Frankfurt University of Applied Sciences
Fachbereich 2 - Studiengang Informatik

Bachelorthesis
zur Erlangung des akademischen Grades
Bachelor of Science

Konzeption und Realisierung eines Smart Mirror mit Hilfe eines Einplatinencomputers

Autor: Sven Fritz
Matrikel-Nummer: 1096233
Referent: Herr Prof. Dr. Baun
Korreferent: Herr Prof. Dr. Deegener
Abgabe: 30. November 2017

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit selbstständig und nur unter Zuhilfenahme der ausgewiesenen Hilfsmittel angefertigt habe. Alle Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter der Angabe der Quellen kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Frankfurt am Main, 29. November 2017

_____

Sven Fritz

Danksagung

Auf diesem Weg möchte ich mich bei allen bedanken, die mich bei meinem Studium begleitet haben. Insbesondere möchte ich mich bei meiner Familie bedanken.

Für die Betreuung und die Idee zu dieser Abschlussarbeit danke ich Prof. Dr. Christian Baun. Auch möchte ich mich bei Prof. Dr. Matthias Deegener für seine Bereitschaft, als Korreferent aufzutreten, bedanken.

Zusammenfassung

Diese Bachelorthesis beschäftigt sich mit der Erstellung eines smarten Spiegels. Hierzu gehören die Auswahl und der Zusammenbau der genutzten Komponenten sowie die Installation und die Konfiguration der Spiegelsoftware auf einem Raspberry Pi. Im Laufe der Arbeit wird gezeigt, wie weitere Funktionalitäten hinzugefügt werden können. Außerdem wird der aktuelle Stand der Technik bezüglich smarter Spiegel beleuchtet und auf rechtliche Fragen sowie auf die Datenschutzproblematik bei der Verwendung einer Sprachsteuerung eingegangen. Abschließend wird ein Ausblick auf zukünftige Anwendungsmöglichkeiten und Technologien gegeben.

Abstract

This bachelor thesis deals with the creation of a smart mirror. This includes the selection and assembly of the components used, as well as the installation and configuration of the mirror software on a Raspberry Pi. During the course of the thesis it will be shown how additional functions can be incorporated. Furthermore, the current state of technology relating to smart mirrors will be presented and legal as well as privacy issues resulting from the use of voice control will be expounded. Finally, a perspective is given on future application possibilities and technologies.

Inhaltsverzeichnis

Abbildungsverzeichnis VII

Abkürzungsverzeichnis VIII

1 Einleitung 1

1.1 Ziel 4

1.2 Datenschutz 5

1.3 Rechtliche Fragen 6

2 Stand der Technik 8

2.1 Smart Mirror 8

2.1.1 Komplettlösungen 8

2.1.2 Softwarelösungen 9

2.2 Raspberry Pi 11

2.3 Alternative Einplatinencomputer 13

2.4 Raspbian 13

2.5 Electron 14

2.1 AngularJS 14

2.2 REST API 15

3 Design 16

3.1 Hardwarekomponenten 17

3.1.1 Raspberry Pi 17

3.1.2 microSD-Karte 18

3.1.3 WLAN-Adapter 18

3.1.4 Webcam mit Mikrofon 18

3.1.5 Display 19

3.1.6 Netzteil 19

3.2 Softwarekomponenten 19

3.2.1 Raspbian 20

3.2.2 Smart Mirror Software ... 20
3.3 Spiegelscheibe ... 21
3.4 Rahmen ... 21
4 Implementierung ... 22
4.1 Zusammenbau des Spiegels ... 22
4.1.1 Spiegelscheibe ... 22
4.1.2 Rahmen ... 23
4.1.3 Montage der Komponenten ... 23
4.2 Raspbian-Installation ... 25
4.3 Raspbian-Konfiguration ... 26
4.4 Smart Mirror Software ... 28
4.4.1 Installation ... 28
4.4.2 Basiskonfiguration ... 29
4.4.3 Konfiguration der Smart Mirror Software ... 34
4.5 Erstellen eigener Plug-Ins ... 42
4.5.1 RMV ... 43
4.5.2 Wikipedia ... 48
4.5.3 Plex ... 50
4.5.4 CSS ... 56
4.5.5 Definition des Sprachbefehl ... 57
4.5.6 Einfügen der Plug-In-Komponenten ... 58
5 Fazit ... 60
5.1 Weitere Einsatzmöglichkeiten ... 61
5.2 Ausblick ... 62
Literaturverzeichnis ... 64

Abbildungsverzeichnis

Abbildung 1.1: Smart Home Marktprognose 1

Abbildung 2.1: Raspberry Pi 11

Abbildung 3.1: Blockschaltbild 17

Abbildung 3.2: Softwarearchitektur des Smart Mirror 20

Abbildung 4.1: Skizze des Rahmens 23

Abbildung 4.2: Rückseite des aufgebauten Spiegels 24

Abbildung 4.3: Etcher 26

Abbildung 4.4: Oberfläche des Smart Mirror 35

Abbildung 4.5: URL von Google Maps mit Geodaten 38

Abbildung 4.6: Medien-Information eines Liedes in Plex 51

Abbildung 4.7: Plex-Liedinformationen in XML 51

Abbildung 5.1: Ansicht des Spiegels 60

Abkürzungsverzeichnis

API ..... Application Programming Interface

ARM ..... Advanced RISC Machines

AUX ..... Auxiliary

BGB ..... Bürgerliches Gesetzbuch

CPU ..... Central Processing Unit

CSS ..... Cascading Style Sheets

DIV ..... Division

DP ..... DisplayPort

DVI ..... Digital Visual Interface

GB ..... Gigabyte

GPIO ..... General Purpose Input/Output

HAFAS ..... HaCon Fahrplan-Auskunfts-System

HDMI ..... High Definition Multimedia Interface

HTML ..... Hypertext Markup Language

HTTP ..... Hypertext Transfer Protocol

HTTPS ..... Hypertext Transfer Protocol Secure

ID ..... Identifier

IoT ..... Internet of Things

IP ..... Internet Protocol

JSON ..... JavaScript Object Notation

KI ..... Künstliche Intelligenz

LAN ..... Local Area Network

MIT ..... Massachusetts Institute of Technology

NAS ..... Network Attached Storage

OAuth ..... Open Authorization

QR ..... Quick Response

RAM ..... Random-Access Memory

REST ..... Representational State Transfer

RISC ..... Reduced Instruction Set Computer

RMV ..... Rhein-Main-Verkehrsverbund

RSS ..... Really Simple Syndication

SD ..... Secure Digital

URL ..... Uniform Resource Locator

URI ..... Uniform Resource Identifier

USB ..... Universal Serial Bus

VGA ..... Video Graphics Array

WLAN ..... Wireless Local Area Network

XML ..... Extensible Markup Language

1 Einleitung

Laut einer konservativen Marktprognose des Prüfungs- und Beratungsunternehmens Deloitte werden bis zum Jahr 2020 etwa eine Million Smart Home Haushalte in Deutschland existieren. Im Jahr 2013 existierten laut dieser Studie 315.000 Smart Home Haushalte.[1] Dies entspricht einem durchschnittlichen Anstieg von etwa 18 Prozent pro Jahr. Abbildung 1.1 stellt den Verlauf der Prognose für den oben genannten Zeitraum dar. Es zeigt sich, dass das Thema Smart Home aktuell immer mehr an Bedeutung gewinnt.

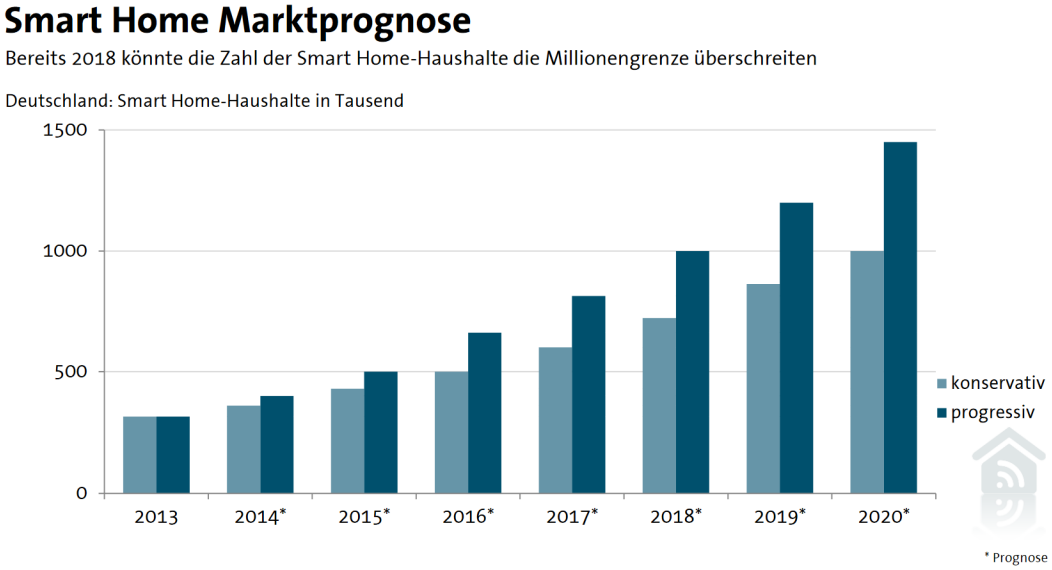

Abbildung 1.1: Smart Home Marktprognose

Quelle: https://www.bitkom.org/Presse/Presseinformation/Eine-Million-Smart-Homes-bis-2020.html, abgerufen am 3. Oktober 2017

Unter Smart Home versteht man die intelligente Vernetzung von Haushaltsgeräten, Sicherheitsanwendungen und Unterhaltungskomponenten zu Hause. Dies ermöglicht automatische Abläufe, die ferngesteuert werden können und somit das Leben der Bewohner vereinfachen. Smarte Gebäude besitzen eine bessere Energieeffizienz, mehr Gebäudesicherung und bieten durch intelligente Vernetzung mehr Komfort für die

[1] Vgl. (Bitkom, 2014).

Bewohner. Zum Beispiel könnte ein Kühlschrank durch Erkennung seines Inhalts abgestimmte Rezepte nennen. Morgens, wenn der Wecker klingelt, wird das Badezimmer geheizt und die Kaffeemaschine schaltet sich ein.[2]

Hierbei spielt der Begriff „Internet of Things" (IoT) eine große Rolle. Internet of Things wird im Deutschen auch „Internet der Dinge" genannt. Unter Dingen versteht man physische und virtuelle Gegenstände. Hier muss zwischen Verbrauchergeräten und Industriegeräten unterschieden werden. Für das Smart Home sind in erster Linie die Verbrauchergeräte relevant. Diese Verbrauchergeräte können Haushaltsgegenstände wie ein Kühlschrank oder eine Kaffeemaschine sein. Das Konzept von Internet of Things ist, Geräte zu vernetzen, sodass sie miteinander kommunizieren können. Die Geräte können intelligenter agieren und Prozesse automatisieren, da ihnen Informationen zur Verfügung stehen, die sie ohne Anbindung zu anderen Geräten oder zum Internet nicht hätten. Dies erleichtert das Leben der Smart Home Bewohner. Hierfür kann man sich folgendes Szenario vorstellen: Ein Bewohner befindet sich beim Einkaufen, während ein anderer Bewohner Nahrungsmittel aus dem Kühlschrank nimmt. Der smarte Kühlschrank setzt die entnommenen Nahrungsmittel auf die Einkaufsliste, welche mit dem Smartphone synchronisiert wird. Somit können während des Einkaufs noch Veränderungen an der Liste vorgenommen werden. Geräte, die über ein IoT-Konzept verfügen, nennt man „smart".

Für die Bewohner ist es komfortabel, ein zentrales Gerät zur Verfügung zu haben, über dessen Bedienoberfläche alle smarten Geräte im Haus bedient werden können. Dieses Gerät sollte außerdem in der Lage sein, Abläufe in dem Gebäude zu automatisieren.

Sprachassistenten übernehmen zurzeit die Aufgabe der Bedienung. Über sie kann zum Beispiel Musik abgespielt, die Beleuchtung geregelt oder der Einkaufszettel erstellt werden. Die Eingabe erfolgt über Sprache. Viele große Hersteller bieten einen Sprachassistenten an. Amazon hat im Jahre 2015 Amazon Alexa (Echo) auf den Markt gebracht. Nach der Veröffentlichung des Sprachassistenten war es das meistverkaufte

[2] Vgl. (Schiller, 2017).

Gadget bei Amazon. Bei Amazon hält sich der Sprachassistent seither unter den Top Ten in der Rubrik Elektronik.[3]

Amazon Alexa ist zurzeit der bekannteste Sprachdienst. Mehr als 70 Prozent der Nutzer von Sprachassistenten nutzen Alexa. Am zweithäufigsten genutzt ist der Sprachassistent von Google. Googles Sprachassistent heißt Google Home und wird von 23,8 Prozent der Nutzer genutzt. Google Home ist erst seit August 2017 in Deutschland verfügbar. Googles Sprachassistent ist etwas günstiger als Amazons Alexa, verfügt aber dennoch über sehr ähnliche Funktionalitäten. Beide Sprachassistenten können aus dem jeweiligen Musik-Store die gekaufte Musik abspielen. Es kann auch auf die Musik von Spotify, Deezer und Sound Cloud zugegriffen werden. Das Licht der Lampen des Beleuchtungssystems Philips Hue kann ebenfalls über sie gesteuert werden. So können die Lichtintensität und die Farbe per Spracheingabe eingestellt werden. Außerdem können über den Chromecast oder den Amazon Fire-TV sowie Fire-Stick Videos aus dem YouTube-Portal oder aus kommerziellen Streaming-Diensten wie Netflix oder Amazon Video abgespielt werden. Die verbleibenden 5,6 Prozent teilen sich die restlichen Sprachassistenten auf dem Markt. Hierzu gehört das Apple HomeKit von Apple, Cortana von Microsoft und Bixby von Samsung. All diese Sprachassistenten haben das Potenzial, ein Smart Home zu steuern. Teilweise verfügen sie schon über Schnittstellen zu smarten Geräten. Sprachassistenten könnten durchaus das künftige Bedienungsmodul sein, welches das Smart Home steuert.[4]

Sprachassistenten haben aber einen klaren Nachteil. Alle der genannten Sprachassistenten besitzen kein Display und können Informationen daher nicht visuell darstellen. Studien belegen, dass Gelesenes besser in Erinnerung bleibt als Gehörtes. Laut dieser Studien erinnert sich ein Mensch an 20 Prozent des Gelesenen. Nur 10 Prozent des Gehörten bleibt in Erinnerung.[5]

Es ist also von Vorteil, wenn Informationen visuell dargestellt werden können und nicht nur akustisch. Optimal wäre hierfür ein Gegenstand, den man täglich betrachtet und der schon in jedem Haushalt integriert ist. Ein Spiegel erfüllt dieses Kriterium. Laut

[3] Vgl. (Türck, 2017).
[4] Vgl. (Perez, 2017).
[5] Vgl. (Wyzowl, 2017).

Umfragen schaut die britische Frau 38 Mal am Tag in den Spiegel und britische Männer 16 Mal.[6] Ein Spiegel wäre also ein geeignetes Bedienungsmodul für ein Smart Home.

1.1 Ziel

Das Ziel dieser Arbeit ist, einen smarten Spiegel zu erstellen. Hierbei werden zunächst existierende Smart Mirror Lösungen vorgestellt. Im Weiteren werden verschiedene Möglichkeiten zur Realisierung eines Smart Mirror dargelegt und dann eine konkrete Implementierung eines Spiegels detailliert erläutert.

Der Spiegel soll ähnlich wie bei den genannten Sprachassistenten über eine Sprachsteuerung verfügen. Über Sprachbefehle sollen andere Smart Home Geräte bedient werden. Eine weitere Anforderung ist das Anzeigen von nützlichen Informationen auf der Spiegeloberfläche.

Bei der Spracherkennung ist es generell problematisch, dass die Spracheingabe nicht immer richtig interpretiert werden kann. Die Bedingungen während der Spracheingabe sind nicht immer optimal. Teilweise versteht die Spracherkennung einen Befehl nicht, weil er nicht deutlich genug ausgesprochen wurde oder weil Hintergrundgeräusche stören. Dies kann zu Fehlinterpretationen des Sprachbefehls führen. Es ist möglich, dass die Spracheingabe keinen der hinterlegten Befehle erkennt und daher keine Funktion ausgeführt wird. Problematischer ist, wenn die Spracheingabe falsch interpretiert wird, und ein anderer Befehl als der gewünschte erkannt und somit eine falsche Funktion ausgeführt wird. Zum Beispiel könnte anstatt der Beleuchtung die Kaffeemaschine eingeschaltet werden oder die Beleuchtung schaltet sich nicht in dem gewünschten, sondern in einem anderen Raum ein. Um diese Problematik zu umgehen, soll neben der Sprachsteuerung eine weitere Eingabemöglichkeit für die Spiegelbedienung betrachtet werden.

[6] Vgl. (Hagan, 2012).

1.2 Datenschutz

Laut der Bundesbeauftragten für Datenschutz Andrea Voßhoff sind intelligente Sprachassistenten, die ihre Umgebung ständig belauschen, aus Sicht des Datenschutzes kritisch zu bewerten. Die verbreitetsten Sprachassistenten verarbeiten die Daten nicht lokal, sondern schicken die Rohdaten an amerikanische Server, wo sie gespeichert und weiterverarbeitet werden. Google, Amazon und Microsoft verfügen über die Funktion, den Sprachaufzeichnungsverlauf ihrer Sprachassistenten anzuzeigen. Somit können Nutzer selbst nachvollziehen, welche Sprachbefehle gesprochen wurden. Allerdings sind die Datenschutzklauseln unklar formuliert und es befindet sich kein Hinweis darauf, wie lange die Daten gespeichert werden. Die Daten besitzen somit keine Löschfrist. Apple hingegen schreibt in der Datenschutzklausel, dass die gespeicherten Daten gelöscht werden, wenn der Sprachdienst Siri deinstalliert wird. Apple verknüpft die Spracherkennung nicht mit der Apple ID, sondern erstellt eine individuelle Kennung für jeden Siri-Nutzer. Durch Deinstallieren und Installieren der Siri-Anwendung kann die Kennung zurückgesetzt werden.

Problematisch ist auch, dass in der Cloud nicht nur Sprachaufnahmen gespeichert werden. Die Sprachassistenten schicken auch weitere Daten an die Hersteller. Hierzu gehören zum Beispiel IP-Adressen, Daten zu Hard- und Software, Such- und Shopping-Informationen sowie Standorte. Wegen mangelnder Transparenz ist nicht klar, zu welchem Zweck der Hersteller diese Daten speichert.

Die Nutzer müssten darüber informiert werden, wie, in welchem Umfang und wo die erfassten Daten verarbeitet werden und ihre Einwilligung dazu geben. Das Problem ist allerdings, dass zum Beispiel auch Sprachdaten von Besuchern aufgenommen werden, ohne deren vorheriges Einverständnis. Auch diese Daten werden in der Cloud gespeichert. Diese datenschutzrechtliche Schwachstelle ergibt sich dadurch, dass die digitalen Assistenten zurzeit noch nicht auf einzelne Personen trainiert werden können.

Heikel ist auch, dass Google sich vorbehält, Informationen aus verschiedenen Quellen zusammenzufügen. Hier könnte zum Beispiel die Stimme eines Gastes erkannt werden, da dieser den Sprachdienst von Google auch zu Hause nutzt.

Die Hersteller der Sprachassistenten sind zudem nicht die einzigen, die die in der Cloud gespeicherten Daten auswerten können. Diverse US-amerikanische Dienste und Behörden haben auch Zugriff auf die Daten, die auf US-amerikanischen Servern liegen. Im Jahre 2016 gab es bereits den Fall, dass die US-Polizei aufgrund eines Strafverfahrens die von Amazon Echo übertragenen Informationen auswertete.[7]

1.3 Rechtliche Fragen

Auch die Frage der Haftung spielt eine Rolle bei der Spracherkennung. Wenn der Sprachassistent die Eingabe falsch versteht und irrtümlich einen anderen als den gewünschten Befehl ausführt, ergeben sich Haftungsfragen.

Ist der Sprachassistent mit dem Smart Home verknüpft, könnte er zum Beispiel versehentlich eine smarte Tür verriegeln, sodass Personen eingesperrt oder ausgesperrt werden. Grundsätzlich haftet der Hersteller der virtuellen Assistenten. Nutzer müssen gegebenenfalls jedoch nachweisen, dass die Fehlfunktion nicht durch sie entstand. Das Szenario ist aber selbst für den Hersteller des Smart Home Gerätes kaum zu rekonstruieren. Deshalb ordnete der Bundesverband der Verbraucherzentralen eine Überprüfung des gesetzlichen Rahmens für smarte Produkte an. Des Weiteren sollen für die Lebensdauer eines smarten Geräts regelmäßige Updates eingeführt werden, damit keine Sicherheitslücken entstehen.

Eine weitere Haftungsfrage ergibt sich bei digitalen Vertragsabschlüssen mittels Sprache. Der Alexa-Sprachassistent ist in der Lage, Kaufgeschäfte abzuhandeln. Hierzu fragt Alexa „Willst du den Artikel jetzt kaufen?“, woraufhin der Nutzer den Kauf bestätigen kann. Dies ist jedoch noch kein Kaufvertrag, denn die Bestätigung stellt nur ein rechtlich bindendes Angebot zum Abschluss eines Kaufvertrags dar. Ein wirksamer Kaufvertrag entsteht erst bei der Bestätigung des Anbieters oder beim Senden des Artikels. Die Bestätigung wird meist als E-Mail verschickt. Wird die Frage zum Bestätigen des Einkaufes nicht gestellt, gibt es keinen gültigen Kaufvertrag laut §312j BGB.

[7] Vgl. (Heidrich & Mäkeler, 2017).

Bestellt eine nicht berechtigte Person einen Artikel, kann der Kaufvertrag aufgehoben werden. Dies nennt man Handel unter fremden Namen. Um über Alexa Artikel zu bestellen, muss ein Kode vom Nutzer genannt werden. Kennt die nicht berechtigte Person den Kode, weil der Nutzer den Kode nicht ausreichend gesichert hat, ist der Kaufvertrag dennoch unwirksam. Die einzige Ausnahme besteht bei Wissen des Nutzers und Tolerierung der nicht genehmigten Transaktionen. In diesem Fall würde der Nutzer haften.[8]

[8] Vgl. (Heidrich & Mäkeler, 2017).

2 Stand der Technik

Dieses Kapitel befasst sich mit den auf dem Markt befindlichen Smart Mirror Lösungen und geht auf mögliche Hardware- und Softwarekomponenten einer eigenen Lösung ein.

2.1 Smart Mirror

Zurzeit gibt es noch ein geringes Angebot an Smart Mirror Geräten. Der Smart Mirror Markt fängt langsam an zu wachsen. Immer mehr Smart Mirror Angebote finden sich auf dem Markt. Im Folgenden werden verschiedene Lösungen vorgestellt.

2.1.1 Komplettlösungen

Mehrere Hersteller bieten zurzeit Smart Mirror Geräte an. Hierzu gehört auch Samsung. Die Samsung Smart Mirror Produkte sind Bestandteil der LME Modellreihe. Die angebotenen Spiegel können extern gesteuert werden. Diese Funktion kann zum Beispiel ein Smartphone übernehmen. Die Spiegel verfügen über MagicInfo, was das Anzeigen von Inhalten aus verschiedenen Quellen ermöglicht. Durch über hundert Vorlagen kann der Benutzer die für ihn am besten geeignete Darstellungsform nutzen. Die Spiegel verfügen über einen Näherungssensor, der ermöglicht, unterschiedliche Informationen abhängig vom Abstand des Betrachters anzuzeigen. So könnte ein Spiegel in einem Restaurant Bilder der angebotenen Speisen anzeigen und bei Näherung des Betrachters zur Speisekarte wechseln. Über zwei USB Schnittstellen können Bilder und Videos von Speichermedien abgespielt werden. Des Weiteren verfügen die Spiegel über HDMI, DVI, DP und weitere Anschlussmöglichkeiten für weitere Geräte. Die LME Reihe gibt es in zwei verschiedenen Größen. Der Spiegel mit einer Anzeigegröße von 32 Zoll kostet etwa 1000 €. Bei dem größeren Model handelt es sich um einen 55 Zoll großen Spiegel, welcher etwa 2000 € kostet.[9]

[9] Vgl. (Samsung, 2017).

Ein weiterer Smart Mirror Hersteller ist das deutsche Unternehmen Mues-Tec. Das Unternehmen entwickelt seine eigene Smart Mirror Reihe. Anders als bei der Samsung LME Reihe geht die Anzeige nicht über die gesamte Spiegelfläche, sondern es werden nur in einem Ausschnitt des Spiegels Informationen angezeigt. Der Spiegel kann über ein integriertes Touchpad bedient werden. Zum Ein- oder Ausschalten muss zweimal kurz hintereinander auf den Spiegel getippt werden. Das Betriebssystem des Spiegels basiert auf Android. Aus dem Google Play Store können weitere Anwendungen heruntergeladen werden. Über Widgets können die Informationen der Anwendungen auf der Startseite angezeigt werden. Standardmäßig zeigt der Spiegel das Wetter und die Uhrzeit an. In dem Spiegel sind Lautsprecher, ein Mikrofon und eine Kamera integriert. Es ist somit möglich, auch Videos mit Ton zu schauen oder Musik zu hören. Die Kamera und das Mikrofon ermöglichen Videounterhaltungen mit Skype. Über Googles Sprachassistenten können Spracheingaben getätigt werden. Das eingebaute Bluetooth ermöglicht weitere Geräte wie die Personenwaage oder die elektrische Zahnbürste mit dem Spiegel zu koppeln. Beim Wiegen kann dann der Verlauf des Gewichts angezeigt werden. Für jeden Benutzer kann ein eigenes passwortgeschütztes Konto angelegt werden. Je nach Größe kosten die Spiegel zwischen 2500 und 4000 €.[10]

Spiegelshop24 entwickelt zurzeit einen eigenen Smart Mirror. Der Spiegel soll im unteren Preissegment angesiedelt sein und wird auf jegliche Interaktion verzichten. Angezeigt werden die Uhrzeit, das Wetter und aktuelle Nachrichten.[11]

2.1.2 Softwarelösungen

Die existierenden Softwarelösungen werden von namhaften Herstellern und kleineren Unternehmen, aber auch von Privatpersonen, die ihre Projekte veröffentlichen, angeboten.

Unter anderem stellt Microsoft ein Smart Mirror Projekt unter dem Namen Magic Mirror zur Verfügung. Die Software läuft auf einem Raspberry Pi der zweiten oder dritten Generation unter Windows 10 IoT Core. Bei der Anwendung handelt es sich

[10] Vgl. (Mues-Tec, 2017).
[11] Vgl. (Spiegelshop24, 2017).

um eine auf Node.js basierende Web-Anwendung. Die Magic Mirror Anwendung kommuniziert mit den Microsoft-Servern und holt sich Informationen ausschließlich von Microsoft. Der Magic Mirror kann Gesichter erkennen. Hierfür muss ein Konto auf dem Microsoft-Server angelegt werden. Nähert sich eine registrierte Person, nimmt die Kamera das Gesicht auf und schickt die Daten an den Microsoft-Dienst. Erkennt dieser das aufgenommene Gesicht, zeigt der Spiegel die zu dem Benutzerkonto passenden Informationen an. Wenn kein Benutzer angemeldet ist, werden Uhrzeit und Wetterdaten angezeigt.[12]

GlanCR arbeitet an der Erstellung eines eigenen Smart Mirror. Das Unternehmen stellt die Software frei zur Verfügung. GlanCR hat ein eigenes Betriebssystem namens Mirr.OS für den Raspberry Pi entwickelt. Nutzbar ist hier die zweite Generation mit einem WLAN-Adapter oder die dritte Generation der Raspberry Pi Reihe. Sobald der Raspberry Pi startet, wird das Smart Mirror Interface geladen. Um den Spiegel zu konfigurieren, muss sich beim ersten Starten der Spiegel-Software in das vom Raspberry Pi ausgestrahlte WLAN angemeldet werden. Hier kann der Raspberry Pi dann mit dem eigenen WLAN verknüpft werden. Der Spiegel wird über eine Web-Schnittstelle konfiguriert und bedient. Er zeigt die Uhrzeit, das Wetter und allgemeine Nachrichten an.[13]

Im privaten Bereich existieren eine Vielzahl an Smart Mirror Lösungen, die über Plattformen wie GitHub oder SourceForge veröffentlicht sind. Um einige dieser Smart Mirror Anwendungen haben sich Communitys gebildet, die ständig neue Funktionen in die Software einfügen. Dies ist auch bei der Smart Mirror Software von Evan Cohen der Fall. Was einst als Wochenendprojekt begann, wird nun schon seit über zwei Jahren weiterentwickelt. Auf der Anzeige werden allgemeine Informationen wie die Uhrzeit, das Wetter, aktuelle Nachrichten und Termine dargestellt. Der Smart Mirror von Evan Cohen lässt sich über eine Sprachsteuerung bedienen. Über Spracheingaben können dann zum Beispiel Videos von YouTube abgespielt werden. Die Spiegel-Software kann entweder auf einen Raspberry Pi der zweiten oder dritten Generation mit Raspbian oder einer anderen Linux-Distribution installiert werden. Genau wie der Magic Mirror

[12] Vgl. (Pavia, Stimac, & Richards, 2016).
[13] Vgl. (GranCr, 2017).

von Microsoft handelt es sich bei der Spiegel-Software um eine Node.js-Anwendung. Allerdings verfügt der Smart Mirror von Evan Cohen über viel mehr Funktionen als der Magic Mirror.

2.2 Raspberry Pi

Der Raspberry Pi ist ein Einplatinencomputer. Einplatinencomputer sind Computersysteme, die nur aus einer Leiterplatte bestehen. Alle notwendigen elektronischen Komponenten sind auf dieser Leiterplatte integriert.

Der Raspberry Pi wurde ursprünglich entwickelt, um bei Jugendlichen das Interesse für das Programmieren zu wecken. Der Grund hierfür war die sinkende Anzahl an Informatikstudenten an der Universität Cambridge. Ziel war es, einen günstigen Computer zu bauen, auf dem Jugendliche programmieren lernen konnten. Das erste Raspberry Pi Modell wurde Februar 2012 veröffentlicht. Innerhalb von 24 Stunden wurden über 100.000 Exemplare verkauft. Nur vier Jahre später kam die dritte Generation der Raspberry Pi Reihe auf den Markt.[14]

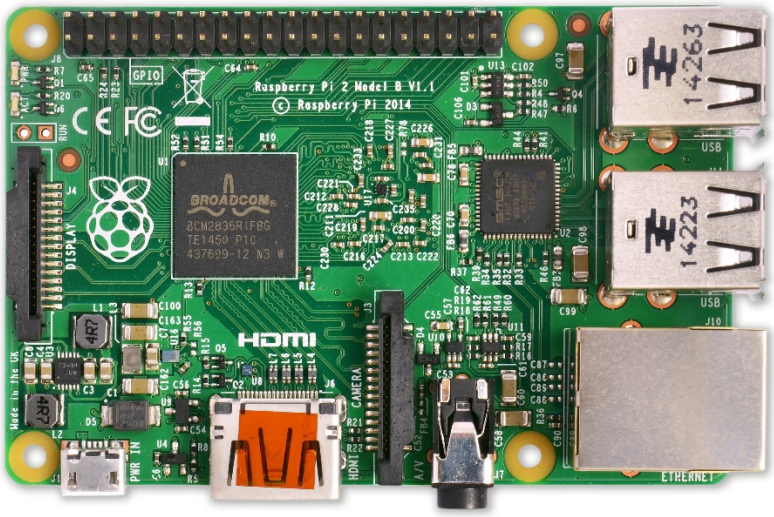

Abbildung 2.1: Raspberry Pi

Quelle: https://de.wikipedia.org/wiki/Raspberry_Pi, abgerufen am 3. Oktober 2017

[14] Vgl. (Schmidt, 2017).

Abbildung 2.1 zeigt einen Raspberry Pi der zweiten Generation. Zurzeit gibt es neun verschiedene Modelle des Raspberry Pi:

- 1. Generation
  - Raspberry Pi Model A (2012)
  - Raspberry Pi Model A+ (2014)
  - Raspberry Pi Model B (2012)
  - Raspberry Pi Model B+ (2014)
- 2. Generation
  - Raspberry Pi 2 Model B (2015)
  - Raspberry Pi 2 Model B 1.2 (2016)
- 3. Generation
  - Raspberry Pi 3 Model B (2016)
- Zero
  - Raspberry Pi Zero (2015)
  - Raspberry Pi Zero W (2017)

Der Raspberry Pi ist so groß wie eine Kreditkarte und besitzt einen microUSB-Anschluss zur Stromversorgung. Es gibt ein HDMI und einen AUX Ausgang für Bild und Ton. Bei der 3. Generation können über vier USB-Anschlüsse weitere Geräte an den Raspberry Pi angeschlossen werden. Ältere Modelle verfügen über zwei USB Schnittstellen. Über einen Ethernet-Anschluss kann der Raspberry Pi an ein Netzwerk angeschlossen werden. Ein Kartenslot ermöglicht die Nutzung von microSD-Karten als Datenträger. Auch hier gibt es Unterschiede zu älteren Modellen, bei denen SD-Karten als Datenträger zum Einsatz kamen. Durch 40 GPIO-Pins sind weitere Anschluss-möglichkeiten gegeben. Die dritte Generation des Raspberry Pi besitzt einen ARM Cortex-A53 Quad Core Prozessor und 1 GB RAM. Der Raspberry Pi läuft mit Linux-Distributionen, Android, Raspbian oder Windows10 IoT Core. Die Kosten des Raspberry Pi belaufen sich auf ca. 35 €. Der Zero Pi ist schon ab 10 € erhältlich. Die Zero-Reihe ist eine Abwandlung des eigentlichen Raspberry Pi. Der Raspberry Pi Zero

operiert auf einer kleineren Platine. Des Weiteren verfügt der Raspberry Pi Zero über andere Anschlüsse.[15]

Für die angebotenen Smart Mirror Softwarelösungen wird meist ein Raspberry Pi der zweiten oder dritten Generation empfohlen. Preislich liegen die beiden Modelle nicht weit auseinander. Die dritte Generation ist leistungsstärker als die zweite Generation. Die Leistungsfähigkeit eines Raspberry Pi 2 Model B reicht schon vollkommen aus, um eine Smart Mirror Anwendung zu betreiben. Benutzt man einen Raspberry Pi 2, wird ein extra WLAN-USB-Stick benötigt. In der dritten Generation sind ein WLAN- und ein Bluetooth-Adapter bereits auf der Platine installiert.

2.3 Alternative Einplatinencomputer

Alternativ zum Raspberry Pi gibt es den Banana Pi. Der Banana Pi besitzt einen A83T Octa-Core und verfügt über 2 GB RAM. Es können Linux-Distributionen, Android und Raspbian genutzt werden, um den Banana Pi in Betrieb zu nehmen. Der Banana Pi ist leistungsstärker als der Raspberry Pi 3. Ein Banana Pi kostet zurzeit etwa 50 €.

Eine weitere Alternative ist der Orange Pi. Der Orange Pi besitzt einen 1,6 GHz starken Quad Core Prozessor und 2 GB RAM. Auf dem Orange Pi können wie auch bei den anderen Einplatinencomputern entweder Linux-Distributionen, Android oder Raspbian installiert werden. Zwar ist der Orange Pi nicht so leistungsstark wie der Banana Pi, aber er ist schneller als der Raspberry Pi 3. Der Preis beläuft sich auf etwa 30 €.[16]

2.4 Raspbian

Raspbian ist das offizielle Betriebssystem für den Raspberry Pi und frei erhältlich. Es ist eine Linux-Distribution auf Basis von Debian. Raspbian wurde extra für den Raspberry Pi entworfen und ist für die Raspberry Pi Hardware optimiert. Raspbian ist nicht nur ein Betriebssystem, sondern enthält auch über 35.000 Software-Pakete, die

[15] Vgl. (Flaßkamp, 2016).
[16] Vgl. (Berkemeyer, 2016).

für den Raspberry Pi optimiert wurden. Genau wie Debian besitzt Raspbian das Grundprinzip der Stabilität.

Die erste Generation des Raspberry Pi besitzt eine CPU-Version, die von dem aktuellen Linux-Kernel nicht mehr unterstützt wird. Daher sind in Raspbian zwei Kernel-Versionen enthalten. Beim Starten des Raspberry Pi wird der für die CPU erforderliche Kernel aktiviert. Es gibt eine Lite-Version des Raspbian Betriebssystem, welche keine Desktop-Oberfläche besitzt. Diese Version benötigt weniger Speicherplatz als das Raspbian mit Desktop-Unterstützung.[17]

2.5 Electron

Electron ist ein Framework zur Erstellung von Web-Anwendungen. Programmiert werden die Anwendungen mit JavaScript, HTML und CSS. Mit Electron können die Web-Anwendungen auf Windows, MacOS und Linux übersetzt werden. Zur Darstellung der erzeugten Web-Anwendungen nutzt Electron den mitgelieferten Browser Chromium sowie Node.js. Electron ist ein quelloffenes Projekt von GitHub. Ursprünglich wurde Electron im Rahmen der Entwicklung des Quelltext-Editors Atom entwickelt. Erst im Jahre 2013 wurde Electron als separates Framework für die Öffentlichkeit freigegeben. Firmen wie Microsoft, Facebook, Slack und Docker benutzen Electron, um Web-Anwendungen zu kreieren. Auch die Smart Mirror Softwarelösungen von Microsoft und von Evan Cohen sind mit Electron erstellte Web-Anwendungen.[18]

2.1 AngularJS

AngularJS ist ein Framework für client-seitige Web-Anwendungen. Das Framework wurde von Google entwickelt und ist ein quelloffenes Projekt. AngularJS arbeitet nach dem Prinzip der Single-Page-Web-Anwendung.[19] Hierfür werden beim Starten der

[17] Vgl. (Wolski, 2017).
[18] Vgl. (Ackermann, 2017).
[19] Vgl. (Wikipedia, 2017).

Anwendung alle Darstellungsmodelle geladen. Beim Wechsel von einem Modell in ein anderes wird das neue Modell angezeigt und das alte in den Hintergrund geschoben.

Das Konzept der Single-Page-Web-Anwendung hat den Vorteil, dass das Darstellen der Webseiten schneller erfolgen kann als bei herkömmlichen Webseiten. Bei der Single-Page-Web-Anwendung sind die Darstellungsmodelle und die Geschäftslogik auf der Client-Seite vorhanden. Es müssen nur die darzustellenden Informationen vom Server angefragt und in das aktuell anzuzeigende Darstellungsmodell geladen werden. Herkömmliche Webseiten müssen die vollständige HTML-Seite vom Web-Server holen. Die Übertragung der Informationen benötigt einen geringeren Datenfluss als eine gesamte HTML-Seite und ist somit performanter.[20]

2.2 REST API

Der Begriff REST API setzt sich zusammen aus REST für „Representational State Transfer“ und API für „Application Programming Interface“. REST API ist eine Programmierschnittstelle, welche sich an den Paradigmen und dem Verhalten des World Wide Web orientiert. Hierbei beschreibt die Schnittstelle die Kommunikation zwischen Client und Server. Die REST API ermöglicht, dass verteilte Systeme miteinander kommunizieren können. Hierfür nutzt es standardisierte Verfahren wie HTTP/S, URI, JSON und XML.[21]

[20] Vgl. (Zeigermann, 2015).
[21] Vgl. (Srocke & Karlstetter, 2017).

3 Design

Zweck eines Smart Mirror ist, den Benutzer zu informieren und eine einfache Bedienung des Smart Home zu ermöglichen. Die angezeigten Informationen beinhalten unter anderem die aktuelle Uhrzeit, Wetterdaten, Nachrichten aus aller Welt, Termine, Aufgabenlisten sowie Informationen zum Fernsehprogramm. Zusätzlich ist die Darstellung von Daten verschiedener Anwendungen wie etwa YouTube und Fitbit möglich. Außerdem soll der Spiegel smarte Geräte im Haushalt bedienen können. Hierzu gehören das Abspielen von Musik oder die Steuerung von Smart Home Komponenten wie das Beleuchtungssystem Philips Hue.

Es ist eine unauffällige Integration des Spiegels in die Wohnung, insbesondere in das Badezimmer, wünschenswert, damit die Nutzung des Smart Mirror problemlos in den Alltag integriert werden kann. Das Resultat ist ein Spiegel mit einem neutral gestalteten Holzrahmen, welcher an der Wand befestigt werden kann und bei Nichtbenutzung wie ein normaler Spiegel aussieht.

Für die Bedienung des Smart Mirror soll eine benutzerfreundliche Steuerung gewählt werden. Die Bedienung erfolgt über eine Sprachsteuerung, welche eine leichte und schnelle Eingabe ermöglicht, ohne dass ein weiteres Gerät oder die Nutzung der Hände benötigt werden. Aus Datenschutzgründen und möglichen Fehlfunktionen bei Störgeräuschen oder zu leisem Reden soll eine weitere Steuerung zur Verfügung gestellt werden. Optionen für diese zusätzliche Bedienung sind ein berührungsempfindlicher Bildschirm, ein Tastenfeld, eine Infrarot-Fernbedienung, eine Tastatur, eine mobile Anwendung oder eine Webseite.

Die Problematik bei Benutzung eines berührungsempfindlichen Bildschirms ist, dass die Spiegeloberfläche sehr schnell verschmiert. Ein Tastenfeld und eine Infrarot-Fernbedienung verfügen aufgrund der wenigen Tasten nur über beschränkte Eingabemöglichkeiten. Bei der Nutzung einer Tastatur wäre eine Eingabe nicht beschränkt, jedoch muss diese als externes Gerät an den Spiegel angebracht werden, welches der problemlosen Integration in ein Badezimmer widerspricht. Bei der Benutzung einer mobilen Anwendung und einer Webseite wird ebenfalls ein externes Gerät benötigt.

Als zusätzliche Steuerung des Smart Mirror wird die Nutzung einer Website gewählt. Diese ist über mehr Geräte nutzbar als eine mobile Anwendung und es müssen keine weiteren Geräte direkt an den Spiegel angeschlossen werden.

Im Folgenden wird das Design des Spiegels dargelegt, welches auf eine Sprachsteuerung mit zusätzlicher Web-Schnittstelle angepasst ist. Dies beinhaltet die Beschreibung der Hardwarekomponenten, sowie die Ausführung der benötigten Softwarekomponenten für die Inbetriebnahme des Smart Mirror. Im Anschluss wird auf die erforderliche Spiegelscheibe und den Aufbau des Rahmens eingegangen.

3.1 Hardwarekomponenten

Der Smart Mirror besteht aus mehreren Komponenten. Diese sind in Abbildung 3.1 dargestellt und werden in den folgenden Kapiteln beschrieben.

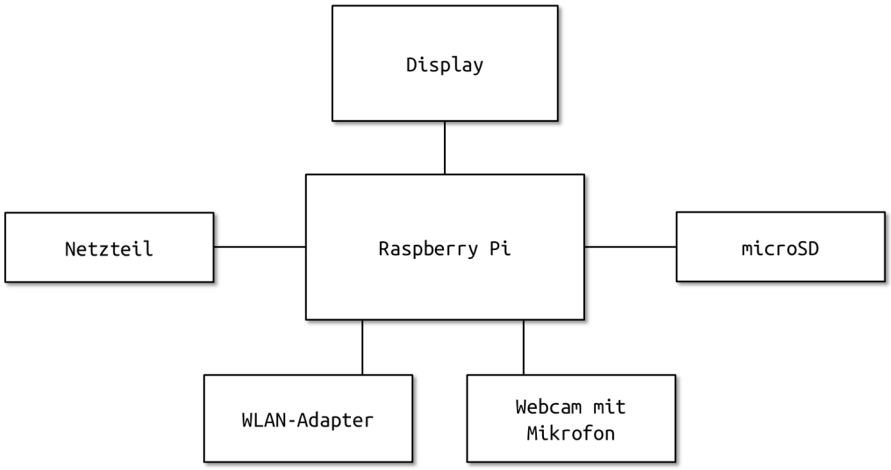

Abbildung 3.1: Blockschaltbild

3.1.1 Raspberry Pi

Für die Benutzung des Smart Mirror wird ein Raspberry Pi der zweiten oder dritten Generation empfohlen. Zu Beginn des Smart Mirror Projekts stand ein Raspberry Pi 2 Model B zu Verfügung. Da die zweite Generation der Raspberry Pi Reihe über

genügend Rechenleistung für die Smart Mirror Software verfügt, wird das zur Verfügung stehende Model benutzt.

3.1.2 microSD-Karte

Als Speichermedium dient eine microSD-Karte, die in den microSD-Karten-Slot des Raspberry Pi gesteckt wird. Auf dieser Karte befinden sich neben dem Betriebssystem die Smart Mirror Software sowie die Konfigurationsdaten. Da Raspbian bereits über 4 GB belegt, ist eine Speicherkarte mit einer Kapazität von 8 GB oder mehr erforderlich. Es ist üblich, für den Raspberry Pi microSD-Karten der Klasse 10 zu benutzten. Über die Klasse definiert sich die Schreibgeschwindigkeit der Karte. Klasse 10 garantiert eine Schreibgeschwindigkeit von 10 Megabyte pro Sekunde.[22]

3.1.3 WLAN-Adapter

Um den Raspberry Pi an ein Netzwerk anzuschließen, kommen zwei Möglichkeiten in Betracht. Bei der ersten Variante wird über den vorhandenen Ethernet-Anschluss ein LAN-Kabel angebracht. In der zweiten Variante verbindet sich der Raspberry Pi mit einem vorhandenen WLAN. Um störende Kabelverbindungen zu vermeiden, wird die WLAN-Variante verwendet. Da der genutzte Raspberry Pi über keinen integrierten WLAN-Adapter verfügt, wird ein WLAN-Adapter vom Typ „EDIMAX EW-7811UN“ an eine der USB-Schnittstellen angeschlossen. Für die Nutzung des Adapters muss kein Treiber installiert werden, da Raspbian diesen Adapter bereits unterstützt.

3.1.4 Webcam mit Mikrofon

Für die Spracheingabe wird das Mikrofon einer Webcam benutzt. In dem Smart Mirror Projekt von Evan Cohen wird empfohlen, die Kamera PlayStation Eye von Sony zu benutzen. Aber auch andere Webcams, die ein integriertes Mikrofon besitzen, können genutzt werden.[23] Da eine Microsoft LifeCam VX-700 vorhanden war, wird diese für

[22] Vgl. (raspberry.tips, 2015).
[23] Vgl. (smartmirror.io, 2017).

das Projekt benutzt. Die Webcam ist über eine USB-Schnittstelle an dem Raspberry Pi angeschlossen.

3.1.5 Display

Grundsätzlich kann jedes Display, das von dem Raspberry Pi angesteuert werden kann, für die Darstellung der Spiegelinformationen genutzt werden. Für dieses Smart Mirror Projekt wird ein 19 Zoll großer Monitor von Eizo, der FlexScan S1932-SH, eingesetzt. Der Monitor besitzt eingebaute Lautsprecher, die für die Tonausgabe genutzt werden. Der Monitor verfügt über einen VGA- und einen DVI-Anschluss. Der Raspberry Pi besitzt einen HDMI-Ausgang, der das Video- und das Audiosignal in digitaler Form liefert. Es bietet sich an, den DVI-Anschluss des Monitors zu nutzen, da dieser ebenfalls die digitale Übertragung erlaubt und außerdem das Audiosignal mit übertragen wird. Um die beiden Anschlüsse miteinander zu verbinden, ist ein passender Adapter erforderlich. Es wird ein entsprechendes Kabel eingesetzt, das auf der einen Seite einen HDMI-Stecker und auf der anderen Seite einen DVI-Stecker besitzt.

3.1.6 Netzteil

Für die Stromversorgung des Raspberry Pi wird ein Netzteil mit einer Ausgangsspannung von 5 Volt benötigt. Hierfür kann ein handelsübliches Netzteil mit einem microUSB-Stecker genutzt werden. Beim Raspberry Pi 2 ist darauf zu achten, dass das Netzteil eine Stromstärke von mindestens 1,2 Ampere liefern kann, um einen sicheren Betrieb zu gewährleisten.[24]

3.2 Softwarekomponenten

Für den Spiegel wird eine quelloffene Smart Mirror Software eingesetzt, die bereits über umfangreiche Funktionen verfügt. Als Basis dient das Betriebssystem Raspbian.

[24] Vgl. (Radke, 2013)

3.2.1 Raspbian

Das Betriebssystem Raspbian[25] wird für dieses Projekt genutzt. Von dem Entwickler der Smart Mirror Software wird das Betriebssystem Raspbian empfohlen. Raspbian ist eine extra für die Raspberry Pi Hardware optimierte Linux-Distribution. Das Betriebssystem läuft stabil und bietet die beste Unterstützung der Hardware. Außerdem beinhaltet Raspbian bereits einen Großteil der für die Spiegelsoftware benötigten Pakete.

3.2.2 Smart Mirror Software

Für dieses Smart Mirror Projekt wird die Software von Evan Cohen eingesetzt.[26] Die genutzte Version läuft stabil und verfügt bereits über viele Anbindungen zu diversen Internetdiensten. Die Software wird von Evan Cohen und anderen Entwicklern weiter gepflegt und regelmäßig um weitere Funktionalitäten erweitert. Die Software ist als Web- Anwendung implementiert. Die Softwarelösung ist gut dokumentiert und durch Nutzung von Electron ist ein einfaches Hinzufügen weiterer Funktionalitäten über Plug-Ins möglich.

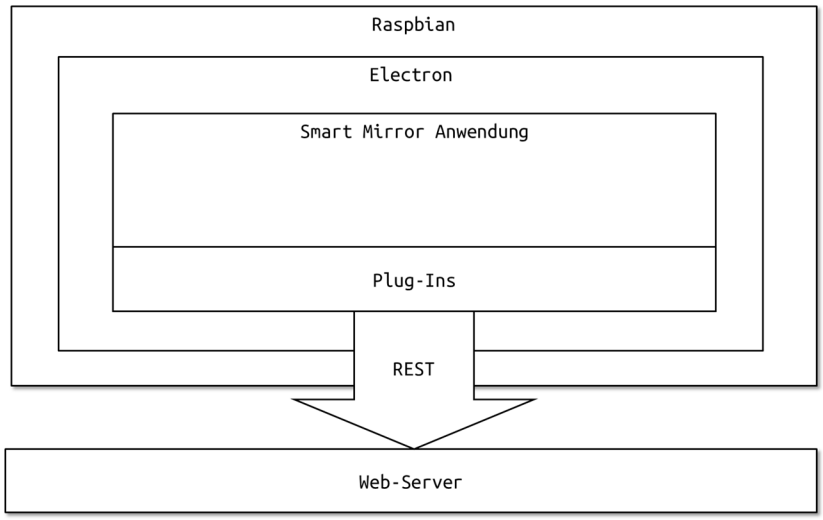

Abbildung 3.2: Softwarearchitektur des Smart Mirror

[25] Raspbian Jessie, veröffentlicht am 05.07.2017.
[26] https://smart-mirror.io/, Version 0.0.14, veröffentlicht am 08.06.2017.

Der Smart Mirror von Evan Cohen ist ein quelloffenes Projekt auf GitHub, das über die MIT[27]-Lizenz kostenloses nutzbar ist. Die MIT-Lizenz erlaubt es, die Software zu nutzen, zu verändern, zu verbreiten und zu verkaufen, ohne dass der Ersteller für mögliche Schäden haftet[28]. Abbildung 3.2 stellt die Softwarearchitektur des Smart Mirror dar.

3.3 Spiegelscheibe

Zum einen soll die Spiegelscheibe das Licht, das von der Vorderseite kommt, reflektieren, zum anderen sollen die Informationen, die der dahinterliegende Monitor darstellt, durch die Spiegelscheibe hindurchscheinen. Hierzu muss die Spiegelscheibe halbdurchlässig sein. Befindet sich ein Betrachter vor dem Spiegel, so kann er sein Spiegelbild wie auch die Darstellung des hinter der Spiegelscheibe befindlichen Monitors sehen, da die hellen Lichtpunkte auf dem Monitor durch die Scheibe hindurchscheinen.

Halbdurchlässige Spiegelscheiben können in individuellen Größen bestellt werden. Eine kostengünstigere Variante ist, eine normale Glasscheibe zu verwenden und diese mit einer spiegelnden Sichtschutzfolie zu versehen. Um Kosten zu sparen, wird für dieses Projekt die Variante mit der Folie benutzt.

3.4 Rahmen

Ein Rahmen soll die Spiegelscheibe umranden und diese mit dem Monitor mechanisch verbinden. Bedingt durch die individuellen Maße wird der Rahmen aus Holz selbst gebaut.

[27] Massachusetts Institute of Technology
[28] Vgl. (Open Source Initiative, 2017).

4 Implementierung

In diesem Kapitel wird beschrieben, wie der Smart Mirror implementiert wird. Hierfür wird erst der Spiegel aus seinen Komponenten zusammengebaut. Im Weiteren wird die Inbetriebnahme des Raspberry Pi und die Installation und die Konfiguration der Smart Mirror Software beschrieben. Zuletzt werden drei zusätzliche Plug-Ins entwickelt und in die Smart Mirror Software integriert.

4.1 Zusammenbau des Spiegels

Vor der Montage aller Komponenten muss erst die Spiegelscheibe vorbereitet und der Holzrahmen zusammengebaut werden.

4.1.1 Spiegelscheibe

Für die Spiegelscheibe wird eine normale Glasscheibe mit einer Spiegel-Sichtschutzfolie versehen. Die Glasscheibe sollte etwas größer als das 19 Zoll große Display sein. Vor der weiteren Bearbeitung muss die Glasscheibe erst gründlich gereinigt werden. Befinden sich noch Dreck, Fussel oder Staub auf der Scheibe, entstehen beim Auftragen der Folie kleine Luftblasen.

Vor dem Auftragen der Spiegel-Sichtschutzfolie müssen die Glasscheibe und die Folie mit Wasser eingesprüht werden. Das Wasser sollte einen Tropfen Spülmittel enthalten, damit die Oberflächenspannung des Wassers gebrochen wird. Dies hat den Vorteil, dass sich das Wasser besser verteilt und somit Luftblasen zwischen der Folie und der Glasscheibe vermieden werden können.

Nach Auflegen der Spiegel-Sichtschutzfolie befindet sich zwischen der Glasscheibe und der Folie eine Wasserschicht. Mit einer Plastikrakel wird nun das Wasser unter der Folie weggedrückt. Somit ist die selbstklebende Folie direkt auf der Glasscheibe befestigt. Die an den Rändern überstehende Folie wird mit einem scharfen Messer abgeschnitten.

4.1.2 Rahmen

Vier zu einem Rechteck verschraubte Holzleisten bilden das Grundgerüst und fassen den Monitor und die Spiegelscheibe. Der Rahmen ist etwas größer ausgelegt als die Spiegelscheibe, damit genug Spielraum für das Einsetzen der Scheibe gegeben ist. Auf diesen Holzrahmen werden vier Holzbretter, wie in Abbildung 4.1 gezeigt, angebracht. Die vier Holzbretter werden mit den Holzleisten verleimt.

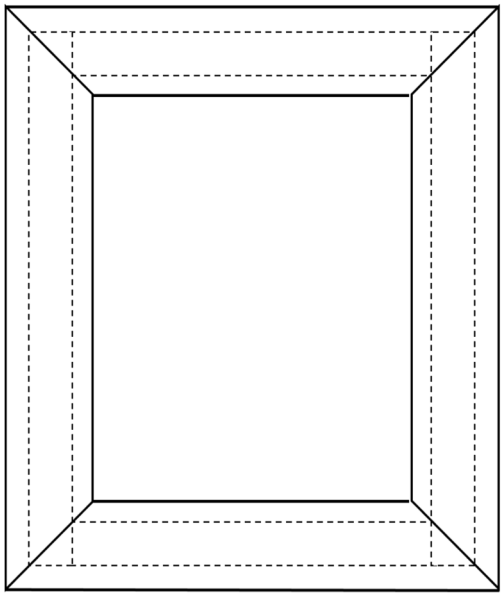

Abbildung 4.1: Skizze des Rahmens

4.1.3 Montage der Komponenten

Nun werden alle mechanischen und elektronischen Komponenten montiert. Zuerst wird die Spiegelscheibe in den Holzrahmen eingelassen. Es ist darauf zu achten, dass sich die mit Folie beschichtete Seite auf der Innenseite befindet. Direkt dahinter wird das Display angebracht. Hierfür muss das Displaygehäuse vor dem Einsetzen des Displays entfernt werden. Die vorderen Kanten des Metallrahmens sollten mit einem schwarzen Klebeband abgeklebt werden, damit diese nicht durch die halbdurchlässige Spiegelscheibe durchscheinen. Das Adapterkabel wird mit dem DVI-Anschluss des Displays und mit dem HDMI-Anschluss des Raspberry Pi verbunden. Der Raspberry Pi kann mit Kabelbindern am Display fixiert werden. An den Raspberry Pi werden die

Webcam und der WLAN-Adapter über die USB-Schnittstellen angeschlossen. Der microUSB-Anschluss für die Stromversorgung wird mit dem Netzteil verbunden. Bevor die microSD-Karte in den Karten-Slot gesteckt wird, muss auf diese noch das Betriebssystem Raspbian aufgespielt werden. Die Rückseite des fertig aufgebauten Spiegels ist in Abbildung 4.2 zu sehen.

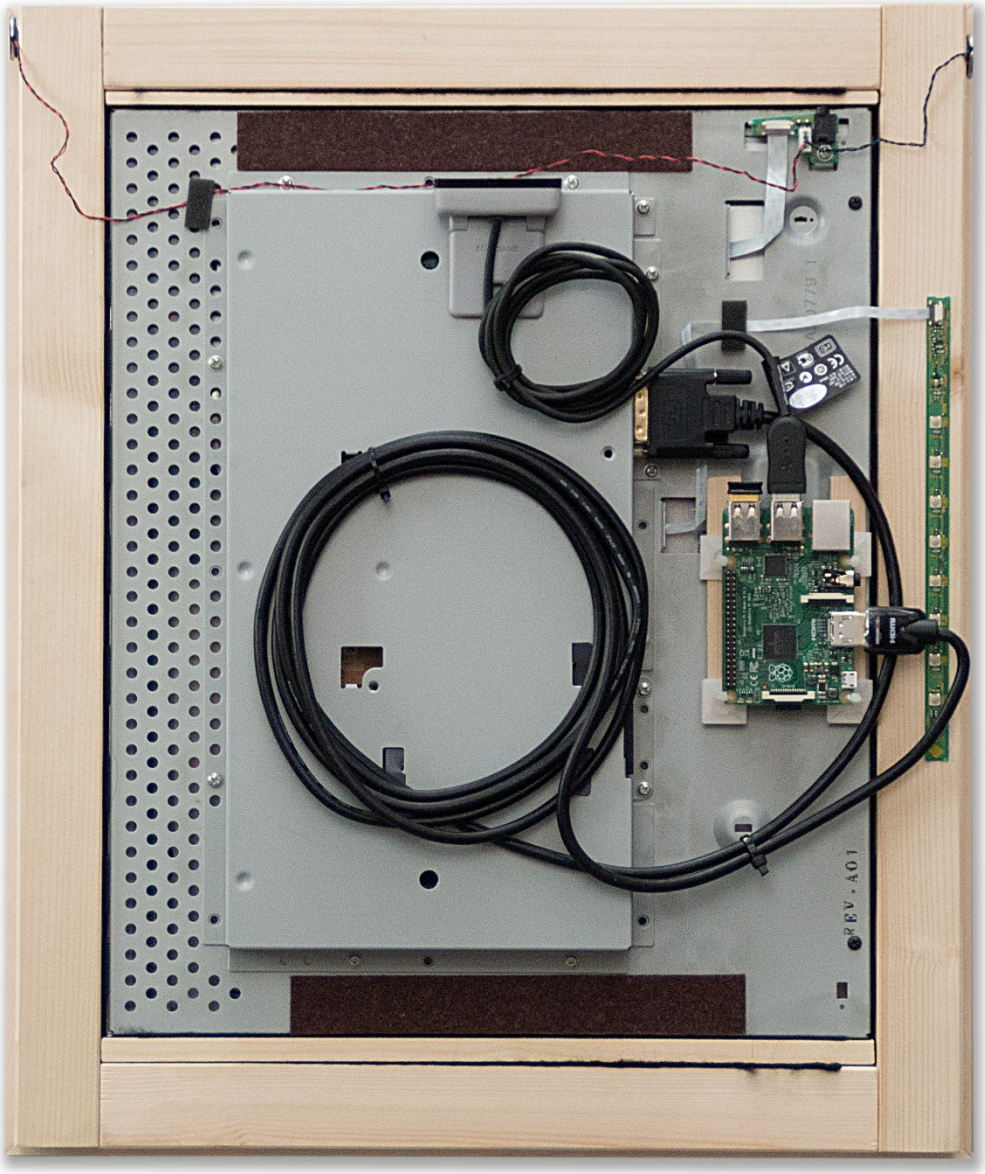

Abbildung 4.2: Rückseite des aufgebauten Spiegels

4.2 Raspbian-Installation

Zu Beginn muss das Betriebssystem Raspbian auf die microSD-Karte aufgespielt werden. Da das Betriebssystem nicht im Raspberry Pi auf die microSD-Karte installiert werden kann, erfolgt dies auf einem weiteren Computer. Hierzu wird ein Abbild des Betriebssystems benötigt. Dieses Abbild findet man auf der Internetseite der Raspberry Pi Foundation[29] unter der Rubrik „Downloads“. Auf dieser Seite befinden sich mehrere Betriebssysteme für den Raspberry Pi. Klickt man auf das Raspbian-Logo, gelangt man zu zwei Raspbian-Varianten. Da für den Smart Mirror eine graphische Oberfläche benötigt wird, muss die Variante mit der Desktopoberfläche heruntergeladen werden. Um an das Abbild zu gelangen, muss die heruntergeladene Zip-Datei entpackt werden. Unter Windows kann zu diesem Zweck die Anwendung 7-Zip benutzt werden, auf einem Linux basiertem Betriebssystem kann unzip benutzt werden. Für einen Mac eignet sich die Software The Unarchiver. Sobald das Abbild extrahiert wurde, kann es auf die microSD-Karte geschrieben werden.

Das Raspbian Abbild muss in einem bootbaren Zustand auf die microSD-Karte geschrieben werden. Hierfür wird ein Programm benötigt, das diese Aufgabe übernimmt. Die Raspberry Pi Foundation empfiehlt die Anwendung Etcher. Etcher ist mit Windows, MacOS und Linux kompatibel. Das Programm kann auf der offiziellen Etcher Internetseite heruntergeladen werden.[30]

Abbildung 4.3 zeigt die Bedienoberfläche von Etcher. Zuerst wählt man die Abbilddatei und die zu beschreibende microSD-Karte aus. Ein Klick auf die Schaltfläche „Flash!“ startet den Schreibvorgang. Die beschriebene microSD-Karte kann in den Kartenslot des Raspberry Pi gesteckt werden.

[29] https://www.raspberrypi.org/
[30] https://etcher.io/

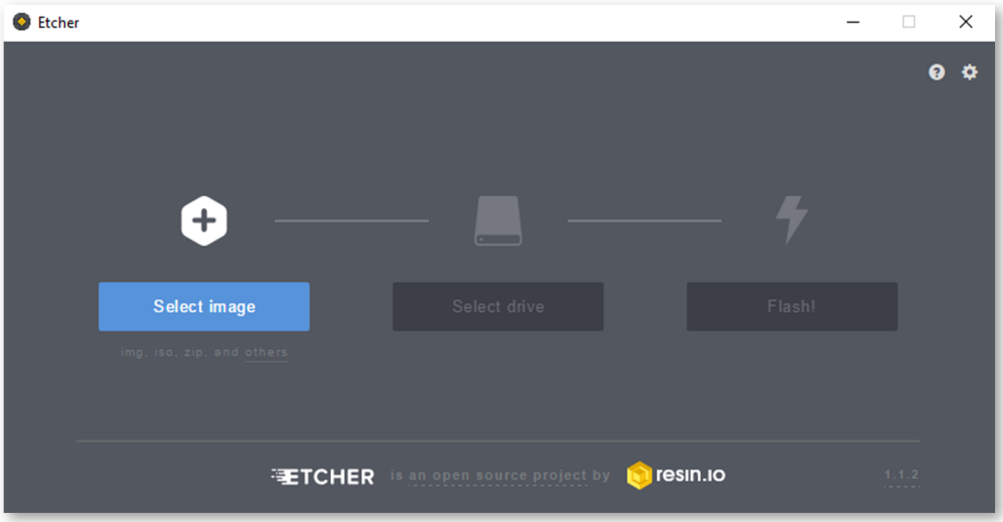

Abbildung 4.3: Etcher

4.3 Raspbian-Konfiguration

Für die Erstkonfiguration ist es erforderlich, eine Maus und eine Tastatur anzuschließen. Danach wird der Raspberry Pi gestartet. Während des Bootens meldet sich die graphische Benutzeroberfläche mit dem Benutzer „pi“ an. Der Benutzer gehört zur Gruppe „sudoer“ und kann Befehle mit Administrationsrechten ausführen. Nach dem Starten erscheint auf dem Desktop eine Warnung mit der Nachricht, dass für den Benutzer das Standardpasswort geändert werden sollte. Diese Nachricht kann erst einmal ignoriert werden, das Passwort wird später im Laufe der Konfiguration geändert.

Die erste Änderung, die durchgeführt werden muss, sind die Spracheinstellungen. Diese können unter „Application Menu > Preferences > Raspberry Pi Configuration“ eingestellt werden. Sobald man auf „Raspberry Pi Configuration“ klickt, öffnet sich ein neues Fenster. Unter der Kategorie „Localisation“ können Sprache, Zeitzone, Tastatur und „WiFi Country“ eingestellt werden. Bei „Set Locale…“ wird bei „Language“ „de (German)“ ausgewählt. Beim „Feld Country“ fügt man „DE (Germany)“ ein und bestätigt die Einstellung mit „ok“. Als nächstes wird bei der Zeitzone nach der „Area“ gefragt. Hier stellt man „Europe“ ein. Bei der „Location“ muss dann „Berlin“ angegeben werden. Bei der Tastatureinstellung muss „Germany“ als „Country“ angegeben werden.

In der „Variant“-Zeile muss berücksichtigt werden, ob die Tastatur Besonderheiten besitzt. Wenn dies nicht der Fall ist, kann „German“ ausgewählt werden. „WiFi Country“ wird auf „DE Germany“ abgeändert. Damit sind die Spracheinstellungen durchgeführt.

In der Kategorie „System“ kann ein anderer Hostname gewählt werden. Da der Raspberry Pi als Smart Mirror genutzt wird, bietet es sich an, den Hostnamen in „smartmirror“ zu ändern. Auch kann hier das Passwort für den Benutzer geändert werden. Als „current password“ muss hier „raspberry“ angegeben werden. Nach der Bestätigung der getätigten Einstellungen muss ein Neustart durchführt werden, damit die Änderungen wirksam werden.

Im nächsten Schritt können die Netzwerkeinstellungen durchgeführt werden. Hierzu wird das LXTerminal geöffnet und folgender Befehl eingegeben:

```
sudo nano /etc/network/interfaces
```

Nano ist ein Texteditor in Linux. Mit ihm ist es möglich, Textdateien darzustellen und zu bearbeiten. Dieser Befehl muss in diesem Fall mit Administrationsrechten aufgerufen werden, da die Datei sonst nicht gelesen und verändert werden kann. Sobald sich der Texteditor geöffnet hat, kann die Bearbeitung der Datei beginnen. Als Erstes müssen einige Zeilen auskommentiert werden. Sobald man ein „#“ einfügt, ist der nachfolgende Text in dieser Zeile auskommentiert. Folgenden Zeilen müssen auskommentiert werden:

```
allow-hotplug wlan0
iface wlan0 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Die folgenden Zeilen werden hinzugefügt:

```
auto wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa.conf
```

Die WLAN-Informationen werden in einer weiteren Datei definiert. Mit folgendem Befehl wird die Datei erstellt und kann direkt bearbeitet werden:

```
sudo nano /etc/wpa.conf
```

Es öffnet sich wieder der Texteditor Nano. Diesmal ist die Datei noch leer. Folgende Zeilen müssen eingegeben werden:

```
network={
ssid="WLAN-Name"
proto=RSN
key_mgmt=WPA-PSK
pairwise=CCMP TKIP
group=CCMP TKIP
psk="WLAN-Passwort"
}
```

Für „WLAN-Name“ und „WLAN-Passwort“ müssen die Werte des eigenen WLANs eingegeben werden. Auch hier ist wieder ein Neustart erforderlich.

4.4 Smart Mirror Software

Vor der Installation der Smart Mirror Software muss Raspbian auf den neusten Stand gebracht werden. Alle verfügbaren Updates können über zwei Befehle heruntergeladen und installiert werden.

```
sudo apt-get update
sudo apt-get upgrade
```

Nach der Durchführung der beiden Befehle befinden sich die aktuellen Versionen aller Anwendungen auf dem Raspberry Pi.

4.4.1 Installation

Die Installation der Smart Mirror Software erfolgt über ein bash-Skript, welches mit folgendem Befehl geladen und ausgeführt wird:

```
curl -sL https://raw.githubusercontent.com/evancohen/smart-
                mirror/master/scripts/pi-install.sh | bash
```

In dem Skript werden verschiedene Aspekte bearbeitet. Der erste Aspekt ist die Überprüfung, ob es sich um einen Raspberry Pi der zweiten oder dritten Generation handelt. Die Smart Mirror Anwendung funktioniert nur auf diesen beiden Raspberry

Pi Generationen. Ist das verwendete Gerät nicht kompatibel, wird die Installation abgebrochen. Nach dieser Überprüfung erfolgt die Installation verschiedener Anwendungen, die notwendig für die Benutzung des Smart Mirror sind. Für den Smart Mirror ist eine Mindestversion für Node.js vorgegeben, die auf dem Raspberry Pi installiert sein muss. Auch dies überprüft das Skript und installiert die neuste Version, falls nötig. Der nächste Schritt ist das Herunterladen des Smart Mirror Quelltexts. Über das Git-Repository kann der Quelltext geklont und auf den Raspberry Pi geladen werden. Außerdem müssen die AngularJS-Bibliotheken installiert werden. Der Mauszeiger wird so konfiguriert, dass er ausgeblendet wird, sobald die Anwendung geöffnet ist. Auch der Bildschirmschoner wird dann deaktiviert, der Smart Mirror verfügt über einen eigenen Bildschirmschoner. Als Letztes wird ein Alias zum Starten der Smart Mirror Software angelegt.

4.4.2 Basiskonfiguration

Nach der Installation der Smart Mirror Software müssen zuerst einige Konfigurationen durchgeführt werden, bevor die Spiegelsoftware genutzt werden kann. Hierzu gehören die Anzeige- und Audioeinstellungen, die Konfiguration der Spracherkennung und das Trainieren des Schlüsselwortes sowie das automatische Starten der Spiegelsoftware beim Hochfahren des Raspberry Pi.

4.4.2.1 Konfiguration des Displays

Der Smart Mirror kann im Quer- oder im Hochformat benutzt werden. Wird das Display im Hochformat aufgestellt, muss die Anzeige um 90 Grad gedreht werden. Dies kann in der Datei /boot/conf.txt geändert werden.

```
sudo nano /boot/conf.txt
```

Hier muss eine Zeile eingefügt werden, falls diese noch nicht vorhanden ist:

```
display_rotate=1
```

4.4.2.2 Audio Konfiguration

Bei dieser Konfiguration müssen die Kanäle für den Audioeingang und -ausgang definiert werden. Für die Ausgabe muss keine Auswahl getroffen werden, da das Audiosignal standardmäßig über das HDMI-Kabel übertragen wird. Anders ist es beim Audioeingang über das Mikrofon. Hier muss definiert werden, woher das Audiosignal genommen werden soll. Für den Smart Mirror wird eine Webcam benutzt. Um herauszufinden, wie die angeschlossene Webcam angesprochen werden kann, wird folgender Befehl ausgeführt:

```
arecord -l
```

Es werden die möglichen Aufnahme-Geräte angezeigt. Dies könnte wie folgt aussehen:

```
**** List of CAPTURE Hardware Devices ****
card 1: Camera [Vimicro USB2.0 UVC Camera], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Wichtig sind die Card- und die Device-Nummer der Webcam, welche für den Audioeingang benutzt werden soll. Aus den beiden Werten setzt sich die Gerätekennung nach folgendem Muster zusammen: „hw:“ gefolgt von der Card- und der Device-Nummer, getrennt durch ein Komma. In dem obigen Beispiel sieht der Wert folgendermaßen aus: „hw:1,0“. Laut der Smart Mirror Dokumentation ist die Gerätekennung für die Audioausgabe über das HDMI-Kabel „hw:0,0“. Beide Gerätekennungen müssen in die Konfigurationsdatei geschrieben werden. Mit dem folgenden Befehl werden erst die alten Audioeinstellungen gelöscht und eine neue Datei erstellt, in die die Konfigurationsdaten geschrieben werden können:

```
rm -f ~/.asoundrc && nano ~/.asoundrc
```

Der Texteditor Nano wird wieder gestartet. In die leere Datei müssen folgende Zeilen eingegeben werden:

```
pcm.!default {
  type asym
   playback.pcm {
     type plug
     slave.pcm "HDMI-Gerätekennung"
```

```
    }
    capture.pcm {
      type plug
      slave.pcm "Webcam-Gerätekennung"
    }
}
```

In die beiden „slave.pcm“-Werte müssen die zuvor bestimmten Gerätekennungen eingetragen werden.

4.4.2.3 Spracherkennung

Für die Spracherkennung (Speech Recognition) wird ein Cloud-Dienst von Google benutzt. Hierzu muss man sich beim Cloud-Dienst von Google[31] registrieren. Mit dem eingerichteten Account kann der Dienst „Google Cloud Speech API“ aktiviert werden. Hierfür muss der Cloud-Dienst unter der Rubrik „API & Dienste“ gesucht werden. In der Unterrubrik „Bibliothek“ sind alle APIs und Dienste aufgelistet. Ist der Cloud-Dienst aktiviert, kann der API-Schlüssel heruntergeladen werden. Die Funktion kann auch bei „API & Dienste“ gefunden werden. Klickt man auf die Unterrubrik „Zugangsdaten“, werden alle erstellten API-Schlüssel angezeigt. Um einen neuen API-Schlüssel zu generieren, klickt man auf „Anmeldedaten erstellen“. Als Nächstes muss ausgewählt werden, welche Identifizierungsart benutzt werden soll. Für die Google Cloud Speech API muss ein Dienstkontoschlüssel erstellt werden. Als Allererstes muss das Konto ausgewählt werden, für das dieser Schlüssel freigegeben werden soll. Hier kann das Konto „Neues Dienstkonto“ ausgewählt werden. Bei „Name des Dienstkontos“ kann zum Beispiel „smart-mirror“ angegeben werden. Als Schlüsseltyp muss „JSON“ ausgewählt werden. Nach Generierung des Schlüssels wird dieser heruntergeladen und in das Smart Mirror Verzeichnis unter dem Dateinamen „keyfile.json“ abgelegt.

[31] https://console.cloud.google.com

4.4.2.4 Trainieren des Schlüsselwortes

Der Spiegel hört permanent mit, was im Raum gesprochen wird. Versucht der Spiegel, alles Gesagte zu interpretieren und daraus Befehle abzuleiten, kann es leicht passieren, dass dieser in einem Gespräch zwischen zwei oder mehreren Personen einen Befehl interpretiert und diesen ausführt. Besser ist, den Spiegel gezielt mit einem sogenannten Schlüsselwort anzusprechen. Befehle werden nur dann vom Spiegel ausgeführt, wenn diese mit dem Schlüsselwort eingeleitet werden. Ist das Schlüsselwort zum Beispiel „Smart Mirror“, so muss ein Befehl mit genau damit beginnen, also etwa: „Smart Mirror, wie wird heute das Wetter?“. Fällt während einer Unterhaltung im Raum die Frage „Wie wird heute das Wetter?“, wird diese vom Spiegel nicht fälschlicherweise als Befehl interpretiert, da die Frage nicht mit dem Schlüsselwort eingeleitet wurde.

Um eine hohe Zuverlässigkeit bei der Schlüsselworterkennung zu erzielen, muss der Spiegel auf das Schlüsselwort trainiert werden. Dieses kann frei gewählt werden. Im Folgenden wird gezeigt, wie man den Spiegel auf das Schlüsselwort „Smart Mirror“ trainiert.

Hierfür muss auf dem Raspberry Pi das Terminal geöffnet werden. Im Terminal muss man erst einmal in das Smart Mirror Verzeichnis gelangen. Daraufhin muss ein Befehl ausgeführt werden, mit dem das Training des Schlüsselwortes beginnen kann. Folgende Befehle werden hierfür benötigt:

```
cd smart-mirror
npm run train-model
```

Der zweite Befehl startet ein Browser-Fenster und öffnet die Snowboy-Internetseite. Snowboy ist eine Software, die es ermöglicht, Schlüsselwörter („Hotwords“) zu trainieren. Diese Software kann zu privaten Zwecken kostenlos genutzt werden. Hierzu ist eine Anmeldung erforderlich, die mit einem GitHub-, Google- oder Facebook-Konto erfolgen kann. Nach erfolgreicher Anmeldung gelangt man zu einem bereits bestehenden Smart Mirror Projekt. Hier kann man ein eigenes Model erstellen. Klickt man auf „Record and download the model“, öffnet sich ein Unterfenster. In diesem müssen die Eingaben zur Erstellung des Models getätigt werden. Diese Eingaben unterteilen sich in drei unterschiedliche Angaben. Zu Allererst wird nach allgemeinen Angaben gefragt. Die Eingabefelder sind bereits vorbelegt und können beibehalten

werden. Mit der Schaltfläche „Record my voice“ gelangt man in die nächste Eingabeansicht. Hier muss drei Mal aufgenommen werden, wie man „Smart Mirror“ sagt. Die Eingabe kann entweder direkt über ein Mikrofon erfolgen oder eine bereits vorhandene Aufnahme sein, welche hochgeladen werden muss. Bessere Ergebnisse erzielt man mit der Nutzung des Mikrofons. Wurden die drei Spracheingaben getätigt, kann zur nächsten Eingabeansicht gewechselt werden. Dies wird über die Schaltfläche „Test the model“ erreicht. In der neuen Eingabeansicht werden erst einmal nach persönlichen Informationen gefragt. Hierzu gehören das Geschlecht und die Altersgruppe. Sind die beiden Angaben eingegeben, kann das Model getestet werden. Hierzu klickt man auf die Schaltfläche „Run the test“. Auch hier muss wieder „Smart Mirror“ gesagt werden. Sobald der Test die Wörter „Smart Mirror“ erkennt, gibt er „test succesfull“ aus. Über die Schaltfläche „Save and download“ kann die pmdl-Datei heruntergeladen werden. Danach muss diese in das Smart Mirror Verzeichnis verschoben und in „smart_mirror.pmdl“ umbenannt werden.

Der Smart Mirror vergleicht lokal das Gesagte mit dem hinterlegten Muster. Erst wenn das Schlüsselwort erkannt wurde, wird die nachfolgende Spracheingabe zu einem Cloud-Dienst gesendet und dort analysiert.

4.4.2.5 Autostart

Die Smart Mirror Anwendung soll beim Booten des Raspberry Pi direkt gestartet werden. Dies erfolgt über einen Cronjob. Cronjobs werden genutzt, um Aufgaben an bestimmte Ereignisse oder Zeiten zu binden. Der Job muss hierzu in die Datei /etc/crontab eingefügt werden. Mit folgendem Befehl kann die Datei bearbeitet werden:

```
sudo nano /etc/crontab
```

In die Datei muss eine neue Zeile für den Job eingefügt werden, die das Ereignis, den Benutzer und den Pfad zu dem auszuführenden Skript enthält. Die Smart Mirror Software liefert bereits ein Skript, das den Start der Anwendung durchführt. Es muss folgende Zeile eingefügt werden:

```
@reboot    pi    /home/pi/smart-mirror/scripts/bash-start.sh
```

Nun wird beim Hochfahren des Raspberry Pi die Smart Mirror Anwendung automatisch gestartet.

4.4.3 Konfiguration der Smart Mirror Software

Befindet man sich im Home-Verzeichnis des Benutzers „pi“, startet folgender Befehl die Smart Mirror Anwendung:

```
mirror start
```

Führt man den Befehl aus, öffnet sich die Smart Mirror Anwendung. Die Anwendung arbeitet im Vollbildmodus. Auf der linken Seite oben wird die aktuelle Uhrzeit angezeigt. Im unteren Bereich befindet sich ein Hinweis, über welchen Sprachbefehl sich der Benutzer die Liste aller Befehle anzeigen lassen kann. In der Mitte des Bildschirms befindet sich eine URL und ein QR-Kode, der diese URL darstellt.

Unter dieser URL gelangt man auf eine Web-Oberfläche, welche zur alternativen Bedienung des Smart Mirror genutzt werden kann. Die Befehle können hier entweder in ein Eingabefeld mit einer Tastatur eingegeben oder hineingesprochen werden. Auf der Seite befinden sich außerdem noch fünf Schaltflächen, die zum Beispiel den Vollbildmodus beenden und einschalten. Auch kann hier der Schlafmodus aktiviert werden.

Durch Klicken auf das Zahnradsymbol in der rechten oberen Ecke öffnet sich die Konfigurationsseite. Hier können einige generelle Einstellungen vorgenommen werden. Auch die bereits vorinstallierten Plug-Ins müssen hier vor der Benutzung konfiguriert werden.

Abbildung 4.4: Oberfläche des Smart Mirror

Abbildung 4.4 stellt die Oberfläche des fertig konfigurierten Smart Mirror dar.

4.4.3.1 Generelle Einstellungen

Zuerst sollte man die generellen Einstellungen vornehmen. Hierzu gehört die Sprache des Smart Mirror. Standardmäßig ist in der Smart Mirror Software die englische Sprache eingestellt. Gibt man in dem Feld „Language“ „de-DE“ an, wird die Sprache auf Deutsch geändert. Des Weiteren kann das Layout der Anzeige auf dem Spiegel verändert werden. Standardmäßig ist das „main“-Layout ausgewählt. Zusätzlich steht das „icesnow“-Layout zur Verfügung. Layouts werden mittels CSS-Dateien definiert. Eigene Layouts können einfach durch Erstellen und Hinzufügen der entsprechenden CSS-Dateien realisiert werden. In dem Feld „Commands Per Page“ kann angegeben werden, wie viele Sprachbefehle auf einer Seite angezeigt werden sollen.

4.4.3.2 Spracheinstellung

In der Spracheinstellung werden die Schlüsselwörter definiert, die den Spiegel aktivieren. Standardmäßig ist das Schlüsselwort „smart mirror“ definiert. Hier ist der Name des Schlüsselworts und der Pfad zu der Datei angegeben, die das zuvor mit Snowboy erstellte Modell enthält. Durch Klicken auf die Schaltfläche „+“ kann man weitere Schlüsselwörter hinzufügen. Daraufhin erscheinen zwei neue Felder, in denen der Name des neuen Schlüsselworts und der Pfad zu der entsprechenden Modelldatei eingegeben werden müssen. Des Weiteren befindet sich in der Spracheinstellung ein Feld mit dem Pfad zur Authentifizierungsdatei des Google Cloud Speech Dienstes. Auch die Mikrofoneinstellungen befinden sich hier. Zum einen kann das Aufnahmegerät ausgewählt werden, zum anderen kann die Sensibilität des Mikrofons justiert werden.

4.4.3.3 Remote Zugriff

Bei den Remote Einstellungen gibt es zwei Optionen, die verändert werden können. Die erste Option erlaubt es, den Remote-Zugriff zu aktivieren oder zu deaktivieren. Die zweite Einstellungsoption ist die genutzte Portnummer für das HTTP-Protokoll. Standardmäßig ist der Port auf 8080 gelegt. Der Benutzer kann hier jeden beliebigen Port wählen. Die Portnummer sollte sich im Intervall von 1024 bis 65535 befinden, da die restlichen Ports für Systemdienste reserviert sind.

4.4.3.4 Auto-Timer

Der Auto-Timer schaltet das Anzeigen der Informationen aus, wenn der Spiegel eine bestimmte Zeit nicht bedient wurde. Hierfür muss erst einmal die Art des Displays ausgewählt werden. Soll die Funktion nicht genutzt werden und der Spiegel permanent Informationen anzeigen, kann hier „Disabled“ ausgewählt werden. Direkt darunter kann eine Aufweckzeit eingestellt werden, zu der das Display des Spiegels wieder startet. Unter „Auto Sleep Wait Time“ kann die Anzahl der Minuten angegeben werden, nach denen die Anzeige abgeschaltet wird. Hier befinden sich auch die Pfade der Skripte, die zuständig für das Ausschalten und Einschalten des Bildschirms sind.

4.4.3.5 Kalender

Auf dem Spiegel können verschiedene Kalender und die darin enthaltenen Termine angezeigt werden, indem man die entsprechenden Kalender-URLs auf der Konfigurationsseite eingefügt. Eine Kalender-URL erhält man, indem man den betroffenen Kalender freigibt. Dies ist bei Microsoft und Google unter Kalender-Einstellungen möglich. Bei Apple kann der Kalender über die iCloud freigegeben werden.

4.4.3.6 Fitbit

Fitbit ist ein Unternehmen, das sich im Bereich „Technik für Fitness“ spezialisiert hat. Hierfür entwickelt Fitbit Armbänder und Uhren, welche die Bewegungen und biometrische Daten des Trägers aufzeichnen. Die Geräte besitzen Sensoren, über die Bewegungen, Geodaten oder auch die Herzfrequenz erfasst werden. Die gesammelten Daten werden über Bluetooth mit dem Handy oder Tablet synchronisiert und in die Cloud geladen. Über die Fitbit-App können die Daten analysiert und angezeigt werden. Zum Beispiel kann der Nutzer hier einsehen, wann seine Tiefschlafphasen waren. Auch die Strecken, die man gejoggt oder mit dem Fahrrad gefahren ist, können angezeigt werden. Der Spiegel kann diese Daten darstellen, sobald sie in der Cloud liegen. Hierzu muss man bei Fitbit eine neue Anwendung registrieren. Dies kann auf der Seite https://dev.fitbit.com/apps/new geschehen. Sobald man sich mit seinem Fitbit-Konto angemeldet hat, kann die Registrierung der Smart Mirror Anwendung erfolgen. Hierbei werden eine OAuth 2.0 Client-ID und ein Client-Secret generiert. Beide Informationen werden auf der Smart Mirror Konfigurationsseite unter Fitbit eingetragen.

4.4.3.7 Geoposition

Wird unter dem Konfigurationspunkt „Geodaten“ die lokale Position eingegeben, kann der Smart Mirror die passenden Wetterdaten und Karten anzeigen. Die Geoposition besteht aus Längen- und Breitengrad, welche mit Hilfe von Google Maps ermittelt werden können. Nach Eingabe des Ortes in dem Suchfeld befinden sich in der neu geöffneten URL (siehe Abbildung 4.5) diese beiden Werte nach dem „@“-Zeichen, getrennt durch ein Komma.

Abbildung 4.5: URL von Google Maps mit Geodaten

4.4.3.8 Giphy

Giphy ist ein Portal, auf dem sich zu verschiedenen Themen animierte Bilder befinden. Der Smart Mirror liefert die Möglichkeit, nach bestimmten Kategorien zu suchen. Der Smart Mirror kann aus dieser Kategorie eine zufällig ausgewählte Animation anzeigen. Giphy verfügt über eine API-Schnittstelle. Der API-Schlüssel wird hier nicht für jede Anwendung erstellt, sondern es gibt einen allgemeinen Schlüssel, den jeder nutzen kann. Der Schlüssel ist schon auf der Smart Mirror Konfigurationsseite eingefügt.

4.4.3.9 Begrüßung

Der Spiegel zeigt eine Begrüßung an, die unter „Greeting“ definiert wird. Hier gibt es zwei Optionen. Man definiert entweder allgemeine oder auf den Morgen, den Mittag, den Abend oder die Nacht abgestimmte Begrüßungen. Um allgemeine Begrüßungen zu definieren, muss „Randomly All Day“ ausgewählt werden. Diese Einstellung liefert die Option eine oder mehrere Begrüßungen zu definieren, die zufällig angezeigt werden. Möchte man sich für jede Tageszeit eine andere Begrüßung anzeigen lassen, wählt man „By Time Of Day“ aus. Nun gibt es vier verschiedene Tageszeiten, zu denen man Begrüßungen hinzufügen kann.

4.4.3.10 Home Assistent

Home Assistent ist ein quelloffenes Projekt zur Bedienung von Smart Home Geräten. Die Smart Home Geräte können mit dem Home Assistent verknüpft werden. Der Home Assistent zeigt Informationen zu diesen Geräten an. Des Weiteren ist der Home Assistent in der Lage, Prozesse zu automatisieren. Zum Beispiel kann er automatisch das Raumlicht dimmen, wenn die Bewohner einen Film schauen möchten. Unter der Rubrik Home Assistent auf der Smart Mirror Konfigurationsseite befinden sich drei Eingabefelder. Für die erste Eingabe wird das API-Passwort benötigt. Dies findet sich in der Konfigurationsdatei „configuration.yaml“ des Home Assistent unter dem Punkt

„http“ in der Variable „api_password“. Bei der zweiten Eingabe wird nach dem Intervall gefragt, in welchem die dargestellten Informationen aktualisiert werden. Das Intervall wird in Minuten angegeben. Für das letzte Eingabefeld wird die Adresse des Servers, auf dem der Home Assistent läuft, angegeben. Diese setzt sich aus der IP-Adresse oder auch dem Hostnamen und der Portnummer für den Home Assistent zusammen. Nun kann der Spiegel Informationen der mit dem Home Assistent verknüpften Geräte anzeigen.

4.4.3.11 Philips Hue

Philips Hue ist ein Beleuchtungssystem, das sich über das Netzwerk steuern lässt. Die Leuchtmittel verfügen über diverse Funktionen wie das Dimmen der Leuchtstärke oder das Ändern der Farbe. Diese Funktionen können über eine Handy-Anwendung oder auch den Home Assistent bedient werden. Leuchtmittel können logisch zu Räumen gruppiert werden. Die Beleuchtung dieser Räume kann über den Spiegel angesteuert werden. Für die Nutzung über den Spiegel sind die IP-Adresse des Philips Hue Steuermoduls und der Benutzername erforderlich.

4.4.3.12 Bing Karten

Der Spiegel verfügt über die Funktion, Karten anzuzeigen. Diese Karten bezieht der Smart Mirror über die von Microsoft zur Verfügung gestellten API für Bing Maps. Auf der Seite „https://www.bingmapsportal.com/“ kann ein Konto angelegt und ein API-Schlüssel generiert werden. Auf der Smart Mirror Konfigurationsseite wird dieser Schlüssel bei der Traffic Kategorie eingefügt. In der Traffic Kategorie können außerdem Strecken definiert werden, die man häufig zurücklegt. Dies könnte zum Beispiel die Fahrt zur Arbeitsstelle sein. Eine neue Strecke fügt man hinzu, indem man dieser einen Namen gibt, den Start- und Endpunkt der Strecke eingibt und das Fortbewegungsmittel auswählt. Hier gibt es die Optionen Auto (Driving), öffentliche Verkehrsmittel (Transit) und Laufen (Walking). Außerdem muss ein Zeitraum eingegeben werden, in dem die Strecke auf dem Display angezeigt wird. Der Smart Mirror zeigt dann den Namen der Strecke sowie die aktuell benötigte Zeit an.

4.4.3.13 Nachrichten

Der Spiegel kann Nachrichten von Yahoo anzeigen. Die Serverkommunikation funktioniert über ein RSS-Protokoll. RSS steht in der Version 2.0 für „Really Simple Syndication“. Es wird genutzt, um Schlagzeilen zu veröffentlichen. Um das Anzeigen der Nachrichten zu konfigurieren, müssen die entsprechenden RSS Feeds eingefügt werden. Hierzu findet sich auf der Seite https://developer.yahoo.com/rss// eine Liste von Themenbereichen. Für die interessanten Bereiche können die RSS-Links kopiert und auf der Smart Home Konfigurationsseite unter RSS eingefügt werden.

4.4.3.14 Last.FM

Last.FM ist ein Streaming-Dienst für Musik. Der Smart Mirror kann anzeigen, welcher Musiktitel aktuell von dem jeweiligen Benutzer gehört wird. Auf der Seite https://www.last.fm/api muss man sich mit Konto des Benutzers anmelden und den Smart Mirror als neue Anwendung hinzufügen, um einen API-Schlüssel zu generieren. Dieser Schlüssel wird unter Last.FM auf der Smart Mirror Konfigurationsseite eingefügt.

4.4.3.15 YouTube

Der Smart Mirror kann Videos von YouTube abspielen. Hierzu ist der entsprechende API-Schlüssel notwendig. Der Schlüssel kann bei Google auf der Internetseite https://console.developers.google.com generiert werden. Hierzu muss die Kategorie „APIs & Dienste“ ausgewählt und in der „Bibliothek“ der Dienst „YouTube Data API v3“ aktiviert werden. Auf der geöffneten Seite kann über die Schaltfläche „Anmeldedaten erstellen“ ein API-Schlüssel generiert werden. Hierfür muss als Plattform „Webserver (z. B. Node.js, Tomcat)“ ausgewählt werden, welche so eingestellt wird, dass sie nur auf öffentliche Daten zugreifen soll. Der generierte API-Schlüssel wird auf der Smart Mirror Konfigurationsseite unter YouTube eingefügt.

4.4.3.16 SoundCloud

Der Spiegel kann über die im Display integrierten Lautsprecher Audiodateien von dem Musikdienst SoundCloud abspielen. Hierfür muss die Smart Mirror Anwendung unter

der Seite https://developers.soundcloud.com/docs/api/ eingefügt werden. Der generierte API-Schlüssel kann unter der Kategorie SoundCloud auf der Smart Mirror Konfigurationsseite eingefügt werden.

4.4.3.17 Todoist

Todoist ist ein Web-Dienst, in dem zu erledigende Aufgaben eingegeben und verwaltet werden. Die Aufgaben können mit einem Ablaufdatum und mit einem Wiederholungszyklus versehen werden. Der Spiegel kann die zu erledigenden Aufgaben anzeigen, Einträge löschen und neue hinzufügen. Den dafür notwendigen API-Schlüssel erhält man nach Anmeldung auf der Startseite https://todoist.com/ in der Kategorie „Integrationen“ unter „Einstellungen“. Hier ist der API-Schlüssel zu finden, welcher in die Smart Mirror Konfigurationsseite unter Todoist eingefügt wird.

4.4.3.18 TV-Shows

Unter der Einstellung „TV-Shows“ kann angegeben werden, für welche TV-Serien man über die nächsten Sendetermine informiert werden möchte. Hierzu muss außerdem ein Erneuerungsintervall angegeben werden, wie oft die Information aktualisiert werden soll.

4.4.3.19 Wetter

Der Smart Mirror kann das aktuelle Wetter sowie die Wettervorhersage für die kommenden Tage anzeigen. Hierfür wird der Dienst von Dark Sky genutzt. Dark Sky stellt Wetterinformationen über eine API bereit. Um einen API-Schlüssel zu erhalten, muss man sich auf der Seite https://darksky.net/dev registrieren. Diesen kann man unter „weather“ in der Smart Mirror Konfigurationsseite einfügen. Das Aktualisierungsintervall steht hier schon auf zwei Minuten und „units“ steht auf dem Wert „auto“. Diese Einstellungen kann man beibehalten.

4.5 Erstellen eigener Plug-Ins

Die Spiegel Software ermöglicht das Hinzufügen eigener Plug-Ins. Alle Plug-Ins befinden sich in dem Ordner:

```
plugins/
```

Für die Erstellung eines neuen Plug-Ins wird hier ein weiterer Ordner mit dem Namen der neuen Funktion angelegt. In dem neuen Ordner werden folgende, teilweise optionale Dateien erstellt:

- `config.schema.json`: In dieser JSON-Datei ist hinterlegt, welche Eingaben für das Plug-In benötigt werden.
- `index.html:` Die HTML-Datei definiert das Anzeigemuster für die darzustellenden Informationen.
- `controller.js:` In dieser JavaScript-Datei befindet sich die AngularJS-Controller-Logik.
- `service.js:` Die JavaScript-Datei enthält den AngularJS-Service für die Client-Server-Kommunikation.

Die HTML-Datei „index.html“ nutzt CSS-Klassen für die Darstellung auf dem Display. Für jedes Smart Mirror Layout existiert eine CSS-Datei in folgendem Ordner:

```
app/css
```

Werden für die Darstellung neue Klassen benötigt, müssen diese in die von dem konfigurierten Layout genutzte CSS-Datei eingefügt werden.

Für jede unterstützte Sprache existiert eine eigene Sprachdatei, in der die Sprachbefehle definiert werden. Die Sprachdateien befinden sich in folgendem Ordner:

```
app/locales
```

Neue Sprachbefehle für die Steuerung eines neuen Plug-Ins müssen in die jeweilige Sprachdatei hinzugefügt werden.

Außerdem müssen für jedes neu erstellte Plug-In die drei Dateien „index.html“, „controller.js“ und „service.js“ aus dem Plug-In-Verzeichnis in die Datei „index.html“ im Smart Mirror Verzeichnis eingebunden werden.

In den folgenden Kapiteln wird beschrieben, wie Plug-Ins für RMV, Wikipedia und Plex zu dem Spiegel hinzugefügt werden.

4.5.1 RMV

RMV steht für Rhein-Main-Verkehrsverbund und ist ein Verbund des öffentlichen Nahverkehrs im Rhein-Main-Gebiet. Über eine API-Schnittstelle stellt der RMV Fahrplanauskünfte zur Verfügung. Auf der Seite https://opendata.rmv.de/site/start.html muss man sich registrieren und eine Anwendung mit den Daten des Smart Mirror anlegen. Nach ein paar Tagen erhält man eine E-Mail mit dem API-Schlüssel, der Dokumentation zur API und der URL[32], unter der der Dienst zur Fahrplanauskunft erreichbar ist.

RMV nutzt das HaCon Fahrplan-Auskunfts-System (HAFAS). Diese Software für die Fahrplanauskunft wird von einigen großen Eisenbahngesellschaften und Verkehrsverbunden in Europa eingesetzt.[33]

4.5.1.1 Konfigurationsseite

Jedes dieser Verkehrsunternehmen bietet den Zugang zu seiner Fahrplanauskunft unter einer eigenen URL an, der sogenannten Basis-URL. Um mit dem Plug-In auch andere HAFAS-Anbieter als den RMV nutzen zu können, werden die Basis-URL und der API-Schlüssel nicht in den Dateien des Plug-Ins selbst definiert, sondern es wird jeweils auf einen Konfigurationseintrag verwiesen. Außerdem wird auch die Anzahl der anzuzeigenden Verbindungen als Konfigurationsparameter angelegt. Die Datei „config.schema.json“ sieht wie folgt aus:

```
{
      "schema": {
            "hafas": {
                  "type": "object",
                  "title": "HAFAS Settings",
```

[32] https://www.rmv.de/hapi/
[33] Vgl. (Wikipedia, 2017a).

```
                "properties": {
                "baseurl": {
                      "type": "string",
                      "title": "HAFAS Base URL"
                },
                "key": {
                      "type": "string",
                      "title": "HAFAS API Key"
                },
                "number": {
                      "default":3,
                      "type":"number",
                      "title": "Number of Trips"
                }
                }
          }
},
"form": [
          {
                "type": "fieldset",
                "key": "hafas",
                "expandable": true,
                "order":0
          }
     ]
}
```

Im Schema werden die drei benötigten Angaben aufgelistet und Beschriftung und Eingabetyp definiert. Bei dem Konfigurationspunkt für die Anzahl der angezeigten Verbindungen wird ein Standardwert gesetzt.

4.5.1.2 Darstellung

Die Darstellung der Bahnverbindungen auf dem Display wird in der Datei „index.html“ definiert. Diese HTML-Datei sieht wie folgt aus:

```
<div ng-controller="hafas" class="hafas-container" ng-show="focus == 'hafas'">

     <div ng-repeat="trip in route">
          <ul ng-repeat="leg in trip.LegList.Leg">
```

```
                <li>{{leg.Origin.name}}</li>
                <li ng-show="leg.name">{{leg.name}}</li>
                <li ng-show="!leg.name">{{leg.type}}</li>
                <li>{{leg.Destination.name}}</li>
            </ul>
            </br>
      </div>
</div>
```

Ein DIV-Container bildet den Rahmen der Darstellung und beinhaltet den Controller-Aufruf, die CSS-Klassen und die Anzeigebedingung. Der Controller mit dem Schlüssel „hafas“ wird aufgerufen. Für die Darstellung wird eine neue CSS-Klasse namens „hafas-container“ benötigt. Der Container soll nur angezeigt werden, wenn der Fokus auf „hafas“ gesetzt ist. Der Controller übermittelt die Liste der möglichen Verbindungen in der Scope-Variable „route“. Jeder dieser Verbindungen („trip“) soll angezeigt werden. Hierfür gibt es die Direktive „ng-repeat“, welche für jeden Eintrag einer Liste das genutzte Element wiederholt. Somit wird für jede mögliche Verbindung („trip“) ein DIV-Container dargestellt. Jede dieser möglichen Verbindungen besteht aus mindestens einem genutzten Transportmittel („leg“). Jedes Transportmittel soll zusammen mit der zu fahrenden Strecke aufgeführt werden. Hierfür wird die Direktive „ng-repeat“ für das Element „ul“ genutzt, damit für jeden Teilabschnitt die Startstation, das Transportmittel und die Endstation in einer Auflistung dargestellt werden. Ist das Transportmittel nicht vorhanden, wird der Fortbewegungstyp angezeigt. Das ist der Fall, wenn der Teilabschnitt zu Fuß zurückzulegen ist.

4.5.1.3 Controller

Der Controller beinhaltet die Sprachfunktionen, welche die anzuzeigenden Informationen an die HTML-Seite weitergibt. Dies ist im Folgenden zu sehen:

```
function hafas($scope, $filter, HafasService, SpeechService, Focus) {

  //Show HAFAS
  SpeechService.addCommand('hafas', function (origin, destination) {

    HafasService.generateRoute(origin, destination)
.then(function (response){
```

```
      $scope.route = $filter('limitTo')(response.Trip, config.hafas.number);
    });

    Focus.change("hafas");
  });
}

angular.module('SmartMirror')
    .controller('hafas', hafas);
```

Es wird eine Funktion „hafas“ definiert, welche später als Controller-Funktion für die RMV-Funktionalität übergeben wird. In der Funktion wird als Erstes ein neuer Sprachbefehl zum „SpeechService“ hinzugefügt. Der neue Sprachbefehl setzt sich aus einem Schlüssel und einer Funktion zusammen. Die Funktion bekommt zwei Parameter übergeben, wobei der erste die Startstation und der zweite die Endstation ist. Die Funktion für den Sprachbefehl ruft die Funktion „generateRoute“ aus dem Skript „service.js“ mit den beiden eingegangenen Parametern auf. Der Service liefert eine Liste von Verbindungen, welche erst auf die Anzahl der anzuzeigenden Verbindungen begrenzt wird und dann an die HTML-Seite weitergegeben wird. Die Sprachfunktion beinhaltet außerdem ein Verschieben des Fokus, was die HTML-Seite „index.html“ des Plug-Ins in den Vordergrund rückt. Die letzte Anweisung in der Controller-Datei fügt den RMV-Controller zur Smart Mirror Anwendung hinzu.

4.5.1.4 Service

Für die Client-Server-Kommunikation mit HAFAS wird der „HafasService“ erstellt. Dieser Service besteht aus den Funktionen:

- generateRoute
- getStop
- getRoute

Die Funktion „generateRoute“ erstellt eine Liste von Verbindungen und sieht wie folgt aus:

```
service.generateRoute = async function(origin, destination) {
  var originstop = await service.getStop(origin);
```

```
  var destinationstop = await service.getStop(destination);
  return service.getRoute(originstop.id, destinationstop.id);
}
```

In der Funktion werden die Start- und die Endstation mithilfe der ebenfalls im Service enthaltenen Funktion „getStop“ ermittelt. Für die Funktion „getRoute“ werden die Stationsidentifikationsnummern in der Parameterübergabe benötigt. Die Funktionsaufrufe für die Ermittlung der Start- und Endstation müssen vor Aufruf der Funktion „getRoute“ abgeschlossen sein und werden somit mit einem „await“ versehen. Die Funktion liefert das Ergebnis der Funktion „getRoute“.

Die Funktion „getStop“ sieht wie folgt aus:

```
service.getStop = function(input) {
  var deferred = $q.defer();

  $http.get(config.hafas.baseurl + "location.name?accessId=" + config.hafas.key
                + "&input=" + input + "&format=json&maxNo=1")
  .then(function (response) {
    deferred.resolve(response.data.stopLocationOrCoordLocation[0]
.StopLocation);
  });

  return deferred.promise;
}
```

Die Funktion bekommt einen Parameter übergeben, welcher über einen API-GET an den HAFAS-Dienst weitergegeben wird. Dieser spezielle API-Aufruf bekommt eine Liste der für den übergebenen Parameter infrage kommenden Stationen zurück. Der erste Listeneintrag wird von der Funktion zurückgegeben.

Die letzte Funktion heißt „getRoute“ und sieht wie folgt aus:

```
service.getRoute = function(originid, destinationid) {
  var deferred = $q.defer();

  $http.get(config.hafas.baseurl + "trip?accessId=" + config.hafas.key +
                "&originId=" + originid + "&destId=" + destinationid +
                "&format=json")
  .then(function (response){
    deferred.resolve(response.data);
```

```
  });

  return deferred.promisea;
}
```

Für diese Funktion werden die Identifikationsnummern der Start- und Endstation benötigt. Beide Stationsidentifikationsnummern werden über einen API-GET an den HAFAS-Dienst geschickt. Der HAFAS-Dienst liefert eine Liste der möglichen Verbindungen zwischen den beiden angegebenen Stationen. Diese Liste wird von der Funktion zurückgegeben.

4.5.2 Wikipedia

Die Wikipedia API-Schnittstelle wird genutzt, um Inhalte von Wikipedia auf Anfrage auf dem Spiegel anzuzeigen. Für das Lesen der Artikel wird kein API-Schlüssel benötigt, dieser wird nur benötigt, um Artikel hinzuzufügen oder zu ändern. Für das Wikipedia Plug-In wird daher keine „config.schema.json“ Datei benötigt.

4.5.2.1 Darstellen

Die Darstellungsform der Artikel ist in der Datei „index.html“ definiert. Diese sieht wie folgt aus:

```
<div ng-controller="wikipedia" class="wikipedia-container" ng-show="focus ==
                'wikipedia'">

     <h2>{{header[0]}}</h2>
     </br>
     {{page[0]}}

</div>
```

Der DIV-Container für den Rahmen ruft den Controller auf und übernimmt die Attribute der neu zu erstellenden CSS-Klasse „wikipedia-container“. Sobald der Controller den Fokus in „wikipedia“ ändert, wird der DIV-Container mit den Informationen angezeigt. Der Controller übergibt zwei Variablen an die HTML-Datei. Die Überschrift des Artikels befindet sich in der Variable „header“ und der eigentliche Artikel in „page“.

Um die Überschrift in einer größeren Schrift anzuzeigen, wird das Element „h2“ benutzt.

4.5.2.2 Controller

Der Wikipedia-Controller definiert das Hinzufügen eines Sprachbefehls zum „SpeechService“. Dieser sieht wie folgt aus:

```
SpeechService.addCommand('wikipedia', function (input) {

            WikipediaService.getPage(input).then(function (response){
                  $scope.header = response[1]
                  $scope.page = response[2];
            });

            Focus.change("wikipedia");
      });
```

Der Sprachbefehl besteht aus einem Schlüsselwort und der aufzurufenden Funktion. Diese Funktion erhält die Spracheingabe als Parameter. Die Funktion „getPage“ aus der Datei „service.js“ wird mit dieser Eingabe aufgerufen. Das zurückgegebene Ergebnis wird in Überschrift und Inhalt aufgeteilt und an die HTML-Datei übergeben.

4.5.2.3 Service

Der Wikipedia-Service übernimmt die Kommunikation mit dem Wikipedia-Server und implementiert die Funktion „getPage“, die wie folgt aussieht:

```
service.getPage = function(input) {
  var deferred = $q.defer();

  $http.get("https://en.wikipedia.org/w/api.php?action=opensearch&limit=2
                &format=json&search=" + input)
  .then(function (response) {
    deferred.resolve(response.data);
  });

  return deferred.promise;
}
```

Der GET-Aufruf der API übermittelt den übergebenen Parameter an den Wikipedia-Server. Das zurückgeschickte JSON-Objekt wird von der Funktion zurückgegeben.

4.5.3 Plex

Plex ist ein Medienserver, der Filme, Serien und Musik verwaltet. Der Plex-Server ist auf einem NAS-System installiert und soll genutzt werden, um Musik auf dem Spiegel über das Netzwerk abzuspielen. Plex verfügt über eine API-Schnittstelle, die zur Wiedergabe von Musik genutzt werden kann.

4.5.3.1 Konfigurationsseite

Um API-Anfragen an den Server zu senden, wird die URL des Plex-Servers benötigt. Diese setzt sich zusammen aus der IP-Adresse oder dem Hostnamen des Plex-Servers und dem vom Plex-Dienst genutzten Port. Für die Authentifizierung wird der Plex-Token eines registrierten Plex-Benutzers benötigt. Für diese drei Angaben werden Konfigurationseinträge erstellt, auf die in den Plug-In-Dateien verwiesen wird. Das Schema für Plex in der Datei „config.schema.json“ sieht wie folgt aus:

```
"plex": {
      "type": "object",
      "title": "PLEX Settings",
      "properties": {
                  "host": {
                  "type": "string",
                  "title": "IP or Host of PLEX-Server"
                  },
                  "port": {
                  "type": "string",
                  "title": "Port of PLEX-Server"
                  },
                  "token": {
                  "type": "string",
                  "title": "X-Plex-Token"
                  }
      }
}
```

Die Portnummer des Plex-Dienstes findet man, wenn man über die Web-Oberfläche auf den Plex-Dienst zugreift. Standardmäßig wird die Portnummer 32400 vom Plex-Dienst genutzt.

Um den Plex-Token zu ermitteln, müssen die Informationen eines Titels, eines Albums, eines Films oder einer TV-Serie geöffnet werden. Klickt man auf die in Abbildung 4.6 gezeigte Schaltfläche „XML-Datei anzeigen“, öffnet sich eine neue Seite, in welcher die Medieninformationen im XML-Format dargestellt werden.

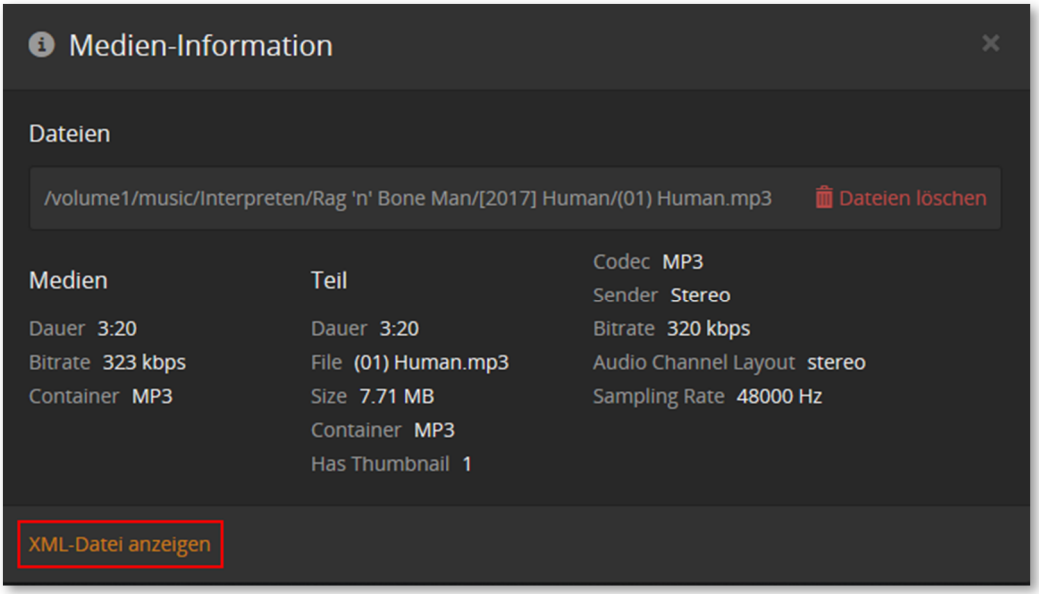

Abbildung 4.6: Medien-Information eines Liedes in Plex

In der URL dieser Seite wird, wie in Abbildung 4.7 gezeigt, der Plex-Token als Parameter übergeben.

Abbildung 4.7: Plex-Liedinformationen in XML

4.5.3.2 Darstellung

In der Datei „index.html“ befindet sich die Darstellungsform des Plex-Plug-Ins:

```
<div ng-controller="plex" class="plex-container" ng-show="focus == 'plex'">
      <table>
            <tr>
                  <th>
                  <img class="plex-image" ng-src="{{thumb}}" />
                  </th>
                  <th>
                  Title: {{title}} </br>
                  Album: {{album}} </br>
                  Interpret: {{interpret}}
                  </th>
            </tr>
      </table>
      <iframe ng-if="plextrack" class="animate-grow" ng-src="{{plextrack}}"
                  /></iframe>
</div>
```

Der DIV-Container für den Rahmen ruft den Plex-Controller auf und wird angezeigt, wenn der Fokus auf „plex“ liegt. Die CSS-Klasse „plex-container“ wird für dieses Plug-In neu erstellt. Die Informationen des wiedergegebenen Musiktitels werden in einer Tabelle dargestellt. Die Tabelle besteht aus einer Zeile und zwei Spalten, wobei in der ersten Spalte das Album-Cover und in der zweiten Spalte Informationen zum Musiktitel dargestellt werden. Diese Informationen setzen sich aus Titel, Album und Interpret zusammen. Nach der Tabellendefinition wird ein „iframe“ zum Einbetten des Musik-Streams genutzt. Der Stream wird nur dargestellt, wenn die Bedingung der „ng-if“-Direktive erfüllt ist. Dies wird später zum Beenden des Streams genutzt.

4.5.3.3 Controller

Die Controller-Funktion fügt dem Sprachdienst drei neue Sprachbefehle hinzu und definiert die notwendigen Funktionen.

Der erste Sprachbefehl soll ein Musikstück unter Angabe des Titels abspielen und sieht wie folgt aus:

```
SpeechService.addCommand('plex', function (input) {
  PLEXService.getTrack(input).then(function (response){

    setMusic(response.MediaContainer.Metadata[0]);
  });
  Focus.change("plex");
});
```

Die eingefügte Sprachbefehls-Funktion ruft die Funktion „getTrack“ des Service auf, der eine Liste der passenden Musiktitel zurückliefert und übergibt den ersten Listeneintrag an die Funktion „setMusic“. Der Fokus wird in „plex“ geändert, um die Liedinformationen anzuzeigen.

Die Funktion „setMusic“ wird für die Erstellung der Stream-URL und die Übergabe der Informationen an die HTML-Datei genutzt, um die Wiedergabe der Musik zu starten und sieht wie folgt aus:

```
var setMusic = function(metadata) {

      var plexurl = "http://" + config.plex.host + ":" + config.plex.port +
                  metadata.Media[0].Part[0].key + "?X-Plex-Token=" +
                  config.plex.token;
      $scope.plextrack = $sce.trustAsResourceUrl(plexurl);

      $scope.thumb = "http://" + config.plex.host + ":" + config.plex.port +
                  metadata.thumb + "?X-Plex-Token=" + config.plex.token;

      $scope.title = metadata.title;
      $scope.album = metadata.parentTitle;
      $scope.interpret = metadata.grandparentTitle;
      if(metadata.originalTitle)
      {
                  $scope.interpret = metadata.originalTitle;
      }

}
```

Für den Musik-Stream wird eine URL erstellt, unter der die Musik wiedergegeben werden kann. Diese URL wird als vertrauenswürdige URL in der Variablen „plextrack“ an die HTML-Datei weitergegeben. Es wird eine weitere URL für die Darstellung des

Album-Covers erstellt. An die HTML-Seite werden außerdem Titel, Interpret und Album weitergegeben.

Beim zweiten Sprachbefehl werden der Titel und der Interpret übergeben, um das gewünschte Lied wiederzugeben. Das Hinzufügen des Sprachbefehls sieht wie folgt aus:

```
SpeechService.addCommand('plex_interpret', function (track, interpret) {
      PLEXService.getTrack(track).then(function (response){

            var media = response.MediaContainer.Metadata[0];

            for(var i = 0; i < response.MediaContainer.size; i++)
            {
                  if(response.MediaContainer.Metadata[i].grandparentTitle.
                  toLowerCase() == interpret.toLowerCase())
                  {
                  media = response.MediaContainer.Metadata[i];
                  break;
                  }
                  if(response.MediaContainer.Metadata[i].originalTitle.
                  toLowerCase() == interpret.toLowerCase())
                  {
                  media = response.MediaContainer.Metadata[i];
                  break;
                  }
            }

            setMusic(media);
      });
      Focus.change("plex");
});
```

Der Plex-Dienst unterstützt nicht die Suche nach Liedern unter gleichzeitiger Angabe von Interpret und Titel. Die kombinierte Suche muss selbst implementiert werden. Hierfür wird im Plug-In erst eine Titelsuche durchgeführt und dann nach dem Interpreten gefiltert. Es wird, wie auch beim vorherigen Sprachbefehl, zuerst einmal die Funktion „getTrack“ des Plex-Dienstes aufgerufen, um eine Liste aller in Frage kommenden Lieder zu erhalten. Diese Liste wird mit dem eingegebenen Interpreten verglichen. Bei der ersten Übereinstimmung werden die Informationen zu dem Lied an

die Funktion „setMusic“ übergeben. Der Fokus wird auf die HTML-Seite des Plug-Ins verschoben.

Der dritte Sprachbefehl wird zum Beenden der Musikwiedergabe hinzugefügt:

```
SpeechService.addCommand('plex_stop', function () {
      Focus.change("default");
      stopMusic();
});
```

Soll die Musikwiedergabe beendet werden, wird der Fokus wieder auf „default“ gesetzt, woraufhin die Startseite des Spiegels angezeigt wird. Die Funktion „stopMusic“ des Controllers wird aufgerufen, um die Musikwiedergabe zu beenden.

Die Wiedergabe der Musik soll außerdem bei einem Wechsel der angezeigten Informationen beendet werden. Folgende Funktion übernimmt diese Aufgabe:

```
$rootScope.$on('focus', function (targetScope, newFocus, oldFocus) {
      if(oldFocus == "plex" && newFocus != "plex"){
            stopMusic();
      }
})
```

Sobald während der Musikwiedergabe der Fokus geändert wird, wird die Funktion „stopMusic“ aufgerufen.

Die Funktion „stopMusic“ beendet die Wiedergabe und sieht wie folgt aus:

```
var stopMusic = function() {
      $scope.plextrack = "";
}
```

Hier wird die Stream-URL gelöscht, die zuvor von der Funktion „setMusic“ an die HTML-Datei übergeben wurde. Da in der HTML-Datei definiert ist, dass der Stream nur dargestellt wird, wenn eine Stream-URL vorhanden ist, wird die Wiedergabe beendet.

4.5.3.4 Service

Der Service beinhaltet eine Funktion namens „getTrack“, welche eine Liste von Liedern passend zu den Suchkriterien liefert und wie folgt implementiert ist:

```
service.getTrack = function(input) {
  var deferred = $q.defer();

  $http.get("http://" + config.plex.host + ":" + config.plex.port +
                "/search?type=10&X-Plex-Token=" + config.plex.token +
                "&query=" + input, {'headers' : {'Accept':
                'application/json'}})
  .then(function (response) {

    deferred.resolve(response.data);
  });

  return deferred.promise;
}
```

Die Funktion nimmt die als Parameter übergebene Eingabe und schickt sie an den Plex-Server. Hier muss außerdem ein Header übergeben werden, um ein JSON Objekt als Antwort zu erhalten. Die Liste der passenden Musikstücke wird von der Funktion zurückgegeben.

4.5.4 CSS

In der CSS-Datei wird definiert, wie die in der HTML-Datei dargestellten Elemente angezeigt werden. Die CSS-Dateien befinden sich unter „css“ im Ordner „app“ des Smart Mirror Verzeichnisses. Standardmäßig ist die Datei „main.css“ in die Smart Mirror Anwendung integriert. Wird ein anderes Layout und damit eine andere CSS-Datei für den Smart Mirror benutzt, muss diese geändert werden. Folgende Zeilen müssen in die CSS-Datei eingefügt werden:

```
.hafas-container, .wikipedia-container {
      width: 350px;
      text-align: left;
      margin-left: auto;
      margin-right: auto;
      font-weight:bold
}

.plex-container{
      width: 350px;
```

```
      text-align: left;
      margin-left: auto;
      margin-right: auto;
}

.plex-image{
      width: 100px;
}
```

In der CSS-Datei werden Klassen definiert, die in den HTML-Dateien genutzt werden. Alle drei Container benötigen Einstellungen für die Breite, die Textausrichtung und den äußeren Rahmen. Beim Wikipedia- und RMV-Container wird die Schrift fett dargestellt, damit die Informationen besser lesbar sind. Damit das Album-Cover, welches beim Plug-In Plex angezeigt wird, eine feste Größe besitzt, wird unter „.plex-image“ die Breite des darzustellenden Cover-Bilds festgelegt.

4.5.5 Definition des Sprachbefehl

In dem Verzeichnis „app“ befindet sich der Unterordner „locales“. In diesem Ordner befindet sich zu jeder unterstützten Sprache eine JSON-Datei, in der alle Sprachbefehle für die jeweilige Sprache definiert sind. Für die deutschen Sprachbefehle ist die Datei „de.json“ verantwortlich. In diese Datei müssen folgende Objekte unter „commands“ hinzugefügt werden:

```
"hafas": {
      "text": "Von ________ nach ________",
      "voice": "von *origin nach *destination",
      "description": "Sucht eine Verbindung zwischen zwei Haltestellen"
},
"wikipedia": {
      "text": "Was ist ________",
      "voice": "Was ist (ein) *input",
      "description": "Sucht eine Beschreibung zum befragten Thema raus"
},
"plex": {
      "text": "Spiele das Lied________",
      "voice": "Spiele das Lied *input",
      "description": "Spielt den gewünschten Track"
```

```
 },
"plex_interpret": {
      "text": "Spiele ________ von ________",
      "voice": "Spiele *track von *interpret",
      "description": "Spielt den gewünschten Track des Interpreten"
},
"plex_stop": {
      "text": "Stopp Musik",
      "voice": "Stopp (die) Musik",
      "description": "Stopt die Musik Wiedergabe."
}
```

Ein Sprachbefehl wird unter einem Objekt eingefügt, welches die gleiche Benennung wie der im Controller an den „SpeechService“ übergebene Befehlsschlüssel haben muss.

Ein Sprachbefehl besitzt drei Attribute, von denen die Attribute „text“ und „description“ für die Anzeige der Kommandoliste genutzt werden. In dem Attribut „voice“ wird der zu sprechende Text definiert, für den die Sprachfunktion aufgerufen werden soll. Wörter in Klammern sind optional und die mit einem „*“ markierten Wörter werden als Parameter interpretiert.

Die Sprachbefehle sind nun in der Kommandoliste aufgeführt und können vom Spiegel erkannt werden.

4.5.6 Einfügen der Plug-In-Komponenten

Die Dateipfade der neuen HTML-Dateien werden in die Datei „index.html“ im Smart Mirror Verzeichnis eingefügt. Hier wird festgelegt, wo die Informationen auf dem Display angezeigt werden. Da alle drei eingefügten Plug-Ins in der Mitte des Spiegels angezeigt werden sollen, müssen die Einträge in den DIV-Container mit der CSS-Klasse „middle-center“ eingefügt werden. Dies erreicht man durch folgende Zeilen:

```
<div ng-include="'plugins/hafas/index.html'"></div>
<div ng-include="'plugins/wikipedia/index.html'"></div>
<div ng-include="'plugins/plex/index.html'"></div>
```

Die Referenzen zu den neuen Controller-Dateien werden zu den bereits existierenden hinzugefügt. Diese Stelle ist mit dem Kommentar „<!-- Controllers -->“ markiert. Das Einfügen der Controller-Dateien sieht wie folgt aus:

```
<script src="plugins/hafas/controller.js"></script>
<script src="plugins/wikipedia/controller.js"></script>
<script src="plugins/plex/controller.js"></script>
```

Ebenso wird mit den neuen Service-Dateien verfahren. Diese befinden sich an der Stelle, die mit dem Kommentar „<!-- Services -->“ markiert ist. Die Service-Dateien fügt man mit folgenden Zeilen ein:

```
<script src="plugins/hafas/service.js"></script>
<script src="plugins/wikipedia/service.js"></script>
<script src="plugins/plex/service.js"></script>
```

Die eingefügten Plug-Ins werden nun beim erneuten Starten der Smart Mirror Anwendung geladen und können dann genutzt werden.

5 Fazit

Der Smart Mirror ist implementiert und einsatzbereit. Wie in Abbildung 5.1 zu sehen, scheinen die angezeigten Informationen durch den halbtransparenten Spiegel hindurch und können vom Nutzer betrachtet werden. Der Spiegel zeigt standardmäßig die Begrüßung, die Uhrzeit und Wetterinformationen an. Aktuelle Nachrichten aus aller Welt werden unten mittig im Spiegel eingeblendet.

Abbildung 5.1: Ansicht des Spiegels

Es gibt Schnittstellen zu smarten Geräten wie zu dem Beleuchtungssystem Philips Hue, mit welchem Lichter im Haus gesteuert werden können. Auch kann der Spiegel Informationen zu Smart Home Geräten anzeigen, welche in den Home Assistent integriert wurden. Für weitere Smart Home Geräte können neue Schnittstellen für die Bedienung erstellt werden. Der Smart Mirror kann somit genutzt werden, um das Smart Home zu bedienen.

An drei Beispielen wurde gezeigt, wie eigene Plug-Ins für den Spiegel entwickelt werden können und wie diese in die Smart Mirror Anwendung integriert werden. Über die RMV-API können Fahrplanauskünfte des öffentlichen Nahverkehrs angezeigt werden. Durch die Wikipedia-API ist es möglich, Erläuterungen zu den angefragten Begriffen zu erhalten. Der Smart Mirror ist außerdem in der Lage, Musik von einem Plex-Server abzuspielen.

Beim Suchalgorithmus von Plex ergibt sich allerdings das Problem, dass bei Angabe des Titels in Kombination mit einem Interpreten kein passender Musiktitel gefunden wird. Die im Plug-In programmierte Suchfunktion stellt keine optimale Lösung dar, da sie nur unterschiedliche Groß- und Kleinschreibung toleriert und ansonsten auf exakte Gleichheit prüft. Sobald nur ein Zeichen differiert, wird der gewünschte Titel nicht gefunden. Die Abfrage ist nicht ausreichend fehlertolerant. Hier sollte die Fehlertoleranz der Suchfunktion verbessert werden.

5.1 Weitere Einsatzmöglichkeiten

Weitere Einsatzmöglichkeiten des Smart Mirror sind vorstellbar. Der smarte Spiegel könnte sich über Bluetooth mit anderen Gegenständen wie die Personenwaage oder die elektrische Zahnbürste verbinden. Der Spiegel könnte dann das aktuelle Gewicht und den Gewichtsverlauf anzeigen oder während des Zähneputzens Pflegetipps geben. Da diese Information nur für die jeweilige Person relevant ist, könnten Profile angelegt werden. Eine Gesichtserkennungssoftware aktiviert dann immer das passende Profil.

Nützlich wäre auch eine Anwendung zur Kommunikation mit Freunden und Bekannten. Hierzu könnte WhatsApp oder Skype integriert werden, um Nachrichten zu senden oder Videogespräche zu führen.

Über Amazon Prime Video, Netflix oder Plex könnten Filme oder Serien abgespielt werden. Über Plex ist es auch möglich, die Videowiedergabe auf anderen Abspielgeräten zu steuern. Somit könnte zum Beispiel die Videowiedergabe in einem anderen Raum von dem Spiegel aus gestartet oder gestoppt werden.

Auch könnten WLAN-Lautsprecher über eine API angesprochen werden. Somit könnte ein neuer Sprachbefehl hinzugefügt werden, der zum Beispiel Musik im Wohnzimmer abspielen würde.

Zurzeit kann jede Person, die Zugriff auf das WLAN hat, die Konfiguration des Spiegels ändern. Deshalb wäre es sinnvoll, eine Authentifizierung in die Konfigurationsseite der Smart Mirror Anwendung einzufügen. Die Einstellungen des Spiegels wären dann vor unbefugtem Zugriff geschützt.

5.2 Ausblick

Das Thema Smart Home gewinnt immer mehr an Bedeutung. Damit wird der Bedarf an einfachen Bediengeräten wachsen, was zu einer steigenden Nachfrage nach smarten Spiegeln führen wird.

Ein Smart Mirror wird sich zukünftig durch immer bessere Benutzerfreundlichkeit auszeichnen. Der Einsatz von Künstlicher Intelligenz (KI) wird zum Beispiel ermöglichen, die Stimmung des Benutzers aus seiner Stimmlage zu erkennen. Ist der Nutzer etwa gut gelaunt und möchte Musik hören, könnte die KI dies erkennen und das passende Genre abspielen.

Ein smarter Spiegel könnte neue Anwendungen zur Verfügung stellen, welche den Nutzer über sein Äußeres beraten. Hierfür könnten Kleidungsstücke, Make-Up, Accessoires und Frisuren in das Spiegelbild des Nutzers eingeblendet und an die Bewegungen des Nutzers angepasst werden. Man könnte zwischen verschiedenen Outfits wechseln und diese bei Gefallen direkt über den Spiegel bei einem Online-Anbieter bestellen.

Zukünftige Technologien werden die Art und Weise, wie ein Benutzer mit einem zentralen Bediengerät für das smarte Heim interagiert, deutlich ändern. Neben der Spracheingabe wird die Gestenerkennung immer größere Bedeutung gewinnen.

Vorstellbar ist auch, dass die Darstellung der Informationen als dreidimensionales Hologramm in den Raum projiziert wird. So gesehen ist die heutige Form des smarten Spiegels nur eine Zwischenlösung auf dem Weg zur zukünftigen interaktiven Bedienung eines smarten Heims.

Literaturverzeichnis

Ackermann, P. (2017). *Desktopanwendungen mit JavaScript entwickeln*. Abgerufen am 5. November 2017 von https://www.heise.de/developer/artikel/Desktopanwendungen-mit-JavaScript-entwickeln-3609943.html?seite=3

Berkemeyer, K. (2016). *Raspberry Pi: Die Top-Alternativen im Check*. Abgerufen am 3. November 2017 von http://www.chip.de/news/Raspberry-Pi-Das-sind-die-Top-Alternativen_98494061.html

Bitkom. (2014). *Eine Million Smart Homes bis 2020*. Abgerufen am 3. Oktober 2017 von https://www.bitkom.org/Presse/Presseinformation/Eine-Million-Smart-Homes-bis-2020.html

Flaßkamp, M. (2016). *Welcher Raspberry Pi? Alle Modelle im Vergleich*. Abgerufen am 2. November 2017 von http://praxistipps.chip.de/welcher-raspberry-pi-alle-modelle-im-vergleich_41923

GranCr. (2017). *Smart Mirror selbst bauen*. Abgerufen am 24. Oktober 2017 von https://glancr.de/smart-mirror-selbst-bauen/

Hagan, P. (2012). *Looking in the mirror DOES make you more anxious about your looks*. Abgerufen am 11. Oktober 2017 von http://www.dailymail.co.uk/femail/article-2097717/Looking-mirror-DOES-make-anxious-looks.html

Heidrich, J., & Mäkeler, N. (2017). Alexa, darfst du das? *c't 2017, Heft 22*, S. 86-87.

Mues-Tec. (2017). *Smart Spiegel*. Abgerufen am 24. Oktober 2017 von https://www.mues-tec.de/epages/79550038.sf/de_DE/?ObjectPath=/Shops/79550038/Categories/%22Smart%20Spiegel%22

Open Source Initiative. (2017). *The MIT License*. Abgerufen am 12. November 2017 von https://opensource.org/licenses/MIT

Pavia, A., Stimac, S., & Richards, M. (2016). *Building an IoT Magic Mirror with Hosted Web Apps and Windows 10*. Abgerufen am 24. Oktober 2017 von https://blogs.windows.com/msedgedev/2016/05/31/magic-mirror-hosted-web-app/#HuVmsUkgtUZltXgf.97

Perez, S. (2017). *Amazon to control 70 percent of the voice-controlled speaker market this year*. Abgerufen am 4. Oktober 2017 von https://techcrunch.com/2017/05/08/amazon-to-control-70-percent-of-the-voice-controlled-speaker-market-this-year/

Radke, J. (2013). Raspberry Pi - Antworten auf die häufigsten Fragen. *c't 2013, Heft 10*, S. 140.

raspberry.tips. (2015). *Die richtige SD-Karte für den Raspberry Pi*. Abgerufen am 6. November 2017 von https://raspberry.tips/raspberrypi-einsteiger/die-richtige-sd-karte-fuer-den-raspberry-pi/

Samsung. (2017). *ML55E*. Abgerufen am 24. Oktober 2017 von http://displaysolutions.samsung.com/digital-signage/detail/848/ML55E

Schiller, K. (2017). *Smart Home für Einsteiger*. Abgerufen am 3. Oktober 2017 von https://www.homeandsmart.de/was-ist-ein-smart-home

Schmidt, F. (2017). *Raspberry Pi im Wandel: Alle Modelle, Unterschiede und Kauftipps*. Abgerufen am 2. November 2017 von http://www.computerbild.de/artikel/cb-Tipps-PC-Hardware-Raspberry-Pi-Modelle-Vergleich-17632421.html

smartmirror.io. (2017). *Hardware*. Abgerufen am 6. November 2017 von https://docs.smart-mirror.io/docs/hardware.html

Spiegelshop24. (2017). *Smart Mirror zeigt aktuelle News und das Wetter an!* Abgerufen am 24. Oktober 2017 von https://spiegelshop24.com/info/smart-mirror.html

Srocke, D., & Karlstetter, F. (2017). *Was ist eine REST API?* Abgerufen am 5. November 2017 von https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/

Türck, A. (2017). *Wie Sprachassistenten die Werbung verändern*. Abgerufen am 5. Oktober 2017 von Pilot: https://www.pilot.de/neuigkeiten/wie-sprachassistenten-die-werbung-veraendern/

Wikipedia. (2017). *AngularJS*. Abgerufen am 25. November 2017 von https://de.wikipedia.org/wiki/AngularJS

Wikipedia. (2017a). *HAFAS - Wikipedia*. Abgerufen am 10. November 2017 von https://de.wikipedia.org/wiki/HAFAS

Wolski, D. (2017). *Raspbian: Das kann das OS für den Raspberry Pi*. Abgerufen am 7. November 2017 von https://www.pcwelt.de/a/raspbian-das-kann-das-os-fuer-den-raspberry-pi,3448329

Wyzowl. (2017). *The Power of Visual Communication Infographic*. Abgerufen am 11. Oktober 2017 von http://blog.wyzowl.com/power-visual-communication-infographic/

Zeigermann, O. (2015). *Single Page-Anwendungen – Vorteile von Web und Desktop kombiniert*. Abgerufen am 25. November 2017 von https://www.embarc.de/single-page-anwendungen/