

Exzellenzcluster
Cognitive Interaction Technology
Ambient Intelligence
Dr. Thomas Hermann

Abschlussarbeit zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

**Konzeption und Implementation eines flexiblen und
benutzerfreundlichen Systems für die gestenbasierte
Interaktion mit schaltbaren Komponenten in
intelligenten Räumen**

Antonios Zaroulas
August 2014

Universität Bielefeld, Technische Fakultät

Studiengang:

Kognitive Informatik

Betreuer, Erstgutachter:

M.Sc. Alexander Neumann

Zweitgutachter:

Dipl.-Inform. René Tünnermann

Inhaltsverzeichnis

1. Einleitung	1
2. Zuordnung	3
2.1 Thematischer Hintergrund	3
2.1.1 Home Automation	3
2.1.2 Natural User Interfaces (NUI)	4
2.2 Ausblick	5
3. Anforderungsanalyse und Konzeption	6
3.2 Nutzfalldiagramm	6
3.3 Anforderungen	7
3.4 Domänen-Modell	9
3.4.1 Beschreibung der Konzepte	10
4. Realisierung	11
4.1 Verwendete Hardware	11
4.1.1 Microsoft Kinect	11
4.1.2 Geräte und Standards zur Home Automation	12
4.1.3 Tinkerforge	12
4.1.4 Funksteckdosen	12
4.2 Verwendete Software und Programmbibliotheken	13
4.2.1 OpenHAB	13
4.2.2 OpenNI Framework	14
4.2.3 REST Web-Services und Jersey	14
4.2.4 XML Verarbeitung	15
4.2.5 Tinkerforge API Bindings	15
4.3 Systemarchitektur	15
4.3.1 Realisierung der Zeigegesten-Erkennung	16
4.3.1.1 Interpretation der Zeigegeste anhand des Body-COM	16
4.3.1.2 Interpretation der Zeigegeste mithilfe der Schnittpunktberechnung	16
4.3.2 Realisierung der Initialisierungsgeste	19
4.3.2.1 Initialisierung mithilfe der Psi-Pose	19
4.3.2.2 Initialisierung mithilfe von NITE's Gesture recognition	20
4.3.3 Realisierung des Client/Server Modells	20
4.3.3.1 Kommunikationsschnittstelle zu OpenHAB	20
4.3.3.2 Aufbau des Web-Servers	21
4.3.4 Konfiguration	21
4.3.5 Realisierung des Panels	23
4.3.6 Erweiterbarkeit des Systems (Add-ons)	24
4.3.7 Betriebsmodi des Systems	25
5. Evaluation	26
5.1 Systemevaluation	26
5.2 OpenHAB Kompatibilität	27
5.3 Benutzerstudie	27
5.3.1 Versuchsbeschreibung	28
5.3.2 Versuchsdurchführung	28
5.3.3 Diskussion	29
6. Fazit	30
6.1 Ausblick	30
6.2 Zusammenfassung	30
A. Literaturverzeichnis	31
B. Versicherung	34
C. Material	36

1. Einleitung

Durch die zunehmende Integration und Unterstützung von Computern vielfältiger Form in unserem Alltag wird das Paradigma der „Ambient Intelligence“ in nächster Zeit immer mehr an Relevanz gewinnen. Das Thema beschäftigt sich mit einer intelligenten Umgebung, die sensitiv und adaptiv auf die Anwesenheit von Menschen und Objekten reagiert und dabei den Menschen bei der Lösung von Aufgaben unterstützt. In diesen Kontext muss man sich auch Gedanken über die Interaktion und Bedienmuster mit den Komponenten in der Umgebung machen. Künstliche Eingabegeräte, wie Tastatur oder Maus, werden wahrscheinlich im Prozess der Verbreitung von „Ambient Intelligence“ durch benutzerfreundliche Schnittstellen ersetzt werden. Gestensteuerung als natürliche Interaktionsform bietet hierfür vielversprechende Möglichkeiten.

Die technischen Gegebenheiten zu gestenbasierten Mensch-Maschine-Schnittstellen haben sich in letzter Zeit durch hinzukommen immer moderneren Technologien sehr stark verbessert. Verantwortlich dafür ist auch die Kinect aus der Unterhaltungselektronik. Diese ist eine Tiefendbildkamera, die Bewegungen und Gesten von Nutzern in Echtzeit erfasst und Entwicklern damit die Möglichkeit verschafft sie zu verarbeiten. Die Einführung der Kinect hat durch ihre hohe Verkaufszahlen [1] gezeigt, welches Potenzial hinter der Gestensteuerung steckt und das solche neuartigen Schnittstellen eine große Zukunft haben. Gesten erweisen sich in diesem Kontextwechsel als sehr praktisch, weil sie so nicht nur zur Kommunikation zwischen Menschen dienen, aber auch zur Interaktion mit Maschinen genutzt werden können. Aus diesem Grund werden Entwickler in nächster Zeit nicht nur Lösungen im Bereich der Unterhaltungselektronik bereitstellen, sondern auch zur Unterstützung im Wohnalltag.

Gestenbasierte Interaktion mit intelligenten Umgebungen kann in den Alltag übertragen werden, sodass Menschen etliche Komponenten im Wohnbereich wie Lampen, Jalousien, Heizung oder Steckdosen steuern können. So eine Umgebung könnte dem Nutzer bei der Bewältigung von Tätigkeiten im Wohnalltag unterstützen und Komfort bieten. Die vorliegende Arbeit befasst sich intensiv mit diesem Thema und stellt eine Lösung dazu bereit.

Ziel dieser Arbeit ist es also ein System zur gestenbasierten Interaktion mit alltagsgebräuchlichen Komponenten zu konzipieren und zu entwickeln, welches in intelligenten Räumen eingesetzt werden soll. Der Nutzer soll so in der Lage sein, mit Hilfe von Gesten, Komponenten ansteuern zu können. Das System soll Kompatibel zu etlichen Komponenten und leicht zu bedienen sein, sodass es auch im Alltag integriert werden kann.

Das System soll sich für Leute anbieten die einen Komfort anstreben, aber auch von mobilitätseingeschränkten Menschen verwendet werden können, um mit dessen Hilfe diverse Aufgaben ohne großen Aufwand zu bewältigen.

Die folgende Arbeit gliedert sich wie folgt: in Kapitel 2 wird zunächst die Arbeit thematisch zugeordnet und die beiden Konzepte der Home Automation und der Natural User Interfaces vorgestellt und beschrieben. In Kapitel 3 folgt dann die Konzeption des Systems. Dazu wird anfangs als Einführung ein Beispielszenario und ein Nutzfalldiagramm beschrieben, um daraufhin eine Anforderungsanalyse ableiten zu können. Es folgt ein Domänenmodell in dem alle vorkommenden Konzepte spezifiziert werden. Die verwendete Hardware und Software sowie die Systemarchitektur und die Realisierung der einzelnen Systemprozesse werden in Kapitel 4 beschrieben. In Kapitel 5 wird das entwickelte System mithilfe von Probanden evaluiert. Schließlich folgt ein Ausblick und ein Fazit der Arbeit.

2. Zuordnung

In diesem Kapitel wird die Thematik dieser Arbeit kategorisch zugeordnet und notwendige Hintergrundinformationen präsentiert. Dazu werden die einzelnen Kategorien und deren aktueller Stand erläutert. Schließlich folgt ein kleiner Ausblick auf die zukünftige Integration im Alltag der einzelnen Kategorien.

2.1 Thematischer Hintergrund

Die Thematik dieser Arbeit umfasst intelligente Räume und ist deshalb im Bereich der Home Automation zugeordnet. Da zusätzlich das vorliegende System eine gestenbasierte Steuerung bereitstellt und diese im Gebiet der „Natural User Interfaces“ einzuordnen ist werden unten beide Konzepte nochmal ausführlicher beschrieben.

2.1.1 Home Automation

Das Konzept der Home Automation beschäftigt sich mit der digitalen Steuerung und Vernetzung von Geräten und Haushaltsanlagen in intelligenten Räumen. Es umfasst Steuerungseinheiten und diverse Industriestandards die in Kapitel 4 beschrieben werden um Lampen, Jalousien, Hifi-Anlagen, Alarmanlage etc. zu regeln oder sie in automatischen Steuerungsabläufen mit einzubeziehen.

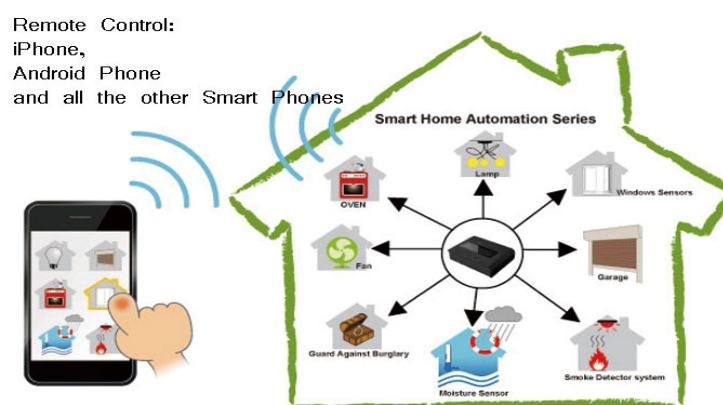


Abbildung 2.2: Home Automation Beispiel. [4]


Home Automation und seine Infrastruktur will dazu beitragen Wohnen komfortabler und sicherer zu Gestalten. Dazu kann es wichtige Teilaufgaben im Bereich des „Ambient Assisted Living“ [2] übernehmen.

Allerdings hat das Konzept noch nicht vollständig Einzug in den kommerziellen Markt gefunden, obwohl Technologien bereits existieren. Ein wesentlicher Grund dafür sind die zum Teil hohe Anschaffungskosten aber auch die teilweise nicht ausgereiften und zu komplexen Steuerungen.

Ein wichtige Rolle in diesem Konzept spielen die vielfältigen Benutzerschnittstellen. Mit diesen hat der Benutzer Zugriff auf Informationen von Sensoren, die sich an der Infrastruktur beteiligen. Damit können Geräte oder Haushaltsanlagen gesteuert werden. Im Zeitalter wo Smartphones und Tablets allgegenwärtig sind, ist es naheliegend, das zahlreiche Hersteller von Home Automation Systemen grafische Benutzeroberflächen für diese anbieten. An dem Punkt liegt auch der Unterschied zum System dieser Arbeit. Das vorliegende System setzt auf die gestenbasierte Steuerung und stellt somit eine sogenannte natürliche Benutzerschnittstelle bereit (NUI).

2.1.2 Natural User Interfaces (NUI)

NUI sind Benutzerschnittstellen, die eine Verbindung zu Realität aufbauen, indem sie menschliche Handlungen aus dem Alltag, wie Gesten oder Sprache, zur Interaktion mit technischen Geräten abbilden. So kann beispielsweise die NUI eines Systems per Sprachsteuerung bedient werden oder auch virtuelle Objekte auf einem Touchscreen ähnlich verschoben oder manipuliert werden, wie reelle Objekte. Durch die bereits bestehenden Wissensstrukturen dieser Handlungsweisen können Systeme oder Programme im Vergleich zu anderen die auf klassischen GUI's oder CLI setzen, sehr intuitiv bedient werden.



	CLI - Command Line Interface	GUI - Graphical User Interface	NUI - Natural User Interface
Primäre Eingabemedien	Tastatur	Tastatur und Maus	Finger und Sprache
Interface	Abstrakt (Codes)	Indirekt (Metapher)	Unmittelbar und direkt
	Text	Grafik	Objekte
Denken in	Zahlen und Codes	Symbolen	Objekten
Interaktion	Unnatürlich	Semi-Natürlich	Natürlich
	Gelernt	Wiedererkennend	Intuitiv
Mediale Ausprägung	Monomedial	Multimedial	Multimodal
Zielerreichung	Getrieben	Explorativ	Kontext-sensitiv
User Experience	Nüchtern	Anschaulich	Erlebnisorientiert
Wirkungseffizienz	Gering	Mittel	Hoch

Abbildung 2.1: Unterschiede von CLI, GUI und NUI . [3]

Die Verantwortung der NUI-Entwickler liegt also darin, die Bedienungsmöglichkeiten der Schnittstellen möglichst realitätsnah zu gestalten um diese Intuition zu gewährleisten.

Im Bezug zur vorliegenden Arbeit spielt die Auswahl der Gesten zur Ansteuerung der Haushaltsanlagen eine wichtige Rolle für den Nachweis der Benutzerfreundlichkeit. Eine leichte und schnelle Benutzung des System kann dazu führen, dass der Benutzer es akzeptiert und problemlos im Alltag integrieren kann.

2.2 Ausblick

Das Konzept der Home Automation wird in nächster Zeit an Popularität gewinnen. Dies liegt auch daran, dass auf dem Markt immer mehr Systeme hinzukommen, wodurch auch die Anschaffungskosten sinken werden. Durch die Vernetzung von Steuergeräten und Komponenten im Haus werden auch rund um die Uhr Daten eingesammelt und so individuelle Profile von Nutzern erstellt werden, was auch möglicherweise Dritten zugutekommt. Dieses Eindringen in den Lebensalltag der Nutzer hat auch Folgen für den Schutz der Privatsphäre. Dieses wichtige und komplexe Thema ist jedoch nicht Teil dieser Arbeit.

Auf der anderen Seite werden auch Entwickler aufgrund von immer komplexer werdenden Systemen und Softwarearchitekturen Natural User Interfaces in Anspruch nehmen. Gerade im Zeitalter von Smartphones und Tablet-PCs, werden Sprach- und Gestensteuerungen immer intelligenter, wodurch in nächster Zeit konventionelle Schnittstellen möglicherweise komplett abgeschafft werden könnten.

3. Anforderungsanalyse und Konzeption

Es folgt nun die Konzeption des Systems. Dazu wird zu Beginn des Kapitels ein Beispielszenario und ein Nutzfalldiagramm beschrieben, um den möglichen Aufbau und das Verhalten des Systems, zu erfassen. Danach folgt die Stellung der konkreten Anforderungen, die das System erfüllen soll. Schließlich wird ein Domänen-Modell vorgestellt und jedes Konzept einzeln spezifiziert.

3.1 Beispielszenario

Als Einführung in die Problematik wird folgend ein möglicher Anwendungsfall beschrieben. Dieser umfasst eine Vielzahl an Anforderungen die an ein solches System gestellt werden:

Max Mustermann sitzt an einem dunklen Winterabend auf dem Sofa seiner intelligenten Wohnung. Dort stellt er die Füße auf dem Hocker und öffnet das Buch, dass er heute Mittag per Post zugeschickt bekommen hat. Als Max zu lesen beginnt, bemerkt er, dass es im Raum zu dunkel ist. Zum Glück sind in Max' Wohnung alle Lichter mit einem System vernetzt, welches er allein durch Gestik mittels eines Sensors steuern kann. Dazu winkt Max dem Sensor 1 bis 2 Sekunden zu, bis er ein Rückmeldeton von sich gibt, der die Bereitschaft zur Interaktion signalisiert. Er zeigt mit der Hand auf eine Wandleuchte der Vorderwand worauf hin diese angeht. Als er bemerkt, dass diese Leuchte nicht direkt auf ihn zeigt, winkt er dem Sensor nochmal zu, zeigt gleichzeitig mit der einen Hand auf dieselbe Wandleuchte und mit der anderen auf die Leuchte nebenan. Jetzt geht die erste Leuchte aus und die zweite an, wobei diese für ein besseres Lichtverhältnis sorgt, mit dem Max sein Buch lesen kann. Nachdem er ein paar Seiten durchgelesen hat, beschließt er aufzuhören und Schlafen zu gehen. Er winkt ein letztes mal dem Sensor zu und zeigt in Richtung der Jalousien, damit diese heruntergefahren werden.

Nach einiger Zeit beschließt Max mit seiner Freundin Erika zusammenzuziehen und gestaltet die Wohnung kurzerhand nach Erikas Bedürfnissen um. Nach der Fertigstellung setzt sich Max an den Rechner und öffnet ein Konfigurationsprogramm, womit er spezifizieren und dem Sensor zeigen kann, wo sich alle ansteuerbaren Komponenten im Raum befinden.

3.2 Nutzfalldiagramm

Das obige Beispielszenario zeigt, wie so eine Interaktion zwischen Nutzer und System aussehen könnte.

Um das Verhalten des Systems während und nach der Interaktion mit dem Nutzer zu beschreiben, wird das in Abbildung 3.1 gezeigte Nutzfalldiagramm vorgestellt. Es bietet einen Einblick, wie Soft- und Hardware des Systems zusammenwirken werden und welche Fälle daraus entstehen:

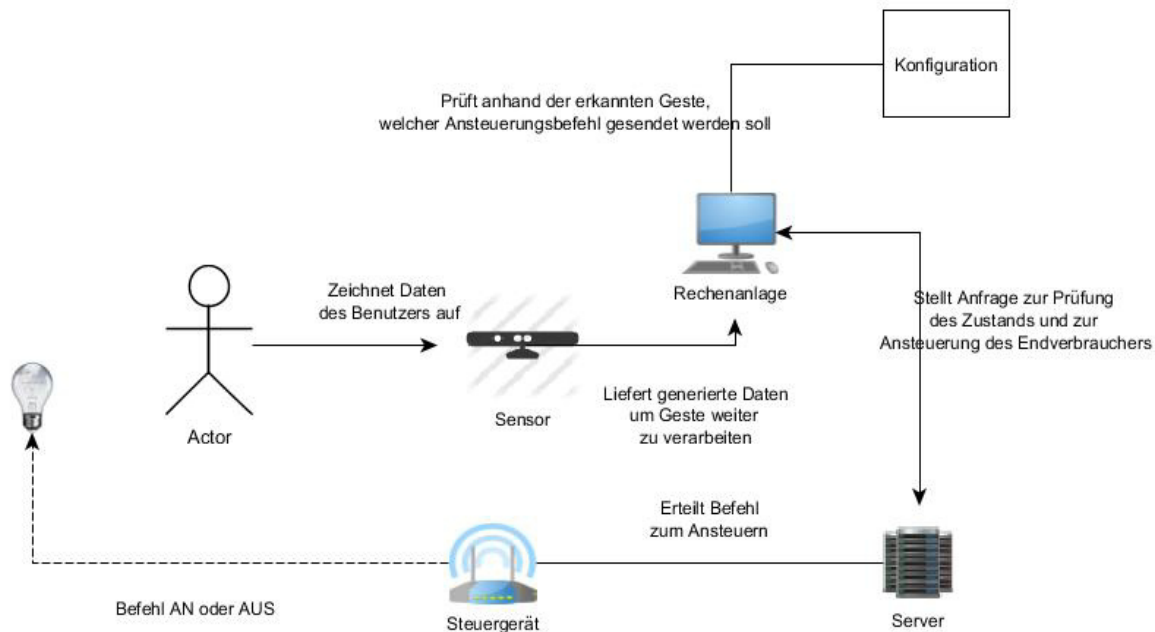


Abbildung 3.1: Nutzfalldiagramm

Wie auch im Beispielszenario befindet sich der Nutzer vor dem Sensor (siehe Abbildung 3.1) und gibt eine Anweisung mittels einer Zeigegeste. Der Sensor erfasst die Geste und liefert die Daten an die angeschlossene Rechenanlage. Dort werden die Gesten des Nutzers analysiert und anhand der Konfigurations-Datei die entsprechende Komponente ausfindig gemacht. Es folgt ein Verbindungsaufbau zum Server und eine Anfrage zum derzeitigen Zustand der Komponente. Je nachdem in welchem Zustand diese sich befindet, wird eine passende Anfrage zum Umschalten gestellt. Der Server nimmt die Anfrage entgegen und baut eine Verbindung zum Steuergerät auf. Dieses gibt letztlich nach erhaltenem Befehl das Signal zum Umschalten der Komponente, die in diesem Nutzfalldiagramm eine Lampe ist.

3.3 Anforderungen

Aus dem Beispielszenario und dem Anwendungsfalldiagramm können schon eine Vielzahl an Anforderungen abgeleitet werden. In diesem Abschnitt werden alle Anforderungen an das zu realisierende System gestellt und analysiert. So kann später eine klare Architektur erstellt werden.

- Das System soll in der Lage sein beliebig viele Nutzer im Blickfeld des Sensors erkennen zu können.

Begründung: Unter realistischen Umständen können sich mehrere Personen im Raum befinden, die auch potenzielle Nutzer dieses Systems sein können. Diese müssen vom System berücksichtigt werden.

- Das System soll in der Lage sein, eine eindeutige und intuitive Initialisierungsgeste des Benutzers identifizieren zu können, um so eine Sitzung zu starten.

Begründung: Damit das System nicht jede Bewegung des Nutzers als Geste interpretiert muss es wissen, dass folgende Geste ihm gilt.

- Das System soll in der Lage sein, erkennen zu können, an welche Stelle in den Raum ein Benutzer hinzeigt. Dabei soll es auch erkennen können, wenn der Benutzer mit beiden Händen gleichzeitig an verschiedene Stellen zeigt.

Begründung: Um die NUI des Systems möglichst intuitiv bedienen zu können (siehe Kapitel 2), sind deiktische Gesten eine vielversprechende Lösung. Sie werden im Alltag häufig genutzt, um auf Objekte zu deuten. So reduziert sich der initiale Lernaufwand des Nutzers.

- Das System soll in der Lage sein, unterscheiden zu können, welcher Benutzer eine Interaktion anfragt und auch nur dessen Anweisungen befolgen. Alle übrigen Benutzer müssen in diesem Moment irrelevant für das System sein.

Begründung: Um redundante Ansteuerungen von Komponenten zu umgehen, ist diese Anforderung unerlässlich. So können Konflikte beim Vorhandensein mehrerer Benutzer, vermieden werden.

- Es soll das aktuell verarbeitete Tiefenbild des Sensors, in einen Panel angezeigt werden, indem auch alle erkannten Benutzer farblich hervorgehoben sein sollen.

Begründung: Diese optionale Anforderung wird gestellt um einen Einblick in das System zu bieten, welcher für Entwickler oder Neugierige interessant sein kann.

- Es soll gekennzeichnet werden, wenn das System die Initialisierungsgeste des Nutzers erkannt hat und somit auf eine Anweisung wartet .

Begründung: Damit der Benutzer in Kenntnis gesetzt wird, wann er in Richtung einer Komponente im Raum zeigen kann, muss er zunächst ein Feedback vom System erhalten können.

- Das System soll eine Kommunikations-Schnittstelle zu OpenHAB [5] bereitstellen, um entgegennehmende Anweisungen dorthin weiterleiten zu können. Dabei soll aber keine Abhängigkeit bestehen, sodass das System auch andere Serverarchitekturen ansprechen kann.

Begründung: OpenHAB (Siehe Kap. 4.2.1) unterstützt etliche Industriestandards und Bussysteme zur Ansteuerung von Komponenten. Auf diese Weise können Stellglieder und Sensoren von zahlreichen Herstellermarken vom System benutzt werden.

- Es soll ein Web-Server (kompatibel zum OpenHAB-Protokoll) bereitgestellt werden, der vom System Steuersignale entgegennimmt um handelsübliche Funksteckdosen anzusteuern.

Begründung: Die Flexibilität des Systems ist zwar Aufgrund der OpenHAB Schnittstelle gewährleistet, aber soll mit Hilfe des Servers auch eigenständig benutzt werden können.

- Eine Konfiguration soll bearbeitet und gespeichert werden können. Dies soll sowohl textbasiert, als auch visuell mit Hilfe eines Konfigurationsprogramms möglich sein.

Begründung: Mit dem Konfigurationsprogramm sollen technisch nicht versierte Nutzer die Möglichkeit haben das System zu konfigurieren.

- Das System soll eine erweiterbare Architektur bereitstellen. Der Begriff erweiterbar bedeutet dabei, dass man das System um weitere „Add-ons“ bereichern kann.

Begründung: Um es nicht nur bei einem Ein- und Ausschalten einer Komponente zu belassen, können mit den „Add-ons“ auch, soweit es eine Komponente zulässt, andere Zustände eingenommen werden. „Add-ons“ sollen gestartet werden können, wenn der Benutzer in Richtung dieser Komponenten zeigt.

3.4 Domänen-Modell

Aus den Anforderungen kann nun eine klare Domäne definiert werden. Um eine visuelle Vorstellung des Problems zu repräsentieren, muss diese Domäne in Problem relevante Konzepte zerlegt werden und deren Zusammenhang durch ein Diagramm dargestellt werden. Das folgende Domänen-Modell realisiert diese visuelle Repräsentation der zerlegten Konzepte und deren Beziehung zueinander.

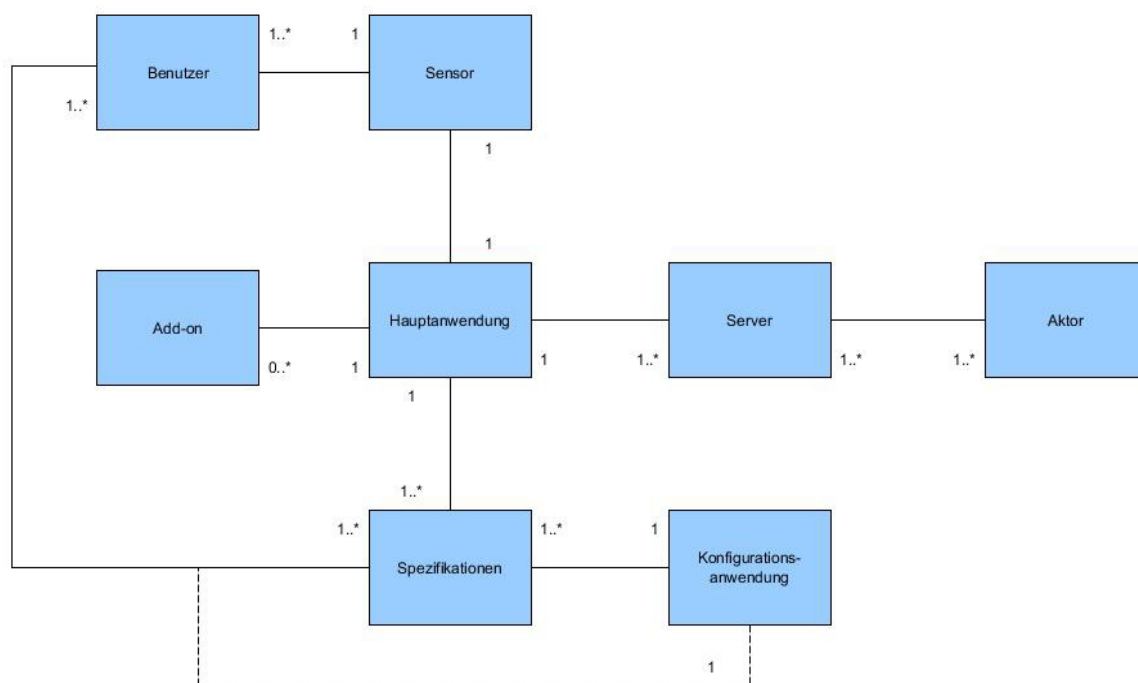


Abbildung 3.2: Domänen-Modell des Systems

3.4.1 Beschreibung der Konzepte

Benutzer:

Erstellen Spezifikationen direkt oder mittels der Konfigurationsanwendung und interagieren mit dem System durch dem Sensor.

Sensor:

Zeichnet Daten vom Umfeld auf, die von der Hauptanwendung weiter verarbeitet werden.

Hauptanwendung:

Die Anwendung ist das Kern-Konzept des Systems. Hier werden alle Daten vom Sensor ausgewertet, Gesten analysiert, das Panel gezeichnet, die Client-Schnittstelle zur Kommunikation mit dem Server bereitgestellt und nach Bedarf, Aufgaben an Add-ons weitergegeben. Die jeweiligen Anweisungen werden von den Spezifikationen bezogen.

Spezifikationen:

Beinhaltet alle kontextabhängigen Informationen zur Verhaltensweise des Systems.

Konfigurations-Anwendung:

Stellt eine Anwendung mit Benutzeroberfläche bereit, in der Spezifikationen geladen und vom Benutzer manipuliert werden können.

Add-on:

Werden nach Bedarf von der Hauptanwendung einzeln aufgerufen und übernehmen dessen Aufgabe Gesten zu analysieren. Beeinflussen somit das Verhalten des Systems.

Server:

Werden Anfragen der Hauptanwendung aus, um jeweilige Aktoren anzusteuern. Die one-to-many Beziehung mit der Hauptanwendung ist vordefiniert, aufgrund der Flexibilität des Systems, beliebig viele Serverarchitekturen ansprechen zu können.

Aktor:

Erhalten Steuersignale von den Servern, um jeweilige Komponenten ansteuern zu können.

4. Realisierung

Dieses Kapitel beinhaltet den Hauptteil dieser Arbeit und beschreibt den Entwicklungsprozess des Systems. Es werden zunächst die verwendete Hardware, Software und Bibliotheken beschrieben, die für die Realisierung des Systems verwendet wurden. Anschließend wird aus dem Domänen-Modell und dessen Konzepten die vollständige Systemarchitektur konkretisiert und dessen Umsetzung detailliert beschrieben.

4.1 Verwendete Hardware

In folgendem werden Hardware Komponenten und deren Eigenschaften beschrieben, die für die Realisierung des Systems verwendet wurden bzw. verwendet werden können.

4.1.1 Microsoft Kinect

Die Kinect ist ein Multisensor Gerät dass von Microsoft und PrimeSense für die Spielekonsole Xbox 360 entwickelt wurde und Benutzern erlaubt, Spiele mittels Gesten oder Sprachbefehlen zu steuern. Abbildung 4.1 veranschaulicht die Sensorleiste und ihre wichtigsten Hardware-Komponenten.



Abbildung 4.1: Kinect und ihre Sensoren [7]

Der Tiefensensor der Kinect verfügt über eine Infraroteinheit (siehe Abb. 4.1), welche ein Muster von Punkten projiziert, die vom Graustufen CMOS Sensor (ebenfalls Abb. 4.1) zurück gelesen werden. Der Sensor erkennt reflektierende Segmente des Punktemusters und wandelt ihre Intensitäten in Entfernung um. Jedes vom Tiefensensor erzeugte Frame hat eine VGA Auflösung von 640x480 Pixel mit 11 Bit Tiefenwerte, die 2,048 Empfindlichkeitsstufen zu Verfügung stellen [18 S.2].

Mit den gewonnenen Informationen kann somit die Position des Benutzers oder seiner einzelnen Körperglieder in Bezug zur Kinect im Raum bestimmt werden (x, y, z Koordinaten), was im realisierten System genutzt wurde um Zeigegesten zu berechnen.

4.1.2 Geräte und Standards zur Home Automation

Im Bereich der Home Automation existieren eine bereits große Menge von interoperablen Systemen mit kombinierbaren Aktoren und Sensoren. Die Kombinierbarkeit von Geräten setzt allerdings voraus, dass sie dem selben Standard folgen. Von populären Standards wie ZigBee [8], Z-Wave [9] oder KNX [10] existieren zahlreiche Geräte die in OpenHab alle zusammengebracht werden und somit auch mit dem System dieser Arbeit angesteuert werden können.

4.1.3 Tinkerforge

Tinkerforge [11] ist ein Baukastensystem, bestehend aus Microcontrollerbausteinen genannt Bricks und aus Bricklets, welche Bricks um viele Fähigkeiten erweitern können.

Ein Bricklet Modul das zur Ansteuerung von Funksteckdosen vorgesehen ist, ist das Remote Switch Bricklet. Dieses ist mit einem 433 Mhz Transceiver und einer SMA Antenne ausgestattet. Es lässt sich über einen MasterBrick betreiben, an dem es per Kabel angeschlossen wird.

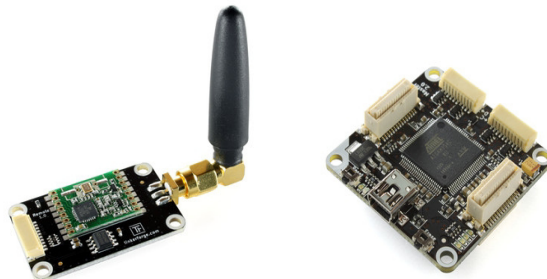


Abbildung 4.2: Links: Remote Switch Bricklet. Rechts: Master Brick [11].

Die Hardware wird im System als Steuergerät für Funksteckdosen verwendet, welche vom Web-Server Steuersignale empfängt.

4.1.4 Funksteckdosen

Funksteckdosen [11] sind drahtlos mittels einer Fernbedienung schaltbare Steckdosen. Jede handelsübliche Funksteckdose verfügt über 10 DIP (Dual Inline Package) Schalter, mit deren Kombination sich verschiedene Binärcodes einstellen lassen. Die ersten Fünf DIP-Schalter sind für das Einstellen des House-Codes, mit dem sich eine Funksteckdose zu einer Gruppe zuordnen lässt. Die restlichen Fünf Schalter sind für das Einstellen des Receiver-Codes, mit dem sich eine Funksteckdose innerhalb eines Systems unterscheiden lässt. Durch die Kombination der DIP-Schalter ergeben sich $2^2 * 5 = 160$ verschiedene Binärcodes, mit dem sich Funksteckdosen innerhalb eines Systems eindeutig unterscheiden lassen.

Um sie mit dem Remote Switch Bricklet ansteuern zu können, müssen sie auf einem 433 Mhz Frequenzband kommunizieren und auf dem PT2262 oder HX2262 IC basieren.

Solche handelsüblichen Funksteckdosen bieten keine Netzwerktopologie, wie es in ZigBee oder Z-Wave der Fall ist an und haben auch keine bidirektionale Verbindung zum Steuergerät, um ihren Zustand abzufragen zu können. Sie sind jedoch Aufgrund ihrer sehr geringen Anschaffungskosten und leichten Verfügbarkeit eine akzeptable Lösung und wurden im Web-Server als anzusteuern Einheit eingesetzt.

4.2 Verwendete Software und Programmbibliotheken

Wie auch im oberen Abschnitt folgt nun Beschreibung der verwendeten Software und Programmbibliotheken, die für die Realisierung des Systems aus den gestellten Anforderungen heraus bevorzugt worden sind. Ihre Eigenschaften werden kurz erläutert und dazu wird begründet für welchen Zweck sie im System eingesetzt werden.

Dazu ist zu erwähnen, dass für alle folgenden Bibliotheken und APIs, Implementierungen oder Wrapper in Java [21] existieren und für die Realisierung des Systems ausschließlich diese Programmiersprache verwendet wurde.

4.2.1 OpenHAB

OpenHAB [5] ist eine in Java entwickelte und auf der OSGi Plattform [12] basierende Software für die Integration diverser Home Automation Systeme und Standards (siehe 4.1.2). Sie ermöglicht übergreifende Automatisierungsregeln und bietet einheitliche Benutzer-Schnittstellen (Native OS Client, Native Android Client, Web-Frontend) genannt Sitemaps an. OpenHAB wird unter der Eclipse Public Licence entwickelt und ist OpenSource. Das Projekt ist in zwei Teilen unterteilt. Der erste ist die OpenHAB-Runtime die den Serverprozess entspricht. Da sie auf OSGi basiert, wird eine modulare Architektur zu Verfügung gestellt, die das Hinzufügen von Bindings erlaubt und so OpenHAB um weitere Standards bereichert werden kann. Der Zweite Teil ist der OpenHAB-Designer, der eine Konfigurations-Oberfläche auf Eclipse Basis bereitstellt.

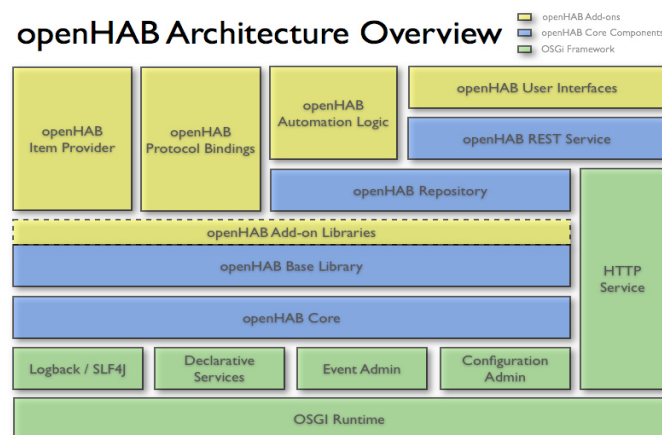


Abbildung 4.3: OpenHAB Architektur [5].

OpenHAB stellt eine REST-API zu Verfügung (siehe Abbildung 4.3), sodass es im System dieser Arbeit integriert werden kann. Sie kann genutzt werden um Steuerkommandos an Items aus der Sitemap zu senden oder um deren Zustand abfragen zu können.

4.2.2 OpenNI Framework

Das OpenNI-Framework [18 S.3] stellt eine Schnittstelle zu einer Vielzahl von Sensor-treibern (in dem Fall der Kinect) zur Verfügung, indem es dessen Daten auswertet und Funktionalitäten bereitstellt. Dazu bietet es Middleware-Komponenten an, die in das Interface einbezogen werden können, sodass Sensordaten auch auf High-Level Ebene manipulieren zu können.

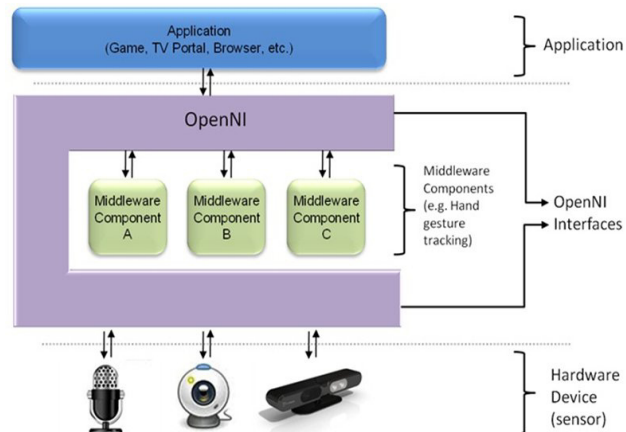


Abbildung 4.4: OpenNI Framework Schichten [13].

In Abbildung 4.4 sind die drei Schichten des Frameworks zu sehen. Eine Middleware-Komponente mit der Handbewegungen und Gesten erkannt werden können und im Rahmen dieser Arbeit zusammen mit dem Interface verwendetet wird, ist die von NITE.

Auf Sensor Funktionalitäten und High-Level Abstraktionen gelangt man über den Production-Node, der das Hauptelement des OpenNI Interfaces ist. Er liefert eine Vielzahl an Generatoren mit denen auf Sensor- und Middleware Features zugegriffen werden kann. Generatoren die im System verwendet wurden sind der Depthgenerator, um Datenflüsse an Tiefenwerten zu beziehen und der Usergenerator für die Repräsentation des kompletten Körpers oder einzelner Abschnitte davon (Tracking). Für die Verarbeitung von Gesten wird NITEs Sessionmanager verwendet, der eine Vielzahl an Detektoren für die Gestenerkennung bereitstellt.

4.2.3 REST Web-Services und Jersey

Das Representational State Transfer (kurz REST) [20] ist ein Architektur Modell, welches beschreibt, wie das Web funktionieren könnte. Es ist lediglich ein Programmierparadigma und keine feste Norm oder Standard, sodass es als Referenz und als Anleitung dient, wie Standards für das Web gerecht eingesetzt werden können. Nach dem REST-Prinzip wird eine Ressource von einem Web-Server zu Verfügung gestellt und kann eindeutig über eine URL identifiziert werden.

Dabei existieren Operationen mit denen bestimmt werden kann, ob die Ressource gelesen, neu angelegt, gelöscht, aktualisiert oder aufgelistet werden soll. Diese Operationen sind typische HTTP-Methoden. Die adressierte Ressource hat eine Repräsentation, die in jeden Format sein kann (XML, JSON, Reiner Text, etc.).

Mit JAX-RS [14] bietet Java einen Standard zum deklarieren von REST-basierten Web-Services an. Eine Referenzimplementierung dieses Standards stellt das Projekt Jersey [15] dar, mit dem sich entweder ein Endpunkt definieren lässt, sodass der REST Web-Service in einem Servlet-Container (z.b. TomCat) oder über einen in Java (seit SE 6) mitgelieferten Mini-Http-Server laufen kann. Letzterer wurde auch genutzt um einen REST-Basierten Server aufzustellen, der Zustandsänderungen von Ressourcen entgegennimmt, um sie in Form von Steuersignalen an das MasterBricklet weiterleiten zu können, aber auch um Clients vom derzeitigen Zustand der Ressource informieren zu können.

4.2.4 XML Verarbeitung

Um XML-Dateien [6] verarbeiten zu können existieren zahlreiche APIs. Zwei Bekannte davon sind DOM und StAX. Da es sich um reine API's handelt, bietet Java mit JDOM [16] und der Java StAX Variante [17] konkrete Implementierungen dieser API's an.

XML bietet eine strukturierte und hierarchische Form zur Beschreibung von Daten und wurde deshalb in dieser Arbeit zur Realisierung der Konfigurationsdatei genutzt. Die Elemente der Datei werden in der Hauptanwendung mit Hilfe von JDOM geladen und weiterverarbeitet, während auf der anderen Seite StAX genutzt wird, um neue Elemente in der XML-Datei von der Konfigurationsanwendung zu erstellen.

4.2.5 Tinkerforge API Bindings

Die API Bindings ermöglichen es Brick und Bricklets auf Hochsprachen-Ebene heraus zu steuern. Dafür wird eine Verbindung zum Brick Daemon [11] erstellt. Jeder Funktionsaufruf erzeugt ein TCP/IP Paket, das zum Brick verschickt wird.

Die Bindings werden auf dem Web-Server dieser Arbeit verwendet um Steuersignale an das Master-Brick zu versenden.

4.3 Systemarchitektur

Anhand der gewonnenen Informationen zur verwendeten Software und Hardware kann jetzt eine konkrete Systemarchitektur vorgestellt werden. Abbildung 4.5 stellt diese Architektur dar. Im folgenden Abschnitt wird die Umsetzung der Systemkomponenten im einzelnen beschrieben.

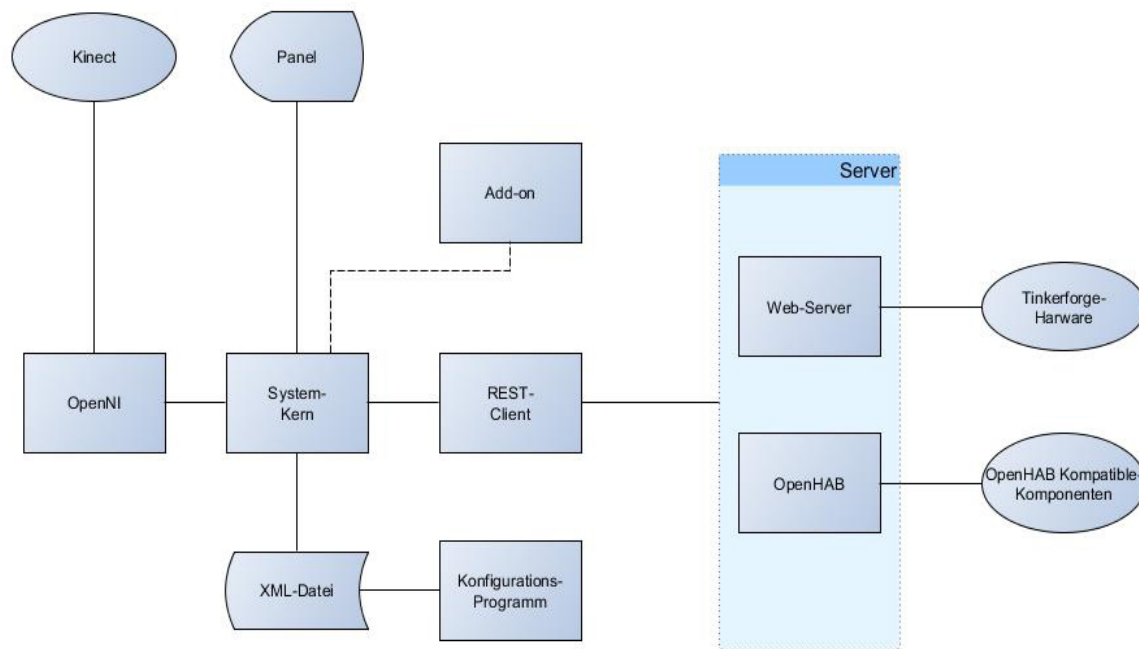


Abbildung 4.5: Systemarchitektur in Form eines Flussdiagramms.

4.3.1 Realisierung der Zeigegesten-Erkennung

Für die Erkennung der Zeigegesten im Systemkern wurden zwei Lösungsansätze in Betracht gezogen. Im Folgenden werden beide vorgestellt und begründen, welcher davon bevorzugt worden ist.

4.3.1.1 Interpretation der Zeigegeste anhand des Body-COM

Mit dieser Methode wird die Position der Hand anhand ihrer relativen Position zum Körpermassenzentrum (Body-COM) bestimmt. Mit Hilfe von OpenNi's SkeletonCapability können Informationen von Position der Hand und Körpermassenzentrum relativ zum Sensor bezogen werden. Diese liefert OpenNI in Form von dreidimensionalen Koordinaten. Mit den Koordinaten kann damit Winkel und Abstand (z-Achse) der Hand vom Körpermassenzentrum berechnet werden. Auf diese Weise kann eine berechnete Pose mit der Position einer Komponente im Raum assoziiert werden. Dieser Lösungsansatz ist leicht zu realisieren, bringt aber viele Nachteile mit sich. Das System muss viele Fallunterscheidungen bewerkstelligen, damit es bestimmen kann, welche Komponente angesteuert werden soll. Dazu muss jedes mal die Position des Benutzers im Raum berücksichtigt werden, da die Handpose an jeder Stelle anders interpretiert werden muss.

4.3.1.2 Interpretation der Zeigegeste mithilfe der Schnittpunktberechnung.

Bei der Schnittpunktberechnung geht es konkret um einen Gerade-Ebenen-Schnittpunkt. Dazu müssen erstmals Gerade und Ebene Mathematisch definiert werden:

Eine Gerade kann im euklidischen Raum in der Parameterform durch einen Stützvektor: \vec{S}_G und einen Richtungsvektor: \vec{R}_G dargestellt werden:

$$\vec{x} = \vec{S}_G + t * \vec{R}_G \quad (1)$$

Der Stützvektor ist dabei der Ortsvektor eines beliebigen Punktes auf der Geraden und der Richtungsvektor bildet sich aus der Differenz zu einem anderen beliebigen Punkt. Der Wert von t entspricht genau einen Punkt der Geraden. Eine Ebene kann in der Normalenform ebenfalls durch einen Stützvektor: \vec{S}_E und einen Normalenvektor: \vec{n} dargestellt werden:

$$\vec{n} \cdot (\vec{x} - \vec{S}_E) = 0 \quad (2)$$

Der Stützvektor ist auch, wie in der Geradengleichung, der Ortsvektor eines beliebigen Punktes in der Ebene und der Normalenvektor ist ein Vektor, der senkrecht auf der Ebene steht. Er kann aus dem Skalarprodukt zweier zueinander senkrechten Richtungsvektoren \vec{R}_1 und \vec{R}_2 berechnet werden:

$$\vec{n} = \vec{R}_1 \cdot \vec{R}_2$$

Um einen Schnittpunkt von Gerade und Ebene zu berechnen, muss die Geradengleichung in der Normalenform der Ebenengleichung eingesetzt werden (1 in 2):

$$\vec{n} \cdot (\vec{S}_G - \vec{S}_E) + t * \vec{n} \cdot \vec{R}_G \quad (3)$$

Falls das Skalarprodukt von $\vec{n} \cdot \vec{R}_G = 0$, so sind Ebene und Gerade Parallel zueinander und die Existenz des Schnittpunktes kann widerlegt werden. Ansonsten wird (3) nach t aufgelöst und in der Geradengleichung (1) eingesetzt:

$$\vec{x} = \vec{S}_G + \frac{-\vec{n} \cdot \vec{S}_G - \vec{S}_E}{\vec{n} \cdot \vec{R}_G} \quad (4)$$

Das Ergebnis dieser Gleichung ist der genaue Schnittpunkt von Gerade und Ebene.

Im Bezug zur Gesteninterpretation des Systems kann dieses Verfahren eingesetzt werden um die Zeigerichtung des Armes des Benutzers zu berechnen. Analog zur Geradengleichung kann als Stützvektor \vec{S}_G mit Hilfe von OpenNI's SkeletonCapability die Koordinaten des Ellenbogens bezogen werden, um sie als Endpunkt des Vektors zu definieren. Der Richtungsvektor \vec{R}_G hat dann als Ursprung die Koordinaten des Ellenbogens und als Endpunkt die der Hand des Benutzers.

Um eine Ebene zu definieren, können die Maximalwerte und Nullpunkt des Koordinatensystems verwendet werden, umso virtuelle Wände im Raum simulieren zu können. Die maximalen Werte X_{max} , Y_{max} , Z_{max} die Raumbreite, Raumhöhe und Raumtiefe simulieren sollen, können wie in Abschnitt 4.3.4 gezeigt wird frei bestimmt werden. Zum besseren Verständnis zeigt Abbildung 4.6 wie die Linke Wand als Ebene definiert werden kann:

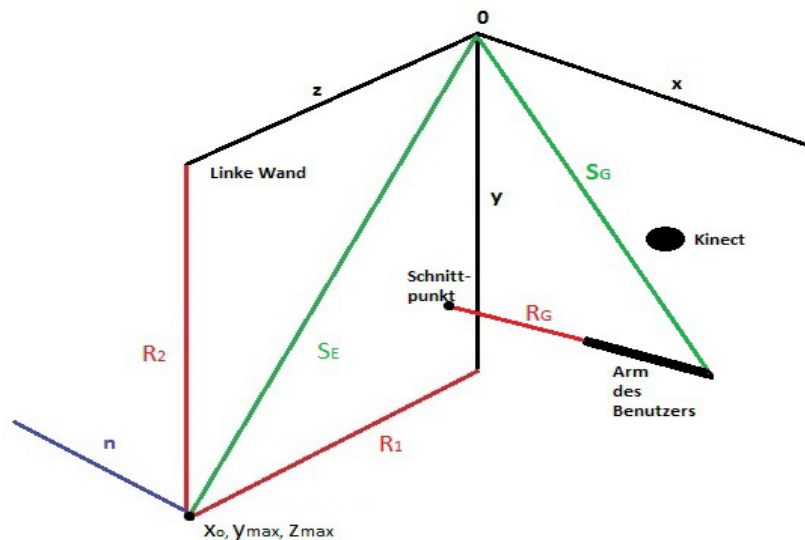


Abbildung 4.6: Graphische Darstellung der definierten linken Wand als Ebene.

Zu sehen sind in Abbildung 4.6 eine definierte Ebene aus den beiden Richtungsvektoren R_1 und R_2 und Stützvektor S_E . Mithilfe der beiden Richtungsvektoren lässt sich der Normalvektor bestimmen und somit kann die Ebenengleichung (3) aufgestellt werden. Der Arm des Benutzers in der Abbildung berechnet sich wie oben beschrieben. Analog zur linken Wand können auf diese Weise auch alle anderen Wände im Raum als Ebene definiert werden.

Das System kann mit der Schnittpunktberechnung bestimmen auf welche Wand der Nutzer zeigt und damit die anzusteuern Komponente ausfindig machen. Durch ein paar einfache Fallunterscheidungen des Schnittpunktwertes lässt sich eine Wand auch in mehrere Bereiche unterteilen, sodass auch mehr als eine Komponente dort drauf zugeordnet werden kann. Abbildung 4.7 zeigt wie eine Wand in Bereichen unterteilt werden kann:

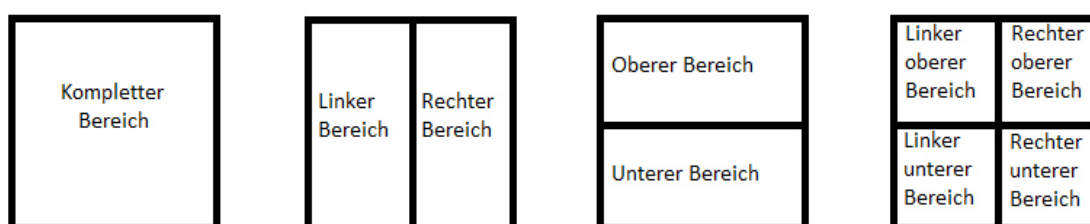


Abbildung 4.7: Unterteilung der Wand in Bereiche.

Mit Hilfe der Schnittpunktberechnung kann also mit geringen Rechenaufwand die Zeigegeste des Nutzers effizient berechnet werden. Dazu ist die Position des Nutzers anders als der in Abschnitt 4.3.1.1 vorgestellte Lösungsansatz irrelevant für die Berechnungen. Aus diesem Grund wurde für die Zeigegesten-Erkennung im System dieser Lösungsansatz verwendet.

4.3.2 Realisierung der Initialisierungsgeste.

Für die Interpretation der Initialisierungsgeste (siehe 3.3) im Systemkern wurden ebenfalls zwei Lösungsansätze in Betracht gezogen:

4.3.2.1 Initialisierung mithilfe der Psi-Pose

Um eine Verfolgung des Nutzers, genannt Tracking (siehe 4.2.2), vor dem Sichtfeld des Sensors durchführen zu können, benötigt das OpenNI-Framework anfangs eine Kalibrierung vom Körper des Nutzers. Das Framework bietet die Möglichkeit die Kalibrierung anhand einer Psi-Pose [WR14] einzuleiten. Nach erfolgreicher Abwicklung wird der Nutzer getrackt, womit konkrete Werte der Position des Nutzers oder dessen Körperabschnitte (in Form von Koordinaten) bezogen werden können. Diese Tatsache kann ausgenutzt werden um die Psi-Pose als Initialisierungsgeste im System zu verwenden. Dazu wartet das System anfangs auf eine Psi-Pose. Nach Erkennung der Pose und erfolgreicher Kalibrierung wird der Benutzer getrackt, wobei ein Bestätigungston erklingt und somit in der Lage auf Komponenten zu zeigen. Nach erfolgreicher Durchführung wird das Tracking beendet und das System wartet erneut auf die Psi-Pose des Nutzers. Abbildung 4.8 demonstriert die einzelnen Schritte dieser Prozedur:

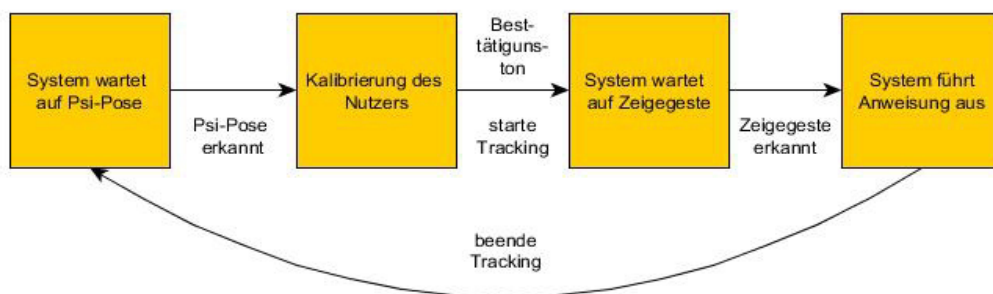


Abbildung 4.8: Schritte eines Interaktions-Verlaufs mit Hilfe der Psi-Pose

Nach Abschluss der Kalibrierung und dem Warten auf eine Zeigegeste muss das System für ein bestimmtes Zeitintervall inaktiv bleiben, damit der Nutzer nach Ausführung der Psi-Pose die Hand in Richtung der Komponente positionieren kann. Innerhalb dieses Intervalls kann es vorkommen, dass die Hand auf dem Weg zur gewünschten Positionierung auch auf andere Komponenten zeigen kann. In dieser inaktiven Phase werden sie vom System ignoriert.

Mit Hilfe der Psi-pose können also Situationen klar Unterschieden werden, damit keine Konflikte während der Interaktion zwischen Nutzer und System entstehen können. Dazu wird ein Nutzer nur dann getrackt, wenn er auch eine Interaktion mit dem System herbeirufen möchte, womit unnötiger Rechenaufwand eingespart wird. Allerdings hat dieser Lösungsansatz einen großen Nachteil. Damit die Psi-Pose erkannt werden kann, muss der Nutzer sie stehend vor dem Sensor einnehmen, womit die Alltagstauglichkeit des Systems in Frage gestellt werden könnte.

4.3.2.2 Initialisierung mithilfe von NITE's Gesture recognition

Die NITE Middleware Komponente (siehe 4.2.2) ist in der Lage einen Datenfluss von Handpositionen aufzunehmen, um daraus Gesten ableiten zu können [18 S.3]. Das System kann so mittels NITE's Sessionmanager benachrichtigt werden, sobald eine unter etlichen selektierte Gesten des Benutzers erkannt wird. Eine geeignete Geste für die Initialisierung ist in diesem Fall die Winkgeste (genannt waving im Kontext von NITE). Sie wird von Menschen benutzt um Aufmerksamkeit zu erreichen und kann für die Initialisierung des Benutzers eingesetzt werden. Da die Kalibrierungspose (Psi-Pose) von der Winkgeste ersetzt wird, kann OpenNI auf automatische Kalibrierung umgestellt werden. Abbildung 4.9 demonstriert diesen Interaktionsverlauf:

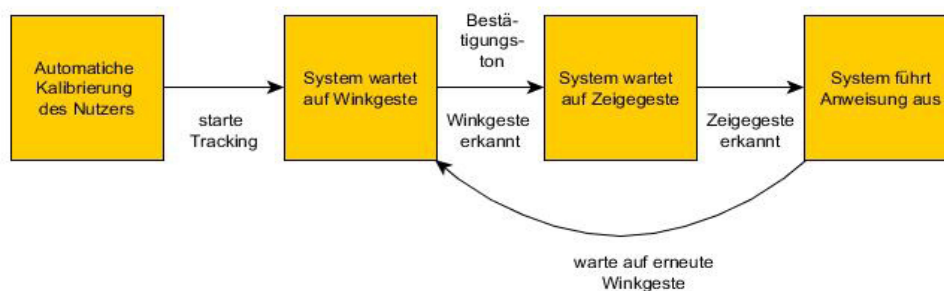


Abbildung 4.9: Schritte eines Interaktion-Verlaufs mit Hilfe einer Winkgeste

Das inaktive Zeitintervall (3 Sekunden) nach Erkennung der Winkgeste muss auch mit diesem Verfahren eingeplant werden.

Es hat sich innerhalb der Entwicklungsphase des Systems herausgestellt, dass die automatische Kalibrierung des Nutzers ohne Probleme verläuft, selbst wenn der Nutzer sitzend kalibriert werden muss.

Da die Winkgeste funktionell im Kontext der Initialisierungsgeste besser geeignet ist als die Psi-Pose und aufgrund der Eigenschaft der schnellen Auto-Kalibrierung des Nutzers wurde im System das Verfahren dieses Lösungsansatzes verwendet.

4.3.3 Realisierung des Client/Server Modells

4.3.3.1 Kommunikationsschnittstelle zu OpenHAB

Um auf OpenHAB-items (siehe 4.2.1) mittels REST-API, zugreifen zu können benutzt die Client-Schnittstelle des Systems eine REST-URL der Form [5]:

- `http://<Host>:<Port>/rest/items/<Itemname>` (1)

Durch einfügen von konkreten Werten für <Itemname> können Items (Komponenten) als Ressourcen gekennzeichnet werden. Die REST-URL zum Erfragen des Zustands der Ressource hat die Form:

- `http://<Host>:<Port>/rest/items/<Itemname>/state` (2)

OpenHAB liefert dazu einen Wert in Form von reinen Text (Media Type), der den momentanen Zustand der Ressource zeigt (z.B Ein oder Aus). Um den Zustand ändern zu können kann eine POST-Anfrage [20] durchgeführt werden, wobei der gewünschte Zustand der Ressource auch in Form von reinen Text an OpenHAB gesendet wird. Da OpenHAB diverse items verwaltet, muss dieser Text passend zum Zustand der Ressource sein [5].

Der Ablauf im System verläuft also wie folgt: Das System interpretiert die Zeigegeste des Nutzers und bezieht die passende REST-URL (1) der Komponente aus der Konfigurationsdatei (siehe 4.3.2). Diese übergibt das System an der Client-Schnittstelle. Nachdem zunächst der derzeitige Zustand der Komponente mittels (2) abgefragt wird, erfolgt eine POST-Anfrage zu umschalten der Komponente, die ebenfalls aus der Konfigurationsdatei bezogen wird.

4.3.3.2 Aufbau des Web-Servers

Der REST-basierte Web-Server ist mit der JAX-RS Referenzimplementierung Jersey realisiert worden (siehe 4.2.3). Seine Kommunikationsschnittstelle ist kompatibel zum OpenHAB-Protokoll, sodass im System keine Unterscheidung im Bezug zur Ansteuerung getroffen werden muss. Im Web-Server sind die Receiver-Codes (siehe 4.2.3) der Funksteckdosen als Ressourcen gekennzeichnet und haben eine REST-URL der Form:

- `http://<host>:<port>/homecontrol/socket/<receiver code>`

Analog zu OpenHAB hat die REST-URL zum Abfragen des Zustandes der Funksteckdosen die Form:

- `http://<host>:<port>/homecontrol/socket/<receiver code>/state`

Da es keine Möglichkeit gibt den Zustand der Funksteckdose direkt zu ermitteln (unidirektionale Verbindung vom Remote-Bricklet zu Funksteckdose), speichert der Web-Server Zustände intern ab, was bedeutet, dass beim Start des Servers eine Synchronisierung mit den Funksteckdosen erfolgen muss. Der Server ist so implementiert, dass er Zustände der ersten Vier Funksteckdosen der ersten Gruppe (siehe 4.1.4) verwalten kann.

Der Zustand der Ressource kann ebenfalls analog zu OpenHAB mittels einer POST-Anfrage geändert werden, mit dem Unterschied, dass der Web-Server nur EIN und AUS Anfragen entgegennimmt. Diese leitet er mittels Tinkerforge-Schnittstelle weiter an das Master-Brick.

4.3.4 Konfiguration

Das System bezieht alle für den Betrieb erforderlichen Informationen anhand einer hierarchisch strukturierten Konfigurationsdatei in Form von XML. Die Datei besteht aus ein Wurzelement [6] „configuration“ und dieses schließt vier weitere Elementen ein:

roomDepth: Dieses Element muss mit einen Zahlenwert belegt werden der die Raumtiefe angibt in mm. Er ist erforderlich für die Unterteilung der Wände im Raum (z_{\max} in Abschnitt 4.3.1.2) und wird vom System automatisch geändert, sobald ein Benutzer sich tiefer im Raum befindet als dieser Wert.

roomHeight: Das Element muss mit einen Zahlenwert belegt werden der die Raumhöhe angibt ebenfalls in mm (y_{\max} in Abschnitt 4.3.1.2).

roomWidth: Das Element beinhaltet die Raumhöhe in mm (x_{\max} in Abschnitt 4.3.1.2).

Das vierte Element ist „set“ und kann beliebig oft in der Datei vorkommen. Das Element beinhaltet Konfigurations-Sätze und muss oder kann folgende Unterelemente beinhaltet:

wall : Jeder Konfigurations-Satz beinhaltet genau eine Wand und kann mit den Werten FRONT, REAR, LEFT, RIGHT und CEILING belegt werden. Das Element beinhaltet die Information der Wand an der eine spezifizierte Komponente positioniert ist.

heightLocation: Das Element wird mit den Werten TOP und BOTTOM belegt, kann aber auch vernachlässigt werden. Mit dieser Information kann die Wand in oberen und unteren Bereich unterteilt werden (Abschnitt 4.3.1.2).

widthLocation: Wird mit den Werten LEFT und RIGHT belegt, kann aber ebenfalls, wie das heightLocation-Element vernachlässigt werden. Mit dieser Information kann die Wand in linken und rechten Bereich unterteilt werden.

itemURL: Das Element beinhaltet die REST-URL der anzusteuernenden Komponente (Abschnitt 4.3.3).

onCommand: Das Element beinhaltet die Post-Anfrage zum Einschalten der Komponente (bei OpenHAB-items kann dieser Wert variieren).

OffCommand: Das Element beinhaltet die POST-Anfrage zum Ausschalten der Komponente.

accretionValue: Das Element beinhaltet den Namen eines Add-ons. Falls ein Konfigurations-satz ein accretionValue statt einer REST-URL, ein onCommand und ein offCommand beinhaltet, startet ein Add-on (Abschnitt 4.3.6) sobald ein Nutzer in Richtung der angegebenen Wand zeigt.

Die Datei kann von dem in Abbildung 4.5 veranschaulichten Konfigurationsprogramm geladen und verwaltet werden. Dieses stellt eine visuelle Ansicht der Datei zu Verfügung und bietet den Nutzer die Möglichkeit alle Elemente der Datei zu bearbeiten oder neue zu erstellen. Abbildung 4.10 veranschaulicht die GUI des Programms und zeigt die Bereiche in den Elemente und Werte der XML-Datei visualisiert sind:

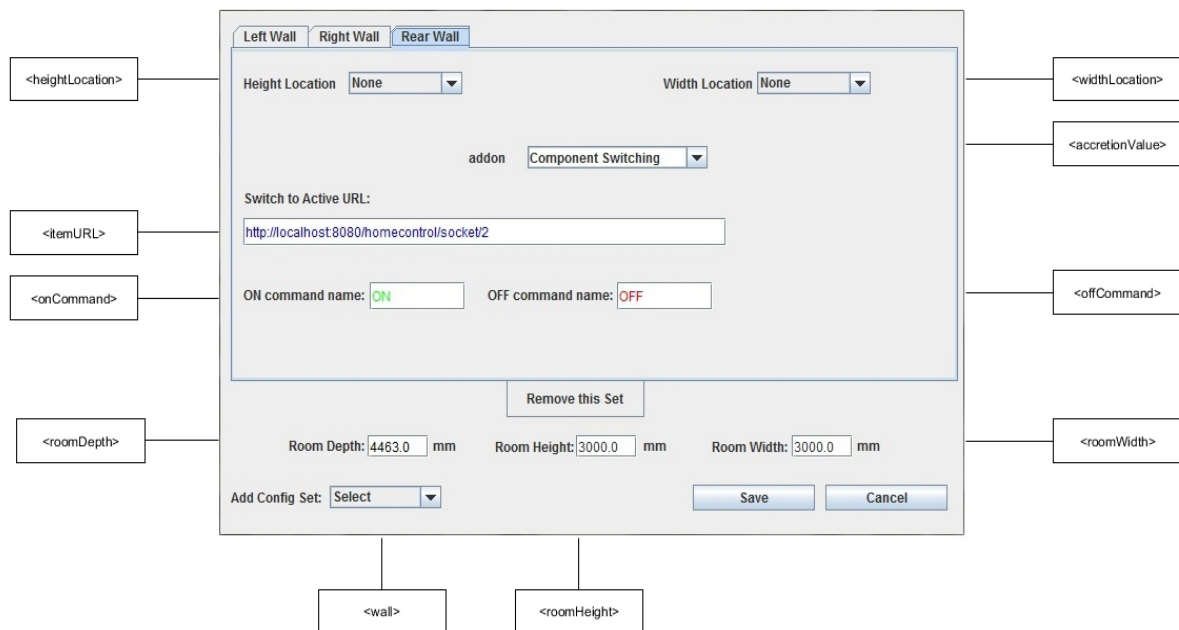


Abbildung 4.10 Konfigurationsprogramm

4.3.5 Realisierung des Panels

Das Panel wurde mithilfe von OpenNI's User- und Depthgenerator (Abschnitt 4.2.2) realisiert. Der Depthgenerator stellt die erfassten Tiefenwerte des Sensors zur Verfügung, sodass sie vom Panel bezogen werden können um die Szene zu erstellen. Der Usergenerator enthält dabei die Information zu jedem Pixel, ob es zu einem der Nutzer gehört oder Teil der Szene ist [18 S.21]. Auf diese Weise werden im Panel alle erkannten Nutzer farblich hervorgehoben. Abbildung 4.11 zeigt wie im Panel die Szene und ein erkannter Nutzer dargestellt werden. Die Linien im Oberbereich des Nutzers werden mithilfe von Koordinatenwerten (x,y) der Körperabschnitte gezeichnet.



Abbildung 4.11: Ausgabe des Panels

4.3.6 Erweiterbarkeit des Systems (Add-ons)

Für die Bereitstellung einer beliebig erweiterbaren Systemarchitektur wurde als Vorlage das Entwurfsmuster des State-Pattern [19 S.192] verwendet. In diesen sind Add-ons als Zustände des System gekennzeichnet und kapseln verschiedene Verhaltensweisen. Abbildung 4.12 zeigt ein Klassendiagramm-Abschnitt der Systemarchitektur, die mit Vorlage dieses Patterns realisiert wurde.

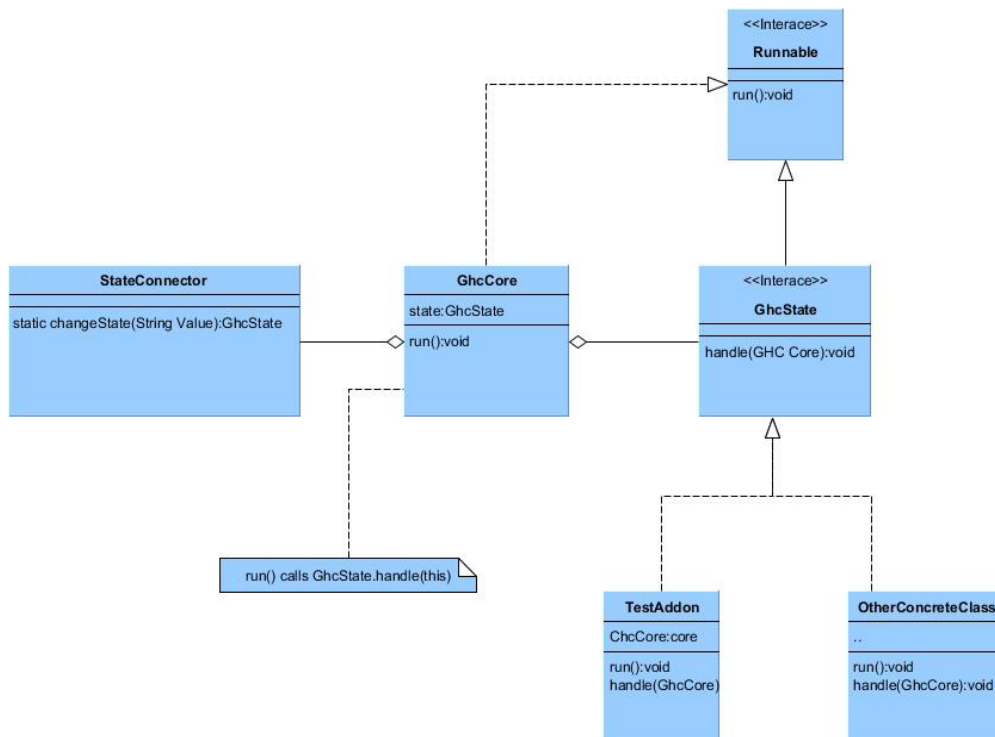


Abbildung 4.12: Klassendiagramm-Abschnitt des Systems.

Auf dem Diagramm ist zu erkennen, dass die Systemkern-Klasse(**GhcCore**) selbst nicht die Add-on-Klassen kennt, sondern lediglich das Interface(**GHCState**) aufruft sobald eine Zustandsänderung erwünscht wird und übergibt sich dabei selbst als Referenz. Konkrete Add-on-Klassen müssen dieses Interface realisieren um Aufgaben der Systemkern-Klasse übernehmen zu können. Die **StateConnerctor**-Klasse besitzt alle Objekte der Add-on-Klassen und weist sie mit den passenden Werten unter den sie auch in der Konfigurationsdatei angegeben werden können. Add-ons werden aufgerufenm wenn das System im Aktiven Modus (Abschnitt 4.4) wechselt und sind nach erfolgreicher Abarbeitung dafür zuständig es wieder (mittels Aufruf von `setToStandByMode()`) in Bereitschaftsmodus zu versetzen.

4.3.7 Betriebsmodi des Systems

Für ein ordnungsgemäßen Betrieb mit mehreren Nutzern wurde das System so konstruiert, dass es drei Betriebsmodi durchläuft. Abbildung 4.13 demonstriert diese:

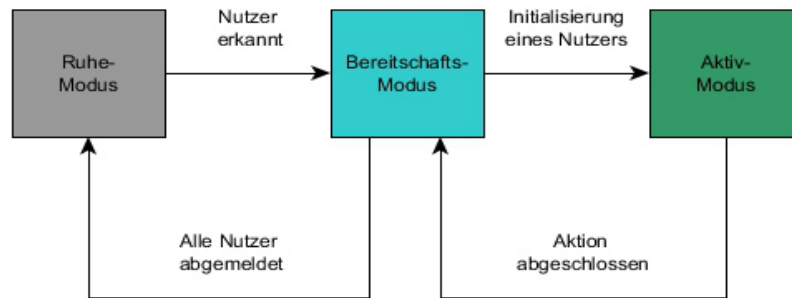


Abbildung 4.13: Betriebsmodi des Systems

Im Ruhemodus erzeugt das System keine Informationsströme bis ein Nutzer erkannt wird. Nach der Identifizierung eines Nutzers wechselt es im Bereitschaftsmodus. In diesem werden alle Nutzer getrackt und haben danach die Möglichkeit eine Initialisierungsgeste durchzuführen. Falls ein Nutzer zu winken beginnt wird er in diesem Modus fokussiert und nach erfolgreicher Interpretation der Geste gibt das System einen Bestätigungston von sich und wechselt anschließend in den Aktivmodus. In diesen Modus werden alle anderen Nutzer vom System vernachlässigt. Der fokussierte Nutzer hat im Aktivmodus die Möglichkeit auf eine oder mit zwei Armen auf zwei Komponenten zu zeigen die vom System umgeschaltet werden sollen. Nach dem Umschalten wird das System automatisch erneut im Bereitschaftsmodus versetzt. Abhängig von der Konfiguration kann der Nutzer auch auf einen Bereich zeigen um ein Add-on starten zu können. Das System verweilt dann im aktiven Modus bis das Add-on es wieder im Bereitschaftsmodus versetzt (Abschnitt 4.3.6).

5. Evaluation

In diesem Kapitel wird anfangs die Systemevaluation vorgestellt in der das Schnittpunkt-Berechnungsverfahren im Bezug zu den Zeigegesten bewertet wird und anschließend die Tracking-Robustheit, die mit mehreren Nutzern erprobt wurde, erläutert. Im letzten Teil des Kapitels wird schließlich eine Benutzerstudie vorgestellt, die im Rahmen dieser Arbeit durchgeführt wurde.

5.1 Systemevaluation

Durch das im Abschnitt 4.3.3.1 vorgestellte Schnittpunkt-Berechnungsverfahren ist es gelungen Zeigegesten von Benutzern systematisch zu berechnen. Die Voraussetzung ist, dass der Sensor parallel zu den Wänden und mittig im Raum positioniert sein muss, damit die ermittelten Werte des Systems mit den der realen Wänden übereinstimmen können. Ist diese nicht gegeben, kann es zu Fehlinterpretationen des Systems bei der Deutung von Zeigegesten kommen. Des Weiteren muss für die Zeigegesten-Erkennung eine präzise Tracking-Genauigkeit des Nutzers vorhanden sein, die in einigen Situationen nicht vorhanden ist. In der Benutzerstudie (Abschnitt 5.4) konnte so eine Situation festgestellt werden, wenn eine Komponente hinter den Nutzer positioniert ist und dieser sich umdrehen muss um auf sie zu zeigen. In manchen Fällen verliert der Sensor den Arm des Nutzers, sodass das System die Zeigegeste nicht korrekt interpretieren kann.

Mit dem von NITE genutzten Winkgesten-Erkennungsverfahren (vorgestellt in Abschnitt 4.3.2.2) ergeben sich erfahrungsgemäß keine konkreten Schwierigkeiten und hat sich als robuste Lösung zur Initialisierung erwiesen.

Die Robustheit des Trackings wurde mit Hilfe von Fünf Benutzern, erprobt. Dazu positionierten sich alle nebeneinander vor dem Sensor (siehe Abbildung 5.1). Mit Hilfe des Panels (Abschnitt 4.3.5) konnte festgestellt werden, dass bei dichter Positionierung der Benutzer die Tracking-Genauigkeit abnahm, sodass manche Zeigegesten vom System nicht berücksichtigt werden konnten.

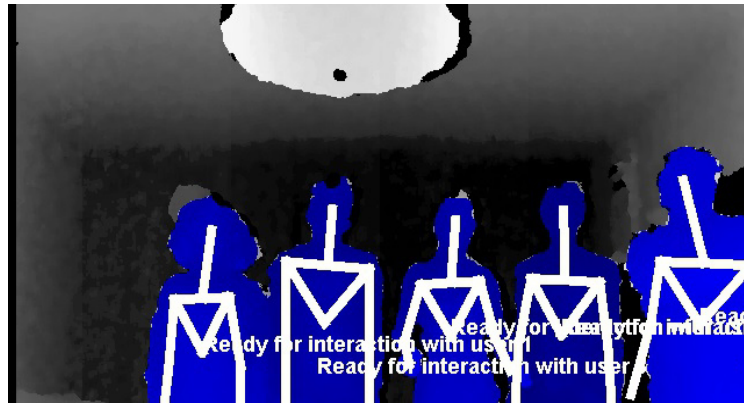


Abbildung 5.1: Ausgabe des Panels mit 5 Nutzern.

5.2 OpenHAB Kompatibilität

OpenHAB stellt eine Demo zu Verfügung [5] die mit der Kommunikations-Schnittstelle des Systems verknüpft werden kann. Um dies zu Überprüfen wurde die Demo gestartet und folgende REST-URL's und dazugehörige POST-Kommandos (siehe Abschnitt 4.3.4) von Demo-items in der Konfigurationsdatei übernommen:

- http://localhost:8080/rest/items/Light_FF_Bath_Ceiling
onCommand: ON offCommand: OFF
- http://localhost:8080/rest/items/Light_FF_Bath_Mirror
onCommand: ON offCommand: OFF
- http://localhost:8080/rest/items/Heating_FF_Bath
onCommand: 100, offCommand: 0

Zur Darstellung der Demo-Sitemap wurde folgender Link in einem Internet-Browser eingefügt:

- <http://localhost:8080/openhab.app?sitemap=demo>

Anhand der Darstellung der Items in der Sitemap konnte so festgestellt werden, dass diese vom System ansteuerbar waren.

5.3 Benutzerstudie

Im Rahmen dieser Arbeit wurde ein Versuch durchgeführt, indem das System mit Hilfe von insgesamt 6 Probanden auf qualitative Ebene und Praxistauglichkeit getestet wurde.

5.3.1 Versuchsbeschreibung

Das System wurde in einem herkömmlichen Wohnzimmer aufgestellt in dem insgesamt drei installierte Funksteckdosen über den Web-Server anzusteuern waren (Abbildung 5.2). An den Funksteckdosen wurden Leuchten angeschlossen die links, rechts und auf der vorderen Seite vom Sensor positioniert wurden. Die Probanden wurden aufgefordert einzeln den Raum zu betreten um den Versuch zu starten ohne dabei zu wissen, wie das System funktioniert. Jeder Person wurde erklärt, dass das System mithilfe von Gesten drei Leuchten im Raum ansteuern kann. Zusätzlich wurden die Probanden darauf hingewiesen, dass es sich um ein Gehörloses System handelt, welches nur über ein Auge verfügt (Sensor) und dessen Aufmerksamkeit visuell erregt werden muss um Anweisungen zu befolgen.

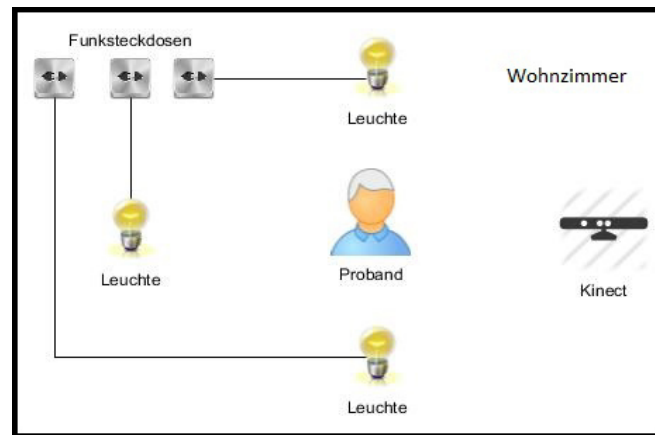


Abbildung 5.2: Schematische Zeichnung des Versuchortes

5.3.2 Versuchsdurchführung

In der Durchführung des Versuchs wurde den Probanden die Aufgabe gestellt herauszufinden, wie das System initialisiert werden kann und anschließend sie es dazu bringen erst eine Leuchte dann zwei gleichzeitig anzusteuern. Anschließend wurden alle Probanden befragt.

Während der Versuchsdurchführung konnte festgestellt werden, dass alle Probanden ohne Vorwissen in Richtung des Sensors sahen und anfangen zu winken. Als der Bestätigungston des Systems erklang, zeigten alle Probanden ohne zu zögern in Richtung der Leuchte. Als sie gebeten worden eine Lösung zu finden, wie zwei Leuchten gleichzeitig angesteuert werden können, zeigten sie mit gestreckten Armen auf linke und rechte Leuchte, womit alle Probanden die Aufgabe lösten.

Bei dem Versuch der Probanden auf die hinter ihnen befindliche Leuchte zu zeigen, konnte wie in Abschnitt 5.1 erwähnt, festgestellt werden, dass das System Probleme hatte, dies zu bewerkstelligen. Diese Aktion ist nur mit zwei Probanden gelungen.

Im folgenden werden alle gestellten Fragen und dazu gehörige Ergebnisse vorgestellt:

- Frage 1: Ist das System leicht zu bedienen und Selbsterklärend?

Alle Probanden beantworteten diese Frage mit Ja und gaben auch an, dass es ein sehr leicht zu bedienendes System sei.

- Frage 2: Kann dich das System bei der Unterstützung von Alltagsaufgaben behilflich sein ohne dich dabei unnötig zu belasten?

Auf diese Frage antworteten ebenfalls alle Probanden mit Ja.

- Frage 3: Spricht dich die Gestenbasierte Ansteuerung an oder bevorzugst du andere Eingabegeräte?

Alle Probanden gaben an, dass die gestenbasierte Ansteuerung zu bevorzugen ist und begründeten, dass es Lustig und was Neues sei. Einige Probanden sagten noch dazu, dass sie auf diese Weise nicht auf Fernbedienungen etc. angewiesen sind.

- Frage 4: Würdest du auch andere Gesten für die Interaktion mit dem System bevorzugen?

Alle Probanden antworten mit nein. Sie begründeten, dass diese Gesten sehr logisch sind und das System auf diese Weise sehr intuitiv zu benutzen ist.

- Frage 5: Würdest du das System bei dir Zuhause benutzen?

50% der Probanden antworteten auf diese Frage mit nein/nicht unbedingt und gaben als Begründung an, dass diese Technologie nicht ihren persönlichen Wohnvorstellungen entspricht. Die restlichen Probanden antworteten auf diese Frage mit Ja.

5.3.3 Diskussion

Die Benutzerstudie hat gezeigt, dass die Auswahl der Gesten im System sich als gut erwiesen hat, sodass es den Anforderungen gerecht, leicht und schnell zu bedienen ist. Es konnte festgestellt werden, dass so ein gestenbasiertes System anderen Eingabegeräten gegenüber zu bevorzugen ist, zumal es Nutzern leicht fällt es zu bedienen aber auch größeren Spaß bereitet. Damit besteht das Potenzial das System im Alltag etablieren zu können um Tätigkeiten -- auf eine innovative Weise -- zu bewältigen.

Allerdings kann aus Frage 5 schlussgefolgert werden, dass nicht alle Probanden aufgeschlossen gegenüber neuartigen Technologien wie diese sind, obwohl sie keine Probleme mit dem Umgang des Systems hatten.

6. Fazit

6.1 Ausblick

Die bereitgestellte Architektur gewährleistet die Erweiterbarkeit des Systems. Durch das hinzukommen von neuen technisch fortgeschrittenen und komplexeren Komponenten könnte das System mit Hilfe der vorgestellten Add-ons angepasst werden. Mit zunehmender Funktionalität kann es dann auch in anderen Alltagsbereichen eingesetzt werden.

Mit einer Schnittstelle zu weiteren Sensoren besteht das Potenzial, das System auch in vielen Räumen verteilt zu nutzen. So eine Bereicherung würde den Benutzer noch mehr Komfort bieten können.

In kommender Zeit werden Tiefensensoren mit neueren Technologien ausgestattet sein, womit sich die Tracking-Genauigkeit enorm verbessern wird. Mit der Schnittstelle zu OpenNI (Abschnitt 4.2.2) können modernere Sensoren auch im System dieser Arbeit verwendet werden. Auf diese Weise können die in Abschnitt 5.1 und 5.2, beschriebenen Probleme behoben werden.

6.2 Zusammenfassung

Im Rahmen dieser Arbeit ist es gelungen ein gestenbasiertes System zur Ansteuerung von schaltbaren Komponenten zu entwickeln. Durch planmäßige und konsequente Konzeption konnten alle Anforderungen erfüllt werden, sodass es im jetzigen Entwicklungsstand im Wohnalltag etabliert und für die Bewältigung von Tätigkeiten benutzt werden kann.

Mit der entwickelten Kommunikations-Schnittstelle zu OpenHAB ist ein hohes Maß an Flexibilität erreicht worden, indem das System mit vielen Komponenten kombiniert werden kann.

Die Gesten, die dem Benutzer zur Interaktion mit dem System zur Verfügung stehen, gleichen denen, die Personen zur Kommunikation nutzen. So wurde erreicht, das System auf einer möglichst intuitiv bedienbare Ebene zu implementieren. Diese Tatsache wurde durch die Nutzerstudie bestätigt.

Diese Arbeit konnte mit der Realisierung des Systems zeigen, wie technische Unterstützung im Alltag mit gestenbasierte Funktionalität erweitert werden konnte. Anhand der vorgestellten Studie konnte nachgewiesen werden, dass diese Funktionalität Komfort und Bequemlichkeit einbringt.

A.Literaturverzeichnis

- [1]: "Microsofts "Kinect" Schafft Es Ins Guinness Buch Der Rekorde." *WAZ*. N.p., n.d. Web. 10 Aug. 2014. <<http://www.derwesten.de/spiele/news/microsofts-kinect-schafft-es-ins-guinness-buch-der-rekorde-id4428517.html>>.
- [2]: "Fraunhofer-Allianz Ambient Assisted Living." *Fraunhofer-Gesellschaft*. N.p., n.d. Web. 10 Aug. 2014. <http://www.fraunhofer.de/de/institute-einrichtungen/verbuende-allianzen/Ambient_Assisted_Living.html>.
- [3]: "Natural User Interfaces - Die Kunst, Nutzung Intuitiv Zu Gestalten." *Webmagazin*. N.p., n.d. Web. 10 Aug. 2014. <<http://webmagazin.de/design/user-experience/Natural-User-Interfaces-Kunst-Nutzung-intuitiv-zu-gestalten>>.
- [4]: "Smart Home." #49. N.p., n.d. Web. 10 Aug. 2014. <<http://she777.com/smart-home/49/smart-home/>>.
- [5]: "Welcome to OpenHAB - a Vendor and Technology Agnostic Open Source Automation Software for Your Home. Build Your Smart Home in No Time!" *OpenHAB*. N.p., n.d. Web. 10 Aug. 2014. <<http://www.openhab.org/>>.
- [6]: "XML Und XML-Derivate." *SELFHTML: Einführung / Web-Technologien /*. N.p., n.d. Web. 10 Aug. 2014. <<http://de.selfhtml.org/intro/technologien/xml.htm>>.
- [7]: "Kinect." *Kinect*. N.p., n.d. Web. 10 Aug. 2014. <<http://kinect.mkoetting.de/>>.
- [8]: "ZigBee Alliance Home." *ZigBee Alliance Home*. N.p., n.d. Web. 10 Aug. 2014. <<http://www.zigbee.org/>>.
- [9]: "FeelingSecureis EasywithZ-Wave." *Z-Wave : Home Control*. N.p., n.d. Web. 10 Aug. 2014. <<http://www.z-wave.com/>>.
- [10]: "News." *KNX*. N.p., n.d. Web. 10 Aug. 2014. <<http://www.knx.de/knx-de/>>.
- [11]: "Startseite – Tinkerforge." *Startseite – Tinkerforge*. N.p., n.d. Web. 10 Aug. 2014. <<http://www.tinkerforge.com/de/>>.
- [12]: "OSGi Community Event 2014." *OSGi Alliance*. N.p., n.d. Web. 10 Aug. 2014. <<http://www.osgi.org/Main/HomePage>>.

- [13] "Kinect: How to Install and Use OpenNI on Windows - Part 1." *Yannick Lorient*. N.p., n.d. Web. 10 Aug. 2014. <<http://yannickloriot.com/2011/03/kinect-how-to-install-and-use-openni-on-windows-part-1/>> .
- [14] "Java API for RESTful Services (JAX-RS)." — *Project Kenai*. N.p., n.d. Web. 10 Aug. 2014. <<https://jax-rs-spec.java.net/>>.
- [15] "Jersey." *Jersey*. N.p., n.d. Web. 09 Aug. 2014. <<https://jersey.java.net/>>.
- [16] "JDOM." *JDOM*. N.p., n.d. Web. 10 Aug. 2014. <<http://www.jdom.org/>>.
- [17] "The Streaming API for XML (StAX)." *StAX*. N.p., n.d. Web. 10 Aug. 2014. <<http://stax.codehaus.org/>>.
- [18] Davison, Andrew. *Kinect Open Source Programming Secrets: Hacking the Kinect with OpenNI, NITE, and Java*. New York: McGraw-Hill, 2012. Print.
- [19] Goll, Joachim, and Manfred Dausmann. *Architektur- Und Entwurfsmuster Der Softwaretechnik: Mit Lauffähigen Beispielen in Java*. Wiesbaden: Springer Vieweg, 2014. Print.
- [20] "Galileo Computing :: Java 7 - Mehr Als Eine Insel - 13 RESTful Und SOAP Web-Services." *Galileo Computing :: Java 7 - Mehr Als Eine Insel - 13 RESTful Und SOAP Web-Services*. N.p., n.d. Web. 10 Aug. 2014. <http://openbook.galileocomputing.de/java7/1507_13_002.html>.
- [21] "Java+You, Download Today!" *Java.com: Java + You*. N.p., n.d. Web. 10 Aug. 2014. <<https://www.java.com/en/>>.

B.Versicherung

Versicherung gemäß § 10a, Absatz 5 der Prüfungs- und Studienordnung für das Bachelorstudium (BPO) an der Technischen Fakultät der Universität Bielefeld vom 31. März 2009.

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig erarbeitet und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Bielefeld, den 11. August 2014

Antonios Zaroulas

C.Material

Compact Disc

Inhalt:

- Quellcode der Arbeit (GestureHomeControl, SocketControllServer, GhcConfigurator)
- Arbeit als PDF