Magic Mirror

Florian Vogel

11. März 2019

Inhaltsverzeichnis

Glossar

Bash	Befehls Sprache im UNIX Umfeld.
DMIPS	Dhyrostone mega instructions pro second, Angabe der Instruktionen pro Sekunde eines Prozessorkerns.
DVI	Digital visual interface, auf deutsch Digitale visuelle Schnittstelle zum übertragen von Videodaten.
HDMI	High definition multimedia interface, auf deutsch hochauflösende Multimedia-Schnittstelle.
HTTP	Hypertext transfer protocol, zustandsloses Protokoll zur Übertragung von Daten.
JSON	JavaScript Object Notation, kompaktes Datenformat in einer einfachen lesbaren Textform.
JTAG	Joint Test Action Group, Standard zum Testen und Debuggen von integrierten Schaltungen.
LCD	Liquid crystal display, auf deutsch Flüssigkristallanzeige.
MQTT	Message Queuing Telemetry Transport, ein offenes Nachrichtenprotokoll um zwischen Maschinen Nachrichten auszutauschen.
Node-Red	Tool um verschieden Hardware zusammen zu verknüpfen.
OpenOCD	Open on-chip debugger, wird verwendet über Hardwareadapter für beispielsweise Debugging von Controllern.
Opensource	Software die von Dritten eingesehen, gelesen und bearbeitet werden kann.
pm2	Es handelt sich um einen node.js Prozess Manager.

QOS	Levels für MQTT Übertragung: 0 → nur einmal, 1 → mindestens einmal, 2 → genau einmal.
Rest API	Representional state transfer für Application programing interface.
SD Karte	Secure Digital Memory card, auf deutsch sichere digitale Speicherkarte.
SSID	Service set identifier, Name eines WLAN Netzwerks.
URL	Uniform resource locator, Identifikation und Lokalisation einer bestimmten Ressource.
USB	Universal serial bus, häufig benutzter serieller Bus für Datenübertragung, beispielsweise von Speicherchips.
VGA	Video graphics array, Schnittstelle für Bildübertragung.
WLAN	Wireless local area network, auf deutsch drahtloses lokales Netzwerk.

1 Einführung

Ein Magic Mirror ist ein optisch ansprechendes Anzeigegerät. Es handelt sich um ein Spiegel mit integriertem Bildschirm, wobei es sich bei dem Spiegel um einen sogenannten Spionspiegel handelt. Er ist von einer Seite möglichst reflektierend und von der anderen Seite möglichst durchlässig. Mit dem verbauten Bildschirm ergeben sich beinahe unbegrenzte Möglichkeiten um Informationen zu präsentieren und diese ansprechend darzustellen. Dadurch passt ein Magic Mirror mit passendem Design gut in einen Wohnbereich.
Nun, was soll denn auf solch einem Spiegel angezeigt werden? Natürlich gibt es einige Klassiker, wie zum Beispiel die aktuelle Zeit. Die Möglichkeiten lassen aber viel mehr zu. Es ist beispielsweise auch denkbar jeweils den nächsten Zug von Bern nach Zürich auf dem Magic Mirror anzuzeigen.
Die Interessen für Informationen werden sich mit Sicherheit ändern über die Zeit. Das bedingt eine Konfigurationsmöglichkeit für den Benutzer des Spiegels, mit welcher er anzeigende Informationen ändern kann. Diese genannte Modularität zu erreichen ist eines der Ziele in dieser Semesterarbeit. Weiter soll am Ende ein funktionierender Prototyp eines Magic Mirrors vollendet sein, welcher im Heimbereich eingesetzt wird.

2 Zielsetzung

Das erste und am höchsten gewichtete Ziel ist das Erstellen eines fertigen Prototypes. Dies wird in folgende Teilschritte unterteilt.

- Einfache Anzeigeelemente auf dem Spiegel, wie beispielsweise die Uhrzeit, das Wetter oder einen Kalender.
- Energiesparmodus, dabei wird der Bildschirm über einen externen Infrarotsensor ein- und ausgeschalten.
- Modul auf welches über mobile Applikation zugegriffen werden kann.
- Eine Android Applikation um auf obengenanntes Modul zugreifen zu können.
- Sprachsteuerung welche vom Google Assistant gemacht wird.

Die genannte Punkte werden schrittweise umgesetzt. Somit hat der letzte Punkt die niedrigste Priorität und wird als optional erachtet.

3 Ausgangslage

Die Idee des Magic Mirror ist bekannt. Es gibt bereits zahlreiche Versionen davon online zum Nachbau. Es gibt Vorschläge für das Spiegelglas, den Bildschirm, die Recheneinheit, sogar wie der Rahmen des Spiegels aufgebaut werden kann. Deshalb ist es grundsätzlich einmal notwendig, in diesem Dschungel von Ideen eine gute Zusammensetzung zu finden.
Sehr prominent tritt dabei ein Opensource Projekt auf, welches sich MagicMirror^2 nennt [2]. Es ist das einzig verfügbare Projekt, welches modular aufgebaut ist und zudem von einer grossen Community unterstützt wird. Deshalb muss als erstes der Einsatz dieses Projekts abgewogen werden.

3.1 Bewertungskriterien Opensource Projekt

Um zu evaluieren, ob mit dem Opensource ProjMein Standortekt MagicMirror^2 weitergearbeitet werden kann oder ein komplett eigenständiger Ansatz gewählt werden muss, werden folgende Bewertungskriterien genauer analysiert:

- Modularität und Erweiterbarkeit
- Einarbeitungszeit
- Hardwareeinsatz

Die folgenden drei Unterkapitel behandeln kurz diese Punkte jeweils für das Opensource Projekt, sowie die komplette Eigenentwicklung. Dabei werden Bewertungen angegeben zwischen 1-10, wobei 10 sehr gut ist.

3.1.1 Modularität und Erweiterbarkeit

Die Startseite des Opensource Projekts wirbt bereits mit grosser Modularität. Wie gross diese ist, zeigt sich durch die Liste der bereits implementierten Erweiterungen, welche auch auf dem Magic Mirror angezeigt werden können. Bereits mehrere hundert solcher Zusatzmodule können eingebunden werden [4].
Die Erweiterbarkeit ist ebenfalls gut, da die Einbindung der einzelnen Module über eine zentrale Stelle (eine Konfigurationsdatei) gemacht werden kann.
Bewertung: 8 → keine Informationen darüber, wie gut Module getestet wurden

Bei einer Eigenentwicklung muss von Begin klar die Modularität und Erweiterbarkeit berücksichtigt werden. Dies ist gut machbar, da es beim Designen noch keine weiteren Abhängigkeiten gibt. Jedoch ist dies mit Zusatzaufwand verbunden.
Bewertung: 5 → die Modularität zu gewährleisten bedeutet aber ein grosser Mehraufwand

3.1.2 Einarbeitungszeit

Das Opensource Projekt rühmt sich damit, auch für Leute ohne nennenswerten technischen Hintergrund einsetzbar zu sein. Diese Aussage bestätigt sich, denn innerhalb einer Stunde ist es möglich, das Projekt auf einem vorbereiteten Raspberry Pi (Raspian auf SD Karte) lauffähig zu haben und mit der Grundkonfiguration zu betreiben. Deshalb wird die Einarbeitungszeit als klein eingeschätzt.
Bewertung: 8 → da Einarbeitung in Modulbearbeitung noch dazukommt

Bei der Eigenentwicklung verhält sich die Einarbeitungszeit unterschiedlich je nach verwendeter Methode. Was jedoch sicherlich stark ins Gewicht fällt ist die Konzeptfindung, welche vom Vorwissen abhängt. Dadurch kann die Einarbeitungszeit minimiert werden, jedoch steigt damit sicherlich die Zeit der Konzeptphase.
Bewertung: 6 → Konzeptphase kann sehr lange dauern

3.1.3 Hardwareeinsatz

Die verwendete Hardware des Opensource Projekts ist nicht definiert. Das Projekt wurde adaptiert um auf Smartphones und Tablets zu laufen, ist aber auch auf einem Rapberry Pi 2 oder Raspberry Pi 3 verwendbar. Diese Portabilität ist optimal, da in dieser Semesterarbeit eine low-cost Hardwareplatform verwendet werden soll.
Bewertung: 8 → da es auf den meisten gängigen Platformen läuft, aber nicht auf allen (z.b. Probleme beim Raspberry Pi 1)

Bei einer Eigenentwicklung ist der Einsatz der Hardware frei und kann eingeplant werden. Eine low-cost Hardwareplatform zu verwenden stellt kein Problem dar.
Bewertung: 10 → Projekt kann auf Hardware angepasst werden

3.2 Designentscheid

Anhand der Bewertung aus den vorangehenden Kapiteln fällt der Entscheid klar zu Gunsten vom Opensource Projekt aus. Das bedeutet, dass als Platform für die Software das MagicMirror^2 Projekt verwendet wird. Von dieser Grundlage aus, werden die nötigen Anpassungen und Erweiterungen gemacht in den einzelnen Modulen.

4 Aufbau

Unter einem Kapitel mit dem Namen Aufbau können mehrere Erwartungshaltungen zutreffen. Zum einen sicherlich der Aufbau des Magic Mirrors aus Sicht der verwendeten Materialien und Baugruppen. Zum anderen, und das ist für diese Semesterarbeit das Relevante, der Aufbau der Software und deren Zusammenspiel mit der Hardware. Um jedoch das erstgenannte nicht komplett aussenvor zu lassen, hier ein Sinnbild wie der Magic Mirror aufgebaut wird. Darauf zu sehen ist der benötigte Bildschirm, das Spionglas und ein Rahmen.

Abbildung 1: Bauelemente für den Magic Mirror [2]

4.1 Auswahl Hardware

Die Auswahl der Hardware beinhaltet mehrere Einzelteile. Die folgenden Unterkapitel gehen auf die verschiedenen Teile ein. Zur Veranschaulichung jedoch vorweg eine Übersicht über die Komponenten.

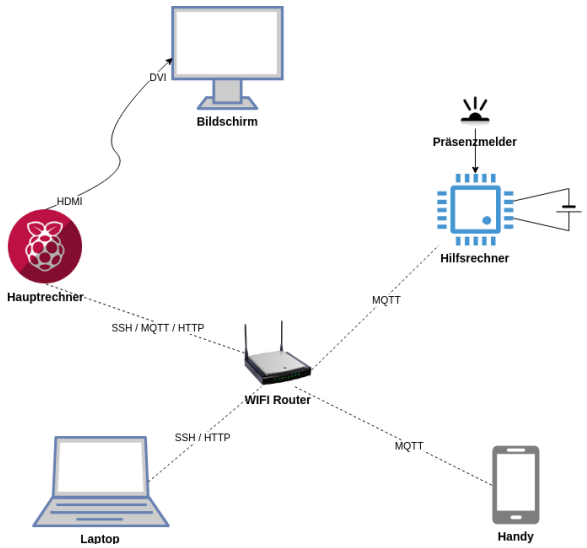

Abbildung 2: Hardware Komponentenübersicht

4.1.1 Bildschirm

Der Bildschirm ist für den Magic Mirror natürlich zentral. Jedoch ist bei der Auswahl nicht besonders viel zu beachten. Bei dieser Semesterarbeit wird ein 24 Zoll LCD Monitor verwendet, der einen DVI Anschluss hat und noch kein HDMI unterstützt. Die Auswahl fällt aufgrund des Projektbudgets auf diesen Monitor. Er war gratis verfügbar und ist nicht weiter in Gebrauch. Allfällige Inkompatibilitäten können durch Adapterkabel (HDMI → DVI) gelöst werden. Der Bildschirm wird durch ein Netzgerät mit 230 Volt betrieben.

Abbildung 3: 24"Monitor von Magic

4.1.2 Hauptrechner

Wichtig ist für den Hauptrechner, eine einfache Anbindung an das Heimnetzwerk zu ermöglichen. Weiter muss direkt eine Video Schnittstelle wie HDMI, DVI oder VGA vorhanden sein. Dies funktioniert alles mit einem Raspberry Pi 3 sehr gut, die Netzwerkverbindung sogar auch kabellos mittels Onboard-WLAN Chip. Weiter wird das Opensource Projekt von einer grossen Community auf dem Raspberry Pi eingesetzt, was ein grosser Pluspunkt im Falle von Fragen mit dem Projekt ist. Ein weiterer Pluspunkt des Raspberry Pi's ist die Verfügbarkeit und die tiefen Kosten. Diese Gründe machen ein Raspberry Pi 3 Model B zum Hauptrechner in dieser Semesterarbeit.

Abbildung 4: Raspberry Pi 3 Model B

4.1.3 Infrarot Präsenzmelder

Wie in der Zielsetzung erwähnt, soll der Magic Mirror möglichst Energieeffizient sein. Der grösste Energieverbrauch liegt sicherlich beim Betreiben des Monitors. Deshalb soll dieser automatisch ausschalten, wenn niemand zu Hause ist. Diese Detektion kann mit einem Infrarot Präsenzmelder gemacht werden. Aus Kostengründen wird der HC-SR501 eingesetzt. Dieser bietet die Möglichkeit, über ein Potentiometer die Sensitivität einzustellen um eine optimale Anpassung an den Einsatzraum zu gewährleisten.
Der Melder funktioniert ab einer Speisespannung von 3.3V. Bei einer Detektion wird ein Puls am Ausgangspin generiert.

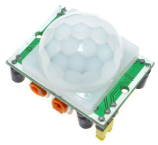

Abbildung 5: Infrarot Präsenzmelder HC-SR501

4.1.4 Hilfsrechner

Um den erwähnten Puls vom Infrarot Präsenzmelder zu verarbeiten und einen Befehl an den Magic Mirror zu senden, muss noch eine weitere Recheneinheit vorhanden sein. Natürlich könnte auch das Raspberry Pi diese Aufgabe übernehmen. Jedoch müsste dann der Präsenzmelder direkt über ein Kabel verbunden sein mit dem Magic Mirror. Dies ist nicht wünschenswert, da der Präsenzmelder entfernt vom Magic Mirror aufgestellt soll, um Bewegungen im Raum zu erkennen bevor der Betrachter den Magic Mirror sieht. Damit bleibt genügend Zeit um den Bildschirm zu starten und der Betrachter findet den aktiven Magic Mirror vor.
Diese Aufgabe der Verarbeitung von einem Puls erfordert keine aufwändigen Berechnungen. Wichtig ist aber, dass der Hilfsrechner eine Möglichkeit hat ins Netzwerk zu kommen um mit dem Magic Mirror zu kommunizieren. Der ESP32 erfüllt diese Anforderung sehr gut, da er einen WLAN Chip integriert. Weiter ist das Board sehr kostengünstig erwerbbar. Es handelt sich um einen Chip mit zwei Kernen und 600 DMIPS

4.1.5 WLAN Router

Die Kommunikation soll in einem Heimnetz funktionieren, wobei eine Verbindung zum Internet bestehen muss, um alle Funktionalitäten des Magic Mirrors ausschöpfen zu können. Da beispielsweise der Infrarot Präsenzmelder nicht kabelgebunden installiert werden soll, muss ein wireless Netz vorhanden sein, sprich WLAN. Es kann auch ein Hotspot von einem mobilen Gerät

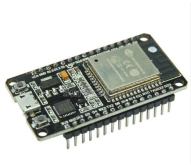

Abbildung 6: ESP32 Board mit Microcontroller und WLAN

erstellt werden um das Netz herzustellen. Der Magic Mirror, wie auch der Infrarot Präsenzmelder verbinden sich jedoch auf eine fix eingestellte SSID.

4.1.6 Laptop

Der Laptop im System wird verwendet um Konfigurationen des Magic Mirror zu ändern. Der Vorgang ist so, dass eine SSH Verbindung zum Raspberry Pi aufgebaut wird, wo die Konfiguration angepasst werden kann.

4.1.7 Handy

Eine Android Applikation wird gebaut, um mit dem Magic Mirror interagieren zu können. Mindestens die Version Android Kitkat 4.4 wird dafür benötigt. Dies ist bei Geräten neuer als 2013 standardmässig gegeben und dadurch wird eine grosse Unterstützung erreicht.

4.2 Auswahl Software

Auf verschiedenen Teilen im System läuft unterschiedliche Software. Die Eigenentwicklungen oder Erweiterungen an bestehender Software sind im Bild unten aufgeführt. Die einzelnen Softwareteile sind in den folgenden Kapiteln detailierter beschrieben und erläutert. Hier ist bloss eine Übersicht gegeben.

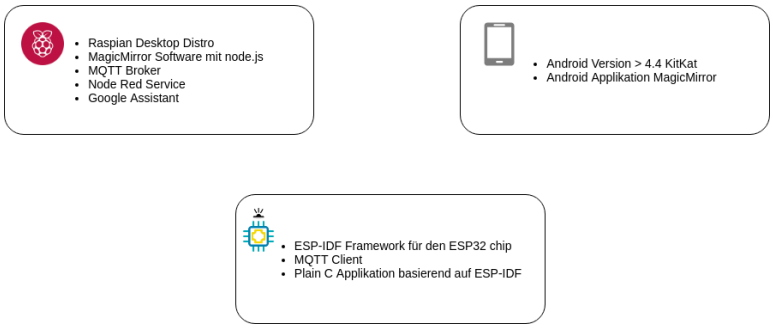

Abbildung 7: System Software Overview

4.2.1 Raspian auf Raspberry Pi

Der einfachste Weg um einen Desktop als Ausgabe zu haben, ist die Raspian Desktop Distribution für das Raspberry Pi zu verwenden. In diesem Projekt wird die aktuellste Version von Raspian Stretch vom 13. November 2018 verwendet, welche die Kernelversion 4.14 beinhaltet.

4.2.2 MagicMirror^2

Das MagicMirror^2 kann direkt von GIT geklont werden. Die Möglichkeit besteht, dies direkt vom Raspberry Pi zu machen. Um aber das Projekt nicht direkt auf dem Raspberry Pi konfigurieren zu müssen besteht die Variante mittels Kopiervorgängen über das Netzwerk einzelne Konfigurationsinhalte auf dem Laptop anzupassen und anschliessend auf das Raspberry Pi zu kopieren. Durch die grosse Community des Projekts ergibt sich ungefähr monatlich ein neues Stablerelease, was einfach mit *git pull* upgedatet werden kann.

4.2.3 MQTT Broker

Auf dem Raspberry Pi wird der wohl bekannteste MQTT Broker verwendet. Namentlich der Mosquitto Broker. Die Installation ist sehr einfach und kann über den üblichen Paketmanager abgehandelt werden.

4.2.4 Node-Red

Node-Red wird verwendet, um den externen Infrarot Präsenzmelder in das System zu integrieren und auf dessen publifizierte Nachrichten zu agieren. Der Dienst ist erreichbar im Netzwerk unter dem Port 1880. Das heisst mit *http://magicmirror:1880* wird der Dienst erreicht und dessen Verhalten kann geändert werden. Dabei ist "magicmirror"der Netzwerkname vom Raspberry Pi.

4.2.5 Android Applikation

Die Entwicklung einer Android Applikation ist am besten unterstützt mit dem Android-Studio. Dabei engt man die Unterstützung auf Android Geräte ein, was jedoch in diesem Projekt kein Problem ist.

4.2.6 ESP-IDF Programm

Um die Funktionalitäten vom ESP32 Microcontroller zu benutzen ist ein Framework sehr hilfreich. *Espressif IoT Development Framework* [1] ist das meistbenutzte und bietet neben Benutzung der IO Pins auch direkte Netzwerkinteraktion über WLAN, sogar das MQTT Protokoll ist im Framework implementiert und bereit zur Verwendung.

5 Magic Mirror auf dem Raspberry Pi

Wie bereits beschrieben, wird die Opensource Software von MagicMirror[2] als Grundlage verwendet. Deshalb wird hier als erstes der Ablauf beschrieben um diese Software auf dem Raspberry Pi aufzusetzen. Dabei wird davon ausgegangen, dass das Raspberry Pi bereits mit einer bootbaren SD Karte versehen ist.

5.1 Aufsetzen von MagicMirror[2]

Es gibt ein vorbereitetes Bash Skript, mit dem die Installation automatisch ausgeführt wird. Der Befehl dafür ist:

pi@magicmirror bash -c "$(curl -sL https://raw.githubusercontent.com/MichMich/MagicMirror/master/installers/raspberry.sh)"

Die Installation erstellt im */home/pi* einen Ordner mit dem Namen *MagicMirror*, welcher das ganze Projekt beinhaltet.
Mit einem Startup-Skript kann der Projekt einfach beim Booten gestartet werden. Das Skript hat folgenden Inhalt:

cd /MagicMirror
DISPLAY=:0 npm start

Es trägt den Namen *mm.sh* und wird mit dem Tool *pm2* zum Raspberry Bootprozess hinzugefügt.
Nach diesen kurzen Schritten wird bereits beim Starten des Raspberry Pi's automatisch der MagicMirror$_2$ gestartet mit der Default-Konfiguration, welche folgendermassen daherkommt:

Abbildung 8: Standardanzeige vom MagicMirror$_2$

5.2 Integration Node-Red und MQTT

Der MQTT Broker kann über den Paketmanager geholt werden. Folgender Befehl ist dafür nötig:

pi@magicmirror sudo apt-get install -y mosquitto mosquitto-clients

Danach ist er gestartet und kann von Node-Red verwendet werden. Erst muss aber auch Node-Red aufgesetzt werden. Dies kann auch automatisiert erfolgen mit dem Aufruf von:

pi@magicmirror bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)

Nach erfolgter Installation kann Node-Red zum Autostart hinzugefügt werden mit dem Befehl:

pi@magicmirror sudo systemctl enable nodered.service

Sobald Node-Red läuft, kann der grafische Designer erreicht werden über die URL *http://magicmirror:1880*. Dort wird schlussendlich die ganze Funktionalität, welche Node-Red übernimmt beschrieben. Folgendes wird gefordert:

- Subscriben an Topic von Infrarot Präsenzmelder.
- Nachricht im Topic prüfen.
- Trigger für Zeitmessung.
- Entscheiden ob Bildschirm an- oder ausgeschaltet werden soll.
- Entsprechendes Skript aufrufen.

Die geforderten Punkte können grafisch folgendermassen dargestellt werden in Node-Red: Der dargestellte Ablauf ist nicht komplex. Links ist mit dem

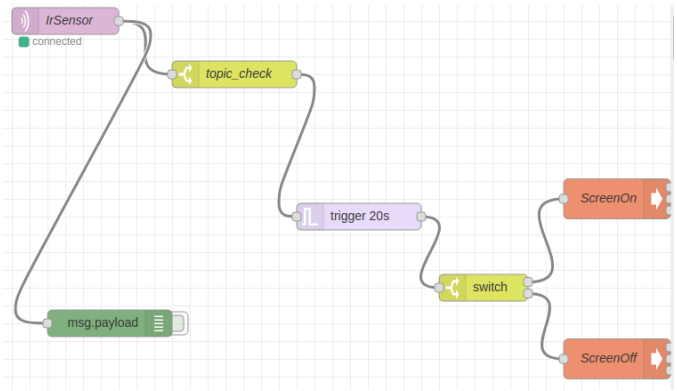

Abbildung 9: Node-Red grafische Implementation

IrSensor die Registration auf dem Topic */topic/magicmirror/irsensor* gemacht unter Localhost, da der MQTT auf demselben Gerät läuft. Mit dem Block *topic_ check* wird geprüft, ob eine neue Nachricht mit dem Inhalt ïrsensor onërhalten wurde. Alles andere wird ignoriert. Falls eine solche Nachricht erhalten wurde, wird vom *trigger 20s* eine Nachricht mit dem Inhalt önän den *switch* gesendet, welcher dann *ScreenOn* ausführt. Falls nach 20 Sekunden keine weitere Nachricht mit ïrsensor on"beim *trigger 20s* angelangt ist, sendet er automatisch ein öffßum *switch*, welcher dann *ScreenOff* ausführt. *ScreenOn* führt folgendes Kommando aus um den Bildschirm einzuschalten:

tvservice -p && sudo chvt 6 && sudo chvt 7

Beim *ScreenOff* ist es der folgende Befehl um den Bildschirm auszuschalten:

tvservice -o

5.3 Implementation eigenes MagicMirror² Modul

Auf dem Magic Mirror sollen mittels einer Android Applikation Notizen geschrieben und gelesen werden können. Beim Durchsuchen nach einem passenden Modul welches dies bereits anbietet ist folgendes hervorgekommen: *MMM-Memo* [3].
Jedoch erfüllt die Implementation dieses Modules noch nicht alle Anforderungen, welche sind:

- Rest API wird angeboten als Schnittstelle zum Modul.
- Mehrere Notizgruppen könne hinzugefügt werden. Hier gewünscht sind *TODO* und *Shopping*.
- Ansprechend gestaltbar.

Der erste Punkt wird nicht unterstützt. Das Modul erwartet neue Notizen als Teil der URL, als Beispiel um Brot auf die Shoppingliste zu schreiben müsste folgender Zugriff gemacht werden:

http://magicmirror:8080/AddMemo?memoTitle=SHOPPING&item=Brot&level=INFO

Um es dann wieder zu entfernen, braucht es folgenden Aufruf:

http://magicmirror:8080/DisplayMemo?memoTitle=SHOPPING&item=1

Dabei muss die Übersicht über die einzelnen Notizen behalten werden, da sie jeweils nur noch über deren Index (siehe 1 für Brot) angesprochen werden können. Dies soll nicht Aufgabe der Android Applikation sein.
Die Lösung um dies zu Umgehen ist die Implementation einer Rest API, welche jeweils den kompletten Satz an Notizen zurückliefert oder überschreibt. Folgende Zugriffe werden implementiert:

To get all notes: *http://magicmirror:8080/GetCompleteNote*

To write all notes: *http://magicmirror:8080/AddCompleteNote*
To delete all notes: *http://magicmirror:8080/DeleteAllNotes*

Die Notizen werden als JSON übertragen. Dabei werden HTTP-Get Befehle für die Zugriffe *GetCompleteNote* und *DeleteAllNotes* verwendet und ein HTTP-Post Befehl für den Befehl *AddCompleteNote* mit dem JSON-Objekt im Body. Das JSON-Objekt sieht wie folgt aus:


```
[
    {
        "memoTitle" : "shopping",
        "level" : "INFO",
        "item" : "Brot",
        "timestamp" : 2019-01-15T19:30:42.499Z"
    },
    {
        "memoTitle" : "shopping",
        "level" : "INFO",
        "item" : "Salat",
        "timestamp" : 2019-01-15T19:35:37.201Z"
    },
    {
        "memoTitle" : "todo",
        "level" : "INFO",
        "item" : "Wohnung putzen",
        "timestamp" : 2019-01-15T19:35:39.57Z"
    }
]
```


Abbildung 10: JSON Beispiel

Die einzelnen Notizen werden in einem Array abgelegt. Beim aufgeführten Beispiel werden Brot und Salat zur Gruppe *Shopping* und die Aufgabe Wohnung putzen zur Gruppe *TODO* hinzugefügt.

Die nötigen Erweiterungen an dem Modul belaufen sich auf ein einzelnes JavaScript-File, wo zusätzliche Behandlungen gemacht werden müssen, je nach dem welche Anfrage kommt. Dabei werden die bisherigen (wo die ganze Information in der *url* vorhanden ist), weiter unterstützt.

5.4 Anpassen der Anzeigeelemente auf dem Magic Mirror

Diese Anpassungen können sehr einfach behandelt werden. Beim Starten des MagicMirror[2] wird eine Konfigurationsdatei gelesen, welche alle Daten über die zu ladenden Module beinhaltet. Es handelt sich um folgende Datei:

/home/pi/MagicMirror/config/config.js

Die einzelnen Module werden darin als JSON-Objekte beschrieben. Alle verwendeten Module sind im untenstehenden Bild aufgeführt:

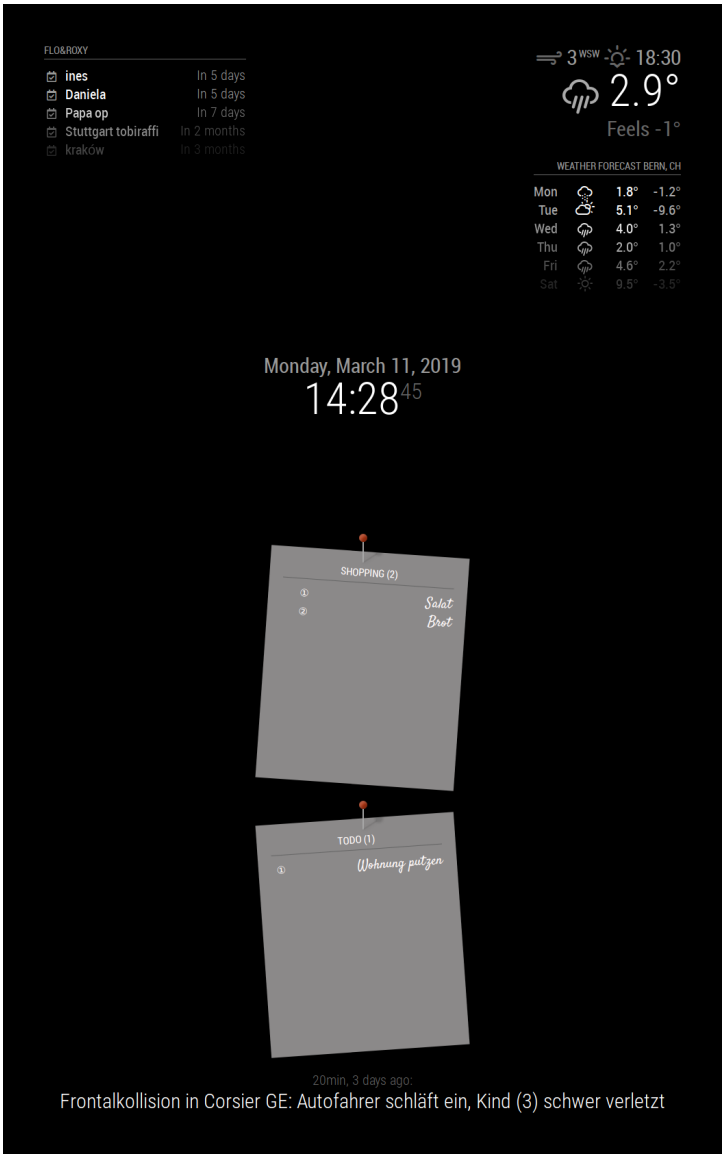

Abbildung 11: Magic Mirror Screenshot mit allen Modulen

6 Applikation für Präsenzdetektion

6.1 Framework Übersicht

Mit dem Framework von Espressif [1] ist eine sehr umfängliche Basis gegeben, um Funktionalitäten auf dem ESP32 Microcontroller zu programmieren. Neben Zugriffen auf die Peripherien des Controllers, sind verschiedene Protokolle im Framework integriert, FreeRTOS kann eingebunden werden, wie auch ein Monitoring Dienst, der als Ausgabekonsole verwendet werden kann. Als Übersicht das Bild:

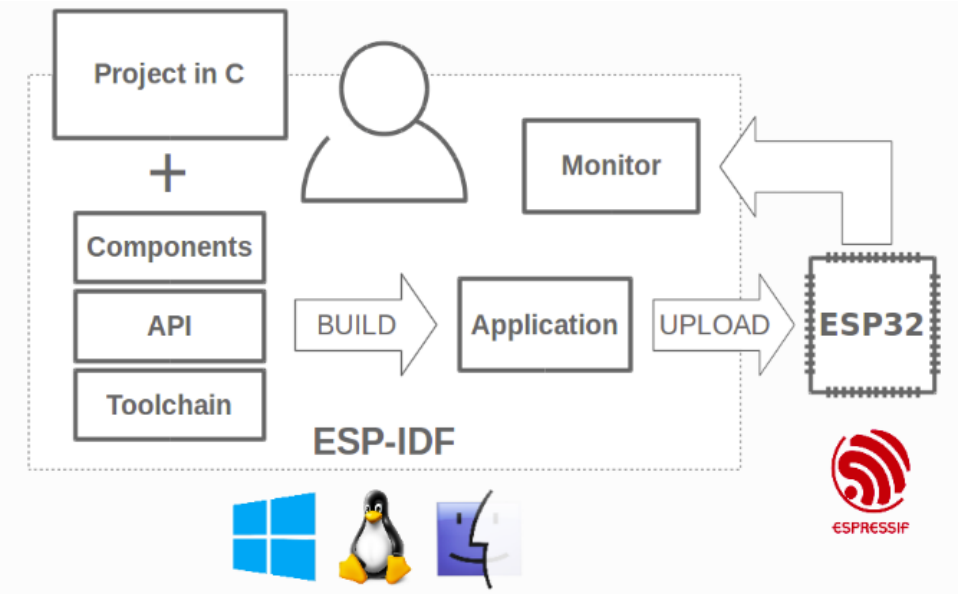

Abbildung 12: ESP32 Framework Übersicht

Zudem hat das Entwicklungsboard direkt einen eingebauten USB-JTAG Adapter, wie auch eine OpenOCD Debugschnittstelle.

6.2 Ablauf des Programmes

Der schrittweise Ablauf des Programmes ist in folgendem Diagramm dargestellt.

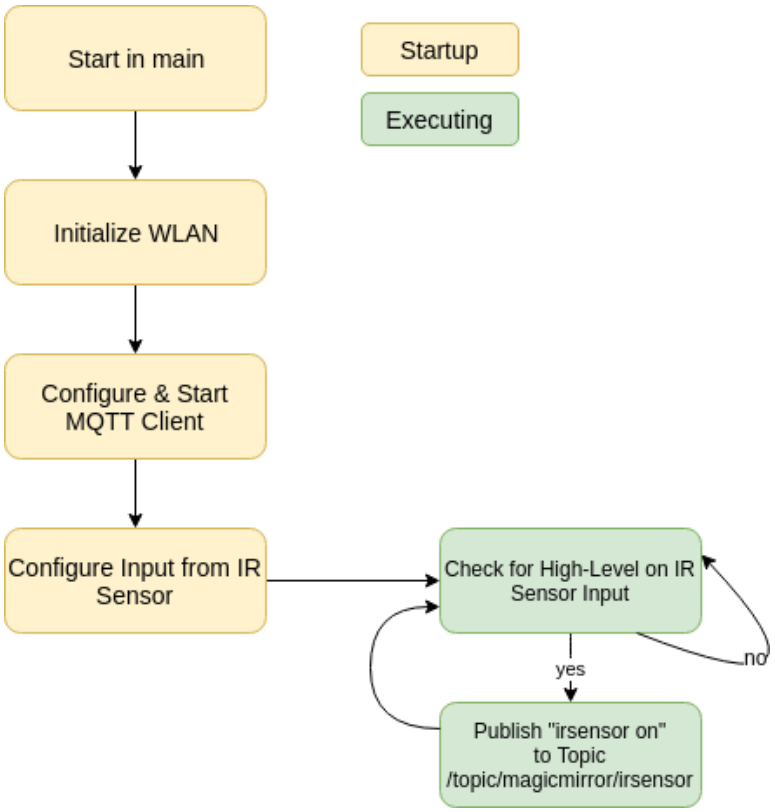

Abbildung 13: ESP32 Softwareablauf

Als Beispiel wird der C-Code aufgeführt, um eine Detektion des Infrarot Präsenzmelders mittels MQTT zu publizieren:

```
char buf[20];
strcpy(buf, "irsensor on");
esp_mqtt_client_publish(client, TOPIC_PATH_SENSOR, buf, 0, 0, 0);
```

Dabei ist die Funktion *esp_mqtt_client_publish* Teil des Frameworks und erwartet als Übergabeparameter ein Handle vom initialisierten MQTT-Client, das Topic (in dem Fall */topic/magicmirror/irsensor*), die Daten (*irsensor on*), die Länge der Daten (falls kein String gesendet wird), die QOS (hier nur einmal) und ein Flag ob die Nachricht bei einem neuen Subscriber gleich vom Broker gesendet werden soll (hier nicht eingeschaltet).

7 Android Applikation für Notizen

Die Aufgabe der mobilen Android Applikation ist das bedienen der beschriebenen Rest API Schnittstelle des Magic Mirrors um Notizen zu senden und zu verwalten. Wie erwähnt wird die Applikation mit Hilfe von Android-Studio entwickelt.

7.1 App Bedienung

Der Benutzer muss einfach zwischen den verschiedenen Notizgruppen navigieren können. Wünschenswert ist dies mit einer Wischbewegung zu machen, da es intuitiv ist. Android-Studio bietet bereits solche Projekte als Einstiegspunkt an. Als Grundlage hat das sogenannte *Tabbed Activity* gedient, was bereits seitliche Scrollingeffekte mit einer Actionbar integriert.
Die finale Applikation kann somit zwischen den verschiedenen Notizgruppen mit links und rechts Wischen wechseln. Siehe untenstehendes Bild für die Veranschaulichung.

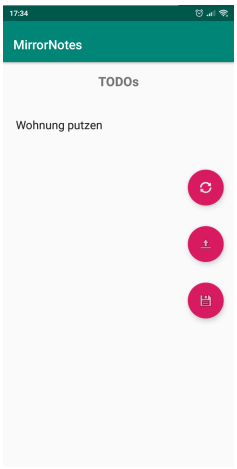

Abbildung 14: App TODOs

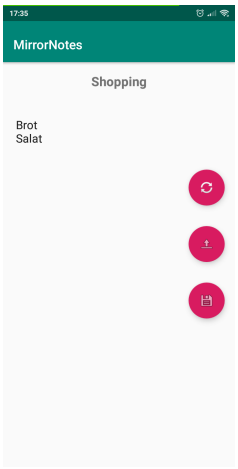

Abbildung 15: App Shopping

Die drei ersichtlichen Buttons haben folgende Funktionen, aufgeführt auch von oben nach unten:

- Download von Magic Mirror → Aktualisiert Notizen auf der App mit den Notizen auf dem Magic Mirror.
- Upload auf Magic Mirror → Aktualisiert die Notizen vom Magic Mirror mit den angepassten Notizen auf der App.
- Printscreen → Speichert lokal einen Printscreen der aktuellen Ansicht, um Notizen auch ausserhalb des Heimnetzes zur Verfügung zu haben.

7.2 App Programmierung

Die Funktionalität des Apps ist überschaubar. Um trotzdem eine gewisse Flexibilität zu erreichen wurde die Software folgendermassen aufgebaut.

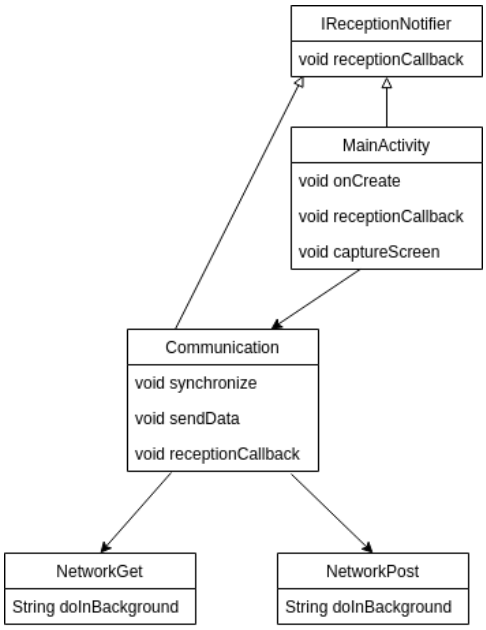

Abbildung 16: Klassendiagramm Android App

Sehr auffällig ist dabei die doppelte Verwendung eines Interfaces auf verschiedenen Ebenen. Jedoch definiert das Interface *IReceptionNotifier* eine Callback-Funktion, die einen String als Argument beinhaltet. Dies wird auf beiden Ebenen benötigt.
Die Kommunikation mit dem Magic Mirror geschieht in den untersten Netzwerk Klassen (*NetworkGet* und *NetworkPost*). Diese werden jeweils von einem eigenen Backgroundtask gehandelt, sobald eine Netzwerkkommunikation erwartet wird. Dies um nicht die Benutzereingaben auf dem Userinterface zu behindern.
Eine Ebene darüber, bei der *Communication* Klasse wird das Parsen von den JSON Objekten durchgeführt, um dann eine Liste mit Strings an die *MainActivity* weiterzuleiten. Diese Liste wird dann dem Benutzer angezeigt.

8 Google Assistant

Ein zusätzliches Feature für den Magic Mirror ist die Spracherkennung. Nötig dazu ist bloss ein Mikrophon. Um den Magic Mirror anzuweisen Musik zu spielen, soll ebenfalls ein Lautsprecher eingebunden werden. Da mit dem Raspberry Pi 3 die Unterstützung für Bluetooth vorhanden ist, kann sogar ein Bluetooth-Lautsprecher verwendet werden.

8.1 Evaluation

Als Mikrophon wird ein low-cost USB Mikrophon von Rondaful verwendet, welches in untenstehendem Bild gezeigt ist. Als Bluetooth-Lautsprecher wird

Abbildung 17: USB Mikrophon

ein etwas teureres Gerät verwendet, nämlich der protable Bluetooth-Speaker Charge 3 von JBL. Mit diesen beiden Geräten kann evaluiert werden, inwie-

Abbildung 18: Bluetooth Speaker JBL Charge 3

fern der Google Assistant sinnvol verwendet werden kann im Magic Mirror. Punkte die untersucht werden während der Evaluation:

- Mögliche Distanz um Sprachbefehle zu senden.
- Qualität der Musikausgabe.

8.2 Musikausgabe

Mit den Tools *PulseAudio* und *BluetoothCtl* kann der Lautsprecher mit dem Raspberry Pi verbunden werden um anschliessend ein Soundfile abzuspielen. Das Resultat dieses Tests war positiv. Die Qualität ist vergleichbar mit jeder anderen Verbindung zu diesem Bluetooth-Speaker.

8.3 Mikrophon Spracheingabe

Die Einrichtung des Mikrophones am Raspberry Pi ist sehr einfach. Plug-and-Play ist ein guter Beschrieb dazu. Die Qualität der aufgezeichneten Sprache war jedoch sehr negativ. Mit dem Befehl

arecord -D plughw:1,0 -d 3 test.wav && aplay test.wav

kann eine Aufnahme von drei Sekunden gemacht werden und wird anschliessend gleich abgespielt (hier über den 3.5mm Audio Ausgang).
Der Test hat gezeigt, dass bei einer Entfernung von > 1 Meter die Qualität kaum mehr verständlich ist. Bei < 1 Meter versteht man zumindest noch Sprache. Jedoch ist dieses Resultat eine klare Aussage gegen dieses Feature der Spracheingabe für den Magic Mirror.
Der wahrscheinlichste Grund für diese schlechte Qualität ist das günstige Mikrophon. Jedoch ist es nicht mehr im Bereich dieser Semesterarbeit weitere Untersuchungen in diesem Bereich zu tätigen.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Espressif. Espressif iot development framework. 2019. URL: `https://docs.espressif.com/projects/esp-idf/en/latest/`.

[2] Teeuw Michael. Magicmirror opensource project. 2019. URL: `https://magicmirror.builders/`.

[3] Schnibel. Notes modul fãijr den magicmirror. 2019. URL: `https://github.com/schnibel/MMM-Memo`.

[4] Various. Magicmirror modules. 2019. URL: `https://github.com/MichMich/MagicMirror/wiki/3rd-party-modules`.