



TECHNIKERARBEIT

# **ENTWICKLUNG EINER SMART HOME ZENTRALE AUF BASIS EINES RASPBERRY PI**

Felix KUSCHEL & Manuel STARZ

Betreut durch  
Matthias KOHLER

4. April 2021

# Inhaltsverzeichnis

<b>1 Vorwort</b>	<b>4</b>
1.1 Einleitung . . . . .	4
1.2 Projektrahmen . . . . .	4
1.3 Aufgabenstellung . . . . .	4
1.4 Zusatzinformationen . . . . .	4
1.5 Definition Smart Home Zentrale . . . . .	5
1.6 Datenschutzhinweis . . . . .	5
<b>2 Zielsetzung</b>	<b>6</b>
2.1 Konzeption . . . . .	6
2.2 Anforderungen und gewünschte Features . . . . .	6
2.3 Erweiterungsmöglichkeiten . . . . .	7
<b>3 Herangehensweise</b>	<b>8</b>
3.1 Hardware . . . . .	8
3.2 Software . . . . .	9
3.2.1 Home Assistant . . . . .	9
3.2.2 Zigbee2MQTT . . . . .	10
<b>4 Zeitplan</b>	<b>12</b>
<b>5 Komplettübersicht</b>	<b>13</b>
5.1 Das „fertige“ Produkt . . . . .	13
5.2 Kostenaufstellung . . . . .	15
5.2.1 Beschaffungskosten . . . . .	15
5.2.2 Arbeitszeit . . . . .	16
5.2.3 Kostenberechnung für die 3D-Druckteile . . . . .	16
5.2.4 Kosten Präsentationsboard . . . . .	17
5.2.5 Gesamtkosten . . . . .	17
<b>6 Hardware</b>	<b>18</b>
6.1 Raspberry Pi 4 . . . . .	18
6.2 Aufbau des Prototypen . . . . .	19
6.3 Gehäuse . . . . .	20
6.3.1 Erster Test zur Fertigung des Gehäuses . . . . .	20
6.3.2 Modellentwicklung am Objekt . . . . .	23
6.3.3 Herstellung des Gehäuses . . . . .	30
6.4 Erstellung des RPi-HATs . . . . .	33
6.5 Präsentationsaufbau . . . . .	35
<b>7 Software</b>	<b>36</b>
7.1 Überblick . . . . .	36
7.2 Installationsskript . . . . .	36
7.3 Home Assistant Supervised . . . . .	38
7.4 Mosquitto Broker . . . . .	39
7.5 Zigbee2MQTT . . . . .	41
7.6 Erstellung des Betriebssystem-Image . . . . .	42

---

<b>8 Epilog &amp; Fazit</b>	<b>45</b>
8.1 Fehler & Probleme . . . . .	45
8.2 Danksagung . . . . .	46
<b>9 Quellen</b>	<b>47</b>
9.1 Dokumentationsquellen . . . . .	47
9.2 Verwendete Software . . . . .	48
9.3 Verwendete Hardware . . . . .	48
<b>10 Anhang</b>	<b>49</b>
10.1 Lastenheft . . . . .	49
10.2 Installationsskript . . . . .	51
10.3 YAML-Dateien . . . . .	61
10.3.1 Home Assistant Konfiguration . . . . .	61
10.3.2 Zigbee2MQTT Konfiguration . . . . .	61
10.3.3 Zigbee Geräteauflistung . . . . .	62
10.4 Gehäuse-Zeichnungen . . . . .	63
10.5 Raspberry Pi HAT E-Schema . . . . .	68
10.6 Hardware-Dokumentationen . . . . .	69
<b>11 Verzeichnisse</b>	<b>70</b>

## 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 Erreichens 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. März 2021 angesetzt. Es stand zur Umsetzung des Projekts ein unterrichtsfreier Tag pro Woche zur Verfügung.

Des Weiteren wurden die in dem Zeitraum zur Verfügung stehenden Ferien zur Umsetzung des Projekts genutzt. Die Projektbetreuung erfolgt seitens der Schule durch Herr Matthias Kohler. Da das Projekt nicht in Zusammenarbeit mit einem Unternehmen durchgeführt wird, gibt es keine weiteren Betreuer. Die Materialkosten für die im Projekt genutzte Hardware wurde von uns selbst getragen. Der im Lastenheft (vgl. Anhang 10.1: Lastenheft) erwähnte Verein „Smart Home e.V.“ ist ein fiktiver Verein und dient lediglich als Platzhalter.

## 1.3 Aufgabenstellung

Das Projekt ist aus dem Zusammenschluss der Abschlussarbeitsideen von uns, Felix Kuschel und Manuel Starz, entstanden und wurde in 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, mit der Smart Home Geräte mehrerer Hersteller miteinander verknüpft werden können und so die Notwendigkeit entfällt verschiedene sogenannte Hubs zu müssen.

Das Gerät soll darüber hinaus noch über einen Touchscreen steuerbar sein und die Parameter aller verbundenen Smart Home Geräte anzeigen. Dies umfasst unter anderem den Status von Leuchtmitteln, die Parameter von Thermostaten, sowie den Zustand von Tür- und Fensterkontakte. Die genaue Aufgabenstellung kann dem abgegebenen Lastenheft im Anhang (10.1: Lastenheft entnommen werden).

## 1.4 Zusatzinformationen

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

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

Ursprünglich war dafür die Verwendung des schulinternen GIT geplant. Dieses stand zum Zeitpunkt der Erstellung der Dokumentation allerdings nicht zur Verfügung<sup>1</sup>, weshalb GitHub genutzt wurde.

## 1.5 Definition Smart Home Zentrale



(a) Amazon Alexa Echo Show 8

(b) Google Nest Hub

(c) Glancr Smart Mirror

Abbildung 1: Smart Home Zentralen

Smart Home Zentralen, auch Smart Hubs oder im speziellen Fall des Smart Mirror von Glancr 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 oder 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 Home Assistant, Google Home und openHAB.

## 1.6 Datenschutzhinweis

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

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

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



(a) Home Assistant



(b) Google Home



(c) openHAB

Abbildung 2: Smart Home Software

## 2 Zielsetzung

Als Ziel für das Projekt war ein funktionsfähiges Smart Home Hub mit Zigbee-Anbindung auf Basis eines Raspberry Pi 4 geplant. Das Smart Home Hub in seiner Grundfunktion soll in der Lage sein, die mit ihm verbundenen Geräte über den Zigbee-Standard anzusteuern. Zusätzlich sollte die Möglichkeit einer Erweiterung mit einem Sprachassistenten möglich sein.

### 2.1 Konzeption

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

Dann konnten wir uns an die Beschaffung der unserer Meinung nach notwendigen Komponenten für das Projekt machen.

### 2.2 Anforderungen und gewünschte Features

Die Anforderungen an das Projekt lauteten 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. Zusätzlich hinaus waren von unserer Seite noch folgende Features geplant:

- Ein- und Ausgabe über einen Touch-Bildschirm
- Einbindung des Sprachassistenten MyCroft zur Steuerung der verbundenen Endgeräte

Zusätzlich haben wir ein Installationsskript und ein Betriebssystem-Image für den Raspberry Pi zu erstellt.

Um das Endprodukt wertiger gestalten zu können, haben wir zusätzlich ein Gehäuse für die Hardware entworfen.

## 2.3 Erweiterungsmöglichkeiten

Als eine Erweiterung des Projekts ist ein sogenannter Raspberry Pi HAT, eine aufsteckbare Erweiterung für den Raspberry Pi geplant. Die Idee dazu entstand in Rücksprache mit unserem Projektbetreuer, der den HAT darüber hinaus als Unterrichtsmaterial im Laborunterricht einsetzen wollte. Dieser sollte die Hardware des Projekts, falls möglich auf einer Platine vereinen.

Der HAT sollte folgende Bauteile enthalten:

- 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 in der Schule

Der HAT wurde aber aufgrund der aktuellen Pandemie-Situation und den damit verbundenen Liefer- und Zollschwierigkeiten verworfen. Die Planung des HAT ist daher nicht vollständig, kann aber bei Bedarf ergänzt werden (vgl. Abschnitt 6.4: Erstellung des RPi-HATs).

## 3 Herangehensweise

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

### 3.1 Hardware

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

- das Projekt sollte nach Möglichkeit von durchschnittlich begabten Hobbyhandwerkern durchgeführt werden
- die Fertigstellung des Projekts sollte mit handelsüblichen Werkzeugen möglich sein
- die Grundplattform sollte der 2020 herausgebrachte Raspberry Pi 4 8GB sein<sup>3</sup>
- das Gerät sollte Zigbee-Endgeräte verwalten können, da diese von vielen Herstellern vertriebenen werden.

Die Beschaffung der Teile lief dank Online-Versandhandel relativ problemlos. Beim Erstellen des Prototyps haben wir nur aus Bildschirm, dem Raspberry Pi und dem Zigbee-USB-Stick zusammengebaut (vgl. Abschnitt 6.2: Aufbau des Prototypen). Die Vorlage Standfüße stammt vom Hersteller des Bildschirms Sunfounder<sup>4</sup>. Diese haben wir aus PETG<sup>5</sup> auf dem 3D-Drucker Ender 3 selbst gedruckt. Im Folgenden haben wir uns mit der Software des Projekts befasst (vgl. Abschnitt 3.2: Software und Kapitel 7: Software).

Nachdem der Prototyp soweit funktionsfähig war, haben wir das Präsentationsmodell entworfen. Dabei ging es uns vornehmlich um die Unterbringung der geplanten Komponenten in einem einfach herzustellendem Gehäuse. Hierfür waren einige Verlängerung der Kabel nötig, um Stromzufuhr, Netzwerk und die Antenne des Zigbee-Sticks nach außen zu führen (vgl. Abschnitt 6.3.2: Modellentwicklung am Objekt und Abschnitt 5.1: Das „fertige“ Produkt).

Das Gehäuse wurde nach dem Druck mit dem Bildschirm verklebt und die einzelnen Komponenten eingebaut (vgl. Abschnitt 6.3.3: Herstellung des Gehäuses). Nach dem Zusammenbau haben wir einen Präsentationsaufbau, der die Funktionsweise unseres Prototyps in einem „typischen“ Heimnetzwerk darstellen soll (vgl. Abschnitt 6.5: Präsentationsaufbau).

Näheres zur Umsetzung der Hardware im Kapitel 6: Hardware.

<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

<sup>5</sup> PETG: Glycol modifiziertes PET

## 3.2 Software

### 3.2.1 Home Assistant

Im Bereich Software haben wir uns bereits verfügbare Lösungen für Smart Home Zentralen angeschaut. Ursprünglich war unser Plan, eine eigene Softwarelösung mit Web-Interface oder einer App-Anbindung für Smartphones zu programmieren.

Diese Idee haben wir nach kurzer Zeit aber wieder verworfen, da dieser Markt sowohl mit kostenlosen als auch kostenpflichtigen Lösungen mehr als gesättigt ist. Von den kostenlosen und quelloffenen Lösungen war für uns Home Assistant am interessantesten. Vor allem wegen der großen Menge an Erweiterungsmöglichkeiten 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 einfache Basisversion des Home Assistants, während die Supervised Version ein Addon-Repository integriert.<sup>6</sup>

Zusätzlich vertreibt Home Assistant auch noch eine fertige Out-of-the-Box-Lösung namens Home Assistant Blue<sup>7</sup> für 140\$, allerdings gibt es nur wenige Länder, in denen das Gerät verfügbar ist und es besitzt anders als unsere Lösung integrierten keinen Bildschirm.

Für unsere Anwendung war das eigene Betriebssystem-Image nicht verwendbar, da sich auf dem System keine grafische Oberfläche nachinstallieren ließ. Aus diesem Grund haben wir uns für zuerst die Programm-Variante des Home Assistants entschieden.

Allerdings gab es bei der Installation der Supervised Version auf unterschiedlichen Versionen des Raspberry Pis Probleme. Die Core-Version war für unsere Zwecke ungeeignet, da der Arbeitsaufwand für die Integration von zusätzlichen Komponenten und Systemerweiterungen für die Zielgruppe des Projekts zu komplex wäre.

Während unserer Testphase hat sich gezeigt, dass die Verknüpfung von Home Assistant Core, Mosquitto Broker sowie Zigbee2MQTT nicht zu dauerhaft reproduzierbaren Erfolgen geführt hat und sich deshalb nicht wie von uns geplant automatisieren ließ. Deshalb haben wir uns für die Container-Variante mit Docker entschieden. Das Installationsskript (vgl. 10.2: Installationsskript) installiert die benötigten Programme und übernimmt die Grundeinrichtung des Home Assistants bis zu dem Punkt, an dem die Einrichtung über die Weboberfläche durch den Nutzer erfolgt.

Um neben dem Skript noch eine weitere Art der Installation zu ermöglichen, bieten wir

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

<sup>7</sup> siehe <https://www.home-assistant.io/blue>

die Möglichkeit der Nutzung eines Images. Für die beiden in Verwendung befindlichen Raspberry Pi Versionen<sup>8</sup> haben wir entsprechend nach Ablauf des Installationsskripts ein Backup-Image erzeugt. Der Benutzername und das Passwort entsprechen bei den Images den standardmäßig in Raspberry Pi OS hinterlegten Nutzern und sollten beim booten sicherheitshalber geändert werden.

Nähtere Informationen hierzu im Kapitel 7: Software.

### 3.2.2 MQTT und Zigbee2MQTT

Um mit Zigbee die von uns angestrebte Steuerung und Integration von Leuchtmitteln und Geräten zu ermöglichen, benötigen wir den Übertragungsstandard MQTT. Dieser bietet die Möglichkeit, die Zigbee-Daten an den Home Assistant weiter zu reichen. Wir haben uns schon frühzeitig auf den Mosquitto Broker zur Verarbeitung des Zigbee-Protokolls entschieden.

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

### Beschreibung MQTT

MQTT is an OASIS standard for IoT connectivity. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the “Internet of Things” world of connected devices, and for mobile applications where bandwidth and battery power are at a premium.<sup>9</sup>

### Beschreibung Mosquitto Broker

Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.

The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers.

The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular mosquitto-pub and mosquitto-sub command line MQTT clients.<sup>10</sup>

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

<sup>9</sup> MQTT.org (2021): What is MQTT

<sup>10</sup> mosquitto.org (2021): Eclipse Mosquitto<sup>TM</sup> An open source MQTT broker

## Beschreibung Zigbee2MQTT

Zigbee2MQTT is made up of three modules, each developed in its own Github project. Starting from the hardware (adapter) and moving up; zigbee-herdsman connects to your Zigbee adapter and makes an API available to the higher levels of the stack. For e.g. Texas Instruments hardware, zigbee-herdsman uses the TI zStack monitoring and test API to communicate with the adapter. Zigbee-herdsman handles the core Zigbee communication. The module zigbee-herdsman-converters handles the mapping from individual device models to the Zigbee clusters they support. Zigbee clusters are the layers of the Zigbee protocol on top of the base protocol that define things like how lights, sensors and switches talk to each other over the Zigbee network. Finally, the Zigbee2MQTT module drives zigbee-herdsman and maps the zigbee messages to MQTT messages. Zigbee2MQTT also keeps track of the state of the system. It uses a database.db file to store this state; a text file with a JSON database of connected devices and their capabilities.<sup>11</sup>

<sup>11</sup> Koen Kanders (2021): Zigbee2MQTT

## 4 Zeitplan

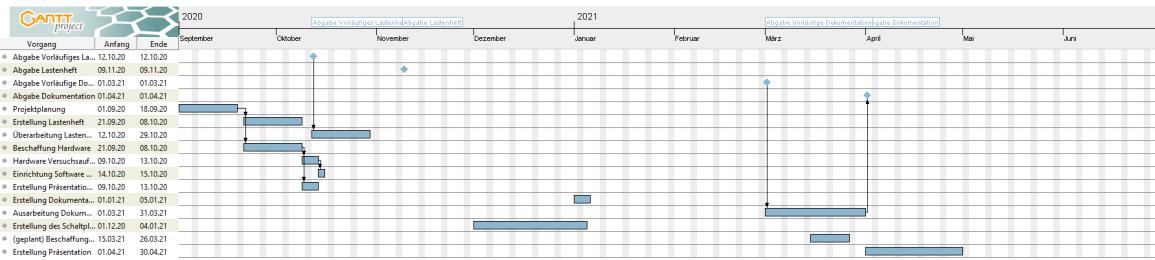


Abbildung 3: Gantt-Diagramm des Projektablaufs

Um einen besseren Überblick der zu erledigenden Aufgaben zu haben, haben wir mit GanttProject ein Gantt-Diagramm mit den geplanten Aufgaben und Meilensteinen unseres Projekts angelegt (vgl. Abb. 3: Gantt-Diagramm des Projektablaufs). Die eigentliche Projektorganisation war als SCRUM Projekt geplant.

Für diesen Zweck haben wir das KANBAN-Board in Microsoft Teams genutzt (vgl. Abb 4: KANBAN-Board in Microsoft Teams). Auf Grund dessen haben wir stets einen Überblick über die noch offenen und die bereits erledigten Aufgaben.

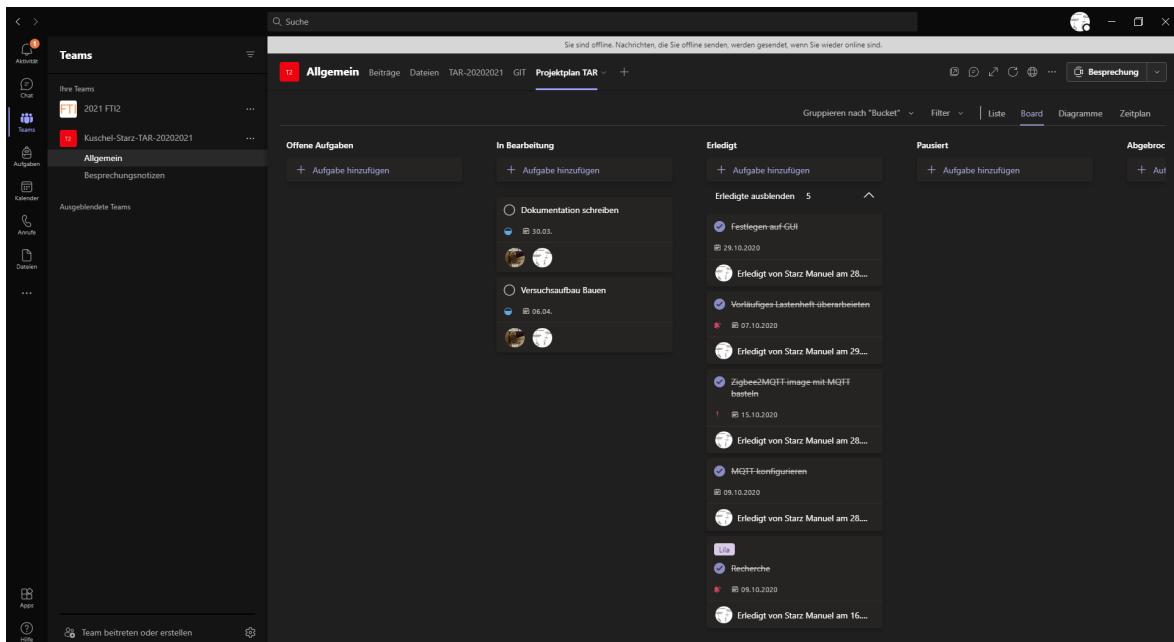


Abbildung 4: KANBAN-Board in Microsoft Teams

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

## 5 Komplettübersicht

### 5.1 Das „fertige“ Produkt



Abbildung 5: Smart Home Zentrale

In Abbildung 5 wird das Ergebnis der Technikerarbeit von Felix Kuschel und Manuel Starz präsentiert. Die Smart Home Zentrale verfügt über eine ZigBee-Antenne, einem 10.1" großen Touchbildschirm und der Möglichkeit, per LAN und oder WLAN mit dem Heimnetz in Verbindung zu treten.

Das Gehäuse ist für die Wand-Montage (vgl. Abb. 5: Smart Home Zentrale) oder als Aufstellgerät (vgl. Abb. 6: Gehäuse mit Füßen (Render in Meshmixer)) entworfen worden.

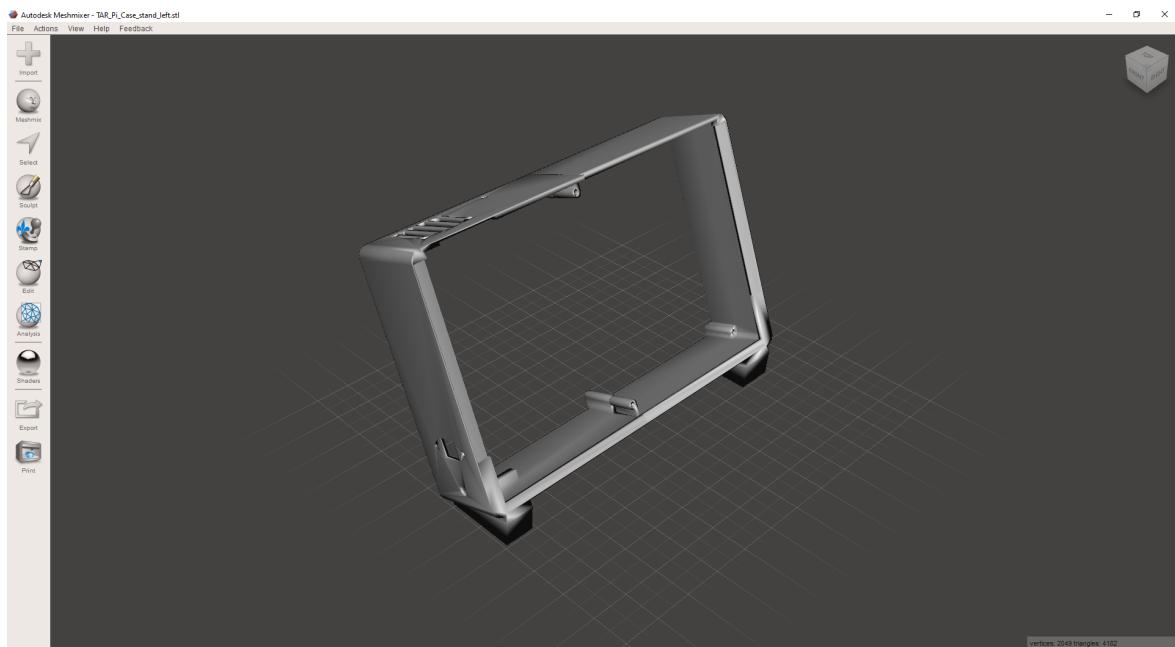


Abbildung 6: Gehäuse mit Füßen (Render in Meshmixer)

## 5.2 Kostenaufstellung

Im Anschluss haben wir die Kosten des Projekts dargestellt (vgl. Tab 4: Gesamtkostenberechnung)

### 5.2.1 Beschaffungskosten

Produkt	Menge	Kosten pro Stück	Kosten insgesamt
Raspberry Pi 4 Modell B 8 GB	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 5,5 mm Stecker & Buchse	1	8,29 €	8,29 €
dasFilament PTGE schwarz 1,75 mm	0,52	24,95 € <sup>12</sup>	11,61€ <sup>13</sup>
USB-Verlängerung	1	9,99 €	9,99 €
<b>Gesamtkosten</b>			<b>309,37 €</b>

Tabelle 1: Beschaffungskosten

### 5.2.2 Arbeitszeit

Gerechnet wird in Stunden, ein Arbeitstag hat 8 Stunden.

Projektschnitt	Dauer (in h)
Recherche	60
Themenfindung	6
Anfertigung Lastenheft	20
Beschaffung Hardware	4
Entwicklung Prototyp	40
Konstruktion Gehäuse	40
Software Installation	8
Entwurf Skript	80
Erstellung Images	4
Entwurf Raspberry Pi HAT	30
Dokumentation	80
Funktionsprüfung	20
Präsentationsaufbau	8
<b>Gesamt</b>	<b>400</b>

Tabelle 2: Erfassung der Arbeitszeiten

Bei dem aktuell gültigen Mindestlohn von 9,50 €<sup>14</sup> kommen wir so auf Arbeitskosten von 3800 €.

### 5.2.3 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 aufteilen: Material und Energie. Die Materialkosten lassen sich leicht erfassen, da wir die Kosten der Spule Filament mit 24,95 € und einem Gewicht von 800 g Filament auf der Spule gegeben haben. So ergibt sich ein Preis von 0,03(12...)€ ( $K_{Material/g}$ ) pro Gramm verwendetem Filament. Der Druck der Gehäuseteile verbrauchte 423 g Material ( $M_{Material}$ ), das heißt, dass die Materialkosten sich auf 13,19 € belaufen.

Die Energiekosten zu berechnen ist etwas komplexer. Der 3D-Drucker verbraucht circa 120 W pro Stunde( $V_{\emptyset/h}$ )<sup>15</sup>, was bei einem Strompreis von 0,31 € / kWh ( $K_{kWh}$ ) bedeuten würde, dass wir pro Stunde etwa 0,0372 € Stromkosten haben. Bei einer Gesamtdruckzeit von 36 Stunden und 36 Minuten( $T_{Druck}$ ) ergeben sich also Stromkosten von 1,37 €.

Addiert man nun die Kosten für Material und Energie, so ergibt sich ein Fertigungspreis von 12,98 € ( $K_{gesamt}$ ). Dabei ist der Zeitaufwand für die Veredelung, also Schleifen und Lackieren, noch nicht mit einkalkuliert.

<sup>14</sup> Deutscher Gewerkschaftsbund (2021): Mindestlohn 2021/2022: Was ändert sich?

<sup>15</sup> Robert (2019): Antwort auf Creality Ender 3 printer power consumption? - 3dprinting Stack Exchange

### 5.2.4 Kosten Präsentationsboard

Produkt	Menge	Kosten pro Stück	Kosten insgesamt
OSB Platte	1	11,22 €	11,22 €
Kanthalz	4,2 m	1,38 €	2,76 €
Lampenfassung E27 Schwarz	2	2,99 €	5,98 €
Winkelverbinder Gleichschenklig	4	4,99 €	19,96 €
Konstruktionswinkel	2	1,19 €	2,38 €
Schrauben	1	12,99 €	12,99 €
Mantelleitung NYM-J 3 x 1,5 Grau 5 m	1	4,99 €	4,99 €
Abzweigkasten Aufputz	3	1,59 €	4,77 €
Schutzkontakt Flachstecker mit seitlicher Einführung	1	1,69 €	1,69 €
Schutzkontakt Steckdose	1	6,99 €	6,99 €
König KNM-FTM10 Tablet Wandhalterung	1	10,55 €	10,55 €
Philips HUE Smart Plug	1	29,99 €	29,99 €
<b>Gesamtkosten</b>			<b>113,27 €</b>

Tabelle 3: Kosten Präsentationsaufbau

### 5.2.5 Gesamtkosten

Position	Kosten
Materialkosten	309,37 €
Stromkosten Druck	1,37 €
Stundenlohn	3800,00 €
Kosten Präsentationsboard	113,27 €
<b>Gesamtkosten</b>	<b>4224,01 €</b>

Tabelle 4: Gesamtkostenberechnung

## 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 agieren zu können. Der Raspberry Pi 4 ist mit einem ARM Cortex-A72 Prozessor ausgestattet, der über 4 Kerne verfügt. Zusätzlich hat der Raspberry Pi 4 über einen Gigabit-Netzanschluss. In der Makerszene ist der Raspberry Pi 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 größer 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>16</sup>

<sup>16</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.

Auf der Detailansicht der Rückseite (vgl. Abb. 7c: Detailansicht der Rückseite) erkennt man den Raspberry Pi mit dem Zigbee-Stick und der angeschlossenen Antenne auf der rechten Seite. Auf der linken Seite ist das Controller-Board des Bildschirms. Dies dient gleichzeitig als Stromversorgung für den Raspberry Pi.

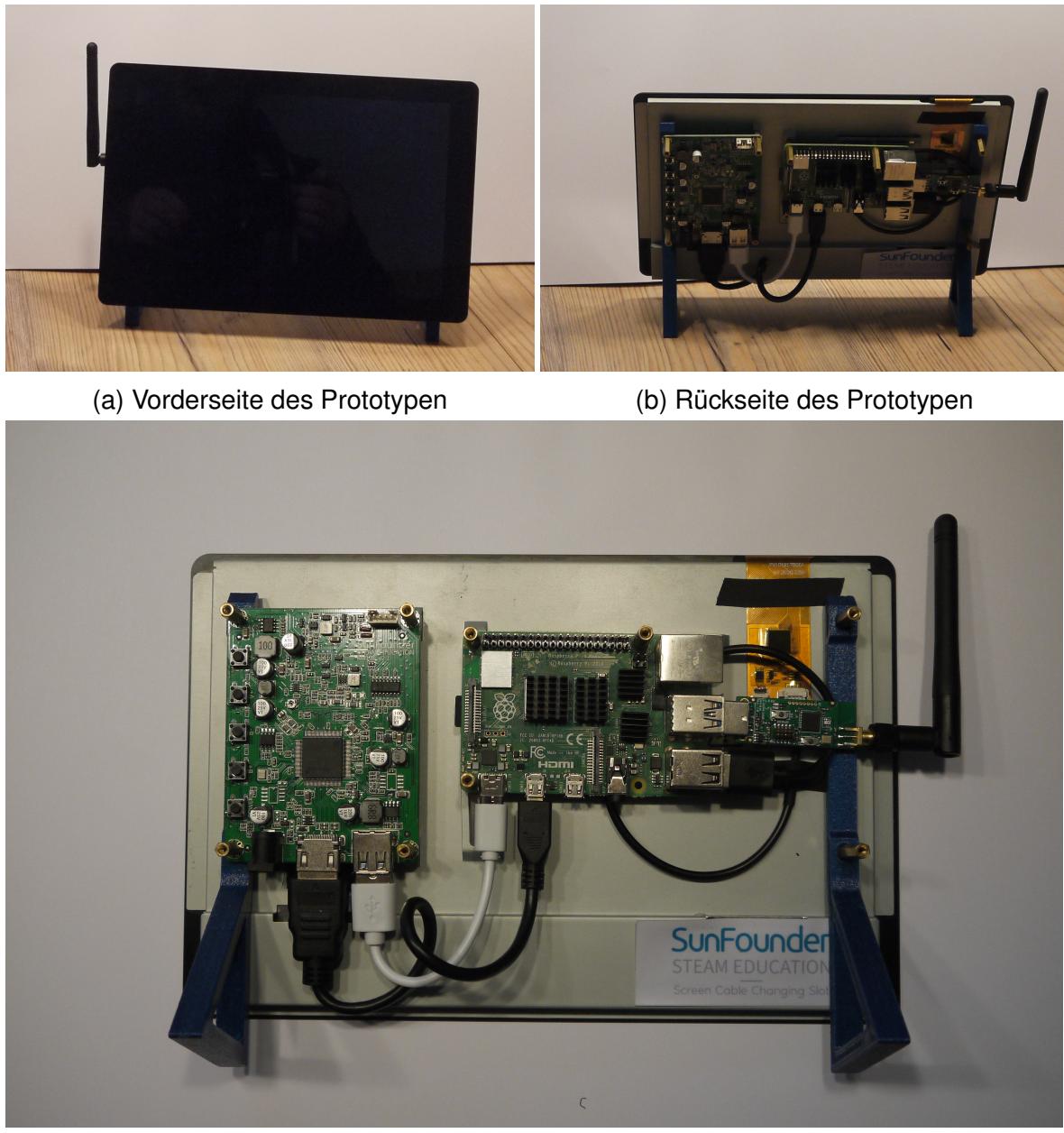


Abbildung 7: Prototyp der Smart Home Zentrale

## 6.3 Gehäuse

### 6.3.1 Erster Test zur Fertigung 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



Abbildung 8: Abmessungen Bildschirm Rückseite

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

Nach Überprüfung der Maße haben wir dann allerdings festgestellt, dass der zur Verfügung stehende 3D-Drucker, der Ender 3 Pro der Firma Creality3D, ein maximales Druckvolumen von 235 x 235 x 220 mm besitzt und somit das Gehäuse nicht wie ursprünglich geplant aus einem Stück, sondern in mehreren Teilen gedruckt werden muss.

Die ursprüngliche Konstruktion musste geändert werden. Das Gehäuse besteht nun aus vier Teilen, zwei davon bilden jeweils die Seitenwände, während zwei weitere den Deckel des Gehäuses bilden. Die Teile werden mit langen M3 Senkkopfschrauben verbunden, die zusätzlich als Verschluss des Gehäuses dienen.

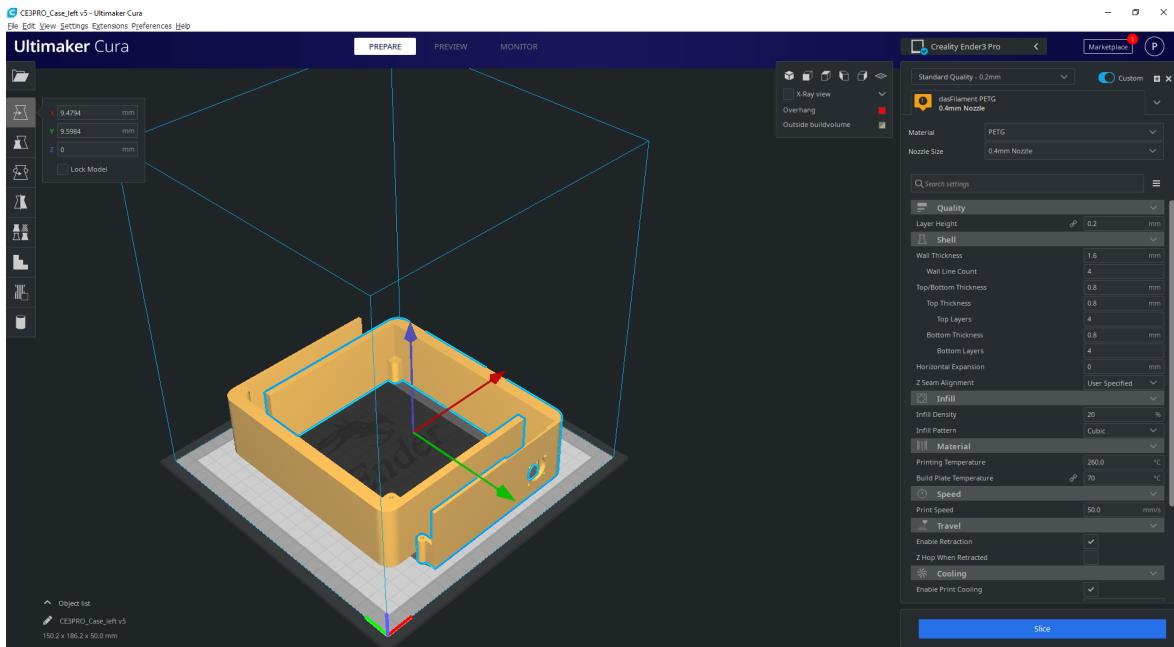


Abbildung 9: Platzierung der beiden Gehäusewände in CURA

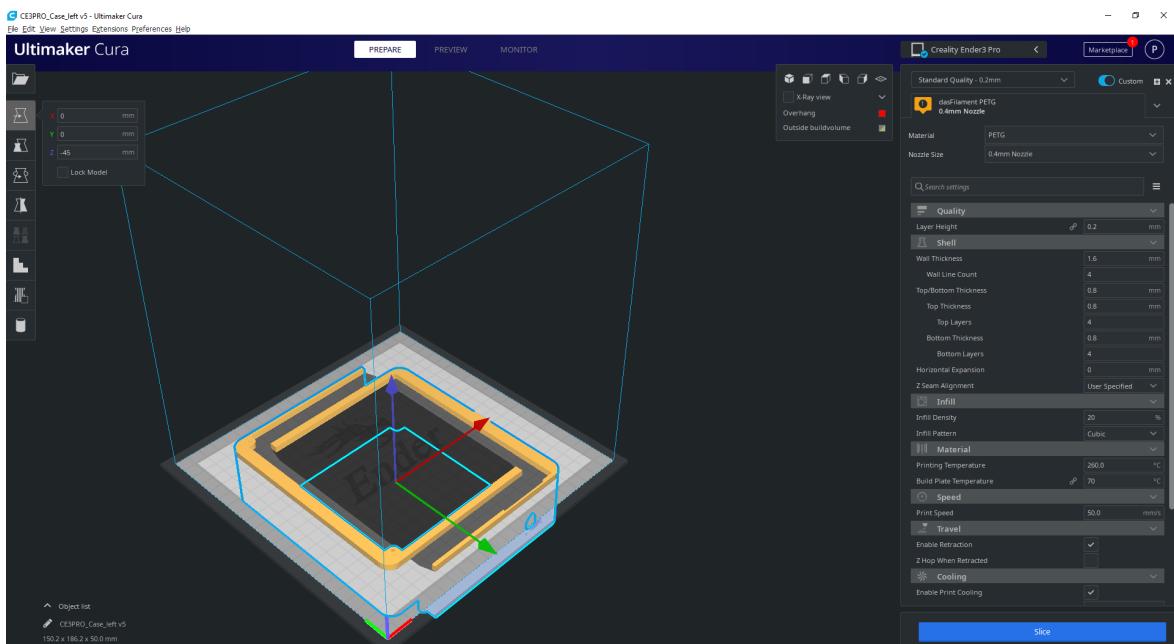


Abbildung 10: Verschiebung der Modelle entlang der Z-Achse um 45 mm

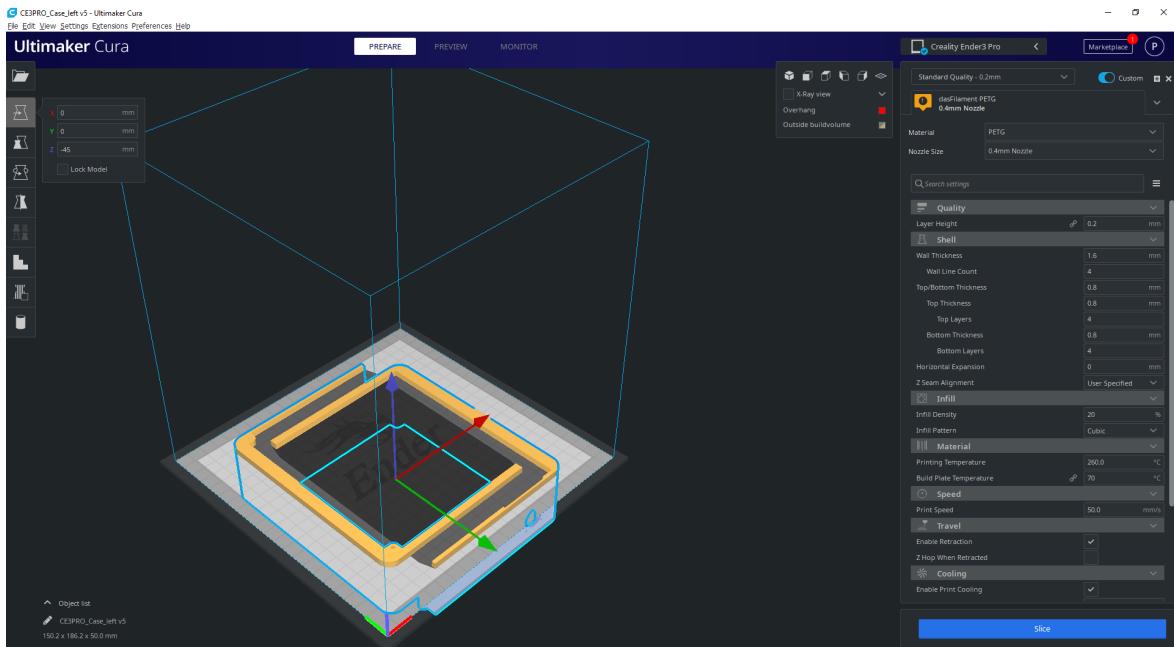


Abbildung 11: Slicen der Modelle



Abbildung 12: Anlegen des Testdrucks

Um die Genauigkeit der Konstruktion zu testen, haben wir eine 5 Millimeter hohe Schablone ausgedruckt, die am Bildschirm ausprobiert wurde. Der Ausdruck entstand mit Hilfe des Slicers CURA (vgl. Abb. 9: Platzierung der beiden Gehäusewände in

CURA), in dem die Modelle der beiden Seitenteile so angeordnet wurden, dass lediglich 5 Millimeter des Teils im druckbaren Bereich des Druckers verblieben (vgl. Abb. 10: Verschiebung der Modelle entlang der Z-Achse um 45mm). Anschließend haben wir die Teile noch auf dem Druckbett zentriert (vgl. Abb. 11: Slicen der Modelle). Nach dem Ausdrucken war klar, dass die Maße des ersten Versuchs nicht genau genug waren (vgl. Abb. 12: Anlegen des Testdrucks, weshalb wir zum Ausmessen einen digitaler Messschieber mit einer Nachkommastelle und einer Genauigkeit von  $\pm 0.2$  mm verwendeten).

### 6.3.2 Modellentwicklung am Objekt

**Grundskizze** Nachdem das Testmodell (vgl. Abb. 12: Anlegen des Testdrucks) nicht zu 100 % gepasst hat, haben wir die Abmessungen neu erfasst und diese in Fusion 360 übertragen (vgl. Abb. 10.4: Gehäuse-Zeichnungen). Um Material für den 3D-Druck zu sparen, haben wir eine Papierschablone erstellt, die Zeichnung dann im Maßstab 1:1 auf Papier ausgedruckt, ausgeschnitten und am Bildschirm angelegt. So haben wir die Maße angepasst und den Plan entsprechend ergänzt(vgl. Abb. 10.4: Gehäuse-Zeichnungen).

Die Zeichnung haben wir in zwei eigenständige Dateien gesplittet, um die linke und die rechte Seite des Gehäuses konstruieren zu können.

Die Grundabmessung des Bildschirms haben wir in Fusion 360 als neue Zeichnung angelegt. Hierzu haben wir ein einfaches Rechteck mit den entsprechenden Außenmaßen gezeichnet (vgl. Abb. 13a: Zeichnen der Außenmaße) und anschließend die Ecken mit einem Radius von 7 mm abgerundet (vgl. Abb. 13b: Abrunden der Ecken). Die Zeichnung haben wir in der Mitte geteilt, um die beiden Hälften des Gehäuses unabhängig von einander konstruieren zu können.

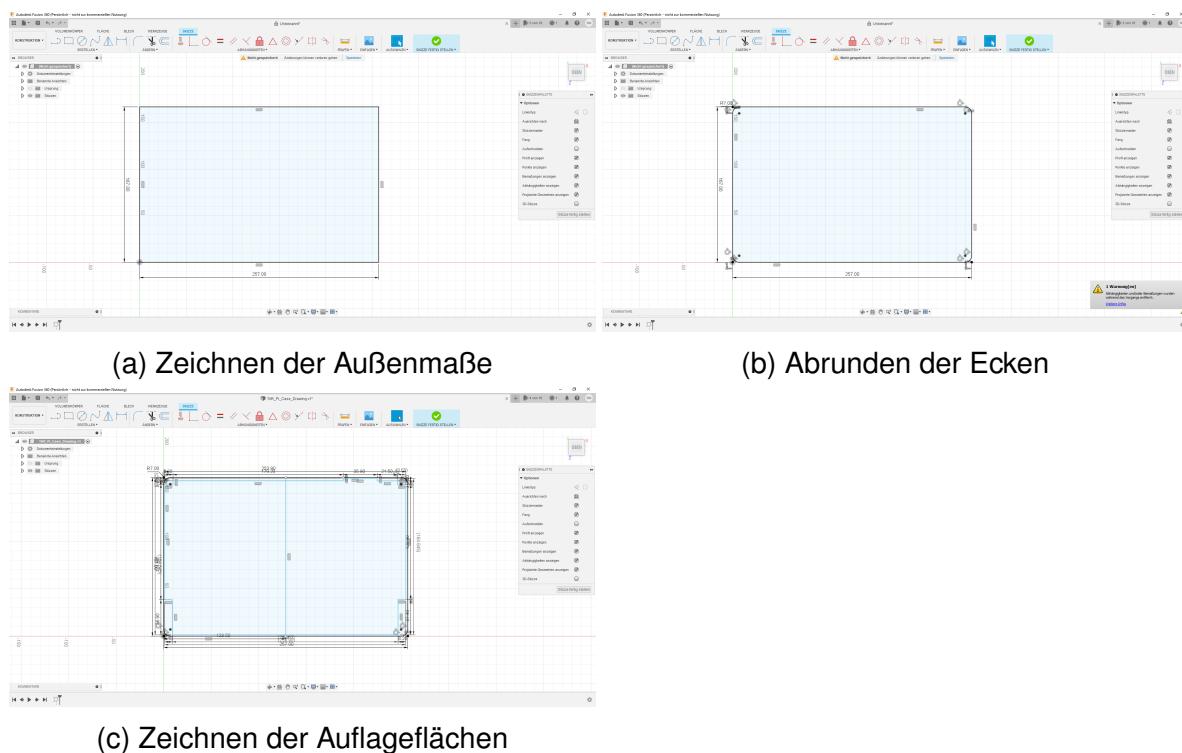


Abbildung 13: Grundzeichnung als Basis des Modells

Die so entstandene Zeichnung (vgl. Abb. 13c: Zeichnen der Auflageflächen für das Gehäuse) haben wir dann in eine weitere Datei kopiert, die als Grundlage für die beide Seitenteile dient. Ausgehend von dieser Zeichnung aus haben wir die beiden Wandteile unabhängig voneinander entworfen.

**Linkes Wandteil** Beim linken Wandteil haben wir die Zeichnung um 2 mm entlang der Z-Achse nach oben extrudiert (vgl. Abb. 14a: Extrusion der Grundzeichnung), um eine Auflagefläche zu erzeugen. Auf der Oberseite haben wir eine weitere Zeichnung gelegt, um die Kontaktfläche für die Verklebung mit dem Bildschirm zu erweitern (vgl. Abb. 14b: Erstellung der Folgezeichnung um Blech), die wir dann wie zuvor um 3 mm entlang der Z-Achse extrudiert haben (vgl. Abb. 14c: Extrusion der neuen Zeichnung) um auf die gleiche Höhe des Blechs auf der Rückseite des Bildschirms zu gelangen. Diese sollte dem Gehäuse genug Auflagefläche auf dem Bildschirm bieten, um die Verklebung so stabil wie möglich zu machen. Auf die nun entstandene Erhöhung haben wir eine Zeichnung des Rahmens von 3 mm Stärke erstellt (vgl. Abb. 14d: Zeichnung der Hauptwandstärke), die wir dann um 6 mm entlang der Z-Achse extrudiert haben, um für die Verbinder der Schrauben genügend Platz vor dem herausstehenden Blechbereich im unteren Teil des Bildschirms zu bieten (vgl. Abb. 14e: Extrusion der Hauptwand).

Auf die nun obenliegende Seite des Gehäuses haben wir die Zeichnung der Verbindungsstücke gelegt (vgl. Abbildung 14f: Zeichnung der Verbindungsstücke), um 19 mm entlang der Z-Achse extrudiert (vgl. Abb. 14g: Extrusion der Hauptwand mit Verbindungsstücken). Im Kreismittelpunkt der Zeichnungen eine Bohrung für M3-Gewinde gesetzt (vgl. Abb. 14h: Bohrung für Schrauben). Auf die Oberseite haben wir die Zeichnung für die Verbindung von Deckel und dem Seitenteil gesetzt (vgl. Abb. 14i: Zeichnung der Deckelverbindung) und um 25 mm entlang der Z-Achse extrudiert (vgl. Abb. 14j: Extrusion der Hauptwand mit Deckelverbindung), damit der von uns verwendete 40 mm Lüfter im Gehäuse Platz findet. Dies haben wir ebenfalls mit Bohrungen für M3-Gewinde versehen (vgl. Abb. 14k: Bohrungen der Deckelverbindung).

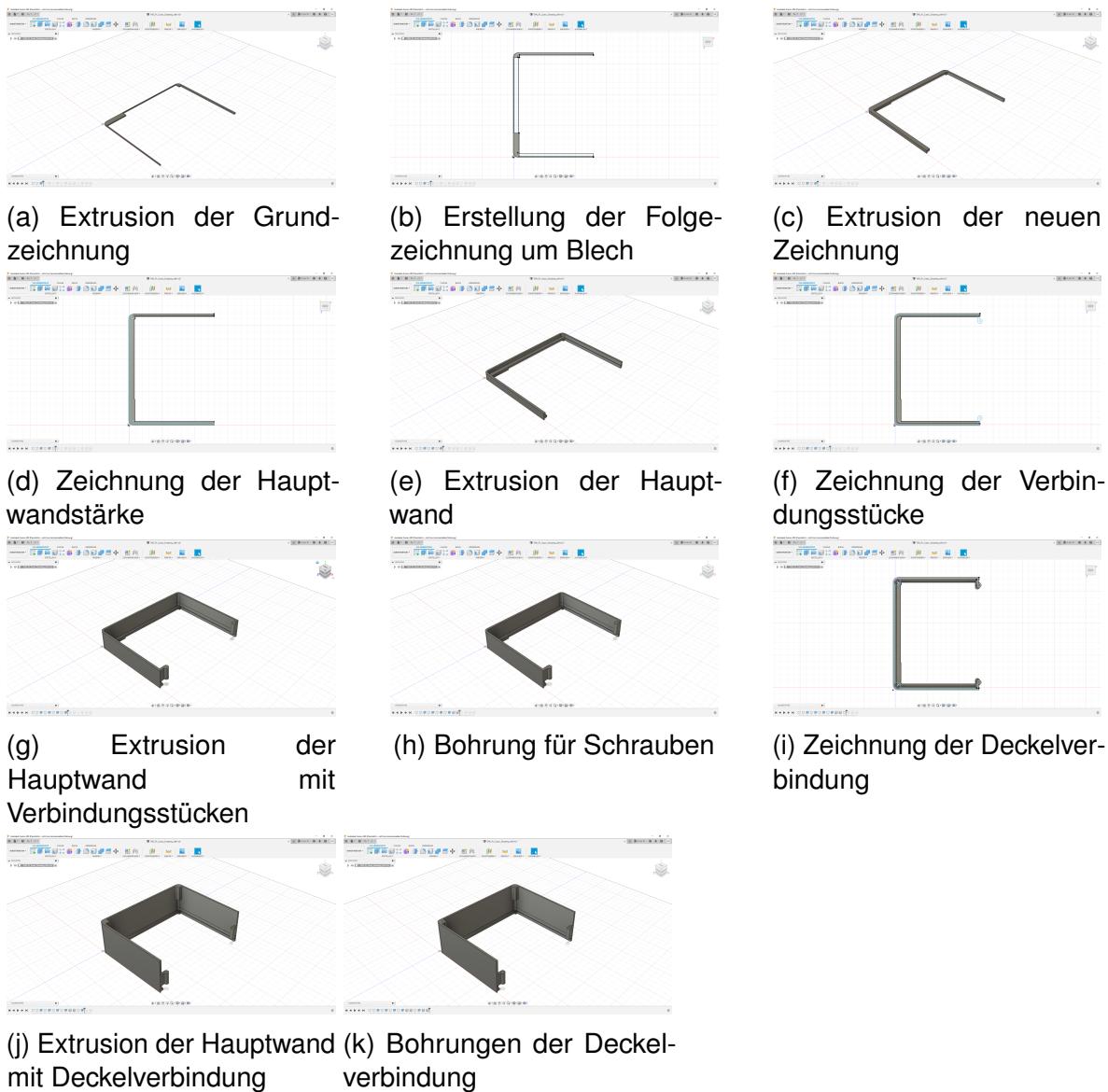


Abbildung 14: Entwurf des linken Wandteils

**Rechtes Wandteil** Beim rechten Teil des Gehäuses ist die Vorgehensweise weitestgehend die selbe wie beim linken Gehäuseteil (vgl. Abb. 15a: Extrusion der Grundzeichnung - Abb. 15h: Bohrung für Schrauben mit Abb. 14: Entwurf des linken Wandteils). Der Unterschied zwischen beiden Teilen, abgesehen von den Verbindungsstücken und der Aussparung für das Flachkabel des Bildschirms, sind zusätzlich die Aussparungen für Lüfter, Antenne, Netzwerkanschluss und Strombuchse.

Für den Lüfter und die Antenne haben wir auf dem oberen Teil des Gehäuses eine Zeichnung aufgelegt (vgl. Abb. 15i: Zeichnung für Lüfteröffnung & Antennenausgang), die wir dann ins Negative extrudiert haben, um die gezeichnete Fläche aus dem Gehäuse zu löschen (vgl. Abb. 15j: Extrusion der Lüfteröffnung). Mit der selben Vorgehensweise 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. 15k: Zeichnung der Aussparung für Stromanschluss & Netzwerkbuchse) und diese dann ebenfalls ins Negative extrudiert (vgl. Abb. 15l: Extrusion der Netzwerkbuchse).

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. 15m: Zeichnung des Stromsteckerblocks) und extrudiert (vgl. Abb. 15n: Extrusion des Stromsteckerblocks), um die Möglichkeit zu haben, die Buchse mit dem 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. 15o: Zeichnung der Aussparung) und dann um einige Millimeter ins Negative verkürzt, um die Aussparung zu generieren (vgl. Abb. 15: Entwurf des rechten Wandteils).

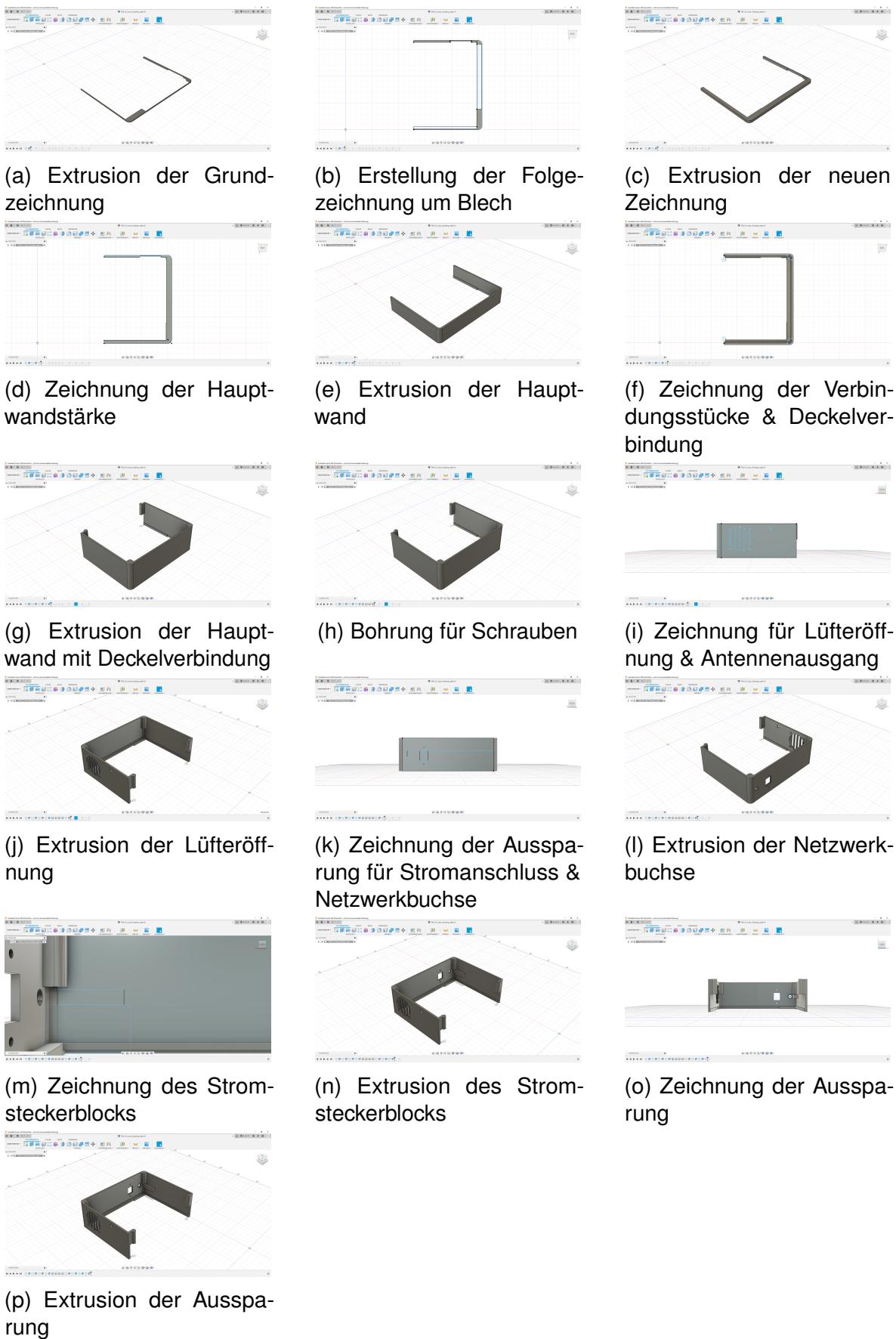


Abbildung 15: Entwurf des rechten Wandteils

**Gehäuserückwand** Das Design der Gehäuserückwand basiert nur rudimentär auf der in Abb. 13: Grundzeichnung als Basis des Modells 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. 16a: Grundrisszeichnung), welche den Zeitaufwand für den Druck von zwei unterschiedlichen Teilen für die Rückseite reduzieren soll.

Nachdem wir die Zeichnung extrudiert haben (vgl. Abb. 16b: Extrusion der Grundform), haben wir noch einen Vorsprung auf der kurzen Seite des Bauteils extrudiert (vgl. Abb. 16c: Extrusion des Vorsprungs), um die beiden Rückwandteile besser verbinden zu können. Anschließend haben wir ein weiter Teil der inneren Zeichnung ins Negative extrudiert, um diesen Bereich freizustellen (vgl. Abb. 16d: Erstellen der Aussparung) und um beim Druck der Teile Material zu sparen.

Damit für eventuelle Toleranzen zwischen den beiden Teilen Platz ist, haben wir auf dem Vorsprung die Aussparung für die darunterliegende Schraubendurchführung erhöht (vgl. Abb. 16e: Anpassung für Toleranz) und den Vorsprung von unten mit einer Fase versehen, was den Materialverbrauch für das Teil weiter reduziert (vgl. Abb. 16f: Abfasen des Vorsprungs). Zudem haben wir die obere Kante (vgl. Abb. 16g: Abfasen der Außenseite), die Innenseite (vgl. Abb. 16h: Anbringung der Fase an der Innenseite) und auch den Vorsprung (vgl. Abbildung 16i: Anbringung der Fase am Vorsprung) mit einer Fase versehen.

Um die Bohrungen an die richtige Stellen setzen zu können, haben wir eine Hilfszeichnung auf die Außenfläche, also die Außenseite der Rückwand, aufgelegt (vgl. Abb. 16j: Hilfszeichnung für Bohrungen). Dort haben wir dann drei M3-Bohrungen für Senkkopfschrauben gesetzt (vgl. Abb. 16k: Bohrungen mit Aussparung für Senkkopfschrauben).

Für die Aufhängungen auf der Rückseite des Gehäuses haben wir eine weitere Zeichnung aufgelegt (vgl. Abb. 16l: Zeichnung für Wandaufhängungslöcher) und dann ins Negativ extrudiert (vgl. Abb. 16m: Extrusion der Wandaufhängungslöcher). So kann das Gehäuse mit zwei Schrauben an der Wand angebracht werden. Um beim Aufhängen nicht an einen bestimmten Typ Schrauben gebunden zu sein, haben wir eine kleine Fase an den engeren Teil der Aufhängungen gelegt (vgl. Abb. 16n: Abfasen der Wandaufhängungslöcher).

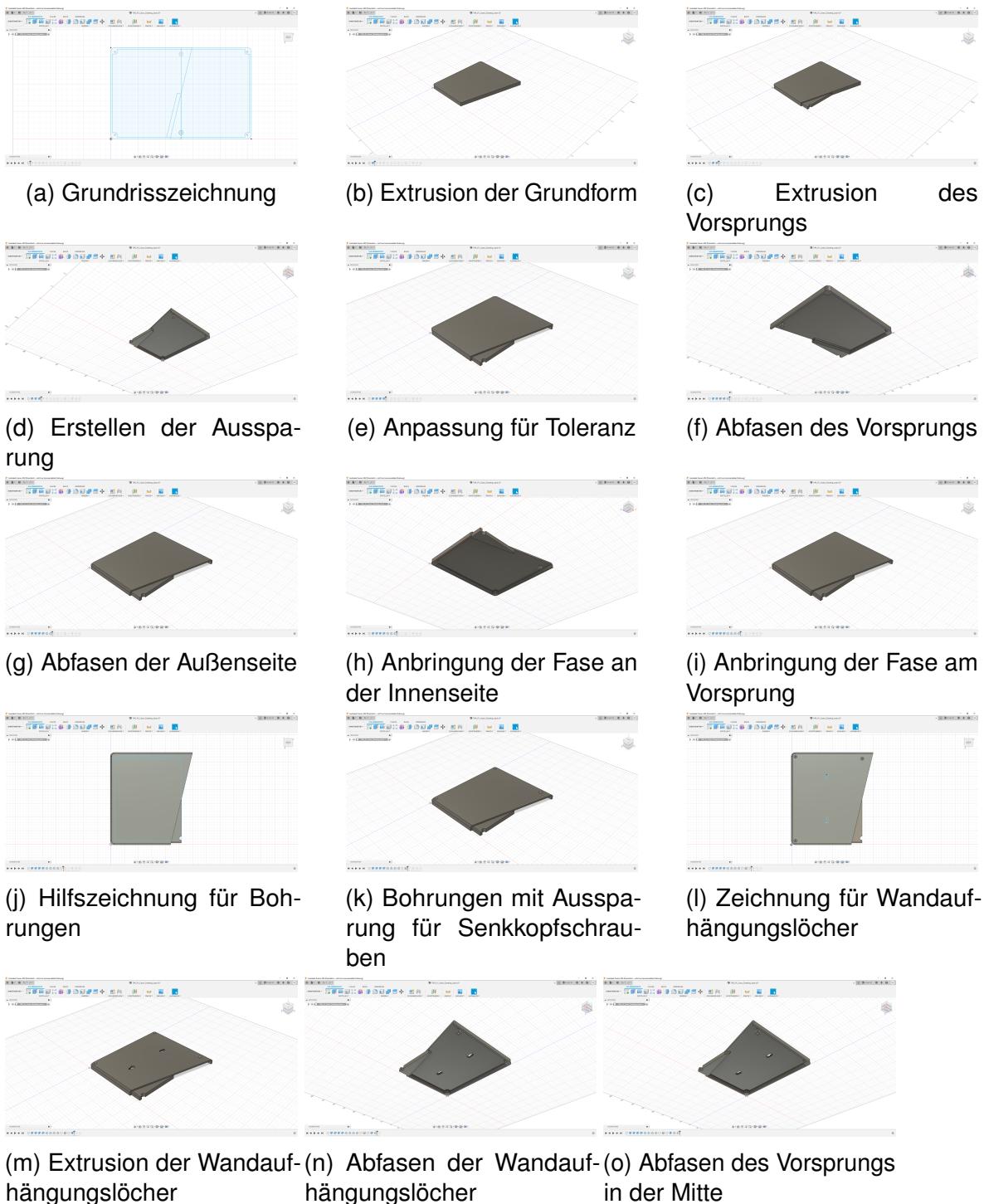


Abbildung 16: Entwurf der Gehäuserückwand

### 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 (vgl. Abb. 17: 3D-Drucker Ender 3 Pro (mod)) druckt die zuvor erstellten STL-Dateien mit PETG-Filament der Firma dasfilament. Der für den 3D-Druck nötige G-Code haben wir mit Ultimaker CURA generiert und mit Hilfe von OctoPrint (vgl. Abb. 18: OctoPrint-Weboberfläche) 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)

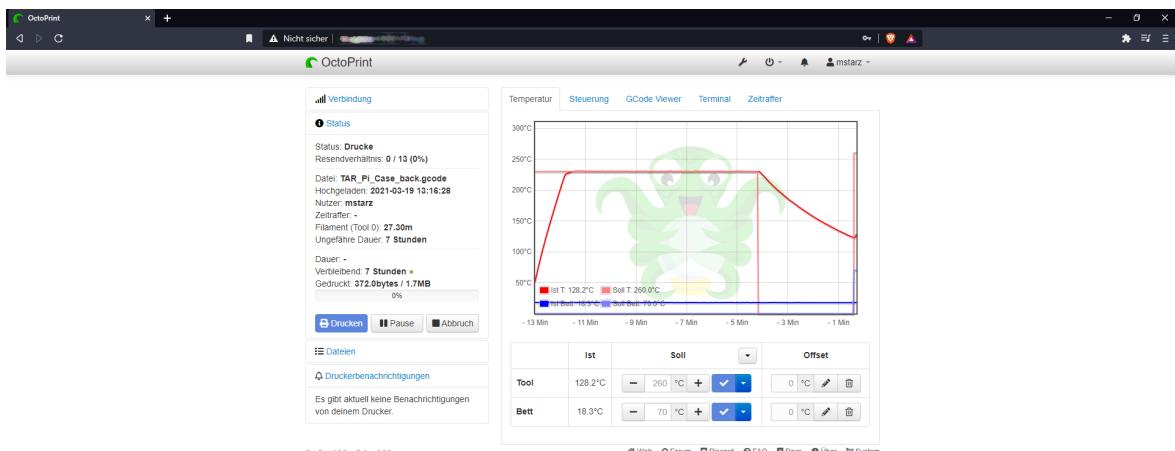
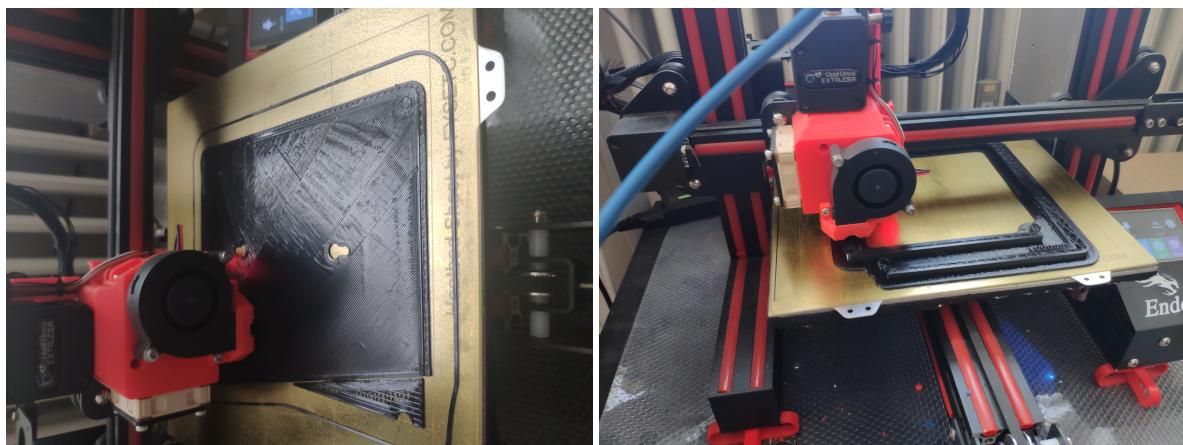


Abbildung 18: OctoPrint-Weboberfläche

Die gedruckten Teile (vgl. Abb. 20: Ausgedruckte Teile des Gehäuses) werden dann zum Teil mit Zwei-Komponenten-Epoxid-Kleber verbunden (vgl. Abb. 21a: Verklebung des Gehäuses) und mit Hilfe von Schleifpapier und einigen Schichten Klarlack zu einem klavierlackähnlichen Finish veredelt (vgl. Abb. 21b: Abgeschliffene und

lackierte Teile).

Die Seitenwände werden dann mit dem Bildschirm mit Hilfe des Zwei-Komponenten-Epoxidklebers und Heißkleber permanent verklebt (vgl. Abb. 21c: Gehäuse und Bildschirm verklebt (von vorne)). Danach wurden die Platinen und Kabel im Gehäuse angebracht (vgl. Abb. 21c: Gehäuse und Bildschirm verklebt (von vorne)). 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

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. 20b: Linke Seite) 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>17</sup> Teilebibliotheken sind ebenso 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>18</sup>. Als Grundlage für den HAT sollte die Projektvorlage „Raspberry Pi - 40-Pin HAT“ von Jon Buford dienen. Diese Vorlage richtete sich nach den offiziellen HAT-Spezifikationen für den Raspberry Pi<sup>19</sup>. Zu Beginn des Projekts haben wir in KiCAD die GPIO-Ports anhand der offiziellen Dokumentation beschriftet, um das Nachschlagen der Belegung für die Zukunft zu vermeiden.

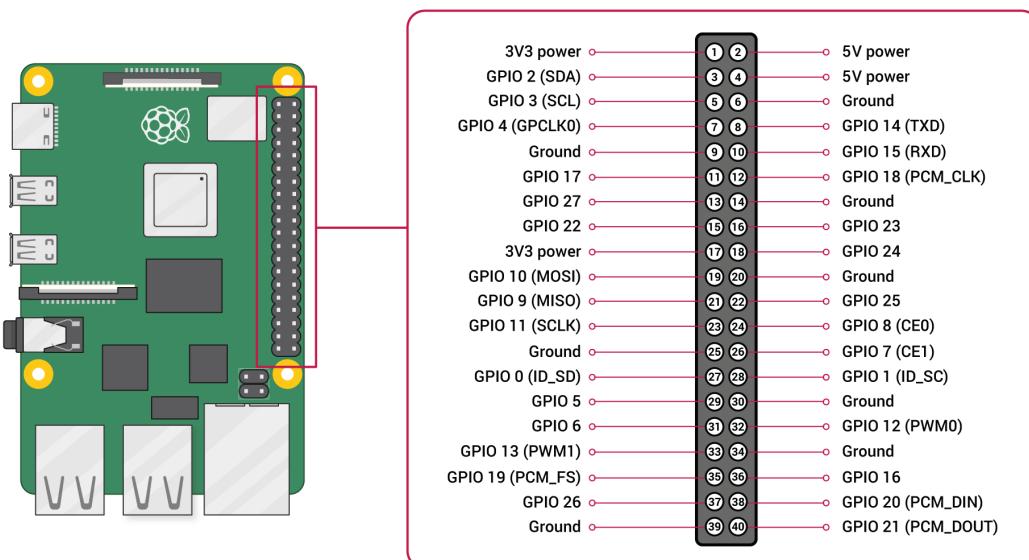


Abbildung 22: Raspberry Pi 4 GPIO-Pins

Anschließend haben wir die (unserer Meinung nach) benötigten Komponenten auf den Schaltplan gezogen beziehungsweise vorher importiert und dann diese sauber verdrahtet (vgl. Abb. 23: Aktueller Stand HAT-Design).

<sup>17</sup> Siehe <https://www.kicad.org/download/>

<sup>18</sup> Siehe <https://www.digikey.de/de/resources/design-tools/kicad>

<sup>19</sup> Siehe <https://github.com/raspberrypi/hats>

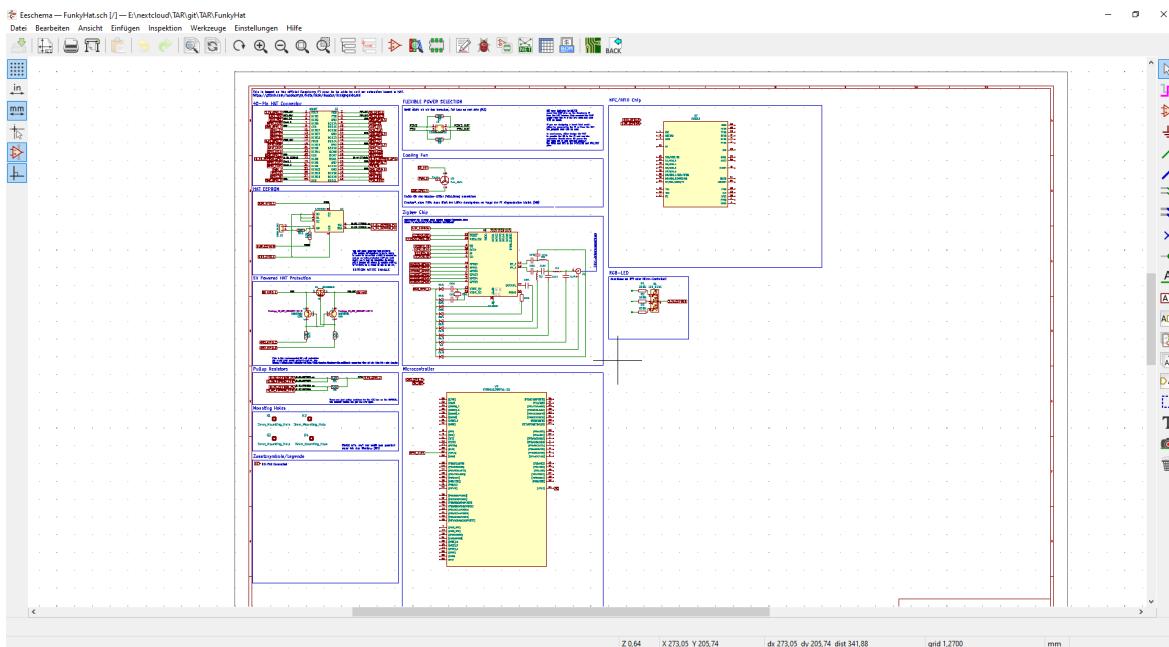


Abbildung 23: Aktueller Stand HAT-Design

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 HAT's den Zeitrahmen sprengen würde und so haben wir Ende Februar 2021 beschlossen, dieses Projekt vorerst zurückzustellen und uns auf die Fertigstellung unserer bisherigen Testerfolge und Prototypen zu fokussieren. Die KiCAD-Projekt-Dateien stehen, wie alle anderen Dateien des Projekts nach Abgabe der Technikerarbeit frei über das entsprechende GITHUB zur Verfügung.

## 6.5 Präsentationsaufbau

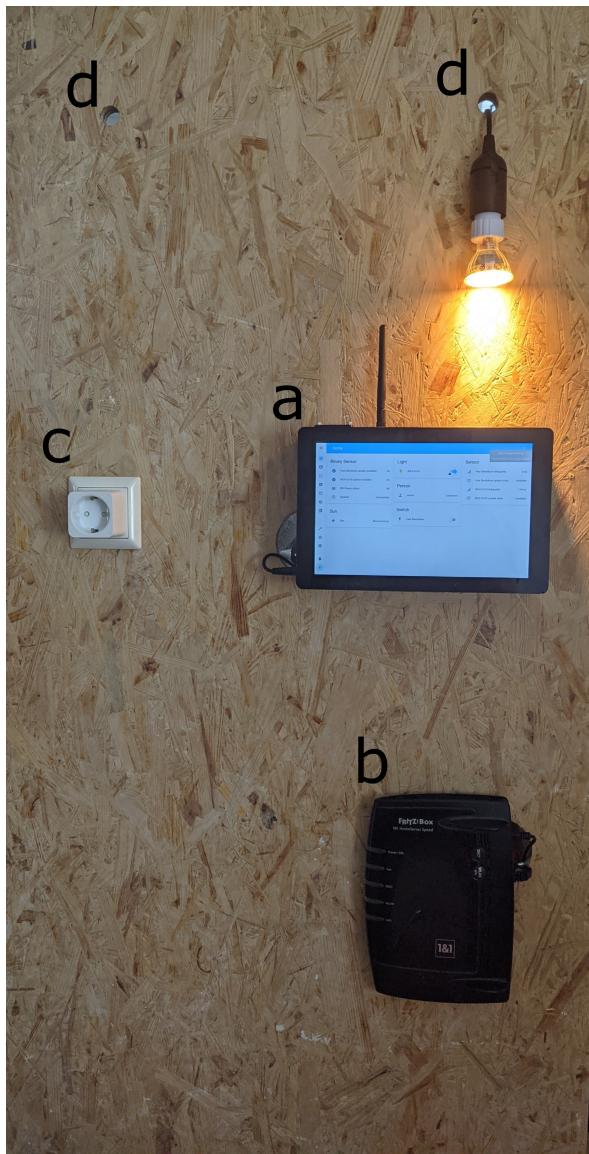


Abbildung 24: Präsentationsaufbau

Um die Ergebnisse unseres Technikerarbeit und unserem Prototypen präsentieren zu können, haben wir einen Präsentationsaufbau auf Basis einer OSB-3-Verlegeplatte gebaut (vgl. Abb. 24: Präsentationsaufbau). Dieser ist neben der Smart Home Zentrale mit einem Router, zwei Zigbee-fähigen Glühbirnen, einer Phillips Hue Zigbee Steckdose und einem Amazon Fire8 HD Tablet ausgestattet. Der Aufbau soll die Funktion der Smart Home Zentrale in der Steuerung von Smart Home Komponenten praktisch aufzeigen und wurde von uns selbst gebaut.

## 7 Software

### 7.1 Überblick

Auf der Softwareseite dient Raspberry Pi OS<sup>20</sup> als Basis für die Smart Home Zentrale. Darauf können die gewählten Softwarelösungen verwendet werden:

- 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. 10.2) erstellt, dass das Betriebssystem entsprechend vorbereitet und den 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, weshalb wir uns für die Verwendung des Add-on Shops entschieden haben.

### 7.2 Installationsskript

Das Installationsskript (s. Anhang 10.2: Installationsskript) 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.

Das Installationsskript des Home Assistant Supervisors verfügt über viele Ausgaben, sowie eine nicht mit einem Parameter automatisierbare Abfrage. Aus diesem Grund haben wir das Skript in unser Installationsskript kopiert, 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 eine Erfolgsmeldung.

Die einzelnen Aufgaben, die das Installationsskript durchführt, wurden in eigene Funktionen ausgegliedert. Dadurch lassen sich am Skript selbst schneller Änderungen durchführen. Das Skript erkennt dabei 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 abzulehnen (vgl. 10.2: Installationsskript Zeile 34 ff). Gibt der Nutzer keinen Operanden oder einen ungültigen Operanden an, wird ein Hilfetext mit der Funktion `show_help` (vgl. 10.2: Installationsskript Zeile 34ff) ausgegeben, der den Nutzer über die Operatoren, sowie eine allgemeine Verwendung des Skripts informiert.

<sup>20</sup> Raspberry Pi OS with Desktop, Kernel Version 5.4

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 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>21</sup> durchführt.

Ob die oben genannten Abhängigkeiten vorhanden sind, wird bei der Installation unabhängig von der Nutzereingabe geprüft und gegebenenfalls bei Fehlen dieser nachinstalliert (vgl. 10.2: Installationsskript Zeile 85ff & 10.2 Zeile 297ff: Installationsskript).

Sollte das Skript ausgeführt werden, ohne das Docker auf dem Raspberry Pi installiert ist und nur die Installation des Home Assistant Supervised angefordert wird, wird Docker mit installiert (vgl. 10.2 Zeile 284ff: Installationsskript). 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>22</sup>.

Für Docker wird ein Skript von `docker.com` heruntergeladen und ausgeführt (vgl. 10.2: Installationsskript Zeile 108ff). Der Abschnitt der Home Assistant Installation basiert auf dem Bash-Skript von Home Assistant selbst<sup>23</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>21</sup> GITHUB: `supervised-installer` <https://github.com/home-assistant/supervised-installer>

<sup>22</sup> Für das Systemupdate und die Installation der Abhängigkeiten.

<sup>23</sup> GITHUB: `installer.sh` <https://github.com/home-assistant/supervised-installer/blob/master/installer.sh>

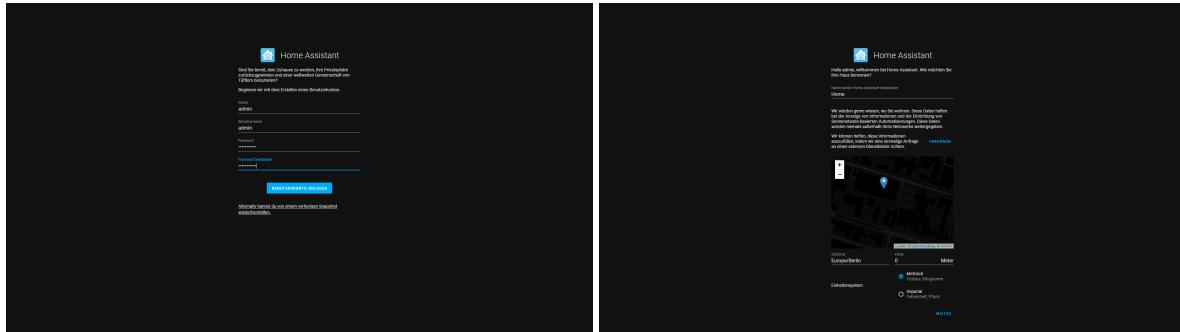
Eine Übersicht und Kurzerklärung der Funktionen im Skript:

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 Home Assistant 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 Komplettinstallation	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 5: Skript-Funktionen

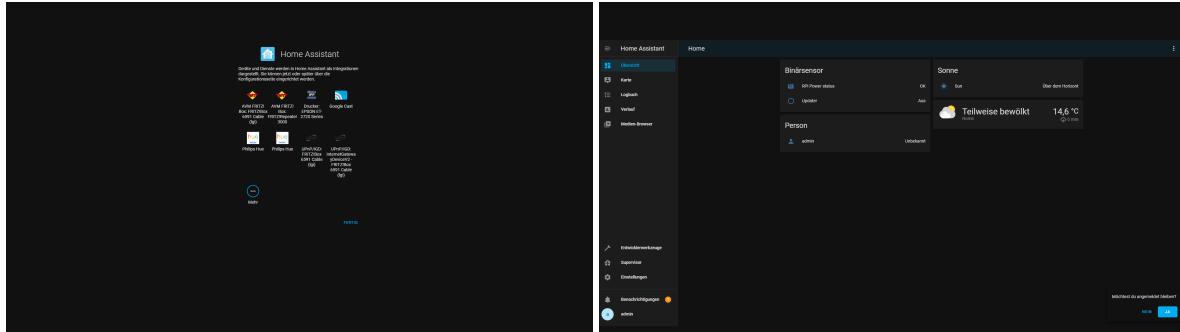
### 7.3 Home Assistant Supervised

Nachdem das Installationsskript (vgl. Abschnitt 10.2: Installationsskript) fehlerfrei Home Assistant Supervised installiert hat, können wir mit der Einrichtung über die Weboberfläche 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 (vgl. Abb. 25a: Anmeldebildschirm) in welchen wir unseren Benutzeraccount mit Passwort für Home Assistant anlegen. Anschließend legen wir den Namen und den Standort des Home Assistant fest (vgl. Abb. 25b: Festlegen von Name und Standort). Der Standort dient der Ermittlung von Wetterdaten und wird für das Geofencing benötigt. Auf dem nächsten Bildschirm (vgl. Abb. 25c: Mögliche Integrationen werden erkannt) können wir bereits vorhandene Smart-Home-Systeme wie Google Cast und Phillips Hue integrieren. Da wir allerdings eine eigene Integration von Zigbee Leuchtmitteln anstreben, überspringen wir diesen Schritt. Nun leitet uns der Home Assistant auf seine Startseite weiter (vgl. Abb. 25d: erstes Dashboard). Dieses Dashboard wird von Home Assistant aus allen ihm bereits bekannten Geräten bzw. Entitäten generiert.



(a) Anmeldebildschirm

(b) Festlegen von Name und Standort



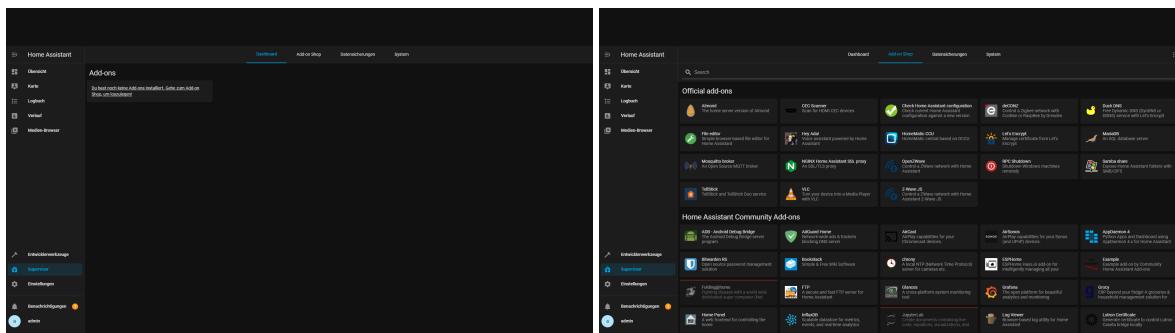
(c) Mögliche Integrationen werden erkannt

(d) erstes Dashboard

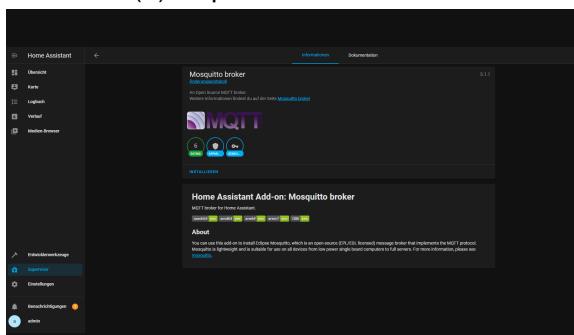
## 7.4 Mosquitto Broker

Als erste wichtige Erweiterung benötigen wir für unseren Home Assistant einen MQTT Broker. Da wir eine Supervised Version des Home Assistant auf unserem Pi verwenden, nutzen wir für die Installation den Supervisor Add-on Shop von Home Assistant (vgl. Abb. 26e: Startseite Integrationen ).

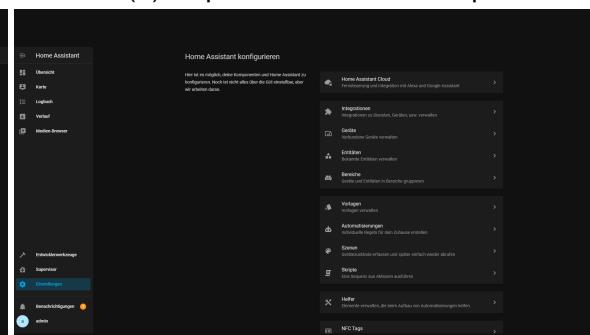
Hier wählen wir den Mosquitto Broker zur Installation aus (vgl. Abb. 26f: Suchfunktion Integrationen ). Um den Mosquitto Broker zu aktivieren, navigieren wir über den Menüpunkt „Einstellungen“ zum Punkt „Integrationen“ und suchen dort nach MQTT. Wir aktivieren die Verknüpfung von Home Assistant und Mosquitto Broker, indem wir die Schaltfläche „Suche aktivieren“ klicken. Nach Durchführung dieser Schritte ist der Mosquitto Broker aktiv.



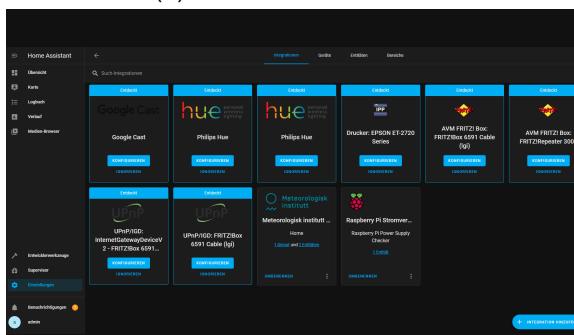
#### (a) Supervisor Dashboard



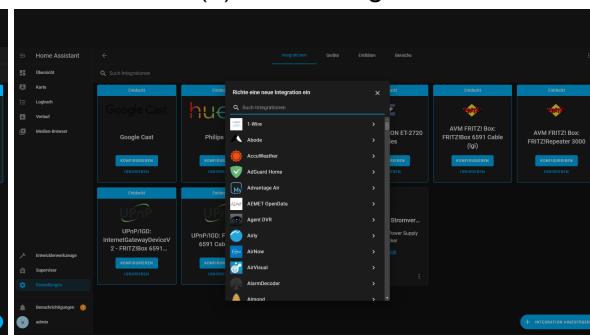
(c) MQTT add-on Seite



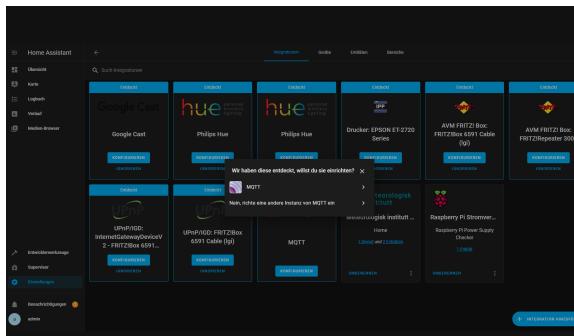
(b) Supervisor Add-on Shop



### (e) Startseite Integrationen



### (f) Suchfunktion Integrationen



(g) Suche nach MQTT

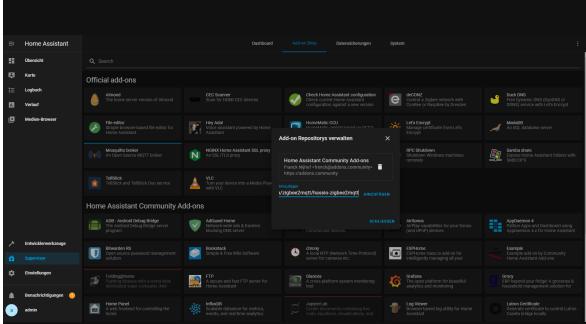
## 7.5 Zigbee2MQTT

Um das offizielle Zigbee2MQTT Home Assistant Add-on nutzen zu können, müssen wir zunächst im Supervisor Add-on Shop (vgl. Abb. 27a: Repository in Add-on Shop einpflegen) das Repository<sup>24</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 (vgl. Abb. 27b: Zigbee2MQTT Add-on Seite). An diesem Punkt bietet sich uns die Möglichkeit, die Zigbee2MQTT „configuration.yaml“ einzusehen (vgl. Abb. 27c: Configuration.yaml und Anhang 10.3.1: Home Assistant Konfiguration).

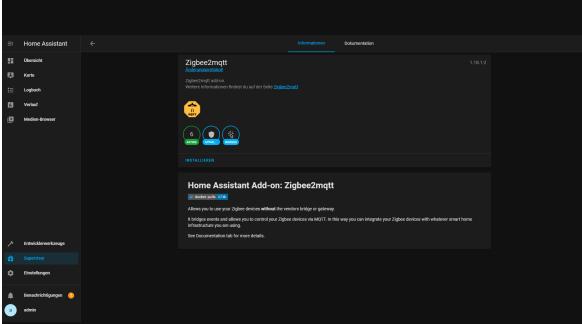
Hier ist darauf zu achten, dass der Parameter von „permit-join“ lediglich für das Einlernen von neuen Zigbee-Geräten auf „true“ gesetzt wird. Um einen sauberen ersten Start von Zigbee2MQTT gewährleisten zu können, muss nun noch der „Serial Port“ des Zigbee Adapters überprüft werden. Zu diesem Zweck navigieren wir über „Supervisor“ (vgl. Abb. 27d: Supervisor Systeminformationen) zu „System“ (vgl. Abb. 27d: Supervisor Systeminformationen) und dort unter „Host“ zu „Hardware“. In diesem Fenster sehen wir sämtliche Hardware, die mit dem Host-System, also in unserem Fall dem Raspberry Pi, verbunden ist (vgl. Abb. 27d: Supervisor Systeminformationen). Nachdem sicher gestellt ist, dass der „Serial Port“ mit dem tatsächlichen Port übereinstimmt, können wir das Add-on starten (vgl. Abb. 27f: Aktiviertes Zigbee2MQTT Add-on).

Das Einlernen von neuen Leuchtmitteln und Geräten gestaltet sich, sofern die Datei „configuration.yaml“ (s. Anhang 10.3.3: Zigbee Geräteauflistung) beachtet wurde, deutlich einfacher. Zuerst muss das Leuchtmittel oder Gerät nach Herstellerangabe zurückgesetzt werden. Danach wird das Gerät automatisch von Zigbee2MQTT erfasst und eingelernt. Eingelernte Geräte werden in der Datei „devices.yaml“ (vgl. 10.3.3: Zigbee Geräteauflistung) gespeichert und sind dadurch als Entitäten in Home Assistant steuerbar.

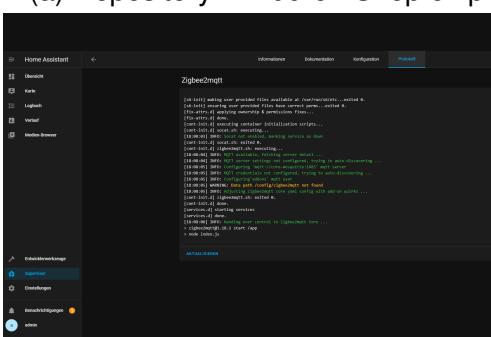
<sup>24</sup> GITHUB: <https://github.com/zigbee2mqtt/hassio-zigbee2mqtt>



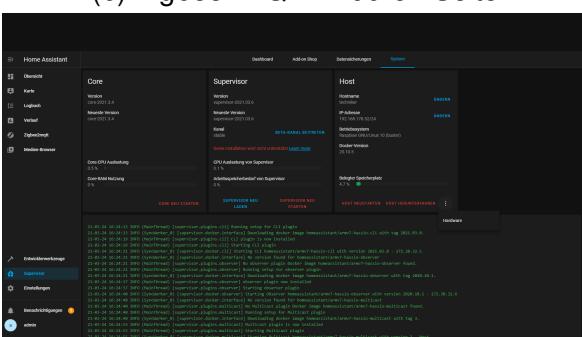
(a) Repository in Add-on Shop einpflegen



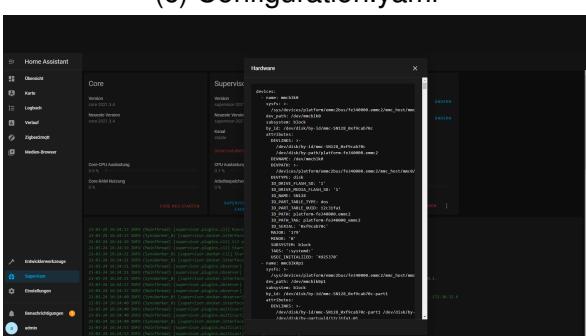
(b) Zigbee2MQTT Add-on Seite



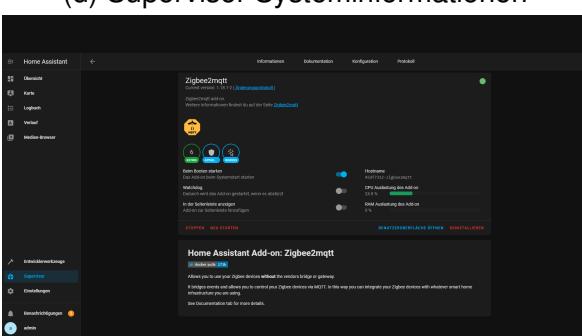
(c) Configuration.yaml



(d) Supervisor Systeminformationen



(e) Auflistung der Hardware



(f) Aktiviertes Zigbee2MQTT Add-on

## 7.6 Erstellung des Betriebssystem-Image

Nach erfolgreichem Test des Installationsskripts<sup>25</sup> haben wir für je einen Raspberry Pi 3 und einen Raspberry Pi 4 ein Image erstellt.

Hierzu haben wir mit Hilfe des Raspberry Pi Imagers eine 8 Gigabyte große Micro-SD-Karte<sup>26</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. 28a). Dementsprechend wurden für die beiden Versionen folgende Einstellungen vorgenommen:

<sup>25</sup> Siehe Abschnitt 10.2: Installationsskript

<sup>26</sup> Zum Zeitpunkt der Erstellung des Images kleinstmögliche vorhandene Micro-SD-Karte

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

Tabelle 6: Überblick über die Images für die Raspberry Pis

Nachdem das 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.

Der Shell-Befehl hierfür lautet:

```
ssh pi@<Domänename>
```

. Mit der Verwendung des Standardpassworts haben wir uns mit den Raspberry Pis verbunden (vgl. Tab. 6: Überblick über die Images für die Raspberry Pis).

Anschließend haben wir das Skript auf dem Pi eingerichtet und gestartet (vgl. 28b). Nach der Ausführung des Skriptes (vgl. 28c) wurde die SD-Karte mit Win32 Disk Imager in eine .img-Datei konvertiert (vgl. 28d). Anschließend wurde das Image mit 7Zip komprimiert, um Speicherplatz zu sparen (vgl. 28e). Diese Images können dann mit dem Raspberry Pi Imager<sup>27</sup> wieder auf SD-Karten geflasht werden.

Die Images<sup>28</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>27</sup> Alternative: balnea Etcher

<sup>28</sup> Stand: Ende März 2021

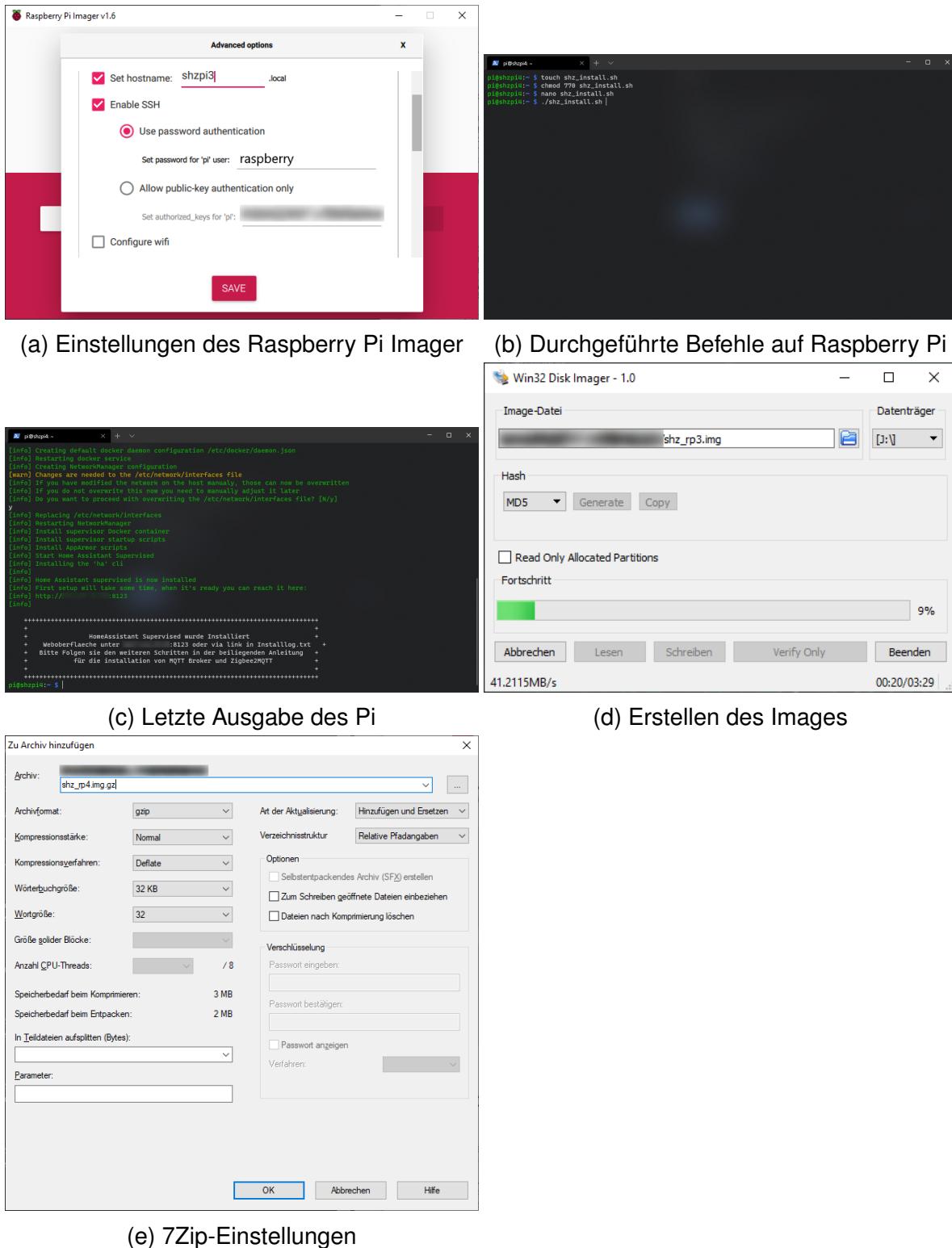


Abbildung 28: Erstellen der Images

## 8 Epilog & Fazit

Zum Abschluss möchten wir sagen, dass wir nach knapp fünf Monaten eine recht ansehnliche Lösung für Smart Home Besitzer entwerfen konnten. Dieses Projekt ist in dieser Zeit zu einer Herzensangelegenheit von uns beiden geworden und wird noch einige Zeit über die Abgabe dieser Technikerarbeit hinaus von uns erweitert und betreut.

Unser Ziel wird es sein, das ganze Projekt als ein Baukastensystem für den technisch versierten Smart Home Besitzer, als auch für den Anfänger im Smart Home Bereich auf den Markt zu bringen.

### 8.1 Fehler & Probleme

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

- L<sup>A</sup>T<sub>E</sub>Xist toll
- Ein Leerzeichen kann über einige Stunden Arbeit entscheiden
- Bei einem Test einer Version des Installationsskripts haben wir stundenlang nach einer Lösung gesucht, warum die Installation des Home Assistant fehlgeschlug, bis wir feststellten, dass unser Testsystem lediglich über WLAN mit unserem Netzwerk vergebunden war. Der Network-Manager aber während der Installation neu gestartet wurde, wodurch die WLAN Verbindung neu startete und unsere SSH Verbindung zum PI abbrach. Darüber hinaus war das Installationsskript nicht mit der langen Wiederverbindungszeit im WLAN klargekommen.
- Stützstrukturen können auch mal Spaghetti produzieren
- Da wir uns auch als Ziel gesetzt haben, auf einem Raspberry Pi sowohl Desktop Umgebung als auch Home Assistant zu installieren, disqualifizierte sich von Anfang an das Home Assistant OS Image. Es gab für uns also keine andere Möglichkeit, als Home Assistant auf unserem Raspberry Pi nachzuinstallieren. Nach einigen Fehlschlägen in der Einrichtung von Home Assistant Core, Mosquitto broker und Zigbee2MQTT entschieden wir uns zunächst für eine Installation der Komponenten in Docker Containern. Als wir aber auch hier auf nicht durchgehend reproduzierbare Erfolge stießen und zeitgleich auch eine Integration des Home Assistant Supervisor anstrebten, fiel unsere finale Wahl auf die Installation des Home Assistant Supervised Docker Container, der von Home Assistant für Fortgeschrittene Nutzer über GitHub angeboten wird. Dies ermöglichte uns die Nutzung der offiziellen MQTT und Zigbee2MQTT Add-ons. Um dennoch unser Ziel einer vereinfachten Installation erfüllen zu können, schrieben wir ein Installationsskript, welches benötigte Programme und Konfigurationen automatisch installiert. Außerdem erstellten wir zwei fertige Images, die ein Neuaufsetzen des Betriebssystems mit vorinstallierten Home Assistant Supervised Containers deutlich beschleunigen.

## 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 herumquälen durften.

## 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)  
<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](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)
- MQTT.org (2021): What is MQTT  
<https://mqtt.org/faq/>
- mosquitto.org (2021): Eclipse Mosquitto<sup>TM</sup> An open source MQTT broker  
<https://mosquitto.org/>
- Koen Kinters (2021): Zigbee2MQTT  
<https://github.com/koenkk/zigbee2mqtt#internal-architecture>
- Deutscher Gewerkschaftsbund (2021): Mindestlohn 2021/2022: Was ändert sich?  
<https://www.dgb.de/themen/++co++6ca263de-fb0e-11e9-bdcf-52540088cada#wiehoch2021>

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 Nachrichtenstandard	5.1.1
Zigbee2MQTT	Anbindung von Zigbee an MQTT	1.18.1-2
KiCad EDA	Erstellung von Schaltplan und Gerber-Datei des HAT's	5.1.8
TexMaker	Erstellung der Dokumentation	5.0.4
<a href="https://www.overleaf.com">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 Images aus den Micro-SD-Karten	1.0
7Zip	Komprimieren der Images	19.00
<a href="https://shellcheck.net/">shellcheck.net/</a>	Überprüfung von Bash-Skripten	Web
<a href="https://deepl.com/translator">deepl.com/translator</a>	Übersetzungssoftware	Web
Microsoft Teams	KANBAN Board und die Kommunikation	1.4.00.7174

Tabelle 7: 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 8: 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

```
1  #!/bin/bash
2  #####
3  # Installationsskript für Smart Home Zentrale #
4  #
5  # (c) Felix Kuschel & Manuel Starz           #
6  #####
7  # Skript installiert Docker, Home-Assistant   #
8  # und MQTT-Broker uns Zigbee2Mqtt            #
9  #####
10 # Version: 1.0                                #
11 #####
12 # Variablen für Skript
13 # Version des Pi (3 oder 4)
14 PIVER=$(cat /proc/device-tree/model | cut -d' ' -f3)
15 # Update-Flag
16 UPDATE=0
17 # Docker-Installationsflag
18 DOCKER=0
19 # Homeassistant-Flag
20 HOMEASSISTANT=0
21 # Dependencies-Flag
22 DEPENDENCIES=0
23 # Prüfung ob Dependencies installiert sind (siehe preflightcheck)
24 NWM=0
25 APA=0
26 JQ=0
27 # Log-File
28 LOG=shz_install.log
29 #Zeitstempel
30 STARTTIME=$(date +%c)
31 TIMESTAMP=$(date +%T)
32 # Funktionen für Skript
33 # Hilfe-Funktion
34 show_help () {
35     echo "  Smart Home Zentrale  "
36     echo "---Installationsskript---"
37     echo
38     echo "Das Installationsskript kann mit diversen Startparametern arbeiten."
39     echo "Die uebliche Syntax lautet:"
40     echo "  sudo ./shz_install [-a|-d|-h|-u|-?]"
41     echo
42     echo "  -a:  Installiert alles (Systemupdates, Dependencies, Docker,
43           ↳ Homeassistant)"
43     echo "        Impliziert -d/-h/-u"
44     echo "  -d:  Installiert Docker."
45     echo "  -h:  Installiert Homeassistant."
```

```

46         echo " -u:    Führt ein Systemupdate aus."
47         echo
48         # Eventuell mit -r alles entfernen?
49         echo " -?:    Ruft diese Hilfeseite auf."
50         echo
51         echo "Der grundlegende Ablauf des Skripts wird in der Datei $LOG geloggt."
52         echo
53         exit 1
54     }
55     # Zeitstempel aktualisieren
56     get_time () {
57         TIMESTAMP=$(date +%T)
58     }
59     # Update-Funktion
60     run_update () {
61         get_time && echo "$TIMESTAMP: Führe Systemupdate durch." >>$LOG
62         sudo apt-get -qq update >> $LOG
63         sudo apt-get -qq upgrade -y >> $LOG
64         sudo apt-get -qq autoremove -y >> $LOG
65         return 0
66     }
67     # Festlegen des Machinentyps
68     get_piver () {
69         if [[ "$PIVER" = 4 ]]; then
70             MACHINE=raspberrypi4
71             get_time
72             echo "$TIMESTAMP: Raspberry Pi Version $PIVER erkannt." >> $LOG
73         elif [[ "$PIVER" = 3 ]]; then
74             MACHINE=raspberrypi3
75             get_time
76             echo "$TIMESTAMP: Raspberry Pi Version $PIVER erkannt." >> $LOG
77         else
78             # Fehlermeldung
79             get_time
80             echo "Skript abgebrochen. Pi-Version nicht erkannt/korrekt" >> $LOG
81             exit 1
82         fi
83     }
84     # Installiert die Dependencies.
85     install_dependencies () {
86         get_time && echo "$TIMESTAMP: Installiere Dependencies." >> $LOG
87         # Prüft, ob Networkmanager installiert werden muss (Siehe preflightcheck)
88         if [[ "$NWM" -eq 1 ]]; then
89             get_time && echo "$TIMESTAMP: Installiere Network Manager." >> $LOG
90             sudo apt-get -qq install network-manager -y >> $LOG
91         fi
92         # Prüft, ob AppArmor installiert werden muss (Siehe preflightcheck)
93         if [[ "$APA" -eq 1 ]]; then

```

```

94         get_time && echo "$TIMESTAMP: Installiere AppArmor." >> $LOG
95         sudo apt-get -qq install apparmor-utils -y >> $LOG
96     fi
97     # Prüft, ob JQ installiert werden muss (Siehe preflightcheck)
98     if [[ "$JQ" -eq 1 ]]; then
99         get_time && echo "$TIMESTAMP: Installiere JQ." >> $LOG
100        sudo apt-get -qq install jq -y >> $LOG
101    fi
102    return 0
103 }
104 # Installiert Docker
105 install_docker () {
106     get_time && echo "$TIMESTAMP: Installiere Docker." >> $LOG
107     # Holt sich das Installationsskript für Docker
108     curl -fsSL https://get.docker.com -o get-docker.sh >> $LOG
109     # Ändert die Zugriffsrechte für das Skript
110     chmod 770 get-docker.sh >> $LOG
111     # Führt das Skript aus
112     sudo sh get-docker.sh
113     # Fügt den Standard-Nutzer der Docker-Gruppe hinzu
114     sudo usermod -aG docker pi >> $LOG
115     # Wartet 20s
116     sleep 20s
117     # Gibt 0 zurück
118     return 0
119 }
120 # Installiert Home Assistant Supervised.
121 # Basiert auf https://raw.githubusercontent.com/home-assistant/supervised |
122 # Version vom 29.12.2020.
123 # Enthält NUR die Ausgaben für den Pi 3 und 4 und keine Konsolenausgaben/Abfragen.
124 # Warnungen wandern ins Log-File!
125 install_homeassistant () {
126     get_time && echo "$TIMESTAMP: Beginne Home Assistant Supervised
127     ↳ Installation." >> $LOG
128     get_time && echo "$TIMESTAMP: Meldungen folgen auf Englisch." >> $LOG
129     ARCH=$(uname -m)
130     IP_ADDRESS=$(hostname -I | awk '{ print $1 }')
131     BINARY_DOCKER=/usr/bin/docker
132     DOCKER_REPO=homeassistant
133     SERVICE_DOCKER="docker.service"
134     SERVICE_NM="NetworkManager.service"
135     FILE_DOCKER_CONF="/etc/docker/daemon.json"
136     FILE_INTERFACE="/etc/network/interfaces"
137     FILE_NM_CONF="/etc/NetworkManager/NetworkManager.conf"
138     FILE_NM_CONNECTION="/etc/NetworkManager/system-connections/default"
139     URL_RAW_BASE="https://raw.githubusercontent.com/home-assistant/supervised |
140     ↳ -installer/master/files"
```

```

139     URL_VERSION="https://version.home-assistant.io/stable.json"
140     URL_BIN_APPARMOR="${URL_RAW_BASE}/hassio-apparmor"
141     URL_BIN_HASSIO="${URL_RAW_BASE}/hassio-supervisor"
142     URL_DOCKER_DAEMON="${URL_RAW_BASE}/docker_daemon.json"
143     URL_HA="${URL_RAW_BASE}/ha"
144     URL_INTERFACES="${URL_RAW_BASE}/interfaces"
145     URL_NM_CONF="${URL_RAW_BASE}/NetworkManager.conf"
146     URL_NM_CONNECTION="${URL_RAW_BASE}/system-connection-default"
147     URL_SERVICE_APPARMOR="${URL_RAW_BASE}/hassio-apparmor.service"
148     URL_SERVICE_HASSIO="${URL_RAW_BASE}/hassio-supervisor.service"
149     URL_APPARMOR_PROFILE="https://version.home-assistant.io/apparmor.txt"
150
# Check env
151     command -v systemctl > /dev/null 2>&1 || MISSING_PACKAGES+=("systemd")
152     command -v nmcli > /dev/null 2>&1 || MISSING_PACKAGES+=("network-manager")
153     command -v apparmor_parser > /dev/null 2>&1 ||
154         ↳ MISSING_PACKAGES+=("apparmor")
155     command -v docker > /dev/null 2>&1 || MISSING_PACKAGES+=("docker")
156     command -v jq > /dev/null 2>&1 || MISSING_PACKAGES+=("jq")
157     command -v curl > /dev/null 2>&1 || MISSING_PACKAGES+=("curl")
158     command -v dbus-daemon > /dev/null 2>&1 || MISSING_PACKAGES+=("dbus")
159     if [ ! -z "${MISSING_PACKAGES}" ]; then
160         get_time && echo "$TIMESTAMP: The following is missing on the host
161             ↳ and needs " >> $LOG
162         get_time && echo "$TIMESTAMP: to be installed and configured before
163             ↳ running this script again" >> $LOG
164         get_time && echo "$TIMESTAMP: missing: " "${MISSING_PACKAGES[@]}" >>
165             ↳ $LOG
166         exit 2
167     fi
# Check if Modem Manager is enabled
168     if systemctl is-enabled ModemManager.service &> /dev/null; then
169         get_time && echo "$TIMESTAMP: ModemManager service is enabled. This
170             ↳ might cause issue when using serial devices." >> $LOG
171     fi
# Detect wrong docker logger config
172     if [ ! -f "$FILE_DOCKER_CONF" ]; then
173         # Write default configuration
174         get_time && echo "$TIMESTAMP: Creating default docker daemon
175             ↳ configuration $FILE_DOCKER_CONF" >> $LOG
176         sudo curl -sL ${URL_DOCKER_DAEMON} > "$FILE_DOCKER_CONF"
# Restart Docker service
177         get_time && echo "$TIMESTAMP: Restarting docker service" >> $LOG
178         sudo systemctl restart "$SERVICE_DOCKER"
179     else
180         STORAGE_DRIVER=$(docker info -f "{{json .}}" | jq -r -e .Driver)
181         LOGGING_DRIVER=$(docker info -f "{{json .}}" | jq -r -e
182             ↳ .LoggingDriver)
183         if [[ "$STORAGE_DRIVER" != "overlay2" ]]; then

```

```

180          get_time && echo "$TIMESTAMP: Docker is using
181              ↳ $STORAGE_DRIVER and not 'overlay2' as the storage
182              ↳ driver, this is not supported." >> $LOG
183      fi
184      if [[ "$LOGGING_DRIVER" != "journald" ]]; then
185          get_time && echo "$TIMESTAMP: Docker is using
186              ↳ $LOGGING_DRIVER and not 'journald' as the logging
187              ↳ driver, this is not supported." >> $LOG
188      fi
189      fi
190      # Check dmesg access
191      if [[ "$(sysctl --values kernel.dmesg_restrict)" != "0" ]]; then
192          get_time && echo "$TIMESTAMP: Fix kernel dmesg restriction" >> $LOG
193          echo 0 > /proc/sys/kernel/dmesg_restrict
194          echo "kernel.dmesg_restrict=0" >> /etc/sysctl.conf
195      fi
196      # Create config for NetworkManager
197      get_time && echo "$TIMESTAMP: Creating NetworkManager configuration" >> $LOG
198      sudo curl -sL "${URL_NM_CONF}" > "${FILE_NM_CONF}"
199      if [ ! -f "${FILE_NM_CONNECTION}" ]; then
200          sudo curl -sL "${URL_NM_CONNECTION}" > "${FILE_NM_CONNECTION}"
201      fi
202      get_time && echo "$TIMESTAMP: Changes are needed to the
203          ↳ /etc/network/interfaces file" >> $LOG
204      get_time && echo "$TIMESTAMP: Replacing /etc/network/interfaces" >> $LOG
205      sudo curl -sL "${URL_INTERFACES}" > "${FILE_INTERFACES}";
206      get_time && echo "$TIMESTAMP: Restarting NetworkManager" >> $LOG
207      sudo systemctl restart "${SERVICE_NM}"
208      PREFIX=${PREFIX:-/usr}
209      SYSCONFDIR=${SYSCONFDIR:-/etc}
210      DATA_SHARE=${DATA_SHARE:-$PREFIX/share/hassio}
211      CONFIG=$SYSCONFDIR/hassio.json
212      # Generate hardware options
213      # Hier nur armv7l, alles andere auf Pi nicht möglich...
214      HASSIO_DOCKER="$DOCKER_REPO/armhf-hassio-supervisor"
215      ### Main
216      # Init folders
217      if [ ! -d "$DATA_SHARE" ]; then
218          mkdir -p "$DATA_SHARE"
219      fi
220      # Read infos from web
221      HASSIO_VERSION=$(curl -s $URL_VERSION | jq -e -r '.supervisor')
222      ##
223      # Write configuration
224      sudo cat > "$CONFIG" <<- EOF
225      {
226          "supervisor": "${HASSIO_DOCKER}",
227          "machine": "${MACHINE}",
228      }

```

```

223         "data": "${DATA_SHARE}"
224     }
225     EOF
226     ##
227     # Pull supervisor image
228     get_time && echo "$TIMESTAMP: Install supervisor Docker container" >> $LOG
229     docker pull "$HASSIO_DOCKER:$HASSIO_VERSION" > /dev/null
230     docker tag "$HASSIO_DOCKER:$HASSIO_VERSION" "$HASSIO_DOCKER:latest" >
231         > /dev/null
232     ##
233     # Install Hass.io Supervisor
234     get_time && echo "$TIMESTAMP: Install supervisor startup scripts" >> $LOG
235     sudo curl -sL ${URL_BIN_HASSIO} > "${PREFIX}/sbin/hassio-supervisor"
236     sudo curl -sL ${URL_SERVICE_HASSIO} >
237         > "${SYSCONFDIR}/systemd/system/hassio-supervisor.service"
238     sed -i "s,%%HASSIO_CONFIG%%,$(CONFIG),g" "${PREFIX}/sbin/hassio-supervisor"
239     sed -i -e "s,%%BINARY_DOCKER%%,$(BINARY_DOCKER),g" \
240         -e "s,%%SERVICE_DOCKER%%,$(SERVICE_DOCKER),g" \
241         -e "s,%%BINARY_HASSIO%%,$(PREFIX)/sbin/hassio-supervisor,g" \
242         "${SYSCONFDIR}/systemd/system/hassio-supervisor.service"
243     chmod a+x "${PREFIX}/sbin/hassio-supervisor"
244     systemctl enable hassio-supervisor.service > /dev/null 2>&1;
245     #
246     # Install Hass.io AppArmor
247     get_time && echo "$TIMESTAMP: Install AppArmor scripts" >> $LOG
248     mkdir -p "${DATA_SHARE}/apparmor"
249     sudo curl -sL ${URL_BIN_APPARMOR} > "${PREFIX}/sbin/hassio-apparmor"
250     sudo curl -sL ${URL_SERVICE_APPARMOR} >
251         > "${SYSCONFDIR}/systemd/system/hassio-apparmor.service"
252     sudo curl -sL ${URL_APPARMOR_PROFILE} >
253         > "${DATA_SHARE}/apparmor/hassio-supervisor"
254     sed -i "s,%%HASSIO_CONFIG%%,$(CONFIG),g" "${PREFIX}/sbin/hassio-apparmor"
255     sed -i -e "s,%%SERVICE_DOCKER%%,$(SERVICE_DOCKER),g" \
256         -e "s,%%HASSIO_APPARMOR_BINARY%%,$(PREFIX)/sbin/hassio-apparmor,g" \
257         "${SYSCONFDIR}/systemd/system/hassio-apparmor.service"
258     chmod a+x "${PREFIX}/sbin/hassio-apparmor"
259     sudo systemctl enable hassio-apparmor.service > /dev/null 2>&1;
260     sudo systemctl start hassio-apparmor.service
261     ##
262     # Init system
263     get_time && echo "$TIMESTAMP: Start Home Assistant Supervised" >> $LOG
264     sudo systemctl start hassio-supervisor.service
265     ##
266     # Setup CLI
267     get_time && echo "$TIMESTAMP: Installing the 'ha' cli" >> $LOG
268     sudo curl -sL ${URL_HA} > "${PREFIX}/bin/ha"
269     sudo chmod a+x "${PREFIX}/bin/ha"
270     # Einfach mal 5 Minuten warten...

```

```

267     sleep 5m
268     get_time && echo "$TIMESTAMP: Home Assistant supervised is now installed" >>
269         $LOG
270     get_time && echo "$TIMESTAMP: First setup will take some time, when it's
271         ready you can reach it here:" >> $LOG
272     get_time && echo "$TIMESTAMP: http://${IP_ADDRESS}:8123" >> $LOG
273     echo "Home Assistant supervised is now installed"
274     echo "First setup will take some time, when it's ready you can reach it
275         here:"
276     echo "http://${IP_ADDRESS}:8123"
277     return 1
278 }
279 # Systemprüfung vor Installationsstart
280 preflightcheck () {
281     get_time && echo "$TIMESTAMP: Prüfe System..." >> $LOG
282     if [[ "$DOCKER" -eq 0 ]] && [[ "$DEPENDENCIES" -eq 0 ]] && [[
283         "$HOMEASSISTANT" -eq 0 ]] && [[ "$UPDATE" -eq 0 ]]; then
284         get_time && echo "$TIMESTAMP: Keine Flags gesetzt. Zeige Hilfe und
285             breche Skript ab." >> $LOG
286         show_help
287         exit 1
288     fi
289     # Prüfe Docker
290     get_time && echo "$TIMESTAMP: Prüfe Docker-Installation..." >> $LOG
291     command -v "docker" >/dev/null 2>&1
292     if [[ $? -eq 0 ]] && [[ "$DOCKER" -eq 1 ]]; then
293         get_time && echo "$TIMESTAMP: Docker existiert bereits. Das Programm
294             wird nicht erneut installiert." >> $LOG
295         DOCKER=0
296     elif [[ $? -eq 0 ]] && [[ "$DOCKER" -eq 0 ]]; then
297         get_time && echo "$TIMESTAMP: Docker existiert bereits." >> $LOG
298     else
299         get_time && echo "$TIMESTAMP: Docker existiert nicht. Das Programm
300             wird unabhängig der Nutzerwahl installiert." >> $LOG
301         DOCKER=1
302     fi
303     # Prüfe Dependencies
304     get_time && echo "$TIMESTAMP: Prüfe Dependencies" >> $LOG
305     # Prüfe Network Manager
306     command -v "nmcli" >/dev/null 2 >&1
307     if [[ $? -eq 0 ]]; then
308         get_time && echo "$TIMESTAMP: Networkmanager existiert bereits. Das
309             Programm wird nicht erneut installiert." >> $LOG
310         NWM=0
311     else
312         get_time && echo "$TIMESTAMP: Networkmanager existiert nicht. Wird
313             mitinstalliert." >> $LOG
314         NWM=1

```

```

306         fi
307     # Prüfe AppArmor
308     command -v "aa-status" >/dev/null 2 >&1
309     if [[ $? -eq 0 ]]; then
310         get_time && echo "$TIMESTAMP: AppArmor existiert bereits. Das
311             ↳ Programm wird nicht erneut installiert." >> $LOG
312         APA=0
313     else
314         get_time && echo "$TIMESTAMP: AppArmor existiert nicht. Wird
315             ↳ mitinstalliert." >> $LOG
316         APA=1
317     fi
318     # Prüfe JQ
319     command -v "jq" >/dev/null 2 >&1
320     if [[ $? -eq 0 ]]; then
321         get_time && echo "$TIMESTAMP: JQ existiert bereits. Das Programm
322             ↳ wird nicht erneut installiert." >> $LOG
323         JQ=0
324     else
325         get_time && echo "$TIMESTAMP: JQ existiert nicht. Wird
326             ↳ mitinstalliert." >> $LOG
327         JQ=1
328     fi
329     # Dependencies Sanity Check...
330     if ( [[ "$NWM" -eq 1 ]] || [[ "$APA" -eq 1 ]] || [[ "$JQ" -eq 1 ]] ) && [[
331         ↳ "$DEPENDENCIES" -eq 0 ]]; then
332         get_time && echo "$TIMESTAMP: SANITY-CHECK ERROR! Dependencies
333             ↳ fehlen, werden aber nicht angefordert." >> $LOG
334         get_time && echo "$TIMESTAMP: Dependencies werden entsprechend
335             ↳ installiert." >> $LOG
336         DEPENDENCIES=1
337     else
338         get_time && echo "$TIMESTAMP: SANITY-CHECK erfolgreich." >> $LOG
339     fi
340     # Prüfe Home Assistant
341     get_time && echo "$TIMESTAMP: Prüfe Home Assistant" >> $LOG
342     command -v "hassio_cli" >/dev/null 2 >&1
343     if [[ $? -eq 0 ]]; then
344         get_time && echo "$TIMESTAMP: Home-Assistant existiert bereits.
345             ↳ Programm wird nicht neu installiert." >> $LOG
346         HOMEASSISTANT=0
347     else
348         get_time && echo "$TIMESTAMP: Homeassistant existiert nicht. Wird
349             ↳ mitinstalliert." >> $LOG
350     fi
351     return 0
352 }
353 # Funktionen für die Switch-Case der Übergabeflags.

```

```
345 set_all () {
346     DOCKER=1
347     HOMEASSISTANT=1
348     UPDATE=1
349     DEPENDENCIES=1
350     get_time && echo "$TIMESTAMP: Komplettinstallation angefordert." >>$LOG
351 }
352 set_docker () {
353     DOCKER=1
354     get_time && echo "$TIMESTAMP: Docker Installation angefordert." >> $LOG
355 }
356 set_homeassistant () {
357     HOMEASSISTANT=1
358     get_time && echo "$TIMESTAMP: Home Assistant Installation angefordert." >>
359         $LOG
360 }
361 set_update () {
362     UPDATE=1
363     get_time && echo "$TIMESTAMP: Systemupdate angefordert." >>
364         $LOG
365 }
366 # Haupt-Routine
367 echo "  Smart Home Zentrale  " >> $LOG
368 echo "---Installationsskript---" >> $LOG
369 echo "$STARTTIME" >> $LOG
370 get_time && echo "$TIMESTAMP: Beginne Installation" >> $LOG
371 get_piver
372 get_time && echo "$TIMESTAMP: Lese Eingabeparameter" >> $LOG
373 while getopts "adhu" option; do
374     case ${option} in
375         a ) set_all;;
376         d ) set_docker;;
377         h ) set_homeassistant;;
378         u ) set_update;;
379         \? ) show_help;;
380     esac
381 done
382 preflightcheck
383 if [[ "$UPDATE" -eq 1 ]]; then
384     echo "Starte Update"
385     run_update
386 fi
387 if [[ "$DEPENDENCIES" -eq 1 ]]; then
388     echo "Installiere Dependencies"
389     install_dependencies
390 fi
391 if [[ "$DOCKER" -eq 1 ]]; then
392     echo "Installiere Docker"
```

---

```
391     install_docker
392 fi
393 if [[ "$HOMEASSISTANT" -eq 1 ]]; then
394     echo "Installiere Home Assistant"
395     install_homeassistant
396 fi
```

---

## 10.3 YAML-Dateien

### 10.3.1 Home Assistant Konfiguration

```
1 # Configure a default setup of Home Assistant (frontend, api, etc)
2 default_config:
3
4 # Text to speech
5 tts:
6   - platform: google_translate
7
8 group: !include groups.yaml
9 automation: !include automations.yaml
10 script: !include scripts.yaml
11 scene: !include scenes.yaml
```

### 10.3.2 Zigbee2MQTT Konfiguration

```
1 {
2   "external_converters": [],
3   "devices": [
4     "devices.yaml"
5   ],
6   "groups": [
7     "groups.yaml"
8   ],
9   "homeassistant": true,
10  "permit_join": false,
11  "mqtt": {
12    "base_topic": "zigbee2mqtt",
13    "user": "addons",
14    "password": "eim7ro3Ix08Paech6ireex3ahwisahSh2YooYiec7Quahghaighoafah5AiY8eiw",
15    "server": "mqtt://core-mosquitto:1883"
16  },
17  "serial": {
18    "port": "/dev/ttyACM0"
19  },
20  "advanced": {
21    "log_level": "warn",
22    "pan_id": 6754,
23    "channel": 11,
24    "network_key": [
25      1,
26      3,
27      5,
28      7,
29      9,
```

```
30      11,
31      13,
32      15,
33      0,
34      2,
35      4,
36      6,
37      8,
38      10,
39      12,
40      13
41 ],
42 "availability_blocklist": [],
43 "availability_passlist": []
44 },
45 "device_options": {},
46 "blocklist": [],
47 "passlist": [],
48 "queue": {},
49 "frontend": {
50     "port": 8099
51 },
52 "experimental": {}
53 }
```

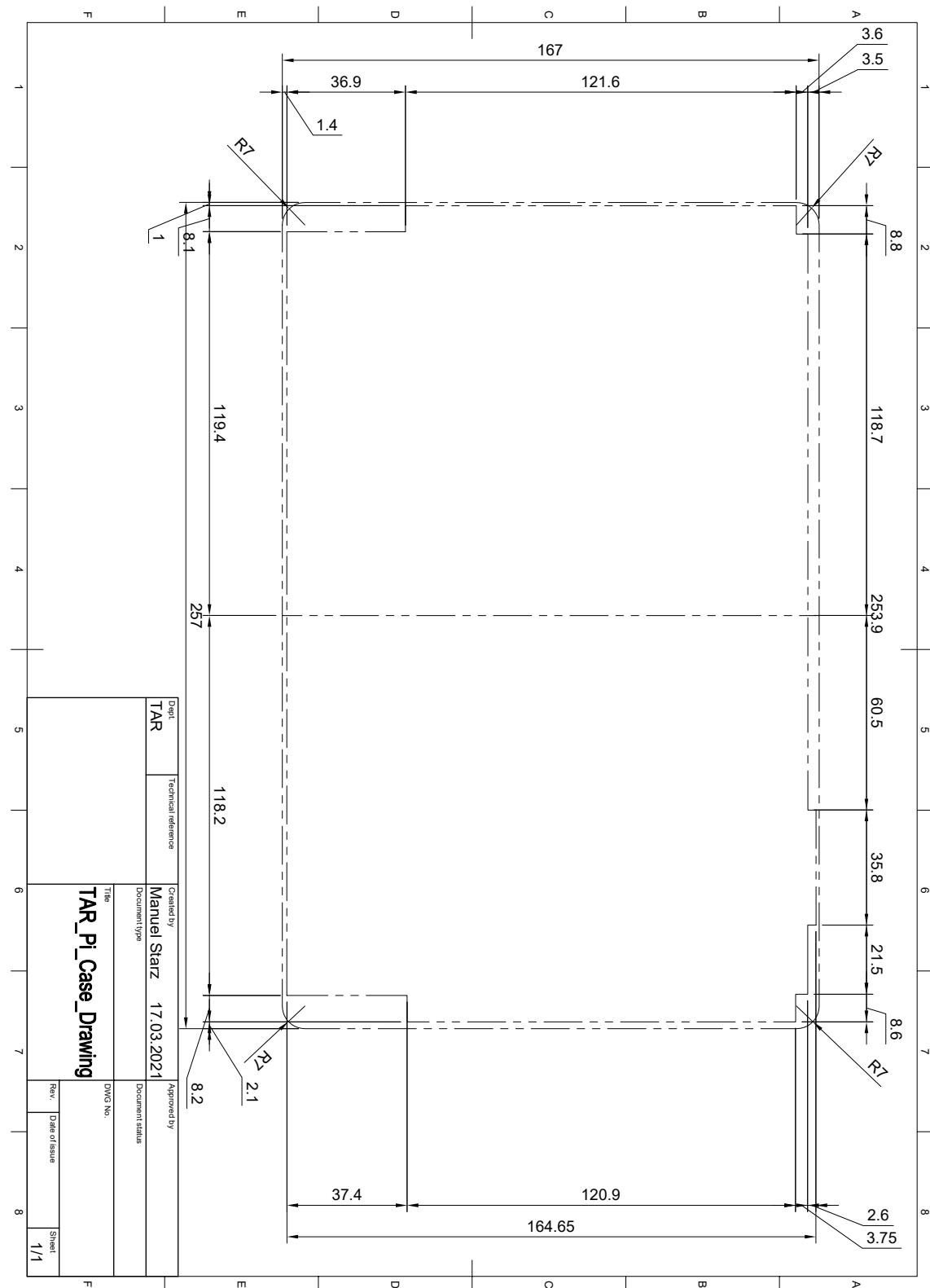
---

### 10.3.3 Zigbee Geräteauflistung

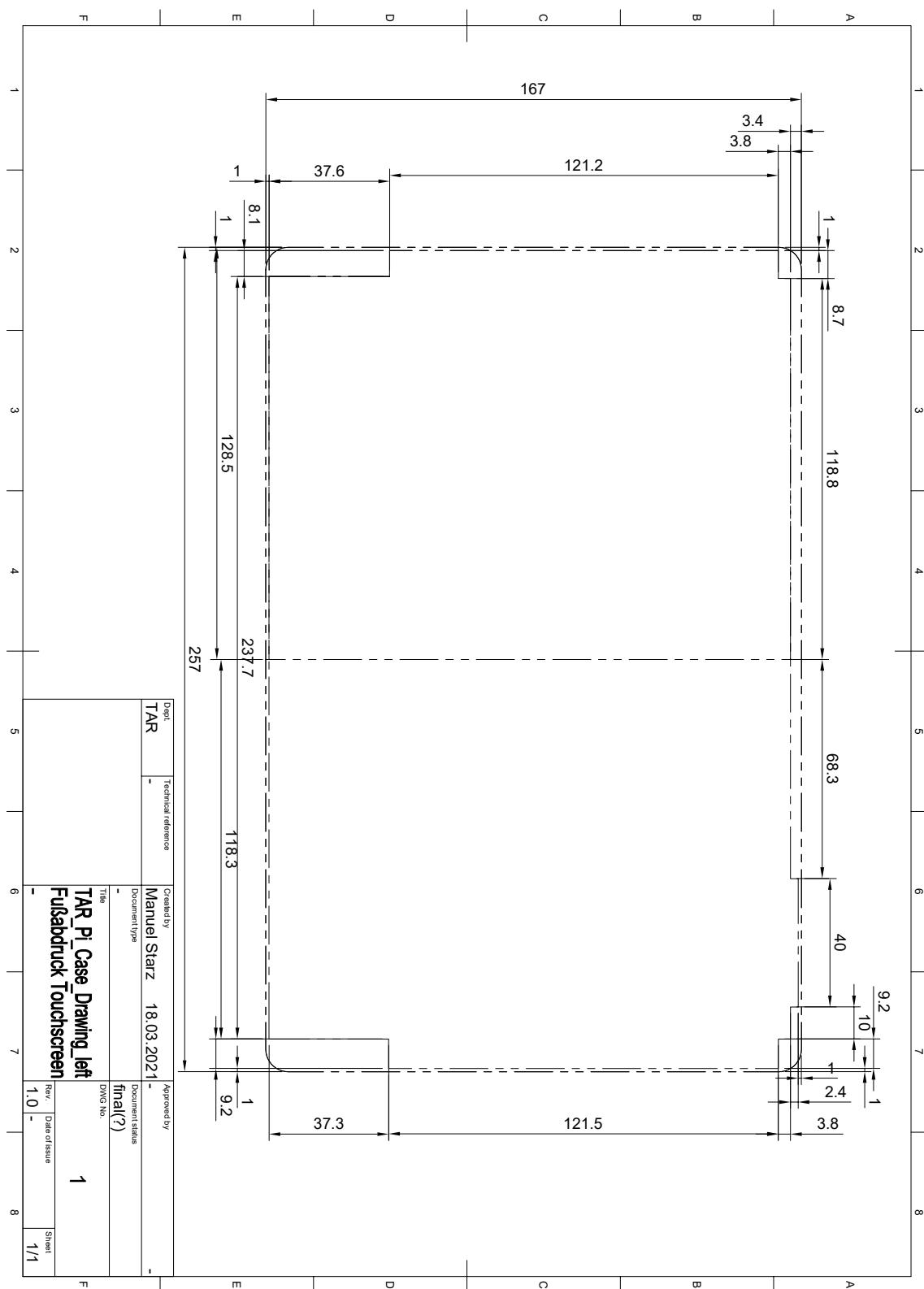
```
1 '0x680ae2ffffe4d979e':
2   friendly_name: IKEA Gu10
3 '0x0017880108e63c66':
4   friendly_name: 'Hue Steckdose '
```

---

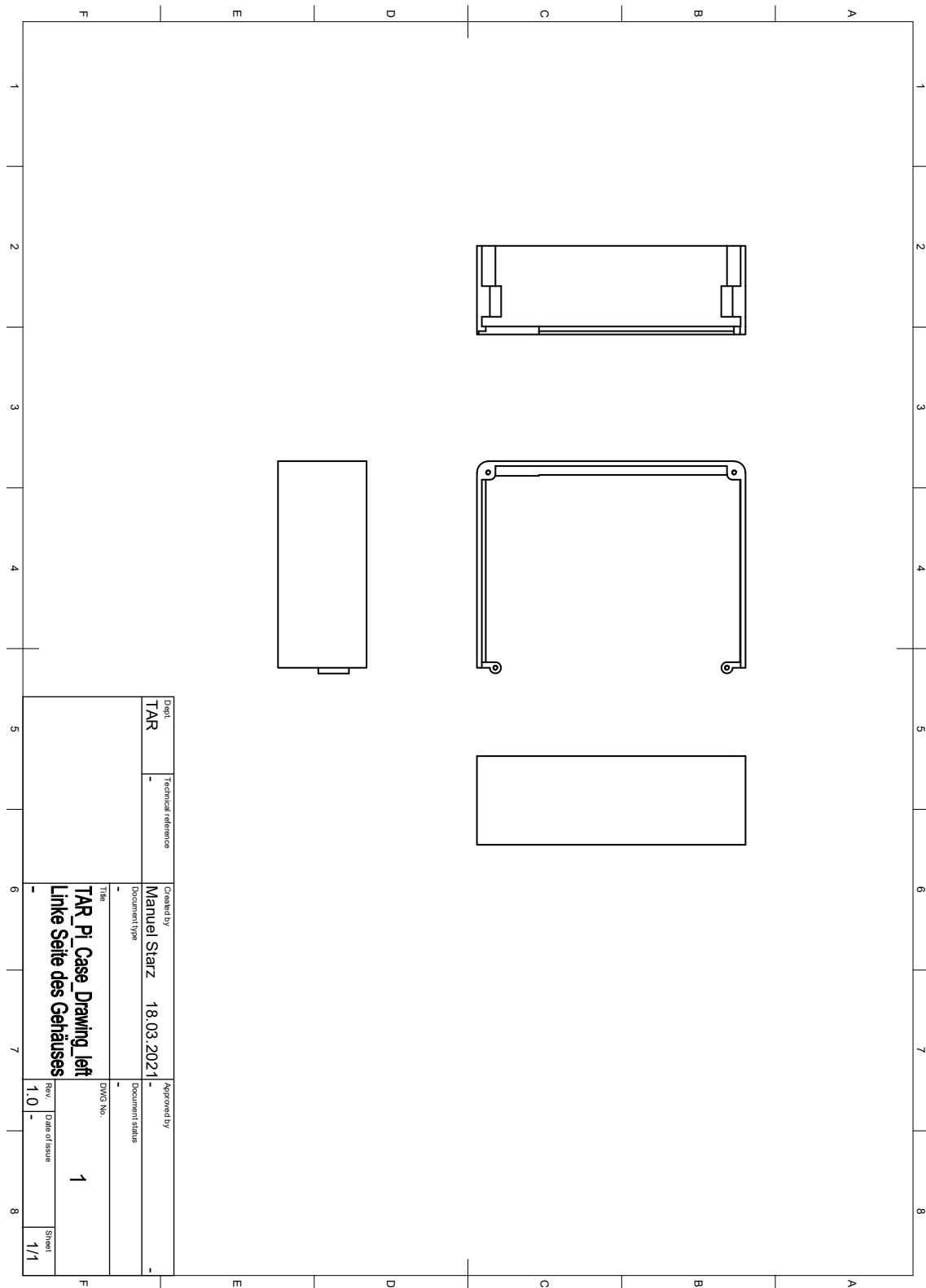
## 10.4 Gehäuse-Zeichnungen



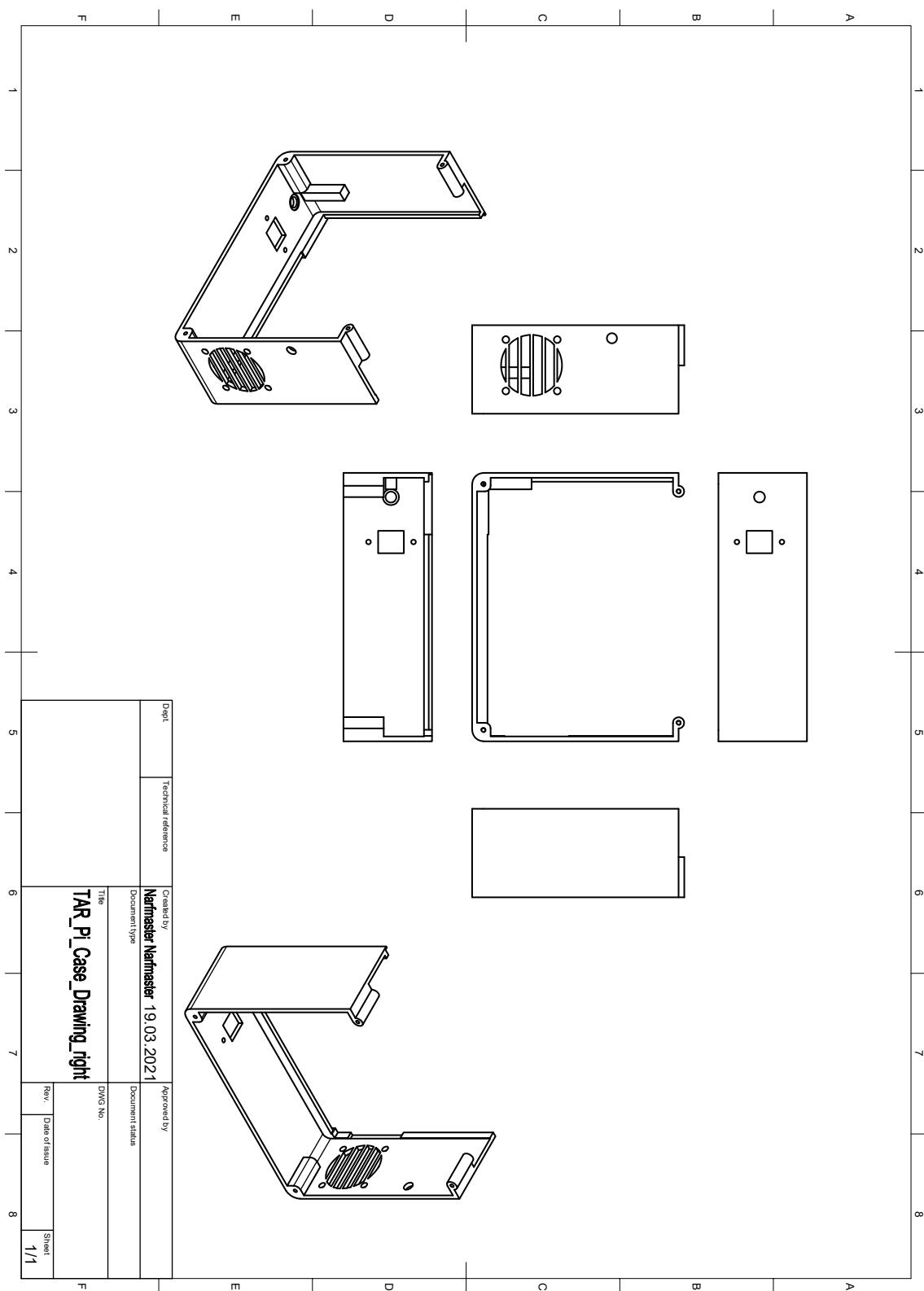
Erster Versuch der Zeichnung des Fußabdruckes des Gehäuses



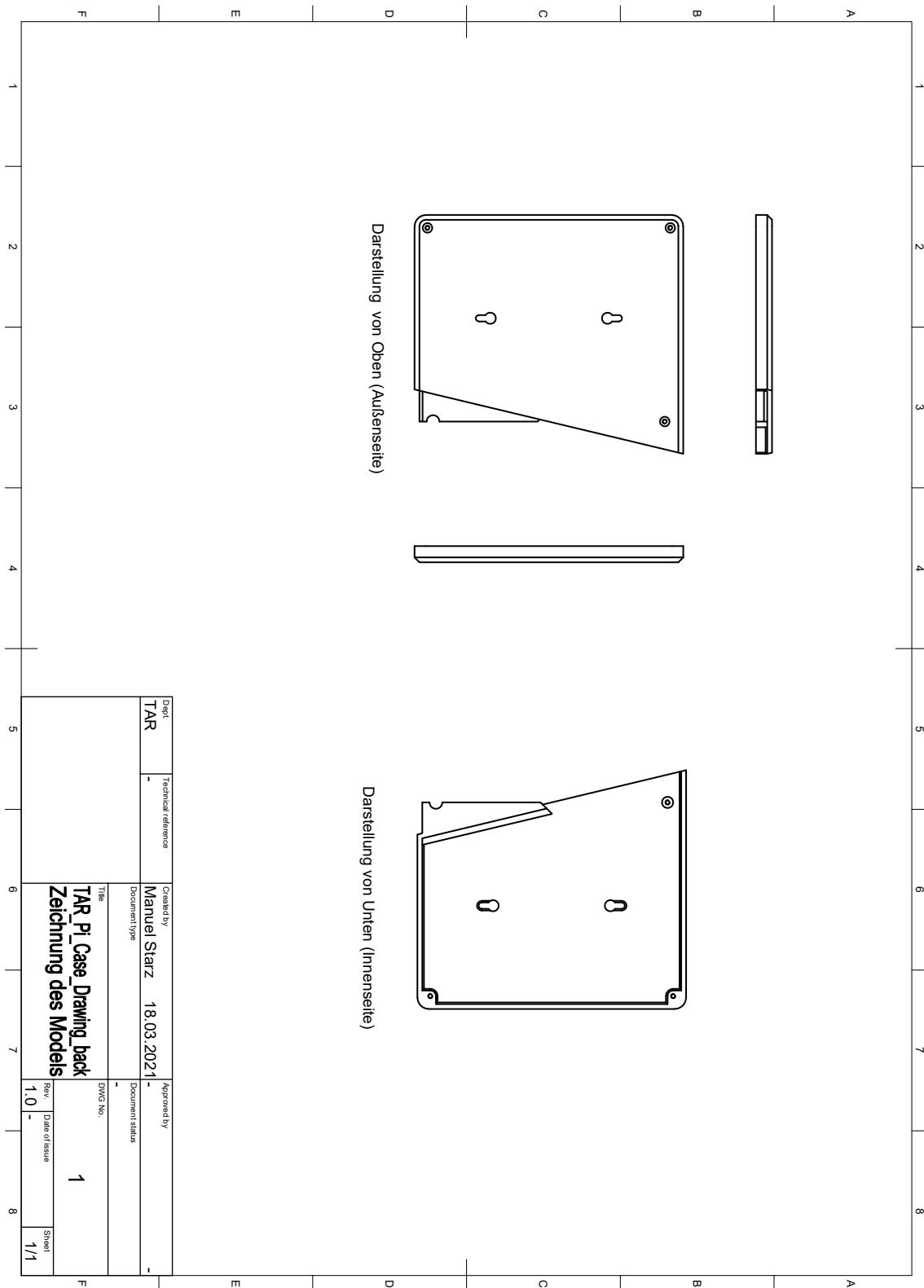
Finale Version der Zeichnung des Fußabdruckes des Gehäuses



Linker Teil des Gehäuses

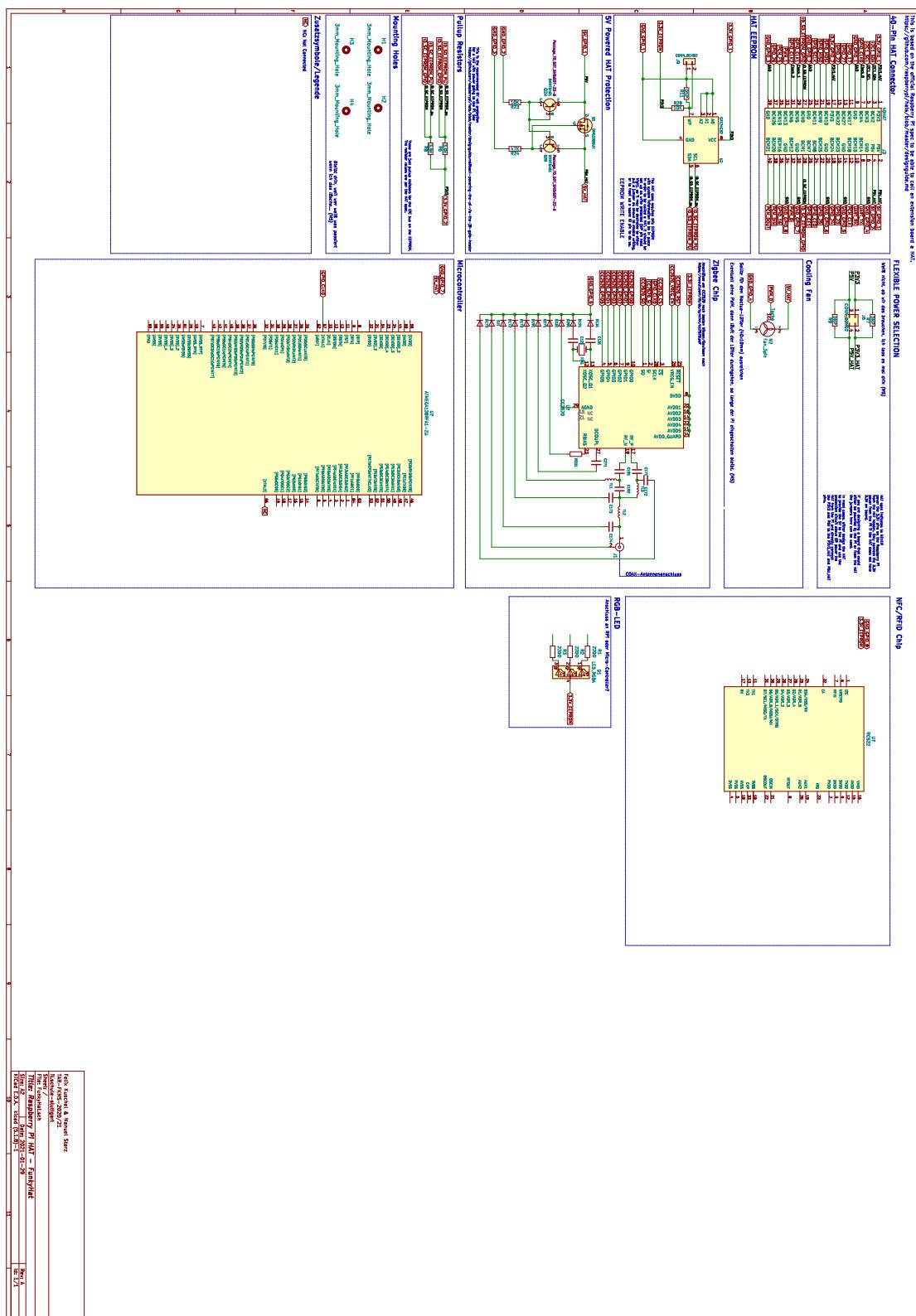


Rechter Teil des Gehäuses



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. zu 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

1	Beschaffungskosten . . . . .	15
2	Erfassung der Arbeitszeiten . . . . .	16
3	Kosten Präsentationsaufbau . . . . .	17
4	Gesamtkostenberechnung . . . . .	17
5	Skript-Funktionen . . . . .	38
6	Überblick über die Images für die Raspberry Pis . . . . .	43
7	Verwendete Software . . . . .	48
8	Verwendete Hardware . . . . .	48

### Abbildungsverzeichnis

1	Beispiele für Smart Home Zentralen . . . . .	5
2	Beispiele für Smart Home Software . . . . .	6
3	Gantt-Diagramm des Projektablaufs . . . . .	12
4	KANBAN-Board in Microsoft Teams . . . . .	12
5	Smart Home Zentrale . . . . .	13
6	Gehäuse mit Füßen (Render in Meshmixer) . . . . .	14
7	Prototyp der Smart Home Zentrale . . . . .	19
8	Abmessungen Bildschirm Rückseite . . . . .	20
9	Platzierung der beiden Gehäusewände in CURA . . . . .	21
10	Verschiebung der Modelle entlang der Z-Achse um 45mm . . . . .	21
11	Slicen der Modelle . . . . .	22
12	Anlegen des Testdrucks . . . . .	22
13	Grundzeichnung als Basis des Modells . . . . .	23
14	Entwurf des linken Wandteils . . . . .	25
15	Entwurf des rechten Wandteils . . . . .	27
16	Entwurf der Gehäuserückwand . . . . .	29
17	3D-Drucker Ender 3 Pro (mod) . . . . .	30
18	OctoPrint-Weboberfläche . . . . .	30
19	Das Gehäuse entsteht . . . . .	31
20	Ausgedruckte Teile des Gehäuses . . . . .	31
21	Fertigung des Gehäuses . . . . .	32
22	Raspberry Pi 4 GPIO-Pins . . . . .	33
23	Aktueller Stand HAT-Design . . . . .	34
24	Präsentationsaufbau . . . . .	35
28	Erstellen der Images . . . . .	44