



TECHNIKERARBEIT

# **ENTWICKLUNG EINER SMART HOME ZENTRALE AUF BASIS EINES RASPBERRYPI**

Felix KUSCHEL & Manuel STARZ

Betreut durch  
Matthias KOHLER

28. März 2021

---

## Inhaltsverzeichnis

## Erklärung

Hiermit erklären wir, dass die vorliegende Technikerarbeit eine eigenständige Leistung darstellt und nicht auf Basis einer bereits vorhandenen Techniker-, Diplom-, oder ähnlichen Arbeit erstellt wurde. Verwendete Quellen haben wir vollständig und nachprüfbar aufgeführt. Bei der Durchführung und Ausarbeitung wurden nur die zulässigen Hilfsmittel verwendet.

Uns ist bewusst, dass bei einem Verstoß gegen diese Erklärung innerhalb der gesetzlichen Einspruchsfristen auch im Nachhinein die Leistungsbewertung aberkannt werden kann. Damit erlischt die Berechtigung zum Tragen der Berufsbezeichnung „staatlich geprüfter Techniker“.

Bedanken möchten wir uns bei Herr Kohler für die Unterstützung und Betreuung bei der Erstellung der Technikerarbeit.

---

Datum, Ort

---

Felix Kuschel

---

Datum, Ort

---

Manuel Starz

# 1 Vorwort

## 1.1 Einleitung

Bei dieser Ausarbeitung handelt es sich um die Abschlussarbeit zur Weiterbildung zum staatlich geprüften Techniker in der Fachrichtung Informationstechnik. Diese Arbeit basiert auf dem in den zwei Jahren erlernten Stoffs sowie selbst erarbeiteten Kenntnissen und dient zur Feststellung des Erreichen des Fortbildungszieles.

## 1.2 Projektrahmen

Das Projekt zur Erstellung einer Smart Home Zentrale wurde von uns, Felix Kuschel und Manuel Starz, durchgeführt. Der Projektzeitraum war vom 1. September 2020 bis zum 30.03.2021 angesetzt. Es stand zur Umsetzung des Projekts ein ununterrichtsfreier Tag pro Woche zur Verfügung.

Des weiteren wurden die in dem Zeitraum zur Verfügung stehenden Ferien zur Umsetzung des Projekts genutzt. Die Projektbetreuung erfolgten seitens der Schule durch Herr Matthias Kohler. Da das Projekt nicht in Zusammenarbeit mit einem Unternehmen durchgeführt wurde, gibt es keine weiteren Betreuer. Die Materialkosten für die im Projekt genutzte Hardware wurde von uns selbst getragen.

## 1.3 Aufgabenstellung

Das Projekt ist aus dem Zusammenschluss der Abschlussarbeitsideen von uns, Felix Kuschel und Manuel Starz, entstanden und wurde mit Rücksprache mit dem betreuenden Lehrer entwickelt.

Durch die große Verfügbarkeit von Smart Home Geräten und den zahlreichen Standards der Anbieter entschlossen wir uns, eine einfache und leicht zu replizierende Lösung zu entwickeln, die Smart Home Geräte mehrerer Hersteller miteinander verknüpft und so die Notwendigkeit mehrerer verschiedener sogenannter Hubs zu eliminieren.

Des weiteren soll das Gerät noch über einen Touchscreen steuerbar sein und die Werte der verbundenen Smart Home Geräte anzeigen. Dies umfasst unter anderem den Status von Leuchtmitteln, die Werte von Thermostaten sowie den Zustand von Tür- und Fensterkontakten. Die genaue Aufgabenstellung kann dem abgegebenen Lastenheft im Anhang entnommen werden.

## 1.4 Zusatzinformationen

Die Rohdaten des Projekts wurden der Einfachheit in einem GIT-Projekt zusammengefasst. Dadurch konnten die Durchführenden unabhängig voneinander an dem Projekt und der Dokumentation arbeiten. Der Link zu dem Projekt lautet:

<https://github.com/Pharias/TAR>

Ursprünglich wurde für die Verwendung des schulinternen GIT verwendet. Dies stand zum Zeitpunkt der Erstellung der Dokumentation allerdings nicht zur Verfügung<sup>1</sup>, weshalb eine Alternative genutzt wurde.

<sup>1</sup> War lt. Herr Kohler nicht verfügbar aufgrund Sicherheitsrisiken

## 1.5 Definition Smart Home Zentrale



Abbildung 1: Smart Home Zentralen

Smart Home Zentralen, auch Smart Hubs oder Smart Mirrors genannt, sind Geräte, die als zentraler Knotenpunkt in einem Smart Home Netzwerk sitzen und dort Informationen verarbeiten, weiterleiten und darstellen können.

Diese Geräte werden von den meisten Herstellern mit und ohne Bildschirm geliefert, um entweder ein neues Smart Home aufzubauen oder ein bestehendes Smart Home zu erweitern.

Bei einem Smart-Home-Hub handelt es sich um eine schlaue Zentrale, durch die all deine intelligenten Geräte miteinander vernetzt werden – und dadurch erst wirklich ihren gesamten Leistungsumfang ausschöpfen.<sup>2</sup>

Als Smart Home Zentrale können auch Software-Lösungen gezählt werden, die mit den im Netz befindlichen Smart Hubs kommunizieren und die Informationen mit Hilfe eines Web-Interfaces oder einer Smartphone-Anbindung darstellen und steuern können. Beispiele hierfür sind homeassistant.io, openHAB und Google Home.



Abbildung 2: Smart Home Software

## 1.6 Datenschutzhinweis

Aus Datenschutzgründen sind lokale IP-Adressen und lokale Domänennamen innerhalb der Dokumentation unkenntlich gemacht.

<sup>2</sup> Li (2017): Was ist ein Smart-Home-Hub? Alles über die intelligente Zentrale

## 2 Zielsetzung

Als Ziel für das Projekt war ein funktionsfähiges Smart Home Hub mit Zigbee-Anbindung auf Basis eines Raspberry Pi 4 geplant. Im Laufe des Projekts kam dann noch eine Erweiterungskarte für den Raspberry Pi, ein sog. Raspberry Pi HAT, hinzu, welcher aber aufgrund der vorliegenden Lage mit der COVID-19-Pandemie und den damit verbundenen Liefer- und Zollschwierigkeiten verworfen wurde. Die Planung des HAT ist daher nur rudimentär und nicht vollständig, kann aber bei Bedarf nach Beenden des Projekts durchgeführt werden. Das Smart Home Hub in seiner Grundfunktion soll in der Lage sein, die mit ihm verbundenen Geräte über den Zigbee-Standard anzusteuern. Darüber sollte die Möglichkeit einer Erweiterung mit einem Sprachassistenten gegeben sein.

### 2.1 Konzeption

Zu Beginn des Projekts haben wir Nachforschung angestellt und uns bereits vorhandene Open-Source-Lösungen im Bereich Smart Home angeschaut. Dabei sind wir neben dem Smart Mirror GLANCR auch auf die Gesamtlösung homeassistant.io sowie openHAB gestoßen. Darauf hin haben wir ein Grundkonzept in unserem Lastenheft zusammengefasst und dieses in Absprache mit unserem Betreuungslehrer, Herr Kohler, ausgearbeitet.

Nach Abgabe des Lastenhefts haben wir uns dann an die Beschaffung der unserer Meinung nach nötigen Komponenten für das Projekt gemacht.

### 2.2 Anforderungen und gewünschte Features

Die Anforderungen an das Projekt lauteten demnach wie folgt:

- Anbindung von ZigBee-fähigen Endgeräten
- Steuerung der angebundenen Endgeräte
- Übermittlung der Zustände der angebundenen Geräte an z.B. ein Smartphone

Diese Anforderungen lassen sich mit einem Raspberry Pi und einem Zigbee-USB-Stick realisieren. Darüber hinaus waren von unserer Seite noch folgende Features gewünscht:

- Ein- und Ausgabe über einen Touch-Bildschirm
- Einbindung eines Sprachassistenten zur Steuerung der eingebundenen Endgeräte

Nach Fortschritt des Projekts kam bei einer Rücksprache mit unserem Projektbetreuer die Idee auf, einen Raspberry Pi Hat speziell für das Projekt zu entwickeln. Dieser sollte die Hardware des Projekts falls möglich auf einer Platine vereinen, die dann auf den Raspberry Pi aufgesteckt werden konnte.

Diese Erweiterungsplatine sollte folgende Eigenschaften besitzen:

- ZigBee-Controller und Antenne

- NFC-Controller und Antenne
- RGB-LED zur Statusanzeige
- Anschluss für Lüfter
- Sensoren für:
  - Luftfeuchtigkeit
  - Temperatur
  - Luftdruck
- Pins zum Anschluss an Versuchsaufbau für Laborgebrauch

Die Erweiterungsplatine wurde aber wie zuvor aufgrund der aktuellen Pandemie-Situation und den damit verbundenen Beschaffungsschwierigkeiten verworfen. Darauf wurde dann klar, dass eine Erstellung eines Installationsskripts sowie Image für den Raspberry Pi eine sinnvolle Ergänzung der Projektarbeit wäre. Zusätzlich haben wir ein Gehäuse für die Hardware geplant, um das Endprodukt wertiger gestalten zu können.

## 3 Herangehensweise

Nach Festlegung der Anforderungen haben wir uns dann an die Beschaffung und der benötigten Materialien und die Einrichtung der Hard- und Software gemacht. Hierfür haben wir zum Teil bereits vorhandene Hardware, z.B. den Raspberry Pi 4 mit weiteren Komponenten wie dem Touch-Bildschirm und den Lautsprechern sowie dem Mikrofon ergänzt. Damit ist die Umsetzung des Projekts in zwei Teile geteilt:

### 3.1 Hardware

Bei der Entwicklung der Hardware haben wir für unser Projekt intern einige Rahmenbedingungen festgelegt:

- Das Projekt sollte nach Möglichkeit von durchschnittlichen Bastlern durchgeführt werden
- Zur Fertigstellung des Projekts sollte mit Handelsüblichen Bastler Werkzeugen möglich sein
- Die Grundplattform sollte der 2020 herausgebrachte Raspberry Pi 4 8GB sein<sup>3</sup>
- Das Gerät sollte möglichst von vielen Herstellern vertriebenen Zigbee-Endgeräte verwalten können.

Die Beschaffung der Teile lief dank Online-Versandhandel relativ problemlos. Beim Zusammenbau haben wir den Prototyp lediglich mit dem Bildschirm, dem Pi und dem Zigbee-USB-Stick aufgebaut (vgl. Abschnitt ??: ??). Die beiden Standfüße stammen vom Hersteller des Bildschirms Sunfounder<sup>4</sup>. Diese haben wir aus PETG auf dem Ender 3 gedruckt. Anschließend haben wir die Softwareseite des Projekts bearbeitet (vgl. Abschnitt ??: ?? & Kapitel ??: ??).

Nachdem der Prototyp soweit funktionsfähig war, haben wir uns daran gemacht, das Präsentationsmodell zu entwerfen. Dabei ging es uns vornehmlich um die Unterbringung der geplanten Komponenten in einem simplen Gehäuse. Hierfür waren einige Verlängerungskabel nötig, um Stromzufuhr, Netzwerk und die Antenne des Zigbee-Sticks nach außen zu leiten (vgl. Abschnitt ??: ?? & Abschnitt ??: ??). Das Gehäuse wurde dann nach dem Druck mit dem Bildschirm verklebt und die einzelnen Komponenten verbaut (vgl. Abschnitt ??: ??). Nachdem das Gerät soweit fertig war, haben wir einen Versuchsaufbau zur Präsentation aufgebaut, der die Funktionsweise des Geräts in einem „typischen“ Heimnetzwerk darstellen soll (vgl. Abschnitt ??: ??).

Näheres zur Herangehensweise im Kapitel ??: ??.

### 3.2 Software

#### 3.2.1 Home Assistant

Auf der Softwareseite haben wir uns bereits verfügbare Lösungen für Smart Home Zentralen angeschaut. Ursprünglich war unser Plan, eine eigene Softwarelösung mit

<sup>3</sup> Wikipedia: Raspberry Pi - Generations

<sup>4</sup> Sunfounder: 10.1 Inch Touch Screen for Raspberry Pi(NEW) - 3D-printed Touch Screen Support

Web-Interface oder einer App-Anbindung für Smartphones selbst zu programmieren. Diese Idee haben wir nach kurzer Zeit aber wieder verworfen, da dieser Markt bereits mehr als gesättigt ist mit kostenlosen als auch kostenpflichtigen Lösungen. Von den kostenlosen und quelloffenen Lösungen war für uns Home Assistant am interessantesten, vor allem wegen der breiten Menge an Addons und der unserer Meinung nach recht ansprechenden Web-Oberfläche. Home Assistant verfügt über drei grundlegende Möglichkeiten, installiert zu werden:

- Als eigenes Betriebssystem-Image (Home Assistant Operating System)
- Als Docker Container
- Als Programm auf einem Server

Unabhängig von der Installationsart wird der Home Assistant in zwei Versionen angeboten:

- Als Core Version
- Als Supervised Version

Die Core Version ist die grundlegende Version des Home Assistants während die Supervised Version ein Addon-Repository integriert hat.<sup>5</sup>

Darüber hinaus vertreibt Home Assistant auch noch eine fertige Out-of-the-Box-Lösung namens Home Assistant Blue<sup>6</sup> für 140\$, allerdings gibt es nur wenige Länder, in denen das Gerät verfügbar ist und es besitzt anders als unsere Lösung keinen Bildschirm.

Für unsere Anwendung war das eigene Betriebssystem-Image nicht verwendbar, da sich auf dem System keine grafische Oberfläche nachinstallieren ließ. Deshalb haben wir uns zuerst auf die Programm-Variante des Home Assistants entschieden. Allerdings gab es bei der Installation der Supervised Version auf unterschiedlichen Pi-Versionen Probleme und die Core-Version war für unsere Zwecke ungeeignet, da hier der Arbeitsaufwand für die die Integration von zusätzlichen Komponenten und Systemerweiterungen für die Zielgruppe des Projekts zu komplex ist. Außerdem hat sich während unserer Testphase gezeigt das die Verknüpfung von Home Assistant Core, Mosquitto Broker sowie Zigbee2MQTT zu nicht dauerhaft reproduzierbaren Erfolgen führt und sich deshalb leider nicht wie von uns geplant automatisieren lässt. Deshalb haben wir uns für die Container-Variante mit Docker entschieden. Das Installationsskript (vgl. ??: ??) installiert die benötigten Programme und übernimmt die Grundeinrichtung des Homeassistants bis zum Punkt an dem die Einrichtung in der Weboberfläche durch den Nutzer stattfindet.

Um neben dem Skript noch eine weitere Art der Einrichtungsmöglichkeit anzubieten zu können, haben wir von den SD-Karten nach Ablauf des Installationsskripts ein Backup-Image erstellt und dieses entsprechend der beiden in Verwendung befindlichen Raspberry Pi Versionen<sup>7</sup> erstellt. Der Benutzer und das Passwort entsprechen bei den Images den standardmäßig in Raspberry Pi OS hinterlegten Nutzern und sollten beim booten sicherheitshalber geändert werden  
Näheres zur Herangehensweise im Kapitel ??: ??.

<sup>5</sup> siehe <https://www.home-assistant.io/installation/>

<sup>6</sup> siehe <https://www.home-assistant.io/blue>

<sup>7</sup> Raspberry Pi B Version 3 und Version 4(8GB)

### 3.2.2 MQTT und Zigbee2MQTT

Um die von uns Angestrebte Steuerung und Integration von Zigbee Leuchtmitteln und Geräten zu ermöglichen, benötigen wir den Übertragungsstandard MQTT und die Möglichkeit die Zigbee Daten über diesen an Home Assistant weiter zu reichen. Wir haben uns schon sehr früh auf einen MQTT broker und die Anwendung für Verarbeitung des Zigbee-Protokolls entschieden.

- Mosquitto broker (Opensource MQTT broker)
- Zigbee2MQTT als Zigbee Schnittstelle

#### Beschreibung Mosquitto broker

Eclipse Mosquitto ist ein Open Source (EPL/EDL lizenziert) Message Broker, der das MQTT-Protokoll in den Versionen 5.0, 3.1.1 und 3.1 implementiert. Mosquitto ist leichtgewichtig und eignet sich für den Einsatz auf allen Geräten, von stromsparenden Einplatinencomputern bis hin zu kompletten Servern.

Das MQTT-Protokoll bietet eine leichtgewichtige Methode zur Durchführung von Messaging unter Verwendung eines Publish/Subscribe-Modells. Dadurch ist es für das Internet der Dinge geeignet, z. B. für Sensoren mit geringem Stromverbrauch oder mobile Geräte wie Telefone, eingebettete Computer oder Mikrocontroller.

Das Mosquitto-Projekt bietet auch eine C-Bibliothek zur Implementierung von MQTT-Clients und die sehr beliebten mosquitto-pub und mosquitto-sub Kommandozeilen-MQTT-Clients.<sup>8</sup>

#### Beschreibung Zigbee2MQTT

Zigbee2MQTT besteht aus drei Modulen, die jeweils in einem eigenen Github-Projekt entwickelt wurden. Beginnend bei der Hardware (Adapter) und aufsteigend; zigbee-herdsman verbindet sich mit Ihrem Zigbee-Adapter und stellt eine API für die höheren Ebenen des Stacks zur Verfügung. Für z.B. Texas Instruments Hardware verwendet zigbee-herdsman die TI zStack Monitoring und Test API, um mit dem Adapter zu kommunizieren. Zigbee-herdsman übernimmt die zentrale Zigbee-Kommunikation. Das Modul zigbee-herdsman-converters übernimmt das Mapping von einzelnen Gerätmodellen auf die von ihnen unterstützten Zigbee-Cluster. Zigbee-Cluster sind die Schichten des Zigbee-Protokolls, die über dem Basisprotokoll liegen und z. B. definieren, wie Lampen, Sensoren und Schalter über das Zigbee-Netzwerk miteinander kommunizieren. Schließlich steuert das Zigbee2MQTT-Modul zigbee-herdsman und bildet die Zigbee-Nachrichten auf MQTT-Nachrichten ab. Zigbee2MQTT behält auch den Status des Systems im Auge. Es verwendet eine database.db-Datei, um diesen Zustand zu speichern; eine Textdatei mit einer JSON-Datenbank der angeschlossenen Geräte und ihrer Fähigkeiten.<sup>9</sup>

<sup>8</sup> mosquitto.org (Übersetzt mit DeepL): <https://mosquitto.org/>

<sup>9</sup> github.com (Übersetzt mit DeepL): <https://github.com/koenkk/zigbee2mqtt>

## 4 Zeitplan

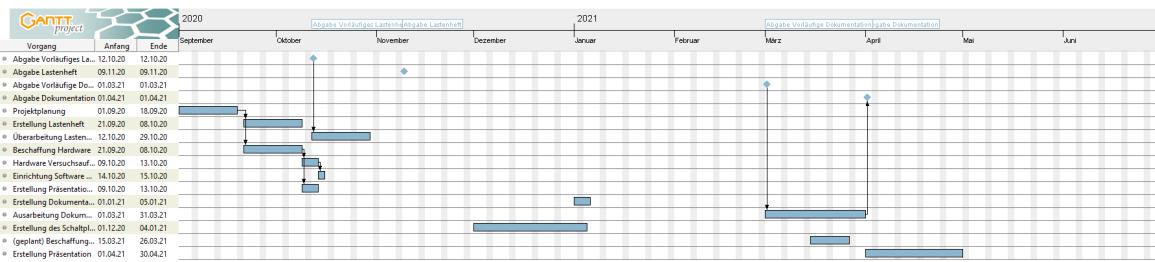


Abbildung 3: gantt-Diagramm des Projektablaufs

Um einen besseren Überblick der Aufgaben zu haben, haben wir mit GanttProject ein Gantt-Diagramm mit den Aufgaben und Meilensteinen unseres Projekts angelegt (vgl. Abb. ??: ??). Die eigentliche Projektorganisation war als SCRUM Projekt geplant. Hierfür haben wir das KANBAN-Board in Microsoft Teams genutzt (vgl. Abb ??: ??). Dadurch haben wir stets einen Überblick über die noch zu erledigenden, offenen und erledigten Aufgaben.

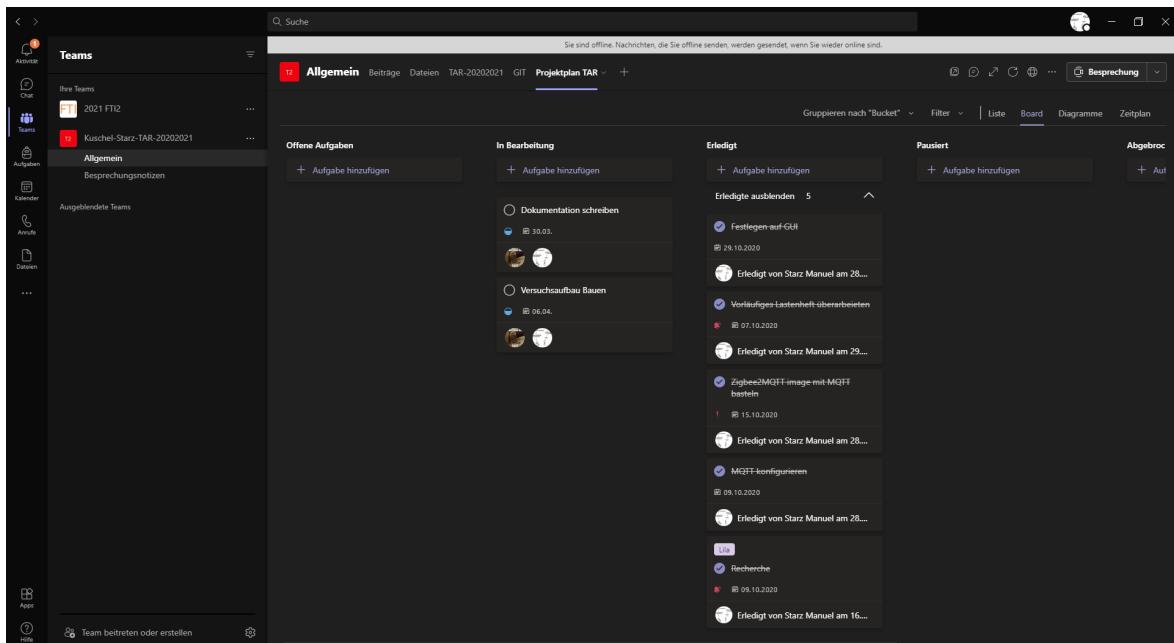


Abbildung 4: KANBAN-Board in Microsoft Teams

Microsoft Teams hat uns auch als Kommunikationsplattform während der Lockdown-Zeiten. Dadurch konnten wir während der üblichen Zeiten mit Herr Kohler in Verbindung treten.

## 5 Komplettübersicht

### 5.1 Das „fertige“ Produkt



Abbildung 5: Smart Home Zentrale

Dies ist das Ergebnis der Technikerarbeit von Felix Kuschel und Manuel Starz. Eine Smart Home Zentrale auf Basis von Homeassistant mit einer ZigBee-Antenne, einem 10.1" großen Touchbildschirm und der Möglichkeit, per LAN und WLAN mit dem Heimnetz in Verbindung zu treten.

Das Gehäuse ist für die Wand-Montage konzipiert (vgl. Abb. ??: ??), die Nutzung eines Gehäuse um die Smart Home Zentrale auf einer Kommode oder in einem Regal zu platzieren (mit einem leichten Winkel) ist ebenfalls möglich. (vgl. Abb. ??: ??)

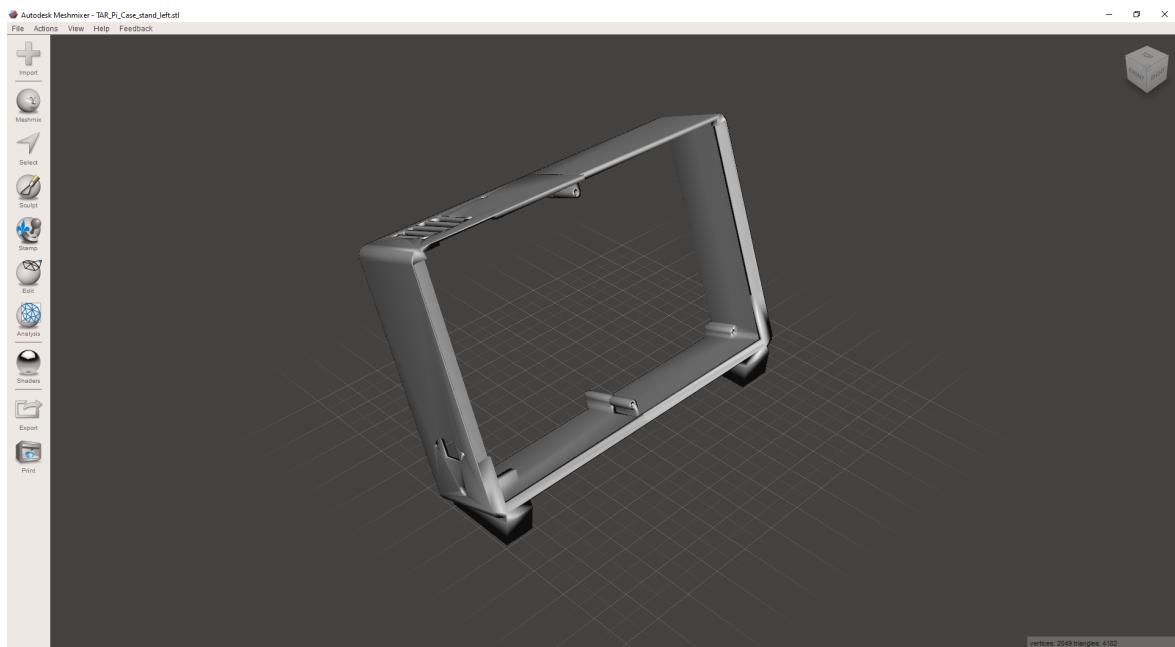


Abbildung 6: Gehäuse mit Füßen (Render in Meshmixer)

## 5.2 Kostenaufstellung

Nachfolgen haben wir die Kosten des Projekts aufgelistet (vgl. Tab ??: ??

### 5.2.1 Beschaffungskosten

Produkt	Menge	Kosten/Stk	Kosten/Gesamt
Raspberry Pi 4 Modell B 8GB	1	87,22€	87,22€
SanDisk Extreme microSD 128 GB	1	19,99€	19,99€
SUNFOUNDER RPi 10.1" Touch Display	1	129,99€	129,99€
Noctua NF-A4x10 5v	1	12,90€	12,90€
ITSTUFF CC2531 Zigbee USB-Stick	1	14,90€	14,90€
ITSTUFF Alu-Kühlkörper Set RPi 4	1	4,91€	4,91€
Eightwood SMA Verlängerung	1	7,99€	7,99€
VCE 5,5mm Stecker & Buchse	1	8,29€	8,29€
dasFilament PTGE schwarz 1,75mm	1	24,95€ <sup>10</sup>	11,61€ <sup>11</sup>
USB-Verlängerung	1	9,99€	9,99€
<b>Gesamtkosten</b>			<b>307,79€</b>

Tabelle 1: Kostenberechnung

### 5.2.2 Kostenberechnung für die 3D-Druckteile

Die Kosten lassen mit folgender Formel darstellen:

$$K_{gesamt} = (K_{Material/g} \cdot M_{Material}) + (T_{Druck} \cdot K_{kWh} \cdot V_{\emptyset/h})$$

Die Kosten der 3D-Druckteile lassen sich in zwei Einzelpositionen, Material und Energie. Die Materialkosten lassen sich leicht berechnen, da wir die Kosten der Spule Filament mit 24.95€ und einem Gewicht von 800g Filament auf der Spule. Dies bedeutet einen Preis von 0,02(74...)€ ( $K_{Material/g}$ ) pro Gramm verwendetem Filament. Der Druck der Gehäuseteile verbraucht 423g Material ( $M_{Material}$ ), dass heißt, dass die Materialkosten sich auf 11,61€ belaufen.

Die Energiekosten zu berechnen ist etwas komplexer. Der 3D-Drucker verbraucht circa 120W pro Stunde( $V_{\emptyset/h}$ )<sup>12</sup>, was bei einem Strompreis von 22,3€ct / kWh ( $K_{kWh}$ ) bedeuten würde, dass wir pro Stunde etwa 2,676€ct Stromkosten haben. Bei einer Gesamtdruckzeit von 36 Stunden und 36 Minuten( $T_{Druck}$ ) ergibt sich also Stromkosten von 97,9416€ct.

<sup>12</sup> Robert (2019): Antwort auf Creality Ender 3 printer power consumption? - 3dprinting Stack Exchange

Zählt man nun die Kosten für Material und Energie zusammen, ergibt sich ein Fertigungspreis von 12,59€ ( $K_{gesamt}$ ). Dabei ist die Veredelung (Schleifen & Lackieren) noch nicht mit einkalkuliert.

## 6 Hardware

### 6.1 Raspberry Pi 4

Als Basis für die Smart Home Zentrale haben wir einen Raspberry Pi in der Version 4 mit 8 GB gewählt, um als von anderen Servern unabhängige Plattform zu agieren. Der Raspberry Pi 4 ist mit einem ARM Cortex-A72 Prozessor ausgestattet, der über 4 Kerne verfügt. Darüber hinaus verfügt der Raspberry Pi 4 über einen Gigabit-Netzanschluss. Darüber hinaus ist der Raspberry Pi in der Bastlerszene weit verbreitet und dient bei anspruchsvoller Projekten als Kern

Der Raspberry Pi hat sich seit der ersten Veröffentlichung Anfang 2012 weltweit schon millionenfach verkauft und erfreut sich immer noch großer Beliebtheit. Denn viele Raspberry Pi User haben nicht nur einen Einplatinen-Computer zu Hause, sondern teils 4-5 Stück. Der eine fungiert als HD-Mediacenter mit externer Festplatte für das heimische Kino oder als Internetradio mit Display, der nächste als Webcam-Server für die Kameraüberwachung mit Livestream auf das Handy, dann noch einer für die Hausautomatisierung wie bspw. die Heizungs- oder Lichtsteuerung und noch einer als einfacher WLAN-Druckerserver oder als Mini-Computer zum allgemeinen Surfen im Internet, um ein paar wenige Anwendungsszenarien zu nennen. Sie merken, der kleine „Tausendsassa“ kann nicht nur viel, sondern ist zudem auch noch extrem günstig und eben das macht den Reiz aus. Lediglich den Hang zum Programmieren sollten Sie mitbringen und selbst nicht mal das, denn Sie können auch einfach nach Anleitung aus Foren oder Büchern nachprogrammieren, oder ganz bequem ein fertiges Image auf den Raspberry Pi installieren - trauen Sie sich! <sup>13</sup>

<sup>13</sup> [reichelt.de](http://reichelt.de): Produktbeschreibung des Raspberry Pi 4

## 6.2 Aufbau des Prototypen

Der Prototyp war eine Kombination aus dem verwendeten Touchscreen sowie dem Raspberry Pi 4 und dem Zigbee-Stick. Dieser Prototyp diente als Entwicklungsplattform für die Software.

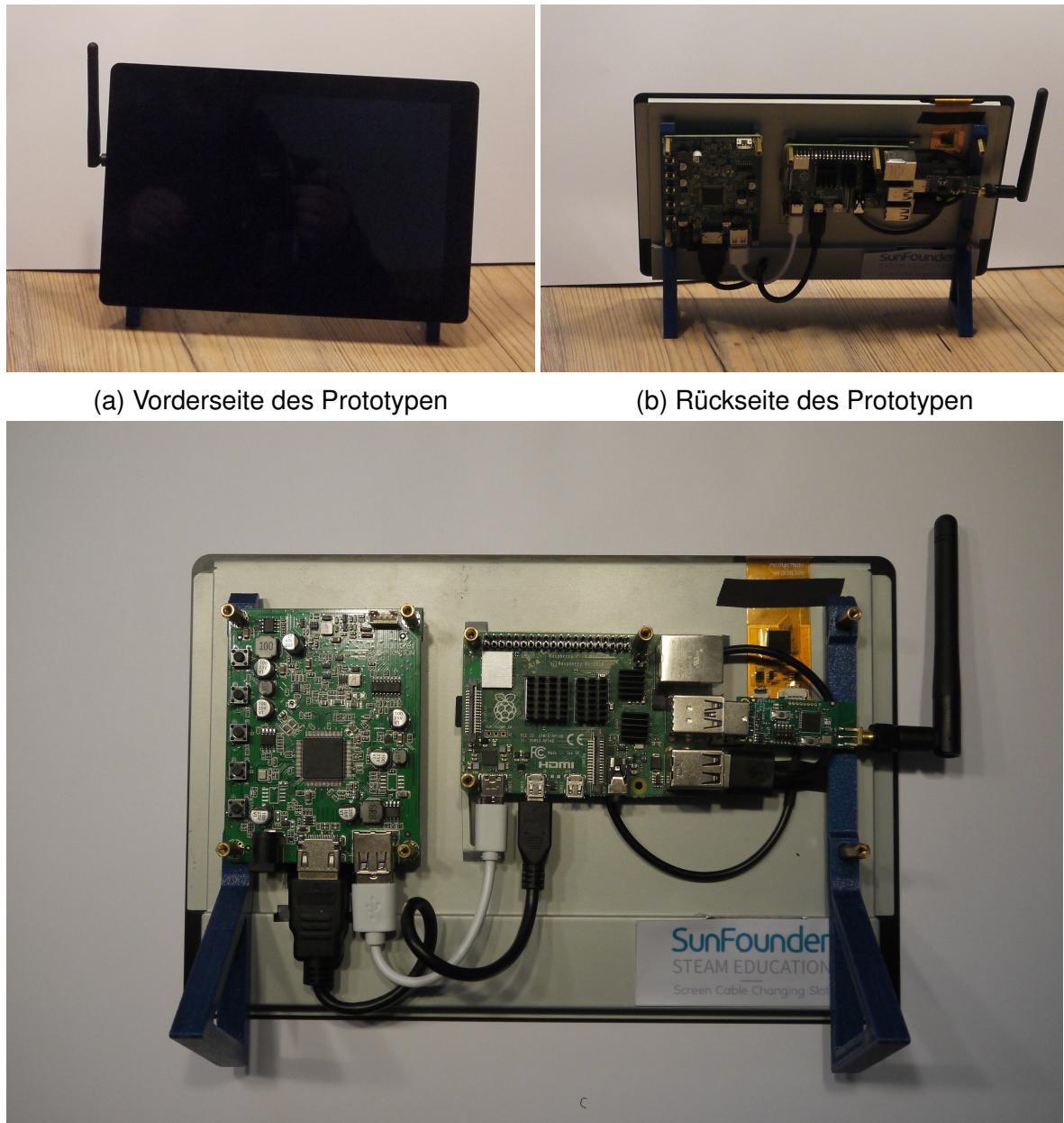


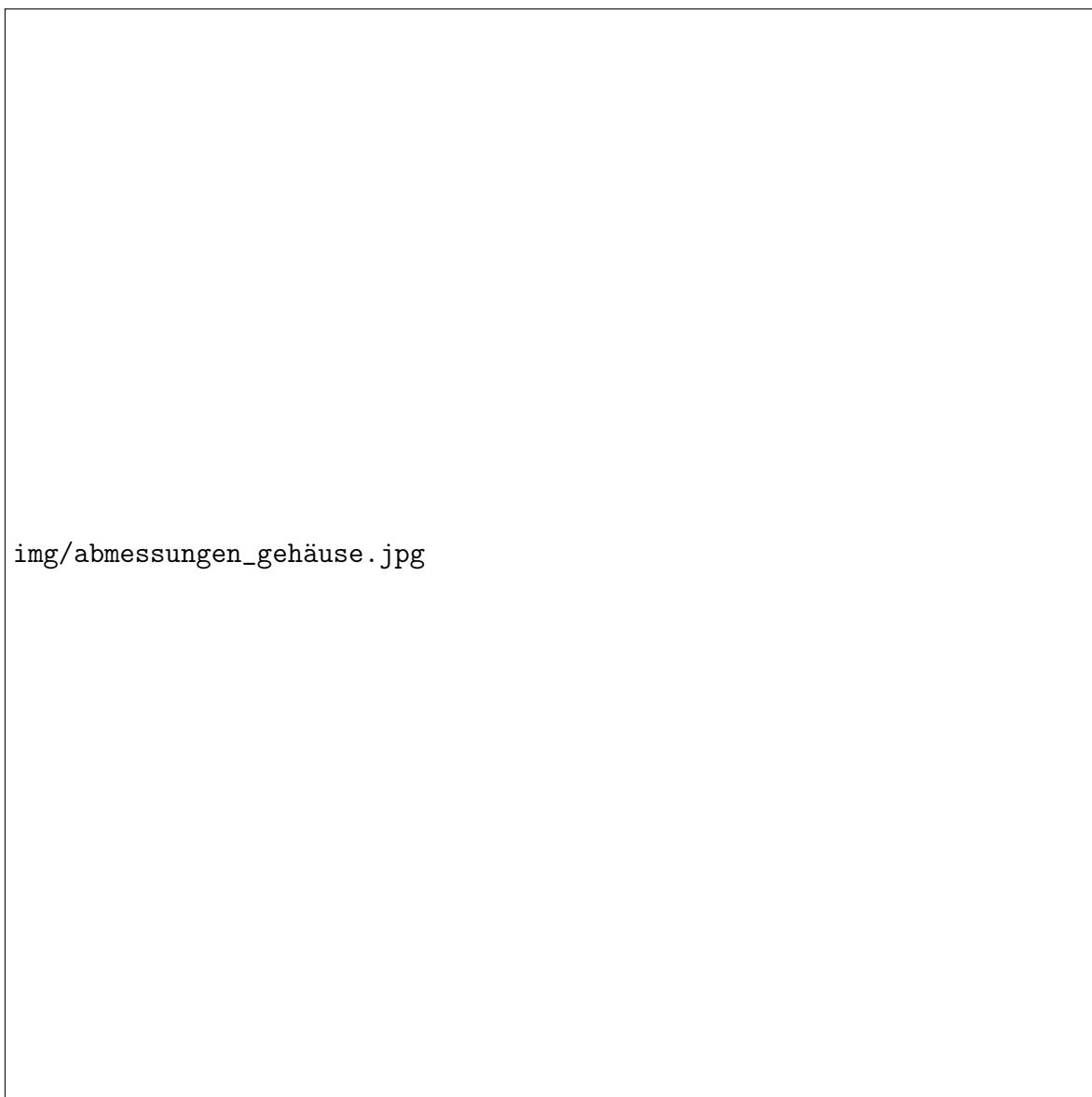
Abbildung 7: Prototyp der Smart Home Zentrale

## 6.3 Gehäuse

### 6.3.1 Erster Test des Gehäuses

Zur Erstellung des Gehäuses haben wir die Maße des Bildschirms als Anhaltspunkt genommen. Die erste Messung haben wir mit einem Meterstab durchgeführt. Der Bildschirm hatte an der Hinterseite eine Erhebung, weshalb wir diese ebenfalls ausgemessen haben. Die Maße beliefen sich nach der ersten Messung auf:

- 255,5 mm Breite
- 167 mm Höhe
- Kantenradius 10 mm



img/abmessungen\_gehäuse.jpg

Abbildung 8: Abmessungen Bildschirm Rückseite

Anhand dieser Maße haben wir dann in Fusion 360 eine Grundplanzeichnung erstellt und ein 3D-Modell gefertigt.

Nach Überprüfung der Maße haben wir dann allerdings festgestellt, dass der zur Verfügung stehende 3D-Drucker, ein Ender 3 Pro der Firma Creality3D, ein maximales Druckvolumen von 235x235x220mm besitzt und somit das Gehäuse nicht wie ursprünglich geplant aus einem Stück sondern in mehreren Teilen gedruckt werden muss. Hierfür haben wir eine Änderung der Konstruktion durchgeführt. Das Gehäuse besteht nun aus vier Teilen, zwei davon bilden jeweils die Seitenwände während zwei den Deckel des Gehäuses bilden. Die Teile werden mit langen M3 Senkkopfschrauben verbunden, die zusätzlich als Verschluss des Gehäuses dient.



img/druck\_gehäuse\_001.png

Abbildung 9: Platzierung der beiden Gehäusewände in CURA



img/druck\_gehäuse\_002.png

Abbildung 10: Verschiebung der Modelle entlang der Z-Achse um 45mm

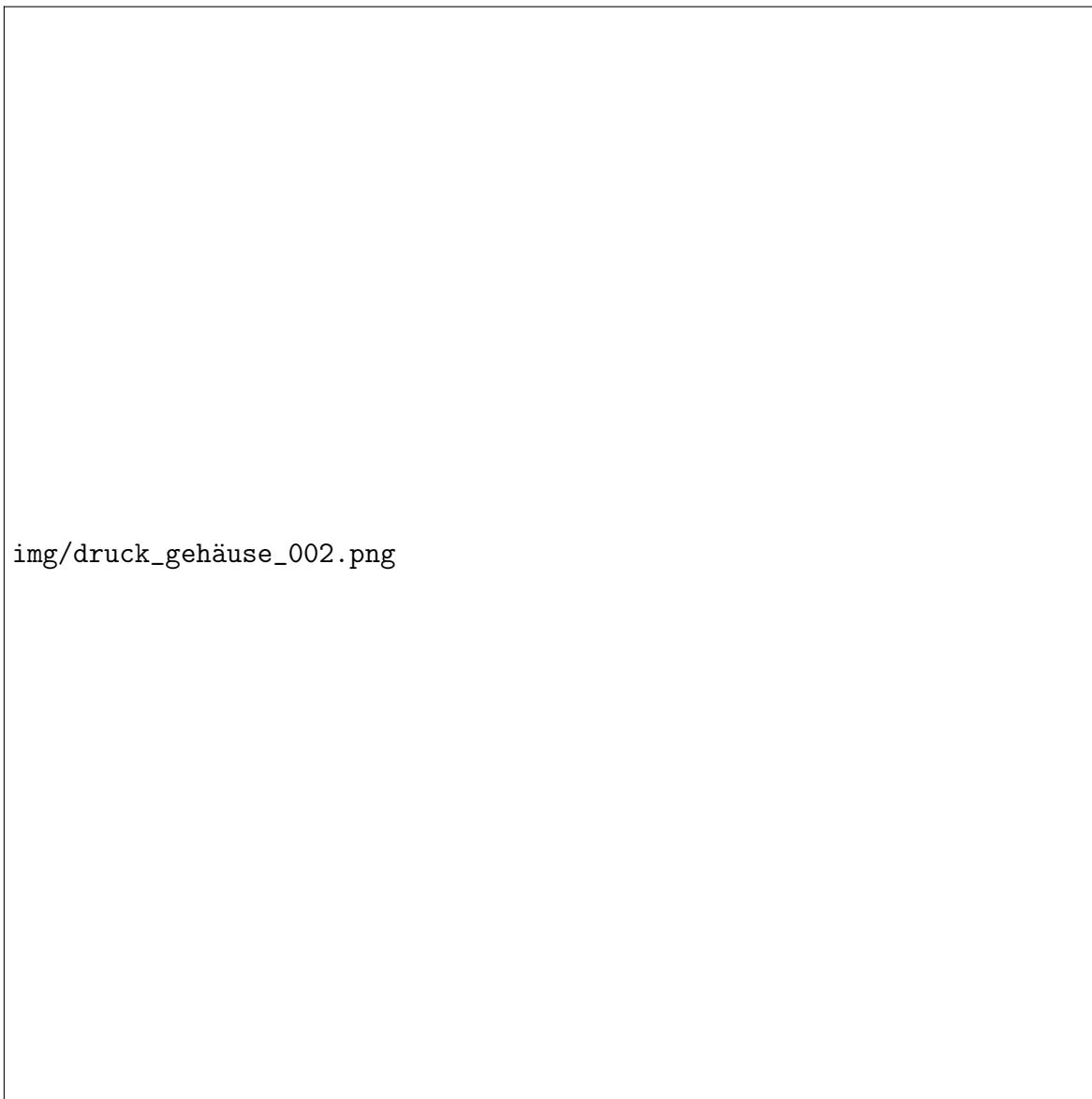


Abbildung 11: Slicen der Modelle



Abbildung 12: Passung des Testdrucks

Um die Genauigkeit der Konstruktion zu testen, haben wir eine 5 Millimeter hohe Schablone zum Testen ausgedruckt, die an den Bildschirm angelegt werden kann. Diese entstand mit Hilfe des Slicers CURA (vgl. Abb. ??: ??), in dem die Modelle der beiden Seitenteile so angeordnet wurden, dass lediglich 5 Millimeter des Teils im druckbaren Bereich des Druckers verblieben (vgl. Abb. ??: ??). Anschließend haben wir die Teile noch auf dem Druckbett zentriert (vgl. Abb. ??: ??).

Nach dem Ausdrucken war klar, dass die Maße des ersten Versuchs nicht genau genug waren (vgl. Abb. ??: ??, weshalb wir zum Ausmessen „genauere“ Messwerkzeuge zur Hand nahmen<sup>14</sup>

### 6.3.2 Modellentwicklung am Objekt

**Grundskizze** Nachdem das Testmodell (vgl. Abb. ??: ??) nicht zu 100 % gepasst hat, haben wir die Abmessungen neu geklärt und diese in Fusion 360 übertragen (vgl. Abb. ??: ??). Um Material für den 3D-Druck zu sparen, haben wir die Zeichnung dann im Maßstab 1:1 auf Papier gedruckt, ausgeschnitten und angelegt.

Da hier einige Maße immer noch nicht gestimmt haben, haben wir den Plan überarbeitet (vgl. Abb. ??: ??). Die so entstandenen Bemaßungen waren dann korrekt. Daraufhin wurde dann die Zeichnung in zwei eigenständige Dateien gesplittet, um die linke und die rechte Seite des Gehäuses zu konstruieren.

<sup>14</sup> Digitaler Messschieber mit einer Nachkommastelle & einer Genauigkeit von  $\pm 0.2\text{mm}$

Die Grundabmessung des Bildschirms werden in Fusion 360 als neue Zeichnung angelegt. Hierzu haben wir ein einfaches Rechteck mit den entsprechenden Außenmaßen gezeichnet (vgl. Abb. ??: ??). Anschließend haben wir die Ecken mit einem Radius von 7mm abgerundet (vgl. Abb. ??: ??) und die Zeichnung dann in der Mitte geteilt, um die beiden Hälften des Gehäuses unabhängig von einander konstruieren zu können.

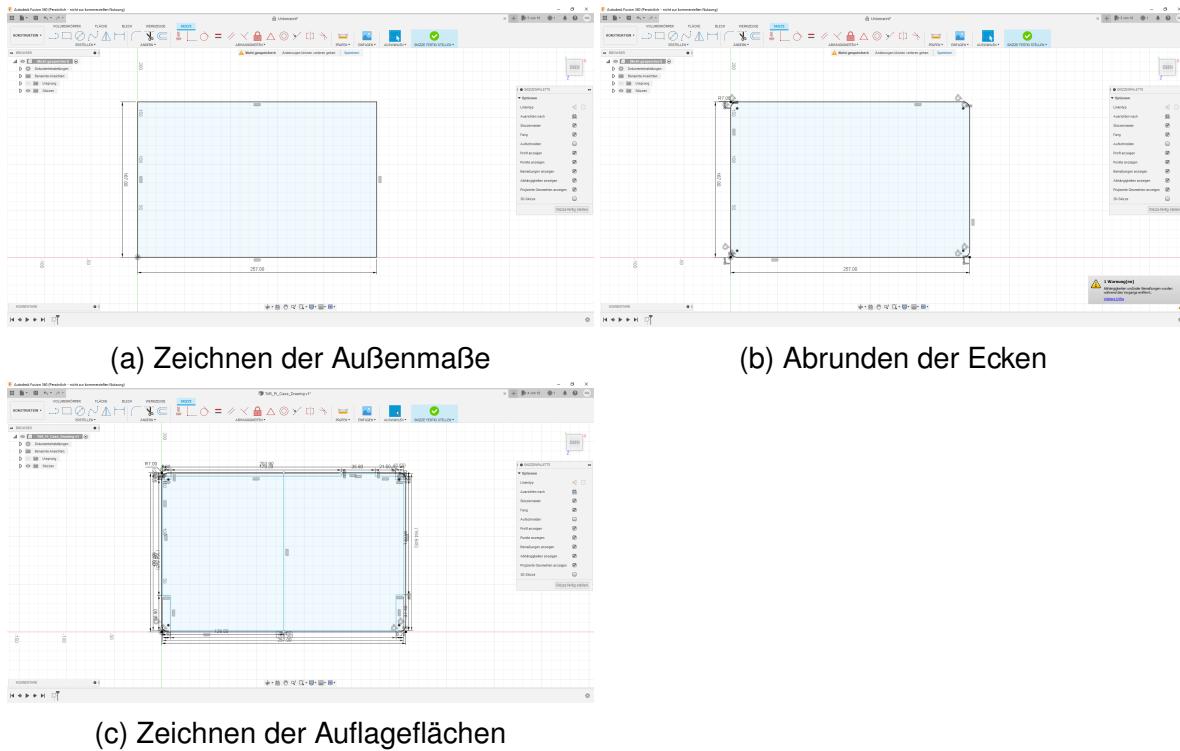


Abbildung 13: Grundzeichnung als Basis des Modells

Die so entstandene Zeichnung (vgl. Abb. ??: ??) haben wir dann in eine weitere Datei kopiert, die so als Grundlage für die beiden geteilten Seitenteile dienen. Von hier aus haben wir die beiden Wandteile mehr oder weniger unabhängig voneinander entworfen.

**Linkes Wandteil** Beim linken Wandteil haben wir den entsprechende Teil der Zeichnung um 2 mm entlang der Z-Achse extrudiert (vgl. Abb. ??: ??). Auf der erhöhten Seite haben wir eine weitere Zeichnung gelegt, die die Bleche an der Rückseite des Bildschirms überdecken soll. (vgl. Abb. ??: ??), die wir dann wie zuvor um 3 mm entlang der Z-Achse extrudiert haben (vgl. Abb. ??: ??). Diese sollte dem Gehäuse genug Auflagefläche an dem Bildschirm bieten, um die Verklebung so stark wie möglich zu machen. Auf die nun entstandene erhöhte Seite haben wir eine Zeichnung der „tatsächlichen“ Wandstärke von 3 mm erstellt (vgl. Abb. ??: ??). Diese haben wir dann um 6 mm entlang der Z-Achse extrudiert, um für die Verbinder genügend Platz vor dem klobigen Blechbereich im unteren Teil des Bildschirms zu bieten (vgl. Abb. ??: ??). Auf die nun obenliegende Seite haben wir die Zeichnung der Verbindungsstücke gelegt (vgl. Abbildung ??: ??) und um 19 mm entlang der Z-Achse extrudiert (vgl. Abb. ??: ??) und im Kreismittelpunkt der Zeichnungen eine Bohrung für M3-Gewinde gesetzt (vgl. Abb. ??: ??). Auf die nun entstandene

Oberfläche haben wir die Zeichnung für die Verbindung mit dem Deckel gesetzt (vgl. Abb. ??: ??), um 25 mm entlang der Z-Achse extrudiert (vgl. Abb. ??: ??) und ebenfalls mit Bohrungen für M3-Gewinde versehen (vgl. Abb. ??: ??).

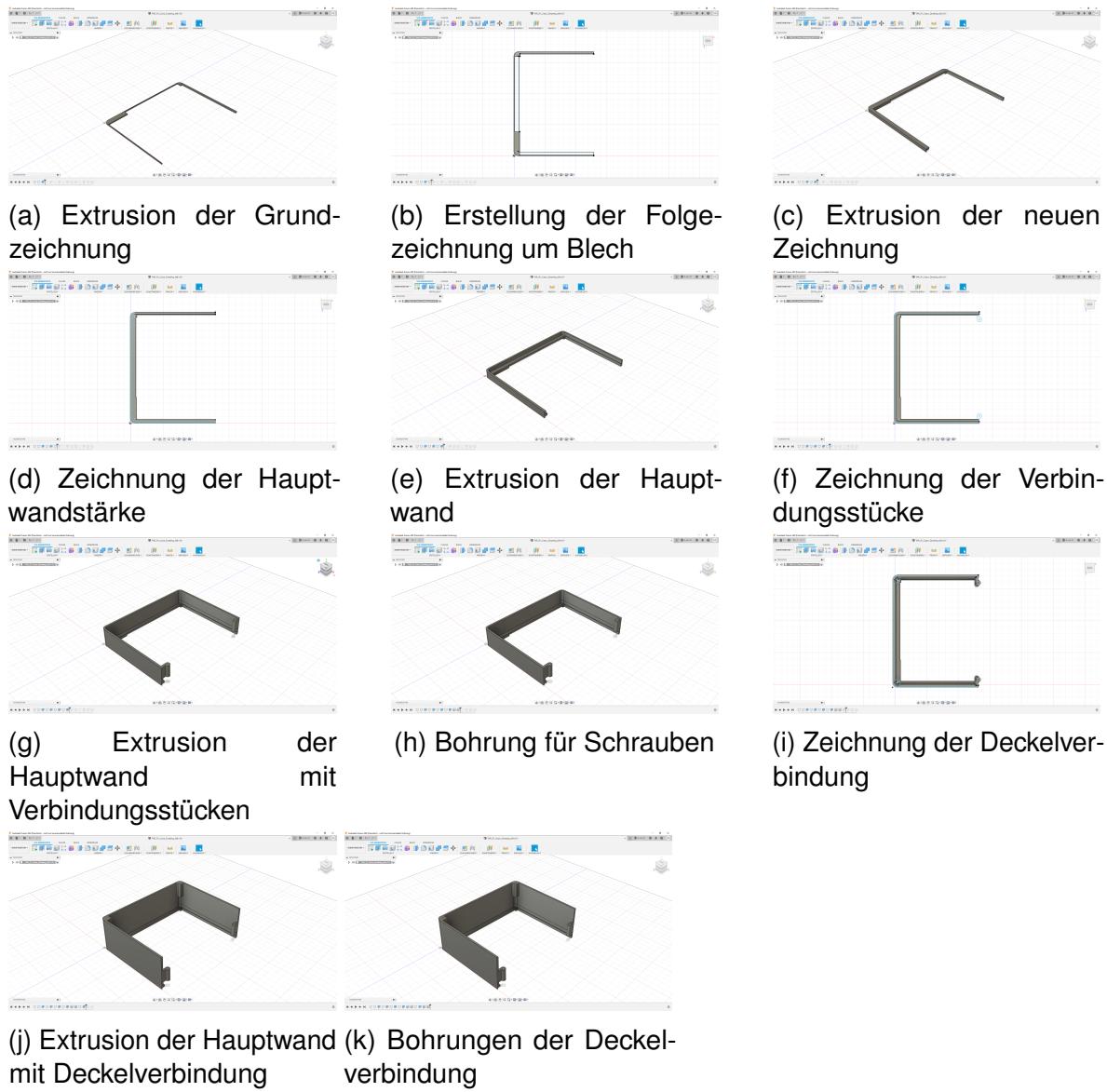


Abbildung 14: Entwurf des linken Wandteils

**Rechtes Wandteil** Beim rechten Teil des Gehäuses ist die Vorgehensweise weitestgehend die selbe wie beim linken Gehäuseteil (vgl. Abb. ??: ??- Abb. ??: ?? mit Abb. ??: ??). Der Unterschied zwischen beiden Teilen, abgesehen von den Verbindungsstücken und der Aussparung für das Flachkabel des Bildschirms, sind die Aussparungen für Lüfter, Antenne, Netzwerkanschluss und Strombuchse. Für den Lüfter und die Antenne haben wir auf der „Oberseite“ des Gehäuses eine Zeichnung aufgelegt (vgl. Abb. ??: ??), die wir dann ins Negative extrudiert haben, was die gezeichnete Fläche aus dem Körper löscht (vgl. Abb. ??: ??). Auf ähnliche Weise haben wir die Aussparung für eine RJ45-Verlängerung und eine 5,5 mm DC-Buchse gesetzt. Zuerst haben wir die Zeichnung für die Aussparung auf die Seite gelegt (vgl. Abb. ??: ??) und diese dann ebenfalls ins Negative extrudiert (vgl. Abb. ??: ??). Um für die Befestigung der 5,5 mm DC-Buchse eine Unterlage im Gehäuse zu haben, haben wir der Grundriss eines Rechtecks auf die Innenseite des Gehäusebodens gelegt (vgl. Abb. ??: ??) und extrudiert (vgl. Abb. ??: ??), um die Möglichkeit zu haben, die Buchse im Gehäuse zu verkleben. Damit die Buchse nicht zu tief im Gehäuse steckt, haben wir eine Zeichnung eines konzentrischen Kreises auf die bereits vorhandene Aussparung auf die Innenseite gelegt (vgl. Abb. ??: ??) und dann um einige Millimeter ins Negative extrudiert, um die Aussparung zu generieren (vgl. Abb. ??: ??).

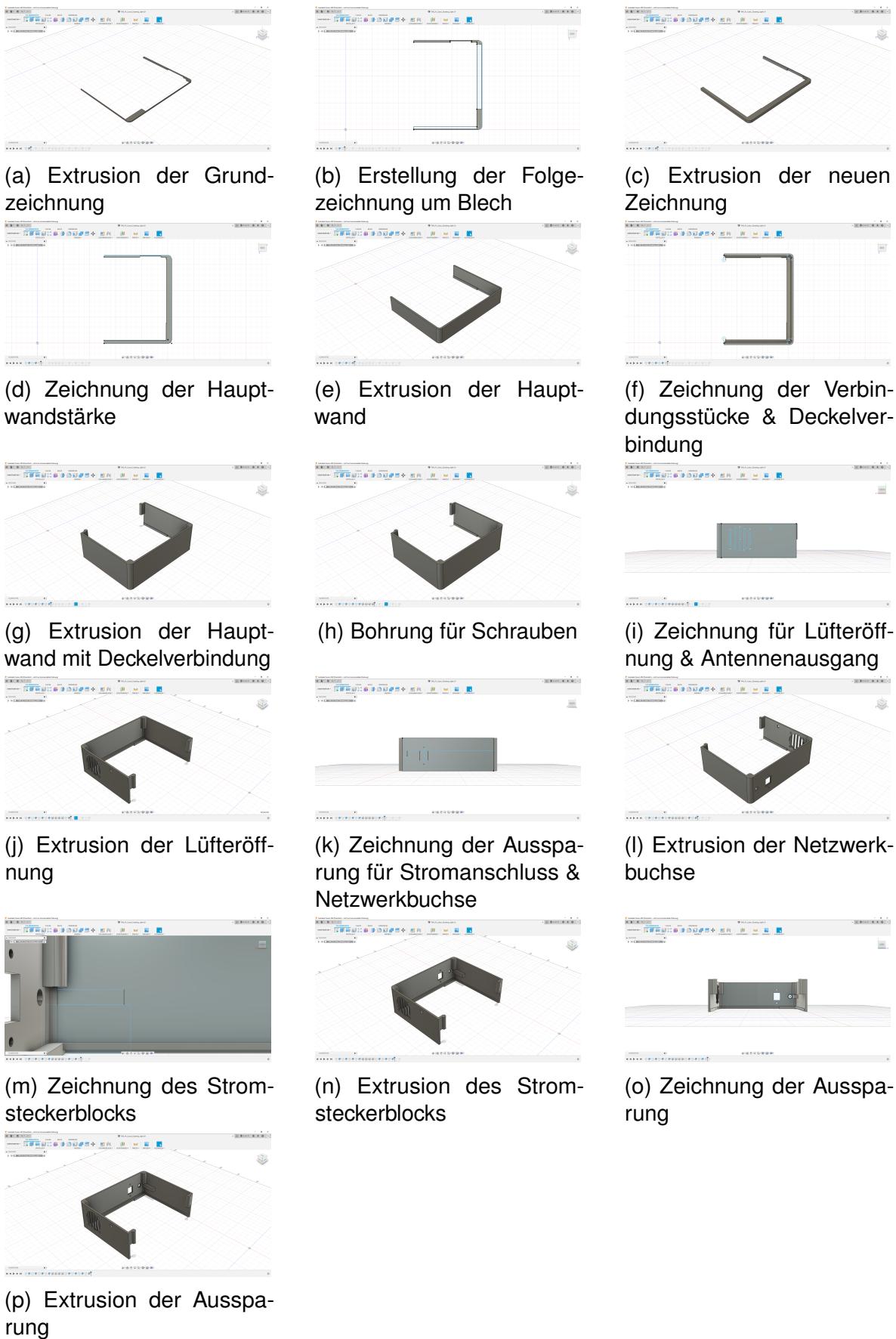


Abbildung 15: Entwurf des rechten Wandteils

**Gehäuserückwand** Das Design der Gehäuserückwand basiert nur rudimentär auf der in Abb. ??: ?? erstellten Zeichnung. Die Außenmaße stimmen zwar überein, die Zeichnung für die Auflage auf den Seitenwänden haben wir aber neu erstellt. Zusätzlich haben wir eine Diagonale zur punktsymmetrischen Trennung der Teile eingefügt (vgl. Abb. ??: ??), welche die Zeit für die Konstruktion von zwei unterschiedlichen Teilen für die Rückseite reduzieren soll. Die Zeichnung haben wir dann entsprechend extrudiert (vgl. Abb. ??: ??). Um die Teile nach dem Druck besser verbinden zu können, haben wir ein Vorsprung auf der kurzen Seite des Bauteils extrudiert (vgl. Abb. ??: ??). Um beim Druck Material zu sparen, haben wir ein weiter Teil der inneren Zeichnung ins Negative extrudiert, um den Bereich freizustellen (vgl. Abb. ??: ??). Für eventuelle Toleranzen zwischen den beiden Teilen haben wir auf dem Vorsprung die Aussparung für die unterliegende Schraubendurchführung erhöht (vgl. Abb. ??: ??). Zusätzlich haben wir den Vorsprung von unten mit einer Fase versehen, um den Materialverbrauch für das Teil weiter zu reduzieren (vgl. Abb. ??: ??). Des weiteren haben wir die obere Kante (vgl. Abb. ??: ??) und die Innenseite (vgl. Abb. ??: ??) mit einer Fase versehen. Eine ähnliche Fase haben wir auch an dem Vorsprung angebracht (vgl. Abbildung ??: ??). Um die Bohrung am die richtige Stelle zu setzen, haben wir eine Hilfszeichnung auf die Außenfläche gesetzt (vgl. Abb. ??: ??). Anschließend haben wir drei M3-Bohrungen für Senkkopfschrauben gesetzt (vgl. Abb. ??: ??). Um das Gehäuse an der Wand anbringen zu können, haben wir eine Zeichnung auf die Oberseite des Gehäuses gelegt (vgl. Abb. ??: ??) und dann ins Negativ extrudiert (vgl. Abb. ??: ??). Um bei der Anbringung nicht an einen bestimmten Typ Schrauben gebunden zu sein, haben wir eine kleine Fase an den engen Teil der Aufhängungen gelegt (vgl. Abb. ??: ??).

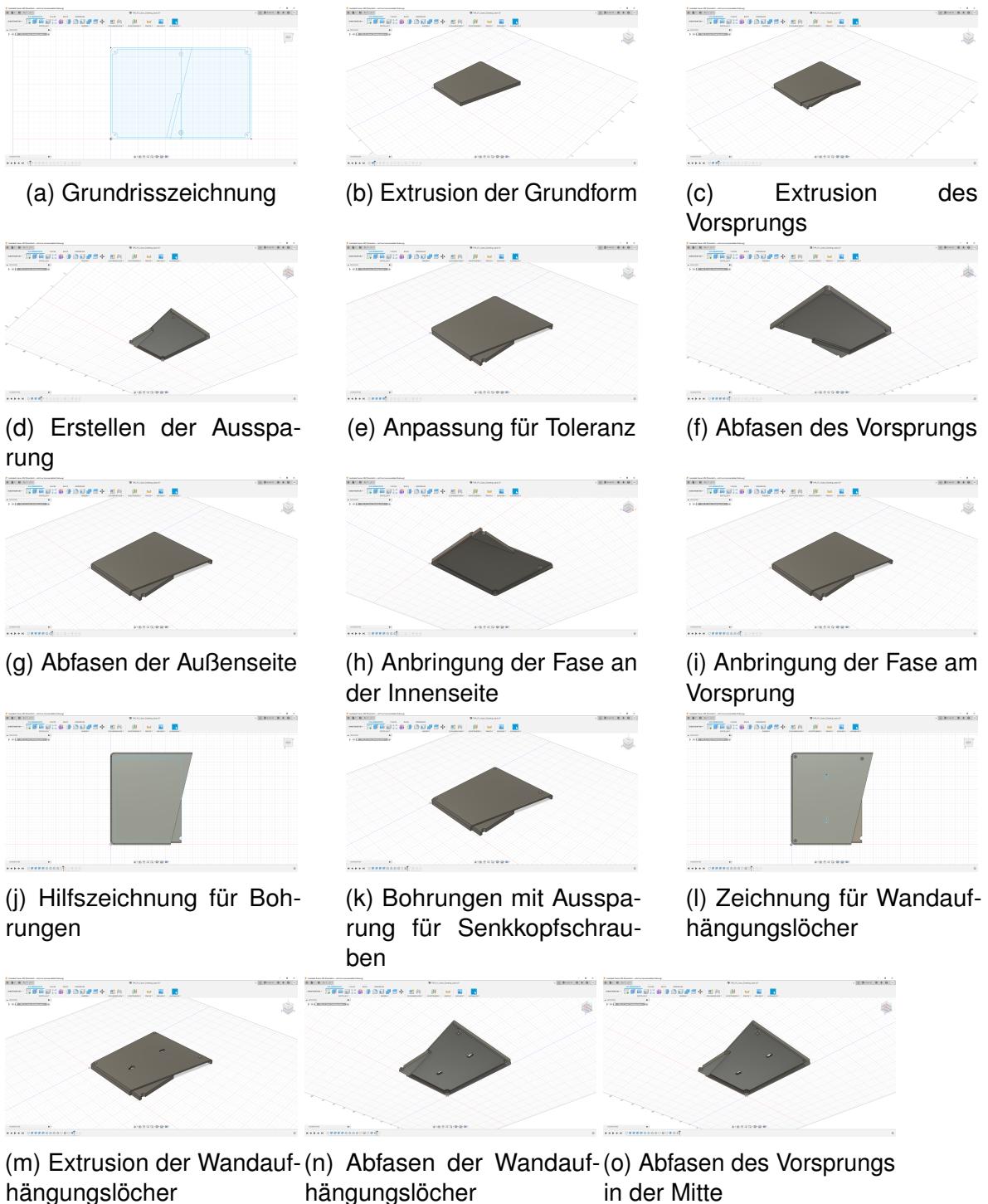


Abbildung 16: Entwurf der Gehäuserückwand

### 6.3.3 Herstellung des Gehäuses

Zur Herstellung des Gehäuses kommt ein 3D-Drucker der Marke Creality 3D zum Einsatz. Der stark modifizierte Ender 3 Pro von Manuel Starz (vgl. Abb. ??: ??) druckt die zuvor erstellten STL-Dateien mit PETG-Filament der Firma dasfilament. Der für den 3D-Druck nötige G-Code wird mit Ultimaker CURA generiert und mit Hilfe von OctoPrint (vgl. Abb. ??: ??) an den Drucker übertragen. Das Gehäuse besteht aus vier Einzelteilen, je zwei Teile für die Seitenwände und zwei Teile für den Deckel.



Abbildung 17: 3D-Drucker Ender 3 Pro (mod)

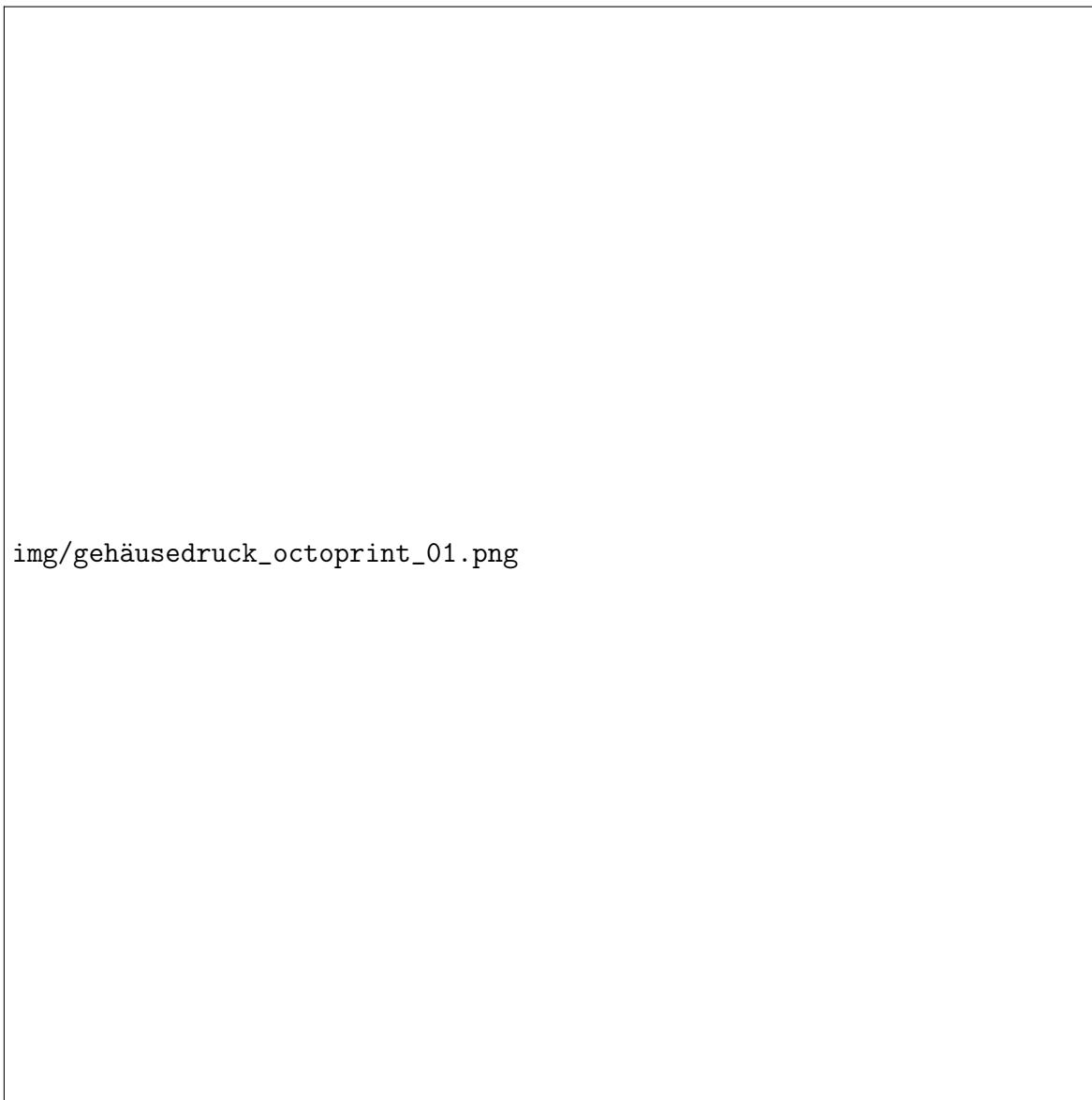
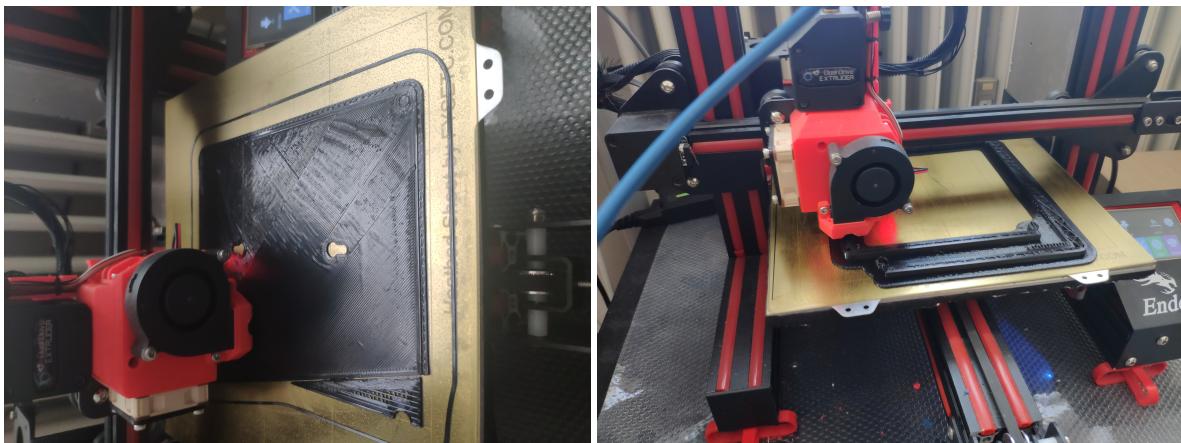


Abbildung 18: OctoPrint-Weboberfläche

Die gedruckten Teile (vgl. Abb. ??: ??) werden dann zum Teil mit Zwei-Komponenten-Epoxidkleber verbunden (vgl. Abb. ??: ??) und mit Hilfe von Schleifpapier und einigen Schichten Klarlack zu einem klavierlackähnlichen Finish veredelt (vgl. Abb. ??: ??). Die Seitenwände werden dann mit dem Bildschirm mit Hilfe des Zwei-Komponenten-Epoxidklebers und Heißkleber permanent verklebt (vgl. Abb. ??: ??). Danach wurden die Platinen und Kabel im Gehäuse angebracht (vgl. Abb. ??: ??). Die Rückseite besteht der Einfachheit halber aus zwei identischen, punktsymmetrischen Teilen, die ebenfalls miteinander verklebt wurden.



(a) Druck eines der Deckel-Teile

(b) Druck der Wand-Teile

Abbildung 19: Das Gehäuse entsteht



(a) Rückseite

(b) Linke Seite

(c) Rechte Seite

Abbildung 20: Ausgedruckte Teile des Gehäuses



Abbildung 21: Fertigung des Gehäuses

Laut Ultimaker CURA beträgt die Gesamtdruckdauer des Gehäuses 36 Stunden 36 Minuten und verbraucht insgesamt 423 Gramm des verwendeten Filaments (dasFilament PETG Schwarz). Damit belaufen sich die Materialkosten des Gehäuses auf 11,61€<sup>15</sup>.

Bei der Herstellung ist es aber zu einem Problem gekommen. Die Stützstrukturen haben den Druck am Lüftereinlass gestört, was dazu führt, dass das Lüftergitter an der linken Seitenwand (vgl. Abb. ??: ??) nicht richtig gedruckt wurde und die losen Filamentstücke zu Macken an den Gehäusewänden führen.

## 6.4 Erstellung des RPi-HATs

Zur Schaltplanzeichnung haben wir das OpenSource-Programm KiCAD genutzt. Dieses ist für zahlreiche Betriebssysteme verfügbar.<sup>16</sup> Teilebibliotheken sind zahlreich im Internet verfügbar, wir haben uns auf den Anbieter SnapEDA beschränkt, da dieser die meisten gängigen Bauteile für Elektronik-CAD-Systeme anbietet. Eine weitere Bezugsquelle ist der Verkäufer Digi-Key Electronics, der einen Großteil seines Sortiments an SMD-Bauteilen als Bibliothek anbietet<sup>17</sup>. Als Grundlage für den HAT sollte die Projektvorlage „Raspberry Pi - 40-Pin HAT“ von Jon Buford dienen. Diese

<sup>15</sup> Genauere Kostenrechnung siehe Abschnitt ??: ??

<sup>16</sup> Siehe <https://www.kicad.org/download/>

<sup>17</sup> Siehe <https://www.digikey.de/de/resources/design-tools/kicad>

Vorlage richtete sich nach den offiziellen HAT-Spezifikationen für den Raspberry Pi<sup>18</sup>. Zu Beginn des Projekts haben wir in KiCAD die GPIO-Ports anhand der offiziellen Dokumentation beschriftet, um das Nachschlagen der Belegung in Zukunft zu vermeiden.

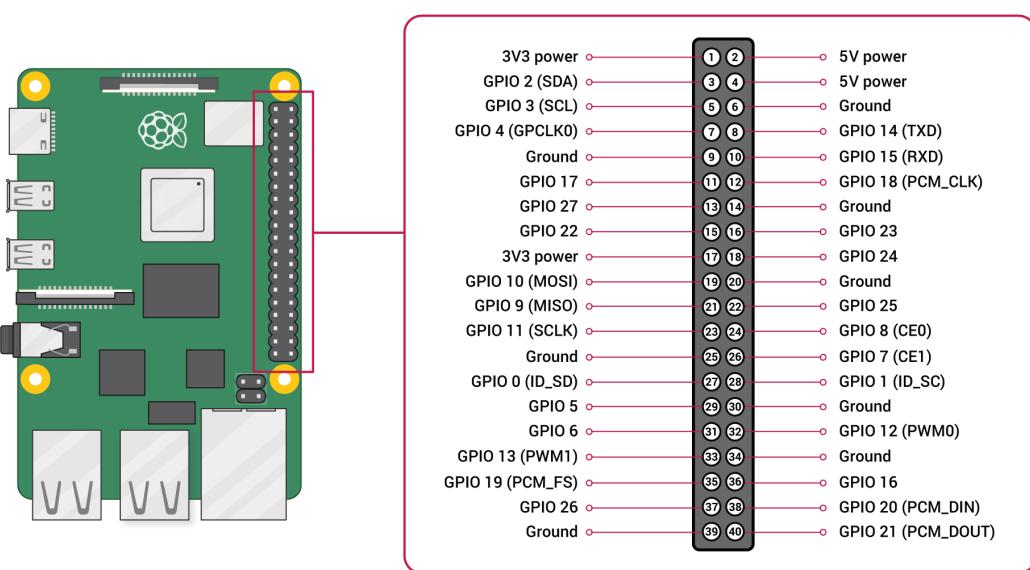


Abbildung 22: Raspberry Pi 4 GPIO-Pins

Anschließend haben wir die (unserer Meinung nach) benötigten Komponenten auf den Schaltplan gezogen beziehungsweise vorher importiert und dann versucht, diese sauber zu verdrahten (vgl. Abb. ??: ??).

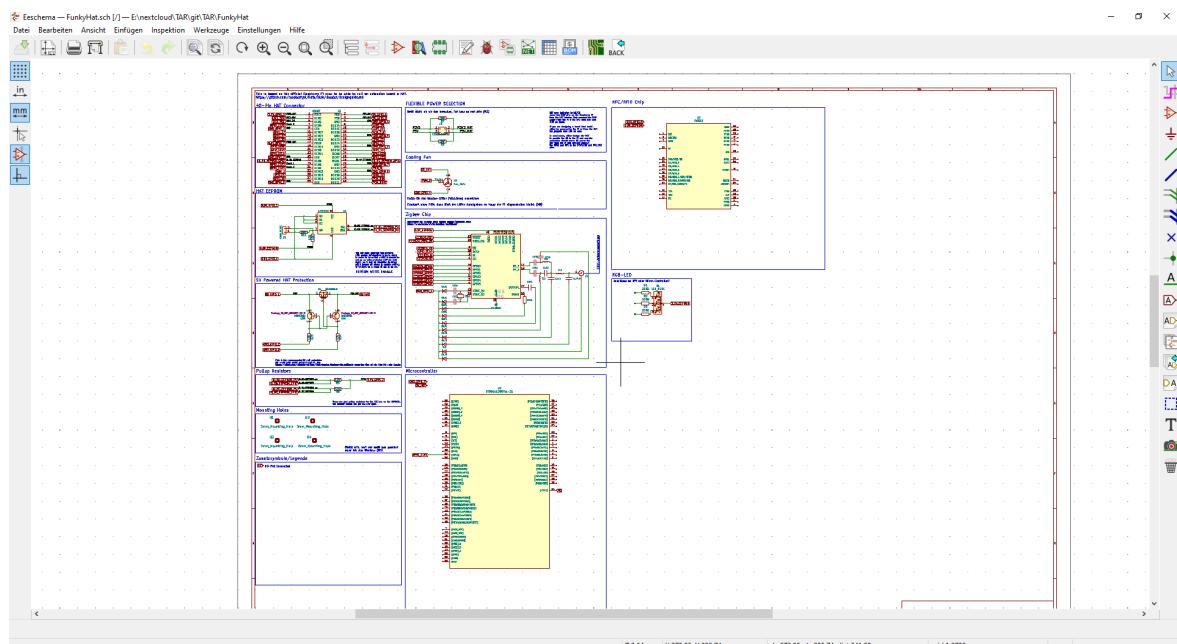


Abbildung 23: Aktueller Stand HAT-Design

<sup>18</sup> Siehe <https://github.com/raspberrypi/hats>

Aufgrund der bereits weit vorangeschrittenen Zeit im Projektplan, der aktuellen COVID-19-Pandemie und unserer Unwissenheit im Bezug auf Schaltplanentwicklung sowie der von uns verwendeten Hardware mussten wir aber einsehen, dass die Entwicklung und Fertigung eines HATs den Zeitrahmen sprengen würde und so haben wir zu Ende Februar 2021 beschlossen, dieses Projekt vorerst zurück zu stellen und uns auf die Fertstellung unserer bisherigen Testerfolge und Prototypen zu fokussieren. Die KiCAD-Projekt-Dateien stehen, wie alle anderen Dateien des Projekts nach Abgabe frei über das entsprechende GITHUB zur Verfügung.

## 6.5 Präsentationsaufbau



Abbildung 24: Präsentationsaufbau

Um unsere Technikerarbeit präsentieren zu können, haben wir einen Präsentationsaufbau aus XXX gebaut (vgl. Abb. ??: ??). Dieser ist neben der Smart Home Zentrale auch mit einem Heimrouter und einer Zigbee-fähigen Glübirne ausgestattet. Zusätzlich haben wir im Netz des Heimrouters ein Tablet eingebunden, welches ebenfalls auf die Weboberfläche der Smart Home Zentrale zugreifen kann (vgl. Abb. ??: ??).

# Platzhalter-Bild

Abbildung 25: Plandiagramm des Aufbaus

## 7 Software

### 7.1 Überblick

Auf der Softwareseite dient Raspberry Pi OS<sup>19</sup> als Basis für die Smart Home Zentrale. Darauf laufen die gewählten Softwarelösungen

- Home Assistant Supervised (Web Oberfläche und Verwaltung)
- Mosquitto Broker (MQTT Broker)
- Zigbee2mqtt (Zigbee-Schnittstelle)

Zur Installation der einzelnen Komponenten und für die Vorkonfiguration des Systems haben wir ein bash-Skript (vgl. ??) erstellt, dass das System entsprechend vorbereitet und Home Assistant Supervised installiert. Wir haben uns für die oben genannte Implementierung von MQTT und Zigbee2MQTT entschieden da unsere Versuche den Mosquitto Broker sowie Zigbee2MQTT als separate Docker Container zu installieren zwar erfolgreich waren allerdings die Ergebnisse nicht zuverlässig reproduzierbar waren. Eine Installation Beider Komponenten via Home Assistant Supervisor Add-on Shop stellt eine Zuverlässige Methode dar und unterscheidet sich hinsichtlich der Funktionalität nicht.

### 7.2 Installationsskript

Das Installationsskript (vgl. ??) dient dazu, die im Projekt verwendeten Softwarekomponenten möglichst ohne großen Aufwand und mit so wenig Nutzereingaben wie möglich zu installieren und einzurichten. Da auf den Raspberry Pis das Debian-Derivat Raspberry Pi OS läuft, handelt es sich beim Installationsskript um ein BASH-Skript. Da das Installationsskript des Home Assistant Supervisors viele Ausgaben sowie eine nicht mit einem Parameter automatisierbare Abfrage besitzt, haben wir das Skript in unser Installationsskript kopiert und die Abfrage entsprechend überbrückt und die Ausgaben in die von unserem Skript genutzte Log-Datei umgeleitet. Dadurch bekommt der Nutzer im Idealfall lediglich kurze Statusmeldungen über den aktuellen Stand und bei Abschluss der Installation die Erfolgsmeldung. Die einzelnen Aufgaben, die das Installationsskript durchführt, wurden in eigene Funktionen ausgegliedert. Dadurch lassen sich am Skript selbst schneller Änderungen durchführen. Dabei erkennt das Skript über Operatoren, welche Schritte durchgeführt werden sollen und kann so auf neu aufgesetzten als auch auf bereits länger im Benutzung befindlichen Raspberry Pis ausgeführt werden. So besteht die Möglichkeit, beispielsweise die Installation von Docker oder dem Systemupdate nicht zu wählen (vgl. ?? Zeile 34 ff). Gibt der Nutzer keinen Operanden oder einen ungültigen Operanden an, wird ein Hilfetext mit der Funktion `show_help` (?? Zeile 34ff) ausgegeben, der den Nutzer über die Operatoren sowie eine allgemeine Verwendung des Skripts informiert.

Möchte der Nutzer das komplette Smart Home Zentralen Paket selbst installieren, ist der Operand `-a` an den Befehl `./sh_install` anzuhängen. Dadurch wird die Installation vollständig durchgeführt, das heißt, dass ein Systemupdate durchgeführt wird, die Abhängigkeiten Network Manager, AppArmor und JQ, sowie Docker und der

<sup>19</sup> Raspberry Pi OS with Desktop, Kernel Version 5.4

Docker-Container Home Assistant Supervised für Raspberry Pi installiert werden. Möchte der Nutzer die Installation ohne die Ausführung eines vorhergehenden Systemupdates und ohne die Installation von Docker durchführen, kann er das Skript mit dem Operand -h starten, was lediglich den Home Assistant Supervised nach dem offiziellen Skript für die Installation<sup>20</sup> durchführt.

Ob die Abhängigkeiten vorhanden sind, wird bei der Installation unabhängig von der Nutzereingabe geprüft und gegebenenfalls bei Fehlen nachinstalliert (?? Zeile 85ff & ?? Zeile 297ff).

Sollte Das Skript, ohne das Docker auf dem Raspberry Pi installiert ist, ausgeführt werden und nur die Installation des Home Assistant Supervised angefordert werden, wird Docker mit installiert (?? Zeile 284ff). Diese und andere Fehlervermeidungen übernimmt die Funktion preflightcheck, die eventuell vom Nutzer falsch gesetzte Parameter sauber setzt. Die Installation erfolgt weitestgehend über den internen Paketmanager apt<sup>21</sup>. Für Docker wird allerdings ein Skript von docker.com heruntergeladen und ausgeführt (vgl. Abschnitt ??: ?? Zeile 108ff). Der Abschnitt der Home Assistant Installation basiert auf dem Bash-Skript von Home Assistant selbst<sup>22</sup>, wir haben aber die Terminal-Ausgaben des Skripts in unsere Log-Datei umgeleitet und die Abfrage zur Neukonfiguration des Network-Managers unterdrückt. Lediglich die Ausgaben des Docker-Installationsskripts erscheinen nun noch zusätzlich zu den Statusmeldungen unseres Skripts auf dem Bildschirm.

<sup>20</sup> GITHUB: supervised-installer <https://github.com/home-assistant/supervised-installer>

<sup>21</sup> Für das Systemupdate und die Installation der Abhängigkeiten.

<sup>22</sup> GITHUB: installer.sh <https://github.com/home-assistant/supervised-installer/blob/master/installer.sh>

Eine Übersicht und Kurzerklärung der Funktionen im Skript:

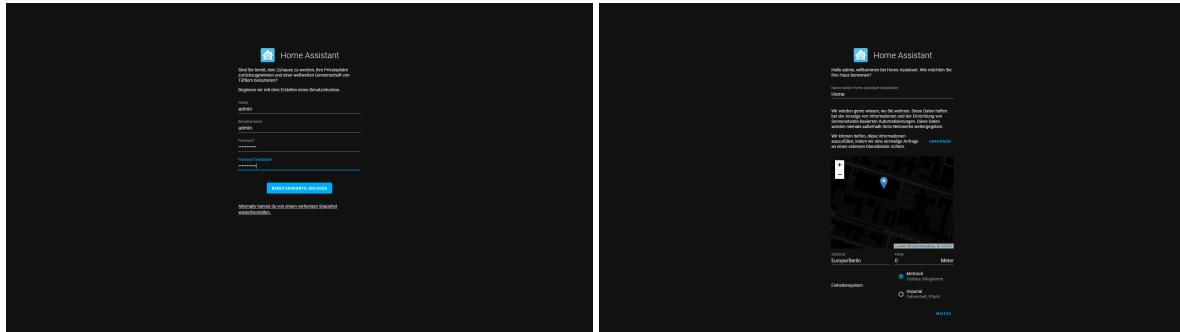
Funktion	Kurzbeschreibung	Zeile
show_help	Anzeige der Hilfe im Terminal.	34
get_time	Aktualisiert die Variable „TIMESTAMP“.	56
run_update	Führt apt-get update, apt-get upgrade & apt-get autoremove mit automatischer Bestätigung aus.	60
get_piver	Ermittelt die Version des Raspberry Pis und bricht bei unbekannter Version ab.	68
install_dependencies	Installiert Network-Manager, AppArmor und JQ bei Bedarf.	85
install_docker	Lädt das Installationsskript von Docker herunter und führt dieses aus.	105
install_homeassistant	Führt Systemvorbereitung und die Installation von Homeassistant durch.	125
preflightcheck	Überprüft das System, ob die Abhängigkeiten installiert sind und ob Docker oder Home Assistant bereits installiert sind.	277
set_all	Setzt die Variablen für eine Komlettinstallation	345
set_docker	Setzt die Variablen für eine Installation von Docker	352
set_homeassistant	Setzt die Variablen für eine Installation von Home Assistant	356
set_update	Setzt die Variablen für ein Systemupdate	360

Tabelle 2: Skript-Funktionen

### 7.3 Home Assistant Supervised

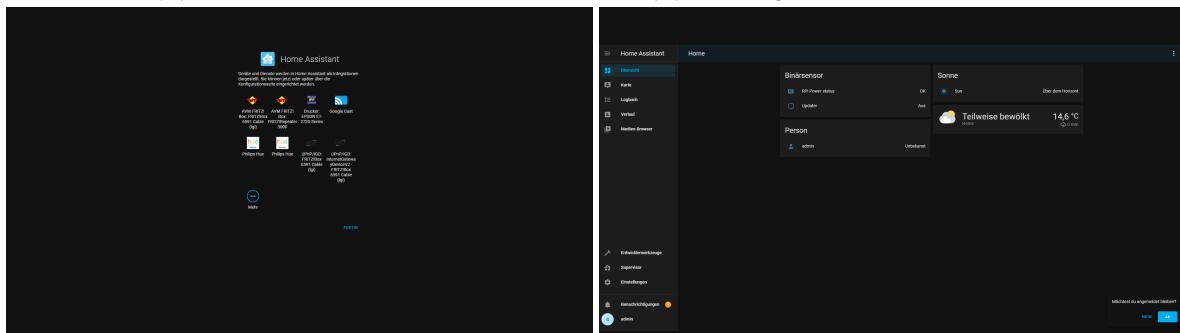
Nachdem Das Installationsskript (vgl. Abschnitt ??: ??) fehlerfrei Home Assistant Supervised installiert hat, können wir mit der Einrichtung fortfahren. Dafür öffnen wir, im selben Netzwerk wie unser Raspberry Pi, die Webseite <http://<raspberry-pi-adresse>:8123> mit einem beliebigen Browser.

Hier sehen wir einen Anmeldebildschirm (vlg. Abb. ??: ??) in welchen wir unseren Benutzeraccount mit Passwort für Home Assistant anlegen (vgl. Abb. ??: ??). Anschließend legen wir den Namen und den Standort des Home Assistant fest (vgl. Abb. ??: ??). Der Standort dient der die Ermittlung von Wetterdaten und wird für das Geofencing benötigt. Auf dem Nächsten Bildschirm (vgl. Abb. ??: ??) können wir bereits Vorhandene Smart-Home-Systeme wie Google Cast und Phillips Hue integrieren. Da wir allerdings eine eigene Integration von Zigbee Leuchtmitteln anstreben, wird dieser Schritt übersprungen. Nun leitet uns der Home Assistant auf seine Startseite weiter (vgl. Abb. ??: ??). Dieses Dashboard wird von Home Assistant aus allen bereits vorhandenen Geräten bzw. Entitäten generiert.



(a) Anmeldebildschirm

(b) Festlegen von Name und Standort



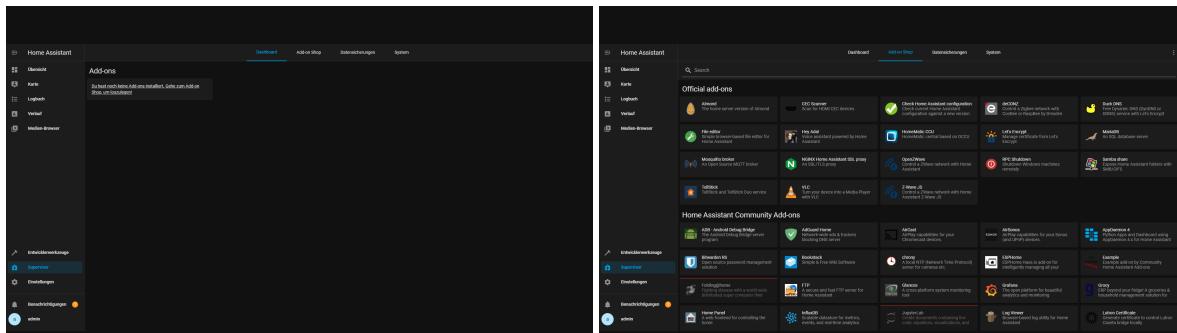
(c) Mögliche Integrationen werden erkannt

(d) erstes Dashboard

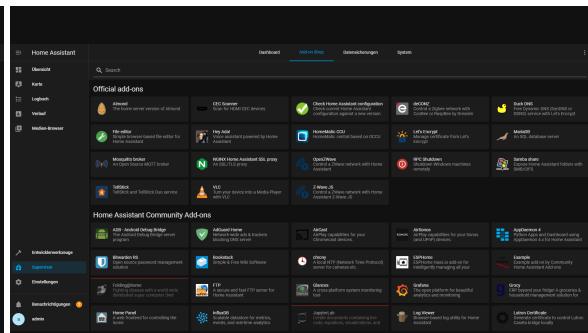
## 7.4 Mosquitto Broker

Als erste wichtige Erweiterung benötigen wir für unseren Home Assistant einen MQTT Broker. Da wir eine Supervised Version des Home Assistant auf unserem Pi installiert haben, nutzen wir für die Installation den Supervisor Add-on Shop von Home Assistant (vgl. Abb. ??: ?? & ??: ??).

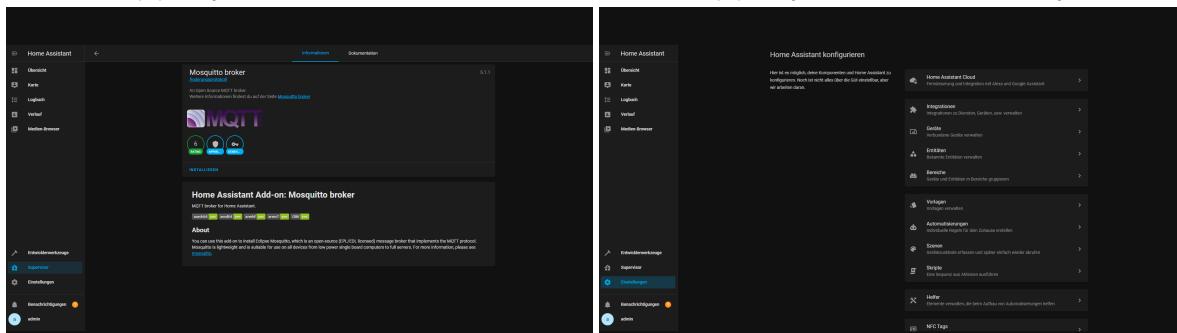
Hier wählen wir den Mosquitto Broker aus und installieren diesen (vgl. Abb. ??: ??). Um den Mosquitto Broker zu aktivieren navigieren wir über den Menüpunkt „Einstellungen“ zum Punkt „Integrationen“ und suchen dort nach MQTT. Wir aktivieren die Verknüpfung von Home Assistant und Mosquitto Broker in dem wir die Schaltfläche „Suche Aktivieren“ markieren und bestätigen. Nach Durchführung dieser Schritte ist der Mosquitto Broker aktiv.



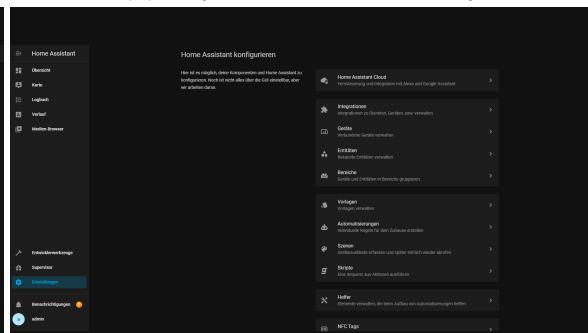
(a) Supervisor Dashboard



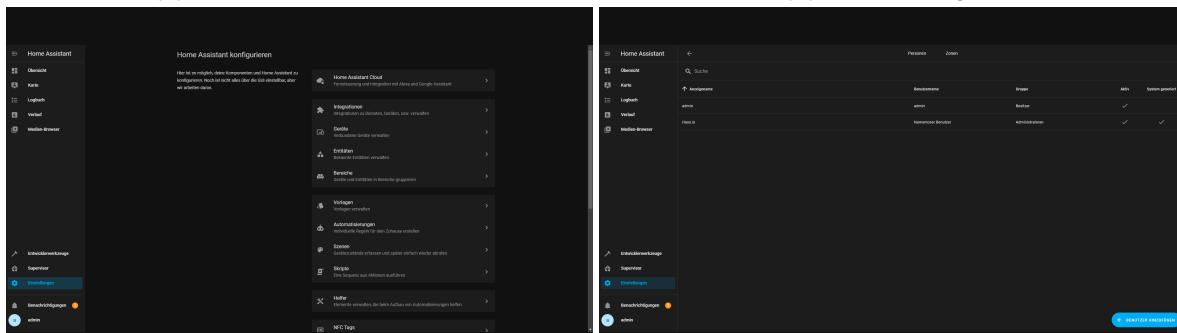
(b) Supervisor Add-on Shop



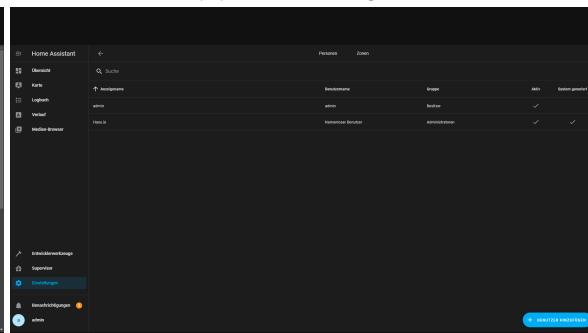
(c) MQTT add-on Seite



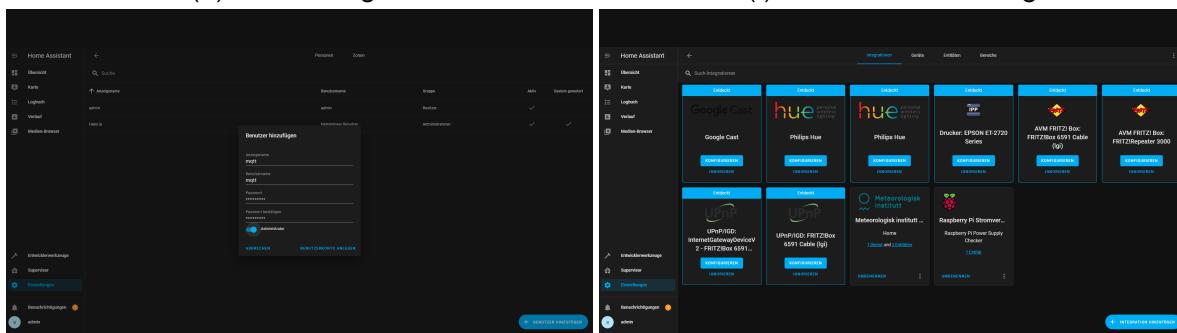
(d) Einstellungen



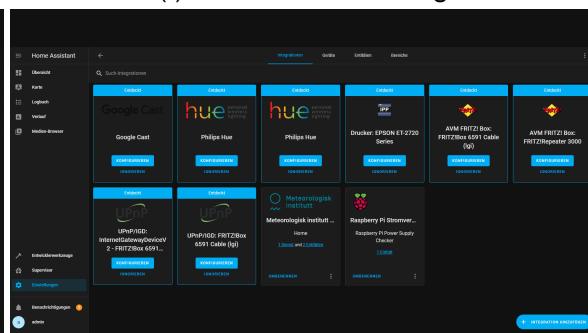
(e) Einstellungen



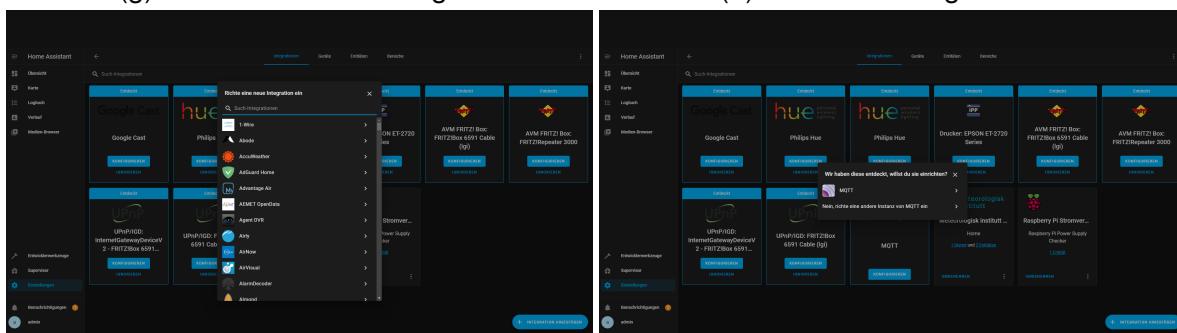
(f) Benutzerverwaltung



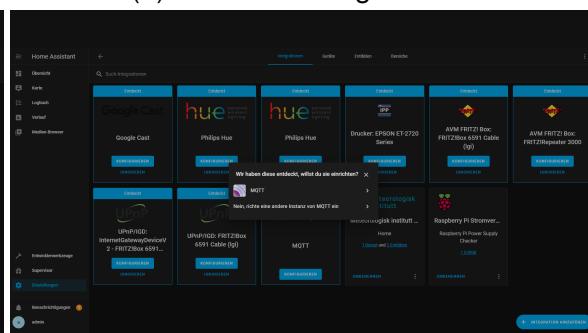
(g) Benutzer MQTT anlegen



(h) Startseite Integrationen



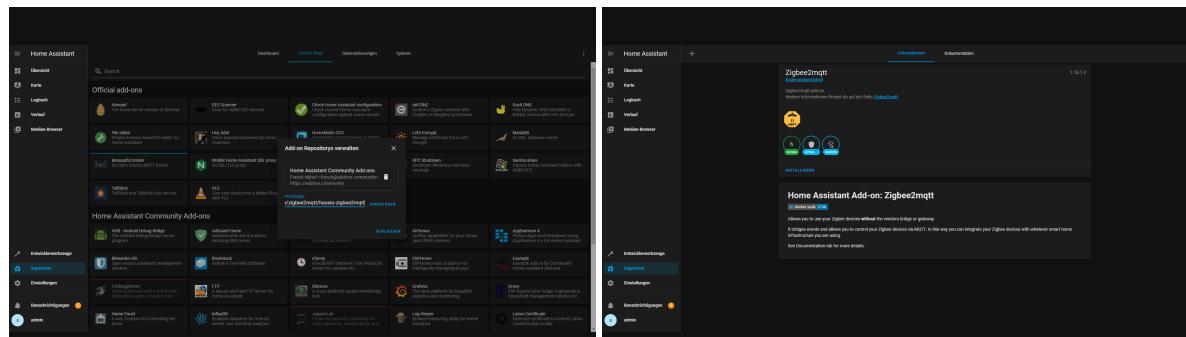
(i) Suchfunktion Integrationen



(j) Suche nach MQTT

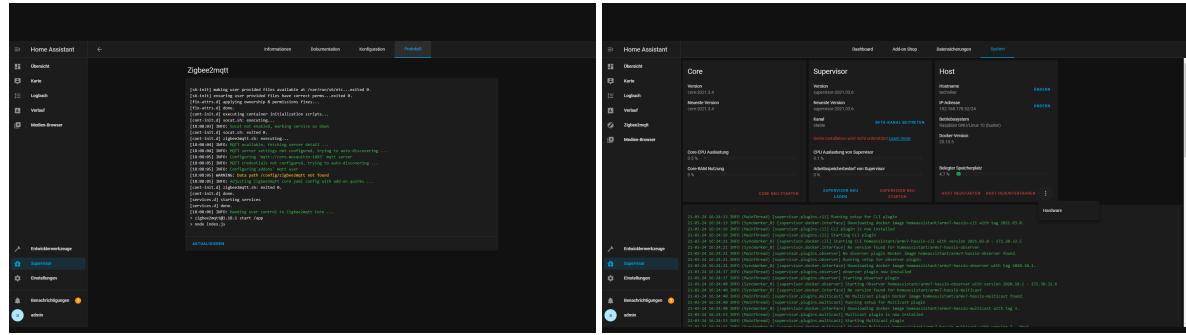
## 7.5 Zigbee2MQTT

Um das Offizielle Zigbee2MQTT Home Assistant Add-on nutzen zu können müssen wir zunächst im Supervisor Add-on Shop(vgl. Abb. ??: ??) das Repository<sup>23</sup> hinzufügen. Hierdurch erhalten wir die möglichkeit das Add-on auf unserem Home Assistant zu installieren. Nachdem das Geschehen ist können wir das Add-on Zigbee2MQTT installieren und starten. An diesem Punkt starten wir das System neu um alle neu installierten Erweiterungen neu zu laden. Um eine Korrekte Konfiguration zu ermöglichen öffnen wir innerhalb des Zigbee2MQTT Add-on's den Reiter Konfiguration. Hier sehen wir die Inhalte der configuration.yaml Datei (vgl. Anhang ??: ??)



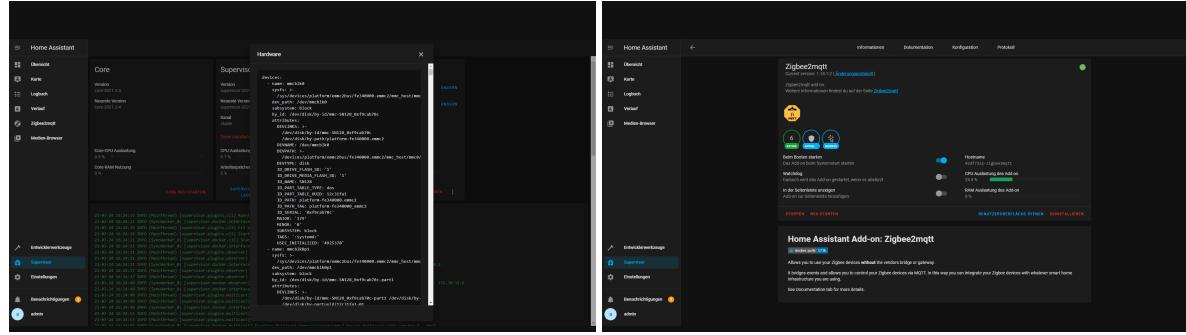
(a) Repository in Add-on Shop einpflegen

(b) Zigbee2MQTT add-on Seite



(c) Configuration.yaml

(d) Supervisor Systeminformationen



(e) Auflistung der Hardware

(f) Aktiviertes Zigbee2MQTT add-on

## 7.6 Erstellung des Betriebssystem-Image

Nach erfolgreichem Test des Installationsskripts<sup>24</sup> haben wir für je einen Raspberry Pi 3 und einen Raspberry Pi 4 ein Image erstellt. Hierzu haben wir mit Hilfe des

<sup>23</sup> GITHUB: <https://github.com/zigbee2mqtt/hassio-zigbee2mqtt>

<sup>24</sup> Siehe Abschnitt ??: ??

Raspberry Pi Imagers eine 8 Gigabyte große Micro-SD-Karte<sup>25</sup> mit Raspberry Pi OS in der 32-Bit-Variante geflasht. Dieser kann über die Tastenkombination [STRG]+[SHIFT]+[X] die Grundeinstellungen wie SSH, Domänennamen und Passwort festlegen (vgl. ??). Dementsprechend wurden für die beiden Versionen folgende Einstellungen vorgenommen:

Image-Name	User	Passwort	Domänenname
shz_rp3.img.gz	pi	raspberry	shzpi3
shz_rp4.img.gz	pi	raspberry	shzpi4

Tabelle 3: Überblick über die Images für die Raspberry Pis

Nachdem die Image auf die SD-Karte geflasht wurden, haben wir die Raspberry Pis mit den entsprechenden Karten bestückt und uns per SSH mit den Geräten verbunden.

Mit Hilfe des Shell-Befehls und der Verwendung des Standardpasswords haben wir uns mit den Raspberry Pis verbunden (vgl. Tab. ??: ??). Anschließend haben wir das Skript auf dem Pi eingerichtet und gestartet (vgl. ??). Nachdem das Skript ausgeführt wurde (vgl. ??) wurde die SD-Karte mit Win32 Disk Imager in eine .img-Datei konvertiert (vgl. ??). Anschließend wurde die Image mit 7Zip komprimiert, um Speicherplatz zu sparen (vgl. ??). Diese Images können dann mit dem Raspberry Pi Imager<sup>26</sup> wieder auf SD-Karten geflasht werden.

Die Images<sup>27</sup> können über folgende Links heruntergeladen werden:

- Raspberry Pi 3 Image: [https://www.mediafire.com/file/bc2o542na3rea7j/shz\\_rp3.img.gz/file](https://www.mediafire.com/file/bc2o542na3rea7j/shz_rp3.img.gz/file)
- Raspberry Pi 4 Image: [https://www.mediafire.com/file/eujoh6s4thpg363/shz\\_rp4.img.gz/file](https://www.mediafire.com/file/eujoh6s4thpg363/shz_rp4.img.gz/file)

<sup>25</sup> Zum Zeitpunkt der Erstellung des Images kleinstmögliche vorhandene Micro-SD-Karte

<sup>26</sup> Alternative: balnea Etcher

<sup>27</sup> Stand: Ende März 2021

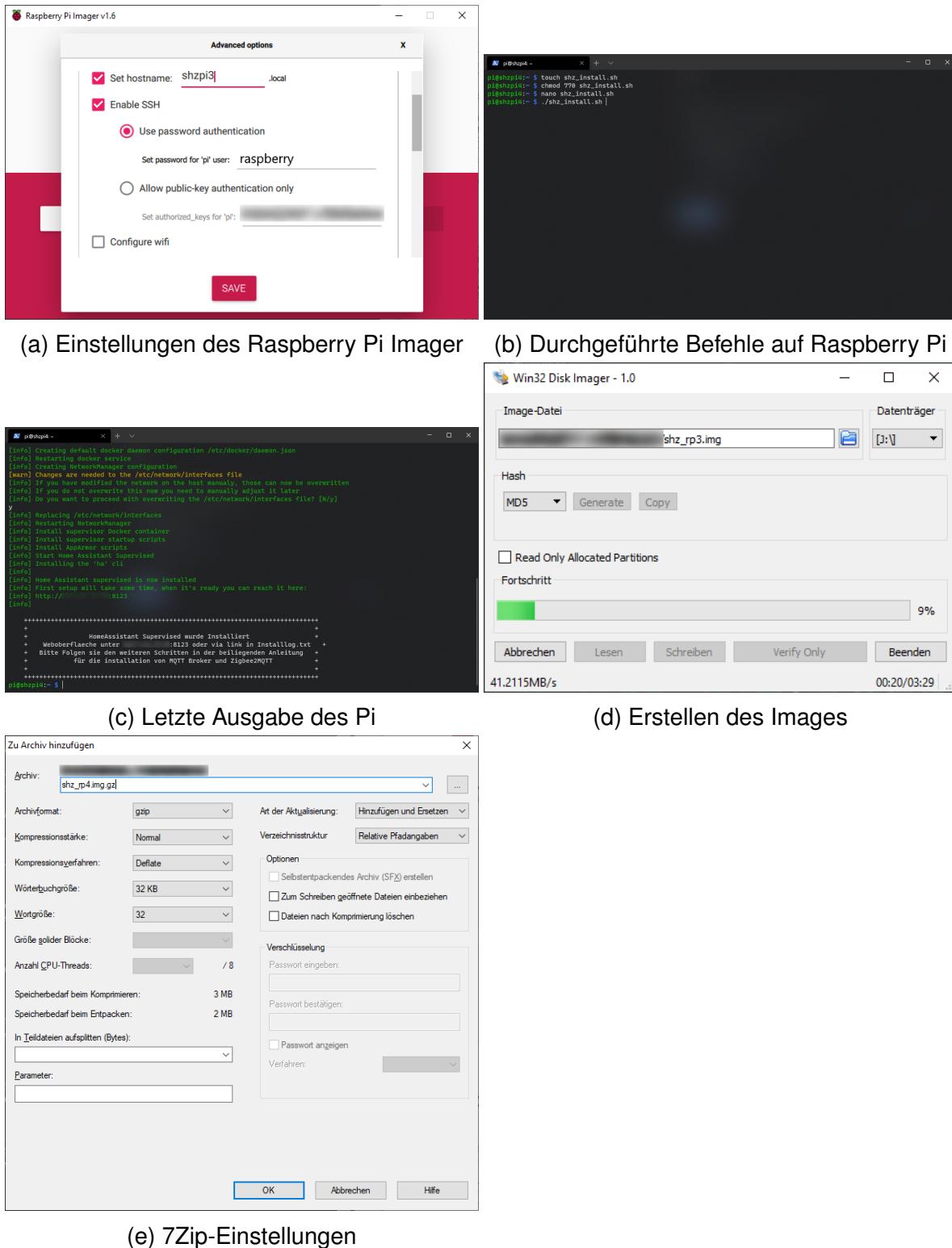


Abbildung 29: Erstellen der Images

## 8 Epilog & Fazit

### 8.1 Fehler & Probleme

Probleme auf die wir während dem Projekt gestoßen sind, Fehler die wir begangen haben und Anekdoten.

- LATEX ist toll
- Ein Leerzeichen kann über einige Stunden Arbeit entscheiden
- Bei einem Test einer Iteration des Installationsskripts haben wir stundenlang nach einer Lösung gesucht, warum die Installation des Home Assistant fehlgeschlug, bis wir feststellten das unser Testsystem lediglich über WLAN mit unserem Netzwerk eingebunden war. Der Network-Manager während der Installation neu gestartet wodurch die WLAN Verbindung neu startet und unsere SSH Verbindung zum PI abbricht. Darüber hinaus war das Installationsskript nicht mit der langen Wiederverbindungszeit im WLAN klargekommen...
- Stützstrukturen können auch mal Spaghetti produzieren
- Eine Fehlentscheidung bei der Home Assistant Version hat uns fast 2 Wochen lang beschäftigt, und unsere Lösung war einfach viel zu Kompliziert und hätte sich auch nicht automatisiert durchführen lassen
- Wir haben uns Stundenlang mit dem Aufsetzen von Docker images von MQTT und Zigbee2MQTT aufgehalten obwohl beides innerhalb von Home Assistant Supervised als Add-on geladen werden können

### 8.2 Danksagung

Danke an unsere Freundinnen, die uns den Rücken so gut es ging freigehalten haben und uns zwei Nerds unterstützt haben wo sie nur konnten. Danke an unsere Familien, die uns mit ihren Weisheiten und Erfahrungen weiterhelfen konnten. Danke an unsere Lehrer, die sich die letzten zwei Jahre mit uns gequält habe.

## 9 Quellen

### 9.1 Dokumentationsquellen

Quellen für Verweise, die in Fußnoten innerhalb dieser Dokumentation erwähnt wurden:

- [github.com: Zigbee2MQTT Projektseite](https://github.com/koenkk/zigbee2mqtt/)  
[https://github.com/koenkk/zigbee2mqtt//](https://github.com/koenkk/zigbee2mqtt/)
- [mosquitto.org: Startseite](https://mosquitto.org/)  
[https://mosquitto.org//](https://mosquitto.org/)

- Li (2017): Was ist ein Smart-Home-Hub? Alles über die intelligente Zentrale  
<https://www.otto.de/updated/ratgeber/erklaert-was-ist-ein-smart-home-hub-80634/>
- Robert (2019): Antwort auf Creality Ender 3 printer power consumption? - 3dprinting Stack Exchange  
<https://3dprinting.stackexchange.com/questions/8616/creality-ender-3-printer-power-consumption#8623/>
- reichelt.de: Produktbeschreibung des Raspberry Pi 4  
<https://www.reichelt.de/raspberry-pi-4-b-4x-1-5-ghz-8-gb-ram-wlan-bt-rasp-pi-4-b-8gb-p276923.html>
- Wikipedia: Raspberry Pi - Generations  
[https://en.wikipedia.org/wiki/Raspberry\\_Pi#Generations](https://en.wikipedia.org/wiki/Raspberry_Pi#Generations)
- Sunfounder: 10.1 Inch Touch Screen for Raspberry Pi(NEW)  
[http://wiki.sunfounder.cc/index.php?title=10.1\\_Inch\\_Touch\\_Screen\\_for\\_Raspberry\\_Pi\\_\(NEW\)#3D-printed\\_Touch\\_Screen\\_Support](http://wiki.sunfounder.cc/index.php?title=10.1_Inch_Touch_Screen_for_Raspberry_Pi_(NEW)#3D-printed_Touch_Screen_Support)

Quellen, die für die Herstellung dieser Dokumentation genutzt wurden:

- Menmiloud Mohammed: L<sup>A</sup>T<sub>E</sub>X-Tutorial.com  
<https://latex-tutorial.com/>
- Overleaf: L<sup>A</sup>T<sub>E</sub>X-Guides  
<https://www.overleaf.com/learn>

## 9.2 Verwendete Software

Software	Verwendung	Version
Raspberry Pi OS	Betriebssystem und Oberfläche für Hardware	5.4
Home Assistant	Betriebssystem und Oberfläche für Smart Home	5.12
Mosquitto broker	Broker für den MQTT nachrichten Standart	5.1.1
Zigbee2MQTT	Anbindung von Zigbee an MQTT	1.18.1-2
KiCad EDA	Erstellung von Schaltplan und Gerber-Datei des Hats	5.1.8
TexMaker	Erstellung der Dokumentation	5.0.4
<a href="#">overleaf.com</a>	Gemeinsame Erstellung der Dokumentation	Web
Fusion360	Erstellung von Gehäusemodell	2.0.9849
CURA	Erstellung von G-Code für 3D-Drucker	4.7
GanttProject	Erstellung von Gantt-Diagrammen	3.0.3
Autodesk Meshmixer	Vorschau-Render der STL-Dateien	3.5.474
paint.net	Bildmanipulation	4.2.15
Win32 Disk Imager	Erstellung der Image aus den Micro-SD-Karten	1.0
7Zip	Komprimieren der Images	19.00
<a href="#">shellcheck.net/</a>	Überprüfung von Bash-Skripten	Web
<a href="#">deepl.com/translator</a>	Übersetzungssoftware	Web

Tabelle 4: Verwendete Software

## 9.3 Verwendete Hardware

Hardware	Verwendung	Version
Raspberry Pi	Hauptplatine für die Smart Home Zentrale	Version 4B (8GB)
CC2531 Zigbee USB Stick mit Firmware	USB-Stick mit ZigBee-Chip	Rev 2.4
Sunfounder 10.1 Touch Screen	Bildschirm und Input für die Smart Home Zentrale	Unbekannt
Creality Ender 3 Pro	Fertigung der Kunststoffteile	Modifiziert

Tabelle 5: Verwendete Hardware

## 10 Anhang

### 10.1 Lastenheft

	TAR	Erstellt am: 04.06.2020 13:35:00
SmartHome e.V. An der Ecke 12 70707 Kleinstadt	Lastenheft	

## Lastenheft

### I. VERANLASSUNG

Der Smart-Home Sektor wird zu einem immer größer werdenden Markt und immer mehr Hersteller liefern neue Produkte um das eigene Haus zu steuern. Um einen zentralen Informationspunkt oder eine zentrale Steuereinheit einzurichten, soll ein Produkt entwickelt werden, welches sich mit Smart-Home-Anwendungen verschiedener Hersteller verbinden kann, um deren Produkte dann per Touchscreen oder wahlweise per Sprachassistent steuern kann.

### II. PROJEKTUMFELD

Für das Projekt stehen diverse Smart-Home-Komponenten zur Verfügung, darunter unter anderem die Smart-Lampen von IKEA, Phillips Hue und diverse Zigbee Komponenten.

### III. ZIELBEDINGUNG

Das Produkt soll eine Lösung zur Verknüpfung von Smart-Home-Geräten via Zigbee-Bridge darstellen und einen Sprachassistenten integrieren, in diesem Fall MyCroft. Darüber soll das Produkt als zentrale Steuereinheit für die angebundenen Sensoren und Geräte dienen und den Status dieser in einem kompakten Display darstellen.

### IV. PRODUKTEINSATZ

Das Produkt dient zur Steuerung eigener Zigbee fähiger Smart-Home-Komponenten. Als solches kann es vom Kunden im eigenen Netz integriert werden und dient dort als Steuerung oder Übersicht der eigenen Smart-Home-Geräten

### V. PRODUKTMERKMALE

- V.1. Hardware in einem Gehäuse (Touch-Bildschirm mit Raspberry Pi 4B als „Gehirn“)
- V.2. Dashboard-Funktion für Smart-Home-Geräte
- V.3. Integration von Zigbee, MQTT
- V.4. GUI erstellt über Java oder als WebUI (eventuell als Smartphone-App)
- V.5. Mögliche Integration von Sprachassistenten MyCroft

### VI. PRODUKTDATEN

Das Gerät soll neben Daten der Sensoren (Luftdruck, Temperatur, Feuchtigkeit, etc.) auch den Zustand von anderen Smart-Home-kompatiblen Geräten anzeigen können (ein- oder ausgeschaltet, Fenster offen, etc.)

### VII. ABLAUFORGANISATION

Die Abläufe für die in Punkt V genannten Vorgänge entsprechen zum Großteil dem geplanten Umfang des Projekts. Zusätzlich soll noch ein Gehäuse entwickelt werden, dass die Hardware (Bildschirm, SBC, Mikrofon und eventuell Kamera und Netzteil) in einem Gerät vereint. Weitere Produktfähigkeiten werden bei Bedarf nachgereicht. Der Projektlauf soll als SCRUM stattfinden, ein Sprint hat die geplante Dauer von einem Monat.

 it.schule stuttgart	TAR	Erstellt am: 04.06.2020 13:35:00
	Lastenheft	

### VIII. PRODUKTLEISTUNG

	Sehr gut	Gut	Normal	Nicht Relevant
Funktionalität		x		
Bedienerfreundlichkeit	x			
Änderbarkeit		x		
Erweiterbarkeit			x	

## 10.2 Installationsskript

## 10.3 YAML-Dateien

10.3.1 Home Assistant Konfiguration

10.3.2 Zigbee2MQTT Konfiguration

10.3.3 Zigbee Geräteauflistung

## 10.4 Gehäuse-Zeichnungen

pdf/gehäuse\_zeichnung\_fu\T1\ss\_abdruck\_v1.pdf

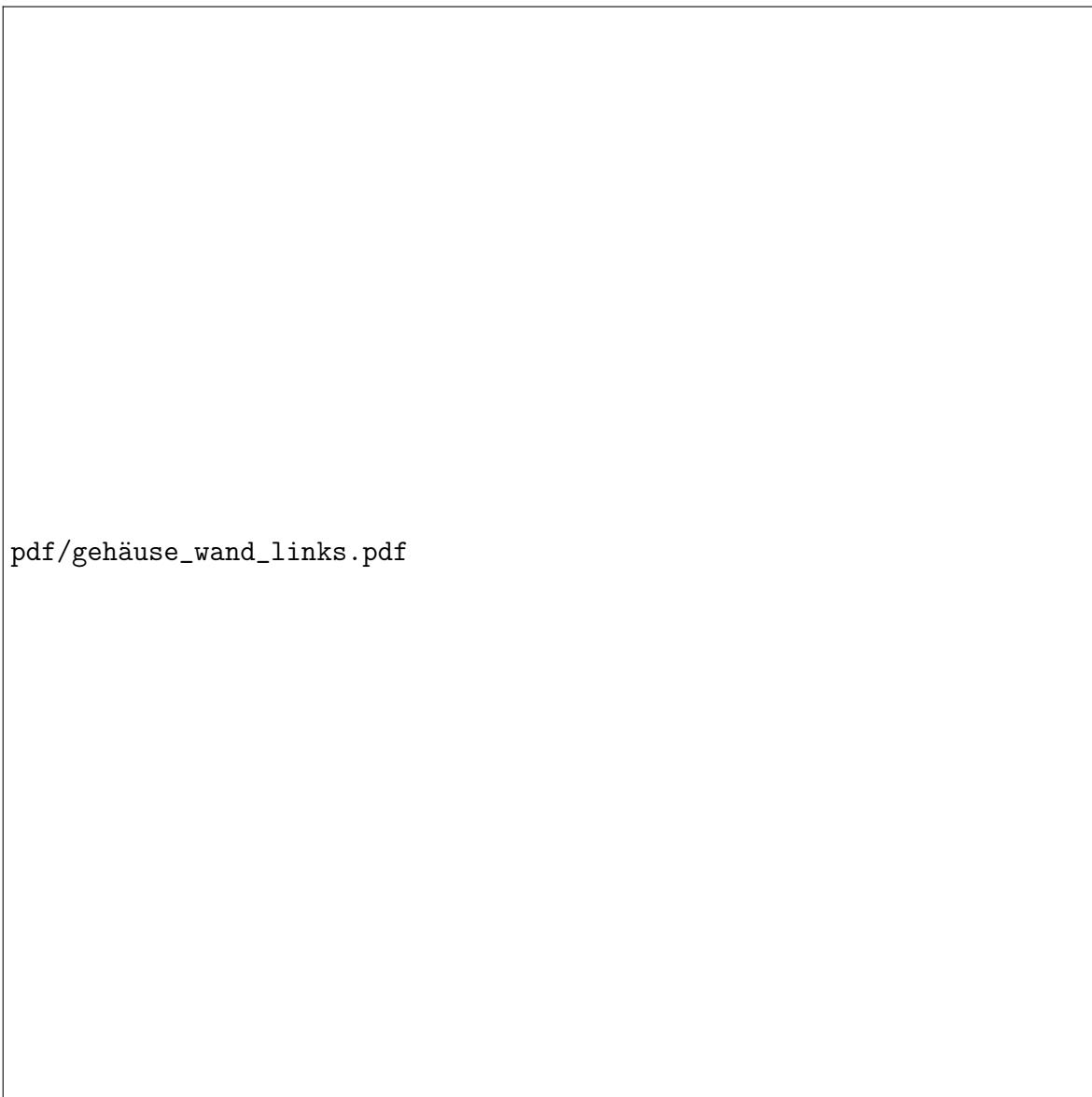
ßabdruck\_v1.pdf

Erster Versuch der Zeichnung des Fußabdrucks des Gehäuses

pdf/gehäuse\_zeichnung\_fu\T1\ss abdruck\_final.pdf

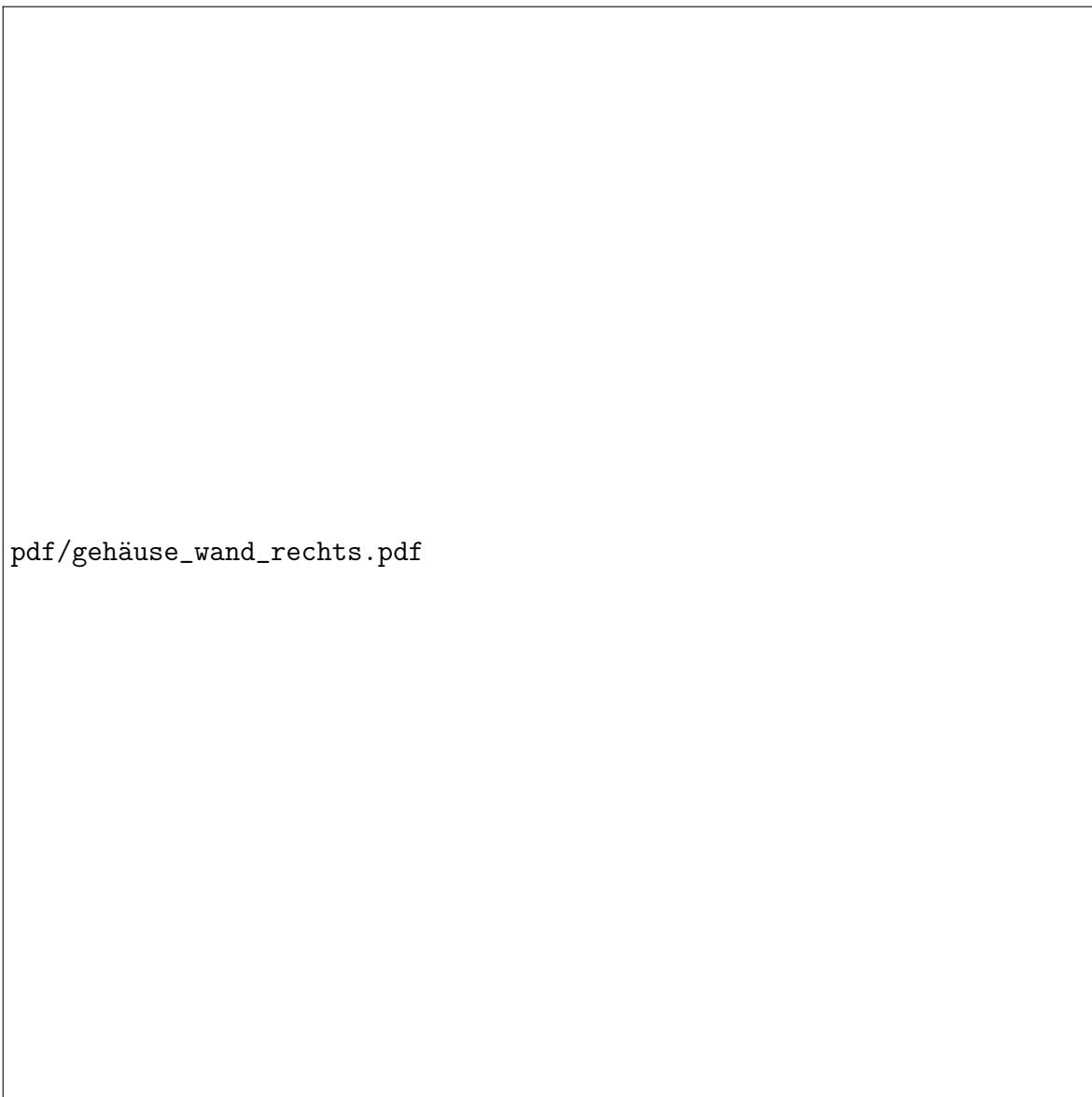
ßabdruck<sub>final</sub>.pdf

Finale Version der Zeichnung des Fußabdrucks des Gehäuses



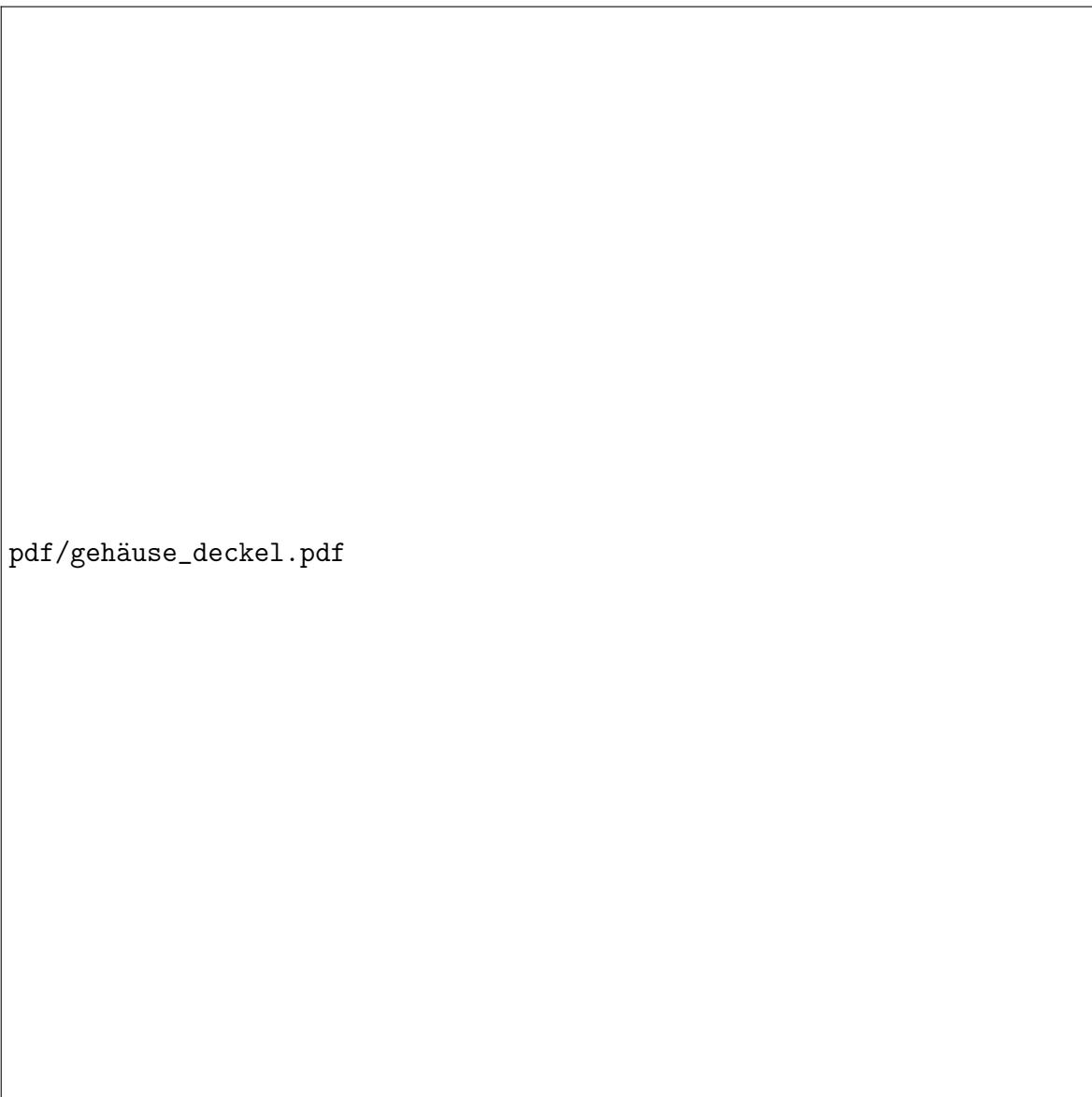
pdf/gehäuse\_wand\_links.pdf

Linker Teil des Gehäuses



pdf/gehäuse\_wand\_rechts.pdf

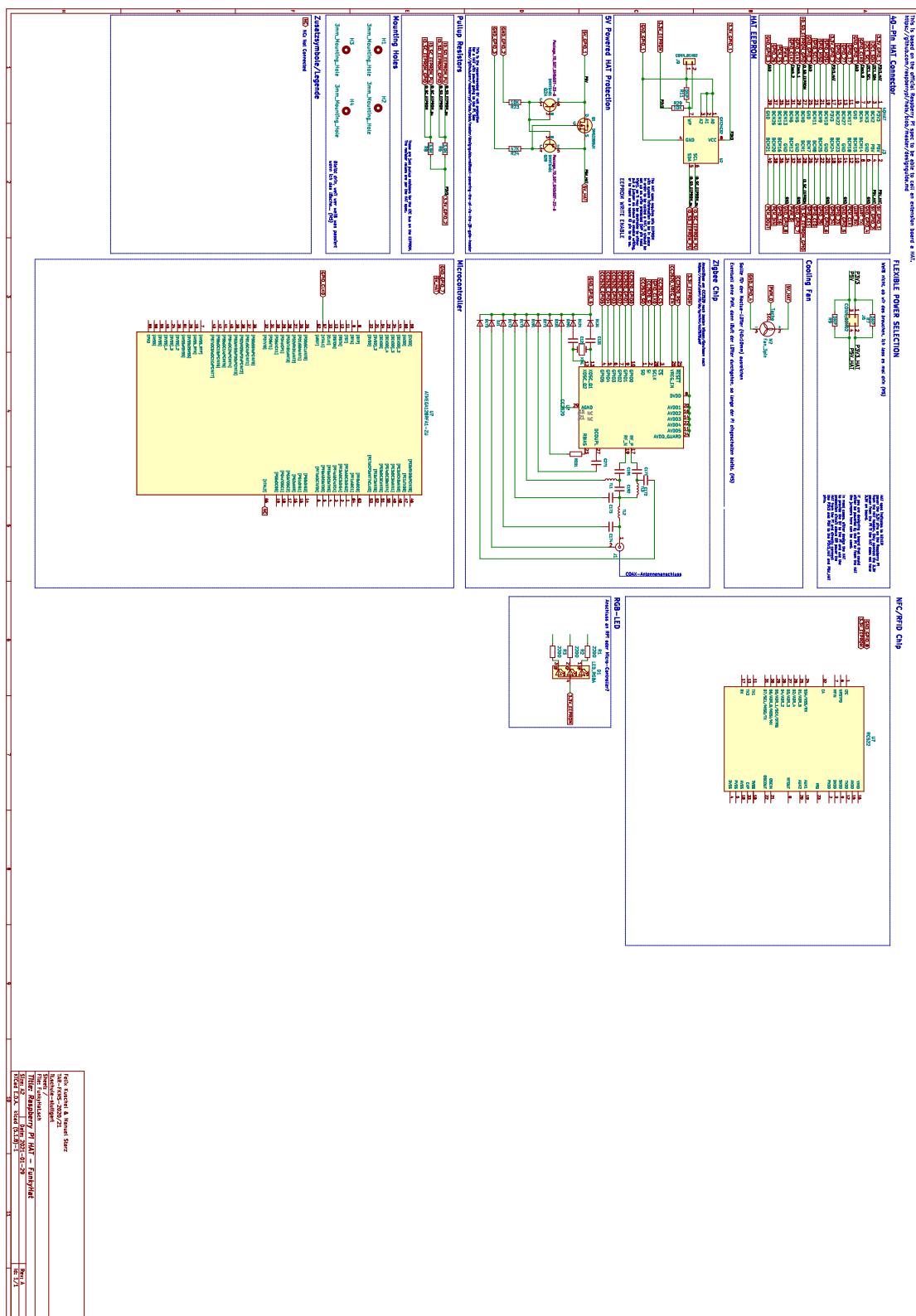
Rechter Teil des Gehäuses



pdf/gehäuse\_deckel.pdf

Ein Teil des Gehäusesdeckels

## 10.5 Raspberry Pi HAT E-Schema



## 10.6 Hardware-Dokumentationen

Durch den Umfang der einzelnen Dokumentationen hier nur eine Auflistung der Dokumentationen mit dem Link zu den PDF im GIT-Projekt bzw. den Herstellerseiten.

- Raspberry Pi:  
[https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi\\_DATA\\_2711\\_1p0.pdf](https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi_DATA_2711_1p0.pdf)
- MiFare MFRC522:  
<https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>
- CC2531 ZigBee SoC:  
<https://www.ti.com/lit/ds/symlink/cc2531.pdf>
- ATmega128RFA1-ZU:  
[https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8266-MCU\\_Wireless-ATmega128RFA1\\_Datasheet.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8266-MCU_Wireless-ATmega128RFA1_Datasheet.pdf)

## 11 Verzeichnisse

### Tabellenverzeichnis

### Abbildungsverzeichnis