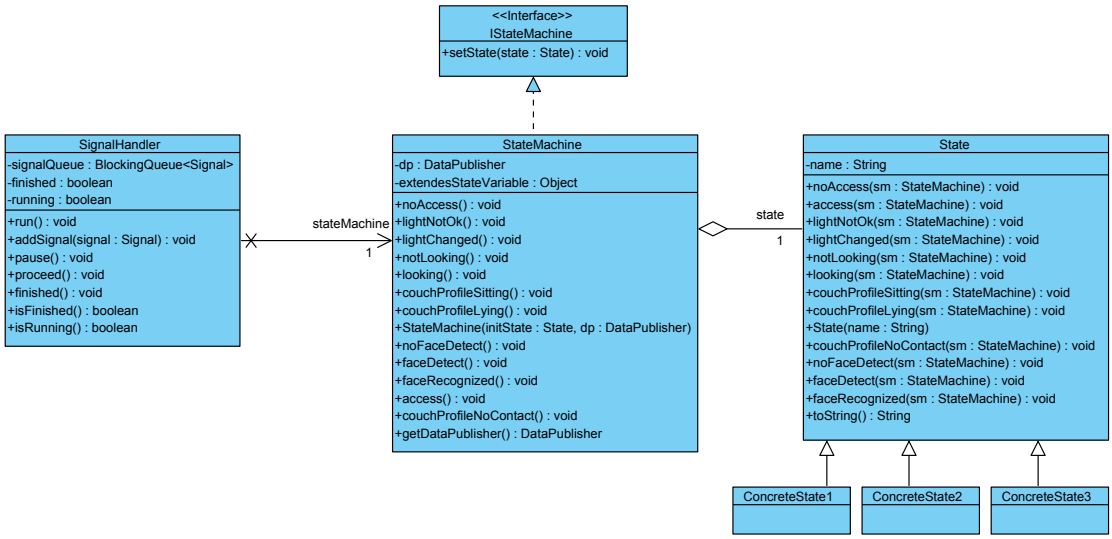

Abbildung 5.9: State machine Pattern

In Abbildung 5.9 sieht man die Umsetzung des State Pattern. Die abstrakte Klasse State enthält alle Transitionen als Methoden, die im Zustandsautomaten eintreffen können. Konkrete Zustände sind als Klassen definiert und erben von State. Die konkreten Zustände enthalten die Realisierung der Methoden der für sie interessanten Transitionen, alle anderen Transitionen werden nicht realisiert. Der aktuelle Zustand wird in der Klasse StateMachine gehalten und kann von einer konkreten Klasse mit der Methode „setState“ geändert werden.
Die Komponente SignalHandler reicht die eintreffenden Signale, wie in 5.2 beschreiben, weiter an die StateMachine und die StateMachine führt die gewünschte Transition aus.

5.4 Fazit

Mit dem entworfenen Design wird die Grundlage für die Realisierung des Systems bereitgestellt. Die vorgestellte Architektur beschreibt das System und die Wechselwirkungen zwischen den Komponenten. Mit einem Sequenzdiagramm wird der Ablauf von eintreffenden Aktionen des Benutzers beschrieben und ein Zustandsautomat spezifiziert das Verhalten des Systems. Flexibilität wurde in diesem System angestrebt. Module können im Nachhinein entworfen werden und hinzugefügt und entfernt werden. Dies wird durch die Systemarchitektur, die als Blackboard-Architektur konzipiert ist, begünstigt. So ist es auch möglich die Komponenten in

verschiedenen System unterzubringen, um ein verteiltes System zu schaffen. Eines der Vorteile des verteilten Systems ist die hohe Performance. In dieser Hinsicht ist dies, bei diesem kleinen System noch nicht erforderlich, jedoch wird diese Option, bei Erweiterung des Systems, offen gehalten

Die Vorstellung des Plug-ins soll hier aufgegriffen werden. Erweiterungsmodule, die dynamisch zur Laufzeit entdeckt und eingebunden werden können, dieses Verhalten kann durch dieses Design ermöglicht werden. Es sind jedoch Verarbeitungselemente notwendig, um die Plug-ins angemessen zu realisieren. Dies wird durch dieses Design nicht automatisiert.

6 Realisierung

In diesen Kapitel soll die Realisierung des Systems „Couch 2.0“ behandelt werden. Dabei werden die Anforderungen und Erarbeitungen in Kapitel 3 hier verwendet. Die im Kapitel 5 vorgestellte Designpunkte sollen Vorlage sein für die Umsetzung des Systems.
Die Agenten sollen gemeinsam mit ihren erforderlichen Umsetzungsmitteln in diesem Kapitel genauer vorgestellt werden. Die Aspekte Ubiquitous Computing und Context aware computing werden gegebenenfalls im Zusammenhang mit der Umsetzung gebracht.

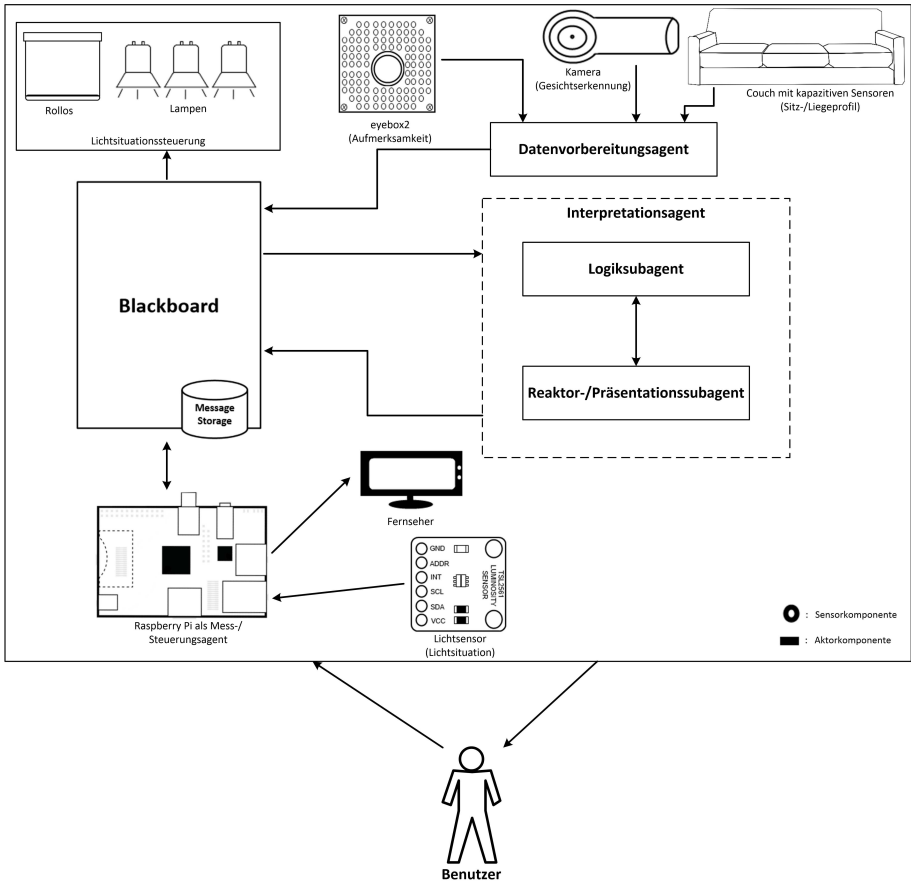

Abbildung 6.1: Laborumgebung

In Abbildung 6.1 ist die Laborumgebung mit ihren Komponenten abgebildet. Dabei wurden die jeweiligen Komponenten, die in 5.1 zu sehen sind, mit der verwendeten Hardware ersetzt.

6.1 ActiveMQ als Blackboard

Das ActiveMQ soll als Blackboard benutzt werden. Die Funktionsweise des ActiveMQ wurde in Abschnitt 4.3.2 vorgestellt. Das ActiveMQ erfüllt die Voraussetzung um als Blackboard (5.1) genutzt zu werden.
In der Metapher werden Nachrichten im Blackboard getan, die von Interessenten abgeholt werden. In ActiveMQ wird dies durch das Publisher-Subscriber-Prinzip verwirklicht. Die Nachrichten werden in Topics empfangen und gesendet.
Es werden die Topics des Systems und deren Bedeutung vorgestellt.

Topic	Aufgabe/Bedeutung
EyeboxData	Es werden die Daten, die für den Kontext Aufmerksamkeit stehen, vom Datenvorbereitungsagenten an den Logiksubagenten, übertragen. Die Daten werden von der xuuk eyebox2 entnommen.
CouchProfile	Die Veränderungen die an der Couch statt finden, werden vom Datenvorbereitungsagenten an den Logiksubagenten übermittelt.
LightSensorData	Die Lichtstärke, die für die Bestimmung des Kontextes Lichtsituation mit bestimmend ist, wird vom Mess-/Steuerungsagenten gemessen und an den Logiksubagenten gesendet.
LightSensorControl	Für Testzwecke, um den Mess-/Steuerungsagenten Befehle zu geben, wie „mess“ oder „stopTest“, in der Regel wird die Lichtstärke in fixen Zeitintervallen gesendet.
FaceRecognitionData	Daten der Gesichtserkennung und -wiedererkennung werden vom Datenvorbereitungsagenten an Logiksubagenten gesendet. Der Logiksubagent gibt bei Bedarf Befehle für die Sendung der Daten.
LP.LIGHTCONTROL	Schnittstelle um die Aktorkomponenten Rollos und Licht im „Living Place“ zu steuern. Befehle werden vom Reaktor-/Präsentationsagenten über das Blackboard direkt an die Aktorkomponenten geschickt.

Tabelle 6.1: Topic und deren Aufgaben/Bedeutung

6.2 Reaktor-/Präsentationssubagent

Bindeglied zwischen den Subagenten

Das Bindeglied zwischen den Subagenten Logiksubagent und Reaktor-/Präsentationsagenten ist ein Connector, der das Observer Pattern realisiert. Befehle treffen vom Logiksubagenten ein.

```
1 public class Connector implements Listener { ...
```

Listing 6.1: Connector

Der Connector ist ein Listener, also ein Observer und wird in einem DataPublisher registriert.

```
1  public class DataPublisher {
2
3    private EventListenerList subscribers;
4    ...
5    public void addSubscriber(Listener subscriber) {
6      this.subscribers.add(Listener.class, subscriber);
7    }
8    ...
9    public synchronized void notifySubscriber(Event e) {
10     for (Listener subscriber :
11                 subscribers.getListeners(Listener.class)) {
12       subscriber.onMessage(e);
13     }
14   }
15 }
```

Listing 6.2: DataPublisher

Im Zustandsautomaten wird bei Bedarf die notifySubscriber-Methode aufgerufen, die als Event die Befehle für den Reaktor-/Präsentationssubagenten beinhaltet.

Lichtsituationssteuerung

Die Lichtsituationssteuerung besteht aus zwei Aktorkomponenten. Sie beschreibt die Reaktionen der Kontextveränderung Lichtsituation und besteht aus der Steuerung der Rollos und der Lichter im „Living Place“.

Rollos

Die Rollos werden im Connector über eintreffende Benachrichtigungen vom Logiksubagenten (in der Zustandsmaschine) gesteuert.

```
1  public class WindowShadeControl {
2  ...
3    public void windowShadeFunc(WindowShadeType wst){
4      String msg = "";
5      if(wst == WindowShadeType.OPEN){
6        msg = windowShadeOpen;
7      }else if (wst == WindowShadeType.CLOSE){
8        msg = windowShadeClose;
9      }else if (wst == WindowShadeType.HALF){
10       msg = windowShadeHalf;
11     }
12
13     LightControlJSON json = new LightControlJSON(msg, ID, VERSION);
14     this.windowShade.changeState(wst);
15     this.amqPublishser.publish(this.jsonParser.makeJsonMsg(json));
16   }
```

Listing 6.3: Rollosteuerung

Die Befehle sind JSON-Nachrichten, die ans Blackboard (ActiveMQ) gesendet werden und die Aktorkomponente direkt ansteuert.

Lichtsteuerung

Die Lichtsteuerung geschieht durch das selbe Prinzip, wie die Rollosteuerung. Befehle treffen ein und werden weiter per JSON-Nachrichten ans Blackboard gesendet. Im Wohnbereich des „Living Place“ gibt es 2 Reihen a 8 Lampen.

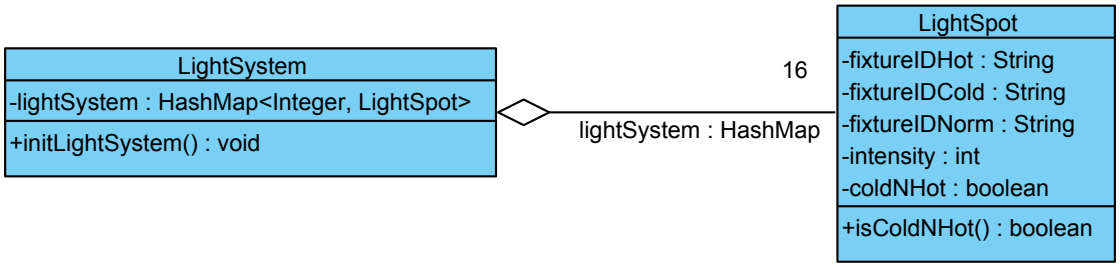

Abbildung 6.2: Lichtsystem

Das Lampensystem wird als eine Lampe betrachtet, bei der Regulierung der Lichstärke wird die Lichtintensität jeder Lampe um die gewünschte Intensität verringert oder vergrößert.

```
1  public class LightControl {
2  ...
3    /**
4     * @brief Increase the light intensity of each spot.
5     * @param intensity
6     */
7    public void increaseLightIntensity(String intensity){
8    ...
9    }
10
11   /**
12    * @brief Decrease the light intensity of each spot.
13    * @param intensity
14    */
15   public void decreaseLightIntensity(String intensity){
16   ...
17   }
18
19   /**
20    * @brief Set the light intensity of a spot.
21    * @param spot
22    * @param intensity
23    */
24   public void setLightIntensity(int spot, String intensity){
25   ..
26   }
27
28   /**
29    * Row:
30    * [1]: 1 2  3  4  5  6  7  8
31    * [2]: 9 10 11 12 13 14 15 16
32    * @param row set light intensity of row 1 or 2
33    * @param intensity
34    */
35   public void setLightIntensityOfRow(int row, String intensity){
36   ..
37   }
```

Listing 6.4: Lampensteuerung

TVsteuerung

Die Fernsehersteuerung bedient sich ebenfalls dem selben Prinzip wie die obigen (6.3, 6.4) Aktorkomponenten.

```
1  public class TVControl {
2  ...
3    public void switchTVON(){
4          ...
5    }
6
7    public void switchTVOFF(){
8      ...
9    }
10
11   public void muteTV(){
12     ...
13   }
14
15   public void unmuteTV(){
16     ...
17   }
```

Listing 6.5: TV Steuerung

6.3 Logiksubagent

Bindeglied zwischen Interpretationsagent und Blackboard

Die Daten, die vom Datenvorbereitungsagenten und vom Mess-/Steuerungsagenten ans Blackboard gesendet werden, werden vom Logiksubagenten aufgenommen. Dies geschieht durch das selbe Prinzip, wie beim Connector (6.1). Es wird das ObserverPattern benutzt. Der ActiveMQ-Controller hat einen Datapublisher, der wiederum hat den Dispatcher, als Observer. Treten neue Nachrichten auf wird der Dispatcher vom ActiveMQ-Controller durch den Datapublisher benachrichtigt (siehe 5.2.2).

```
1  public class Dispatcher {
2  ...
```

```
3   public void onMessage(Event e) {
4
5     if (e.getData() instanceof CouchProfileInfo) {
6       couchProfileLogic.setCouchInfo((CouchProfileInfo) e.getData());
7       synchronized (couchProfileLogic) {
8         couchProfileLogic.proceed();
9       }
10    } else if(e.getData() instanceof EyeboxInfo){
11      eyeboxLogic.setEyeboxInfo((EyeboxInfo) e.getData());
12      synchronized (eyeboxLogic) {
13        eyeboxLogic.proceed();
14      }
15    } else if(e.getData() instanceof LightSensorInfo){
16      lightSensorLogic.setLightSensorInfo((LightSensorInfo)
17
18      synchronized (lightSensorLogic) {
19        lightSensorLogic.proceed();
20      }
21    } else if (e.getData() instanceof FaceRecognitionInfo){
22      faceRecognizeLogic.setFaceRecognitionInfo((FaceRecognitionInfo)
23
24      synchronized (faceRecognizeLogic) {
25        faceRecognizeLogic.proceed();
26      }
27    }
28  }
29  ...
```

Listing 6.6: Dispatcher - onMessage

Die onMessage-Methode wird bei der Benachrichtigung aufgerufen. Der Dispatcher verteilt, je nach Typ.

Abarbeitung der Daten in der Factory

In der Factory gibt es verschiedene Komponenten, die als Threads realisiert sind. Sie führen pro Datentyp eine Logik aus, um die Daten in angemessener Weise zu verarbeiten. Dabei senden die Komponenten ein Signale an den Signalhandler, der speichert die Signale in eine Queue. Die Signale in der Queue werden an den Zustandsautomaten weitergeleitet.

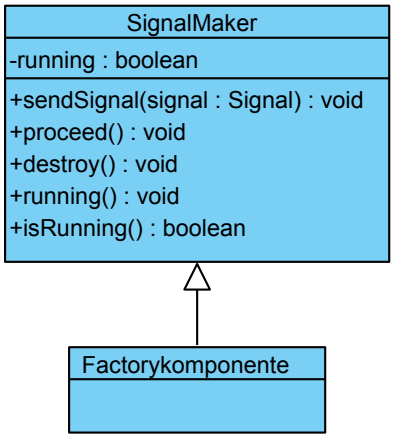

Abbildung 6.3: Lichtsystem

Jede Factorykomponente ist ein Signalmaker. Es werden die Komponenten also die Worker der Factory vorgestellt.

CouchProfileLogic Dieser Worker nimmt die Daten bzgl. der Veränderung der Couch auf. Die Couch-Daten werden aktualisiert und es wird überprüft welche Position der Benutzer hat.

```
1  public class CouchProfileLogic extends SignalMaker {
2  ...
3    /**
4     * @brief Update the couch with the changes.
5     */
6    private void updateCouch() {
7      ArrayList<SofaViewModel> couch = this.couchInfo.getSofa();
8      SofaViewModel sofaSegment = null;
9      for (int i = 0; i < couch.size(); i++) {
10       sofaSegment = couch.get(i);
11       this.currentCouch.setColorAtCouchID(sofaSegment.getID(),
12           sofaSegment.getColor());
13     }
14   }
15
16   /**
17    * @brief Check which position a person has.
18    *
19    * @return Couchprofile NOCONTACT, SITTING, LYING
```

```
20    */
21   public CouchProfile checkPosition() {
22     CouchProfile couchProfile = null;
23     ArrayList<SofaViewModel> couch =
24                         this.currentCouch.getCouch();
25     int redCnt = 0;
26     String id = "";
27     String color = "";
28     for (int i = 0; i < couch.size(); i++) {
29       id = couch.get(i).getID();
30       color = couch.get(i).getColor();
31       if (!(id.equals("00") || id.equals("10") || id.equals("20")
32           || id.equals("30") || id.equals("40")
33                                         || id.equals("50"))) {
34         if (color.equals("Red")) {
35           redCnt++;
36         }
37       }
38     }
39
40     if (redCnt == 0) {
41       couchProfile = CouchProfile.NOCONTACT;
42     } else if (redCnt > 0 && redCnt <= 2) {
43       couchProfile = CouchProfile.SITTING;
44     } else if (redCnt > 2) {
45       couchProfile = CouchProfile.LYING;
46     }
47     return couchProfile;
48   }
49   ...
```

Listing 6.7: CouchProfileLogic

Eine ID steht für einen kapazitiven Sensor der Couch.

Abbildung 6.4: Couch mit IDs

In 3.3.1 wird auf den Kontext Sitz-/Liegeprofil eingegangen.

EyeboxLogic Die Signale die hier an den Signalhandler weitergesendet werden bestimmten den Kontext Aufmerksamkeit. Daten von der xuuk eyebox2 werden hier gefiltert entgegengenommen und direkt weitergesendet. Auf die Filterung wird im in der Erläuterung des Datenvorbereitungsagenten eingegangen.

FaceRecognizeLogic Bestimmung ob Gesichtserkennung oder Gesichtswiedererkennung, weitersenden der Daten an die Zustandsmaschine.

LightSensorLogic Die Lichtstärke wird überprüft auf die gewünschte Anforderung (3.3.3). Der Kontext Lichtsituation wird hier behandelt.

```
1  public class LightSensorLogic extends SignalMaker {
2  ...
3    @Override
4    public void run(){
5      while (this.isRunning()) {
6
7        // need pause
8        synchronized (this) {
9          try {
10           wait();
11         } catch (InterruptedException e) {
12           e.printStackTrace();
13         }
14       }
15
16       if(this.lightSensorInfo.getLux() > LUXMAX ||
17                       this.lightSensorInfo.getLux() < LUXMIN){
```

```
18          Signal signal = Signal.LIGHTNOTOK;
19          signal.setAdditionalData(this.lightSensorInfo);
20          this.sendSignal(signal);
21        }else{
22          this.sendSignal(Signal.LIGHTCHANGED);
23        }
24
25      }
26    }
27    ...
```

Listing 6.8: LightSensorLogic

Zustandsautomat zur Spezifikation des Verhaltens

Der Zustandsautomat dient zur Spezifikation des Verhaltens des System. Es wird kurz auf die Funktionsweise des State Pattern eingegangen und wichtige logische Verarbeitungen werden aufgezeigt.
Der SignalHandler erhält von den jeweiligen Worker in der Factory die Signale. Diese werden nacheinander aus einer Queue entnommen und an die Zustandsmaschine ubermittelt. Die Zustandsmaschine interpretiert die Signale in Form von Transitionen.

```
1  public class SignalHandler extends Thread implements ISignalHandler{
2  ...
3   @Override
4    public void run() {
5
6      Signal signal = null;
7      while(!finished){
8
9        try {
10         signal = this.signalQueue.take();
11       } catch (InterruptedException e1) {
12         e1.printStackTrace();
13       }
14
15       switch (signal) {
16       case COUCHPROFILESITTING:
17         this.stateMachine.couchProfileSitting();
18         break;
```

```
19        case COUCHPROFILELYING:
20          this.stateMachine.couchProfileLying();
21          break;
22        case COUCHPROFILENOCONTACT:
23          this.stateMachine.couchProfileNoContact();
24          break;
25        case ACCESS:
26          this.stateMachine.access();
27          break;
28        case NOACCESS:
29          this.stateMachine.noAccess();
30          break;
31        case LOOKING:
32          this.stateMachine.looking();
33          break;
34        case NOTLOOKING:
35          this.stateMachine.notLooking();
36          break;
37        case LIGHTNOTOK:
38          this.stateMachine.setExtendesStateVariable(
39                            (LightSensorInfo)signal.getAdditionalData());
40          this.stateMachine.lightNotOk();
41          break;
42        case LIGHTCHANGED:
43          this.stateMachine.lightChanged();
44          break;
45        case FACEDETECT:
46          this.stateMachine.faceDetect();
47          break;
48        case NOFACEDETECT:
49          this.stateMachine.noFaceDetect();
50          break;
51        case FACERECOGNIZED:
52          this.stateMachine.setExtendesStateVariable(
53                                  (String)signal.getAdditionalData());
54          this.stateMachine.faceRecognized();
55          break;
56        default:
57          System.out.println("Error_-_Signal_nicht_vorhanden");
58          break;
```

```
59        }
60
61        // need for pause
62        if (this.signalQueue.isEmpty()) {
63          synchronized (this) {
64            while (this.signalQueue.isEmpty() && !finished) {
65              try {
66                wait();
67              } catch (InterruptedException e) {
68                e.printStackTrace();
69              }
70            }
71          }
72        }
73      }
74
75    }
76    ...
```

Listing 6.9: SignalHandler

Zusätzliche Zustände, wie die momentane Lichtstärke oder das Gesicht, welches wiedererkannt wurde, werden in Form von Extended State Variablen dargestellt. Es werden keine extra Zustände benötigt, da es ausreicht diesen Wert in Form einer Variable festzuhalten.

```
1  public class StateMachine implements IStateMachine {
2  ...
3    public StateMachine(State initState, DataPublisher dp) {
4      this.state = initState;
5      this.dp = dp;
6    }
7
8    @Override
9    public void setState(State state) {
10     this.state = state;
11   }
12
13   ...
14
15   public void noFaceDetect(){
16     this.state.noFaceDetect(this);
```

```
17    }
18    ...
```

Listing 6.10: StateMachine

Die Klasse StateMachine hat eine Variable „state“ vom Typ „State“, die den aktuellen Zustand der Zustandsmaschine hält. Bei einer Transition wird die StateMachine als Parameter übergeben und mit der Methode setState in 6.10 kann der Zustand der Zustandsmaschine in einen anderen Zustand gewechselt werden. Im Code 6.10 sieht man ein Beispiel einer Transition (noFaceDetect).
Es sollen einige wichtige Zustände und deren Logiken vorgestellt werden.

Erläuterungen von wichtigen Zuständen

FaceRecognized

In diesem Zustand wird das Ergebnis der Gesichtswiedererkennung des Datenvorbereitungsagenten mit einer Person in der Datenbank verglichen. Ist es der Person laut einer Policy erlaubt zur Zeit der Anfrage das System zu bedienen, so wird das System aktiv, ansonsten bleibt das System im Idle-Zustand.

```
1  public class FaceRecognized extends State {
2  ...
3    public FaceRecognized(StateMachine sm){
4      super("FaceRecognition/FaceRecognized");
5      PeopleDatabase peopleDB = PeopleDatabase.getInstance();
6      this.faceRecognitionProvider =
7                           FaceRecognitionProvider.getInstance();
8
9      Person p = peopleDB.getPerson(
10                         (String)sm.getExtendesStateVariable());
11     if(p == null){
12       sm.setState(new Idle());
13     }else{
14       if(p.getPolicy().isAllowed()){
15         sm.setState(new ActiveMode(sm.getDataPublisher()));
16       }else{
17         sm.setState(new Idle());
18       }
19     }
```

```
20      this.faceRecognitionProvider.destroy();
21    }
22    ...
```

Listing 6.11: Gesichtswiedererkennung

Bei der Policy-Überprüfung wird geschaut, ob die aktuelle Zeit im Rahmen der der Policy ist.

NoLook

Wurde vom System erkannt, dass die Person nicht mehr auf den TV schaut wird je nach Position des Benutzers verschiedene Aktionen durchgeführt.

```
1  public class NoLook extends State{
2  ...
3    @Override
4    public void couchProfileSitting(StateMachine sm) {
5      sm.getDataPublisher().notifySubscriber(new Event(
6                              new TVActionInfo(TVAction.TVMUTE)));
7      sm.setState(new Sitting());
8    }
9
10   @Override
11   public void couchProfileLying(StateMachine sm) {
12     DataPublisher dp = sm.getDataPublisher();
13     dp.notifySubscriber(new Event(new TVActionInfo(TVAction.TVMUTE)));
14     dp.notifySubscriber(new Event(new TVActionInfo(TVAction.TVOFF)));
15     dp.notifySubscriber(new Event(
16                              new WindowShadeInfo(WindowShadeType.CLOSE)));
17     dp.notifySubscriber(new Event(
18               new LightControlActionInfo(LightControlAction.LIGHTOFF)));
19     sm.setState(new Lying());
20   }
21   ...
```

Listing 6.12: Zustand - NoLook

Die Aktionen sind anhand der Events beschrieben. In Zustand **Lying** und **Sitting** werden entsprechend inverse Aktionen durchgeführt, um den Zustand wiederherzustellen, falls der Benutzer seine Aufmerksamkeit dem TV wieder widmet. In Anbetracht dessen ist es nicht erforderlich diese Zustände aufzuzeigen.

LightOK

In diesem Zustand gibt es nur eine Transition und je nach dem in welchen Zustand die Rollos im Wohnbereich sind wird der Zustand der Zustandsmaschine dementsprechend gewechselt.

```
public class LightOK extends State {
...
   @Override
  public void lightNotOk(StateMachine sm) {
    WindowShadeType wst =
      WindowShadeControl.getInstance().getWindowShadeState();
    if(wst == WindowShadeType.OPEN){
      sm.setState(new NoDarkened());
    }else if (wst == WindowShadeType.CLOSE){
      sm.setState(new FullDarkened());
    }else if (wst == WindowShadeType.HALF){
      sm.setState(new HalfDarkened());
    }
  }
  ...
```

Listing 6.13: Zustand - LightOK

FullDarkened

Die Lichtregulierung soll anhand dieses Zustands (6.14) kurz aufgezeigt werden. Die Funktionsweise des Ablaufs wurde in 5.2.3 erläutert.

```
public class FullDarkened extends State {
...
  @Override
  public void lightNotOk(StateMachine sm) {
    LightSensorInfo lsi = (LightSensorInfo)
                    sm.getExtendesStateVariable();
    LightControlAction lca = null;
    LightControl lc = LightControl.getInstance();
    WindowShadeInfo wsi = null;
    if(LightSensorLogic.LUXMAX < lsi.getLux()){
      if(lc.getLightSystem().getLightSystem().get(0).getIntensity()
          <= 0){
        sm.setState(new LightOK());
      }else{
        lca = LightControlAction.DECREASELIGHT;
      }
    }else if(LightSensorLogic.LUXMIN > lsi.getLux()){
      if(lc.getLightSystem().getLightSystem().get(0).getIntensity()
          >= LightControl.MAXINTENSITY){
```

```
20          wsi = new WindowShadeInfo(WindowShadeType.HALF);
21          sm.getDataPublisher().notifySubscriber(new Event(wsi));
22          sm.setState(new HalfDarkened());
23        }else{
24          lca = LightControlAction.INCREASELIGHT;
25        }
26      }
27      ControlLuxInfo cli = new ControlLuxInfo(lsi, lca);
28      sm.getDataPublisher().notifySubscriber(
29                      new Event(cli));
30
31    }
32
33    ...
```

Listing 6.14: Zustand - FullDarkened

Die anderen Zustände sind ähnlich wie dieser Zustand aufgebaut. Es wird nur in der Logik unterschieden in welchen Zustand bei welchen Lichtwerten der Automat wechselt.

6.4 Datenvorbereitungsagent

Gesichtserkennung Provider

Um zu überprüfen welche Personen befugt sind das System zu bedienen wird eine Gesichtswiedererkennung durchgeführt. Für die Gesichtswiedererkennung werden sogenannte Eigengesichter (Eigenface) benutzt.

Eigenface

In der Eigenface-Methode von Turk und Pentland (Turk und Pentland [1991]) werden lokale und globale Merkmale eines Bildes betrachtet. Diese werden heran gezogen um wichtige Informationen zur Identifizierung eines Gesichts zu liefern. Im Gegensatz zu anderen Verfahren, die Dinge wie Abstände zu Teilen des Gesichts als Merkmale klassifizieren, betrachtet die Eigenface-Methode wie bereits gesagt das Gesicht als Ganzes. Ziel ist es Informationen zu finden, die hoch charakteristisch sind, bei einer reduzierten Darstellung.

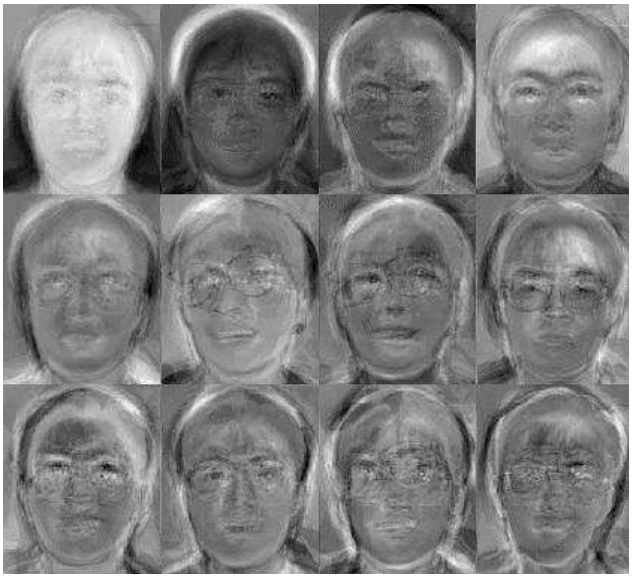

Abbildung 6.5: Eigengesichter
Quelle: PICEF [2013]

Für die Wiedererkennung werden von Trainingsdaten Eigenfaces erstellt. Die Eigenfaces spannen einen Gesichtsraum auf. Um ein unbekanntes Gesicht wiederzuerkennen wird es in den Gesichtstraum projiziert. Es wird ein Gewichtsvektor des unbekannten Bildes mit den Eigenfaces gebildet. Die Projektion wird ebenfalls mit den normalen Trainingsbildern gemacht. Die Gewichtsfaktoren des unbekannten Gesichts werden mit den Gewichtsfaktoren der Trainingsgesichter verglichen. Es kann nun eine Aussage getroffen werden, wo im Gesichtsraum das unbekannte Gesicht liegt und dementsprechend kann gesagt werden zu wem das Gesicht gehört. Weitere Erklärungen können im folgenden Paper Martinovsky und Wagner [2013] nachgelesen werden. Die Methoden für die Gesichtswiedererkennung werden gemäß des Zustandsautomaten aufgerufen.

```
1  public class FaceRecognitionProvider{
2  ...
3    public void faceDetection(int timeout, int period){
4      this.faceRecognizer = new FaceRecognizer(
5                                  timeout, period, this.amc);
6      // start thread to tracking face
7      new Thread(this.faceRecognizer.getFaceRecogPanel()).start();
8     }
9
10   public void faceRecognition(){
```

```
11      this.faceRecognizer.recognize();
12    }
13
14    public void destroy(){
15      this.faceRecognizer.destroyRecognizer();
16    }
17
18    ...
```

Listing 6.15: FaceRecognitionProvider

Da die Benutzung der Kamera für die Gesichtserkennung nicht ständig, sondern nur bei Bedarf benötigt wird, wird der Thread dementsprechend auch bei Bedarf gestartet. Nach einer Wiedererkennung kann, je nach Ausgang, das Objekt zerstört werden und die Kamera auf inaktiv gesetzt werden. Dies spart Ressourcen.
Der Code für die Umsetzung der Eigenface-Methode wurde von Andrew Davison übernommen (Davison [2013]).

eyebox2 Provider

Der Kontext Aufmerksamkeit wird zum Teil durch die xuuk eyebox2 erfasst. Es wird analysiert, ob der Benutzer in Richtung TV schaut oder nicht. Dazu sind einige Vorbereitungen zu treffen. Der Aufbau der eyebox2 spielt eine Rolle für die korrekte Erfassung der Augen.

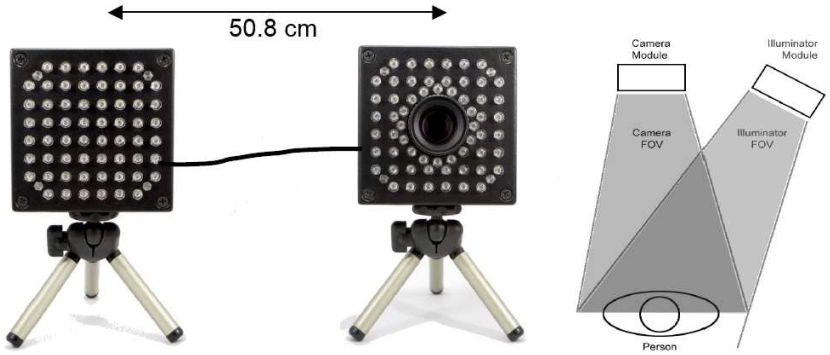

Abbildung 6.6: Aufbau der xuuk eyebox2
Quelle: xuuk [2007]

In Abbildung 6.6 sieht man den Aufbau der eyebox2. Illuminator- (links) und Kamera-Einheit (rechts) müssen einen Abstand von ca. 50.8 cm haben und die Infrarotstrahlen beider Einheiten müssen sich, wie in der Abbildung, überschneiden. Abhängig von der Kulisse sollte sich die Person in einem Abstand zwischen 3.05 m und 9,13 m befinden. Der Aufbau im „Living Place“

ist so realisiert, dass die Person zwischen 3 bis 5 Meter entfernt sitzen kann.
Mit einer mit gelieferte Software können Aufzeichnungen der eyebox2 angesehen werden und zusätzliche Informationen werden aufgezeigt.

Abbildung 6.7: Erkennungssoftware - Gesicht erkannt, Augen schauen Richtung eyebox2

In der Software (6.7) werden Rechtecke um und auf das Gesicht gezeichnet. Jedes Gesicht bekommt eine eindeutige ID, die jedoch aus Datenschutzgründen anders sein kann wenn die Person aus dem Kamerabereich verschwindet und wieder auftaucht. Augen die auf die eyebox2 schauen werden grün eingezeichnet, die darauf nicht schauen orange.
Die Daten, die die eyebox2 liefert, werden über das Netzwerkprotokoll Telnet übermittelt. Die Sendung der Daten erfolgt über ein vordefiniertes Format. Es werden ASCII-Zeichen, also menschenlesbare Nachrichten versendet. Die Daten sind, wenn die Software gestartet wird, auf Port 5015 empfangsbereit.

MESSAGE_ ID	D1	D2	D3	D4	End
%	{1,...,65535}	{1,0}	(YYYY-MM-DD)	(HH:MM:ss:mmm)	CRLF

Tabelle 6.2: Format der eyebox2 Nachrichten

In der Tabelle 6.2 sieht man das Format von eyebox2 Nachrichten. Die MESSAGE_ ID steht für 1: Gesichtsposition, 2: Augenposition, 5: Augen entdeckt, 8: Gesicht entdeckt, 9: Augen gucken und 3,4,6,7 sind reserviert und haben keine Bedeutung. D1 ist die Identifikationsnummer einer Person. D2 steht für 1: erkannt und 0: nicht erkannt. D3 und D4 sind Zeitstempel. Jede Nachricht wird mit einem Wagenrücklauf und Zeilenvorschub beendet.
Um die Nachrichten korrekt interpretiert zu können, wird ein Telnetparser eingesetzt.

```
public class EyeboxDataParser {

   protected EyeboxInfo parse(String data){
        EyeboxInfo info = new EyeboxInfo();
        String dateStr = "";
        Date date = null;
        String msg = data;

        String[] splitResult = msg.split("_");

        info.setMsgID(splitResult[0]);
        info.setPersonID(splitResult[1]);
        info.setStatus(splitResult[2]);

        dateStr = splitResult[3] + "_" + splitResult[4];

        SimpleDateFormat sdfToDate =
            new SimpleDateFormat("yyyy-MM-dd_HH:mm:ss:SSS");
        try {
                date = sdfToDate.parse(dateStr);
        } catch (ParseException e) {
                e.printStackTrace();
        }
        info.setDate(date);

        return info;
    }

  ...
```

Listing 6.16: EyeboxDataParser

Da jeder Nachrichtenteil durch ein Leerzeichen getrennt ist, wird das Parsen erleichtert (6.16). Es herrschen Unregelmäßigkeiten in der Datenbereitstellung der eyebox2. Beispielsweise wird nach einer Erkennung eines Gesicht unmittelbar danach eine Nachricht übermittelt, dass das Gesicht nicht mehr zu sehen ist. Das selbe Verhalten ist auch bei der Erkennung, ob eine Person in Richtung eyebox2 guckt oder nicht, zu beobachten. Es fehlen Toleranzen. Aus diesem Grund wird ein Timer gestartet, der beim längeren Nicht-Gucken bzw. wenn ein Gesicht nicht erkannt wird, ausläuft und daraus geschlussfolgert werden kann, dass tatsächlich eine Person nicht auf die Kamera guckt bzw. nicht mehr da ist.

Timeraktualisierung bei Erkennung

Da die Software der eyebox2 wie bereits beschrieben Unregelmäßigkeiten in der Datenbereitstellung aufweist, werden Zeitstempel für die korrekte Interpretation der Daten eingesetzt. Es gibt jeweils eine long-Variable für die Erkennung des Gesichts und ob Augenkontakt vorherrscht. Diese Variable stellt ein Zeitstempel dar, in der zuletzt eine Erkennung stattgefunden hat. Es findet also ständig eine Aktualisierung des Zeitstempels statt bei Erkennung. Mittels einer Variable kann nun eine Zeitspanne festgelegt werden, für wie lange man keine Erkennung festgestellt hat, um so eine Entscheidung zu treffen, ja eine Erkennung hat statt gefunden oder nein keine Person wurde erkannt oder schaut nicht in Richtung der eyebox2. Bei einer gültigen Veränderung wird diese ans ActiveMQ gesendet.

```
1  public class EyeboxProvider extends Thread {
2
3    /**
4     * @brief changes and checks the given values of modification
5     * @param info the EyeboxInformation
6     * @param currentTime of facedetect or eyelooking
7     * @param kindOfDetect of facedetect or eyeslooking
8     * @param status true for send the change to activemq, false for not
9     */
10   private void changeValue(EyeboxInfo info, PointerWrap<Long>
11   currentTime, PointerWrap<Boolean> kindOfDetect,
12                                PointerWrap<Boolean> status){
13     long time = System.currentTimeMillis();
14     if(currentTime.object + intervalOfNoDetection < time ||
15         (info.getStatus() == DETECT && !kindOfDetect.object)){
16       if(info.getStatus() == DETECT){
17         if(kindOfDetect.object){
18           status.object = false;
19         }else{
20           status.object = true;
21         }
22         kindOfDetect.object = true;
23       }else{
24         if(kindOfDetect.object){
25           status.object = true;
26         }else{
27           status.object = false;
28         }
```

```
29          kindOfDetect.object = false;
30        }
31        currentTime.object = time;
32      }
33    }
34    ...
```

Listing 6.17: EyeBoxProvider

Sitz-/Liegeprofile Provider

Diese Komponente erfasst den Kontext Sitz-/Liegeprofil. Wie bereits in 4.3.4 gezeigt gibt es eine Anwendung welches Daten der kapazitiven Sensoren über eine Server-Client-Schnittstelle liefern kann. Es wird von der ASP.NET SignalR Webapplikation in fixen Zeitintervallen Daten über die Zustände der Sensoren der Couch geliefert. Es ist möglich, dass bei Nichtänderung die Daten ebenfalls geschickt werden, da dies serverseitig so implementiert wurde. Es wäre überflüssige (Daten)Last wenn diese Daten ans ActiveMQ geschickt werden, da keine Änderungen statt finden. Im Datenvorbereitungsagenten werden deshalb die Daten der Webapplikation entgegengenommen, die Änderungen gespeichert und ans ActiveMQ geschickt.

```
1  public class SofaController : System.Web.HttpApplication {
2    protected void Application_Start() {
3     connection.On<List<SofaViewModel>>("UpdateSofaView", (data) =>
4     {
5       list = data;
6       sofa.initSofa(list);
7       jsonMsgSofa = sofa.checkDiff(list)
8       System.Diagnostics.Debug.Write(jsonMsgSofa);
9       if (!jsonMsgSofa.Equals(""))
10      {
11         server.sendMsgToClient(jsonMsgSofa);
12      }
13    });
14    ...
```

Listing 6.18: SofaController

Um die Daten dieser Anwendung erhalten zu können, wird ein SignalR-Client benötigt, der sich mit dem SignalR-Server verbindet. Der SignalR-Server liefert mit der Methode „UpdateSofaView“ die Daten der Couch. Treffen neue Daten ein so reagiert der obige Codefragment 6.18

sensitiv darauf und verarbeitet die Daten. Mit der Methode „checkDiff“ werden tatsächliche Veränderungen festgestellt und erst dann werde diese ans ActiveMQ weitergeschickt.

```
public class Sofa{
 public string checkDiff(List<SofaViewModel> newSofa) {
  List<SofaViewModel> sofa = new List<SofaViewModel>();
  for (int i = 0; i < this.sofa.Count; i++) {
   if (this.sofa[i].SofaID == newSofa[i].SofaID) {
     if(!this.sofa[i].Color.Equals(newSofa[i].Color)){
       sofa.Add(newSofa[i]);
       this.sofa[i].Color = newSofa[i].Color;
     }
   }else {
     System.Diagnostics.Debug.WriteLine("->Error must be the same ID");
   }
  }
  SofaJsonWrapper sWrapper = new SofaJsonWrapper {
    sofa = sofa
  };

  if (sofa.Count == 0) {
    return "";
  }
  return JsonConvert.SerializeObject(sWrapper, Formatting.Indented);
 }
  ...
```

Listing 6.19: Sofa

6.5 Raspberry Pi

Abbildung 6.8: Raspberry Pi

In der Abbildung 6.8 ist der Raspberry Pi abgebildet. Er ist ein kreditkartengroßer Einplatinen-Computer und wurde von der Rasperry Pi Foundation entwickelt. Auf der Seite wird damit geworben, dass der kleine Rechner in der Lage ist, Dinge zu tun, die auch ein Desktop PC machen kann (RaspberryPi [2013]): Tabellen erstellen, Testverarbeitung und Spiele spielen. Außerdem kann er hochauflösende Videos abspielen. Der Rasperry Pi wurde mit dem Ziel entwickelt, eine günstige Entwicklungsplattform und einen günstigen Computer zur Verfügung zu stellen. Ein weiteres Ziel ist es, dass Kinder (in Schulen) den Kontakt mit der Programmierung und Computer im Allgemeinen früh bekommen.
In dieser Arbeit soll der Raspberry Pi als eine prototypische Komponente für die Bestimmung der Lichtverhältnisse und die Steuerung des Fernsehens (als Mess-/Steuerungsagenten) eingesetzt werden. Es wird auf die Vor- und Nachteile des Raspberry Pi's, bezüglich der geforderten Anforderungen in Kapitel 3.4.3 eingegangen. Die komplette Installation des Raspberry Pi's, sowie die Anbindung der erforderlichen Bibliotheken für die Steuerung des Fernsehens über HDMI-CEC und die Benutzung der GPIO wird auch erläutert. Um dies alles genauer zu erleuchten, werden Codefragmente präsentiert und erklärt.

.

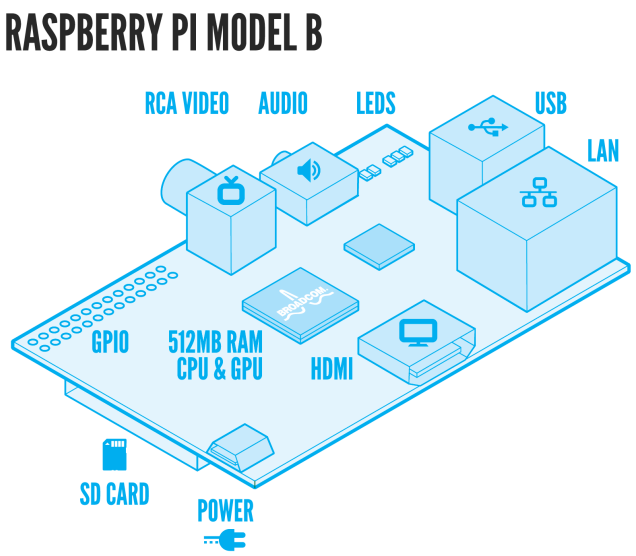

Abbildung 6.9: Raspberry Pi - Modell B
Quelle: (RaspberryPi [2013])

Technische Daten:

	Modell B
Größe:	85,50 mm x 53,98 mm 17 mm
SoC:	Broadcom BCM2835
CPU:	ARM1176JZF-S (700 MHz)
GPU:	Broadcom VideoCore IV
Arbeitsspeicher:	512 MB
Videoausgabe:	FBAS, HDMI
Tonausgabe:	3,5 mm-Klinkenstecker (analog), HDMI (digital)
Netzwerk:	10/100 MBit Ethernet-Controller
Schnittstellen:	16 GPIO-Pins, UART, SPI, I^2C

Quelle: RaspberryPi [2013]

Tabelle 6.3: Daten des Raspberry Pi's

In der Tabelle 6.3 sieht man die technischen Daten des Rasperry Pi's Modell B. Dieses Modell ist der Nachfolger des Modell A. Es wurden in diesem Modell einige Hardware aufgerüstet, z.B. von 256 MB auf 512 MB Arbeitsspeicher, sonst hat sich grundlegend nicht viel verändert. Aus diesen Gründen ist es nicht notwendig das Vorgängermodell vorzustellen. Als Betriebssystem wird eine Abwandlung einer Linux-Distribution (Debian), Raspbian „wheezy“ benutzt.

Das Betriebssystem wird auf einer SDHC-Karte installiert und kann direkt am Raspberry Pi angesteckt und benutzt werden (RaspberryPi [2013]).

6.5.1 Anforderungsanalyse

Die Entscheidung über die Wahl des Raspberry Pi's soll hier diskutiert werden. Es wird Bezug genommen zu den gewünschten Anforderungen in Kapitel 3.4.3. Inwieweit die Hardwareanforderungen vom Rasperry Pi erfüllt werden und gegebenenfalls welche Aspekte beachtet werden müssen um die Anforderungen zu erfüllen.

- In den Anforderungen wurde genannt, dass die Hardware mehrere Aufgaben erfüllen sollte (in diesem Fall Analyse der Lichtsituation und die TV-Steuerung). Diese Anforderung wird vom Raspberry Pi erfüllt, denn es bietet verschiedene Schnittstellen (6.3) für die Ansteuerung verschiedener Komponenten. Der Lichtsensor wird an die GPIO angebracht und die Informationen werden in einem Programm vom Raspberry Pi ausgewertet und an das System geschickt. Vom System erhält der der Raspberry Pi Befehle für die Ansteuerung des TV. Diese Befehle werden durch die Schnittstelle HDMI an den Fernseher weitergeschickt.

- Dank der Größe des Raspberry Pi's (85,50 mm x 53,98 mm 17 mm) kann er unauffällig in der Nähe des TV angebracht werden. Um den Raspberry Pi noch unauffälliger zu gestalten, wird er in einem Case (Schutzhülle) im schwachen Blau getan. So kann das Prinzip des Ubiquitous Computing befolgt werden. Um die Hardwarekomponente gänzlich verschwinden zu lassen, kann ein Platz in der Wand reserviert werden, um es in die Wand zu platzieren. Dies ist jedoch keine Vorstellung fürs „Living Place", da es für Forschungszwecke eingesetzt wird und nicht für den privaten Haushalt.

- Die eingesetzte Hardware (Raspberry Pi Modell B) kostet laut Herstellerseite (RaspberryPi [2013]) 35 Dollar (ca. 27,16 €). Dieser Preis ist verhältnismäßig zu einem Arduino oder PC gering, da mit der gelieferten Hardware ein kleiner Desktop-PC realisiert werden kann, welcher zusätzlich Schnittstellen wie GPIO, UART, SPI und I^2C hat.

- Der Raspberry Pi hat viele Schnittstellen (6.3), die es ermöglichen neue Komponenten anzubinden und somit den Raspberry Pi zu erweitern. Mit den 16 GPIO-Pins können verschiedene Arten von Sensoren und andere Hardwarekomponenten ausgelesen und angesteuert werden, dies bietet daher eine große Bandbreite von unterschiedlichen Projekten. Durch UART, SPI und I^2C können verschiedene Arten von Datenübertragung

ermöglicht werden, welches somit die Erweiterbarkeit steigt. Man könnte Beispielsweise einen Flash ansteuern. Mit der Schnittstelle Ethernet ist es möglich ein Netzwerk aufzubauen und man hat selbstverständlich Zugang zum Internet. Dadurch eröffnet sich ebenfalls ein breites Feld an möglichen Anwendungen. Daten können vom Internet zugänglich gemacht werden oder Austausch von anderen Netzteilnehmern mit den Raspberry Pi (beispielsweise durch ActiveMQ).

- Einer der Anforderungen ist, dass die Einarbeitung in die Hardware keine große Hürde stellen darf. Da der Raspberry Pi, wie ein normaler Desktop-PC gehandhabt werden kann, ist der Aufwand für die Einarbeitung minimal. Es ist nur Anfangs erforderlich, ein Betriebssystem, anhand einer Leitung, die auf der Webseite zu finden ist, zu installieren. Für die Anbringung von weiteren Sensoren und Komponenten sind die erforderlichen Anleitungen zu lesen.

Zusammenfassend kann man sagen, dass der Raspberry Pi ein Low-Power System ist, welches klein und kompakt ist. Es kann für die Entwicklung von schnellen Prototypen herangezogen werden, da es ausreichende Schnittstellen besitzt. Es ist eine Komponente, die von der Funktionalität oberhalb eines Arduino steht, da man gleichzeitig ein Betriebssystem zur Verfügung hat und programmierbare Schnittstellen, jedoch unterhalb eines kompletten PC steht. Der PC hat mehr Rechenleistung, jedoch einen höheren Stromverbrauch. Für diesen Einsatz ist, jedoch der Raspberry Pi geeigneter, da soviel Rechenleistung nicht benötigt wird und die Größe bei einem PC den Faktor ubiquitär behindert würde.

6.5.2 Integration in die Laborumgebung als Mess-/Steuerungsagent

In diesem Abschnitt soll die Integration des Rasperry Pi in die Laborumgebung als Mess-/Steuerungsagent beschrieben werden. Dabei wird zunächst auf die Installation des Raspberry Pi's eingegangen und anschließend wird die Integration der einzelnen Komponenten am Raspberry Pi vorgestellt.

Installation des Raspberry Pi's

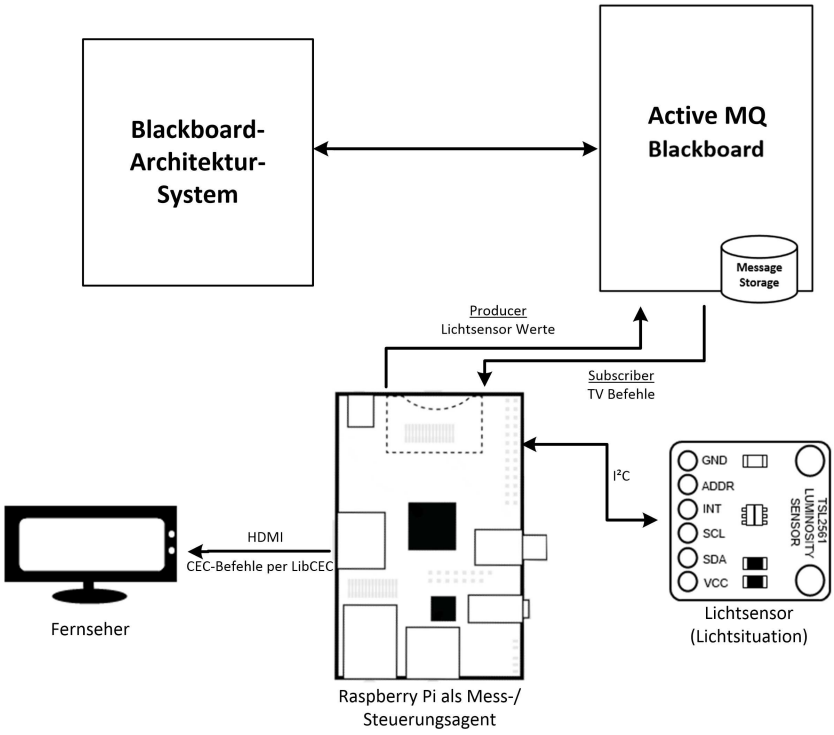

Abbildung 6.10: Raspberry Pi - Aufbau

In der Abbildung 6.10 ist der Aufbau des Mess-/Steueragenten zu sehen. Auf die einzelnen Komponenten wird später eingegangen. Um den Raspberry Pi benutzen zu können wird ein Betriebssystem benötigt. Im Internet werden zahlreiche Betriebssysteme vorgestellt. Auf der offiziellen Internetseite wird jedoch Raspbian empfohlen (RaspberryPi [2013]). Eine Installationsanleitung kann man ebenfalls auf der Internetseite finden.
Nach der Installation kann man sich mit dem Benutzername „pi" und dem Passwort „raspberry" einloggen. Die Programmiersprache, die für die Umsetzung des Mess-/Steuerungsagenten verwendet wird ist C++.

Integration ActiveMQ

Um als ein Agent in der Blackboard-Architektur agieren zu können, wird ActiveMQ benötigt. Die Apache Software Foundation stellt für Anwendungen, die mit C++ geschrieben wurden sind, Includes und Binaries zur Verfügung (Foundation [2013b]). Es muss, für eine erfolgreiche Installation, die jeweilige „ReadMe" Anleitung durchgelesen werden.
Subscriber:

```
class Subscriber : public ExceptionListener,
                           public MessageListener {
public:
    Subscriber(const string& topicName, const string&
            host, const string& port, bool sessionTransacted) :

        connection(NULL),
        session(NULL),
        destination(NULL),
        consumer(NULL),
        sessionTransacted(sessionTransacted),
        host(host),
        port(port),
        topicName(topicName)
         {
    }

    virtual ~Subscriber();

    void close();

    Session* getSession();

    bool isSessionTransacted();

    void init();

    virtual void onMessage(const Message* message) = 0;

    // If something bad happens you see it here as this class
    // is also been registered as an ExceptionListener
    // with the connection.
    virtual void onException(const CMSException& ex AMQCPP_UNUSED);

private:
    void cleanup();
  ...
```

Listing 6.20: C++ - Subscriber

Der Subscriber wird von einer Klasse geerbt und die „onMessage“- Methode muss dabei überschrieben werden. Bei eintreffenden Nachrichten wird dann die „onMessage“- Methode aufgerufen.

Publisher:

```
class Publisher {
public:

    Publisher(const string& topicName, const string& host,
              const string& port, bool sessionTransacted);

    virtual ~Publisher();

    void close(void);

    void init(void);

    void publish(const string& msg);

private:

    void cleanup(void);
};
```

Listing 6.21: C++ - Publisher

Will man eine Nachricht ans ActiveMQ senden, so kann man einen Publisher initialisieren und mit der Methode „publish“ eine Nachricht an den entsprechenden Topic senden.

Integration Sensor

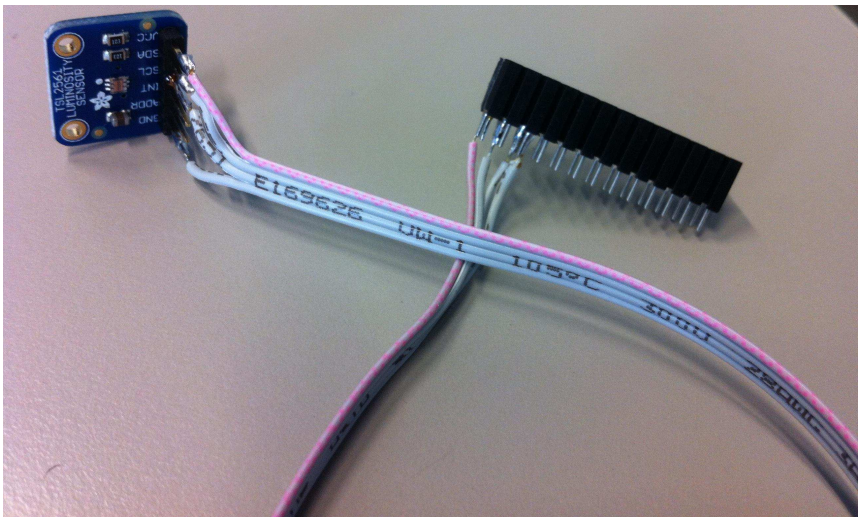

Abbildung 6.11: Lichtsensor

Der Lichtsensor TSL2561 (6.11) wird verwendet, um die Lichtstärke (in Lux) des Raumes zu messen und an den Logiksubagenten zu schicken. Der Lichtsensor liefert Werte von 0.1 bis 40.000+ Lux (Texas Advanced Optoelectronic Solutions Inc. [2005]), dies ist ausreichend für die Betrachtung der Lichtsituation, da Werte im Wohnbereich zwischen 300 und 400 liegen dürfen. Es ist mit dem Sensor möglich verschiedene Präzisionsstufen einzustellen, dies wird erreicht indem die Messzeit einstellbar ist. Die Werte werden für die Beurteilung des Kontextes Lichtsituation herangezogen. Der Lichtsensor wird mit den GPIOs am Raspberry Pi verbunden. Das Senden der Kommandos an den Lichtsensor und die Datenübertragung verläuft über den I^2C Bus des Raspberry Pi's.
Der Sensor soll eine Repräsentant sein für das Umgebungslicht und deshalb wird der Sensor mittig im Raum gelegt, so dass ein großteil des Raumes erfassen kann.

WiringPi und I^2C:

Für die Benutzung der GPIO wird das Projekt Wiring Pi (Gordon [2013]) benutzt. Es wird ebenfalls eine Schnittstelle für die Benutzung von I^2C bereitgestellt. Es wird direkt die I^2C des Raspberry Pi's benutzt.

I^2C:

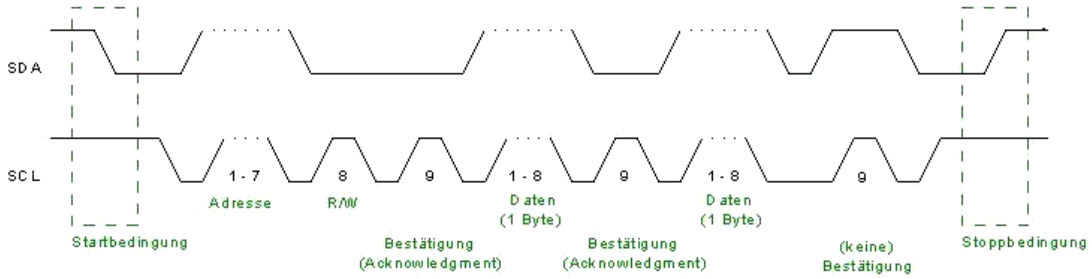

Abbildung 6.12: Datenübertragung eines I^{2}C-Bus
Quelle: (RN-Wissen [2013])

I^{2}C für Inter-Integrated Circuit ist ein von Philips entwickelter serieller Datenbus. Das Master-Slave Prinzip wird mit dem I^{2}C realisiert, sprich das Initiieren des Datenverkehrs kann nur vom Master erfolgen. Über eine Adresse können die angeschlossenen Slaves angesprochen werden. Es folgen Steuerungsbefehle vom Master an den Slave. Je nach Steuerungsbefehl können Daten an den Slave geschickt werden oder vom Slave empfangen werden. Typischerweise werden über die SDA-Leitung die Daten übertragen, die SCL-Leitung bestimmt die Taktfrequenz (NXP [2012]).

WiringPi:

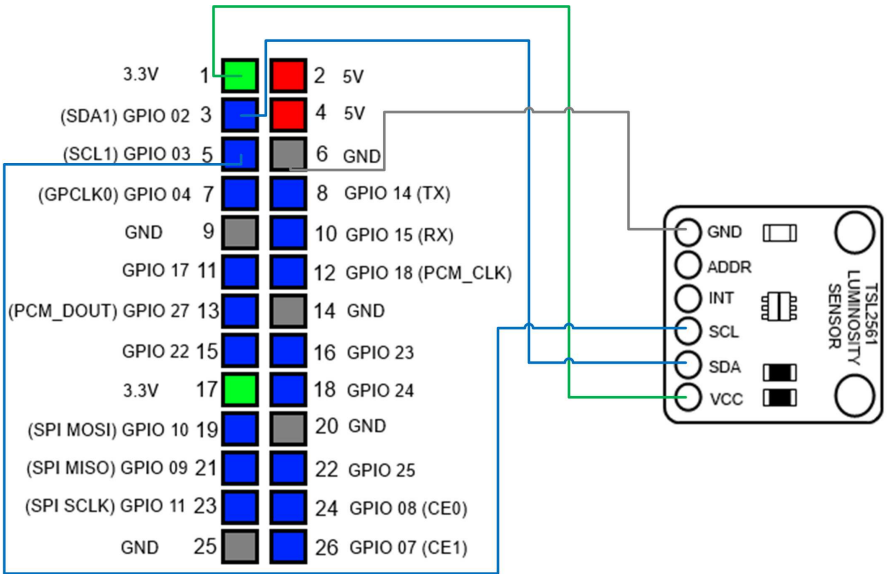

Abbildung 6.13: Lichtsensor verbunden mit GPIO

In Abbildung 6.13 sieht man den Verbindungsplan des Lichtsensor mit den GPIO des Raspberry PI's. WiringPi eine Bibliothek für eine einfache Benutzung der GPIO des Raspberry Pi's, geschrieben in C und kann in Anwendungen, die mit C und C++ geschrieben wurden sind, benutzt werden. Sie bietet Schnittstellen für die Benutzung von I^2C, SPI und UART. Für die Benutzung des Lichtsensor wird die I^2C-Schnittstelle benutzt. Folgende Methoden stehen zur Verfügung:

```
int wiringPiI2CSetup (int devId);

int wiringPiI2CRead (int fd);

int wiringPiI2CWrite (int fd, int data);

int wiringPiI2CWriteReg8 (int fd, int reg, int data);
int wiringPiI2CWriteReg16 (int fd, int reg, int data);

int wiringPiI2CReadReg8 (int fd, int reg);
int wiringPiI2CReadReg16 (int fd, int reg);
```

Listing 6.22: WiringPiI2C Methoden

Es wird mit „wiringPiI2CSetup" ein File descriptor zurück gegeben, der den I^2C-Bus beschreibt, dieser wird dann in den jeweiligen Methoden zum lesen und schreiben verwendet.

Senden der Lichtwerte:

```
void LightSensor::run(void){
  string text = "";
  Json::Value root;
  string lux = "";
  ostringstream ss;
  // in seconds
  long delay = 10;

  while(!running) {
    if(!initPhase){
      //senden der Lichtwerte
      measure();
      ss << getDataFromSensor();
      lux = ss.str();
      jsonParser.writeValueIn(lux,"lux",root);
      text = jsonParser.getJsonMsg(root);
```

```
17        sendMsg(text);
18        sleep(delay);
19      }
20    }
21  }
```

Listing 6.23: LightSensor - run-Methode

Die Lichtwerte werden alle 10 Sekunden gemessen und an den Logiksubagenten per Blackboard gesendet. Dabei muss die Nachricht als JSON-Nachricht umgewandelt werden.

Integration CEC

Um den Fernseher zu steuern wird das Protokoll CEC benutzt. Der Raspberry Pi erhält von Reaktor-/Präsentationssubagenten die Befehle für die Steuerung. Die Befehle werden vom Raspberry Pi direkt an den Fernseher gesendet.
CEC: CEC für Consumer Electronics Control ist eine Schnittstelle für Ansteuerungsfunktionen von elektronischen Konsumergeräten. CEC benutzt einen seriellen, einadrigen Datenbus für die Kommunikation. Die Steuerungssignale werden mit einer Datenübertragungsrate von 400bit/s übertragen. In HDMI-Verbindungen ist das standardmäßig Pin 14. In diesem System wird der Fernseher Samsung UE46C8790 verwendet. Die HDMI-Schnittstelle wird in Samsung Produkten als Anynet+ bezeichnet. Damit man die Funktionalität automatisches Ein- und Ausschalten des TV erhalten will muss man dies einmalig im TV Menü unter Anynet+ (HDMI-CEC) (Testgerät Samsung) einstellen.
libCEC: Für die Verwendung des CEC-Protokoll wird die Bibliothek libCEC (Pulse-Eight [2013]) benutzt. Für die Benutzung der Bibliothek ist es eigentlich erforderlich eine Hardwarekomponente zu kaufen, jedoch kann durch die Benutzung eines libCEC-Client auf einem PC (dem Raspberry Pi) dies umgegangen werden. Durch diese Bibliothek muss man sich nicht um den Kommunikationsaufbau (Handshake) des CEC-Protokoll oder sonstigen Kommunikationsoverhead kümmern.

```
1 void CECOperator::getMethod(const string& method){
2
3   string action = method;
4   bool success = true;
5
6   if(method.compare("ON") == 0){
7     success = switchTVON();
8   }else if (method.compare("OFF") == 0){
```

```
9       success = switchTVOFF();
10    }else if (method.compare("MUTE") == 0){
11      success = muteTV();
12    }else if (method.compare("UNMUTE") == 0){
13      success = unmuteTV();
14    }
15
16    if(!success){
17      action = "error";
18    }
19    sendMsg(action);
20
21  }
22
23  bool CECOperator::muteTV(void){
24      cout << ">>Mute" << endl;
25      return libCEC.deviceMute();
26  }
27
28  bool CECOperator::switchTVOFF(void){
29      cout << ">>Switch_off" << endl;
30      return libCEC.deviceOFF();
31  }
32
33  bool CECOperator::switchTVON(void){
34      cout << ">>Switch_on" << endl;
35      return libCEC.deviceON();
36  }
37
38  bool CECOperator::unmuteTV(void){
39      cout << ">>Unmute" << endl;
40      return libCEC.deviceUnmute();
41  }
```

Listing 6.24: CECoperator

Man erhält vom Reaktor-/Präsentationssubagenten die Befehle für die Steuerung des TV über das Blackboard.

6.6 Regressiontests

Um die Funktionsfähigkeit der einzelnen Komponenten zu überprüfen, kann vor Einsatz des Systems sogenannte Regressiontests durchgeführt werden. Bei Regressiontest werden alle oder eine Teilmenge aller Testfälle wiederholt, um bei Modifikationen von bereits getestet Komponenten Fehler zu entdecken. In der Regel entstehen Fehler bei Korrekturen, Pflege oder Änderung der Software. Durch die große Anzahl der Testfälle werden Regressiontest häufig automatisiert. Es werden Soll- und Ist-Verhalten miteinander verglichen und entsprechende Warnungen werden geliefert (Seidl u. a. [2011]).
Für die Regressiontests in diesem System wird JUnit[1] eingesetzt. Die Testfälle werden aufgezeigt.

```
1 public class RegressionTest {
2   @BeforeClass
3   public static void initRegressionTest(){
```

Listing 6.25: Regressiontest - initRegressionTest

Die benötigten Komponenten werden bevor die Tests starten initialisiert.

```
1   @Test(timeout=3000)
2   public void activeMQFunctionTest() {
```

Listing 6.26: Regressiontest - activeMQFunctionTest

Es werden die Blackboard (ActiveMQ) Funktionalitäten geprüft, sprich das Publisher-Subscriber-Prinzip. Erfolgt nach einer gewissen Zeit (3 Sekunden) keine Antwort vom Blackboard ist dieser Test fehlgeschlagen.

```
1   @Test
2   public void lightFunctionAllSpots() throws InterruptedException{
```

Listing 6.27: Regressiontest - lightFunctionAllSpots

In diesem Regressiontest (6.27) werden die Lampen im Wohnbereich getestet, da die Lampen keinen Zustand liefern können, muss dieser Test halb automatisiert durchgeführt werden. Es wird jede Lampe nacheinander geprüft, bei korrekter Funktionstüchtigkeit wird dies vom Administrator per „Enter“ bestätigt.

```
1   @Test
2   public void windowShadeFunctionTest() throws InterruptedException{
```

Listing 6.28: Regressiontest - windowShadeFunctionTest

[1] Framework zum Testen von Java-Programmen. Geeignet für automatisierte Einzeltests.

Die Rollos im Wohnbereich werden überprüft. Dies erfordert wie im Lampentest ebenfalls eine Bestätigung des Administrators. Die Rollos werden hoch, halb hoch und nach unten gefahren.

```
1  @Test
2  public void tvControlON(){ ...
3  @Test
4  public void tvControlOFF(){ ...
```

Listing 6.29: Regressiontest - tvControl

Die Funktionalität des Fernseher wird im obigen Codeabschnitt (6.29) überprüft. Die korrekte Funktionsweise wird vom Mess-/Steuerungsagenten bestätigt oder auch nicht.

```
1  @Test
2  public void lightMeasure(){
```

Listing 6.30: Regressiontest - lightMeasure

Der Lichtsensor wird in diesem Test (6.30) aufgefordert die Lichtstärke zu messen. Funktioniert der Lichtsensor nicht oder liefert er ungültige Werte, so schickt der Mess-/Steuerungsagent einen negativen Wert.

```
1  @Test
2  public void faceRecognitionTest(){
```

Listing 6.31: Regressiontest - faceRecognitionTest

Die Gesichtswiedererkennung wird in diesem Test (6.31) überprüft. Dabei muss eine Person, die in der Datenbank vorhanden ist vor der Kamera stehen. Dieser Test ist ebenfalls halb automatisiert, da die Positionierung der Person vor der Kamera bestätigt werden muss.

```
1  @Test
2  public void couchSittingLyingNoContact() throws InterruptedException{
```

Listing 6.32: Regressiontest - couchSittingLyingNoContact

Die Couch wird im obigen Code (6.32) getestet. Die Testperson muss sich zunächst auf die Couch sitzen, dann liegen und zum Schluß wieder aufstehen. Dieser Test ist ebenfalls nicht voll automatisiert.

```
1  @Test
2  public void eyeboxFunctionTest(){
```

Listing 6.33: Regressiontest - eyeboxFunctionTest

In diesem Test (6.33) wird die xuuk eyebox2 getestet. Die Testperson wird aufgefordert in die Kamera zu gucken und dann wieder weg zu gucken, dies muss vorher bestätigt werden.

6.7 Fazit

In diesem Kapitel wurde die Realisierung des System auf Grundlage des Design und den Vorüberlegung in der Analyse vorgestellt. Es wurde in der Realisierung versucht die Aussage „Ich glaube zu wissen was der Nutzer für eine Intention hat und wie sein Zustand ist“ zu behandeln. Eine exaktes Rekonstruieren und Deuten des Vorhabens konnte im Rahmen dieser Arbeit nicht behandelt werden, da sogenannte Usability-Tests und Verhaltenstests des Benutzers dafür benötigt werden. Es sollte ein prototypisches System erstellt werden, auf dem man aufbauen kann. Die System wurde größtenteils realisiert, jedoch traten einige Probleme auf. Zunächst soll reflektierend auf die drei Punkte von Dey und Abowd (Dey und Abowd [2013]), die ein Context aware computing System klassifizieren, in 3.1 eingegangen werden. Die Architektur des System ist so aufgebaut, dass man die drei Punkte:

1. Präsentation von Informationen und Dienstleistungen für ein Benutzer
2. Automatische Ausführung eines Dienstes und
3. Markierung von Kontext-Informationen für einen späteren Abruf.

erkennen kann. Die Präsentation von Informationen und Dienstleistungen für ein Benutzer werden vom Reaktor-/Präsentationssubagenten übernommen. Eingehende Befehle vom Logiksubagenten werden umgesetzt und es findet eine Reaktion statt. Das „Living Place“ verändert seinen Zustand und der Benutzer bekommt dies mit. Automatische Ausführung eines Dienstes wird ebenfalls vom Reaktor-/Präsentationssubagenten übernommen. Der Benutzer muss sich nicht bemühen, damit die Reaktionen im „Living Place“ statt finden. Es ist eine einmalige Konfiguration notwendig, um das System hochzufahren. Die Markierung von Kontext-Informationen für einen späteren Abruf wird vom Datenvorbereitungsagenten und Logiksubagenten übernommen. Daten werden ggf. vom Datenvorbereitungsagenten gehalten, um Zustände der Teilsystem (zum Beispiel das Lampensystem) festzuhalten. Der Logiksubagent macht dies auf einer höheren Ebene und verwaltet die Auswirkungen der Kontexte in einem Zustandsautomaten. So kann gezielt auf Veränderung reagiert werden.

Probleme in der Realisierung

Laborumgebung

In der Laborumgebung traten Problem auf, die die richtige Verwendung des Systems behinderte. In der Konzeption des Systems wurden diese Probleme nicht beachtet. Zwei dieser Probleme werden hier kurz erwähnt.

Lichtumgebung Das Lampensystem im Wohnbereich war nicht komplett funktionstüchtig. Es funktionierten nur Teile des Systems (7 von 16 Lampen). Dies erschwerte das Testen des Systems, da bei einem Input nicht der geforderte bzw. erwünschte Output auftrat. Messwerte konnten nicht korrekt verwertet werden. Die Lichtstärke konnte nicht maximal ausgenutzt werden.

Kamera für Gesichtserkennung Die Kamera für die Gesichtserkennung hatte nicht genügend Auflösung, um ein Gesicht von weiten aus einer Entfernung von 5-6 Meter zu erfassen. Eine Zoom-Funktion hat ebenfalls gefehlt. Dies wurde provisorisch gelöst, in dem die Kamera anders positioniert wurde.

Messen des Lichts

Mit einem Sensor konnte der komplette Wohnbereich nicht erfasst werden. Legt man den Sensor beispielsweise nicht zentriert zwischen zwei Lampen, sondern direkt unter einer Lampe, wird die direkte Lichtstärke der einen Lampe gemessen. Das Licht im Raum wird nicht als Gesamtes betrachtet, sondern es wird nur die eine Lampe erfasst. Eine Verbesserung würde so aussehen, dass man mehrere Lichtsensoren im Wohnbereich verteilt. Man erfasst von jedem Bereich die Lichtstärke und bildet aus den Werten einen Mittelwert oder gewichtet Wert (je nach dem ob ein Sensor näher an einer Lampe ist). So kann eine viel bessere Aussage über die Lichtsituation getroffen werden.

HDMI-CEC

Das CEC-Protokoll liefert nicht die gewünschten Funktionalitäten, um den Fernseher zu steuern. Die direkte Steuerung des Fernseher erweist sich als ziemlich schwer. Mit dem CEC-Protokoll kann nicht die direkte Lautstärke des Fernsehers gesteuert werden. Man benötigt einen AVR[2], der mit dem Fernseher angeschlossen ist. Die Lautstärke des AVR kann man per CEC-Protokoll steuern.

Erweiterungsmöglichkeiten

Das System kann erweitert werden. Einige dieser Erweiterung werden kurz genannt. Dabei werden Erweiterungen genannt, die zusätzlich wenig Aufwand benötigen.

[2] Audio-Video-Receiver sind Mehrkanal-Hi-Fi-Verstärker. Sie können beispielsweise verschiedene Audio- und Videokanäle auf ihre analoge und digitale Ausgänge schalten.

Kontexterweiterung

Das System könnte mehr Kontexte betrachten, um den Benutzer noch mehr zu unterstützen und ihn ein besseren Komfort zu liefern. Man kann beispielsweise die laufenden TV-Programme (über EPG[3]) als ein Kontext betrachten. Wird im Fernseher beispielsweise ein Sportereignis gezeigt, dann möchte man nicht, dass wenn man kurz wegschaut oder die Couch verlässt, der Fernseher sich sofort aus schaltet, man könnte sich ja etwas zu Trinken geholt haben. Ebenfalls denkbar wäre die Erweiterung des Systems um den Kontext Benutzersteuerung. Der Benutzer könnte per Spracheingabe oder Gesten dem System mitteilen, dass er kurz die Couch verlässt und der TV sich nicht aus schalten soll.

Zeitverhalten

Das System kann bzgl. einer Studie über Zeitverhalten oder Zeitkonditionen untersucht werden. Es werden verschiedene Zeitszenarien formuliert und deren Lösungen und Realisierung können vorgestellt werden. Es könnte beispielsweise untersucht werden, wie viel Toleranz notwendig ist, falls eine Person nicht mehr den TV anschaut, bis der Fernseher sich aus schaltet. Bei der Bestimmung der Lichtsituation können wetterbedingte Lichtverhältnisse sich nur für Sekunden geändert haben, zum Beispiel wird das Licht kurz von einer Wolke unterbrochen, eine sofortige Reaktion ist hierbei nicht erforderlich. Toleranzen sollten hier untersucht werden.

Benutzeroberfläche

Um den Benutzer eine direkte Schnittstelle mit dem System zu liefern, kann eine Benutzeroberfläche benutzt werden. Es kann beispielsweise im Wohnbereich ein Touch-Monitor aufgestellt werden, mit dem der Benutzer verschiedene Einstellungen des Systems beeinflussen kann. Eventuell möchte man nicht, dass das System ständig läuft oder nur zu gewissen Zeitpunkten.

[3] Electronic Program Guide sind elektronisch verbreitete Informationen über das aktuelle Hörfunk- und Fernsehrprogramm.

7 Schluss

7.1 Zusammenfassung

Diese Arbeit stellt einen Beitrag zur Forschung von Smart Home bzw. Smart Home environment dar. Das Gesamtsystem ist eine Entwicklung eines adaptiven reaktiven Systems. Die Ausarbeitungen in dieser Arbeit zeigt einen Prototypen, welcher verschiedene Kontexte aufnimmt und verarbeitet. Es wird versucht nach dem Prinzip des Ubiquitous Computing zu handeln, also das System so unterzubringen, dass der Benutzer davon nichts erfährt. Die Arbeit erfolgt im Rahmen eines Smart Homes dem „Living Place“.
Im dritten Kapitel werden die Anforderungen an das System gestellt und eine Untersuchung der zu behandelnden Kontexte erfolgt. Es werden Szenarien abgebildet, um eine bessere Vorstellung des Anwendungsgebiets zu bekommen. Um die Laborumgebung besser zu verstehen wird sie im vierten Kapitel vorgestellt. Die verwendete Hard- und Software wird ebenfalls präsentiert.
Das vierte Kapitel zeigt das Design des Systems. Die Architektur bestimmt den Aufbau und die Wechselwirkung der Komponenten. Mit dem Systementwurf wird der konkrete Aufbau des Systems beschrieben, der Ablauf der eintreffenden Aktionen des Benutzers und das Verhalten des Systems.
Das fünfte Kapitel beschreibt die Realisierung des System. Es hat das Design als Grundlage für die Realisierung. Es werden die einzelnen Komponenten (Agenten) vorgestellt und deren Inhalte werden anhand von Codefragmenten vorgestellt. Der Raspberry Pi wird näher beschrieben. Er wird als ein Agent vorgestellt und seine Vorteile werden aufgezeigt. Ein Fazit schließt das Kapitel ab. Es werden Erweiterungen und Probleme aufgezeigt.

7.2 Ausblick

Die Arbeit hat gezeigt, dass es möglich ist ein System zu entwerfen, welches das häusliche Leben eines Menschen vereinfacht. Es vereinfacht es nicht nur, sondern es hilft dem Benutzer und wirkt komfortabel. Bei dem Einsatz eines solchen adaptiven Systems tauchen Fragen auf, die zu klären sind. Wie sieht die Akzeptanz des Benutzers aus? Schafft es ein solches System

Anklang bei den Benutzern zu finden. Durch das Prinzip des Ubiquitous Computing kann das System verschwinden, der Benutzer ist umzingelt von Überwachungsapparaten. Wie soll der Benutzer reagieren? Es ist hierbei wichtig, dass er aufklärt wird.
Ein weiterer wichtiger Punkt ist, inwiefern der Benutzer auf das im hintergrundlaufende System aktiv eingreifen kann. Es muss eine Balance zwischen autonom und greifbar bestimmt werden. Diese Untersuchungen könnten beispielsweise in Usability-Labors durchgeführt werden. Es ist zu untersuchen, wie weit sich der Benutzer helfen lassen will oder ob so eine Hilfe nicht als störend empfunden wird, deshalb ist ein Eingreifen wichtig.
Müssen Abläufe eines Menschen vom System beobachtet werden? Daraus könnte das System aktiv Lernen und verstehen, wie, wann und wo der Benutzer Unterstützung braucht oder nicht. Der Einsatz in einem Smart Environment bedarf noch weiterer wissenschaftlicher Analysen, um praktische Ansätze besser zu gestalten.
Wie reagieren Geräte auf Menschen und wie Menschen auf Geräte, die Wechselwirkung muss weiterhin erforscht werden. In einem Smart environment kann dies analysiert werden. In Companion-Systeme sollen kognitive technische Systeme entwickelt werden. Der Nutzer soll vollkommen individuell erfasst und die Funktionalitäten sollen auf ihn abgestimmt werden. Im „Living Place“ haben wir eine Instanz dieser Diskussion, man findet sie in verschiedenen Bereichen des Lebens.

Listings

Literaturverzeichnis

[PICEF 2013] 2013. – URL http://www.csie.nctu.edu.tw/~ljshen/ccl/eigenface.jpg. – Zugriffsdatum: 19.07.2013

[JSON1 2013] *JSON*. 2013. – URL http://www.json.org. – Zugriffsdatum: 18.03.2013

[Duden:Komfort 2013] *Komfort, der*. 2013. – URL http://www.duden.de/rechtschreibung/Komfort. – Zugriffsdatum: 04.05.2013

[Licht:Fernsehen 2013] *Licht zum Fernsehen*. 2013. – URL http://www.licht.de/de/licht-fuer-zuhause/beleuchtungsbeispiele/content/lichtgebaude/privathaus/lichtgebiet/wohnzimmer/lichtscene/licht_zum_fernsehen/. – Zugriffsdatum: 02.05.2013

[DIN5035 2013] *Richtwerte für Beleuchtungsstärke nach DIN 5035*. 2013. – URL http://www.elektro-fachplanung.de/Fachinfo/Planungshilfen/Beleuchtung/Beleuchtungsstarken/body_beleuchtungsstarken.htm. – Zugriffsdatum: 23.06.2013

[10gen 2013] 10gen: *MongoDB*. 2013. – URL http://www.mongodb.org/. – Zugriffsdatum: 19.03.2013

[ASP.NET 2013] ASP.NET: *SignalR*. 2013. – URL http://www.asp.net/signalr. – Zugriffsdatum: 21.05.2013

[Bundesamt 2010] Bundesamt, Statistisches: Mikrozensus - Fragen zur Gesundheit - Körpermaße der Bevölkerung. (2010), Juni, S. 13

[Buxten 1995] Buxten, William: Ubiquitous Media and the Active Office. In: *Nikkei Electronics* 632 (1995), S. 187–195

[Crockford 2006] Crockford, D.: *The application/json Media Type for JavaScript Object Notation (JSON)*. Juli 2006. – URL http://tools.ietf.org/html/rfc4627. – Zugriffsdatum: 18.03.2013

[Dahm 2006] Dahm, Markus: *Grundlagen der Mensch-Computer-Interaktion.* Pearson Studium, 2006

[Davison 2013] Davison, Andrew: *Face Recognition.* July 2013. – URL http://fivedots.coe.psu.ac.th/~ad/jg/nui08/. – Zugriffsdatum: 19.07.2013

[Dey und Abowd 2013] Dey, Anind K. ; Abowd, Gregory D.: *Towards a Better Understanding of Context and Context-Awareness.* 2013. – Undatiert

[Ellenberg u. a. 2011] Ellenberg, Jens ; Karstedt, Bastian ; Voskuhl, Sören ; Luck, Kai von ; Wendholt, Birgit: An Environment for Context-Aware Applications in Smart Homes. In: *INTERNATIONAL CONFERENCE ON INDOOR POSITIONING AND INDOOR NAVIGATION* (2011), September

[Erman u. a. 1980] Erman, Lee ; Hayes-Roth, Frederick ; Lesser, Victor ; Reddy, Raj: The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. In: *Computing Surveys* 2 (1980). – URL ftp://shelob.cs.umass.edu/pub/Erman_Hearsay80.pdf. – Zugriffsdatum: 10.03.2013

[Foundation 2013a] Foundation, The Apache S.: *ActiveMQ.* 2013. – URL http://activemq.apache.org/. – Zugriffsdatum: 18.03.2013

[Foundation 2013b] Foundation, The Apache S.: *ActiveMQ CPP.* 2013. – URL http://activemq.apache.org/. – Zugriffsdatum: 21.07.2013

[Gamma u. a. 1996] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Design patterns : elements of reusable object oriented software.* Addison-Wesley, 1996

[Glasberg und Feldner 2009] Glasberg, Ronald ; Feldner, Nadja: BITKOM AK Digital Home: Leitfaden zur Heimvernetzung. (2009)

[Gordon 2013] Gordon: *WiringPi.* 2013. – URL https://projects.drogon.net/raspberry-pi/wiringpi/. – Zugriffsdatum: 21.07.2013

[Hering 2012] Hering, E. ; Schönfelder, G. (Hrsg.): *Sensoren in Wissenschaft und Technik.* Springer Fachmedien Wiesbaden GmbH, 2012. – 32–42 S

[Koudounas und Iqbal 2013] Koudounas, Vasilis ; Iqbal, Omar: *MOBILE COMPUTING: PAST, PRESENT AND FUTURE.* 2013. – URL http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/vk5/report.html. – Zugriffsdatum: 28.07.2013

[Lexikon 2013] Lexikon, Technik: *Bestrahlungsstärke*. 2013. – URL `http://www.techniklexikon.net/d/bestrahlungsstaerke/bestrahlungsstaerke.htm`. – Zugriffsdatum: 29.07.2013

[von Luck u. a. 2010] Luck, Kai von ; Klemke, Gunter ; Gregor, Sebastian ; Rahimi, Mohammad A. ; Vogt, Matthias: Living Place Hamburg - A place for concepts of IT based modern living. In: *Hamburg University of Applied Sciences* (2010). – URL `http://livingplace.informatik.haw-hamburg.de/content/LivingPlaceHamburg_en.pdf`. – Zugriffsdatum: 20.05.2013

[Martinovsky und Wagner 2013] Martinovsky, Filip ; Wagner, Philipp: Gesichtserkennung mit Eigenfaces. (2013). – URL `http://www.bytefish.de/pdf/eigenfaces.pdf`. – Zugriffsdatum: 19.07.2013

[Meyer und Rakotonirainy 2003] Meyer, Sven ; Rakotonirainy, Andry: A survey of research on context-aware homes. In: *ACSW Frontiers 03 Proceedings of the Australasian information security workshop conference* (2003), S. 159–168

[Moon u. a. 2007] Moon, Aekyung ; Kim, Hyoungsun ; Kim, Hyun ; Lee, Soowon: Context-Aware Active Services in Ubiquitous Computing Environments. In: *ETRI Journal* 29 (2007), April, Nr. 2

[Nehrig 2003] Nehrig, Oliver: Entwurf und Realisierung eines Beschleunigungssensorsystems auf der Basis von in Silizium integrierten Mikromechanik für die besonderen Anforderungen bei Schwerlasthandhabungssystemen. In: *Vom Fachbereich Elektrotechnik der Gerhard-Mercator-Universität - Gesamthochschule Duisburg* (2003), S. 18

[Nii und Nii 1986] Nii, Yenny ; Nii, Penny: Blackboard Systems. In: *AI Magazine, KNOWLEDGE SYSTEMS LABORATORY Departments of Medical and Computer Science Stanford University* 7-2, 7-3 (1986). – URL `ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/86/1123/CS-TR-86-1123.pdf`. – Zugriffsdatum: 10.07.2013

[NXP 2012] NXP: UM10204 I^2C-bus specification and user manual. 4 (2012). – URL `http://www.nxp.com/documents/user_manual/UM10204.pdf`. – Zugriffsdatum: 21.07.2013

[Otto und Voskuhl 2011] Otto, Kjell ; Voskuhl, Sören: Weiterentwicklung der Architektur des Living Place. In: *Projektbericht Wintersemester 10/11* (2011), S. 1–22. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/`

projekte/master10-11-proj2/otto-voskuhl.pdf. – Zugriffsdatum: 21.05.2013

[Pulse-Eight 2013] Pulse-Eight: *libCEC*. 2013. – URL http://libcec.pulse-eight.com/. – Zugriffsdatum: 22.07.2013

[RaspberryPi 2013] RaspberryPi: *Raspberry Pi*. 2013. – URL http://www.raspberrypi.org. – Zugriffsdatum: 22.05.2013

[Riva u. a. 2005] Riva (Hrsg.) ; Vatalaro (Hrsg.) ; Davide (Hrsg.) ; Alcaniz (Hrsg.): *Ambient Intelligence The Evolution of Technology, Communication and Cognition Towards the Future of Human-Computer Interaction*. IOS Press, 2005

[RN-Wissen 2013] RN-Wissen: *I^2C*. 2013. – URL http://www.rn-wissen.de/index.php/I2C. – Zugriffsdatum: 21.07.2013

[Rosa und Silva 1997] Rosa, Francisco A. ; Silva, António R.: Component Configurer: A Design Pattern for Component-Based Configuration / IN THE 2 ND EUROPEAN CONFERENCE ON PATTERN LANGUAGES OF PROGRAMMING, EUROPLOP '97. 1997. – Forschungsbericht

[Schobersberger u. a. 2007] Schobersberger, Wolfgang ; Gufler, Veronika ; Hoffmann, Georg: Wieviel Licht braucht der Mensch? - Bedeutung von Licht für die Gesundheit. In: *Institut für Urlaubs-, Reise- und Höhenmedizin an der UMIT* (2007)

[Seidl u. a. 2011] Seidl, Richard ; Baumgartner, Manfred ; Bucsics, Thomas: *Basiswissen Testautomatisierung - Konzepte, Methoden und Techniken*. dpunkt.verlag, 2011

[Snyder u. a. 2009] Snyder, Bruce ; Davies, Rob ; Bosanac, Dejan: *ActiveMQ in Action*. Manning, 2009

[Strese u. a. 2010] Strese, Hartmut ; Seidel, Uwe ; Knape, Thorsten ; Botthof, Alfons: Smart Home in Deutschland. In: *Untersuchung im Rahmen der wissenschaftlichen Begleitung zum Programm Next Generation Media (NGM) des Bundesministeriums für Wirtschaft und Technologie / Institut für Innovation und Technik (iit) in der VDI/VDE-IT. Institut für Innovation und Technik* (2010)

[Texas Advanced Optoelectronic Solutions Inc. 2005] Texas Advanced Optoelectronic Solutions Inc. (Veranst.): *TSL2561 LIGHT-TO-DIGITAL CONVERTER*. Dezember 2005

[Turk und Pentland 1991] Turk, Matthew ; Pentland, Alex: Eigenface for Recognition. In: *Journal of Cognitive Neuroscience* 3 (1991), S. 71–86. – URL http://www.face-rec.org/algorithms/PCA/jcn.pdf. – Zugriffsdatum: 18.07.2013

[Ulm u. a. 2010] Ulm, Universität ; Guericke Universität Magdeburg, Otto von ; Neurobiologie, Leibniz-Institut für: *Eine Companion-Technologie für kognitive technische Systeme.* 2010. – URL http://www.sfb-trr-62.de. – Zugriffsdatum: 13.05.2013

[Urich 2008] Urich, Jaroslaw: Context-Aware Systeme: aktuelle Projekte. In: *Seminarausarbeitung im Rahmen der Veranstaltung Anwendungen 2 im Studiengang Informatik (Master of Science) am Studiendepartment Informatik der Fakultät Technik und Informatik der Hochschule für Angewandte Wissenschaften Hamburg* (2008), Februar, S. 17

[Waldorf 2013] Waldorf, Jürgen: licht.de und Fachverband Licht im ZVEI. 2013. – Persönliche Kommunikation

[Weiser 1991] Weiser, Mark: The computer for the 21st century. In: *Scientific American* (1991), S. 94–104. – URL http://wiki.daimi.au.dk/pca/_files/weiser-orig.pdf. – Zugriffsdatum: 20.05.2013

[Weiser und Brown 1997] Weiser, Mark ; Brown, John S.: Beyond calculation. New York, NY, USA : Copernicus, 1997, Kap. The coming age of calm technolgy, S. 75–85. – URL http://dl.acm.org/citation.cfm?id=504928.504934. – Zugriffsdatum: 14.05.2013. – ISBN 0-38794932-1

[xuuk 2007] xuuk: *eyebox2 User Manual.* 1.1, 2007

[xuuk 2013] xuuk: *eyebox2.* 2013. – URL https://www.xuuk.com. – Zugriffsdatum: 02.05.2013

Abbildungsverzeichnis

Glossar

AAL ..... Selbstbestimmtes Leben durch innovative Technik, Seite 14

AVR ..... Audio-Video-Receiver sind Mehrkanal-Hi-Fi-Verstarker. Sie konnen beispielsweise verschiedene Audio- und Videokanale auf ihre analoge und digitale Ausgange schalten., Seite 105

CEC ..... Consumer Electronics Control - Schnittstelle fur Ansteuerungsfunktionen von elektronischen Konsumergeraten. CEC benutzt einen seriellen, einadrigen Datenbus fur die Kommunikation., Seite 56

Der Couch-Potato Jemand, der sich nicht sportlich betatigt, sondern vorwiegend (fernsehend) auf der Couch sitzt oder liegt., Seite 2

EPG ..... Electronic Program Guide sind elektronisch verbreitete Informationen uber das aktuelle Horfunk- und Fernsehrprogramm., Seite 106

FA ..... Funktionale Anforderung, Seite 24

GPIO ..... General Purpose Input/Output, Seite 91

JUnit ..... Framework zum Testen von Java-Programmen. Geeignet für automatisierte Einzeltests., Seite 102

Message Broker . Oft eine Middleware, die eine Nachricht (protokoll-basierend) von einem Sender entgegennimmt und diese an protokoll-basiert an einen Empfanger schickt., Seite 36

Photodiode ..... Eine Halbleiter-Diode, wird oft verwendet, um Licht in elektrische Spannung, elektrischen Strom oder in Informationen zu umzuwandeln., Seite 37

Raspberry Pi .... Gunstiger kreditkartengroßer Einplatinen-Computer mit GPIO-Pins und weiteren Schnittstellen (UART, I^2C, SPI)., Seite 90

RFID ..... radio-frequency identification: Technik um Gegenstande und Lebewesen automatisch zu identifizieren und lokalisieren., Seite 28

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 09. August 2013	
Ort, Datum	Mosawer Nurzai