Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Benutzerzentrierte Entwicklung eines Systems zur ortsensitiven Steuerung von Elementen eines Smart Homes

von

Nicolas Mehlei

als
Forschungsprojekt

betreut von
Prof. Dr.-Ing. Johann Habakuk Israel
Hochschule für Technik und Wirtschaft Berlin

Berlin, 03. April 2018

Inhaltsverzeichnis

Abkürzungen

BTA Biologisch-Technische/r Assistentin/in

CQRS Command-Query Responsibility Segregation

DTO Data Transfer Object

ES Event Sourcing

GUID Globally Unique Identifier

IR Infrarot

JSON JavaScript Object Notation

MQTT Message Queuing Telemetry Protocol

PIR Passives Infrarot

UI User Interface

1. Einleitung

Das eigene Heim ist etwas sehr Persönliches und Individuelles. Für seine eigene Wohnung oder eigenes Haus investiert ein Mensch üblicherweise einiges an Zeit, um diese aus einer schier endlosen Zahl an Gestaltungsmöglichkeiten basierend auf seinem ganz subjektiven Geschmack und seinen Anforderungen gemäß einzurichten.

Diese vielfältigen Möglichkeiten existieren leider nicht in den Verhaltens-Konfigurationmöglichkeiten gängiger Smart Home Lösungen oder sind hinter komplexen Systemen, Oberflächen und Skript-Sprachen verborgen.

Eine Möglichkeit, auf einfache Weise Verhaltensregeln und Anpassungen für das eigene Smart Home definieren zu können, könnte die Einsteigerfreundlichkeit signifikant erhöhen und die Hürde für Smart Home Einsteiger senken.

Diese Arbeit soll den Entwurf und die Entwicklung einer solchen Möglichkeit darstellen.

1.1. Motivation

Der Autor dieser Arbeit hat bereits seit vielen Jahren großes Interesse an dem Thema der Heimautomation, war jedoch nie mit den bereits vorhandenen Lösungen zufrieden. Als Informatik-Student war der Gedanke somit nicht weit entfernt, selber an einer solchen Lösung zu arbeiten.

Die Kombination aus zwei solcher Projekte, welche zusammen eine kleine aber innovative Smart Home Lösung ergaben, kam bei jedem Besucher bisher sehr gut an. Diese Lösung ist jedoch maßgeschneidert entwickelt worden und nur schwer an die divergenten Anforderungen und Wünsche potenzieller weiterer Nutzer anpassbar.

Diese Arbeit soll helfen, diese Lösung weiter voranzutreiben, mehr Anpassbarkeit an den jeweiligen Nutzer zu bieten sowie weitere Funktionalität im Bereich der ortsensitiven Steuerung hinzuzufügen.

2. Situation

Smart Home ist zwar ein eher neuer Begriff, jedoch ist das ehemals Heim- oder Hausautomation genannte Thema keineswegs neu. Dementsprechend gibt es eine Vielzahl vorhandener Lösungen für die Probleme eines Smart Homes.

2.1. Vorhandene Lösungen

Viele der bereits vorhandenen Systeme beschränken sich hierbei jedoch auf einen spezifischen Teilbereich.

Beispielsweise können HomeMatic-basierte Sensoren üblicherweise nur HomeMatic-Aktoren ansprechen. Ein weiteres Beispiel ist das bekannte Lichtsteuersystem Philips Hue, welches z.B. die Definition von (trivialen) Regeln erlaubt für die Steuerung von (Philips Hue-basierten) Leuchten anhand von (Philips Hue-basierten) Bewegungsmeldern. Dies erlaubt zwar eine leichte Anpassung an die eigenen Bedürfnisse, jedoch besteht keine Möglichkeit diese Steuerung auf mehr als nur Lichtquellen (oder eine andere Art von Lichtquellen) auszuweiten.

Diese Vielzahl an Insellösungen sorgt für eine hohe Fragmentierung des Smart Home Marktes. Einen Gegenpol hierzu stellen sogenannte Meta-Systeme dar, welche Teilbereich-übergreifend versuchen, mehrere vorhandene und eigene Lösungen für eine höhere Gesamtfunktionalität zusammenzufassen. Hierzu soll nachfolgend eine Übersicht über solche Meta-Systeme gegeben werden sowie deren potenziell vorhandenen Regelsysteme geprüft werden.

2.1.1. openHAB

Der **open H**ome **A**utomation **B**us (in kurz: openHAB) ist eine in Java entwickelte Open Source Smart Home Plattform. Mit einem Schwerpunkt auf Technologie-agnostischer Anbindung von Komponenten wird hier versucht, eine möglichst breite Anzahl an Komponenten verfügbar zu machen. [17]

Der innere Kern von openHAB wurde abgespalten als Eclipse SmartHome und ist somit inzwischen ein Framework der Eclipse Foundation. [5]

OpenHAB besitzt ein vorhandenes Regelsystem, welches auf einer eigenen Skriptsprache basiert. Es ist jedoch noch in einem experimentellen Stadium und ist durch die Code-basierte Definition ausschließlich für technisch-versierte Benutzer relevant und nutzbar.

2.1.2. FHEM

FHEM, kurz für "**F**reundliche **H**ausautomation und **E**nergie-**M**essung", ist ein in Perl entwickelter Heimautomatisierungsserver. Er unterstützt eine Vielzahl vorhandener Geräte, u.a. Lichtsteuerungssysteme und Sensoren und wird via Web-Oberflächen oder Apps verwendet.

Die Konfiguration von FHEM erfolgt ausschließlich über Konfigurationsdateien, inklusive Perl- und Shell-Skripte. Dies schliesst auch Automatisierungen ein, mit welchen regelähnliche Einstellungen programmiert werden können. [20]

2.2. Problematik

Keins der genannten Systeme bietet eine Möglichkeit, Benutzern die Definition von Regeln zu ermöglichen ohne auf die Verwendung von Skript- oder Programmiersprachen zurückzugreifen. Da dies nicht technisch-versierte Benutzer somit ausschließt, solche Systeme entsprechend nutzen zu können, besteht hier Potenzial dafür, eine Lösung zu schaffen, welche von einer größeren Zielgruppe verwendet werden kann.

2.3. Lösungsansatz

Mögliche Lösungsansätze hierbei wären entweder eine der genannten vorhandenen Lösungen oder eine eigene Lösung dahingehend zu erweitern. Da eine Veränderung einer fremden Lösung weitaus aufwändiger wäre, ohne einen direkt ersichtlichen Vorteil zu bieten, wurde die Modifikation der Eigenentwicklung ausgewählt.

Die genannte eigenentwickelte Lösung hat die Projektbezeichnung *JarvisSuite* und ist ein Verbund aus zwei durchgeführten Projekten. Der Kern ist eine wohnungslokale Anwendung, welche als Meta-Lichtsteuerungssystem fungiert und derzeit die Steuerung von bis zu sechs verschiedenen Lichtsteuerungssystemen vereinen, erweitern und bündeln kann. Dies erlaubt es, z.B. spezifische und angepasste Licht-Situationen auf Knopfdruck hervorzurufen (mit spezifischen Lichtintensitäten- und Farben für jede einzelne Lichtquelle im Raum). Dieser Kern wird erweitert durch eine Tablet-Oberfläche, welche die Steuerung des Systems an zentralen Stellen in einer Wohnung (oder in einem Haus) erlaubt, sowie angepasster Lichtschalter, welche die vorhandenen konventionellen Lichtschalter in einer Wohnung durch erweiterte Alternativen ersetzt und diese ebenfalls an das System koppelt. [14] [15]

Passend zu der (temporären) Bezeichnung *JarvisSuite* kann diese Erweiterung die Bezeichnung *Jarvis-RuleEngine* bekommen.

3. Analyse

In diesem Kapitel sollen die Anforderungen des in Abschnitt 2.3 skizzierten Lösungsansatzes analysiert werden.

3.1. Zielgruppe

Zunächst muss erörtert werden, an welche Zielgruppen bzw. Benutzergruppen sich eine potenzielle Lösung orientieren soll.

Historisch war das Thema Smart Home bzw. Heimautomatisierung eher etwas für gut situierte Eigenheimbesitzer sowie ein Bastelprojekt für jüngere technisch-versierte Menschen. In den letzten Jahren hat sich dies jedoch gewandelt und gerade die jüngeren Zielgruppen fühlen sich immer mehr angesprochen von Smart Home Lösungen bzw. sind eher gewillt, diese in ihren eigenen Wohnungen oder Häusern anzuwenden. [10] [3]

Dabei kann ein Smart Home prinzipiell von allen (erwachsenen) Altersgruppen angewendet werden. Dementsprechend wird in dieser Arbeit versucht, sich nicht auf spezifische Altersgruppen zu beschränken, sondern eine möglichst einfach anzuwendende Lösung zu konzipieren welche wenig technisches Know-how voraussetzen soll und somit von einer breiten Masse verwendet werden kann.

3.2. Interviews

Um die Anforderungen herauszufinden, welche von den Nutzern später an die Software gestellt werden, wurden Interviews mit mehreren beispielhaften Nutzern geführt. Die Interview-Methode wurde gewählt, da sie durch die private Atmosphäre wichtige Fragen erlaubte, deren Antwort dem Teilnehmer[1] in einer Gruppe (bspw. Fokus-Gruppe) ggf. schwerer fallen würde, z.B. betreffend detaillierterer täglicher Abläufe sowie potenzieller nächtlicher Toilettenbesuche.

3.2.1. Ablauf

Das Interview wurde stets damit begonnen, den Teilnehmer darum zu bitten, sein eigenes Heim (sei es eine Wohnung oder ein Haus) zu skizzieren oder zu beschreiben. Dies war hilfreich, da viele Aussagen und Anforderungen des Teilnehmers hierdurch leichter nachzuvollziehen waren. Insbesondere wie verschiedene Räume ineinander übergehen oder wie die Positionierung von Fenstern ist, war von Interesse.

Der nächste Schritt war, den Teilnehmer einige seiner/ihrer üblichen Abläufe beschreiben zu lassen, wie z.B.

- Wenn Teilnehmer nach Hause kommt, nach einem normalen Arbeitstag
- Wenn Teilnehmer nach Hause kommt, nach einer beliebigen Aktivität
- Wenn Teilnehmer schlafen geht
- Wenn Teilnehmer morgens aufsteht, an einem normalen Arbeitstag

[1] Im Folgenden wird die männliche Form für weibliche und männliche Teilnehmer verwendet.

- Wenn Teilnehmer morgens aufsteht, an einem Wochenendtag
- Wenn Teilnehmer nachts aufsteht

Gefolgt wurde dies von der Bitte, den allgemeinen Umgang mit den Themen Licht und Temperatur zu beschreiben.

Abschließend gingen die Interviews in offene Fragen (mit gezielten Nachfragen) über, wie mögliche Optimierungswege der Abläufe aussehen bzw. gewisse Aspekte automatisiert werden könnten. Dies resultierte in Ideen, welche gute Kandidaten für die angezielten Regeln darstellten. Diese wurden zusammen als Karte verfasst ("Karten-Methode"), um die spezifische Anforderung des Teilnehmers als Anwendungsfall festzuhalten. Die Interviews haben im Durchschnitt etwa eine Stunde gedauert.

3.2.2. Teilnehmer

Für die Auswahl der Teilnehmer wurde versucht, eine Mischung aus verschiedenen Beziehungsstatus und Wohnungssituationen zu finden, um bestmöglich anhand von wenigen Teilnehmern einen möglichst großen Bereich an späteren User-Fällen abzudecken. Eine Liste ist in Tabelle 3.1 zu finden.

Teilnehmer	Beziehungsstatus	Wohnungssituation	Geschlecht	Alter	Beruf
Teilnehmer A	Single	alleine wohnend	weiblich	27	Personal
Teilnehmer B	In Beziehung	alleine wohnend	weiblich	27	Lehrerin
Teilnehmer C	In Beziehung	mit Partnerin wohnend	männlich	29	Entwickler
Teilnehmer D	Familie mit Kindern	Eigenes Haus	männlich	35	Unternehmer

Tabelle 3.1.: Liste der Teilnehmer

3.2.3. Ergebnisauswertung

Die Interviews gaben einen sehr guten Einblick in die Verwendung der jeweils eigenen Wohnungen und Häuser der Teilnehmer und zeigte, dass die Wünsche und Anforderungen der Teilnehmer sehr vielfältig und divergent sein können. Bereits bei der kleinen Menge an vier Teilnehmern zeigten sich starke Unterschiede, wie Menschen zu einigen Grundthemen standen. Beispielsweise bevorzugte Teilnehmerin A ständig hell erleuchtete Räume (mit vorzugsweise Deckenlicht), wohingegen Teilnehmerin B eher dauerhaft gedimmtes Licht präferiert und sehr abgeneigt dem Deckenlicht gegenüber ist.

Bei den Interviews kristallisierte sich heraus, dass jeder Teilnehmer genau eine Idee für eine Regel hatte, die ihm/ihr sehr wichtig war bzw. die sich der jeweilige Teilnehmer für seine eigene Wohnung oder sein Haus wünschen würde. Diese waren:

1. Der Bewohner soll gewarnt werden, wenn er versucht die Wohnung zu verlassen und es noch geöffnete Fenster gibt.
2. Der Bewohner soll bei Dunkelheit mit stark gedimmten Licht oder dediziertem Orientierungslicht durch die Wohnung geleitet werden.
3. Das Licht im Flur soll beim Betätigen eines Lichtschalters nur stark gedimmt angehen, wenn sich eine Person im Schlafzimmerbett befindet (um diese nicht zu wecken).
4. Im Badezimmer soll automatisch drei Minuten nach Betreten Musik angehen (und nach dem Verlassen auch wieder ausgehen).

Insbesondere die Regel Nr. 2 trifft jedoch scheinbar auf alle Teilnehmer zu, da alle vier auf die Frage, ob sie sich durch zu wenig Licht in ihrem eigenen Heim regelmäßig weh tun, schnell und energisch bestätigten. Ein Anschalten des normalen Lichts kam nächtlich für keinen der Teilnehmer in Frage, da das grelle Licht zu unangenehm wäre, weswegen eher das Risiko sich zu stoßen in Kauf genommen wird.

Teil I.

Entwicklung eines ersten Prototyps

4. Entwurf

Das Kapitel Entwurf befasst sich mit der Umsetzung der in der *Analyse* herausgefundenen Anforderungen.

4.1. Regel-Definition

Ein Regelsystem muss in der Lage sein, Situationen und Aktionen beschreiben zu können und diese in Verbindung zu einander zu setzen. Eine Regel, welche die in Abschnitt 3.2.3 genannten Anforderungen erfüllen kann, benötigt mindestens drei Bestandteile:

- Auslöser: Wann soll etwas geschehen?
- Bedingung(en): Wann soll dies gelten?
- Aktion(en): Was soll dann passieren?

Hieraus ergibt sich die in Abbildung 4.1 dargestellte Definition.

Abbildung 4.1.: Regel-Definition

Auslöser können hierbei entweder Ereignis-basiert oder Zeit-basiert sein. Bei Ereignis-basierten Auslösern ist optional eine Verzögerung konfigurierbar, welche bei Zeit-basierten Auslösern unnötig ist.

Bildet man nun die in Abschnitt 3 genannten Anforderungen und Wünsche auf dieses Schema ab und fügt der Vollständigkeit halber einige weitere hinzu, so ergibt sich die in Abbildung 4.2 gezeigte Menge an Auslösern, Bedingungen und Aktionen.

Die Bedingungen sind hierbei absichtlich direkt verwendbare Bausteine. Definierbare Bedingungen auf Basis von mathematisch-logischen Formeln und Operatoren würden mehr Flexibilität erlauben, jedoch wäre dies komplexer in der Anwendung, was durch die angepeilte Zielgruppe nicht tragbar wäre.

4.1.1. Systemregeln

Eine Unterart dieser Regel-Definition ergibt sich aus der Notwendigkeit von Systemregeln. Systemregeln werden in diesem Kontext Regeln genannt, welche nicht vom Benutzer definiert wurden, sondern automatisch generiert wurden, um stets eine Basis-Funktionalität gewährleisten zu können, z.B. zum An- und Ausschalten des Lichts mittels der normalen an der Wand montierten Lichtschalter.

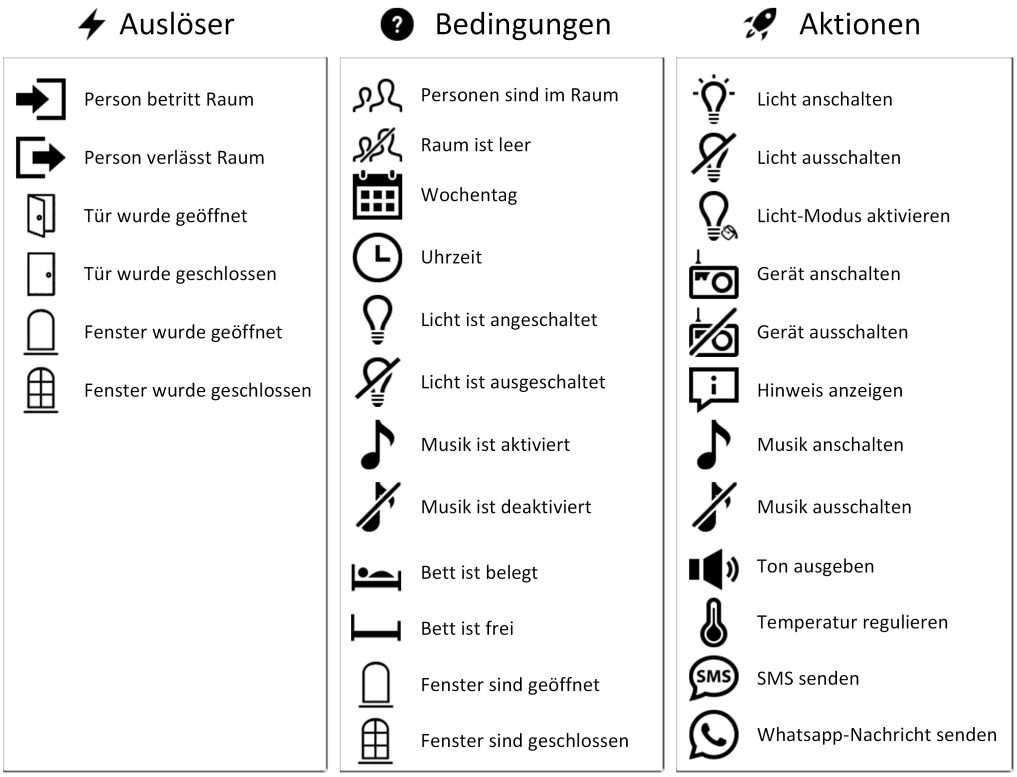

Abbildung 4.2.: Beispiele für Regel-Bestandteile

4.2. Aufbau

Das *Jarvis-RuleEngine*-System besteht aus einer zentralen Software-Komponente sowie mehreren Hardware-Komponenten.

4.2.1. Kommunikation

Die Kommunikation zwischen den einzelnen Komponenten erfolgt via MQTT. MQTT ist ein leichtgewichtiges Kommunikationsprotokoll, welches spezifisch für die Verwendung von "Internet of Things"-Anwendungen entwickelt wurde und auf dem Publish-Subscribe Messaging Pattern basiert. Dadurch eignet es sich hervorragend für die Anbindung vieler kleiner Komponenten wie z.B. Sensoren. [16]

4.2.2. Software

Die Software-Komponente erweitert die bestehende Codebasis von *JarvisSuite* um die *Jarvis-RuleEngine*. Diese besteht aus dem RuleRepository, dem RuleEngine-Manager sowie dem RuleEditor-UI.

RuleRepository

Das RuleRepository dient der Persistierung der im System gespeicherten Regeln. Diese werden als JSON-Daten serialisiert und lokal gespeichert.

Dem Prinzip des Datenbank-agnostischen Designs folgend sind die hierauf aufbauenden, weiteren Module vollkommen vom tatsächlichen Speicherort und der Speicherart abstrahiert. Dies erleichtert es z.B. später, nicht persistierte Regeln (z.B. automatisch generierte temporäre Regeln) zu haben oder stattdessen eine Datenbank als Speicherort zu verwenden. [9]

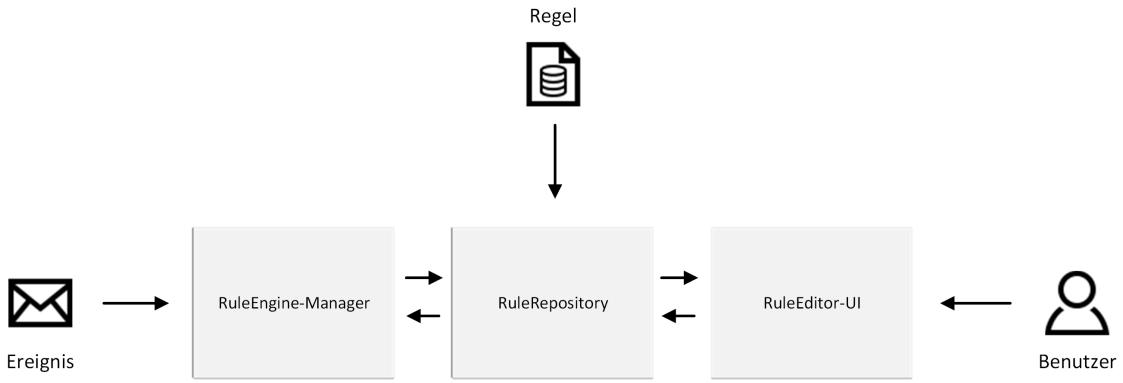

Abbildung 4.3.: Aufbau

RuleEngine-Manager

Der RuleEngine-Manager ist das Kernstück der *Jarvis-RuleEngine*. Er reagiert auf alle Ereignisse, welche im System generiert werden (bspw. durch Bewegung einer Person), überprüft, ob dies eine der vorhandenen Regeln betrifft, und führt diese entsprechend aus.

RuleEditor-UI

Das RuleEditor-UI ist der einzige für den Benutzer direkt sichtbare Aspekt der *Jarvis-RuleEngine*. Diese Bedienungsoberfläche erweitert die bereits vorhandene Oberfläche der *JarvisSuite* um Funktionen zum Hinzufügen und Verwalten von Regeln.

Die Farbgestaltung des UI soll hierbei eher dunkel gestaltet werden. Da das UI zu jeder Uhrzeit verwendet werden kann, typische Bildschirme (insbesondere LCDs) sich jedoch nur begrenzt dimmen lassen, ist es bei einem hellen UI unvermeidlich, dass dies zu nächtlichen Uhrzeiten den Benutzer zu sehr blenden würde.

Darüber hinaus sollen – soweit möglich – Icons zur Unterstützung der Gestaltung verwendet werden. Dies trägt nicht nur zu einer besseren Ästhetik bei (welche ebenfalls wichtig ist für eine gute User Experience), sondern hilft auch die komplexeren Aspekte leichter verständlich zu machen und somit die Einsteigerfreundlichkeit und Bedienbarkeit zu steigern.

Die effektive Gestaltung dieses UI ist maßgeblich entscheidend über die spätere Anwendbarkeit. Daher wird dieses prototypisch entworfen und so lange iteriert und verbessert, bis diese den gestellten Ansprüchen genügt.

Listener-Dienste

Die Listener-Dienste sind dafür da, auf die gesendeten Signale der Sensoren zu reagieren und entsprechende Ereignisse im System zu generieren. Jede Sensor-Art hat exakt einen implementierten Listener, welcher via MQTT mit den jeweiligen Sensoren kommuniziert. Die Listener besitzen hierbei eine Konfigurationstabelle, um die verwendeten Sensor-Geräte zu dem jeweiligen logischen Objekt (Raum, Tür, Fenster, etc.) zuzuordnen, wodurch die Sensoren nur ihre (ohnehin Hardwareseitig vorhandene) Seriennummer kennen müssen und somit zentral konfiguriert werden können.

Eine Alternative hierzu wäre gewesen, den Sensoren, deren technische Erklärung im kommenden Abschnitt folgt, eine eigene Konfigurationsmöglichkeit (z.B. als Weboberfläche über einen eigenen Hotspot) mitzugeben. Für einen späteren Benutzer ist es jedoch einfacher, dies zentral konfigurieren zu können und somit die Sensor-Geräte, was deren Konfiguration angeht, statuslos zu betreiben.

4.2.3. Hardware

Die Hardware-Komponenten sind die Sensoren und Aktoren des Systems. Als Implementationsbasis dienen entweder ESP8266- oder ESP32-basierte Platinen. Die integrierte Wireless LAN Konnektivität der ESP8266-Platinen erleichtert die für die Sensoren notwendige MQTT-Anbindung und darüber hinaus besitzen diese genügend Anschlussmöglichkeiten für beinahe alle geplanten Sensoren. [8]

Nur einer der Sensoren benötigte die erweiterten Anschlussmöglichkeiten des Nachfolgers, dem ESP32. Dieser bietet u.a. mehrere Analog-Digital-Wandler statt des einen des ESP8266. [7]

Theoretisch ließen sich hier auch bereits fertige Sensoren einbinden, jedoch würde dies zu hohen Kosten führen und eine Abhängigkeit auf eben diese kreieren. Des Weiteren gibt es nicht für alle benötigten Sensoren direkt erwerbliche kommerzielle Produkte. Einer späteren Einbindung von kommerziell-erhältlichen Sensoren stünde aber nichts im Wege.

Um die gewünschten Aspekte der Regeln abdecken zu können, bedarf es sowohl Sensoren, welche die ruhende als auch sich bewegende Position der Benutzer in einem Heim zu registrieren. Darüber hinaus ist es notwendig, den Status von Objekten innerhalb eines Heims, wie z.B. Fenster und Türen, registrieren zu können, um diese als Auslöser oder Bedingung in Regeln verwenden zu können.

Personengebundene Regeln werden hierbei absichtlich ausgegrenzt, da die geführten Interviews keinen hohen Bedarf daran ergaben. Im Kapitel 8 wird kurz auf solch eine Erweiterungsmöglichkeit eingegangen.

4.2.4. Personenerkennungsmethoden

Die Menge an gängigen Sensor-Arten bietet eine Reihe an Sensoren, welche eine Person erfassen kann. Diese unterscheiden sich stark in mehreren Aspekten, u.a. die Genauigkeit, die Reichweite, die Latenz sowie ob sich bewegende und/oder still verhaltende Menschen erkannt werden. [13]

Die in Frage kommenden Sensor-Arten sind:

- PIR-Sensoren
- Durchgangssensoren/Lichtschranken
- Ultraschall-Sensoren
- Aufenthalts-Sensoren

Auf Ultraschall-Sensoren wird jedoch verzichtet, da diese für die Anforderungen dieses Projekts prinzipiell dieselbe Aufgabe lösen die auch bereits Lichtschranken-basierte Durchgangssensoren bieten, mit dem Unterschied dass Ultraschall-Sensoren hierbei für die bidirektionale Erkennung doppelt verbaut werden müssten, wodurch zusätzliche Komplexität erforderlich wäre damit diese sich nicht gegenseitig stören.

Auf alle verwendeten Sensor-Arten wird nachfolgend nun weiter eingegangen.

PIR-Sensoren

PIR-Sensoren, oder auch Passive Infrarot-Sensoren, sind sehr wirksam, um sich bewegende Personen innerhalb eines Raumes zu erkennen. Durch eine verbaute Kuppel, welche für die bekannte Optik sorgt, ergibt sich ein großer abgedeckter Bereich innerhalb des verbauten Raumes. [6]

Die Vorteile von PIR-Sensoren sind somit ein großer abgedeckter Bereich sowie eine einstellbare Empfindlichkeit. Nachteilig ist jedoch, dass diese ausschließlich ein binäres Signal ausgeben können (nämlich ob Bewegung erkannt wurde, nicht aber wieviele Objekte sich bewegt haben) sowie – je nach Bauweise – eine gewisse Trägheit wann eine vor kurzem erkannte aber nicht mehr stattfindende Bewegung aufhört erkannt zu werden. [13] [12]

Durchgangssensor (Lichtschranke)

Ein Durchgangssensor, wie z.B. eine Lichtschranke, bietet die Möglichkeit, mit sehr geringer Latenz die Bewegung einer Person erkennen zu können. Wird diese an einer Tür (oder einem sonstigen Durchgang) montiert, so lässt sich durch einfaches Zählen erkennen, ob und wie viele Menschen sich in einem Raum befindet. Um die Richtung, in der sich die Person bewegt, erkennen zu können, bedarf es zwei Lichtquellen und Empfänger, um durch die kurze Verzögerung in der Unterbrechung der Lichtstrahlen die Bewegungsrichtung herauszulesen. Als Lichtquellen stehen beispielsweise Laser sowie Infrarot (IR) zur Verfügung. [13] [4]

Vorteile von Laser sind die kostengünstigeren Komponenten, die Sichtbarkeit auch ohne optische Hilfsmittel sowie die vergleichsweise leichte Fokussierung. Die Sichtbarkeit des Lasers ist jedoch im Rahmen einer Wohnung oder eines Hauses ebenfalls ein Nachteil. Des Weiteren gibt es bei unsachgemäßem Umgang gesundheitliche Bedenken, denn bei einer Betrachtung von mehr als 0,25 Sekunden kann ein Laser die Augen eines Menschen oder Tieres schädigen. [2]

Infrarot-Strahlen haben hingegen den Vorteil, dass diese nicht für das menschliche Auge sichtbar sind. Dies ist für diagnostische Zwecke jedoch auch ein Nachteil. Darüber hinaus sind diese schwerer zu fokussieren (wodurch sich zwei benachbarte Sensoren leichter beeinflussen können), haben üblicherweise eine geringere Reichweite und verschleiern ihre gesundheitliche Gefahr, da durch die fehlende Sichtbarkeit die Schutzmechanismen des menschlichen Körpers (Schließen der Augenlider) nicht greifen. [4] [2]

Als Bauweise für solch einen Durchgangssensor bieten sich Einweg-Lichtschranken oder Reflexions-Lichtschranken an. Einweg-Lichtschranken bestehen aus zwei Komponenten, einen Emitter und einen Empfänger, auf gegenseitigen Türrahmenseiten verbaut. Eine Reflexions-Lichtschranke besitzt beide Komponenten auf derselben Türrahmenseite und benötigt einen sehr spezifisch ausgerichteten Spiegel auf der gegenüberliegenden Seite.

Für die erste Lichtschranken-Version wird – zugunsten der einfacheren Bauweise – ein normaler Laser sowie die Einweg-Bauweise verwendet. Die anderen vorgestellten Möglichkeiten bieten sich jedoch als potenzielle Optimierungen für eine kommende Version an.

Tür-Sensor

Tür-Sensoren sind dafür da, erkennen zu können ob eine Tür derzeit geöffnet oder geschlossen ist, sowie wann sich dieser Zustand ändert. Die Erkennung erfolgt üblicherweise mittels Magnetkontakten, wobei ein Teil an den Türrahmen und eins an die Tür montiert wird. Ist die Tür geschlossen, so befinden sich beide Teile in unmittelbarer Nähe, was magnetisch erkannt werden kann.

Der an den Türrahmen montierte Magnetkontakt ist hierbei mit einem ESP8266-Controller verbunden, welcher den aktuellen Zustand auslesen kann und bei einem Zustandswechsel diesen via MQTT an das System weitergeben kann.

Fenster-Sensor

Der Fenster-Sensor kann Hardware-seitig äquivalent zum Tür-Sensor sein, sollte jedoch auf Software-Seite unterschiedliche Signale an das System weitergeben, damit diese differenziert werden können.

Bettaufenthalts-Sensor

Um die dritte geforderte Regel realisieren zu können, bedarf das System einer Möglichkeit, bemerken zu können, ob sich eine Person derzeit im Bett befindet.

Dies lässt sich mit einem Bettaufenthalts-Sensor (Bed occupancy sensor) realisieren. Solche Sensoren sind eher bekannt aus speziell für das altersgerechte Wohnen ausgerichtete Wohnungen, um bspw. einen Pfleger über ein ungeplantes Aufstehen der älteren Person entsprechend zu informieren. [24]

Das Funktionsprinzip lässt sich jedoch entsprechend abwandeln, um das System darüber zu benachrichtigen, damit eine Regel hierauf reagieren kann, wie dies in einer der in Abschnitt 3 gewünscht wurde.

Mithilfe derselben Touch-Controller, welche bereits in [15] zur Anwendung kamen, kann die Funktionsweise auf ein ganzes Bett hochskaliert werden. Die Erkennung erfolgt mittels kapazitiver Aufladung, wofür bei entsprechend konfigurierter Sensor-Sensibilität keine Berührung vonnöten ist. Dies ermöglicht es, mehrere entsprechend verpackte Kupferleisten über der Matratze zu verteilen, sodass eine möglichst große Fläche des Bettes erkannt werden kann. Dies erfordert zwar eine gewisse Verkabelung, jedoch sind all diese Komponenten unter dem Laken (und potenziell weiteren Textilien wie Matratzenschonern) versteckt, sodass diese nicht bemerkbar sein sollten.

Mithilfe der gleichen technischen Grundlage und Bauweise ließe sich in einer kommenden Version auch eine Art „Couch-Sensor“ entwickeln und somit die Abdeckung der Personenerkennung in einem Raum noch zusätzlich erweitern.

4.3. Skizzierung des grafischen Prototyps

Der grobe Aufbau und die Struktur verschiedener Elemente des UI wurden anhand der Regel-Definition mittels Skizzen erprobt. Diese wurden zusammen mit einem der Teilnehmer erstellt und nach bereits wenigen Versuchen ergab sich eine grobe Vorgabe, wie das UI aufgebaut sein könnte, um alle erforderlichen Eingaben und Ausgaben zu ermöglichen. Ein Bild über einige der entstandenen Skizzen ist in Abbildung 4.4 zu sehen. Als die Skizzen danach schienen, dass es so verwendbar war, wurde anhand derer der grafische Prototyp entwickelt.

Abbildung 4.4.: Skizzen

4.4. Entwicklung des grafischen Prototyps

Basierend auf den gezeichneten UI-Skizzen wurde anschließend der grafische Prototyp entwickelt. Als Realisierungsvarianten für den Prototyp wurden zwei Möglichkeiten evaluiert: Papier-Prototyp (Low Fidelity) oder grafischer Prototyp (High Fidelity). [21]

Hierbei wurde auf den grafischen Prototyp gesetzt, da hierfür weitaus mehr technische Expertise bezüglich der Bildbearbeitung vorhanden war sowie der Möglichkeit durch Copy-Paste gleichbleibende Elemente weiterzuverwenden viel Aufwand sparen kann. Des Weiteren vermittelt ein solch hergestellter Prototyp ein realistischeres Bild der späteren Anwendung. Dies wurde zusätzlich begünstigt durch den möglichen Zugriff auf mehrere Tablets, auf denen der grafische Prototyp genutzt werden kann. Der fertige grafische Prototyp auf einem solchen Tablet ist in Abbildung 4.5 dargestellt.

Zur Erstellung der Mockups wurde Adobe Photoshop und für die Kombination als klickbarer Prototyp InVision (siehe [11]) verwendet.

Abbildung 4.5.: Grafischer Prototyp auf Tablet

Inhalt des grafischen Prototyps sind der Assistent, um eine neue Regel hinzuzufügen, sowie eine Übersicht über alle vorhandenen Regeln mit der Möglichkeit, diese zu aktivieren oder zu deaktivieren.

4.4.1. Benutzergestützte Iterierung

Mithilfe der Interview-Teilnehmer wurde der grafische Prototyp Stück für Stück iteriert, um Erkenntnisse basierend auf Teilnehmermeinungen und Beobachtungen direkt einfließen und somit das User Interface zu optimieren.

Durchgehendes Farbschema

Wie in Abschnitt 4.1 beschriebenen Regel-Definition zu entnehmen ist, besteht jede Regel aus drei Bestandteilen.

Sehr früh in der Iterierung wurden drei Icons eingeführt, welche in Abbildung 4.1 abgebildet sind. Sie symbolisieren die drei Regel-Definitions-Bestandteile und sollen den Wiedererkennungswert steigern. Laut den Aussagen der Teilnehmer war dies jedoch erst der Fall, sobald diese in drei stark unterschiedlichen Farben eingefärbt wurden. Dies lässt sich dadurch erklären, dass Farben preattentiv wahrgenommen werden sowie ein starker Kontrast zu den sonst grau gehaltenen Symbolen

besteht, sodass der Benutzer das Symbol nicht zwingend betrachten und erkennen muss, um zu erkennen um welchen Bestandteil es in einem Bildschirm geht.

Vokabular

Ein Beispiel für einen sehr frühen Fund ist das Wort „Preset“. Die *JarvisSuite* verwendet dieses Wort, um einen Zustand eines Raumes zu beschreiben, inklusive aller hierfür aktivierten Lichtquellen mitsamt der spezifischen Einstellungen (Farbtemperatur bzw. Lichtfarbe sowie Intensität). Andere Lichtsteuerungssysteme verwenden hierfür auch häufig das Wort „Szene“. Beide Wörter wurden nicht auf Anhieb verstanden, was dazu führte dass der Teilnehmer entweder über das unbekannte Wort stutzte, oder die jeweiligen Schaltflächen schlichtweg ignorierte. [19]

Von einer Teilnehmerin wurde das Wort Modus als Alternative vorgeschlagen. Dies wurde von den anderen Teilnehmern als treffender bezeichnet und daher ausgetauscht.

Zurück-Funktion

Darüber hinaus wurde mehrfach der Bedarf an einer Zurück-Funktion ersichtlich, um die Korrektur von Bedienfehlern sowie Entscheidungsänderungen der Nutzers zu ermöglichen. Des Weiteren half es neuen Benutzern, die Möglichkeiten zu erkunden.

5. Implementierung

Im Zuge dieses Kapitels sollen einige ausgewählte Themen, welche sich während der Implementierung der *Jarvis-RuleEngine* ergaben, näher betrachtet werden.

5.1. Aufbau der Beispielwohnung

Während der Entwicklung und dem Test des Systems war es notwendig, dies anhand einer Beispielwohnung durchzuführen, um direkt die Machbarkeit, Angemessenheit und Funktionsfähigkeit testen zu können. In Abbildung 5.1 ist ein Grundriss dieser Beispielwohnung dargestellt.

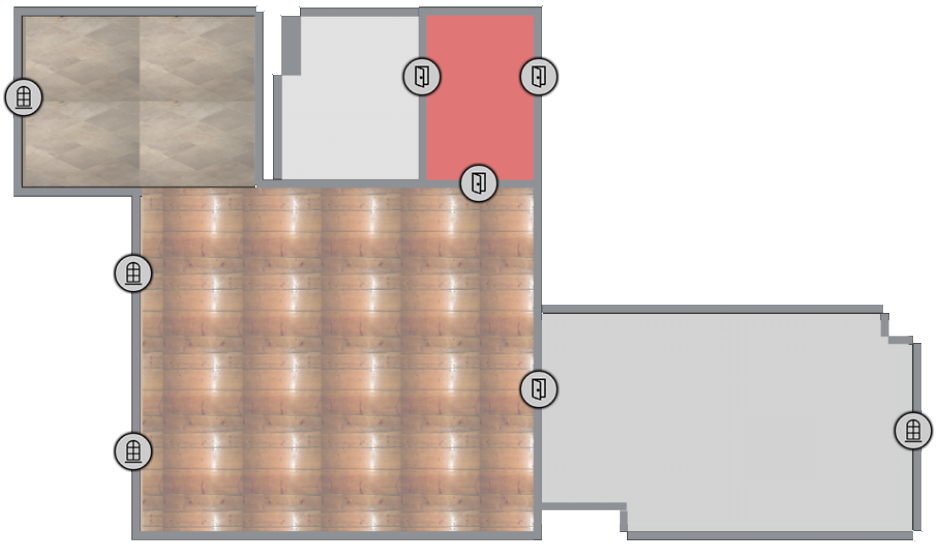

Abbildung 5.1.: Grundriss der Beispielwohnung

Diese Sensorkomponenten wurden für den Testlauf verbaut:

- 6 Lichtschalter
- 5 PIR-Sensor-Komponenten
- 1 Durchgangssensor-Komponente
- 2 Tür-Sensor-Komponenten
- 1 Fenster-Sensor-Komponente
- 1 Bett-Sensor-Komponente

Diese steuerbaren Komponenten konnten verwendet werden:

- 12 kontrollierbare Lampen
- 5 an der Wand montierte Android-Tablets
- 1 kontrollierbarer AV-Receiver

In einem produktiven Einsatz würden natürlich noch weitere Sensoren verbaut werden, jedoch war für die Beispielwohnung die Anzahl durch zeitliche- und finanzielle Ressourcen begrenzt. Dies war jedoch für die geplanten Tests vollkommen ausreichend.

5.2. Durchgangssensor

Während der Konstruktion des Durchgangssensors ergaben sich zwei Probleme, welche bei weitergehenden Entwicklung optimiert werden könnten. Da diese jedoch die prinzipielle Funktionalität nicht eingeschränkt haben und der erreichte Zustand für den funktionalen Prototypen ausgereicht hat, war dies im Kontext dieser Arbeit nicht notwendig.

5.2.1. Ausrichtung

Eine Schwierigkeit beim Bau war die Arretierung und Ausrichtung der Laser-Emitter. Da diese vergleichsweise punktgenau strahlen, eine eher kleine anzustrahlende Zielfläche anstrahlen und einen Weg von etwa einem Meter überbrücken müssen, ist es kompliziert, diese so auszurichten, dass beide Laser-Emitter das jeweilige Ziel exakt treffen.

Diese Problematik wäre ggf. teilweise umgehbar, indem eine Linse auf Empfangsseite verwendet wird, um von einem größeren Bereich den Laserstrahl auf die Photoresistoren zu leiten. Die hierfür notwendigen optischen Recherchen und Planungen hätten jedoch den Rahmen dieser Arbeit überstiegen.

5.2.2. Gesundheitliche Bedenken für den Laser

Ein dauerhaft aktivierter Laserstrahl, montiert auf einer niedrigen Höhe eines Türrahmens (bspw. auf Kniehöhe), ist prinzipiell zunächst kein großes Problem für einen erwachsenen Menschen. Der niedrig klassifizierte Laser ist schwach genug, dass dieser ungefährlich ist, solange ein Mensch nicht über einen längeren Zeitraum direkt in den Laser schaut. Ein erwachsener Mensch wird dies jedoch nicht tun, da die potenzielle Gefahr von Lasern bekannt ist und es keinen Grund gibt, warum er dies tun sollte. Anders sieht dies jedoch bei Kindern oder Tieren aus. Diese haben entweder nicht das Wissen oder das Verständnis über die Gefahr und könnten durch Neugier dazu verlockt sein, längere Zeit in den Laser zu schauen und so die eigenen Augen schädigen.

Eine mögliche Lösung hierbei wäre, dies vonseiten des Sensors zu versuchen zu erkennen. Sobald der Strahl unterbrochen wurde, wird der Emitter für eine kurze Zeit deaktiviert. Nach einer kurzen Wartezeit (wahrscheinlich im Rahmen von wenigen Dutzend Millisekunden bis zu einer halben Sekunde) wird der Laseremitter für wenige Millisekunden aktiviert, um zu testen ob der Strahl weiterhin unterbrochen wird. Dies geschieht so kurz, um bei einer weiterhin sich davor befindenden Person keine lange Bestrahlung (und somit eine Verletzung) zu bewirken. Eine Bestrahlung von bis zu 250 Millisekunden gilt in den meisten Fällen als ungefährlich. [2]

Hierbei müsste jedoch darauf geachtet werden, dass durch die Deaktivierung des Emitters und die damit verbundene Latenz bis eine Erkennung erneut erfolgen kann, die Effektivität des Durchgangssensors nicht zu stark in Mitleidenschaft gezogen wird. Dementsprechend darf die Wartezeit nicht hoch genug sein, dass eine Person währenddessen unerkannt durch die Tür schreiten kann. Darüber hinaus ist dies mitunter nur möglich, wenn der Mikrocontroller für die Steuerung der Laser sowie das Erkennen des Laserstrahls zuständig ist, da die Latenz der Kommunikation zu hoch wäre um für die Test-Laserstrahlen nach einer Wartezeit rechtzeitig zu erkennen.

6. Test

In diesem Abschnitt soll nun gezeigt werden, wie die in Abschnitt 3 definierten funktionalen und nicht-funktionalen Anforderungen auf deren korrekte und ausreichende Umsetzung getestet wurden.

6.1. Test der funktionalen Anforderungen

Das *Jarvis-RuleEngine*-System wurde über einen Zeitraum von fünf Tagen durchgehend innerhalb der Beispielwohnung produktiv angewendet. Hierbei ergaben sich keinerlei Probleme und zu jedem Zeitpunkt war die Funktionalität vollständig einsetzbar.

Darüber wurden zwei Interview-Teilnehmer (Teilnehmer/-in A und B) darum gebeten, jeweils die (während des Interviews) selbst vorgeschlagene Regel-Idee sowie zur Vergleichbarkeit eine allgemeine Regel einzutragen. Wie auch bei der letzten Prototyp-Iteration, so war auch hier kein Problem bei der Eingabe zu beobachten und das Hinzufügen der Regel gelang innerhalb weniger Sekunden.

Die funktionellen Aspekte der *Jarvis-RuleEngine* können somit als ausreichend getestet angesehen werden.

6.2. Test der nicht-funktionalen Anforderungen

Der Test der nicht-funktionalen Anforderungen fokussierte sich auf die Usability des entworfenen und umgesetzten User Interface.

6.2.1. Evaluation der UI-Usability

Die Evaluation wurde gemäß den Grundsätzen der Dialoggestaltung nach EN ISO 9241 durchgeführt. Diese beschreibt sieben Grundsätze, nach denen sich ein Dialog bewerten lässt. Die folgenden Grundsatz-Bewertungen ergaben sich aus Gesprächen mit den Interview-Teilnehmern sowie den eigenen Beobachtungen beim Teilnehmer-gestützten Test.

Aufgabenangemessenheit

Die Aufgabenangemessenheit ist gegeben, da die Benutzer die ihnen gegebenen Vorgaben zum Hinzufügen und Aktivieren/Deaktivieren von Regeln ohne merkliche Probleme durchführen konnten.

Selbstbeschreibungsfähigkeit

Der Dialog ist selbstbeschreibungsfähig, da ausnahmslos jeder Dialogschritt einen Erklärtext besitzt und den Benutzer darauf hinweist, welche Eingabe er tätigen soll. Des Weiteren wurde versucht, die Symbole so intuitiv zu gestalten, wie die vorhandenen Möglichkeiten es zuließen.

Steuerbarkeit

Der Dialog ist eingeschränkt steuerbar. Die Geschwindigkeit ist steuerbar, da der Dialog nicht zeitlich begrenzt ist, erst nach Eingabe des Nutzers fortgefahren wird und Schritte übersprungen werden kann wenn der Benutzer sie nicht durchzugehen wünscht. Des Weiteren ist der Dialogablauf

eingeschränkt umkehrbar durch die implementierte Zurück-Funktion. Diese ist jedoch begrenzt, da Zurück-Schritte, welche unerwünschte Nebeneffekte erzeugen könnten, nicht zugelassen werden.

Erwartungskonformität

Die Erwartungskonformität wird unterstützt dadurch, dass versucht wird so nah am gebräuchlichen Vokabular und dem Verständnis des Benutzers seiner eigenen Wohnung zu sein wie möglich. Dementsprechend werden Fachausdrücke vermieden und die Wohnung in dem echten Grundriss mit zum jeweiligen Raum passenden Texturen dargestellt, um den Wiedererkennungswert der Wohnung seitens des Benutzers zu erhöhen.

Fehlertoleranz

Die Fehlertoleranz ist dadurch gegeben, dass zu jedem Zeitpunkt es ausschließlich möglich ist, valide Eingaben zu tätigen.

Individualisierbarkeit

Die Individualisierbarkeit ist beschränkt auf die Wahl des verwendeten Feedbacks. Zur Auswahl stehen ein akustisches (Sound) und ein taktiles Feedback (Vibration). Je nach Benutzerwunsch können entweder beide, keine oder nur eine der Feedback-Optionen gewählt werden.

Den Interviews war nicht zu entnehmen dass eine weitere Individualisierbarkeit erforderlich oder wünschenswert wäre.

Lernförderlichkeit

Eine Lernförderlichkeit ist bis zu einem gewissen Grad gegeben, da der Benutzer sich frei in dem Dialog bewegen kann und ihn somit erkunden kann. Da Regeln erst beim letzten Schritt nach einer zusätzlichen Abfrage hinzugefügt werden, besteht keine Gefahr für fälschliche oder unerwünschte Eingaben.

Darüber hinaus besteht die erste Dialogseite aus einer kleinen Anleitung und die letzte Dialogseite führt zur Übersicht alle getätigten benutzerseitigen Eingaben.

6.2.2. Auswertung

Das entworfene User Interface besitzt eine vergleichsweise hohe Usability, da alle sieben im vorigen Abschnitt aufgezählte Grundsätze mindestens ausreichend erfüllt wurden. Dies wird dadurch unterstützt, dass mehrere Nutzer dabei beobachtet werden konnten, wie sie die entwickelte Lösung genutzt haben. Hierbei ergaben sich – nach den in der Iterierung im Abschnitt 4.4.1 aufgelisteten Optimierungen – keine offensichtlichen Auffälligkeiten und alle Benutzer schienen vergleichsweise schnell mit dem User Interface zurechtzukommen.

Ein Aspekt, welcher besonders von den Interview-Teilnehmern hervorgehoben wurde, war die simple Darstellungsart in der Regel-Liste, welche angelehnt an eine Formel nach dem Schema „Auslöser + Bedingungen → Aktionen“ aufgebaut ist. Diese wurde als sehr prägnant und verständlich bezeichnet. Eine Teilnehmerin merkte an, dass einige Symbole hierbei noch optimiert werden könnten, da diese den jeweiligen Text ersetzen sollen, was ggf. in der Zukunft mithilfe der Unterstützung eines Designers erfolgen könnte.

7. Zusammenfassung

Das Ziel dieser Arbeit war die Entwicklung eines Systems, mit dem die Elemente eines Smart Homes, wie bspw. Lichtquellen, anhand von der Bewegung des Benutzers gesteuert werden können.

Dabei sollte besonderen Wert auf eine benutzerzentrierte Vorgehensweise gelegt werden, um eine gute Übersicht über die tatsächlichen Anforderungen seitens der späteren Zielgruppe zu bekommen. Diesem Vorgehensmodell folgend wurden zunächst Interviews geführt und gemeinsam mit den Interview-Teilnehmern Wünsche und Skizzen festgehalten.

Im Zuge des Entwurfs wurde ein Regel-System konzipiert, mit dem ein Benutzer das Verhalten seines Smart Homes auf einfache Art und Weise und ohne technische Expertise anhand von Regel-Bausteinen selbst definieren kann.

Ein entwickelter grafischer Prototyp wurde mithilfe der Interview-Teilnehmer weiter iteriert, um gemeinsam Optimierungen zu finden und umzusetzen. Dies konnte mit realistischen Bildschirm-Mockups, interaktivem Verhalten und echter Hardware sehr realitätsnah durchgeführt werden.

Um Tests in produktivnahen Szenarien abhalten zu können, wurden verschiedene Sensor-Komponenten entworfen und gebaut, um bspw. die Bewegung des Benutzers sowie den Status von Objekten im Smart Home abfragen und darauf reagieren zu können.

Insgesamt wurden alle in den Interviews herausgefundenen Anforderungen an das Projekt erfüllt, wodurch es als Erfolg angesehen werden kann. Darüber hinaus bietet es nun eine Ausgangsbasis für weitere Features.

8. Ausblick

Während der Entwurf- und Implementationsphase ergaben sich viele Themengebiete und Ideen, auf denen über diese Arbeit hinaus aufgebaut werden könnte. Innerhalb dieses Kapitels werden nun ausgewählte Thematiken angeschnitten.

8.1. Unterscheidung der Benutzer

Für eine noch stärkere Personalisierung ist es erforderlich, auch einzelne Menschen (bzw. deren Identitäten) unterscheiden zu können. Beispielsweise könnte eine definierte Regel nur ausgelöst werden, wenn eine spezifische Person einen Raum betritt.

Dieses Thema wurde von den in Abschnitt 3.2.2 genannten Interview-Teilnehmern als nicht so wichtig eingestuft, als dass es die zusätzliche Komplexität im Rahmen dieser Arbeit gerechtfertigt hätte. Diese Komplexität rührt daher, dass die meisten der zur Bewegungserkennung nutzbaren Sensoren keine Unterscheidung der Benutzer abdecken können und somit dem System die sensorische Grundlage fehlen würde um darauf zu agieren.

Diese Funktionalität könnte jedoch nachträglich erweitert werden.

8.2. Ausbau zu vollwertigem Smart Home System

Die Kombination aus der *JarvisSuite* und der *Jarvis-RuleEngine* hat einige Vorteile gegenüber vergleichbaren Lösungen. Beide Projekte wurden jedoch prototypisch entwickelt und können somit bisher nicht direkt in produktiven Szenarien angewendet werden. Hierauf könnte jedoch weiter aufgebaut werden und somit auf dieser Basis ein vollwertiges Smart Home System geschaffen werden.

Ein wichtiger Aspekt, welcher für eine Verwendung außerhalb prototypischer Szenarien fehlt, ist eine Möglichkeit das System mitsamt seinen Teilkomponenten und Einstellungen von Laien einrichten zu lassen. Hierbei würden mehr Konfigurations-Flexibilität und z.B. Einrichtungsassistenten sehr viel helfen.

Des Weiteren wäre es sinnvoll, weitere Regelmöglichkeiten anzubieten, um eine noch stärkere Anpassung an den Benutzer zu ermöglichen.

Teil II.

Entwicklung eines zweiten Prototyps

9. Zielsetzung

Im Rahmen des ersten Teils dieser Arbeit wurde ein funktioneller Prototyp entworfen und umgesetzt. Dieser zweite Teil soll nun an dieses Ergebnis anknüpfen und die restlichen im Abschnitt *Analyse* ermittelten Anforderungen erfüllen sowie Optimierungen durchführen. Das Ziel ist erfüllt, wenn die vier Beispiel-Szenarien mittels des Regelsystems erfüllt werden können. Diese Beispiel-Szenarien, basierend auf den Erkenntnissen von Abschnitt 3, waren wie folgt:

1. Der Bewohner soll gewarnt werden, wenn er versucht die Wohnung[1] zu verlassen und es noch geöffnete Fenster gibt.
2. Der Bewohner soll bei Dunkelheit mit stark gedimmten Licht oder dediziertem Orientierungslicht durch die Wohnung geleitet werden.
3. Das Licht im Flur soll beim Betätigen eines Lichtschalters nur stark gedimmt angehen, wenn sich eine Person im Schlafzimmerbett befindet (um diese nicht zu wecken).
4. Im Badezimmer soll automatisch drei Minuten nach Betreten Musik angehen (und nach dem Verlassen auch wieder ausgehen).

Mittels des ersten Prototyps war es möglich, Regeln für zwei der vier genannten Beispiel-Szenarien zu konfigurieren und zu nutzen. Szenario 2 und 4 waren hierbei möglich, 1 und 3 jedoch nicht. Der zweite Prototyp soll nun auf diesem Stand aufbauen und die Konfiguration und Nutzung von Regeln für alle vier Beispiel-Szenarien ermöglichen. Die hierfür notwendigen Änderungen werden nun nachfolgend aufgestellt.

9.1. Bedarf für Beispielszenario 1

Dieses Beispielszenario erfordert eine Erkennungsmöglichkeit, um den Geöffnet-Zustand von Türen und Fenstern erkennen zu können. Im Falle von Türen muss dies mit sehr geringer Latenz geschehen können, um darauf reagieren und den Bewohner warnen zu können, bevor er die Wohnung verlassen hat.

Darüber hinaus ist es erforderlich, eine Möglichkeit zu entwickeln, wie der Bewohner gewarnt werden kann. Hierfür bietet sich die vorhandene Tablet-Oberfläche an, welche in Teil A entwickelt wurde. Diese muss hierfür jedoch für eine solche Anzeige sowie um eine „fernsteuerbaren" Aktivier-Fähigkeit erweitert werden. Letzteres ist erforderlich, da diese Geräte nicht zwingend dauerhaft eingeschaltete Displays besitzen.

9.2. Bedarf für Beispielszenario 3

Das dritte Beispielszenario erfordert zunächst eine Erkennungsmöglichkeit, ob sich derzeit eine Person in einem Bett befindet. Hierfür wurde in Teil A eine Lösung skizziert, wie dies mittels eines selbstentwickeltem Sensor erkannt werden kann.

Des Weiteren bedarf es einer Möglichkeit, hierfür eine Regel definieren zu können. Da beim Betätigen eines Lichtschalters eine automatisch-generierte Systemregel greift, welche im Konflikt mit einer solchen Benutzer Regel stehen würde, muss hierfür eine Lösung gefunden werden.

[1] Im Folgenden ist mit Wohnung auch ein Haus gemeint.

9.3. Optimierung der Präsenzerkennung

Das Thema der Präsenzerkennung ist ein zentraler Bestandteil dieser Arbeit. Teil A beschäftigte sich bereits mit den Möglichkeiten der binären Präsenzerkennung, also ob eine Präsenz erkennbar ist oder nicht. Dies wurde ermöglicht durch passive Infrarotsensoren (kurz: PIR), welche mit geringer Latenz eine – halbwegs-konstante Bewegung vorausgesetzt – vergleichsweise hohe Erkennungrate ermöglichen, jedoch keinerlei Kenntnis über die Anzahl der erkannten Personen bietet und die Unterscheidung, ob die Präsenz nicht mehr vorhanden ist oder sich nur gerade keiner bewegt wird, schwer ist.

Dies soll nun um Möglichkeiten erweitert werden, welche die vorhandenen PIR-basierten Bewegungssensoren mit einer weiteren Sensorart unterstützen. Hierfür wurden in Teil A bereits Lösungen skizziert und auf Machbarkeit untersucht. Diese sollen nun in realisierbare Lösungen überführt und prototypisch entwickelt werden. Insbesondere Beispielszenario 4 würde hiervon profitieren, da die Musik durch die begrenzten Erkennungsmöglichkeiten von Teil A noch mehrere Minuten weiterläuft und dies als störend empfunden werden kann.

10. Entwurf

Um die in der Zielsetzung genannten Ziele zu ermöglichen, bedarf es zunächst einer Weiterentwicklung der vorhandenen Sensoren sowie der Hinzufügung weiterer Sensor-Arten.

10.1. Komponentenübersicht

Basierend auf dem Resultat von Teil A sowie den skizzierten Änderungen der Zielsetzung von Teil B ergibt sich daraus eine gewisse Landschaft an Komponenten. Diese Landschaft ist in Abbildung 10.1 abgebildet und soll zunächst einen groben Überblick bzw. eine Orientierungshilfe darstellen. Auf die jeweiligen Details der neu hinzugekommenen Komponenten sowie die Änderungen vorhandener Komponenten wird nun in diesem Kapitel eingegangen.

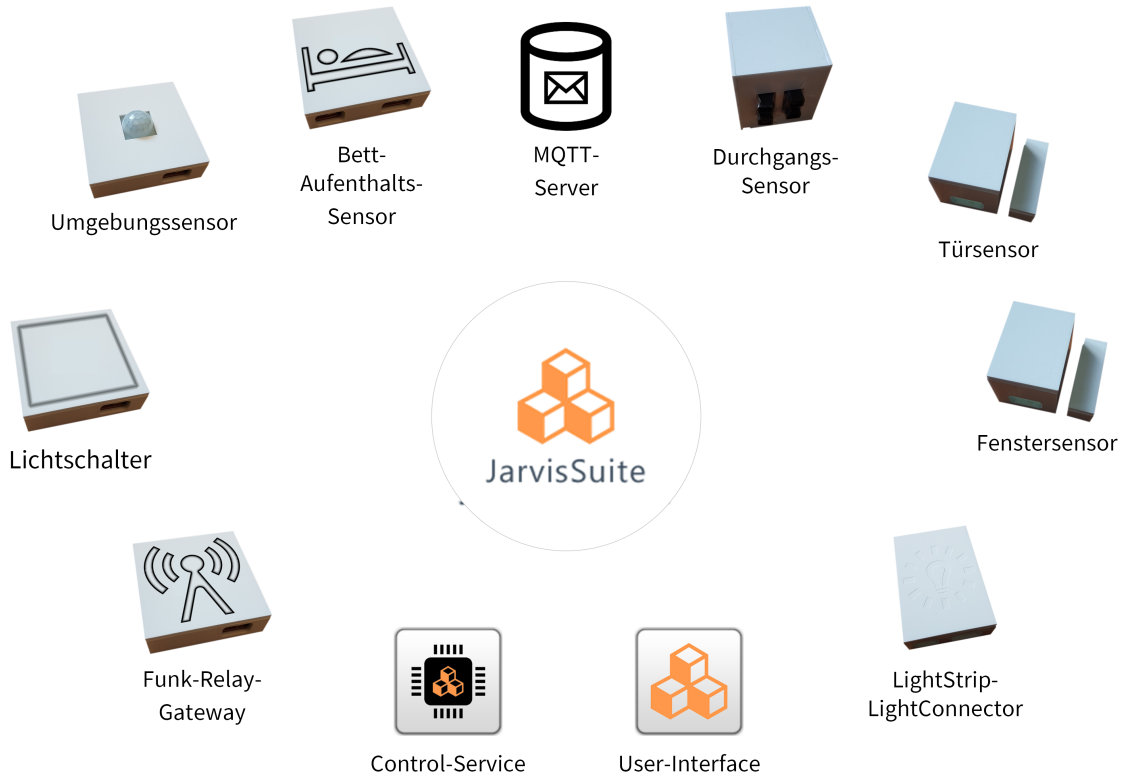

Abbildung 10.1.: System-Landschaft

10.2. Sensoren

Das Gesamtsystem braucht verschiedene Sensoren, um auf Ereignisse reagieren zu können. Aus Teil A konnten die folgenden Sensoren übernommen werden:

- Temperatursensor
- Bewegungssensor

Darüber hinaus kommen nun noch die folgenden Sensorarten hinzu. Diese wurden bereits im Rahmen vom Teil A auf Machbarkeit untersucht, sind aber teilweise erst in Teil B direkt nutzbar gemacht worden.

- Umgebungssensor
- Durchgangssensor
- Bettaufenthaltssensor
- Türsensor
- Fenstersensor

Alle Sensoren werden mit einer festen Stromversorgung mittels externer Netzteile und Stromkabel konzipiert, da eine nicht fest-verkabelte Stromversorgung unnötigen und sehr weitläufigen Mehraufwand bedeuten und eher vom Kernthema ablenken würde. Prinzipiell wäre es jedoch möglich, perspektivisch entweder auf Unterputz-Stromverkabelung oder die Komponenten auf nicht-Netzspannungs-basierte Stromversorgung (bspw. Akkus, Batterien) umzurüsten.

Für alle Komponenten wird ein mittels Lasercutting hergestelltes Acrylglas-basiertes Gehäuse designed, um die Komponenten widerstandsfähiger und pseudo-produktiv-tauglich zu machen sowie auch einen gewissen Brandschutz zu ermöglichen. Dieses Verfahren wurde bereits in [15] angewendet und hat sehr gute Resultate erzielt. Des Weiteren soll jeweils in das Gehäuse ein Symbol eingraviert werden, um die unterschiedlichen Komponenten leicht für den Benutzer unterscheidbar zu machen.

10.2.1. Umgebungssensor

In Teil A wurden experimentelle Bewegungs- sowie Temperatursensoren als eigenständige Komponenten umgesetzt. Diese Separation hatte Vorteile, da durch diese Isolation die Implementierung und Tests der Komponenten vereinfacht wurden. Bei der Überführung in eine produktivtaugliche Version und nachdem deren Implementierung bereits getestet wurde, erscheint es jedoch sinnvoll, diese in eine einzelne Komponente zusammenzuführen. Für beide Komponenten ist es sinnvoll, diese in jedem Raum zu installieren. Die Zusammenführung reduziert somit den Aufwand und die Kosten pro Sensor (da weniger Bauteile vonnöten sind) und erleichtert die Installation, da nur ein Platz reserviert und ein Stromkabel verlegt werden muss.

Der Umgebungssensor enthält die folgenden Sensorkomponenten:

- PIR-Sensor
- Temperatursensor
- Luftfeuchtigkeitssensor
- Barometer

Aufbau

Mittels der beiden Sensorkomponenten, Bosch BME280 und HC-SR501, lassen sich alle vier Sensorarten in einem Gehäuse kombinieren. Als Mikrocontroller kommt, wie auch bei den anderen Sensorkomponenten, ein ESP8266 zum Einsatz, um von der WLAN-Konnektivität zu profitieren.

Für das Gehäuse kann das in [15] konzipierte Acrylglas-basierte Gehäuse verwendet werden. Die einzige hierfür notwendige Änderung ist das Hinzufügen eines quadratischen Ausschnitts im frontalen Gehäuseteil für die herausgeführte Kuppel des PIR-Sensors.

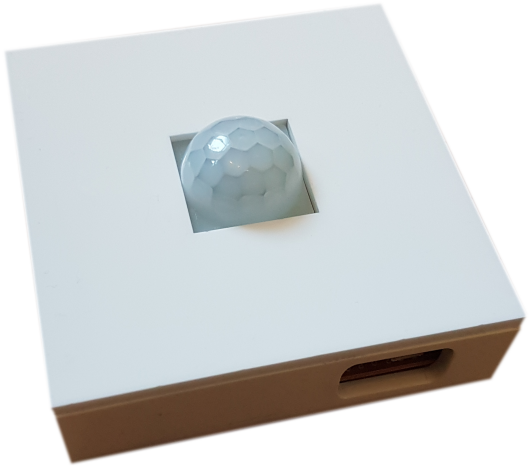

Abbildung 10.2.: Umgebungssensor

Funktionsweise

Die Auswertung und Weiterleitung der Sensordaten unterscheidet sich zwischen den Bewegungs- (PIR) und den Umgebungsdaten.

Die Bewegungsdaten werden alle 100 Millisekunden ausgewertet. Wird eine Bewegung erkannt, so wird eine *HandleMovementCommand*-Nachricht via MQTT an den zuständigen ListenerService (in diesem Falle der *MovementDetectorListener*) übertragen. Nach einer erfolgten positiven Erkennung geht das verwendete Sensorbauteil (HC-SR501) automatisch durch eine Reset-Phase, weswegen nach dem erfolgten Senden der Nachricht für drei Sekunden auf keine neue Bewegung geprüft wird. Dies ist jedoch kein Problem, da die Bewegung ja ohnehin bereits erkannt wurde.

Die restlichen Umgebungsdaten werden alle 60 Sekunden lokal ermittelt und als *HandleUpdatedEnvironmentDataCommand*-Nachricht ebenfalls via MQTT an den zuständigen ListenerService gesendet. Ein Beispiel einer solchen Nachricht ist in Listing 10.3 zu sehen.

```
{
  environmentSensorIdentifier: 12,
  temperature: 23.7,
  humidity: 57,
  pressure: 1004.52
}
```

Abbildung 10.3.: Beispielnachricht von Umgebungssensor

10.2.2. Durchgangssensor

Wie in dem Abschnitt Zielsetzung bereits beschrieben ist eines der Ziele für den zweiten Prototyp die Erkennung der Anzahl der sich in einem Raum aufhaltenden Personen sowie die Optimierung der Präsenzerkennung, um das Verlassen eines Raumes latenzfreier erkennen zu können, als dies von einem PIR-Sensor möglich wäre.

Teil A beschrieb hierfür bereits die grundlegende Funktionsweise und die Vor- und Nachteile, wie dies entweder basierend auf Lasern oder Infrarot implementiert werden kann. Abbildung 10.4 zeigt, wo solch ein Sensor an dem jeweiligen Durchgang (oder Tür) verbaut werden kann.

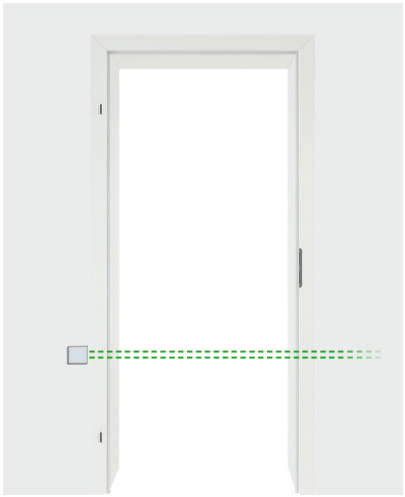

Abbildung 10.4.: Platzierung des Durchgangssensors

Aufbau

Da die Richtung der Bewegung erkannt werden soll, ist es zwingend notwendig zwei Sensoren zu verwenden. Der Sensor, welcher hierfür genutzt werden soll, ist der Sharp GP2Y0A02, da er eine für einen Durchgang sinnvolle Reichweite besitzt (bis zu 1,5 Meter), leicht verfügbar und vergleichsweise kostengünstig ist. Es gäbe noch weitere Sensoren, welche sogar eine geringere Größe besitzen und so leichter verbaut werden könnten, jedoch würde dies das Budget dieser Arbeit überschreiten. Da die bauliche Größe für die prototypische Entwicklung ohnehin nicht relevant ist wird dies hier ignoriert.

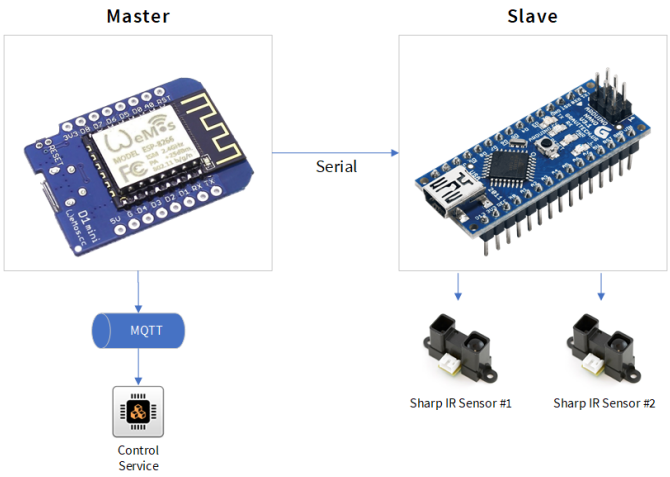

Abbildung 10.5.: Master und Slave Komponente des Durchgangssensors

Für die Anbindung von zwei solcher Sensorkomponenten werden zwei analoge Eingänge benötigt. Der Mikrocontroller ESP8266, welcher bereits in den anderen Sensoren verbaut wurde, besitzt jedoch nur einen solcher Eingänge. Dessen Nachfolger, der ESP32, besitzt hingegen mehr als ein Dutzend solcher Anschlüsse, wäre also eine mögliche Alternative. Sowohl der ESP8266 als auch der ESP32 sind jedoch 3,3 Volt basiert, die Sharp Sensorkomponenten hingegen 5 Volt, wodurch ein analoges Auslesen der Daten erschwert werden würde. Es gibt allerdings viele Arduinos, welche auf 5 Volt basieren, jedoch bringen diese in den seltensten Fällen WLAN-Konnektivität mit sich und sind weitaus kostspieliger. Daher bietet es sich an, das beste aus beiden Welten zu kombinieren und sowohl einen ESP8266 für die Kommunikation und einen Arduino (Nano) für die Ansteuerung

der IR-Sensoren zu verwenden. Der ESP8266 agiert hierbei als „Master“ und verwendet die Daten des „Slaves“ (der Arduino), welcher die Integration und Auswertung der Sensordaten abstrahiert und diese somit dem Master gegenüber aufbereitet zur Verfügung stellt.

Funktionsweise

Wurde eine sich durch den Durchgangssensor bewegende Person erkannt, so wird eine *HandlePassageMovementCommand*-Nachricht via MQTT gesendet. Die wichtigsten Eigenschaften dieser Nachricht sind hierbei das Identifikationsmerkmal des jeweiligen Durchgangssensors sowie *Direction*, also die Richtung in der sich die erkannte Person bewegt hat. Letzteres kann die beiden Werte „From1To2“ sowie „From2To1“ annehmen, wobei diese auch als „Von links nach rechts“ bzw. als „Von rechts nach links“ angesehen werden können. Mittels dieser Werte kann die Bewegung im ControlService verarbeitet werden.

10.2.3. Bettaufenthalts-Sensor

Ein elementarer Bestandteil von Beispielszenario 3 ist die Fähigkeit erkennen zu können, ob sich eine Person in einem hierfür ausgerüsteten Bett befindet. Dies kann entweder mittels Drucksensoren oder durch die Verwendung von kapazitiver Sensorik implementiert werden. Teil A beschrieb bereits einen Ansatz, welcher an das in [18] beschriebene Aufbauschema angelehnt wurde. Dieses Aufbauschema wurde jedoch erweitert mit einem eigenen Algorithmus und der Integration von WLAN-Konnektivität und MQTT-basiertem Messaging.

Aufbau

Für die Integration der kapazitiven Sensorik bietet sich der Touch-Controller “MPR121” an, welcher bereits erfolgreich in den Lichtschaltern (siehe [15]) verwendet wurde. Anders als in den Lichtschaltern, wo eine vergleichsweise kleine Fläche erkennbar gemacht werden muss, ist diese Fläche bei einem Bett sehr viel größer. Da ein liegender Mensch (selbst in einem jüngeren Alter) aber zwingend eine gewisse durchgehende Fläche belegt, reicht es hier aus nur gewisse Teilbereiche der gesamten Bettoberfläche zu testen und somit die Präsenz des sich potenziell darauf befindlichen Menschen zu approximieren.

Abbildung 10.6.: Detektorstreifen

Angebunden an den Touch-Controller sind vier Detektorstreifen. Dies sind Kupferstreifen, welche mittels eines isolierten Kabels mit dem MPR121 verbunden sind. Die Kupferfolie ist hierbei wichtig, da für eine kapazitive Erkennung ein elektrisch leitendes Material verwendet wird. Da Kupferfolie jedoch ein eher unflexibles Mateiral ist, wird es zum Schutz vor Beschädigung in Malertape verpackt, wodurch der innenliegende Kupferstreifen weniger strapaziert wird. Ein sich im Bau befindlicher Detektor-Streifen ist zum besseren Verständnis in Abbildung 10.6 dargestellt. Dem abgebildeten Detektorstreifen fehlt hierbei noch das angelötete Kabel und der obere Schutztape-Streifen. Eine schematische Darstellung des Aufbaus ist in Abbildung 10.7 zu betrachten.

Eine zukünftige Erweiterungsmöglichkeit könnte die Separation in zwei verschiedene Bett-Segmente sein, um bspw. beide Betthälften separat erkennen zu können. Dies würde es erlauben, die Anzahl der Personen im Bett zu approximieren. Eine genaue Erkennung wäre hierbei nur möglich, solange sich jede Person ausschließlich auf einer Betthälfte befindet. Eine solche Erweiterungsmöglichkeit wäre durch die Verdoppelung der Anzahl der Detektor-Streifen möglich. Der MPR121 hat die Möglichkeit, 12 dieser Streifen anzubinden. Die Komponenten können in ein Gehäuse verpackt an das Gestell des Bettes angebracht werden, je nach Bettbauweise sogar verborgen.

Abbildung 10.7.: Aufbau des Bettaufenthalts-Sensors

Funktionsweise

Der MPR121 bietet bereits eine Auto-Kalibrierung, weswegen Schwankungen durch Umweltbedingungen (wie z.B. Temperatur und Luftfeuchtigkeit) bereits ausgeglichen werden. Dies verhindert jedoch nicht, dass es kurzzeitige – im Kontext des Bettaufenthalts – Fehlerkennung geben kann.

Jede Sekunde wertet der verbaute Mikrocontroller die kapazitiven Werte des MPR121 für jeden verbundenen Detektorstreifen aus, berechnet, ob die kapazitiven Werte über einem konfigurierten Schwellenwert liegen (was auf die vorhandene Präsenz deuten würde) und speichert dieses Ergebnis in einen Ringbuffer. Dieser Ringbuffer fasst fünf Werte und verwirft bei einer Speicherung den jeweils ältesten Wert. Sind alle fünf gespeicherten Ergebnisse identisch (bspw. fünfmal eine erkannte Präsenz) und ist dieses Ergebnis ungleich dem letzten gesendeten Wert, so kann davon ausgegangen werden dass sich die Präsenz geändert hat und es wird eine *HandleBedOccupancyChangeCommand*-Nachricht via MQTT versendet.

Die Verwendung des Ringbuffers bringt eine Latenz von fünf Sekunden mit sich (durch die Größe von fünf Werten und die Messung jede Sekunde), jedoch minimiert es die Erkennung von falschen Werten, da hierdurch kleine Schwankungen ausgeglichen werden.

10.2.4. Fenstersensor

Für das Beispielszenario 1 ist es erforderlich, den „Geöffnet“-Zustand von Fenstern und Türen erkennen zu können. Eine solche Erkennung kann z.B. optisch oder magnetisch erfolgen.

Eine optische Erkennung kann beispielsweise über Infrarotsensoren erfolgen, wie sie bereits bei dem Durchgangssensor zur Anwendung kamen. Diese würden neben den Fensterrahmen verbaut werden und erkennen, ob der Fensterrahmen sich in einem geringen Abstand befindet. Wird das Fenster geöffnet (oder angeklappt), so wäre dieser Abstand nicht mehr gegeben und eine Erkennung könnte ausgelöst werden. Ein Nachteil bei der optischen Erfassung ist jedoch, dass (je nach verwendetem Sensorbauteil) ein sehr spezifischer Abstand zum Fenster eingehalten werden muss, damit die Erkennung fehlerfrei funktioniert.

Alternativ kann diese Erkennung auch magnetisch erfolgen. Hierzu können sogenannte Reed-Switches verwendet werden. Hierbei gibt es Paare aus jeweils zwei Elementen. Das kabelgebundene Element ist in diesem Falle nur stromdurchlässig, wenn das zweite Element in unmittelbarer Nähe ist. Wird also das erste Element neben den Fensterrahmen und das zweite Element daneben auf den Fensterrahmen verbaut (so dass diese im geschlossenen Zustand sich nahe beieinander befinden), so kann der Zustand ausgelesen werden.

Ändert sich der erkannte Zustand, so wird eine *HandleWindowChangeCommand*-Nachricht versendet.

10.2.5. Türsensor

Der Türsensor ist Hardware-seitig identisch mit dem Fenstersensor. Die Softwareseite hingegen ist zwar auch ähnlich zu derer des Fenstersensors, sendet jedoch stattdessen ein *HandleDoorChangeCommand* an ein jeweils eigenes MQTT-Topic und besitzt darüber hinaus eine Tür-spezifisches Identifikationmethode.

Perspektivisch könnte das Gehäuse des Türsensors in zwei Varianten existieren, da – anders als bei Fenstern, deren Rahmenhöhe meist ähnlich ist – Türen sowohl, in Relation zum Türrahmen, in erhobener als auch ebenmäßig abschließender Form gebaut sein können. Dies war jedoch im Rahmen dieser Arbeit irrelevant und kann daher als eine mögliche bauliche Optimierung angesehen werden.

10.2.6. Auto-Konfiguration

Mit steigender Anzahl verbauter Komponenten vergrößert sich der Bedarf an automatischen Konfigurationen. Diese besteht bei den Komponenten aus mehreren Aspekten:

Es wird davon ausgegangen, dass eine automatische IP-Zuweisung mittels DHCP erfolgen kann, wie es jeder übliche Router automatisch tut. Alternativ wäre es möglich, eine manuelle Konfiguration zu erlauben, da die verwendeten ESP8266 und ESP32 Mikrocontroller auch ein eigenes WLAN-Netz aufspannen und somit ein Konfigurationsinterface anbieten könnten. Die Zielgruppe hierfür wäre jedoch sehr gering.

Anschließend verbinden sich die Komponenten mit dem lokalen MQTT-Server. Dessen IP-Adresse muss nicht bekannt sein, da diese automatisch ermittelt werden kann. Dies senkt die Anforderungen an die lokale Infrastruktur enorm, da es nicht realistisch wäre eine spezifische IP-Konfiguration vorauszusetzen oder für jeden Sensor dies vom Benutzer konfigurieren zu lassen. Diese automatische Ermittlung erfolgt mittels Multicast DNS (kurz: mDNS). Der Raspberry Pi, welcher als Host für den MQTT-Server sowie den Control-Service fungiert, ist hierbei entsprechend konfiguriert, um seine IP-Adresse im lokalen Netzwerk bekanntzugeben. Die zugreifenden Komponenten können somit über eine Multicast-Anfrage diese IP-Adresse anfordern und entsprechend empfangen.[1] [16] [22]

Alle Sensoren sind prinzipiell statuslos konzipiert. Alle Daten, welche von ihnen gespeichert werden, sind transienter Natur und können somit problemlos bei einem Reset (und somit bei einem bewussten oder unbewussten Neustart, bspw. durch Stromverlust) verworfen werden. Alle Sensoren haben einen eindeutigen ID-Code, welcher im System eingetragen und somit mit einem Objekt assoziiert werden kann (bspw. ein Umgebungssensor mit einem Raum oder ein Bettaufenthaltssensor mit einem im System angelegten Bett). Somit braucht der Sensor keine Informationen über die spätere Nutzung seiner Daten zu haben, sondern kann diese einfach an den MQTT-Server anhand seines ID-Codes weitergegeben und der Control-Service verknüpft diese dann mit dem jeweiligen Objekt.

10.3. Personenerkennung

Die Ausgangslage, die von Teil A übernommen werden kann, basiert bisher ausschließlich auf dem Bewegungssensor (bzw. nun Umgebungssensor). Für eine gute Abdeckung sollen in Teil B nun jedoch mehrere Arten von Sensoren kombiniert werden, nämlich die Umgebungs- sowie die Durchgangssensoren.

Der Umgebungssensor sendet im Falle einer erkannten Präsenz alle drei Sekunden eine entsprechende Nachricht, im folgenden als Präsenz-Signal betitelt. Der Durchgangssensor sendet hingegen pro erkannter durchgehender Person eine Nachricht mit der jeweiligen erkannten Richtung als Eigenschaft dieser Nachricht (Durchgang-Signal). Dies sind alle Informationen, die das System zum derzeitigen Zeitpunkt zur Verfügung hat.

Der Control-Service hält für jeden Raum die aktuelle Anzahl der approximierten Personen vor. Wird ein Durchgang-Signal empfangen, wird für den Quell-Raum der Werk dekrementiert und für den Ziel-Raum inkrementiert, außer es handelt sich um die Wohnungstür. Ist dieser Wert größer als Null und wird für einen definierten Zeitraum (derzeit: drei Minuten) kein Präsenz-Signal empfangen, so geht das System davon aus dass der Raum leer ist (z.B. weil das Verlassen der letzten Person nicht korrekt identifiziert werden konnte) und simuliert für jede – basierend auf dem derzeit approximierten Wert – vorhandene Person das Verlassen des Raumes inklusive der entsprechenden Ereignisse. Wird hingegen ein Präsenz-Signal erkannt, obwohl die approximierte Personenanzahl bei Null liegt und kein Durchgang-Signal erkannt wurde, so verhält sich das System trotzdem so als hätte es ein Durchgang-Signal erhalten. Durch dieses Verhalten ist es gesichert, dass für jedes *PersonEnteredRoomEvent* auch ein *PersonLeftRoomEvent* generiert wird.

Perspektivisch wäre es möglich, den Zeitraum, ab wann von einem leeren Raum ausgegangen wird, adaptiv zu wählen, sodass sich dieser Zeitraum verlängert umso länger vorher Bewegung gemessen wurde. Dies würde es ermöglichen, dass nach längerer Präsenz weniger erkannte Bewegung nötig ist sowie für den Fall eines kurzen Betreten eines Raumes und keiner korrekten Erkennung des Verlassens des Raumes (z.B. durch Versagen des Durchgangssensors) dieses Timeout-Verhalten umso früher eintreten kann. Darüber hinaus wäre es möglich, die Heuristik der kombinierten Präsenzerkennung mit weiteren (theoretisch sogar bereits vorhandenen) Informationen zu erweitern. Beispielsweise wäre es möglich, bei jedweden Statusänderungen von Objekten innerhalb eines Raumes, welche üblicherweise nur durch eine Aktion eines Menschen geschehen könnte (Öffnen eines Fensters, Betreten eines Bettes, Betätigen eines Lichtschalters), dies als Beweis einer Präsenz zu werten und in die Erkennung einzuarbeiten.

10.4. Control-Service

Der Control-Service ist das logische Herzstück des Gesamtsystems. Alle anderen JarvisSuite-Komponenten sind via MQTT hieran angebunden. Wie bereits in Teil A im Detail beschrieben, ist der Control-Service eine C#-Anwendung, welche bspw. auf einem Raspberry Pi betrieben werden kann. Teil B soll hierauf aufbauen und die verwendete Architektur übersichtlicher gestalten.[6]

10.4.1. Architektur

Die Kommunikation der einzelnen Komponenten basiert auf Nachrichten, welche via MQTT übertragen werden. Nachrichten unterteilen sich hierbei in *Commands* und *Events*. Eine Auflistung der Commands und Events des Gesamtsystems ist in Abbildung 10.8 zu sehen. Diese Aufteilung folgt dem Architekturmodell Command-Query Responsibility Segregation (kurz: CQRS).[25] [23]

Commands sind hierbei eingehende Nachrichten, welche bspw. Benutzeraktionen an das System übermitteln, z.B. das Drücken eines Lichtschalters oder das Hinzufügen/Entfernen einer Regel.

Events sind hingegen Änderungen, welche im System stattfinden. Dies kann z.B. durch das Anschalten des Lichts für einen Raum der Fall sein oder wenn erkannt wurde, dass eine Person einen Raum betritt. Auf diese Events können andere Komponenten im System reagieren. Beispielsweise horcht die Tablet-Oberfläche auf *TextDisplayRequested*-Events, um diese Nachricht anzuzeigen.

Abbildung 10.8.: Listen der Commands und Events

10.4.2. Regel-Ausführung

Beispielszenario 3 bringt eine erzwungene Neuerung mit sich. In Teil A wurde beschrieben, wie automatisch sogenannte Systemregeln generiert werden. Diese sorgen dafür, dass auch ohne Benutzerregeln die Nutzbarkeit der minimalen Funktionalität gesichert ist. Nach derzeitigem Stand bedeutet dies, dass pro Raum zwei Regeln erstellt werden, eine um beim Betätigen des Lichtschalters und ausgeschaltetem Licht das Licht anzuschalten und die umgekehrte Version hiervon. In Teil A war es nicht vorgesehen, dass diese Systemregeln sichtbar und deaktivierbar waren. Beispielszenario 3 erfordert es jedoch, dass in dieser Situation (also beim Betätigen des Lichtschalters) stattdessen eine Benutzerregel greifen soll, nicht jedoch die Systemregel.

Dies wäre entweder lösbar, indem der User diese Systemregel explizit deaktivieren kann (Variante A), oder dies automatisch geschieht (Variante B).

Variante A würde erfordern, dass der Benutzer alle Systemregeln in irgendeiner Form angezeigt bekommt und deren Relation zu seinen eigenen Regeln verstehen muss. Eine Wohnung hat üblicherweise mindestens vier Räume, weswegen somit mindestens acht weitere Regeln angezeigt werden müssten.

Variante B bräuchte eine Art Heuristik, welche die Systemregeln in Fällen deaktivieren oder ignorieren kann, wenn es zu einem solchen Konflikt käme. Dies würde somit keinen manuellen Eingriff von Seiten des Benutzers erfordern, könnte aber das Verständnis des Systems erschweren, wenn sich das Verhalten ohne den expliziten Wunsch des Benutzers ändert. Dies kann jedoch gemindert werden, indem entweder beim Erstellen/Bearbeiten der Regel hierzu gewarnt wird oder es bspw. in der Liste der definierten Regeln einen Hinweis bei der diesbetreffenden Regel gibt, die den Benutzer darüber aufklärt.

In Rücksprache mit zwei der im Rahmen der Studie teilgenommenen Benutzer wurde für das weitere Vorgehen Variante B gewählt, da dies einen größeren Komfort für den Benutzer bedeuten dürfte.

10.4.3. Persistenz

Mit der Einführung von User-veränderlichen Inhalten, in diesem Falle die User-spezifizierten Regeln, ist es notwendig, dass das System über eine Persistierung der eigenen Daten verfügt.

Die Speicherung erfolgt hierbei als JSON-Dateien. Dies erlaubt eine simple Serialisierung/Deserialisierung der zu speichernden Inhalte und ermöglicht eine leichte manuelle Anpassung der Werte, was insbesondere bei der Entwicklung Zeit sparen kann.

Des Weiteren werden alle aufkommenden Events in eine Log-Datei geschrieben, wodurch sowohl eine Nachverfolgbarkeit des Verlaufs zu Diagnostikzwecken ermöglicht wird, als auch eine Schnittstelle zur Erweiterung gegeben ist.

10.5. Tablet-Control-App

Das User Interface wurde im Rahmen der vorherigen Arbeiten als Progressive Web App (kurz: PWA) entwickelt, da dies eine Entwicklung mittels plattformübergreifenden Webtechnologien mit nahezu nativer Integration ermöglicht. Hierfür musste eine Manifest-Datei angelegt und in die Webanwendung integriert werden, wonachdessen die Anwendung auf den jeweiligen Tablet-Geräten installiert und fortan verwendet werden konnte. Teil A erweiterte diesen Stand mit experimentellen Oberflächen, um die im Rahmen des grafischen Prototyping entwickelten Oberflächen funktionell zu implementieren. Dies soll nun in Teil B komplettiert werden, sodass der Benutzer neue Regeln definieren und vorhandene Regeln verwalten kann.

Abbildung 10.9.: Regel-Übersicht

Darüber hinaus erfordert das Beispielszenario 1, dass Tablets basierend auf Regeln ferngesteuert aktiviert werden können, damit diese Töne ausgeben und Hinweise/Warnungen anzeigen können. Des Weiteren muss dementsprechend auch eine Möglichkeit gefunden werden, diese Hinweise anzeigen zu können.

10.5.1. Hinweis-Anzeige

Der Stand des User Interfaces am Ende von Teil A war sehr Benutzeraktions-basiert. Die jeweils aktuelle Ansicht wurde nur basierend auf Usereingaben verändert, bspw. einem Klick auf einen Button oder ein Menü-Element. Eine Hinweis-Möglichkeit, wie sie Beispielszenario 1 vorschlägt (beim Verlassen der Wohnung, ergo Öffnen der Wohnungstür), gibt es somit noch nicht.

Das User Interface hat jedoch bereits eine persistente Verbindung zum Control-Service, wodurch der Datenbestand aktuell gehalten wird. Diese Verbindung kann genutzt werden, um basierend auf ausgelösten Regeln Text anzuzeigen.

Um die Funktionalität für weitere Szenarien allgemein zu halten, bietet es sich an dem Benutzer mehr als nur Text als Anzeigemöglichkeit zu bieten. Da eine solche Anzeige sehr dem Dialog-Prinzip ähnelt, erscheint es sinnvoll die üblichen Icons hiervon zur Priorisierung der dargestellten Information zu übernehmen. In Rücksprache mit dem jeweiligen Teilnehmer, auf dessen Grundlage das Beispielszenario entstand, wurde sich auf die drei Prioritäten „Information“, „Warnung“ und „Kritisch“ geeinigt. Die verwendete Priorität ist bei der Erstellung der “Hinweis anzeigen”-Aktion einer Regel festlegbar.

10.5.2. Fernaktivierbarkeit

Eine installierte Progressive Web App kann weder garantiert-dauerhaft auf Netzwerksignale horchen noch basierend auf einem solchen Signal die Anwendung in den Vordergrund bringen, da solche Funktionalitäten zu System-nah sind. Dies lässt sich mittels einer nativen App lösen, jedoch wäre es extremer Aufwand die Web App als native App neu zu implementieren. Daher soll stattdessen die PWA um eine native Wrapper-App ergänzt werden, welche die PWA anzeigt und um die Fähigkeit der Fernaktivierung erweitert.

Diese an Wake-on-LAN erinnernde Funktion kann mittels eines in der Android-App implementierten Background-Service umgesetzt werden, welcher dauerhaft via MQTT auf Signale horcht und so im richtigen Moment das Tablet anschalten kann. Dies muss somit auch nicht zwingend mit der Anzeige eines Hinweises gekoppelt sein, sondern kann auch nur zum anschalten des Tablets (z.B. beim Betreten eines Raumes) kombiniert werden.

11. Implementierung

Im Zuge dieses Kapitels sollen einige ausgewählte Themen, welche sich während der Implementierung ergaben, näher betrachtet werden, z.B. Probleme, deren Lösung zusätzlichen Aufwand notwendig machten.

Im Anhang zu dieser Arbeit befindet sich ein Demonstrationsvideo, welches darüber hinaus die implementierten Komponenten demonstrativ zeigt.

11.1. Erkennungsrate des Umgebungssensors

Beim Überführen eines experimentellen Aufbaus des Umgebungssensors in eine komplett als Platine gelötete und mit einem umgebenden Gehäuse versehenen Version zeigten sich starke Probleme mit dem verwendeten PIR-Sensor. Dieser registrierte ungefähr alle 30 Sekunden einen „false positive“, also eine erkannte Bewegung, auch wenn diese nicht wirklich stattfand.

Dieses Problem trat bei den Bewegungssensoren von Teil A nicht auf, da deren experimenteller und nicht Gehäuse-basierter Aufbau genügend Abstand zwischen dem PIR-Sensor und dem ESP8266 hatte, sodass diese sich nicht beeinflussen konnten. Dieser Abstand würde innerhalb eines Gehäuses jedoch ein zu voluminöses Gehäuse erfordern.

Eine Recherche dieses (im Internet nicht unbekannten) Problems ergab eine Vielzahl von potenziellen Lösungsansätzen, jedoch schienen viele hiervon nicht zwingend helfend zu sein. Die einzigen beiden Ansätze, welche augenscheinlich hilfreich sein könnten, war einen größeren Abstand zwischen den Komponenten zu machen oder den PIR-Sensor von dem ESP8266 zu isolieren. Zielführend war hierbei eine Ummantelung des PIR-Sensors, um diesen von den Interferenzen des ESP8266 zu isolieren. Hierzu wurde zunächst der Sensor vollflächig mit Isolierband ummantelt, ausschließlich die Steckkontakte und die innere Infrarot-Sensor-Komponente unterhalb der Kuppel wurden hierbei ausgelassen. Anschließend wurde Aluminiumfolie mehrlagig um den Sensor gewickelt und diese mittels angelötetem Kabel geerdet. Dieser Aufbau ist in Abbildung 11.1 sichtbar. Das blaue Kabel ist hierbei das Erdungskabel.

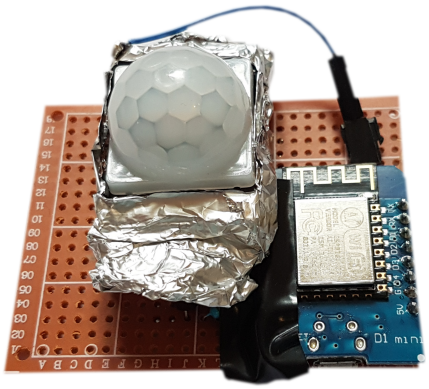

Abbildung 11.1.: Umgebungssensor-Modifikationen

11.2. Erkennungsrate des Durchgangssensors

Im Gegensatz zur Implementation des Master-Teils des Durchgangssensor, der beinahe als trivial beschrieben werden könnte, gestaltete sich die Implementation der Software des Slave-Teils als aufwändiger als gedacht und ergab bis zum Ende der Implementierungszeit gemischte Resultate. Dies hatte mehrere Gründe, welche nachfolgend erläutert werden.

Die gemessenen Daten der Sharp Infrarot-Sensoren waren nur mittels mathematisch aufwendiger Berechnung in leicht weiterzuverwendende Zentimeter-Angaben umzurechnen und waren selbst dann stark schwankend. Durch die Anwendung eines Medianfilters anhand von 9 Werten ließ sich dies mindern, jedoch wurden trotzdem weiterhin bei den Sensoren starke Ausreißer gemessen, jedoch – soweit sich dies erkennen ließ – stets nur bei gleichzeitig einem der beiden Sensoren, wodurch die Verarbeitung dies als Fehler erkennen und damit umgehen konnte.

Des Weiteren ließen die verwendeten Libraries nur eine sequenzielle Abarbeitung der beiden Sensoren zu. Bei steigender Werteanzahl für den Medianfilter bedeutet dies jedoch, dass die Wahrscheinlichkeit steigt, dass eine Bewegung in einer falschen Richtung erkannt wird. Da die Richtung dadurch bestimmt wird, welcher der beiden IR-Sensoren zuerst eine Person erkennt, dies jedoch nur Unterschiede von Bruchteilen einer Sekunde sind und die sequenzielle Abarbeitung somit abwechselnd mehrere Millisekunden mit nur einem Sensor arbeitet, kann es dadurch zu einer falschen Erkennnung kommen. Dies wäre potenziell behebbar, indem die Library umgeschrieben wird, um parallel zwei verschiedene Sensoren ansprechen zu können, jedoch war dieser Aufwand im Rahmen dieser Arbeit nicht tragbar.

Zusätzlich schienen sich die beiden Sensoren teilweise gegenseitig zu beeinflussen, wobei dies nicht abschließend geklärt werden konnte. Die Vergrößerung des Abstandes zwischen den beiden Sensoren sowie ein partielles Überkleben der Sensoroberfläche um eine Überlappung der Erkennungsbereiche zu minimieren konnte jedoch etwas bessere Resultate zu erzeugen.

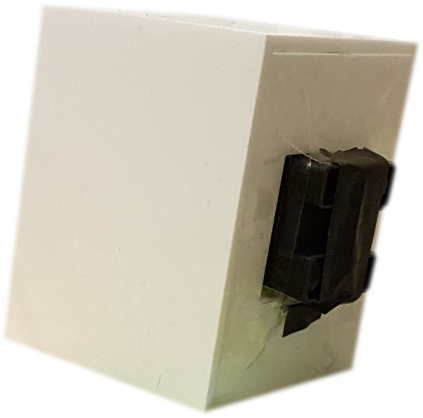

Abbildung 11.2.: Durchgangssensor

In Abbildung 11.2 ist ein fertig gebauter Durchgangssensor dargestellt. Teilweise sichtbar hierauf ist das schwarze Isolierband, mit dem die Sensoren teilweise überdeckt sind. Trotz aller Versuche, die Erkennungsrate weiter zu optimieren, blieb die Erkennungsrate jedoch nur mäßig. Für simple Szenarien, um z.B. nur schneller auf Personenbewegung reagieren zu können (wie dies eines der im Abschnitt Zielsetzung gesetzten Ziele vorgibt), reicht dies aus, insbesondere im Zusammenspiel mit den Umgebungssensoren, jedoch ist hier definitiv noch Spielraum nach oben.

11.3. Konnektivität der Tablet-UIs

Sowohl die Tablet-UI als auch die Wrapper-App benötigen eine konstante Verbindung, um auf dem aktuellen Stand zu bleiben, latenzfrei Kommandos verschicken sowie auf Signale reagieren zu können (z.B. zum Anzeigen von Hinweisen). Während der Entwicklung stellte es sich allerdings als komplex heraus, solch eine konstante Konnektivität zwischen den Tablet-UIs und dem MQTT-Server sowie dem Control-Service zu gewährleisten. Dies hatte mehrere Gründe, je nachdem um welche der beiden Anwendung es sich dabei handelt.

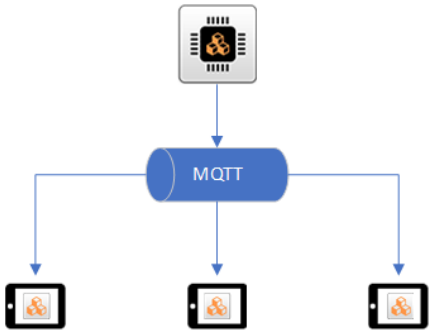

Abbildung 11.3.: Konnektivität zwischen ControlService und Tablets

Die User Interface App hat solch eine Konnektivität auf zwei Ebenen: die Web-Verbindung sowie der Hintergrunddienst.

11.3.1. Web-Verbindung

Die Progressive Web App nutzt Websockets, um die persistente Verbindung herzustellen. Diese Verbindung kann z.B. durch normale transiente Probleme oder durch zu lange Inaktivität beendet werden. In beiden Fällen kann diese Verbindung direkt wieder aufgebaut werden. Überschreitet die Dauer der Inaktivität jedoch eine vom Browser festgelegte Dauer, so wird das Aufbauen einer erneuten Verbindung vom Host (entweder der Browser oder die Browser-Komponente, welche in der Wrapper-App verwendet wird) unterbunden. Um dies zu umgehen, wird das Ereignis des Anzeigens der Oberfläche abgefangen, sodass beim Wiederkehren des Benutzers die Verbindung mit kurzer für den Benutzer sichtbarer Verzögerung wieder aufgebaut werden kann.

11.3.2. Hintergrunddienst

Der Hintergrunddienst der Wrapper-App muss permanent von dem Control-Service erreichbar sein, um das Tablet innerhalb kurzer Zeit einschalten zu können und somit auf Regeln reagieren zu können. Da mobile Datenverbindungen, selbst via WLAN und innerhalb einer relativ kontrollierten Umgebung wie einer Wohnung, unterbrochen werden können ist eine automatische Reconnect-Funktion zwingend notwendig. Das größere Problem war hierbei jedoch weniger die Netzwerkverbindung, sondern dass Android – zumindest in der auf den vorhandenen Tablets genutzten Version – Hintergrundprozesse beendet oder schlafen legt, selbst wenn diese als explizit dauerhaft-laufend („Sticky-Services” im Android-Jargon) definiert wurden. Aus diesem Grund musste ein weiterer Hintergrunddienst entwickelt werden, sodass diese beiden Dienste sich gegenseitig wiederherstellen konnten. Der zweite Hintergrunddienst agiert hiermit als „Watchdog” für den ersten.

12. Test

Zunächst soll hierfür die korrekte Funktionalität der einzeln entwickelten Komponenten separat getestet werden. Abschließen wird der Test mit einer Evaluation, um die Benutzerfreundlichkeit des implementierten Systems sowie die Übereinstimmung der damaligen Wünsche der Studienteilnehmer mit den Möglichkeiten des implementierten Systems vergleichen zu können.

12.1. Unit-Tests

Durch die Verwendung einer CQRS-basierten Architektur für den Control-Service war es vergleichsweise einfach möglich, die Funktionalität des RuleEngine-Managers durch Unit-Tests auf korrekte Funktionalität zu prüfen. Da Regeln ausschließlich durch auftretene Events ausgelöst werden und entweder eigene Events auslösen oder spezifische Schnittstellen ansprechen, kann dies einfach simuliert bzw. durch Mocking nachgestellt werden.

12.2. Sensor-Funktionalität

Die entwickelten Sensoren wurden auf mehrfache Art und Weise getestet:

- Implizit während der Entwicklung
- Durch konstante pseudo-produktive Nutzung in der Beispielwohnung
- Im Rahmen der Evaluation
- Sensorart-spezifische Tests

Auf den letzten Punkt, die Sensorart-spezifischen Tests, sowie potenzielle Beobachtungen während der genannten Tests wird nun nachfolgend eingegangen.

12.2.1. Bettaufenthaltssensor

Die Funktionsweise des Bettaufenthaltssensors wurde von zwei verschiedenen Menschen zu verschiedenen Zeitpunkten getestet, wobei in allen Fällen sowohl das Betreten als auch das Verlassen des Bettes innerhalb des implementierten Erkennungs-Zeitfensters von fünf Sekunden korrekt erkannt wurde. Die beiden Testpersonen hatten hierbei eine unterschiedliche Körpergröße (etwa 178 sowie 163 cm), um eine Relation zwischen Erkennbarkeit und Detektorstreifen-Platzierung auszuschliessen.

Vorhandene Bedenken, dass die Detektorstreifen trotz der Stabilisierung durch Klebeband bei der Verwendung des Bettes in Mitleidenschaft gezogen werden könnten, haben sich als unbegründet herausgestellt. Auch unruhige Schlafphasen und intensive Bewegungen beeinträchtigen den Bettaufenthaltssensor in seiner Funktion nicht. Er ist äußerst strapazierfähig. Auch nach mehreren Wochen ist keines der dünnen Kabel (ab)gerissen.

Die Annahme, dass die Komponenten durch das Klebeband stabilisiert und die gesamte Konstruktion durch das Laken sowie den Matratzenschoner ausreichend geschützt werden, scheint hiermit bestätigt zu sein.

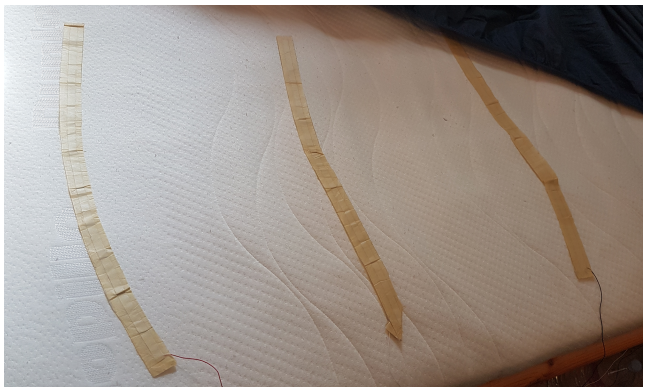

Abbildung 12.1.: Bettaufenthaltssensor Detektorstreifen

12.2.2. Durchgangssensor

Der Durchgangssensor wurde getestet, indem der Umgebungssensor in dem Raum, indem der Durchgangssensor verbaut war, temporär ausgeschaltet wurde und die Anzahl der approximierten Personen des jeweiligen Raums im System der JarvisSuite (welche sich mangels des Umgebungssensors somit ausschließlich aus dem Durchgangssensor speist) mit denen der real im Raum anwesenden Person(en) verglichen wurde, während abwechselnd eine oder zwei Personen den Raum betraten oder verließen. Hierbei wurde versucht, unterschiedliche Bewegungsgeschwindigkeiten und Pausen in der Bewegung miteinzubeziehen.

Bei zielstrebigen Bewegungen war die Erkennungsrate befriedigend ($\geq$ 75%), jedoch hatte die Erkennung sichtliche Probleme bei nicht-zielstrebigen Bewegungen. Diese umfassten bspw. Stehenbleiben im Türrahmen sowie auf halbem Wege wieder umkehren. Die Gründe für die 25% Fehlerrate wurden bereits in Abschnitt 11.2 beschrieben.

12.2.3. Fenstersensor

Der Fenstersensor wurde getestet, in dem das dazugehörige Fenster mehrfach auf- und zugemacht sowie angeklappt wurde, während die empfangenen Commands beobachtet wurden. In ausnahmslos allen Fällen wurde dies innerhalb weniger Millisekunden registriert. Der Fenstersensor kann somit als funktionell getestet angesehen werden.

Abbildung 12.2.: Installierte Fenstersensoren mit unterschiedlichen Gehäusen

Zwei installierte Fenstersensoren sind in Abbildung 12.2 zu betrachten. Links mit einem experimen-

tellen Gehäuse basierend auf vorhandenen Lichtschalter-Gehäusen und rechts mit dem kleineren und finalen Fenstersensor-Gehäuse.

Ein Umstand, der bei der Konzeption nicht bedacht wurde, ist, dass ein Fenstersensor witterungsbeständiger als andere Komponenten sein muss. Während eines Testlaufs, bei dem ein Fenster über längere Zeit geöffnet war, war beim Schließen des Fensters das Gehäuse des Fenstersensors merklich kalt. Die Funktionalität war hiervon jedoch (im Rahmen des Testlaufs) nicht beeinflusst und da es sich hierbei um Prototypen handelt, ist dies zu ignorieren. Bei einer Überführung in eine voll-produktive Iteration sollte dies jedoch näher (und fachmännisch) geprüft werden.

12.2.4. Türsensor

Der Türsensor war ein wichtiger Aspekt in der Evaluation, da er im Zusammenspiel mit der Anzeige einer Warnung verwendet wurde. Hierdurch wurde er automatisch sehr häufig getestet und hat in allen Fällen korrekt funktioniert. Abbildung 12.3 zeigt einen verbauten Türsensor mit extern herausgeführten Magnetkontakten.

Abbildung 12.3.: Installierter Türsensor

12.2.5. Umgebungssensor

Bei guter Platzierung des Umgebungssensors leistete dieser sehr zuverlässige Arbeit. Auch bei geringer Bewegung erkannte der Sensor die Bewegung mit vernachlässigbarer Latenz und nur in einem Falle war ein False-Positive aufgefallen, nämlich als sich die Tür eines Geschirrspülers nach einem durchgeführten Waschgang automatisch öffnete. Es ist wahrscheinlich, dass hierbei nicht die Bewegung ansich registriert wurde, sondern die dabei austretende Hitze (und somit die vom Sensor geprüfte) Infrarotstrahlung erkannt wurde.

Des Weiteren fiel es im Rahmen der pseudo-produktiven Nutzung auf, dass der PIR-Sensor die Präsenz bzw. Bewegung eines Menschen nicht durch die Sicherheitsglas-Scheiben einer Duschwand erkennen konnte. Da das System nach mehreren Minuten (nach derzeitigen Stand: drei Minuten) ohne erkannte Bewegung einen Raum als leer ansieht, kann es je nach Regel-Konfiguration dazu kommen, dass eine „Raum wird leer"-Regel fälschlicherweise ausgelöst wird und somit bspw. das Licht deaktiviert wird. Dies ist jedoch umgehbar, indem der Umgebungssensor an der Decke montiert wird und somit nicht durch das Glas gestört wird. Dies ist ohnehin ein üblicher Ort für einen Präsenzmelder, jedoch wurde im Rahmen des Testlaufs bewusst auf eine Deckeninstallation aus Gründen des Montageaufwands und der Wartungsfreundlichkeit verzichtet.

12.3. Evaluation

Mittels einer benutzerzentrierten Evaluation soll sichergestellt werden, dass das System den gestellten Anforderungen genügen kann.

12.3.1. Teilnehmer

Als Probanden wurden hierfür drei der vier der in Abschnitt 3.2.2 bereits für die Interviews verwendeten Teilnehmer sowie eine weitere Teilnehmerin verwendet werden. Eine Liste der pseudonymisierten Teilnehmer ist in Tabelle 12.1 zu finden.

Teilnehmer	Beziehungsstatus	Wohnungssituation	Geschlecht	Alter	Beruf
Teilnehmer A	Single	alleine wohnend	weiblich	28	Personal
Teilnehmer B	In Beziehung	alleine wohnend	weiblich	28	Lehrerin
Teilnehmer C	In Beziehung	mit Partnerin wohnend	männlich	29	Entwickler
Teilnehmer D	In Beziehung	mit Partner wohnend	weiblich	27	Biotechnikerin

Tabelle 12.1.: Liste der Evaluations-Teilnehmer

12.3.2. Ablauf

Begonnen wurde die Evaluation stets mit einer kurzen Einführung in das System, Dies sollte ungefähr den Rahmen abdecken, den auch eine typische, einem Produkt beiliegende Anleitung erfüllen würde. Anschließend wurde die Aufgabenstellung erklärt. Jeder Proband hat hierbei dieselbe Aufgabenstellung erhalten, um eine Vergleichbarkeit der Ergebnisse und Beobachtungen zu ermöglichen. Die Aufgabenstellung bestand darin, Regeln zu erstellen um die Beispiel-Szenarien zu erfüllen. Diese Beispiel-Szenarien wurden im Rahmen von Teil A in Kooperation mit den Studienteilnehmern formuliert und sind, wie im Abschnitt Zielsetzung erklärt, ein elementarer Bestandteil des Ziels für Teil B.

Nachdem der Teilnehmer genug Zeit hatte die eingegebene Regeln zu definieren und selbst der Meinung war, dass dies nun so funktionieren sollte, durfte er ausprobieren, wie sich die Beispielwohnung durch die neuen Regeln verhielt. Darauffolgend wurde besprochen, ob der Teilnehmer der Meinung ist, ob das Verhalten des Systems mit dem Beispielszenario übereinstimmt und das Ergebnis somit als Korrekt angesehen werden kann.

Abschließend wurde vom Teilnehmer der Attrakdiff-Beurteilungsbogen „Beurteilung der Interaktionstechnik" ausgefüllt und der Teilnehmer bekam danach die Gelegenheit, weiteres Feedback und weitere Verbesserungsvorschläge zu geben sowie eigene Beobachtungen mitzuteilen, was von jedem Teilnehmer sehr wohlwollend und informativ genutzt wurde.

Jede Evaluations-Session hat ungefähr eine Stunde gedauert, wobei das eigentliche Definieren der Regel im Durchschnitt etwa drei Minuten gedauert hat. Eine Auflistung der für den Testlauf verwendeten Komponenten ist im Anhang A.1 zu finden. Der Grundriss der Beispielwohnung ist im Abschnitt 5.1 abgebildet worden.

12.3.3. Ergebnisse

Die Werte des Attrakdiff-Beurteilungsbogens wurden zur Darstellung und Berechnung als Zahlen von 1 bis 7 kodiert, wobei die am meisten links befindliche Option eine 1 symbolisiert und die ganz rechte eine 7. Die Ergebnisse der vier Teilnehmer (A bis D) sowie den jeweiligen Mittelwert und die Standardabweichung sind der Tabelle 12.2 zu entnehmen. Bei der Berechnung des Mittelwerts wurde zur leichteren Lesbarkeit die Polung korrigiert, welche bei dem verwendeten Beurteilungsbogen absichtlich randomisiert ist. Errechnet man nun die für den Attrakdiff-Beurteilungsbogen üblichen Metriken, so ergeben sich die in der Tabelle 12.3 befindlichen Werte.

Teilnehmer	A	B	C	D	Ø (gepolt)	St.Abw.
menschlich, technisch	5	6	2	3	4	1.58
isolierend, verbindend	4	4	6	6	5	1.00
angenehm, unangenehm	2	3	1	4	5.5	1.12
originell, konventionell	4	2	1	1	6	1.22
einfach, kompliziert	3	2	1	2	6	0.71
fachmännisch, laienhaft	2	5	5	1	4.75	1.79
hässlich, schön	6	6	4	6	5.5	0.87
praktisch, unpraktisch	2	2	1	1	6.5	0.50
sympathisch, unsympathisch	2	3	1	4	5.5	1.12
umständlich, direkt	5	5	6	4	5	0.71
stilvoll, stillos	2	1	3	2	6	0.71
voraussagbar, unberechenbar	4	2	1	3	5.5	1.12
minderwertig, wertvoll	5	5	7	7	6	1.00
ausgrenzend, einbeziehend	7	4	6	6	5.75	1.09
bringt mich den Leuten näher, trennt mich von Leuten	6	4	1	4	4.25	1.79
nicht vorzeigbar, vorzeigbar	6	5	6	7	6	0.71
zurückweisend, einladend	6	5	6	6	5.75	0.43
phantasielos, kreativ	6	7	7	7	6.75	0.43
gut, schlecht	2	3	1	1	6.25	0.83
verwirrend, übersichtlich	3	6	5	6	5	1.22
abstoßend, anziehend	4	5	6	5	5	0.71
mutig, vorsichtig	4	3	1	4	5	1.22
innovativ, konservativ	4	1	1	1	6.25	1.30
lahm, fesselnd	6	6	7	7	6.5	0.50
harmlos, herausfordernd	5	5	4	4	4.5	0.50
motivierend, entmutigend	2	2	2	2	6	0.00
neuartig, herkömmlich	4	1	1	1	6.25	1.30
widerspenstig, handhabbar	6	6	6	7	6.25	0.43

Tabelle 12.2.: Liste der Evaluations-Ergebnisse

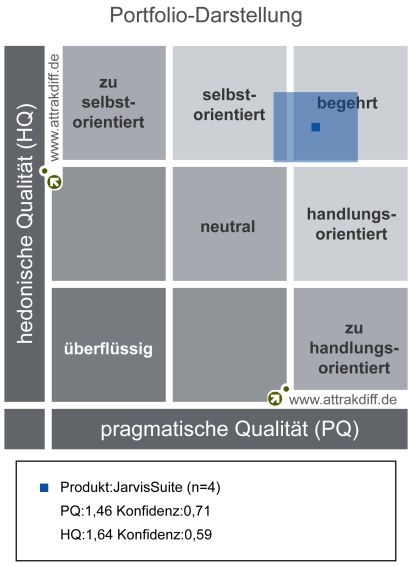

Abbildung 12.4.: Grafische Darstellung der Ergebnisse

Teilnehmer	A	B	C	D	Ø	St.Abw.
Pragmatische Qualität (PQ)	4.57	5.29	6.29	5.71	5.46	0.63
Stimulation (HQS)	4.71	6.14	6.57	6.14	5.89	0.70
Identität (HQI)	5.14	4.57	5.71	6.14	5.39	0.59
Attraktivität (ATTR)	5.71	5.29	6.14	5.43	5.64	0.33
Hedonische Qualität (HQ)	4.93	5.36	6.14	6.14	5.64	0.52

Tabelle 12.3.: Matrix der Evaluations-Ergebnisse

12.3.4. Auswertung

Basierend auf den Beobachtungen und Kommentaren während der Evaluation sowie den Ergebnissen des Beurteilungsbogens lassen sich verschiedene Aussagen treffen. Alle Teilnehmer bestätigten, dass das Verhalten des Systems die jeweils in den Beispielszenarien gegebenen Situationen lösen. Darüber hinaus wurde die eher schlicht gehaltene Oberfläche hervorgehoben, da man sich hierbei auf das wesentliche konzentrieren kann.

Von zwei Teilnehmern wurde positiv erwähnt, dass es möglich ist, dass das automatisch angehende Licht auch zeitgleich mit dem Verlassen des Raumes wieder ausgeht. Die Teilnehmer gaben an, dass sie es von automatisch angehenden Lichtern (wie man es von meist öffentlichen Bereichen kennt) ausnahmslos gewohnt ist, dass das Licht für mehrere Minuten weiter aktiviert bleibt. Dies ist nur durch das Zusammenspiel mit dem entwickelten Durchgangssensor möglich. Es hat jedoch für Verwirrung gesorgt, wenn in den wenigen Fällen, wo der Durchgangssensor die Bewegung nicht korrekt erkannt hat, das Licht dementsprechend länger anblieb.

Die Heuristik zum automatischen Deaktivieren der Systemregeln scheint wie gewünscht zu funktionieren, denn keiner der Teilnehmer schien verwundert zu sein, dass nach dem Anlegen einer Regel, welche in Konflikt mit einer Systemregel stehen würde, ein Lichtschalter anders funktioniert hat. Auf direkte Nachfrage haben alle Teilnehmer auch erklären können, warum in diesem Falle das Licht nicht „normal“ angeschaltet wurde. Die Annahme, dass dies potenziell den Benutzer verwirren kann, scheint also unbegründet.

Teilnehmerin B beschrieb die Anwendung als „super einfach“ und hob besonders hervor, dass es Spaß macht, da man nichts kaputtmachen kann. Im Zweifelsfall, sollte eine Regel nicht das tun was man wollte, löscht man sie schnell und legt innerhalb von wenigen Sekunden eine neue Regel an.

Die zwei „Hauptprobleme“, welche beobachtet werden konnten, waren eine missverständliche Beschriftung zweier Auslöser sowie eine Formulierung bei den Bedingungen. Der Unterschied der Auslöserpaaren „Raum nicht mehr leer“ und „Raum wurde leer“ sowie „Person betritt Raum“ und „Person verlässt Raum“ wurden scheinbar nicht wirklich verstanden. Während „Person betritt Raum“ für jede einen Raum betretende Person ausgelöst werden würde, würde „Raum nicht mehr leer“ nur ein einziges Mal ausgelöst werden bis der Raum wieder leer gewesen wäre. Hier wären andere Formulierungen wie bspw. „*Weitere* Person betritt Raum“ verständlicher. Des Weiteren wird bei der Definition von Bedingungen, bei denen mehrere Elemente ausgewählt werden können (wie z.B. „Fenster geöffnet“), nach der Auswahl die folgende Frage gestellt: „Soll die Regel bei *EINEM* oder *ALLEN* der gewählten Fenster greifen?“. Diese Frage sorgte für große Verwirrung und teilweise auch zu falschen Eingaben im ersten Versuch. Eine Alternative wäre bspw.: „Soll die Regel *NUR* greifen, wenn *ALLE* gewählten Fenster *OFFEN* sind?“, welche alle Teilnehmer sehr viel verständlicher fanden.

Insgesamt kann jedoch durch die in etwa ausgewogene Verteilung von pragmatischer und hedonischer Qualität interpretiert werden, dass die Anwendung sowohl in deren Anwendung sehr funktionell als auch sehr gut anzuwenden ist.

13. Zusammenfassung

Das Ziel dieser Arbeit war die Entwicklung eines Systems, mit dem die Elemente eines Smart Homes, wie bspw. Lichtquellen, anhand von der Bewegung des Benutzers gesteuert werden können.

Aufbauend auf den Resultaten einer durchgeführten benutzerzentrierten Studie und der Entwicklung eines grafischen sowie eines ersten funktionellen Prototyps sollte diese Arbeit einen zweiten Prototyp entwickeln, welcher die angebotene Funktionalität dahingehend erweiterte, dass alle in der Studie gewünschten Beispielszenarien durch das System erfüllt werden können.

Hierzu mussten die zur Verfügung stehenden Sensoren um weitere Sensoren erweitert werden, z.B. Fenster-, Tür-, Durchgangs-, Umgebungs- sowie Bettaufenthaltssensoren. Diese wurden entworfen und prototypisch implementiert, sodass diese im Rahmen einer Evaluation zur Verfügung stehen konnten. Des Weiteren mussten Kernaspekte der Software verbessert werden, um alle neuen benötigten Regelkombinationen abdecken zu können.

Die entwickelte Lösung wurde mittels funktioneller Tests sowie einer durchgeführten benutzerzentrierten Evaluation überprüft. Alle funktionellen Tests waren erfolgreich und die in der Zielsetzung an den Prototyp gestellten Anforderungen wurden hinreichend erfüllt. Da auch die durchgeführte benutzerzentrierte Evaluation zu überwiegend positiven Ergebnissen führte, kann das Projekt als Erfolg angesehen werden.

14. Ausblick

Zusätzlich zu den bereits im Kapitel 8 genannten Ausblicksaspekten vom ersten Teil ergaben sich im Laufe der Umsetzung des zweiten Teils weitere mögliche Optimierungs- und Erweiterungsideen.

14.1. Usability-Optimierung

Basierend auf dem Resultat von Teil B und den Erkenntnissen der Evaluation wäre ein großes Potenzial für weitere Iterierung vorhanden. Weitere Studien würden sicherlich noch mehr Erkenntnisse mit sich bringen, da nach Abschluss von Teil B und dem Vorhandensein einer beinahe komplett ausgestatteten und kontrollierbaren Beispielwohnung das Thema weit weniger abstrakt ist.

14.1.1. Einsteig vereinfachen

Während der Evaluation war zu beobachten, dass die Benutzer sich nach einer kurzen Einführung sehr schnell in die Anwendung eingefunden haben und mit Spaß ausprobiert haben, was alles möglich ist, anfangs jedoch teilweise Zeit brauchten, um auf Ideen zu kommen, was sie denn überhaupt automatisiert haben wollen (exklusive der Beispielszenarien). Dieser Impuls könnte aber ggf. verstärkt werden, indem den Benutzern leichter gezeigt werden kann, was möglich ist und die Benutzer darüber hinaus zu inspirieren. Eine solche Möglichkeit wären bspw. „Regel-Rezepte", also vordefinierte Regeln, welche nur noch auf einen Raum angewandt werden müssen. Die Beispielszenarien geben hierbei bereits Vorschläge hierfür. Beispielsweise könnte es ein Orientierungslicht-Rezept geben, bei dem nur der Raum definiert werden muss und es werden automatisch beide für das Orientierungslicht notwendige Regeln (1x zum Anschalten + 1x zum Ausschalten) erstellt. Eine weitere Möglichkeit hierbei wäre, dass die Regeln nicht automatisch erstellt werden, sondern der Dialog zum Erstellen einer Regel trotzdem durchgeklickt werden muss, die richtigen Optionen für das angewählte Rezept jedoch schon vorausgewählt sind, sodass der Benutzer „nebenbei" sich selbst für das Erstellen schult.

Alternativ oder zusätzlich zu den Regelrezepten könnte es auch hilfreich sein, Demo-Videos zu erstellen und in die Anwendung zu integrieren. Ein „Wie erstelle ich eine Regel"-Video mit gesprochenen Erklärungen könnte den Einstieg sehr viel angenehmer gestalten. Dies könnte z.B. auch beim ersten Start der Anwendung automatisch abgespielt werden.

Teilnehmerin D merkte darüber hinaus an, dass sie es als hilfreich empfunden hätte, an gewissen Punkten mehr Informationen bekommen zu können, z.B. einen Erklärungstext für die gewählte Bedingung, welche über eine „Gedrückt halten"-Geste oder eine Hilfe-Schnittstelle aufgerufen werden. Dies schien hilfreich sein zu können, jedoch kam diese Teilnehmerin (und auch die restlichen) auch so gut zurecht. Dies könnte trotzdem verwendet werden, um den Einstieg weiter zu vereinfachen.

14.1.2. Regeldarstellung

Während der Evaluation ist aufgefallen, dass die Regeln in der Regel-Übersicht zwar leicht wiedererkennbar sind, solange nicht allzu viel Zeit seit dem Eintragen vergangen ist sowie die betrachtende Person auch die Person ist, welche diese Regel erstellt hat, jedoch in anderen Fällen hier noch Optimierungspotenzial besteht. Zwei der Vier Teilnehmer haben diesbezüglich aus eigenem Antrieb hierzu Kommentare fallen lassen. Der Platz auf einem Tablet mit 7 bis 10 Zoll ist sehr begrenzt, aber trotzdem wären hier unterschiedliche Varianten möglich:

Prägnantere Symbolik

Durch die inhärente Begrenzung einer studentisch erzeugten Arbeit mit begrenztem Budget war es unausweichlich, dass die verwendeten Icons primär von offenen Quellen und vom Autor stammten und demnach Kompromisse eingegangen werden mussten. Da das User Interface jedoch sehr Icon-getrieben ist, ist es anzunehmen, dass mit richtigen Designer-angefertigten Icons die Usability und Intuitivität nicht-unwesentlich gesteigert werden könnte.

Kategorisierung

Derzeit ist die Darstellung der Regel-Übersicht als eine lineare eindimensionale Liste umgesetzt. Die Darstellung jeder einzelnen Regel muss möglicherweise gar nicht weiter simplifiziert werden, wenn die Notwendigkeit alle Regeln gleichzeitig darzustellen minimiert wird. Eine Kategorisierung nach Raum würde höchstwahrscheinlich eher unpraktisch sein, da Regeln Raum-übergreifend sein können und dies die Übersichtlichkeit somit leicht brechen könnte. In Absprache mit den Evaluationsteilnehmern wäre eine komplett Benutzer-gegebene Kategorisierung vollkommen ausreichend um für Übersichtlichkeit zu sorgen.

Benennung

Teilnehmer A und D merkten nach mehreren erstellten Regeln an, dass eine Benennung der definierten Regeln die Wiederfindbarkeit vereinfachen könnte. Dies wurde im Rahmen der Studie im Kontext von Teil A damals als nicht zwingend notwendig bewertet, jedoch scheint es mit steigender Regelanzahl doch nötig zu werden.

Automatische Texterklärung

Für interne Diagnostikzwecke enthält die Software einen Algorithmus, welche die Regel-Definition in einen lesbaren Satz (mit der Syntax “IF [Auslöser-Beschreibung] WHEN [Bedingung-Beschreibung] THEN [Aktions-Beschreibung]”) verpackt, sodass in den Log-Dateien die Fehlersuche vereinfacht wird da Regeln leichter vom Entwickler verstanden werden können. Dieses Prinzip könnte prinzipiell (in erweiterter Form) auch für den normalen Benutzer verwendet werden. Jedoch müsste ein Weg gefunden werden, wie diese sehr Text-lastige Ausgabeform in dem sehr begrenzten Platz eines Tablets gut lesbar dargestellt werden könnte.

14.1.3. Umkehrbare Regeln

In den meisten Fällen bestehen die vom Benutzer definierten Regeln aus logischen Pärchen: Gibt es eine Regel zum Anschalten des Lichts oder der Musik, so wird es höchstwahrscheinlich auch eine zum Ausschalten geben. Dies ist nicht zwingend notwendig, da es auch Fälle geben kann, wo dies absichtlich nicht der Fall ist, jedoch kann trotzdem gesagt werden, dass dies überwiegend so der Fall ist.

Da es jedoch unabhängig von der semantisch-logischen Verbindung zwischen diesen Regeln, technisch gesehen nach aktuellem Stand zwei völlig unabhängige und einzeln vom Benutzer definierte Regeln sein müssen, gibt es hier potenziell Optimierungsmöglichkeiten.

Solch eine zweite Regel (der Einfachheit halber hier Umkehrregel genannt, da sie oft den Effekt der ersten Regel umkehrt) kann nicht völlig automatisch erstellt werden, da sich das System nie vollkommen sicher sein kann, welche Bestandteile der Regel umgekehrt werden soll. Ein Beispiel hierfür: Es soll eine Regel geben, welche beim Drücken des Lichtschalters wenn das Licht ausgeschaltet ist und ein bestimmter Zeitbereich gilt, einen spezifischen Licht-Modus aktivieren soll. Soll eine Umkehrregel hier den Zeitbereich invertieren, das Licht ausschalten oder einen anderen Licht-Modus aktivieren? Das kann nur der Benutzer wissen, wie sein Wunsch hier aussieht.

Was jedoch möglich ist: Es gibt viele solcher Konstellationen, wo das System zumindest erahnen kann, was eine sinnvolle Umkehrregel wäre. Nach dem Erstellen der ersten Regel könnte das System dann den Benutzer vorschlagen, diese zweite Regel mit den heuristisch ermittelten Bestandteilen zu erstellen.

Alle vier Teilnehmer bestätigten, dass dies eine sinnvolle und hilfreiche Funktion wäre, jedoch waren 50% der Meinung, dass die Zeitersparnis hierbei fast schon zu gering wäre, da nach ein paar erstellten Regeln der Prozess des Erstellens ohnehin eine sehr schnelle Angelegenheit ist.

Literaturverzeichnis

[1] Badach, Anatol: *Technik der IP-Netze*. München : Carl Hanser Verlag GmbH & Co. KG, 2015. – ISBN 978-3-446-43976-4

[2] baua: Bundesanstalt für Arbeitsschutz und Arbeitsmedizin: *Damit nichts ins Auge geht: Schutz vor Laserstrahlung*. Dortmund, 2005. – ISBN 978-3-88261-678-1

[3] DCTI und bitkom: *DCTI GreenGuide: Smart Home 2015 - Die optimale Lösung für Ihr Zuhause*. 2015. – PDF - Zugegriffen am: 29.09.2017 - Verfügbar unter `http://www.dcti.de/fileadmin/user_upload/GreenGuide_SmartHome_2015_Webversion.pdf`

[4] Die Bibliothek der Technik: *Lichtschranken: Technik und Anwendungen*. Landsberg : verlag moderne industrie, 2000. – ISBN 3-478-93228-9

[5] Eclipse Foundation: *Eclipse SmartHome - A Flexible Framework for the Smart Home*. – Homepage - Zugegriffen am: 25.09.2017 - Verfügbar unter `https://www.eclipse.org/smarthome/`

[6] Engelhardt, Erich F.: *Hausautomation mit Raspberry Pi*. Franzis Verlag, 2016. – ISBN 978-3-645-60313-3

[7] Espressif Systems: *ESP32 Overview — Espressif Systems*. – Homepage - Zugegriffen am: 08.04.2017 - Verfügbar unter `https://espressif.com/en/products/hardware/esp32/overview`

[8] Espressif Systems: *ESP8266EX Overview — Espressif Systems*. – Homepage - Zugegriffen am: 08.04.2017 - Verfügbar unter `https://espressif.com/en/products/hardware/esp8266ex/overview`

[9] Evans, Eric: *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston : Addison-Wesley Professional, 2003. – ISBN 978-0-321-12521-7

[10] INTERNET WORLD Business: *Akzeptanz und Nutzung von Smart Home in Deutschland*. 2017. – Homepage - Zugegriffen am: 29.09.2017 - Verfügbar unter `https://www.internetworld.de/technik/smart-home/akzeptanz-nutzung-smart-home-in-deutschland-1221718.html`

[11] InVision: *InVision — Digital Product Design, Workflow & Collaboration*. – Homepage - Zugegriffen am: 24.09.2017 - Verfügbar unter `https://www.invisionapp.com`

[12] Karvinen, Kimmo ; Karvinen, Tero ; Valtokari, Ville: *Sensoren: Messen und experimentieren mit Arduino und Raspberry Pi*. Heidelberg : dpunkt.verlag, 2015. – ISBN 978-3-86490-160-7

[13] Kleger, Raymond: *Sensorik für Praktiker*. Heidelberg : AZ Fachverlage AG, 1998. – ISBN 3-905214-32-6

[14] Mehlei, Nicolas: *Entwicklung einer Heimautomationslösung mit Multi-System-Kompatibilität und Tablet-Interface*. 2016. – Zugegriffen am: 25.09.2017 - Verfügbar unter `https://www.mehlei.de/Content/Files/Nicolas-Mehlei-Projektarbeit-JarvisSuite.pdf`

[15] Mehlei, Nicolas: *Entwicklung von vernetzten Benutzungsschnittstellen für die Heimautomation*. 2016. – Zugegriffen am: 25.09.2017 - Verfügbar unter `https://www.mehlei.de/Content/Files/Nicolas-Mehlei-Projektarbeit-Lichtschalter.pdf`

[16] MQTT: *MQTT*. – Homepage - Zugegriffen am: 08.04.2017 - Verfügbar unter `http://mqtt.org/`

[17] openHAB Foundation: *openHAB*. – Homepage - Zugegriffen am: 25.09.2017 - Verfügbar unter `https://www.openhab.org`

[18] Pete B.: *Bed Occupancy Sensor — MySensors - create your own Connected Home Experience*. 2015. – Homepage - Zugegriffen am: 30.10.2017 - Verfügbar unter `https://www.mysensors.org/build/bed_occupancy`

[19] Philips Lighting GmbH: *Smart lampen - Meethue — Philips Lighting*. – Homepage - Zugegriffen am: 25.09.2017 - Verfügbar unter `www.meethue.com/de-de`

[20] Rudolf Koenig: *FHEMs Einstiegsseite*. – Homepage - Zugegriffen am: 25.09.2017 - Verfügbar unter `https://www.fhem.de`

[21] Spang, Charlotte ; Reckin, Ronny: *Erstellen und Nutzen von Prototyping in der Softwareentwicklung: Vor- und Nachteile verschiedener Wiedergabetreue (Fidelity)*. Berlin : Berliner Kompetenzzentrum für Usability-Maßnahmen, 2015

[22] Tanenbaum, Andrew S. ; Steen, Maarten van: *Verteilte Systeme : Prinzipien und Paradigmen*. München u.a : Pearson Studium, 2008. – ISBN 978-38-2737293-2

[23] Vernon, Vaughn: *Implementing Domain-Driven Design*. Addison-Wesley Professional, 2013. – Rohfassung – ISBN 978-0-321-83457-7

[24] Wilkes, Birgit: *Smart Home für altersgerechtes Wohnen*. Berlin : Birgit Wilkes, 2016. – ISBN 978-3-8007-4057-4

[25] Young, Greg: *CQRS Documents*. 2010. – Artikel – Zugegriffen am: 18.01.2013 – Verfügbar unter `http://cqrs.files.wordpress.com/2010/11/cqrs_documents.pdf`

A. Anhang

A.1. Test-Komponenten

Nachfolgend eine Auflistung aller Komponenten, welche im Rahmen des Testlaufs verwendet wurden:

A.1.1. Infrastruktur

- Raspberry Pi mit Control-Service
- Funk-Relay-Gateway
- Kontrollierbarer AV-Receiver

A.1.2. Steuerkomponenten

- 4 Tablets
- 6 Lichtschalter

A.1.3. Sensoren

- 3 Umgebungssensoren
- 1 Türsensor
- 4 Fenstersensoren
- 2 Durchgangssensoren
- 1 Bettaufenthalts-Sensor mit 4 Detektorstreifen

A.1.4. Lichtquellen

- 8 Philips Hue Leuchtmittel in E27-Fassungen
- 3 Intertechno-basierte Funk-Schalter
- 2 „RotarySwitch“-basierte Funk-Schalter für Küchen-Arbeitsplattenbeleuchtung sowie nicht-steuerbare LED-Leiste
- 1 „DipSwitch“ Funksteckdose angebunden an eine Arbeitsplatzbeleuchtung
- 5 „LightStrip“ LED-Leisten

A.2. Quelltexte

Da die summierte Länge der Quelltexte zu viel wäre, um diese hier anzuhängen, sind diese in Dateiform angehangen. Nachfolgend eine Liste der Quelltext-Pakete:

- **Source-Code.zip**: Quelltext des ControlServices, der WebUI und der App
- **BedOccupancySensor.ino**: Quelltext des Bettaufenthalts-Sensors

- **EnvironmentSensor.ino**: Quelltest des Umgebungssensors
- **PassageSensorMaster.ino**: Quelltext des Master-Teils des Durchgangssensors
- **PassageSensorSlave.ino**: Quelltext des Slave-Teils des Durchgangssensors
- **LightStripController.ino**: Quelltext des LED-Leisten-Controllers
- **DoorSensor.ino**: Quelltext des Türsensors
- **WindowSensor.ino**: Quelltext des Fenstersensors
- **RoomControlInterface11.ino**: Quelltext des Lichtschalters
- **RemoteSwitchGateway.ino**: Quelltext des RemoteSwitch-Gateways
- **Demo-Video-With-Audio.mp4**: Demonstrationsvideo mit Kommentierung
- **Demo-Video.mp4**: Demonstrationsvideo
- **Präsentation.pptx**: Präsentation

A.3. Regel-Bestandteile

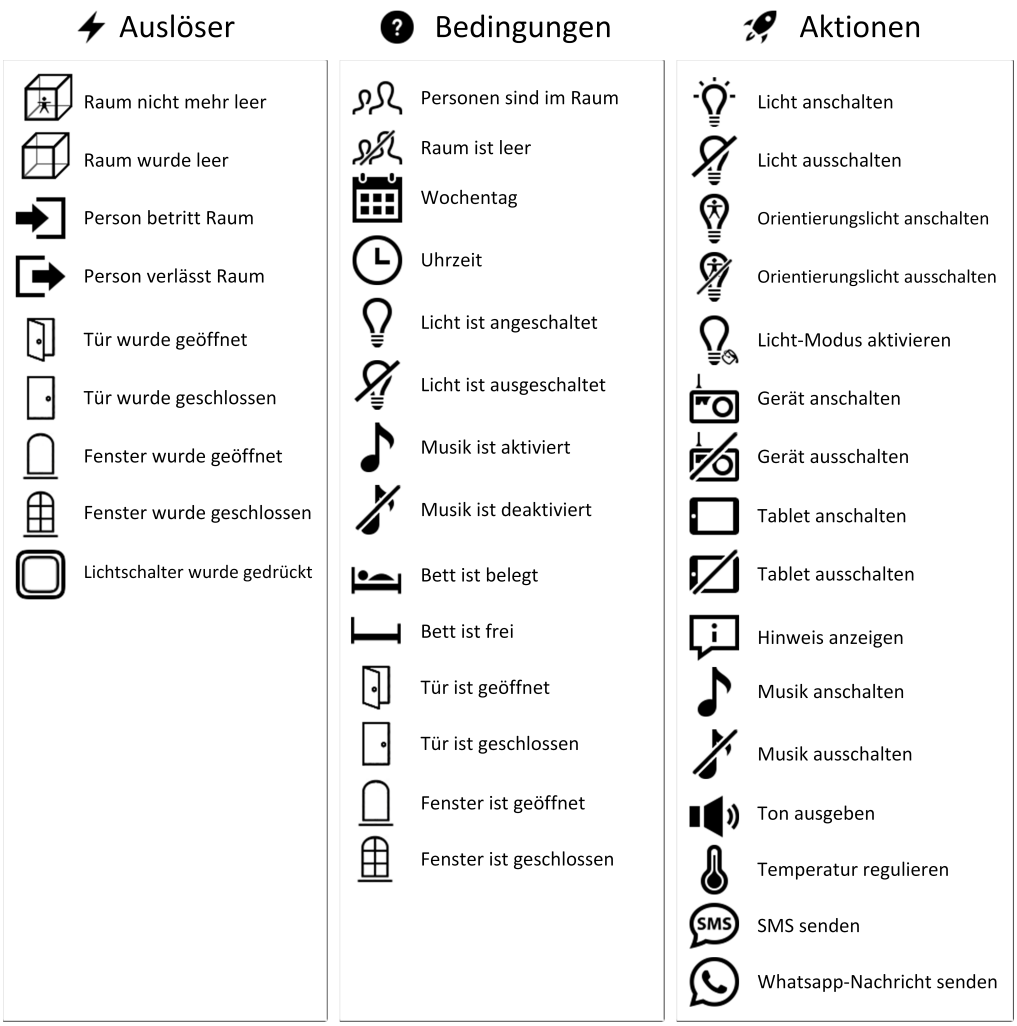

Abbildung A.1.: Beispiele für Regel-Bestandteile