Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

</farinuova>

Entwicklung einer Heimautomationslösung mit Multi-System-Kompatibilität und Tablet-Interface

von

Nicolas Mehlei

als
Independent Coursework Projektarbeit

betreut von
Prof. Dr.-Ing. Thomas Jung
Hochschule für Technik und Wirtschaft Berlin

Berlin, 12. April 2017

Inhaltsverzeichnis

1 Einleitung

Der Mensch ist seit Anbeginn der Zeit stetes bemüht, seine Komfortabilität in allen Bereichen zu steigern. Diesem Gedanken folgend gibt es bereits seit vielen Jahren die Thematik der so genannten Heimautomation, neuerdings auch als „Smart Home“ bezeichnet. Diese Thematik beschäftigt sich mit den Möglichkeiten, wie der Komfort innerhalb der eigenen vier Wände durch verstärkte Automatisierung und komfortablere Benutzungsoberflächen gesteigert werden kann.

Diese Arbeit soll eine grobe Übersicht über die aktuelle Marktsituation bezüglich Heimautomationslösungen darstellen und die Analyse, den Entwurf sowie die Implementation einer prototypischen alternativen Lösung 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 *FariNuova GmbH*, mit der der Autor in vielen Projekten kooperiert, hat eine Vielzahl an zukünftigen Projekten, welche mit Immobilien zu tun haben. Hier besteht der Wunsch, perspektivisch Wohnungen mit sehr spezialisierten (und auf die jeweiligen Bedürfnisse angepassten) Heimautomationslösungen auszustatten. Als der Autor diesem Unternehmen gegenüber den Gedanken einer solchen Entwicklung (oder zumindest der Grundlage hiervon) erwähnte, wurden entsprechende Ressourcen zugesichert, wenn sich die Software als nutzbar herausstellen sollte. Dies bekräftigte den Gedanken des Autors, solch eine Eigenentwicklung durchzuführen, da sich dies mit genügend Zeit und Ressourcenaufwand zu einer wirklich produktiv nutzbaren Lösung entwickeln könnte.

2 Situation

Nachfolgend soll ein grober Überblick über die aktuelle Marktlage gegeben sowie erläutert werden, welcher Aspekt hiervon als verbesserungswürdig angesehen werden könnte. Basierend darauf wird ein möglicher Lösungsansatz skizziert. Um den Fokus dieser Arbeit nicht zu global zu halten, wird die Betrachtung auf die Anforderungen und Möglichkeiten von Privatleuten in Mietwohnungs-Verhältnissen beschränkt.

2.1 Existierende Systeme

Zunächst soll eine Übersicht über vorhandene Lichtsteuerungssysteme, deren Inter-Kompatibilität und deren Möglichkeiten der Steuerung gegeben werden. Hierbei wird absichtlich das Augenmerk auf direkt Consumer-verwendbare Systeme gerichtet und nicht solche, welche durch ihre enormen Möglichkeiten und Flexibilität einen gewöhnlichen nicht stark technisch-veranlagten Benutzer überfordern würden.

2.1.1 Klassisch-elektrisch

Als klassisch-elektrische Systeme sind im Rahmen dieser Arbeit Lichtsteuerungssysteme gemeint, welche über die konventionelle Schaltung von elektrischen Leitungen funktionieren. Dies ist der Quasi-Standard, mit dem ein Großteil der heutigen Wohnungen und Häuser ausgestattet sind. Diese Systeme sind jedoch nicht über automatisierbare Schnittstellen anbindbar sondern erfordern eine manuelle Aktion seitens des jeweiligen Benutzers.

2.1.2 Funk-Systeme

Eine Alternative für klassisch-elektrische Lichtsteuerungssysteme sind sogenannte Funk-Systeme. Diese bestehen meist aus Schaltern, Dimmern, Tastern und Fernbedienungen und kommunizieren über ein Frequenzband von entweder 433 oder 868 MHz.

Die Funktionalität der Lichtquellen ist hierbei effektiv identisch mit denen von klassisch-elektrischen Systemen, bis auf dass die Kommunikation via Funk erfolgt. Diese bieten sich also an, wenn deren Schaltung automatisiert werden soll.

Es gibt hierbei eine Vielzahl an Herstellern, welche sehr ähnlich zueinander aufgebaute Geräte produzieren. Diese geringen Unterschiede (z.B. was die Adressierbarkeit der jeweiligen Geräte angeht) sind jedoch groß genug, dass davon ausgegangen werden muss, dass jede anzubindende Geräte-Art zumindest kleine Anpassungen in der Anbindung benötigt. [1]

2.1.3 Philips Hue

Philips Hue ist ein 2012 erschienenes Produkt, welches aus Lichtquellen und Bedienelementen besteht. Hue war die erste direkt Consumer-taugliche Möglichkeit, farblich-dynamische Lampen in einer Wohnung zu verteilen, welche komfortabel gesteuert werden können.

Hue setzt auf ein *Szenen* genanntes System. Hierbei wird eine Farbe für einen Raum definiert, wodurch sich alle Lampen des jeweiligen Raums auf eben diese Farbe schalten.

Zusätzlich zu einer Auswahl von Lichtquellen mittels der üblichen Sockel bietet das Hue-Sortiment auch alternative Lichtquellen-Arten wie z.B. LED-Lichtleisten sowie Dekorations-orientierte LED-Strahler. All diese Produkte sind in ein Hue System integrierbar und somit darüber steuerbar.

Die Steuerung erfolgt bei Hue entweder über die eigens hierfür entwickelte App (sowie weiterer Apps von Dritt-Anbietern) als auch über spezielle Lichtschalter-artige Eingabegeräte. Die Darstellung der Lichtquellen innerhalb der App-Oberfläche ist nach Räumen kategorisiert und erscheint stets als Liste. [1]

2.1.4 Osram Lightify

Lightify, von der Firma Osram, ist ein sehr ähnliches System zu Philips Hue. Es bedient sich in weiten Teilen derselben Technik, weswegen es möglich ist, Lichtquellen von Lightify von einer Philips Hue Basissstation steuern zu lassen.

2.1.5 KNX

KNX ist der am meisten verbreitete Standard für Hausautomatisierung und somit auch als Lichtsteuerungssystem. Es wird jedoch nur in den seltensten Fällen von Privatpersonen installiert. Da hierfür in der gesamten Wohnung bzw. das gesamte Haus vieradrige Kabelleitungen verlegt werden müssen, ist dies eher eine Lösung für industrielle Gebiete oder als Erweiterung bei einem Hausbau. Dementsprechend wird dieser Standard in den folgenden Abschnitten nicht weiter beachtet. [1]

2.2 Problematik

Eine potenzielle Problematik, welche viele der dargestellten Systeme betrifft, ist die, dass die dargestellten Komponenten (wie z.B. Lichtquellen) der Wohnung in den jeweiligen Oberflächen ausschließlich in Listen-basierten Darstellungen oder maximal als schematischer 2D-Grundriss abgebildet werden. Dies entspricht nicht der Anschauung des das System benutzenden Menschen, denn dieser sieht die Wohnung ja aus einer 3D-Perspektive, nicht aus einer Vogelperspektive und erst recht nicht als Liste. Eine mehr der menschlichen Wahrnehmung entsprechende Darstellungsart könnte also den Umgang mit dem System angenehmer machen.

Darüber hinaus sind die Systeme in den meisten Fällen ausschließlich mit sich selbst kompatibel. Da jedoch selten nur die Produkte eines Herstellers ausreichen, um eine komplette Wohnung mit all ihren potenziellen Eigenheiten auszustatten, sowie ggf. nicht auf bereits vorhandenen (und mitunter fest verbauten) Lichtquellen verzichtet werden sollen, eröffnet sich das Problem, dass keine der vorgestellten Lösungen eine solche Flexibilität – was die Kompatibilität der angebunden Geräte angeht – bietet.

2.3 Lösungsansatz

Ein Lösungsansatz wäre hierbei die Erschaffung einer Alternative zu den vorgestellten vorhandenen Systemen.

Die im vorigen Abschnitt angedeuteten Problematiken beschränken sich auf die Software-Seite der vorhandenen Lösungen, nicht jedoch deren Hardware-Seite. Somit bietet es sich an, auf die vorhandene (gut funktionierende) Hardware der vorgestellten Lösungen zu setzen, jedoch deren Software-Seite zu ersetzen.

Hierbei soll versucht werden, den Funktionsumfang komfortabler zu gestalten. Einerseits soll analysiert werden, ob es nicht möglich wäre, für die Darstellungs eine Form zu wählen, welche eher der Wahrnehmung des Nutzers entspricht. Eine 3D-Darstellung der Wohnung wäre z.B. eine solche

Möglichkeit.

Darüber hinaus soll das bereits vorhandene Konzept der Szenen, wie es z.B. Philips Hue implementiert, erweitert werden. Statt eine einzelne Farbe für einen ganzen Raum zu definieren, nach denen sich alle Lichtquellen richten sollen (selbst wenn diese hierzu gar nicht in der Lage sind, bspw. wenn diese nicht die Möglichkeit besitzen Farben darzustellen), soll ein verbessertes Konzept aufgestellt werden, bei dem es möglich ist zu definieren in welchem Fall welche Lichtquelle mit jeweils welchen Einstellungen aktiviert werden soll. Um sich von dem Begriff der Szenen abzugrenzen (da es sich ja um ein anderes Konzept handelt), wird nachfolgend hierfür der Begriff *Preset* verwendet.

Für Entwicklungs-, Test- sowie Demonstrationszwecke steht eine Wohnung bereit, welche entsprechend den Bedürfnissen der Software angepasst werden kann. Es wird hierbei jedoch darauf geachtet, das Konzept sehr allgemein und nicht nur spezifisch für diese eine Wohnung anzufertigen.

3 Analyse

In diesem Kapitel sollen die Anforderungen des in Abschnitt 2.3 skizzierten Lösungsansatzes analysiert werden. Diese unterscheiden sich in funktionale und nicht-funktionale Anforderungen.

3.1 Funktionale Anforderungen

Funktionale Anforderungen geben die Menge an Funktionen an, welche implementiert werden müssen, damit das im Lösungsansatz skizzierte Ergebnis erreicht werden kann.

Im Falle dieser Anwendung kann dies als drei Haupt-Funktionen formuliert werden:

Wohnung visualisieren

Bevor der Benutzer in der Lage sein kann, spezifische Befehle im System abzusetzen, muss er sich zunächst zurechtfinden können. Hierzu ist es vorteilhaft, wenn sich die Darstellungsart so nah wie möglich am mentalen Modell des Benutzers bezüglich seiner eigenen Wohnung orientiert. Es muss also ein Weg gefunden werden, die Darstellung originalgetreu zu gestalten ohne aber die Bedienung zu erschweren.

Licht an- und ausschalten

Als primäre Funktion soll das Licht der verschiedenen Räume der Wohnung an- und ausgeschaltet werden können.

Presets aktivieren

Über das normale Ein- und Anschalten des Lichts hinaus sollen zusätzlich *Presets* aktiviert werden können. Presets sollen spezifisch angelegte Lichtsituationen für einen Raum darstellen. Sie sollen angeben, welche Lichtquellen in welcher jeweiligen Konfiguration angeschaltet werden sollen.

3.2 Nicht-funktionale Anforderungen

Basierend auf diesen funktionalen Anforderungen lässt sich bereits ableiten, dass es eine Server-Komponente zur Steuerung sowie ein User-Interface geben muss. Nachfolgend werden nun die nicht-funktionalen Anforderungen aufgestellt, welche dafür sorgen, dass die entwickelte Lösung sowohl eine angenehme User Experience als auch eine ausreichende Geräteunterstützung bietet.

3.2.1 Geräteunterstützung von Lichtquellen

Damit eine Abstrahierung unterschiedlicher Lichtquellen, inkl. deren Anbindung, Anforderungen und Eigenheiten, sinnvoll dargestellt und umgesetzt werden kann, bedarf es einer Kompatibilität zu mehreren unterschiedlichen Lichtsystemen.

Basierend auf der zur Verfügung stehenden Hardware wurden fünf Lichtquellen-Arten ausgewählt:

Philips Hue Lichtsteuerungssystem

Die Philips Hue Basisstation bietet eine REST-basierte API, welche über das Netzwerk angesprochen werden kann. Hierüber kann sowohl der aktuelle Stand der verbundenen Lampen abgefragt werden als auch Kommandos zur Veränderung des Zustands abgesetzt werden.

Durch die Anbindung von Hue können automatisch auch alle Osram Lightify Lichtquellen verwendet werden, solange diese in die Philips Hue Basisstation eingebucht werden. Somit kann durch die Integration eines Lichtsteuerungssystems die Lichtquellen-Vielfalt zweier Hersteller angesprochen werden.

Funk-System „Intertechno“

Intertechno ist einer der größeren Hersteller was Funk-basierte Lichtschalter sowie verwandte Produkte angeht. Das Produktangebot umfasst mehrere Geräte, welche es erlauben, konventionelle Lichtquellen nachträglich mit einer Fernsteuerbarkeit auszustatten. Da die Demo-Wohnung bereits drei konventionelle Deckenlampen besitzt, welche in das Gesamtsystem integriert werden müssen um eine allumfassende Steuerung zu gewährleisten, bietet es sich an solche Funkschalter zu unterstützen.

Während das Hue System sehr simpel über das verfügbare Netzwerk angesprochen werden kann so besteht leider für Funk keine vorhandene Schnittstelle zur Kommunikation. Damit diese (und folgende) Funk-basierte Komponente entsprechend angesprochen werden kann, muss eine Möglichkeit zur Funk-basierten Kommunikation hinzugefügt werden. Diese Problematik wird in späteren Abschnitten aufgegriffen.

Funk-System „RotarySwitch“

Wie in Abschnitt 2.1.2 erwähnt, gibt es eine Vielzahl an unterschiedlichen Funk-Systemen. Da diese von verschiedenen Herstellern verbaut werden, können diese am sinnvollsten anhand der Einstell-Möglichkeit der verwendeten Funk-Adresse unterschieden werden. Haben zwei Produkte - nicht zwingend desselben Herstellers - dieselbe Einstell-Möglichkeit, so ist es relativ wahrscheinlich dass diese zueinander kompatibel sind. Dementsprechend wird es mit der *RotarySwitch*-Kompatibilität versucht, eine breite Anzahl an Geräten mit einem rotierbaren Einstell-Rad zu unterstützen.

Funk-System „DipSwitch“

Das verwendete Kommunikationsprotokoll ist hierbei beinahe identisch mit den RotarySwitch-Inkarnationen, jedoch gibt es bestimmte Unterschiede was die Adressierbarkeit der jeweiligen Geräte angeht, weswegen dies trotzdem als eigenständiges System betrachtet werden kann. Von außen betrachtet ist der große Unterschied die Einstell-Möglichkeit der verwendeten Adresse, da diese hier anstatt eines rotierbaren Rads über DIP-Schalter eingestellt werden.

LightStrip

LED-Lichtleisten sind eine sehr praktische Art, um farbliche Akzente oder Sekundärbeleuchtung (z.B. für Stimmungslicht) zu einer bestehenden Lichtquellenausstattung hinzuzufügen.

Das Lichtsystem „LightStrip“ ist eine Eigenentwicklung und ist entsprechend als zusätzliche Erweiterung zu dieser Arbeit konzipiert. Somit soll diese eher als Demonstration einer weiteren Lichtquelle und nicht als markttaugliches Produkt angesehen werden.

3.2.2 Geräteunterstützung vom User Interface

Um den Rahmen dieser Arbeit nicht unnötig auszuweiten wird sich hier auf den mobilen Google Chrome Browser sowie Android-basierte Tablet-Geräte ab Version 4.2.2 und mit mindestens 7 Zoll

Display-Größe beschränkt. Diese Werte wurden einerseits anhand der typischen Spezifikationen für derzeitig kostengünstig erhältliche Tablet-Geräte ermittelt, als auch basierend auf den für dieses Projekt bereits vorhandenen Hardware. Da jedoch auf HTML5-Standardkonformität geachtet wird, ist davon auszugehen, dass der Aufwand um die Unterstützung auf weitere Geräte und Browser zu erweitern, eher gering sein wird.

3.2.3 Usability

Die im vorigen Abschnitt definierte Displaygröße der verwendeten Hardware von 7 Zoll ist zwar eine übliche Größe für solch eine Bedienoberfläche, jedoch ist es trotzdem nicht viel Platz. Da Eingaben jedoch komfortabel und schnell durchgeführt werden sollen, bedarf es einer Oberfläche, bei der die Steuerelemente groß genug sind, um trotzdem beim ersten Versuch getroffen werden zu können. Dies ist stets ein Kompromiss, da in den meisten Fällen mehr Funktionen untergebracht werden könnten als es möglich ist wenn Steuerelemente platzintensiver konzipiert werden.

3.2.4 Erweiterbarkeit

Die große Stärke, welche aus dem Ansatz in Abschnitt 2.3 abgeleitet wurde, ist die Anpassbarkeit an unterschiedliche Umstände, also die Einbindbarkeit verschiedener Lichtquellen unterschiedlicher Herkunft, Funktionsweise und Anbindung. Die in Abschnitt 3.2.1 gelisteten unterstützen Geräte sind hierbei zwar ein sehr guter Anfang, jedoch ist dies selbstverständlich nur ein geringer Teil der potenziell möglichen Geräte. Dementsprechend soll das System auch darüber hinaus erweiterbar sein um perspektivisch noch mehr Lichtquellen einbinden zu können.

4 Entwurf

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

Als Projekt-Bezeichnung für die zu entwickelnde Lösung wurde *JarvisSuite* gewählt. „Jarvis“ ist hierbei eine Referenz . Das Suffix „Suite“ soll andeuten, dass dies ein Verbund aus mehreren Teilanwendungen darstellt und auch eher als Plattform für weitergehende Automatisierungen dienen kann.

4.1 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 und potenziell leistungsschwacher Komponenten untereinander. [5] [6]

Diese Kommunikation erfolgt über verschiedenartige Nachrichten

4.2 Nachrichten

Das Protokoll umfasst vier verschiedene Nachrichten-Arten. Diese Nachrichten werden als JSON-formatierter Text serialisiert übertragen.

Die verschickten Nachrichten unterteilen sich hierbei in zwei Kategorien: *Commands* und *Events*. Dies orientiert sich nach dem Architekturmuster *Command-Query Responsibility Segregation* (kurz: CQRS), nach dem eine Interaktion mit einem System auf zwei verschiedene Arten stattfinden kann. Entweder ist es eine befohlene Änderung (Command) oder die Abfrage des Resultats einer vorangegangenen Änderung (Event). [3] [7]

4.2.1 Commands

Für jede Veränderung, welche am Gesamtsystem (und somit den gesteuerten Lichtquellen) durchgeführt werden soll, bedarf es eines Commands. Das System kennt drei Command-Typen:

ToggleLightCommand

Funktion: Derzeitigen Lichtstatus umschalten. Dementsprechend entweder das Licht an- oder ausschalten.
Parameter: Zonen-ID

ActivatePresetCommand

Funktion: Ein gegebenes Preset aktivieren.
Parameter: Zonen-ID, Preset-Nummer

SwitchOffAllLightsCommand

Funktion: Das Licht in allen vorhandenen Zonen deaktivieren.
Parameter: Keine

4.2.2 Events

Events stellen Veränderungen dar, welche im System stattfanden. Hiervon gibt es zwei verschiedene Event-Typen:

LightSwitchedOffEvent

Ein *LightSwitchedOffEvent* wird im System generiert, wenn für eine Zone das komplette Licht ausgeschaltet wird. Das Gegenstück zu diesem Event, also ein „LightSwitchedOnEvent"', existiert nicht, da es faktisch äquivalent zu einem PresetActivatedEvent mit dem jeweils aktiven Preset ist.

Ein SwitchOffAllLightsCommand erzeugt mehrere Instanzen dieses Events (jeweils eins für jede Zone) anstatt ein kombiniertes Event, da die Zone eine Konsistenzgrenze darstellt und aus Sicht der Zone somit kein Unterschied besteht, ob der Befehl ein kombiniertes oder einzelnes Ausschalten darstellte.

PresetActivatedEvent

Ein *PresetActivatedEvent* wird erzeugt, wenn für einen Raum ein gespeichertes Preset aktiviert wurde. Hierbei ist nicht vorgeschrieben, in welchem Zustand sich das System zuvor befinden muss, also ob das Licht deaktiviert, ein anderes Preset aktiviert oder dasselbe Preset bereits aktiviert (aber Licht derzeit deaktiviert) sein muss. In jedem Falle wird der Zustand des gewünschten Presets hervorgerufen.

4.3 Steuersystem

Das *Steuersystem* ist die Serveranwendung, welche später die Befehle vom User Interface entgegennimmt, den aktuellen System-Zustand kennt und entscheidet, ob und wie der Befehl ausgeführt werden soll.

Implementiert wird diese Anwendung als .NET-Konsolenanwendung in der Programmiersprache C#. Hierbei wird jedoch darauf geachtet, dass diese Anwendung Mono-kompatibel bleibt. Mono ist eine freie Implementation des .NET Frameworks und eine Mono-Kompatibilität ermöglicht es, die Anwendung bspw. auf einem Raspberry Pi zu betreiben. Dies ist wichtig, da es ansonsten nicht-triviale Anforderungen an die Infrastruktur der jeweiligen Wohnung stellen würde, da ein entsprechender Server vonnöten wäre. Ein Raspberry Pi lässt sich jedoch leicht für den dauerhaften Betrieb versteckt verbauen.

Diese Anwendung in der Cloud zu hosten wurde zwar abgewägt, aber dagegen entschieden, da es zu schwer wiegen würde dass bei Verlust der Internet-Konnektivität potenziell die gesamte Wohnung nicht mehr kontrolliert werden könnte.

4.3.1 Datenmodell

Das Steuersystem beinhaltet ein Datenmodell, welches speziell für die Bedürfnisse der Lichtsteuerung konzipiert wurde.

Zone

Eine *Zone* definiert einen spezifischen Bereich innerhalb der Wohnung. Dies ist in den meisten Fällen schlichtweg ein Raum, wurde jedoch begrifflich offener benannt, damit auch Konstrukte, welche nicht direkt als Raum angesehen werden würden, wie z.B. eine Abstellkammer, eine Einbauküche oder ein Balkon, entsprechend abgebildet werden können.

Light

Eine *Light*-Instanz verkörpert eine sich in der Wohnung befindliche Lichtquelle, wobei dieser einer spezifischen Zonen-Instanz zugewiesen ist. Da verschiedene Lichtquellen unterschiedliche Merkmale und Möglichkeiten können (Nur An/Aus oder auch Darstellung von Farbe oder verschiedener Farbtemperaturen?), muss es somit verschiedene Unterarten von Lichtquellen geben. Diese Unterartgen gibt es für jede der unterstützten Lichtanbindungsmöglichkeiten, welche im Abschnitt *Analyse* erfasst wurden.

LightValueSet

Ein *LightValueSet* definiert einen Zustand, welche eine Light-Instanz haben kann. Dies verzweigt sich in die Unterarten:

- DisabledLightValueSet: ausgeschaltetes Licht
- EnabledLightValueSet: eingeschaltetes Licht
- BasicLightValueSet: eingeschaltetes Licht mit spezifischer Helligkeit
- ColorLightValueSet: eingeschaltetes Licht mit spezifischer Helligkeit, Farbton und Sättigung

ZonePreset

Eine *ZonePreset*-Instanz verkörpert die in vorigen Kapiteln beschriebenen Presets. Ein Preset gehört einer spezifischen Zone und beinhaltet für jede Light-Instanz innerhalb dieser Zone ein LightValueSet. Wird dieses Preset aktiviert, so wird für jede von den Light-Instanzen beschriebene Lichtquelle die von dem jeweiligen LightValueSet definierte Lichtsituation hervorgerufen.

4.3.2 LightConnector-Implementationen

LightConnector-Komponenten sind dafür da, die Konnektivität zu den einzelnen Geräte-Arten herzustellen und die entsprechenden Befehle daran weiterzuleiten. Hierfür besitzen diese zwei Funktionen: *SwitchOffLight* sowie *SetLightData*.

Für jede angebundene Geräte-Art bedarf es einer eigenen Unterart des LightConnectors. Basierend auf den in *Analyse* aufgestellten Geräte-Arten ergibt dies somit:

- PhilipsHueLightConnector
- IntertechnoLightConnector
- RotarySwitchLightConnector
- DipSwitchLightConnector
- JarvisLightStripLightConnector

Der *PhilipsHueLightConnector* fungiert als HTTP Client und sendet die Befehle als REST-Anfrage an die Hue API. Der *IntertechnoLightConnector*, *RotarySwitchLightConnector* sowie *DipSwitchLightConnector* geben die Befehle hingegen via dem MQTT-Server an das (in einem späteren Abschnitt beschriebene) Funk-Relay-Gateway weiter. Der *JarvisLightStripLightConnector* gibt die Befehle ebenfalls via dem MQTT-Server weiter, jedoch direkt an das jeweilige LightStrip-Gerät adressiert.

4.3.3 EventLogger

Der *EventLogger* ist eine Komponente, welche Änderungen innerhalb der Zonen-Instanzen registriert und basierend auf der spezifischen Änderung ein entsprechendes Event veröffentlicht. Da die Event-Typen anhand der Interaktionsmöglichkeiten mit dem System modelliert wurden, können Ereignisse von empfangenden Komponenten auf diese granularen Updates reagieren. [4]

4.3.4 ZoneStatePublisher

Im Gegensatz zum EventLogger, welcher granulare Veränderungen publiziert, ist der *ZoneState-Publisher* dafür da, in einem bestimmten Zyklus oder auf Anfrage den aktuellen Datenstand des Steuersystems anderen Komponenten zur Verfügung zu stellen.

Dies ist keine redundante Funktionalität, da beide Komponenten einen unterschiedlichen Use Case erfüllen. Während die Daten des ZoneStatePublishers sinnvoll sind, um die Initialdaten aufzubereiten, sind die Daten des EventLoggers dafür gedacht, diese auf dem aktuellen Stand zu halten. Theoretisch wäre es einer Downstream-Komponente (also einer Komponente, die von dieser hier abhängig ist) auch möglich, später (nach der Initialisierung) empfangene Datenpakete des ZoneStatePublishers mit dem aktuellen Datenstand abzugleichen, um potenziell aufgetretene Dateninkonsistenz durch verlorene EventLogger-Updates auszugleichen. Dies ist theoretisch möglich, aber sehr unwahrscheinlich. Da das System keinen Anspruch auf starke Konsistenz hat, kann dies eher als Erweiterungsmöglichkeit angesehen werden, da der zusätzliche Aufwand sich nach aktuellem Stand nicht lohnen würde.

4.4 Funk-Relay-Gateway

Das in Abschnitt 4.3 beschriebene Steuersystem ist von sich aus nicht in der Lage, Funk-Kommandos abzusetzen, da die zugrundeliegende Hardware (die Server-Hardware, bspw. Raspberry Pi) hierfür keine Schnittstellen zur Verfügung stellt. Dementsprechend bedarf es einer Komponente, welche diesen Aspekt der Kommunikation übernehmen kann.

Hierfür wurde das selbst-konzipierte *Funk-Relay-Gateway* vorgesehen. Aufbauend auf einem ESP8266-genannten Mikrocontroller und zwei verbauter Funk-Module soll es dem Gesamtsystem ermöglichen, Funk-Kommandos an Geräte zu schicken, die nur hierüber steuerbar sind.

Der ESP8266 bietet integrierte Wireless LAN Konnektivität, über die die implementierte Firmware nach Abschluss des Boot-Prozesses Verbindung zum MQTT-Server aufnimmt und auf weiterzuleitende Funk-Kommandos horcht. Empfängt es diese, so nutzt es die Funk-Module um – je nach anzusprechender Geräte-Art und verwendetem Protokoll – ein passendes Steuersignal zu senden.[2]

Die Notwendigkeit von zwei Funk-Modulen ergibt sich daraus, dass zwei verschiedene Code-Bibliotheken verwendet werden mussten um die drei in Abschnitt 3.2.1 geplanten Funk-basierten Lichtquellen anzusprechen. Da jede Bibliothek exklusiven Zugriff auf ein Funk-Modul benötigte, war es zwingend notwendig auch zwei hiervon zu verbauen.

4.5 User Interface

Das *Control-Interface* ist unterteilt in zwei Teile: Der Server-Teil, welcher als ASP.NET Webanwendung konzipiert ist, sowie der Client-Teil, welcher als HTML5-Webanwendung zur Verfügung steht.

Die Kommunikation zwischen diesen beiden Teilen erfolgt via AJAX, um Daten abzufragen oder Befehle abzusetzen, sowie SignalR, um Echtzeit-Aktualisierungen zu erhalten. SignalR ist eine Bi-

bliothek von ASP.NET um Echtzeit-Benachrichtigungen zwischen Anwendungen auszutauschen.

Der Server-Teil agiert hierbei auf den Daten, welche er vom ZoneStatePublisher (siehe Abschnitt 4.3.4) zur Verfügung gestellt bekommt. Direkt beim Start der Anwendung wird hierfür das Topic der ZoneStatePublisher-Komponente abonniert und die entsprechenden Daten angefordert.

4.5.1 Darstellung der Wohnung

In Abschnitt 3.1 wurde angemerkt, dass es vorteilhaft wäre, die Darstellung der Wohnung an die Wahrnehmung des Benutzers anzupassen. Nachfolgend sollen die möglichen Darstellungsarten der Wohnung evaluiert werden:

Variante „Liste“

Eine Listen-basierte Darstellung wäre der einfachste Weg. Diese erfordert keinerlei Kenntnisse über die wirkliche physische Anordnung der Zonen untereinander sowie der Komponenten darin, weswegen es von vielen (sehr allgemein implementierten) alternativen Lösungen verwendet wird. Diese Darstellungsart ist jedoch von den vorgestellten Arten diejenige, welche am weitesten von der Wahrnehmung des Benutzers abweicht.

Variante „2D“

Eine 2D-basierte Darstellung ist z.B. möglich, indem ein Grundriss der Wohnung dargestellt wird. Der Wiedererkennungswert kann hier gesteigert werden, indem durch Farben und Texturen einerseits die Räume leicht voneinander zu unterscheiden sind, als auch – vorausgesetzt es werden dieselben oder ähnliche Farben/Muster verwendet wie auch in der realen Inkarnation der Wohnung – die Assoziation mit den echten Räumen vereinfacht wird.

Variante „3D“

Für den Wiedererkennungswert der dargestellten Wohnung ist 3D die sinnvollste Darstellungsart, da diese am ehesten der Wahrnehmung des Benutzers entspricht. Da sich die zu steuernden Elemente jedoch überall in dem dargestellten 3D-Grundriss befinden können, wäre es notwendig, dem Benutzer Steuerungsmöglichkeiten an die Hand zu geben. Dies könnte z.B. bedeuten, dass der Benutzer den Grundriss um alle drei Koordinatenachsen rotieren kann.

Ein weiterer Nachteil ist der Aufwand, um ein solches 3D-Modell von einer vorhandenen Wohnung anzufertigen. Zwar gibt es bereits Verfahren, um dies anhand von Geräten (bestehend aus Kameras und Tiefensensoren) stark unterstützt durchzuführen, jedoch ist auch bei diesen ein nicht zu vernachlässigender Zusatzaufwand für die Nachbereitung vonnöten. Demnach ist dies höchstwahrscheinlich auch weiterhin ein manueller Prozess. Dies wäre jedoch akzeptabel, da es als Dienstleistung angeboten werden könnte und zumindest bei der Nutzung für das in Abschnitt 1.1 angedeutete Unternehmen sich entsprechend ausgebildete Designer um die Herstellung dieser 3D-Modelle kümmern würden.

Variante „Iso-3D“

Eine Alternative zu der puren 3D-Variante wäre isometrisches 3D. Hierbei wird das gleiche 3D-Modell angezeigt, jedoch erfolgt dessen Projektion aus einem schräg gehaltenen Winkel von einer weiter nach oben versetzten Perspektive. Dies führt dazu, dass theoretisch jedes Objekt innerhalb des 3D-Modells zu jedem Zeitpunkt sichtbar sein sollte ohne dass eine Änderung der Kamera notwendig wäre. Dementsprechend bietet es sich an, das Verändern der Kamera-Position und Kamera-Rotation vonseiten des Benutzers generell zu verbieten, um bei jeder Benutzung der Darstellung denselben Ausgangspunkt zu haben. Dies ist vorteilhaft, da dementsprechend alle verwendbaren Flächen stets dieselben Positionen haben und somit vom Nutzer für eine schnellere und leichtere

Benutzung erlernt werden können.

Um zu verhindern, dass die Wände des 3D-Modells dahinter befindliche Objekte verdecken, bietet es sich an, alle nach vorne gerichteten Wände teiltransparent darzustellen.

Variante „Hybrid“

Da keine der dargestellten Varianten für alle Anwendungsfälle korrekt sein wird, wird mit einer hybriden Lösung geplant. Um dem Benutzer eine Übersicht über die Wohnung zu bieten wird die 2D-Variante verwendet, für die Übersicht über einen Raum stattdessen die Iso-3D-Variante.

4.5.2 Feedback

Für solch ein Bedienelement ist es elementar, eine gute Möglichkeit des Feedbacks anzubieten, um den User erkennen zu lassen ob sein Befehl erkannt wurde.

Die Möglichkeiten des Feedbacks lassen sich in die Kategorien der fünf menschlichen Basissunne unterteilen. Ignoriert man hierbei den Geruchs- sowie Geschmackssinn, welche bei technischen Schnittstellen eher unüblich sind, so verbleiben der Hörsinn, der Sehsinn sowie der Tastsinn.

Hörsinn (Auditives Feedback)

Ein auditives Feedback kann u.a. über einen kleinen Lautsprecher erfolgen, der z.B. einen leisen Piep-Ton bei Erkennung eines Kommandos abspielt. Da die verwendete Plattform – das Android-Tablet – über solch einen Lautsprecher verfügt, kann dies somit leicht implementiert werden.

Auditives Feedback kann jedoch unpraktisch sein, denn ein leiser Ton wird bei entsprechender Umgebungslautstärke ggf. nicht gehört werden, ein zu lauter Ton kann jedoch für den Nutzer selbst oder für sich in der Nähe befindliche (potenziell schlafende) andere Nutzer als störend empfunden werden. Hierbei wird jeder Nutzer eigene Präferenzen haben, daher wird dies später eine konfigurierbare Option werden.

Sehsinn (Visuelles Feedback)

Ein visuelles Feedback wird ermöglicht, indem die Schaltflächen des *Control-Interface* bei einem Druck ihre Farbe ändern. Da diese ohnehin größer dimensioniert sein sollten als der Finger, der diese Schaltfläche drückt, besteht während des Drückens genug Sichtbarkeit auf die Schaltfläche um die Farbveränderung wahrzunehmen.

Das inhärente visuelle Feedback durch das Ein- und Ausschalten des Lichts für die betreffenden Funktionen kann nur teilweise als nutzbares Feedback angesehen werden. Dies würde voraussetzen, dass der Nutzer vom Standort des Bedienens Sichtkontakt mit dem gesteuerten Raum hat. Dies ist aber potenziell nicht der Fall, daher ist diese Art des Feedbacks eher als Bonus zu betrachten in den entsprechenden Fällen.

Tastsinn (Taktiles Feedback)

Ein taktiles Feedback kann erzeugt werden, indem der Vibrationsmotor des Tablet-Geräts nach dem Betätigen einer Schaltfläche kurz aktiviert wird, um so trotz der flachen Beschaffenheit eines Bildschirms das Gefühl für das Berühren zu simulieren.

5 Implementierung

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

5.1 Erschaffung des 3D-Grundrisses

Der 3D-Grundriss wurde in der Anwendung Cinema 4D vom Hersteller Maxon modelliert. Da kein 2D-Grundriss für die Demo-Wohnung vorlag, mussten die Abmessungen zunächst manuell ausgemessen und dann in das 3D-Modell übertragen werden. Dies war ein eher langwieriger Prozess, jedoch war dies für einen Prototypen akzeptabel da jedwede weitere 3D-Grundrisse weniger Zeit in Anspruch nehmen würden. Dies liegt daran, dass oftmals zumindest ein 2D-Grundriss vorliegen sollte und nach dem ersten angelegten Grundriss viele Komponenten hiervon (Wand-Objekte, etc.) für weitere Grundrisse übernommen werden könnten.

Des Weiteren wurde der Grundriss hier von einem Entwickler mit eher mäßigen Modellierungskenntnissen und -Fähigkeiten angelegt. Ein entsprechend ausgebildeter Designer würde höchstwahrscheinlich einen Bruchteil der Zeit erfordern.

Somit kann angenommen werden, dass die Erstellung eines 3D-Grundrisses innerhalb eines Produktivszenarios (wie z.B. als angebotene kommerzielle Leistung) für ein später auf dieses Projekt aufbauende Produkt ein möglicher Prozess sein kann.

5.2 Gestaltung des User Interface

Für solch ein User Interface schien es sinnvoll, auf ein eher dunkles Farbschema zu setzen. Das *Control-Interface* soll zu jeder Uhrzeit gut verwendet werden sollen. Entsprechende Display-Helligkeit, vorausgesetzt ist sowohl ein dunkles als auch ein helles Farbschema über einen Tag gut erkennbar. Jedoch nur ein dunkles Farbschema kann gut bei Nacht dargestellt werden ohne den Benutzer unangenehm zu blenden.

Als sekundärer Grund bietet sich der dunkle Hintergrund an, da die Preset-Farben, welche sich im User Interface wiederfinden, durch den Kontrast stärker zur Geltung kommen können.

Abbildung 5.1 zeigt das User Interface in der Grundriss-Perspektive. Die Grundriss-Grafik zeigt hierbei einen nicht komplett abgeschlossenen Zustand, jedoch ist klar erkenntlich welcher Struktur die Räume folgen und diese sind durch unterschiedliche Texturierung klar voneinander zu unterscheiden.

Wählt man in der Grundriss-Perspektive einen Raum aus, so gelangt man in die entsprechende Zonen-Perspektive, sichtbar in Abbildung 5.2. Hier wird zunächst ein Überblick über die möglichen Befehle gegeben. Diese orientieren sich an den in der *Analyse* aufgestellten erforderlichen Funktionen. Die sich an den Ecken der Schaltflächen befindlichen Farbverläufe geben an, welche primäre Farbe das jeweilige Preset hat, welches über die Schaltfläche aktiviert werden würde.

Die kleine 3D-Perspektive am oberen rechten Rand ist anklickbar und zeigt den aktuellen Raum als Iso-3D Perspektive. Über diese Ansicht werden die verschiedenen Lichtquellen als Piktogram-

Abbildung 5.1: User-Interface in Grundriss-Perspektive

me eingeblendet. Zum finalen Entwicklungsstand bieten diese jedoch noch keine Möglichkeit zur Veränderung. Diese Möglichkeit wird im *Ausblick* weiter erörtert.

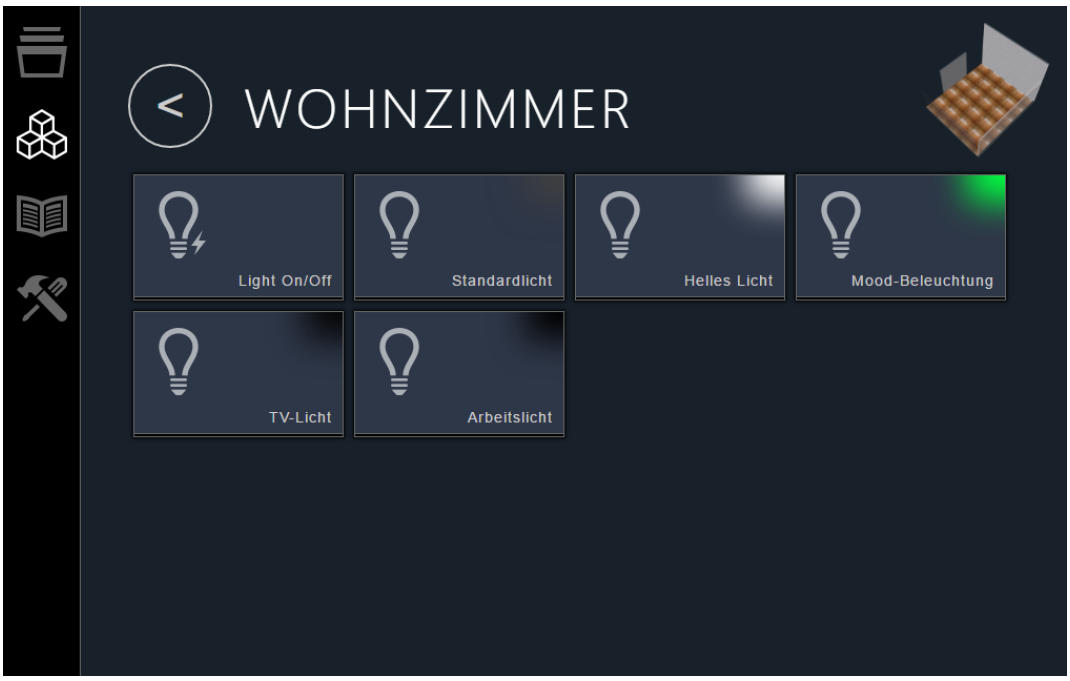

Abbildung 5.2: User-Interface in Zonen-Perspektive

Weitere Impressionen bezüglich des User Interface sind dem zu dieser Arbeit beigelegten Demonstrationsvideo zu entnehmen.

5.3 Umgang mit Farbwerten

Während der Implementierung fiel relativ schnell auf, dass es für die Darstellung einer einzelnen Farbe eine Vielzahl von möglichen Darstellungsarten gibt. Während Farben für die UI sowie die im Lightstrip verbauten Komponenten (und deren Zugriffsbibliotheken) ausschließlich auf RGB-Werte setzen, verwendet Philips Hue - treffend zum Namen - stattdessen den Verbund aus Hue (=Farbton), Helligkeit und Farbsättigung.

Die Skala, welche von Philips Hue verwendet wurde, schien aber ein zu spezifischer Wert um diesen allgemeingültig für alle angebundenen Systeme zu verwenden. Daher wurde stattdessen eine Skala von 256 Werten für das interne Management der *JarvisSuite* verwendet und der LightConnector-Implementation für Philips Hue Lichtquellen ein Skalierungsfaktor hinzugefügt, um beim Senden der Befehle an die Hue API den jeweiligen Hue Wert in die Philips Hue-eigene Skala umzurechnen. Die anderen Light-Connector-Implementationen, deren angebundenen Systeme RGB-Werte erwarten, erhielten stattdessen Umrechnungsfunktionen um diese Werte in das RGB-Farbschema umzurechnen.

5.4 Funk-Relay-Gateway

In Abbildung 5.3 ist die Platine des Funk-Relay-Gateways in dessen geöffneten provisorischen Gehäuse zu sehen. Die untere Platine ist hierbei der ESP8266-Mikrocontroller und die zwei vertikal verbauten kleineren Platinen sind die verwendeten Funk-Module. Diese wurden zusammen auf einer Lochrasterplatine verlötet und sind nicht-störend in einer Ecke der Demo-Wohnung versteckt worden. Da die Reichweite von der Wireless LAN Konnektivität sowie der Funk-Module mehr als ausreichend zu sein scheinen, lässt sich diese Komponente also leicht in eine bestehende Wohnung unterbringen ohne ästhetische oder Platz-basierte Probleme hervorzurufen. Einzig für eine Stromversorgung muss gesorgt werden (ersichtlich an dem in das Gehäuse geführte schwarze Stromkabel).

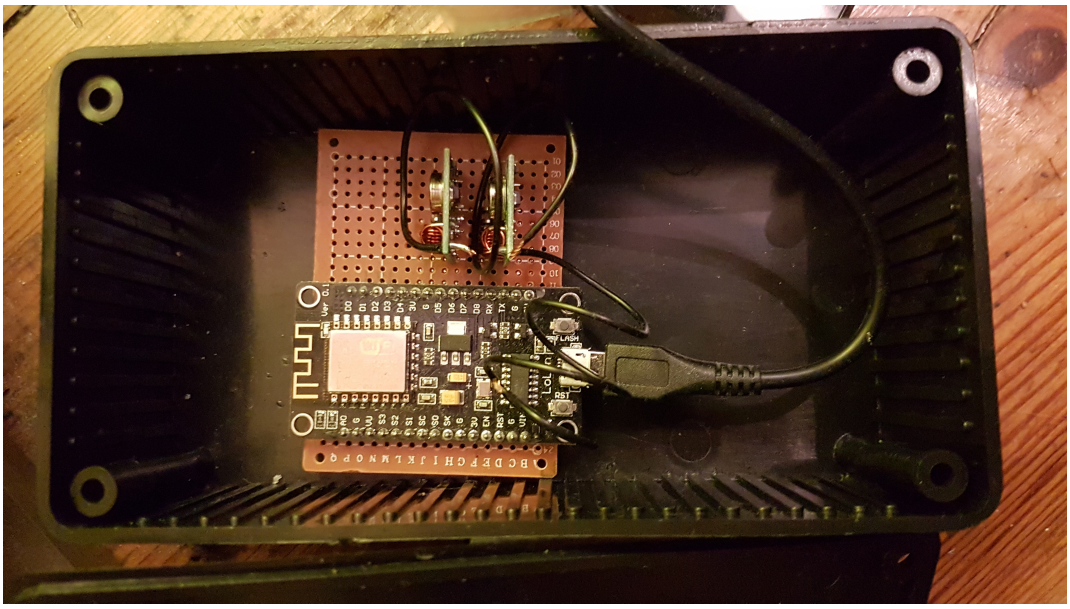

Abbildung 5.3: Funk-Relay-Gateway in provisorischem Gehäuse

6 Test

6.1 Test der funktionalen Anforderungen

Das *JarvisSuite*-Gesamtsystem wurde über einen Zeitraum von zwei Wochen durchgehend innerhalb der Demo-Wohnung produktiv angewendet. Hierbei ergaben sich, bis auf eine nachfolgend erläuterte Anmerkung, keinerlei Probleme und zu jedem Zeitpunkt war die Funktionalität vollständig einsetzbar.

Die funktionellen Aspekte der *JarvisSuite* können somit als ausreichend getestet angesehen werden.

6.1.1 Funk-Relay-Gateway

Bei zu schnell aufeinander folgenden Befehlen, welche über das Funk-Relay-Gateway weitergeleitet werden mussten, ergab sich eine gewisse Latenz. Dies war jedoch erst spürbar, wenn bspw. nach einem Anschalten eines Funk-angebundenen Lichts sofort der Befehl zum Ausschalten desselben Lichts zustandekam. Dies ist selbstverständlich kein allzu typischer Nutzungsfall und behindert somit die produktive Nutzung nicht weiter.

6.2 Test der nicht-funktionalen Anforderungen

Der Test der nicht-funktionalen Anforderungen fokussierte sich auf die korrekte Geräteunterstützung sowie die Usability des entwickelten User Interface.

6.2.1 Geräteunterstützung von Lichtquellen

Während der Testphase wurde die *JarvisSuite* mit einer breiten Palette an angebundenen Geräten betrieben. Folgende Geräte standen hierfür zur Verfügung:

- 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
- 2 „LightStrip“ LED-Leisten

Alle Geräte konnten erfolgreich angebunden und verwendet werden.

6.2.2 Usability

Innerhalb des in Abschnitt 6.1 beschrieben Zeitraums von zwei Wochen produktiver Nutzung konnten mehrere Nutzer dabei beobachtet werden, wie sie die entwickelte Lösung genutzt haben. Hierbei ergaben sich keine offensichtlichen Auffälligkeiten und alle Benutzer schienen vergleichsweise schnell mit dem User Interface zurechtzukommen. Diese eher subjektiven Beobachtungen ersetzen natürlich in keiner Weise eine objektive Studie mit einer aussagekräftigen Menge an Studienteilnehmern. Dies war jedoch nicht Fokus dieser Arbeit, daher wurde hierauf mit Sicht auf den Aufwand verzichtet.

7 Zusammenfassung

Das Ziel dieser Arbeit war die Entwicklung eines Heimautomations-Steuerungssystems, welche in der Lage ist, Funktionen verschiedener Lichtsysteme zu kombinieren und hierfür eine Bedienung mittels eines Tablet-basierten User Interface zu ermöglichen.

Zum Anfang dieser Arbeit wurde hierfür die Situation aktueller Lichtsteuersysteme sondiert und deren Bedienbarkeit, Automatisierbarkeit sowie deren deren Kompatibilität untereinander evaluiert. Hierbei ergab sich ein Lösungsansatz, welcher es ermöglichte, die Vorteile verschiedener dieser Systeme zu verbessern indem man diese miteinander kombinierte und diese Funktionalität unter einer einheitlichen Schnittstelle zur Verfügung stellte.

Während der Analyse wurden die benötigten Funktionen erörtert sowie die Menge der unterstützten Geräte auf fünf festgelegt.

Im Zuge des Entwurfs wurde die Gesamtkomplexität unterteilt und in mehrere Komponenten gruppiert. Eine Komponente, das sogenannte *Funk-Relay-Gateway*, benötigte Hardware welche zunächst konstruiert werden musste. Darüber hinaus wurde der passendste Weg gefunden, die zu steuernde Wohnung in der Anwendung abzubilden.

Im Rahmen eines zweiwöchigen Test-Zeitraums unter Produktivbedingungen erwies sich die entwickelte Lösung als korrekt funktionierend, stabil und produktiv anwendbar.

Insgesamt wurden alle Anforderungen an das Projekt hinreichend erfüllt, sodass es als Erfolg angesehen werden kann. Für weitere Projekte bietet es ein stabiles und erweiterbares Fundament, um weitere Funktionalitäten hinzuzufügen.

8 Ausblick

Während der Entwurf- und Implementationsphase ergaben sich viele Themengebiete und Ideen, welche der Funktionalität zuträglich gewesen wären, jedoch nicht im Rahmen dieser Arbeit genauer betrachtet werden konnten. Da die *JarvisSuite* auch als Plattform für spätere darauf aufbauende Lösungen dienen soll, bieten sich viele Erweiterungsmöglichkeiten an. Innerhalb dieses Kapitels werden nun ausgewählte Thematiken angeschnitten.

8.1 Lichtquellen direkt beeinflussen können

Eine Funktionalität, welche nützlich gewesen wäre, jedoch wegen de hierfür benötigten Aufwands nicht im Rahmen dieser Arbeit umgesetzt werden konnte, ist die Möglichkeit, einzelne Lichtquellen komplett frei anpassen zu können. Dies kann z.B. sinnvoll sein, wenn für ein sehr spezifisches Szenario nur eine einzelne Lichtquelle aktivieren werden soll (ggf. mittels einer benutzerdefinierten Farbe), für das es kein eigenes Preset gibt.

Hierbei müsste dann jedoch die Fragestellung gelöst werden, wie in einem solchen Falle mit den Presets umgegangen werden soll. Der derzeitige Entwicklungsstand wurde so konzipiert, dass zu jedem Zeitpunkt eins der definierten Presets aktiviert ist (wenn auch potenziell mit derzeit ausgeschaltetem Licht). Würde solch eine granulare und freie Anpassbarkeit implementiert werden, so würde man das System in einen (aus Preset-Sicht) undefinierten Zustand bringen können. Dies ist möglich, wird aber sicherlich verschiedene Grenzszenarien hervorrufen, welche gefunden und entsprechend bedacht werden müssten.

8.2 Ersatz für konventionellen Lichtschalter

Trotz der komfortablen Steuerung über das *Control-Interface* bedarf es natürlich noch einer sehr konventionell-gehaltenen Steuerung: der normale Lichtschalter. Durch die zwingend elektrisch-basierte Steuerung eines konventionellen Lichtschalters ist dieser nicht integrierbar in das Gesamtsystem der *JarvisSuite*. Des Weiteren würde dieser auch nur simple „Toggle“-Funktionalität bieten können.

Praktischer wäre es, wenn die dem Nutzer zur Verfügung stehende Funktionalität um das Haupt-Feature der *JarvisSuite* - dem Aktivieren eines Presets - erweitert werden könnte, ohne jedoch die inhärente Intuitivität eines Lichtschalters negativ zu beeinflussen.

Dieses Projekt hätte jedoch den Rahmen dieser Arbeit bei weitem überstiegen und wurde daher im Rahmen einer weiteren akademischen Arbeit durchgeführt. Beide Arbeiten wurden absichtlich komplett unabhängig voneinander entworfen und implementiert, jedoch mit genug Flexibilität, sodass diese nun zueinander kompatibel gemacht werden können. Dies ermöglicht es nun, erweiternde Arbeiten auf den Verbund beider Projekte aufzubauen.

8.3 Logging

Die implementierte EventLogger-Komponente bietet bereits die Grundlage, die dem System widerfahrenen Änderungen in Daten-modellierter Art und Weise abzufragen. Der Gedanke liegt also nahe, dem Benutzer diese Informationen in aufbereiteter Form darstellen zu können, um eine Möglichkeit der Überwachung der Vorgänge innerhalb der eigenen Wohnung zu bieten.

Literaturverzeichnis

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

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

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

[4] Fowler, Martin: *Patterns of Enterprise Application Architecture.* Addison-Wesley Professional, 2002. – ISBN 978-0-321-12742-6

[5] Hohpe, Gregor ; Woolf, Bobby: *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions.* Addison-Wesley Professional, 2003. – ISBN 978-0-321-20068-6

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

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