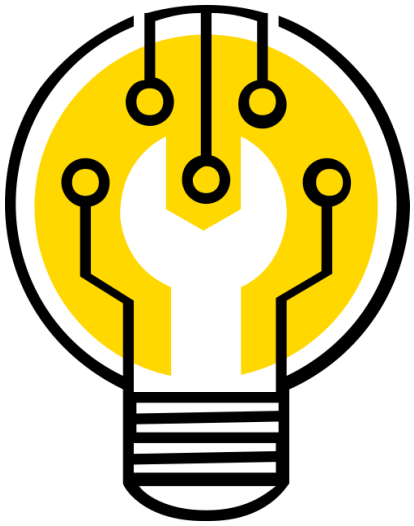
Diplomarbeit

Sensorphalanx für Heimautomation

Projektbezeichnung Thema	Konzipierung und Entwicklung einer Sensorphalanx für Smart Home
Lehrgang, Jahrgang, Klasse	6. Automation 18H-21H
Diplomand	Beckmann, Raphael Gehrenstrasse 21 9230 Flawil Telefon: 079 277 96 13 E-Mail: Raphael.Beckmann1983@gmail.com
Betreuer	
Experte, / Expertin	
Auftraggeber	Der Diplomand
Grafik und Projektmotto	 **Let's build things!**

Danksagungen

Hiermit möchte ich mich bei Alexander Palmer, bedanken, der mir die Möglichkeit eröffnete, ein spannendes und für mich persönlich sehr interessantes Thema im Rahmen meiner Diplomarbeit zu bearbeiten. Insbesondere bedanke ich mich dafür, dass er mich stets mit Ratschlägen und Hilfestellungen betreut hat.

Meinen Freunden möchte ich für die Unterstützung und vor allem meinen Freund Martin für die stetige Motivation danken.

Ein weiterer Dank gilt der Familie Beckmann für die moralische Unterstützung und meine Frau, die auf vieles verzichtet hat, um mir die Ausbildung zu ermöglichen.

Nicht zuletzt möchte ich mich bei meiner Tochter bedanken, der ich diese Arbeit widme. Sie hat mich auf die Wichtigkeit von Gendergerechter Sprache aufmerksam gemacht und in den letzten vier Jahren und besonders in den letzten drei Monaten an vielen Stellen auf ihren «Papi» verzichten müssen.

Management Summary

Anhand der vorliegenden Diplomarbeit soll aufgezeigt werden, wie die Konzeptionierung und Entwicklung eines Prototyps zum Sammeln von Raumdaten für Home-Automations-Systeme realisiert werden kann.

Hierzu wurden zunächst die Anforderungen definiert, unterschiedliche Hard- uns Softwarekomponenten gegeneinander abgewogen und abschliessend bewertet. Die in der Konzeptphase definierten Komponenten wurden zu einem ersten Lochrasterprototypen zusammengeführt. Im Anschluss wurde eine Software für den Prototypen geschrieben und getestet. Um Einsicht in die Sensordaten nehmen zu können, wurde dargelegt, wie das Home-Automations-System «Home Assistant» installiert und für die nötige Darstellung parametriert wird.

Aus den Erkanntnissen des Lochrasterprototypen wurde ein weiterer PCB-Prototyp entwickelt und in Produktion gegeben.

Der realisierte Prototyp bietet eine gute Basis für eine Weiterentwicklung zu einem kommerziell erfolgreichen Bausatz für interessierte Anwender*innen und die gesammelten Erfahrungen ermöglichen bei Veröffentlichung eine Weiterentwicklung des Projekts sowie anderer ähnlicher Projekte.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

A/D	Analog-/ Digital
ARM	Advanced Risc Machines
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
eCO2	Equivalent CO2
EMV	Elektromagnetische Verträglichkeit
ESP	Espressif
GPIO	General Purpose Input/Output
http	Hypertext Transfer Protocol
I²C	I-Quadrat-C
IAQ	Indoor Air Quality
IC	Integrated Circuit
IT	Information Technology
LDR	Light Dependent Resistor
LED	Light Emitting Diode
Li-Ion	Lithium-ion
MQTT	Message Queuing Telemetry Transport
NTC	Positiv Temperature Coefficient Thermistor
PAP	Programmablaufplan
PCB	Printed Circuit Board
PTC	Negative Temperature Coefficient Thermistor
RGB	Red, Green, and Blue
RTC	Real-Time Clock
SCL	Serial Clock
SDA	Serial Data
SMD	Surface-Mount Device
SSID	Service Set IDentifier
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
WLAN	Wireless Local Area Network

XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

Glossar

Abwärtskompatibilität	Erfüllt eine neuere Version einer Software die Anforderungen seines Vorgängers und kann diese ohne Anpassungen ersetzten, spricht man von Abwärtskompatibilität.
Access Point	Gerät, das eine Schnittstelle zwischen kabelloser und kabelgebundener Kommunikation ermöglicht.
Addon	Addons oder Plug-ins sind Softwareergänzungen, die den Funktionsumfang der Hauptanwendung erweitern.
Akkumulator	Wiederaufladbarer Energiespeicher
Aktor	Aktor (oder auch Aktuator) sind Bauelemente, die aktiv in einen Prozess eingreifen können, um z.B. Bewegungen oder Temperaturänderungen herbeizuführen.
Arduino IDE	Die Arduino DIE ist eine Software um Arduino Hardware zu Programmieren und zu bespielen.
Battery Management IC	Spezielle ICs, die in der Lage sind Batterien zu überwachen und vor defekten zu schützen.
Blackbox	Eine Blackbox beschreibt sowohl Soft- als auch Hardware, deren Funktion man nicht einsehen kann und lediglich die Reaktion auf Eingaben beobachtet werden können.
Breakout Board	Breakout Boards ermöglichen es, ICs mittels Pins auf Steckbretter oder Sockel zu setzen, um so die Funktionalität flexibel nutzbar zu machen.
Deep-Sleep	Englisch für Tiefschlaf, diese Funktion ermöglicht den Standby eines ICs bei minimalem Energieverbrauch.
Fusion 360	3D CAD Programm zur Erstellung von Konstruktionen.
Gerberdateien	Gerberdateien sind im ASCII-Format geschriebene Dateien, die Informationen zu jeder Leiterplattenschicht enthalten.
GitHub	GitHub ist eine Plattform zur Versionsverwaltung und Veröffentlichung von Softwareprojekten.
Image	Ist ein Speicherabbild eines Datenträgers.
Initialisiert	Ist die Zuweisung von Anfangswerten zu einem Objekt oder Variablen.
ISO/OSI-Modells	Ist ein in Schichten aufgebautes Referenzmodel für Netzwerkprotokolle.
Jumper	Jumper sind kleine, steckbare Verbindungsbrücken.
konventionell	Beschreibt, unter anderem, Sachverhalte als in gewohnter Weise entsprechend
Laderegler	Kontrolliert den Ladestrom und den Ladezustand eines Akkumulators
Modularität	Aufteilen in verschiedene einzelne Komponenten
NexDock	Ist eine Dockingstation für Smartphons und Minicomputer und erweitert diese um eine Notebook Funktionalität.
Payload	Bezeichnet die Nutzdaten die innerhalb eines Datenpackets übermittelt werden.
Phalanx	Ist eine dichtstehende, militärische Formation
Real Time Clock	Englisch für Echtzeituhr. Eine IC, der, solang er über Energie verfügt, einen exakten Zeitverlauf gewährleistet.
Routing	Beschreibt das Verlegen der Leiterbahnen auf einem PCB.
Sensorphalanx	In Anlehnung des begriffs Phalanx beschreibt der Begriff Sensorphalanx eine Einheit mit kompakt angeordneten Sensoren.
Smart Home	Beschreibt das «intelligente» Heim, das in der Lage ist vorher definierte Routinen abzuarbeiten sowie auf interne oder externe Änderungen zu reagieren.

Watchdog	Beschreibt in der Softwareentwicklung ein Programm, das kontrolliert ob die gewünschte Funktion noch ordnungsgemäss abläuft.

1 Ausgangslage

“Smart Home” und “Home Automation” erfreuen sich immer grösserer Beliebtheit. Fast jeder Neubau wird mit einer zentralen Steuerungseinheit gebaut, die mindestens das Raumklima steuert. Darüber hinaus gibt es viele Varianten und Zusatzfunktionen: Von einfachen Steckdosen- und Einbaugeräteschaltungen über Energiemanagement bis hin zur vollautomatischen Zugangs- und Zutrittsregelungen.

Allerdings gibt es bei Bestandsgebäuden einige Hindernisse, die mit konventionellen Produkten nicht oder nur unzureichend abgedeckt werden können. Beispielsweise dürfen Mieter*innen in Mietwohnungen nur sehr begrenzt Eingriffe in die Elektroinstallation vornehmen. Da es viele Maker*innen gibt, die ebenfalls eine Möglichkeit haben möchten, ihre gemieteten Wohnungen auch im Automationsbereich zu gestalten, besteht hier aus Sicht des Autors eine Marktchance. Diese Marktchance soll mit der Entwicklung eines entsprechenden Bausatzes ergriffen werden, indem der Bausatz den Kunden*innen die Möglichkeit einräumt, ihre eigenen Ideen umzusetzen. Zugleich soll er auch ermöglichen, die Sensordaten dort zu sammeln, wo sie benötigt werden und nicht, wie bei vielen konventionellen Produkten, am Aktor selber.

Ebenso ist das Thema IT-Sicherheit ein wichtiger Punkt. Immer wieder werden gravierende Sicherheitslücken in Smart Home Produkten öffentlich diskutiert. Ein Beispiel hierfür sind WLAN-kompatible Leuchtmittel, die unter Umständen den unverschlüsselten Standort und die WLAN-Daten speichern oder ohne direkte Einwilligung des Nutzers an den Herstellern übermitteln (1). Ein weiterer Faktor ist auch die gewünschte Unabhängigkeit von Herstellerressourcen wie Cloud-Service und App-Entwicklung, da diese ohne konkreten Einfluss des Nutzers offline genommen oder nicht mehr weiterentwickelt werden können.

1.1 Problemstellung

Die Herstellung eines Bausatzes einer Sensorphalanx, die es Maker*innen ermöglicht, Raumdaten zu sammeln und für ihr Smart Home zu nutzen.

1.2 Zielsetzung

Teil dieses Projektes wird es sein, einen lokalen Server für Home Automation aufzusetzen. Vorzugsweise soll dies auf eine günstige und gut verfügbare Hard- und Softwarebasis realisiert werden.

Der Fokus liegt auf der Entwicklung eines Prototyps einer Hardware Sensorphalanx. Diese soll modular aufgebaut werden, sodass es dem User möglich ist, anwendungsspezifisch einzelne Sensormodule auszuwählen und zu verbauen. Zur Kommunikation soll

WLAN verwendet werden. Die Sensorphalanx wird keinen direkten Kontakt mit dem Internet haben, sondern nur mit dem Home Automation-Server kommunizieren.

Die Sensoren Phalanx soll keine CE-Zertifizierung erhalten müssen, da sie nur als rudimentärer Bausatz und in Form von Plänen und Programmcode für erfahrene User zugänglich gemacht werden soll.

Weiter soll die Phalanx über keine Aktoren, die über ein Feedback für den User hinausgehen, verfügen. Da die Phalanx an zentralen Punkten in einer Mietwohnung montiert werden soll, liegt ein Schwerpunkt auch auf der Stromversorgung. Diese soll mittels eines Elektrospeichers erfolgen, der auf einfachem Wege wieder aufladbar sein muss.

1.2.1 Wunschziel

Um anderen Maker*innen zu ermöglichen, die Sensorphalanx selber bauen zu können oder auf der Grundlage des Ergebnisses weiterzuentwickeln, sollen zum Abschluss der Diplomarbeit alle erarbeiteten Inhalte online verfügbar gemacht werden.

1.3 Persönliche Motivation

Der Autor möchte schon länger die Möglichkeit haben, in seiner Mietwohnung ein zusammenhängendes Smart Home System einzusetzen. Um dieses sinnvoll und auf die persönlichen Bedürfnisse ausrichten zu können, hat er sich entschlossen, dieses Smart Home System selbst zu realisieren.

1.4 Rahmenbedingungen und Abgrenzungen

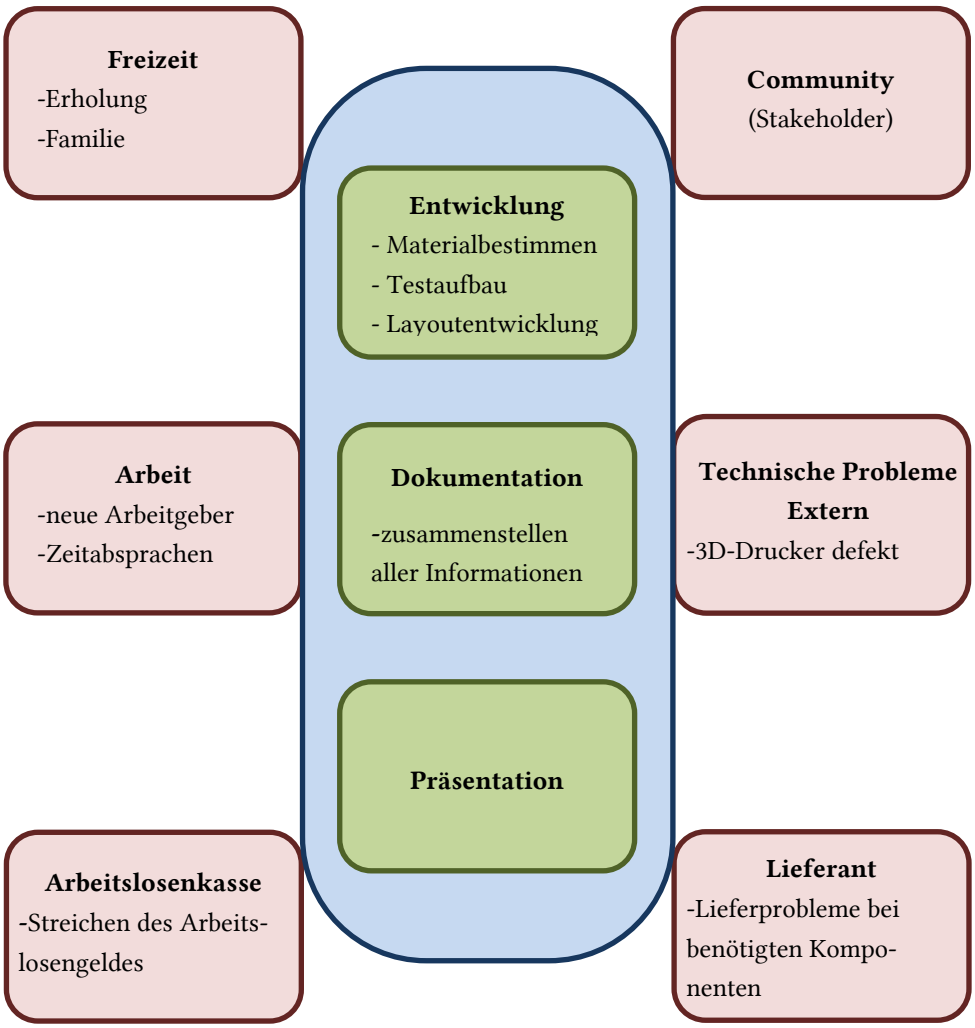


Abbildung 1: Rahmenbedingungen und Abgrenzungen

Das Ergebnis der Diplomarbeit soll ein funktionsfähiger Prototyp einer Sensoreinheit für die Home Automation sein. Diese Einheit soll so dokumentiert werden, dass ein Nachbau für fachkundige Maker*innen gut möglich ist.

Die Schnittstellen nach aussen sind in diesem Projekt nicht zahlreich, da der Diplomand ebenfalls Auftraggeber wie auch Endnutzer ist. Da das Ergebnis aber im Anschluss der interessierten Öffentlichkeit als Bausatz angeboten werden soll, ist vorgesehen, während der Diplomarbeit den Fortschritt auf einen Blog festzuhalten und wenn möglich mit Videos zu begleiten. Dieses ist nicht Teil der Diplomarbeit und erfolgt parallel, sofern dem Diplomanden genug Ressourcen zur Verfügung stehen.

1.5 Termine und Lieferobjekte

Im folgenden Kapitel wird die Terminplanung behandelt, dazu wird zunächst definiert, wann die Meilensteine erledigt sein sollen. Dazu wird ein Terminplan erstellt, um eine Orientierung für den Ablauf des Projekts zu bekommen. Im Anschluss werden noch die Lieferobjekte definiert.

1.5.1 Meilensteinplan

Tabelle 1: Meilensteinplan

Nr.	Meilenstein	Plan Datum	Ist Datum
1	Grobkonzept	30.08.2021	30.08.2021
2	Feinkonzept	28.10.2021	17.11.2021
3	Hausleitsystem betriebsbereit	04.11.2021	26.11.2021
4	Lochrasterprototyp	09.11.2021	19.11.2021
5	Sensorsoftware betriebsbereit	23.11.2021	08.12.2021
6	PCB / Gehäuse Prototypen	06.12.2021	PCB am 16.12.2021 bestellt Gehäusekonstruktion noch offen
7	Diplomarbeit Abschluss	05.01.2022	02.01.2022

1.5.2 Terminplan

Tabelle 2: Terminplan Teil 1

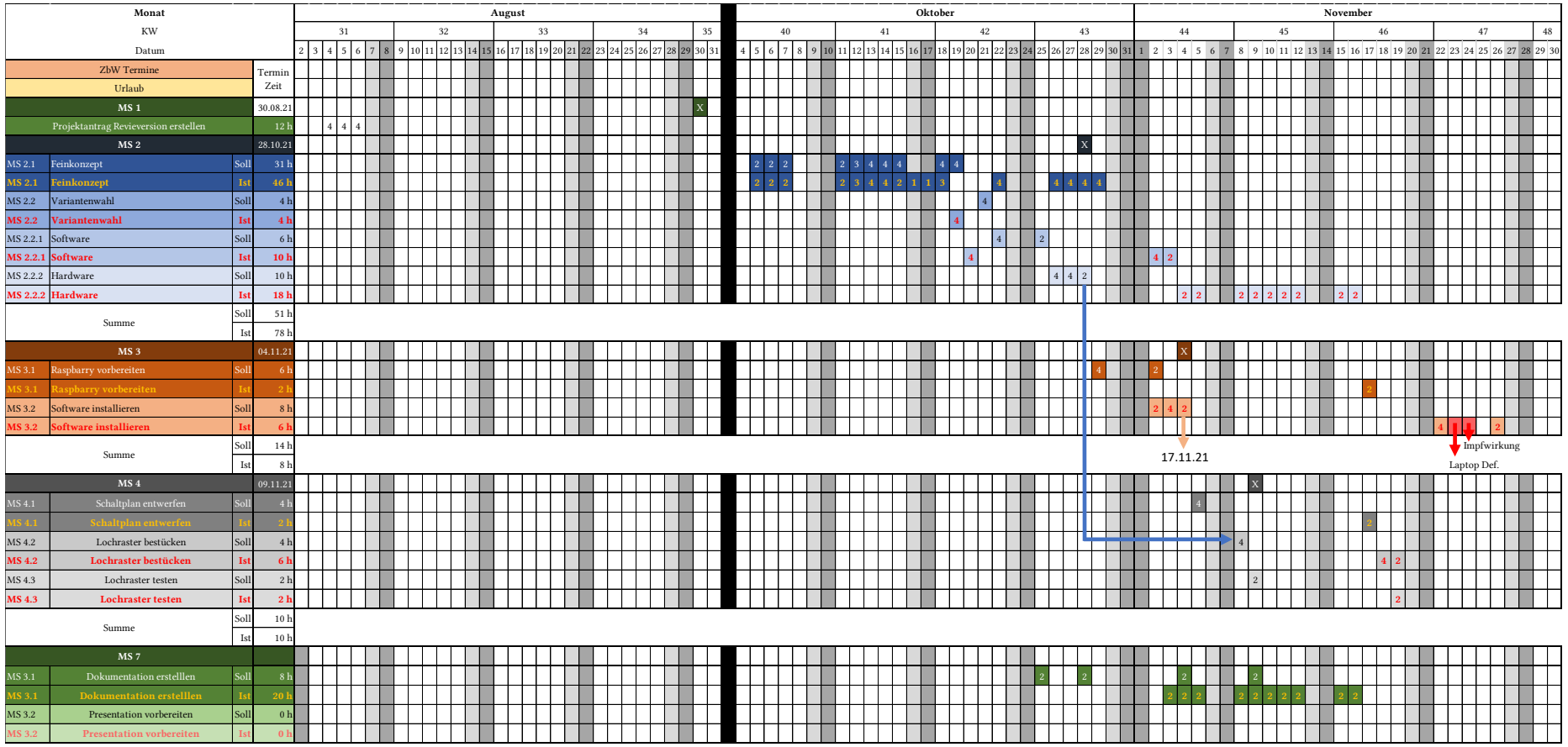
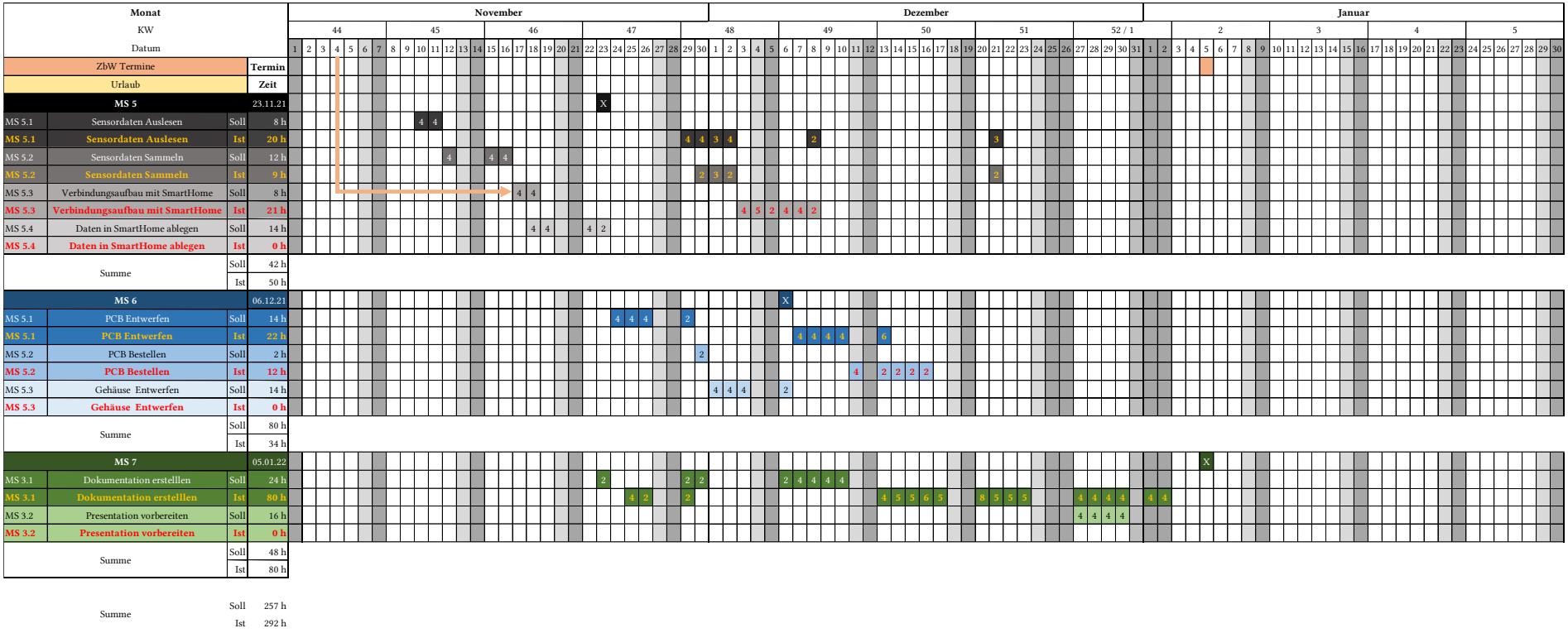


Tabelle 3: Terminplan Teil 2



Für eine bessere Lesbarkeit befindet sich das Excel-File in der Datenablage des Anhangs (Anhang 1).

1.5.3 Lieferobjekte

Tabelle 4: Lieferobjekte

Nr.	Ziel	Muss	Kann
1.	Elektrolayout	X	
2.	PCB-Layout		X
3.	Sensordefinition	X	
4.	OpenHAB auf Raspberry Pi	X	
5.	Aufzeichnung von Messdaten	X	
6.	Veröffentlichung des Fortschrittes		X
7.	Veröffentlichung des Source Code und der Schemata		X
8.	PCB mit SMD Bauteilen bestellt		X
9.	Serienfähiger Prototyp		X
10.	3D gedrucktes Gehäuse		X
11.	Lochraster Prototyp	X	

1.6 Präferenzmatrix für Projektkriterien

Um für die im weiteren Verlauf nötigen Entscheidungen eine Grundlage zu haben, wurde die Präferenzmatrix gewählt. In dieser wurden alle oben genannten Attribute eingetragen und gegeneinander gewichtet. Als Ergebnis gibt es eine Priorisierung der einzelnen Kriterien. Diese werden nach Rang sortiert aufgelistet, wobei bei Gleichstand im Rang das direkte Vergleichsergebnis aus der Matrix herangezogen wird.

Tabelle 5: Präferenzmatrix Projektkriterien

Präferenzmatrix

für das Projekt Sensorphalanx

Gewicht	Rang-folge	Anzahl Nen-nungen		Kriterien
8.791	5	8	a	Modulare Sensoren
8.791	5	8	b	Günstig zu beschaffende Sensoren (Breakout Boards)
4.396	10	4	c	WLAN Anbindung
10.99	2	10	d	Unabhängigkeit von "konventionellen" Anbietern
10.99	2	10	e	Offene Software
5.495	9	5	f	Autonome Energieversorgung
7.692	8	7	g	kabelloses Design
12.09	1	11	h	einfach nachzubauen
2.198	12	2	i	Batteriezustand anzeigen (LED)
2.198	12	2	j	Batteriezustand übermitteln
4.396	10	4	k	unscheinbare Optik
10.99	2	10	l	gut rückbaubar
8.791	5	8	m	hohe Kompatibilität mit Smart Home Systemen
2.198	12	2	n	PCB günstig in der Herstellung
104.4		91		

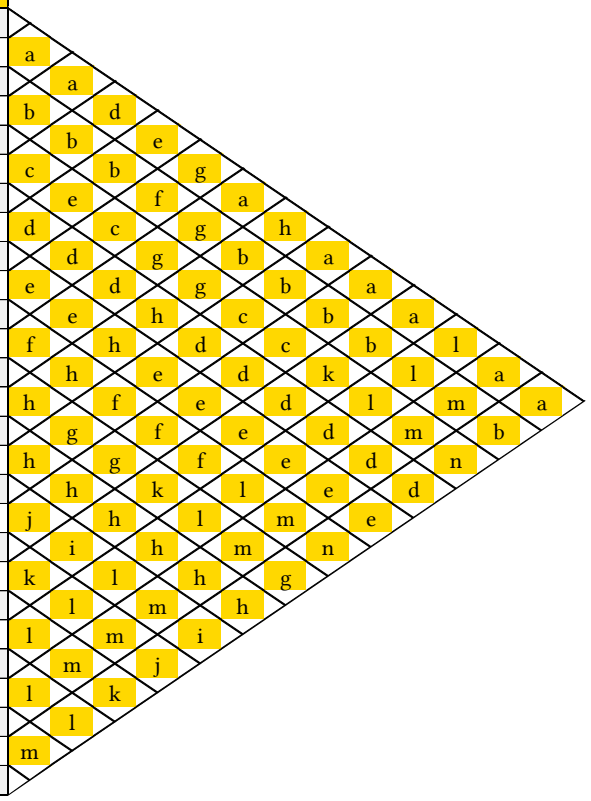


Tabelle 6: Präferenzmatrix Rangliste

Rang	Kriterium
1	einfach nachzubauen
2	Unabhängigkeit von "konventionellen" Anbietern
3	offene Software
4	gut rückbaubar
5	Modulare Sensoren
6	hohe Kompatibilität mit Smart Home Systemen
7	günstig zu beschaffenden Sensoren (Breakout Boards)
8	kabelloses Design
9	autonome Energieversorgung
10	gut rückbaubar
11	WLANAnbindung
12	Batteriezustand übermitteln
13	Batteriezustand anzeigen (LED)
14	PCB günstig in der Herstellung

2 Anforderungsanalyse und Konzeption

Ziel der Anforderungsanalyse ist es die Anforderungen an das Projekt zu ermitteln, zu strukturieren und ein grobes Konzept zu formulieren.

2.1 Ziel und Nutzen Sensorphalanx

Wesentliches Ziel der Diplomarbeit ist die Erstellung eines funktionalen Prototyps einer Sensorphalanx. Diese Phalanx soll in der Lage sein verschiedene Raumdaten zu sammeln und an ein Smart Home Systeme zu übermitteln.

Benutzende haben die Möglichkeit über ein Webinterface die Daten der Sensoreinheit in Echtzeit einzusehen und sofern mehrere Phalanxen vorhanden sind, diese in einer übersichtlichen Darstellung direkt miteinander zu vergleichen. Weiter wird es möglich sein, eine Einsicht in die Langzeitaufzeichnung der Daten nehmen zu können und diese mittels Grafiken zu visualisieren.

2.2 Funktionale Anforderungen

Im Folgenden werden die funktionalen Anforderungen an das Projekt erläutert. Alle Funktionen, die im Zusammenspiel von Sensorphalanx, Smart Home Systeme und Benutzer*in werden festgelegt.

- Die Phalanx sammelt alle Raumdaten und sendet sie an das Smart Home Systeme
- Der/die Nutzer*in kann über ein Webinterface die Istwerte einsehen
- Der/die Nutzer*in hat über das Webinterface Zugriff auf die historischen Daten und kann sich diese grafisch anzeigen lassen

2.2.1 Allgemeine Systemfunktionen

Die allgemeinen Systemfunktionen gliedern sich in:

- Datensammlung: Sammeln und Zusammenstellung für eine Übermittlung an das Smart Home Systeme
- Verarbeitungsfunktionen: Verarbeitung der Informationen im Smart Home Systeme
- Dialogfunktion: Der Nutzer erhält Informationen über ein Webinterface und gegebenenfalls Meldungen bei Über- und Unterschreiten von Grenzwerten

In den nachfolgenden Tabellen werden nur die wichtigsten Funktionen aufgeführt und beschrieben. Dialogfunktionen sind nicht genauer erläutert.

Tabelle 7: Funktionale Anforderungen: Datensammlung

Datensammlung	
Auslesen	Daten der Sensormodule werden zyklisch (60 sec) erfasst
Zusammenführen	Die Sensordaten werden am Ende eines Zyklus mit einem für das Smart Home System eindeutig identifizierbar, zu einer Nachricht zusammengestellt
Senden	Daten werden an das Smart Home Systeme übermittelt

Tabelle 8: Funktionale Anforderungen: Verarbeitungsfunktionen

Verarbeitungsfunktionen	
Ablegen	Die von der Sensorphalanx erhaltenen Daten werden im Smart Home Systeme abgelegt
Auswerten	Aus den gesammelten Daten wird bei gleichen Einheiten der Mittelwert gebildet
Visualisieren	Je nach ausgewählter Ansicht werden die entsprechenden Informationen im Webinterface angezeigt.

2.3 Nicht-funktionale Anforderungen

In den nicht-funktionalen Anforderungen werden alle Eigenschaften festgelegt, die das Projekt später erfüllen soll.

2.3.1 Allgemeine Anforderungen (Rahmenbedingung)

Smart Home Systeme

- Die Basis für das Webinterface wird vom Smart Home Systeme zur Verfügung gestellt und unterliegt den Grenzen dieses Systems, soll aber dennoch benutzerfreundlich und intuitiv zu bedienen sein.

Sensorphalanx

- Nach dem Einbinden in das System soll die Sensorphalanx autonom funktionieren und in Abständen von einer Minute (1 min) die Raumdaten sammeln und an das System übermitteln.
- Es ist sicherzustellen, dass bei nicht Erreichbarkeit des Smart Home Systems keine Daten verloren gehen. Daher werden diese lokal mit Zeitstempel versehen, gesichert und dann bei Wiedererreichbarkeit des Systems die gesammelten Daten übermittelt.
- Die Sensorphalanx soll in der Lage sein mit seiner Energiereserve mindestens 4 Wochen ohne Nutzereingriff funktionieren zu können.
- Um die Umwelt nicht unnötig zu belasten, sollen für die Energieversorgung wiederverwendbare Energiespeicher eingesetzt werden.
- Die Kommunikation zwischen Sensorphalanx und Smart Home Systeme soll verschlüsselt erfolgen können.

2.4 Komponentenauswahl

Da es eine Vielzahl von möglichen Varianten im Bereich Software und Hardware gibt, werden im folgenden Kapitel die verschiedenen Lösungsmöglichkeiten aufgezeigt und der Entscheidungsweg näher beleuchtet.

2.4.1 Smart Home Systeme

Die Basis eines Smart Home Systems stellt die Zentrale dar, die per Funk mit weiteren Komponenten interagiert. Die Zentrale erkennt anhand von angebunden Sensoren Umweltinformationen und reagiert durch das Steuern von Aktoren darauf. Wird die Smart Home Zentrale beispielsweise von einem Bewegungsmelder im Raum über eine Bewegung informiert, kann sie abhängig von anderen Daten wie Helligkeit und Uhrzeit bestimmte Beleuchtungssettings aktivieren. Die Logik des Systems befindet sich somit allein auf der Zentrale, deren sorgsame Auswahl dementsprechend wichtig ist.

Abhängig vom gewähltem Funkstandard des Smart Home Systems (z.B. WLAN oder Bluetooth) stehen unterschiedlich viele Geräte zur Auswahl. Während manche konventionellen Lösungen nur mit Geräten des eigenen Herstellers vernetzbar sind, bieten andere Smart Home Systeme auch die Möglichkeit kompatible Komponenten anderer Hersteller einzubinden. Einen marktübergreifenden Standard gibt es derzeit nicht.

Es gibt grundsätzlich zwei unterschiedliche Herangehensweisen an die Zentrale Smart Home Software: Open Source oder Closed Source.

2.4.1.1 Open Source

Als Open Source bezeichnet man Programmcode, der für die Öffentlichkeit frei zugänglich gemacht wird. Jeder kann diesen einsehen sowie, nach Belieben, verändern, verteilen und weiterentwickeln. (2)

Open Source-Software stützt sich häufig auf Peer-Review und Community-Produktion. Die aus diesem Prozess entstandene Software ist in der Regel flexibler und langlebiger als proprietäre Produkte, da sie von einer Community entwickelt wird und nicht von einem einzelnen Unternehmen, das seine Geschäftsgrundlage sichern möchte. Weit verbreitete Open Source Projekte haben auch den Ruf sichere Software zu sein, da hier viele Augen auf den Code schauen und so Fehler und Sicherheitslücken schneller gefunden werden und gegebenenfalls direkt vom Entdecker gelöst werden können.

2.4.1.2 Closed Source

Close Source oder auch proprietäre Software bezeichnet Software, die die Möglichkeit der Einsicht und Weiterentwicklung stark einschränkt oder z.B. durch Verschlüsselung unmöglich macht. Der Vorteil der Close Source Software ist, dass sie auf klar definierter Hardware eingesetzt werden kann und so passgenau zugeschnitten wird. Weiter bietet diese Art der Software den herstellenden Unternehmen bessere Möglichkeiten der Vermarktung (3).

Als Nachteil ist hier aber zu nennen, dass Weiter ist darauf hinzuweisen das Sicherheitslücken nicht zwingend dem Anbieter gemeldet werden und so negativ genutzt werden können.

2.4.1.3 Entscheidungsbegründung

Aufgrund der Vorgaben aus der Präferenzmatrix die auf Rang zwei „Unabhängigkeit von "konventionellen" Anbietern“ und auf Rang drei „Offene Software“ festlegt, fällt hier die Entscheidung auf eine Open Source Lösung.

2.4.1.4 Open Source Smart Home Systeme

Im Bereich der Open Source Smart Home Systeme gibt es eine schwer zu überblickende Menge an möglichen Softwarevarianten. Da sie Community getrieben entwickelt werden, hält jede Gruppe ihre Variante für die Beste. Das macht eine Auswahl sehr schwierig. Da das Smart Home System in diesem Projekt nur eine untergeordnete Rolle einnimmt, wurde das Auswahlverfahren auf drei Kandidaten eingegrenzt. Diese wurden anhand einer Googelsuche ausgewählt. Als Suchbegriff dienten die Wörter: Open, Source, Smart, Home, Systeme. Ausgewählt wurden die ersten drei Ergebnisse, die direkt auf eine Projektseite verweisen.

Tabelle 9: Google Ergebnisse: Open Source Smart Home Systeme

Name	**Projektseite**
openHAB	https://www.openhab.org/
Home Assistant	https://www.home-assistant.io/
ioBroker	https://www.iobroker.net/

2.4.1.4.1 openHAB

Geschichte

Gemäss Wikipedia wurde das, auf Java geschriebene, Open Source Projekt openHAB im Jahr 2010 erstmalig veröffentlicht. (4) Die aktuelle Version 3.1.0 ist derzeit zum freien Download verfügbar. OpenHAB ist Betriebssystem unabhängig und um zusätzliche Technologien/Protokolle erweiterbar. Die von der Eclipse Public License geschützte und als veröffentlichte Software bietet als Userinterface sowohl die Möglichkeit einer Weboberfläche als auch, Android- oder Apple-iOS Apps. (5)

Verbreitung

Zum Zeitpunkt des Verfassens dieses Textes, waren im Forum von OpenHab 40.400 User registriert und hatten 718.000 Posts verfasst, was auf eine aktive Community schliessen lässt.

Site Statistics

	Last 24 hours	Last 7 days	Last 30 days	All Time
Topics	16	144	504	56.3k
Posts	158	1.3k	5.5k	718k
Users	9	54	246	40.4k
Active Users	754	2.0k	3.3k	—
Likes	35	232	1.3k	125k

Abbildung 2: Screenshot Forum Statistik openHAB (31.10.2021 16:45)

Userinterface

Es gibt ein «standardisiertes» Userinterface auf Web Basis, dieses bietet die Möglichkeit auch Anpassungen vom Nutzer vorzunehmen bis hin eigene Designs zu entwickeln oder Templates zu beziehen.

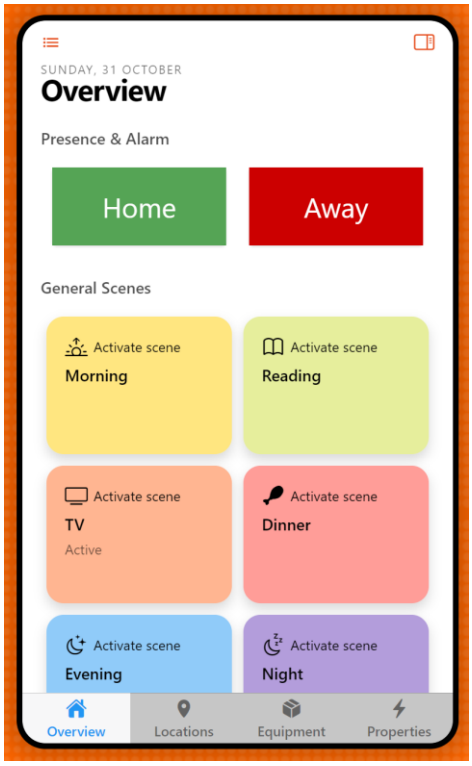


Abbildung 3: Screenshot Overview openHAB 3 Demo

Abbildung 4: Screenshot Temperaturkurve openHAB 3 Demo

Abbildung 5: Screenshot 3D übersichtsplan openHAB 3 Demo

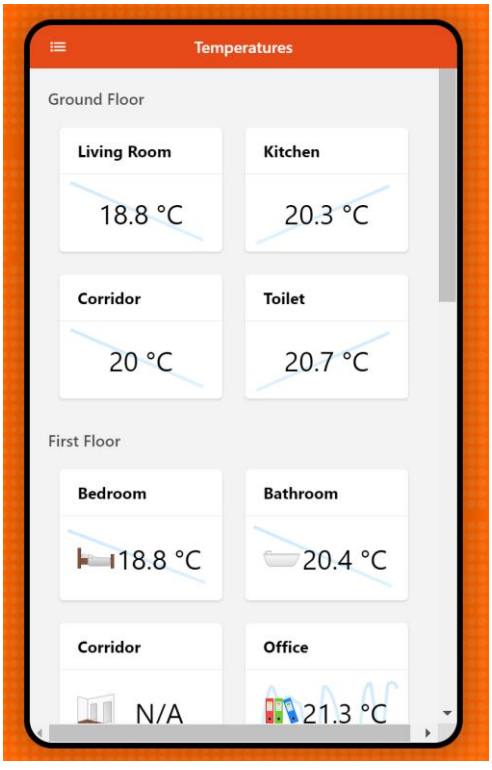


Abbildung 6: Screenshot Temperaturübersicht openHAB 3 Demo

2.4.1.5 Home Assistant

Geschichte

Home Assistant wird seit September 2013 entwickelt und im November 2013 erfolgte die Veröffentlichung der ersten Kernfunktion. Das Smart Home System basiert auf der Programmiersprache Phyton und legt Wert auf die lokale Steuerung und Privatsphäre der Nutzer. Das Projekt verfügt über Open-Source-Anwendungen für Android und iOS. (6)

Verbreitung

Mitte 2020 umfasste das Team der Entwickler mehr als 1930 Mitglieder, die am Kern des Projektes arbeiten. Beim GitHub Bericht «State of the Octoverse» im Jahr 2020 wurde Home Assistant in der Top 10 Liste der Python Projekte, mit den meisten aktiven Mitgliedern (8162) auf Platz zwei geführt. (7)

Zum Zeitpunkt des Verfassens dieses Textes waren im Forum von Home Assistant 110.000 User registriert und hatten 1.600.000 Posts verfasst, was auf eine aktive Community schliessen lässt.

Site Statistics

	Last 24 hours	Last 7 days	Last 30 days	All Time
Topics	103	681	2.7k	130k
Posts	1.1k	7.8k	34.0k	1.6M
Users	124	721	3.0k	110k
Active Users	5.6k	15.3k	25.9k	—
Likes	236	1.6k	7.0k	321k

Abbildung 7: Screenshot Forum Statistik Home Assistant (31.10.2021 16:46)

Useriterface

Auch Home Assistant bietet die Möglichkeit eines webbasierten Userinterfaces, das modular anpassbar ist und ebenfalls die Möglichkeit bietet Templeise für verschiedene Designs einzubinden.

Abbildung 8: Screenshot Temperaturkurve ARS Home Design / Home Assistant Demo

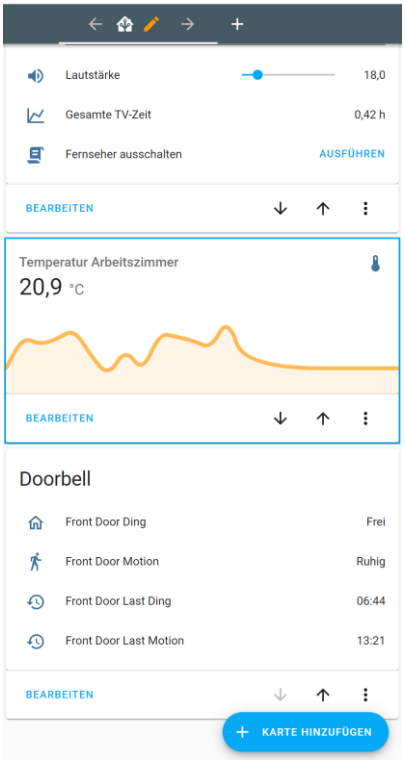


Abbildung 9: Screenshot Temperaturkurve ARS Home Design / Home Assistant Demo

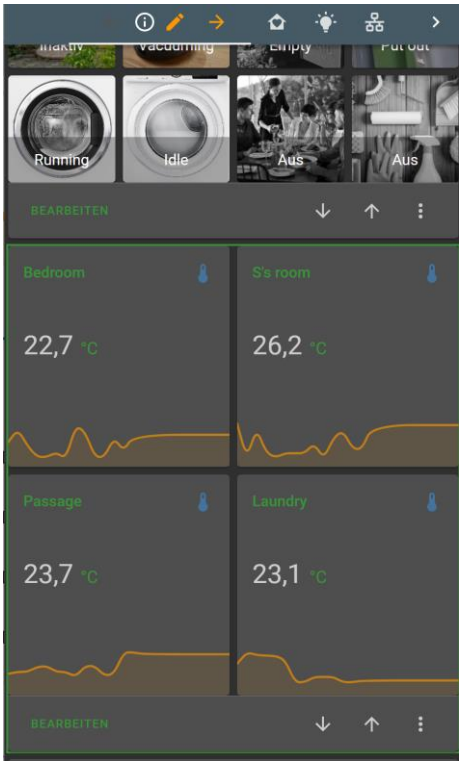


Abbildung 10: Screenshot Temperaturkurven Isa`s mobile friendly LL Design / Home Assistant Demo

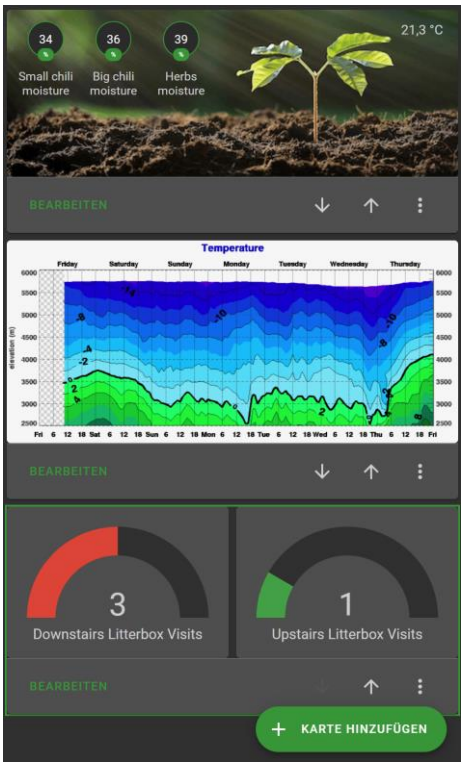


Abbildung 11: Screenshot Temperaturkurven Isa`s mobile friendly LL Design / Home Assistant Demo

2.4.1.5.1 ioBroker

Geschichte

IoBroker hat sich aus dem CCU.IO Projekt entwickelt, existiert aber seit 10.August 2014 als eigenständiges Projekt auf GitHub. Als Basisprogrammiersprache dient JavaScript. (8) Das Projekt verfügt über keine Anwendungen für Android und iOS, die direkt von ioBroker unterstützt werden, lediglich Apps von Drittanbieter sind verfügbar.

Verbreitung

Zum Zeitpunkt des Verfassens dieses Textes waren im Forum von Home Assistant 42.700 User registriert und hatten 694.000 Posts verfasst, was auf eine engagierte Community schliessen lässt.

Abbildung 12: Screenshot Forum Statistik ioBroker (31.10.2021 16:47)

Userinterface

Das ioBroker Webinterface ist Modular aufgebaut und ermöglicht den User die freie Gestaltung der Darstellung, es besteht auch die Möglichkeit die Symbole mit anderen Kits auszutauschen und zu ergänzen.

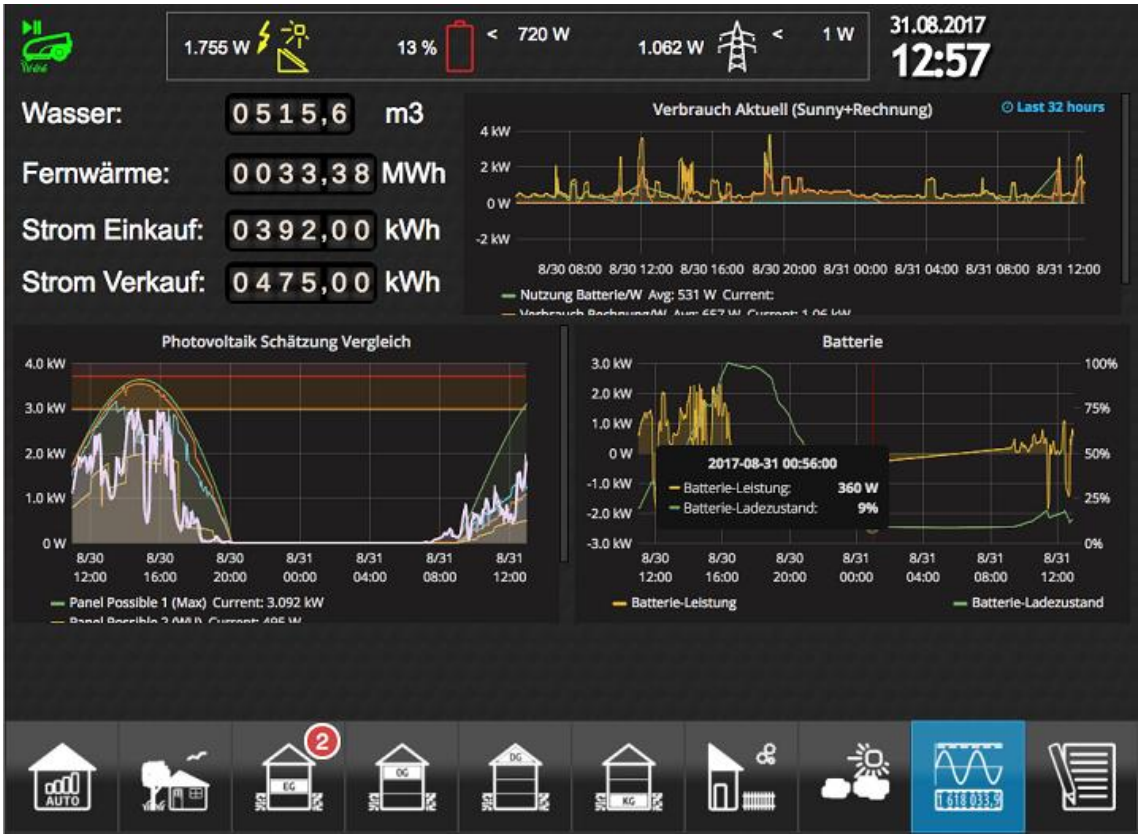


Abbildung 13: Screenshot Beispiel für eine selbst erstellte VIS-Benutzeroberfläche ioBroker (9)

2.4.1.5.2 Entscheidungsbegründung

Da im Grunde kein Smart Home System sich besonders hervorhebt und für das Projekt nicht relevant ist für welches System sich entschieden wird, da eine Adaption auf an anders Smart Home Systeme im Anschluss an die Diplomarbeit jederzeit möglich ist, wurde für die Auswahl des Smart Home System ein besonderes Augenmerk auf die Aktivität der dahinter versammelten Community gelegt. Da innerhalb dieser Gruppe gegeben falls das grösste Interesse am Ergebnis dieser Diplomarbeit vorliegen dürfte und im Bedarfsfall auch der Pool an potenziellen Unterstützern bei Problemen mit dem System am grössten ist.

Tabelle 10: Entscheidungsübersicht Smart Home System

Eigenschaften	openHAB	Home Assistant	ioBroker
Regestreite User	Negativ	Positiv	Negativ
Ansprechendes Interface	Positiv	Positiv	Negativ

2.4.2 Kommunikation / Protokoll

Der gesamte Datenaustausch mit der Sensoreinheit erfolgt über WLAN, da dieses in der Regel in Wohnungen flächendeckend vorhanden ist. Datenprotokolle für Hausautomationssysteme basieren meistens auf dem Layer 7 (Anwendungsschicht) des ISO/OSI-Modells. In diesem Abschnitt werden mehrere Datenprotokolle verglichen und ein passendes für dieses Projekt festgelegt.

2.4.2.1 Message Queuing Telemetry Transport

Beim Message Queuing Telemetry Transport (MQTT) Protokoll handelt es sich um ein offenes Netzwerkprotokoll, das zur Maschine-zu-Maschine-Kommunikation eingesetzt wird und auf einer Client Server («Broker») Struktur basiert. Die Clients senden dem Server Nachrichten mit einem Topic, diese werden vom Server hierarchisch eingestuft und an die Clients, die diese Topics abonniert haben weitergeleitet.

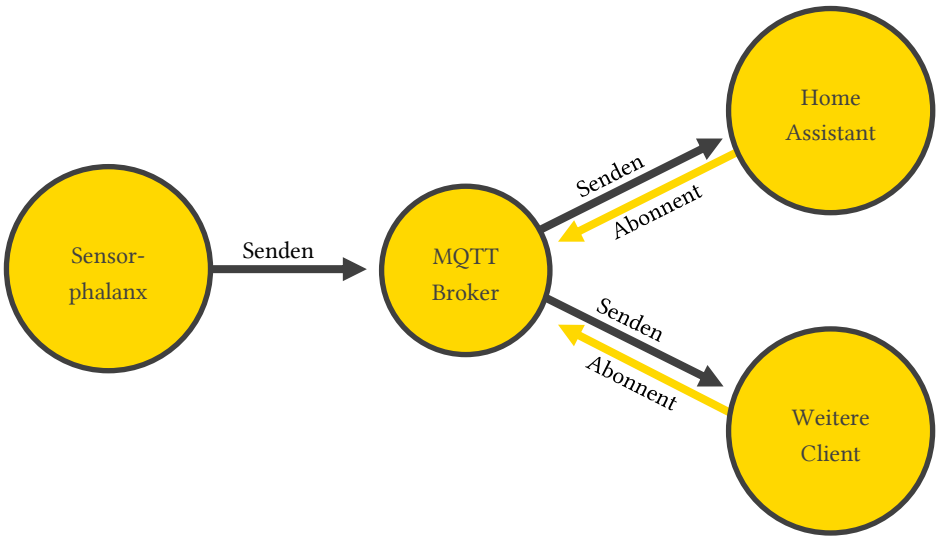


Abbildung 14: Funktionsweise MQTT Senden/Abonnieren Architektur

Das Topic einer Nachricht kann sich wie folgt zusammensetzen: `Wohnung/Kinderzimmer/Temperatur.`

Weiter bekommen die Nachrichten noch einen Quality of Service mit, der wie folgt definiert ist:

at most once die Nachricht wird einmal gesendet und kommt bei Verbindungsunterbrechung möglicherweise nicht an.

at least once die Nachricht wird so lange gesendet, bis der Empfang bestätigt wird, und kann beim Empfänger mehrfach ankommen.

exactly once hierbei wird sichergestellt, dass die Nachricht auch bei Verbindungsunterbrechung genau einmal ankommt.

MQTT bietet auch die Möglichkeit einen «letzten Willen» zu definieren, diese Nachricht wird dann an alle Abonnenten verschickt, wenn die Verbindung zum Clienten unterbrochen wird. (10)

Um die Kommunikation zu steuern verwendet MQTT Kontrollpakete, in der folgenden Tabelle sind alle Kontrollpakete aufgelistet.

Tabelle 11: Kontrollpakettypen (7)

Name	Wert	Flussrichtung	Beschreibung
Reserviert	0	Client zu Server	Reserviert
CONNECT	1	Server zu Client	Verbindungsanfrage
CONNACK	2	Client zu Server oder Server zu Client	Verbindungsbestätigung
PUBLISH	3	Client zu Server oder Server zu Client	Publish Nachricht
PUBACK	4	Client zu Server oder Server zu Client	Publish Bestätigung (QoS 1)
PUBREC	5	Client zu Server oder Server zu Client	Publish empfangen (QoS 2 Zulieferung part 1)
PUBREL	6	Client zu Server oder Server zu Client	Publish veröffentlicht (QoS 2 Zulieferung part 2)
PUBCOMP	7	Client zu Server oder Server zu Client	Publish vollständig (QoS 2 Zulieferung part 3)
SUBSCRIBE	8	Client zu Server	Subscribe Anfrage
SUBACK	9	Server zu Client	Subscribe Bestätigung
UNSUBSCRIBE	10	Client zu Server	Unsubscribe Anfrage
UNSUBACK	11	Server zu Client	Unsubscribe Bestätigung
PINGREQ	12	Client zu Server	Ping Anfrage
PINGRESP	13	Server zu Client	Ping Antwort
DISCONNECT	14	Client zu Server oder Server zu Client	Disconnect Nachricht
AUTH	15	Client zu Server oder Server zu Client	Authentifizierungsaustausch

2.4.2.2 Hypertext Transfer Protocol

Das Hypertext Transfer Protocol (HTTP) wurde 1991 eingeführt und diente dazu Webseiten in Webbrowser zu laden, ist aber nicht darauf beschränkt. Das grösste Problem in Bezug auf die Übermittlung von Daten ist der grosse Overhead, das bedeutet im Vergleich zu alternativen Protokollen entsteht eine deutlich grössere Datenmenge die zu übertragen ist. Von Vorteil ist das sich HTTP schnell erlernen lässt und viele Entwickler sich mit dem Protokoll bereits auskennen. Aufgrund seiner grossen Ausbreitung lässt es sich schnell implementieren dadurch sind schnelle Ergebnisse möglich. (11)

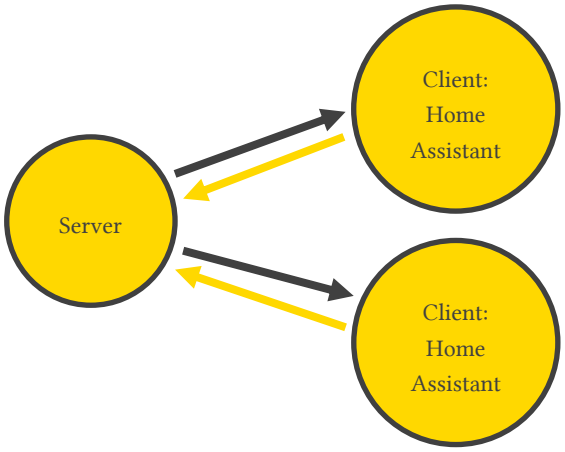


Abbildung 15: Funktionsweise HTTP Server/Client-Architektur

2.4.2.3 Extensible Messaging and Presence Protocol

Das Extensible Messaging and Presence Protocol (XMPP), auch bekannt unter dem Namen Jabber, basiert auf den XML-Standard und ist beliebt im Bereich der Instand Massanger. Da das Protokoll auf XML basiert, hat auch dieses Protokoll den Nachteil des grossen Overheads, der bei jedem Datenaustausch immer mitgesendet werden muss. XMPP ermöglicht aber auch die direkte Kommunikation zwischen verschieden Clients ohne auf einer Serverstruktur zurückgreifen zu müssen.

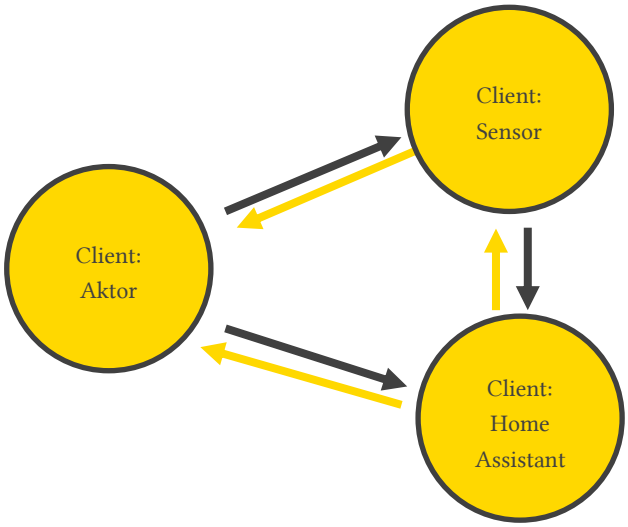


Abbildung 16: Funktionsweise XMPP Client/Client-Architektur

2.4.2.4 Entscheidungsbegründung

Für dieses Projekt wurde sich für das MQTT Protokoll entschieden, da dieses eine kleine Belastung der Datenverbindung verursacht. So ist zu erwarten, dass die Datenverbindung auch bei schlechter WLAN Verbindung gut funktioniert.

Tabelle 12: Entscheidungsübersicht Protokolle

Eigenschaften	MQTT	HTTP	XMPP
Server benötigt	Negativ	Negativ	Positiv
Datengrösse (Overhead)	Positiv	Negativ	Negativ
Netzstabilität nötig	Positiv	Negativ	Negativ

2.4.3 Kommunikation innerhalb der Sensorphalanx

Die Kommunikationsprotokolle innerhalb der Sensorphalanx sind von den Modulen vorgegeben und somit nicht frei wählbar. Zu erwarten sind die Protokolle: Inter-Integrated Circuit und Serial-Bus, die im Folgenden näher erläutert werden.

2.4.3.1 Inter-Integrated Circuit

Inter-Integrated Circuit (I²C) im Deutschen gesprochen als I-Quadrat-C oder englisch I-Squared-C oder I²C wurde 1982 von Philips Semiconductors entwickelt und wird hauptsächlich zur Kommunikation verschiedener Elemente innerhalb eines Geräts verwendet. Das ursprüngliche Patent ist am 1. Oktober 2006 ausgelaufen, so dass keine Lizenzgebühren für die Benutzung von I²C mehr anfallen und so eine grosse Verbreitung stattgefunden hat.

I²C basiert auf einem Master-Slave-Bus Konzept. Initialisiert wird der Datentransfer vom Master, der die adressierten Slaves direkt anspricht.

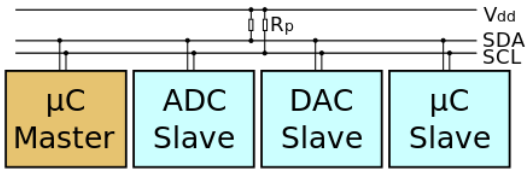


Abbildung 17: I²C-Bus mit Master und Slaves (12)

Für die Kommunikation werden zwei Signalleitungen benötigt: Takt- (SCL = Serial Clock) und Datenleitung (SDA = Serial Data). Beide liegen mit Pull-up-Widerständen an der Versorgungsspannung an. Der I²C-Bus arbeitet mit positiver Logik, das bedeutet ein High-Pegel auf der Datenleitung entspricht einer logischen „1“, der Low-Pegel einer „0“.

Der Master gib den Bustakt vor. Das Taktsignal liegt nur während der Datenübertragung an. Wenn der Slave mehr Zeit braucht, als durch den Takt vorgegeben ist, kann er zwischen der Übertragung mittels Clock-Stretching einzelner Bytes der Taktleitung auf „low“ halten und so den Master bremsen.

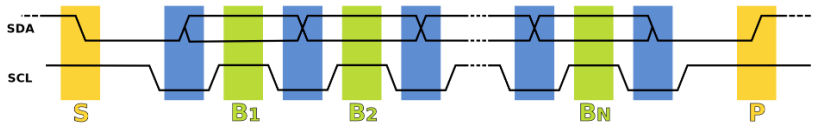


Abbildung 18: Zeitverhalten am I²C-Bus: Die Datenbits B1 bis BN werden vom Start-Signal (S) und dem Stopp-Signal (P) eingerahmt. (12)

Slaves haben eine festgelegte Adresse und werden mittels des ersten, vom Master gesendeten, Byte angesprochen. Das achte Bit (R/W-Bit) informiert den Slave, ob dieser etwas empfangen oder senden soll. Durch die Verwendung der 7 Bit als Adresse sind 112 Knoten auf dem Bus erlaubt (für Sonderzwecke sind 16 der 128 möglichen Adressen reserviert). Da eine feste Adressierung zu Konflikten auf dem Bus führt, weil Adressen nicht mehr einzigartig sind, gibt es bei einigen Modulen die Möglichkeit durch Löten von Brücken die Adressen anzupassen. Alternativ müssen bei gleichen Adressen von Modulen, die zum Zeitpunkt des Auslesens nicht benötigten Module abgeschaltet werden.

Im Verlauf der Zeit wurde auf eine 10-Bit-Adressierung, die durch Nutzung von 4 der 16 reservierten Adressen abwärtskompatibel zum 7-Bit-Standard ist, umgestellt. Dadurch ergeben sich bis zu 1136 Knoten auf einem Bus. (12)

Der I²C-Bus gilt als sehr störanfällig, da häufig mit Übersprechen, Rauschen, EMV-Problemen oder mit Kontaktproblemen (Stecker, Buchsen) zu rechnen ist. Weiter ist er auch nicht für die Überbrückung von grossen Entfernungen geeignet.

2.4.3.2 Universal Asynchronous Receiver Transmitter

Universal Asynchronous Receiver Transmitter (UART) ist eine elektronische Schaltung, zur Realisierung serieller Schnittstellen. Die Aufgabe der UART-Schnittstelle ist das Senden und Empfangen von Daten über eine Datenleitung und ist der Standard der seriellen Schnittstellen an PCs und Mikrocontrollern.

Ähnlich wie I²C werden die Daten im digitaler Datenstrom mit einem festen Rahmen übertragen. Dieser hat am Anfang ein Start-Bit, fünf bis maximal neun Datenbits (abhängig von der Anwendung), einem optionalen Party-Bit und am Ende eine oder zwei Stopp-Bits. Anders als I²C braucht der Sender dem Empfänger keinen Sendetakt zu übermitteln, da der Empfänger den Takt des Senders aus dem Takt der Datenleitung errechnet und sich mit Hilfe des Start- und Stoppbits synchronisiert.

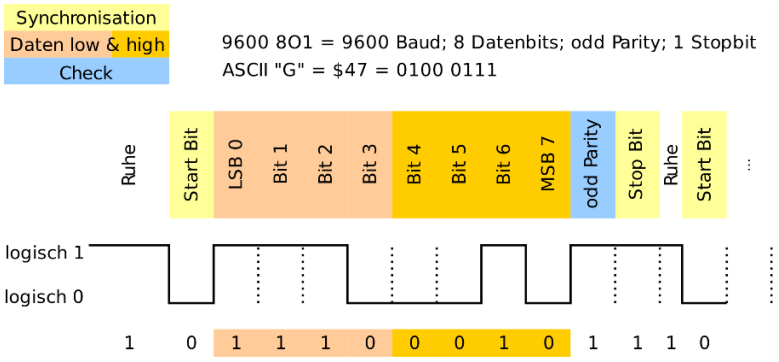


Abbildung 19: Der asynchrone serielle Datenstrom, wie ihn ein sog. CMOS-UART erzeugt (logisch 0 und 1) (13)

Damit die Baugruppen miteinander kommunizieren können, müssen die Empfangsleitung (Rx) der einen und die Sendungsleitung (Tx) der anderen Baugruppe am Stecker gegenüberstehen. Dadurch sind in der Regel nur Master und Slave Verbindungen möglich, Ausnahmen werden lediglich durch die Verwendung eins Null-Modem-Kabel ermöglicht. Dieses erlaubt eine Master mit Master bzw. Slave mit Slave Kommunikation.

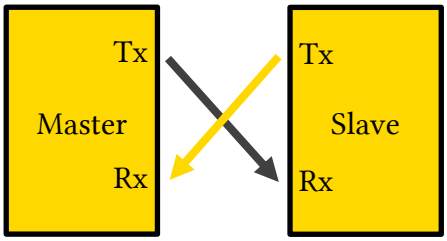


Abbildung 20: Datenaustausch zwischen Master und Slave mit UART Schnittstelle

2.4.4 Hardware Smart Home System

Home Assistant bietet viele Varianten der Installation an, neben den klassischen x86 und x64er Varianten auf Linux und Windows Basis, verfügt es auch über eine eigens entwickelte Hardware namens «Home Assistant Blue». Diese enthält bereits eine vollständige Installation. (14)

Für dieses Projekt wird eine Raspberry Pi verwendet. Dieser stellt eine gut verfügbare Hardware dar, die sowohl in der Variante PI3 und PI4 ausreichend Leistung für die nötige Software zur Verfügung stellt.

2.4.4.1 Raspberry Pi

Ursprünglich wurde der Raspberry Pi vom Ingenieure Eben Upton entwickelt, um Kindern und Jugendliche eine kostengünstige Möglichkeit zu bieten das Programmieren zu lernen. Mittlerweile erfreut sich der Raspberry Pi aber grosser Beliebtheit bei Bastler*innen jeglichen Alters und wurde so zum meist verkauften PCs Grossbritanniens (Mai 2021 40 Millionen Exemplare). Ein erster Prototyp wurde bereits 2006 veröffentlicht. In dem darauffolgenden Boom in der Entwicklung von Smartphone Komponenten kamen aber bald leistungsstarke ARM Prozessoren auf dem Markt, so das Anfang 2012 die erste Serienmodelle des Raspberry Pi für die breite Öffentlichkeit verfügbar waren (15). Der

checkkartengrosse Minicomputer ist mittlerweile in der vierten Generation angekommen. Zusätzlich zu den Modellen 1-4 in ihren Varianten A, A+, B und B+(nicht jede Variante ist für jedes Model verfügbar) gibt es noch zwei Zero Varianten, die sich hauptsächlich in ihrem Formfaktor und Anschlüssen von den normalen Modellen unterscheiden. Weiter gibt es noch mehrere Computermodule, die in ihrer Bauform an DDR2 Ram erinnern und keine üblichen I/O Anschlüsse aufweisen. Diese lassen sich aber mit einem I/O Board bei Bedarf nachrüsten.

2.4.4.1.1 Version 3 B Spezifikation

Tabelle 13: Auszug Hardwarespezifikation Raspberry Pi 3 B (13) (Anhang 2)

Komponente	Spezifikation
CPU	ARM Cortex-A53
Kern(e)	4
CPU Takt	1200 MHz
RAM	1024 MB
LAN	10/100 Mbit/s
WLAN	b/g/n
Bluetooth	4.2
Spannungsversorgung	5 V DC / 2.5 A

2.4.4.1.2 Version 3 B+ Spezifikation

Die Version 3B+ ist die letzte Version des Raspberry Pi 3B (Anhang 3)

Tabelle 14: Auszug Hardwarespezifikation Raspberry Pi 3 B+ (13)

Komponente	Spezifikation
CPU	ARM Cortex-A53
Kern(e)	4
CPU Takt	1400 MHz
RAM	1024 MB
LAN	10/100/1000 Mbit/s
WLAN	b/g/n/ac
Bluetooth	4.2
Spannungsversorgung	5 V DC / 2.5 A

2.4.4.1.3 Version 4 B Spezifikation

Die Raspberry PI 4 B Variante verfügt über vier Ausführungen, die sich lediglich in der Grösse ihres Arbeitsspeichers unterscheiden (Anhang 4)

Tabelle 15: Auszug Hardwarespezifikation Raspberry Pi 4 B (14)

Komponente	Spezifikation
CPU	ARM Cortex-A72
Kern(e)	4
CPU Takt	1500 MHz
RAM	1024 / 2048 / 4096 / 8192 MB
LAN	10/100/1000 Mbit/s
WLAN	b/g/n/ac
Spannungsversorgung	USB Typ C 5 V DC / 3 A

2.4.4.1.4 Entscheidungsbegründung

Aus Gründen der grösseren Leistungsreserve für weitere Projekte und Erweiterungen am Smart Home System ist ein Raspberry Pi Model 4 einem Model 3 vorzuziehen. Allerding sind auf Grund der momentanen Chipkrise (16), diverse Modelle der Raspberry Pi schwer zu bekommen, so dass bei der Bestellung darauf zu achten ist, welches Modell in einer nützlichen Frist, zu einem angemessenen Preis zu bekommen ist.

Tabelle 16: Entscheidungsübersicht Smart Home System Hardware

Eigenschaften	Raspberry Pi 3 B	Raspberry Pi 3 B+	Raspberry Pi 4 B
Prozessortakt	Negativ	Negativ	Positiv
Arbeitsspeicher	Negativ	Negativ	Positiv
LAN 1000 Mbit/s	Negativ	Positiv	Positiv
Lieferbarkeit	Negativ	Positiv	Negativ

Bei der Bestellung wurde sich aus oben genannten Gründen für ein «Das Reichelt Raspberry PI 3 B+ All-In-Bundle» entschieden.

2.4.5 Hardware Sensorphalanx

In den folgenden Kapiteln werden die verschieden Varianten der möglichen Hardware für die Sensorphalanx vorgestellt und beschrieben. Die Auswahl erfolgt im Anschluss an jede Kategorie durch eine einfache Abwägung unter Berücksichtigung, der im Kapitel 1.6, aufgelisteten Kriterien für diese Projekt.

2.4.5.1 Mikroprozessor

Der Mikroprozessor bildet die zentrale Einheit der Sensorphalanx. Seine Hauptaufgabe besteht darin die Daten der Sensoren zu sammeln und vor dem Übermitteln an das Smart Home System aufzubereiten. Weiter muss er die Möglichkeit einer Interrupt Funktion aufweisen, da gegebenenfalls auch Informationen gesammelt werden müssen die unmittelbar übertragen werden sollen (z.B. Bewegung).

2.4.5.1.1 Arduino

Am häufigsten verbreitet in der Maker*innen Szene sind verschiedene Varianten von Arduino Boards als Basis für Projekte. Diese aus Soft- und Hardware bestehende Physical-Computing-Plattform ist quelloffen und besteht aus einer I/O-Plattform mit einem AVR Mikroprozessor und einer Entwicklerumgebung. Diese I/O-Plattformen haben im Laufe der Zeit viele Entwicklungsschritte durchlaufen und, da sie quelloffen sind, auch viele kompatibel Nachahmer gefunden. In der Regel haben Arduino Plattformen keine WLAN Anbindung, diese kann aber mittels Shields nachgerüstet werden. Bei diesen Shields handelt es sich oftmals um Platinen die mittels eines ESP8266 oder ESP32 Microprozessors eine WLAN Schnittstelle erzeugen. Da es mittlerweile aber kompatible I/O-Plattformen gibt, die direkt auf einen ESP8266 oder ESP32 aufbauen, erscheint es logisch diese auch direkt zu verwenden.

2.4.5.1.2 ESP8266

Der ESP8266 (Anhang 5) ist ein, von der chinesischen Firma espressif entwickelter, 32-Bit-Mikrocontroler mit geringem Leistungsbedarf. Er ermöglicht auf Grund seiner integrierten Wireless Local Area Network (WLAN nach IEEE 802.11 b/g/n) und seiner offenen Bauweise die Entwicklung von kompakten Aktor- und Sensormodulen.

In einer Blog-Artikel Serie analysiert, Thorsten von Eicken, den Energieverbrauch unterschiedlicher ESP Module in verschiedenen Betriebszuständen (17).

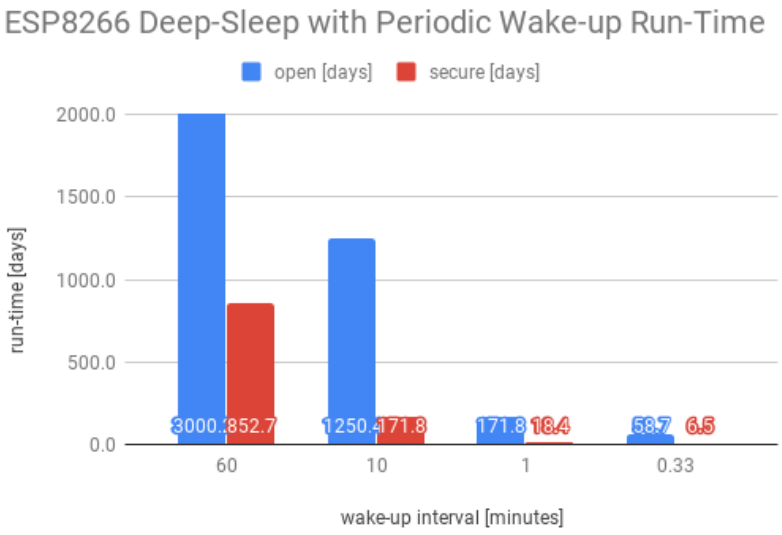


Abbildung 21: ESP8266 Deep-Sleep Laufzeit bei periodischem wecken mit einem 1000mAh Akku (Blau = offenes WLAN Rot= Sicheres WLAN) (17)

Technische Daten:

Prozessor:	Tensilica Xtensa mit 80 MHz
Memory:	Flash 4MB SRAM 160KB
WiFi / Bluetooth	IEEE 802.11 b/g/n bis max. 72,2 Mb/s
Ein/Ausgänge	16 GPIOs
PWM	8
Schnittstellen	SPI, I²C, I2S, UART
Stromverbrauch	80mA

Die Firma Ai-Thinker hat als günstiger Drittanbieter verschiedene Modellvarianten des ESP2866 auf dem Markt gebracht. Diese haben sich auf dem Markt auch entsprechend durchgesetzt.

2.4.5.1.3 ESP32

Der ESP32 (Anhang 6) ist der Nachfolger des ESP2688 und bietet neben einer leistungsfähigeren CPU zusätzlich die Möglichkeit einer Bluetooth Schnittstelle und deutlich mehr GPIO Pins. Diese können alle als Analogeingang oder -ausgang genutzt werden.

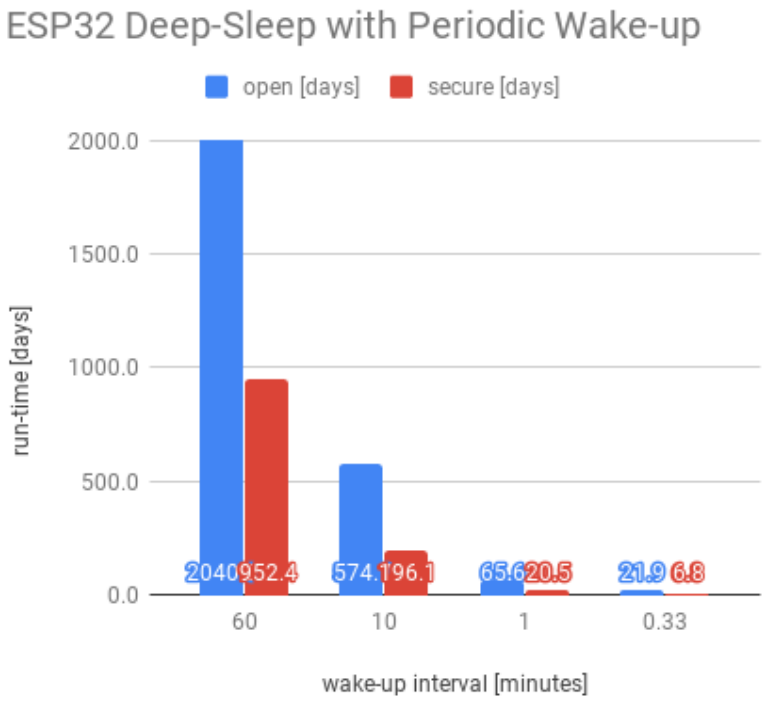


Abbildung 22: ESP32 Deep-Sleep Laufzeit bei periodischem wecken mit einem 1000 mAh Akku (Blau = offenes WLAN Rot= Sicheres WLAN) (18)

Technische Daten:

Prozessor:	Xtensa dual-core 32bit LX6 mit 160 bis 240 MHz
Memory:	Flash 4 MB SRAM 520 KB
WiFi / Bluetooth	802.11 b/g/n – 2.4 GHz bis max. 150 Mb/s Bluetooth Low Energy
Ein/Ausgänge	48 GPIOs
PWM	bis 48 möglich
Schnittstellen	SPI, I²C, I2S, UART, CAN bus 2.0
Stromverbrauch	260 mA

Besonders anzumerken ist, dass Thorsten von Eicken in seinem Artikel auf ein spezielles ESP32 Board aufmerksam macht, dass es eine aussergewöhnlich kleine Stromaufnahme (13uA-15uA) im Deep Sleep Modus hat. (18)

2.4.5.1.4 Entscheidungsbegründung

Um die angestrebte Akkulaufzeit für dieses Projekt zu erreichen ist es wichtig eine möglichst effiziente Ausnutzung der zur Verfügung stehenden Energie zu erreichen. Daher ist es sinnvoll den Energieverbrauch der beiden ESP varianten miteinander zu vergleichen. Der Fokus liegt nicht auf dem im Datenblatt angegebenen Strombedarf. Vielmehr liegt er auf dem Verbrauch im Deep-Sleep Modus, da dieser bei einer zyklischen Abfrage der Sensordaten am Häufigsten aktiv ist.

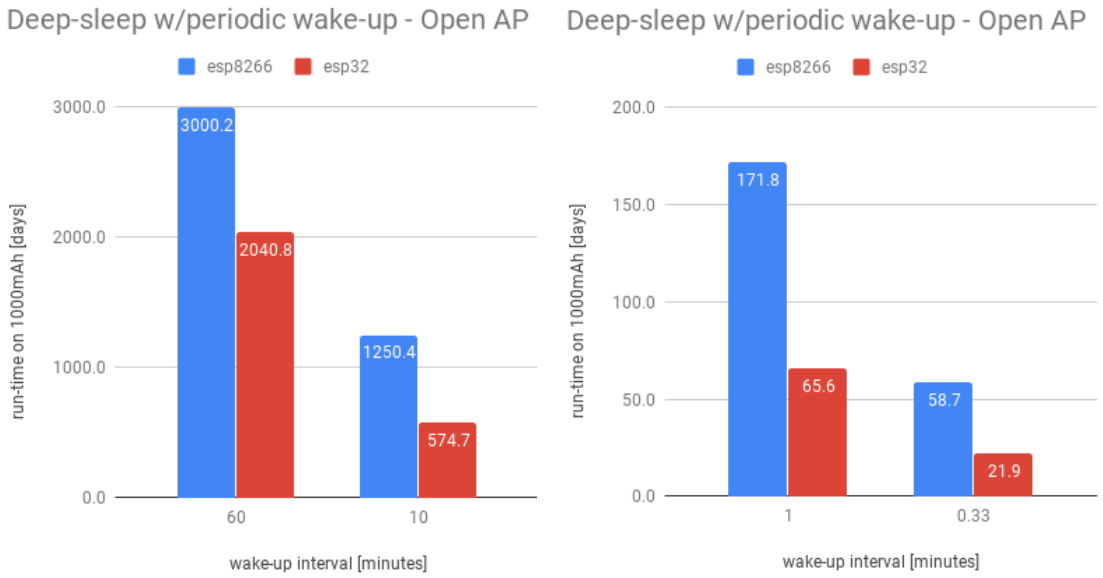


Abbildung 23: ESP8266 6 ESP32 im Vergleich Deep-Sleep Laufzeit bei periodischem wecken mit einem 1000 mAh Akku (19)

Als Ergebnis dieses Vergleichs wird sich für eine ESP32 Board entschieden. Es wird versucht das «EzSBC ESP32-01 Breakout and Development Board» (Anhang 7) zu erhalten. Da der Artikel aber ausschliesslich in den USA bestellt werden kann, wird als alternative das «NodeMCU ESP32» (Anhang 8) Bord der Firma joy-it verwendet.

Tabelle 17: Entscheidungsübersicht Sensor CPU

Eigenschaften	ESP8266	ESP32
Energieverbrauch im Deep Sleep	Negativ	Positiv
Anzahl GPIO-Pins	Negativ	Positiv
Bluetooth	Negativ	Positiv
Anzahl Schnittstellen	Negativ	Positiv

2.4.5.2 Energieversorgung

Aus der Tabelle 17: Präferenzmatrix Rangliste, ist zu entnehmen, dass ein kabelloses Design, sowie eine autonome Energieversorgung, wichtige Kriterien der Sensorphalanx

sind. Daher muss eine Möglichkeit gefunden werden die benötigte Energie bereitzustellen. Dazu gibt es verschiedene Möglichkeiten.

Zum einen wären da handelsübliche Batterien wie beispielsweise Alkali-Batterien oder Zink-Kohle. Diese sind aber auszuschliessen, da sie nur einmalig genutzt werden können und somit Wegwerfprodukte sind.

Weiter gibt es noch die Variante Akkumodule zu verwenden. Diese bieten sowohl die Möglichkeit, bei entsprechender Vorbereitung, direkt an der Phalanx geladen zu werden als auch ausserhalb. Als Möglichkeiten kommen hier Lithium-Polymer-Akkumulatoren (LiPo) oder Lithium-Ionen-Akkumulatoren (Li-Ion) in Frage. Diese werden in dem nächsten Kapitel Unterkapiteln genauer betrachtet.

2.4.5.2.1 Lithium-Ionen-Akkumulator

Lithium-Ionen-Akkumulator ist ein Oberbegriff für Akkumulatoren auf Lithium Basis. Sie haben ein hohes Energie pro Masse Verhältnis reagieren aber empfindlich auf Tiefenentladung und Überladung. Dies kann bis zur Zerstörung führen, weshalb eine Schutzschaltung nötig ist. Handelsübliche Bauformen sind zylindrische Zellen, die eine feste Zellenbezeichnung, eine typische Kapazität und eine definierte Abmessung haben.

Tabelle 18: Auszug Zellendefinition von Lithium-Ionen-Akku (19)

Zellenbezeichnung	Kapazität (in Ah)	Abmessungen (ø × l in mm)
14650	0,9–1,6	14 × 65
16340	0,6–1,0	16 × 34
16500	0,8–1,2	16 × 50
16650	2–3	16 × 65
17500	0,7–1,2	17,3 × 50
17650	1,2–2,5	17 × 65
18350	0,7–1,2	18 × 35
18500	1,1–2,2	18,3 × 49,8
18650	0,8–3,5	18,6 × 65,2

Ein grosser Vorteil von Lithium-Ionen-Akkumulatoren ist, dass sie keinem Memory-Effekt unterliegen, wie er bei Akkumulatoren auf NiCd (Nickel-Cadmium) oder NIMH (Nickelmetallhydrid) Basis vorkommt. Zu erwähnen wäre noch das Lithium-Ionen Akkumulatoren einer Alterung unterliegen, die stark von der Umgebungstemperatur abhängig ist. Als Grund hierfür gelten parasitäre unumkehrbare chemische Reaktionen. (20)

Gefahren beim Umgang mit Lithium-Ionen-Akkus

Mechanische Belastungen, wie in die Zelle eindringende Objekte, können innerhalb der Zelle zu Kurzschlüssen führen. Die daraus resultierenden hohen Ströme führen zur Erhitzung des Akkumulators und können zum Entzünden des Gehäuses führen. Eine chemische Reaktion kann bei einem geladenen Akku zu Überhitzung führen und so ein thermisches Kettenreaktion auslösen, welches die im Akku gespeicherte Energie innert kürzester Zeit in Form von Wärme freisetzt. (20)

2.4.5.2.2 Lithium-Polymer-Akkumulator

Lithium-Polymer-Akkumulatoren sind spezielle Bauformen des Lithium-Ionen-Akkus. Die Besonderheit bildet hier die Konsistenz des Elektrolyts. Der Elektrolyt ist beim Lithium-Polymer-Akku in einer festen bis gallertartigen Folie auf Polymerbasis ver-baut. Dies ermöglicht eine im Design freiere Gestaltung der Zellen, zum Beispiel als flache Zelle. Der Nachteil der freien Gestaltung liegt darin, dass Bauformen sich än-dern können oder ganz vom Markt verschwinden, was eine Nachbeschaffung er-schweren könnte.

Gefahren beim Umgang mit Lithium-Polymer-Akkus

Ebenso wie Lithium-Ionen-Akkus sind Lithium-Polymer-Akkus empfindlich gegen Beschädigungen, Überladen, Tiefentladen, zu hohe Ströme sowie Betreiben ausserhalb des Temperaturbereichs (0-60°C).

2.4.5.2.3 Entscheidungsbegründung

Für dieses Projekt werden Lithium-Ionen-Akkus vom Typ 18650 (Anhang 9) verwendet, da diese eine einheitliche Grösse haben und somit mittels Sockel am PCB befestigt werden können. Aufgrund der standardisierten Bauform ist auch eine einfache Beschaffung möglich.

Tabelle 19: Entscheidungsübersicht Lithium-Akku

Eigenschaften	Lithium-Ionen	Lithium-Polymer
Feste definierte Bauform	Positiv	Negativ
Einfache Beschaffung	Positiv	Negativ
Anpassbare Bauform	Negativ	Positiv
Sockelmontage ohne Stecker	Positiv	Negativ

2.4.5.3 Laden

Das Laden erfolgt, in der Prototypenvariante auf Lochraster, mittels einer fertigen PCB auf Basis des IP5306 Battery Management ICs (Anhang 10). Dieser ermöglicht es den Akku sowohl zu laden als auch den Betreib der CPU während des Ladevorgangs nicht zu unterbrechen. Darüber hinaus verfügt er über alle nötigen Sicherheitsfunktionen, um die Lithiumakkus zu schützen. Der IC bietet gemäss Datenblatt in einigen Varianten die Möglichkeit einer I²C Kommunikation, was in der PCB-Variante ermöglicht diesen direkt auszulesen.

2.4.5.4 Sensoren

Bei der Auswahl ist zu berücksichtigen, dass die Sensorphalanx modular aufgebaut werden soll. Dadurch soll der/die Endnutzer*in die Möglichkeit gegeben werden aus verschiedenen Modulen selbst die für den Bedarf passenden Sensoren auswählen zu können.

Das bedeutet, dass für den Prototypen auch Sensortypen berücksichtigt werden können, dessen Werte schon von anderen Sensoren abgedeckt sind. So kann sich der/die Nutzer*in entscheiden, ob er/sie beispielsweise einen Temperatur-/Feuchtefühler nutzen möchte oder auf einen Sensor der sowohl Temperatur und Luftfeuchtigkeit als auch Luftqualität erfassen kann.Die hier vorgestellten Sensormodule umfassen ein breites Spektrum an möglichen Va-rianten, ist aber sicherlich nicht erschöpfend. Daher ist davon auszugehen, dass auch nach Abschluss der Diplomarbeit weitere Sensormodelle hinzukommen.

2.4.5.4.1 Bewegungserkennung

Zum Erkennen von Bewegungen gibt es aktive und passive Sensoren. Zu den aktiven Möglichkeiten gehören elektromagnetische Wellen (HF, Microwellen oder Dopplerradar) sowie Ultraschall. Als passive und weit verbreitete Variante, ist die Infrarotstrahlungserkennung zu nennen. Diese detektiert kleine Änderungen der Temperatur in der Umgebung, wenn sich zum Beispiel Personen oder Tiere im Erfassungsbereich bewegen. (21)

Es gibt in der Modulbauweise zwei unterschiedliche Baugrössen von Infrarotbewegungsmeldern. Die kleine Variante vom Typ HC-SR505 (Anhang 11) deckt einen Bereich einstellbar bis 3 Meter ab, die grössere, vom Typ HC-SR501 (Anhang 12), bis 7m. Aufgrund der Tatsache das die Sensorphalanx unter der Wohnungsdecke montiert werden soll, ist es sinnvoll die Variante mit einem Erfassungsbereich von 7m zu nehmen. Da in der Regel die Deckenhöhe einer Wohnung bei min 2 - 3 m liegt und so eine grosse Fläche innerhalb eines Raumes abgedeckt werden kann.

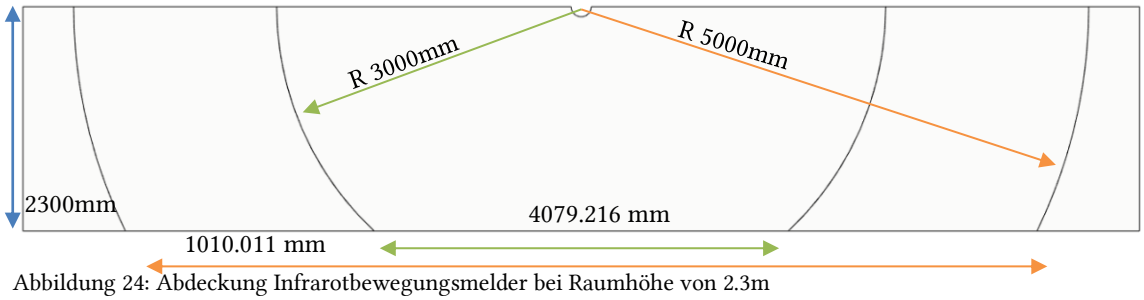


Abbildung 24: Abdeckung Infrarotbewegungsmelder bei Raumhöhe von 2.3m

Tabelle 20: Eigenschaften LHI778 und AS312

Sensor	Technische Daten		Wertung
Sensor Typ: HC-SR501	Stromaufnahme	50 µA	Positiv
	Erfassungsbereich	7 m	Positiv
	Sensorlinse Ø	23 mm	Negativ
Sensor Typ: HC-SR505	Stromaufnahme	< 1 mA	Negativ
	Erfassungsbereich	3 m	Negativ
	Sensorlinse Ø	10.5 mm	Positiv

2.4.5.4.2 Helligkeit

Als Sensor zur Erfassung der Helligkeit bietet sich ein Sensor (Anhang 13) an. Dieser Sensor liegt, da es sich um einen einfachen lichtsensitiven Widerstand handelt der günstig in grossen Mengen produziert werden kann, preislich im Rappenbereich. Der LDR Sensor benötigt lediglich einen Vorwiderstand und wird direkt an einem Analog-In der CPU angeschlossen. Er bietet aber nur die Möglichkeit einen Analogwert auszulesen und hat keine genaue Angabe der Helligkeit als Messwert (in LUX). Da aber zu erwarten

ist, dass die Helligkeit eher als Schwellwert im Home Assistant verarbeitet wird ist eine solcher Wert ausreichend.

Als zukünftige Varianten sind noch Module auf Basis der Bauteile BH1750 (Anhang 14) und TSL2591 (Anhang 15) möglich, beide Sensoren verwenden zur Kommunikation I²C. Für das Projekt wurde sich zunächst gegen die Varianten entschieden da der LDR alles benötigte bietet und deutlich günstiger zu bekommen ist als die Alternativen.

Tabelle 21: Entscheidungsübersicht Lichtsensor

Eigenschaften	LDR	BH1750 und TSL2591
Preis	Positiv	Negativ
Einheiten Ausgabe	Positiv	Positiv
Kompakter Einbau	Positiv	Negativ
Einfache Kommunikation	Positiv	Negativ

2.4.5.4.3 Ladezustandserkennung und Anzeige

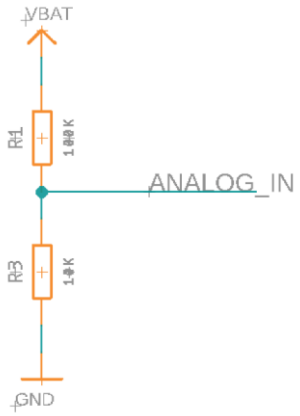


Abbildung 25: Schaltplan für die Spannungsmessung mit dem ESP32

Für die Erfassung des Akkuladezustands wird im Lochrasterprototyp auf einen 100 kOhm und einen 10 kOhm Widerstand zurückgegriffen. Diese werden parallel zum Akku platziert und am Verbindungspunkt der beiden Widerstände wird der Analog-In des EPS32 angeschlossen. Dadurch ist sichergestellt das die Spannung am Analog-In nicht über 3.3V steigen kann, da dieses den Eingang zerstören kann.

In der weiterentwickelten PCB-Variante der Sensoreinheit kann der Ladezustand direkt vom IP5306 Battery Management IC mittels I²C ausgelesen werden.

2.4.5.4.4 Luftqualität

Zur Luftqualitätsmessung steht unteranderem der Sensor vom Typ CCS811 (Anhang 16) zur Verfügung. Dieser Sensor ist in der Lage die Anzahl an flüchtigen, organischen Verbindungen (VOC) zu messen und das Kohlendioxidäquivalent (eCO2) zu berechnen. Zu beachten ist, dass der Sensor unter Umständen eine Burn-In Phase von 48 Stunden benötigt, sowie eine Aufwärmzeit von 20 Minuten nach längerem Aussetzen der Messungen empfohlen wird. Die Schnittstelle zur Kommunikation mit der CPU ist I²C. Über dieser werden je nach eingestelltem Modus eine Messung pro Sekunde, eine Messung alle 10 Sekunden, eine Messung alle 60 Sekunden oder eine Messung alle 250 Millisekunden gesendet.

Als Alternative wäre hier der Sensor vom Typ SGP30 (Anhang 17) zu nennen. Dieser unterscheidet sich im Wesentlichen darein, dass er keine Burn-In Phase benötigt und eine kürzere Aufwärmzeit benötigt. Ein zusätzlicher Vorteil ist, dass der Sensor im Gegensatz zum CCS811 kein I²C -Takt- Stretching benötigt.

Tabelle 22: Eigenschaften CCS811 und SGP30

Sensor	Eigenschaften	Wertung
CCS811	Fester einlese Zyklus	Negativ
	I²C -Takt- Stretching	Negativ
	Burn-In Phase & Aufwärmzeit nötig	Negativ
SGP30	Einlesen bei Bedarf sonst Sleep	Positiv
	Kein I²C -Takt- Stretching	Positiv
	Direkte Messungen möglich	Positiv

2.4.5.4.5 Temperatur

Zur Messung der Temperatur wird eine ST1147 (Anhang 18) der Firma IDUINO verwendet. Dieser basiert auf einem NTC-MF52 9350 (Anhang 19). Der NTC Widerstand weist einen negativen Temperaturkoeffizienten auf, der bei höheren Temperaturen den elektrischen Strom besser leitet als bei tiefen Temperaturen. Alternativ wären hier noch PTC Widerstände, genormte PT100 Temperaturfühler oder ein Modul auf Basis eines DS18B20 (Anhang 20). Diese Alternativen sind aber momentan nur schwer zu bekommen oder deutlich teurer als die oben genannte NTC Variante.

2.4.5.4.6 Kombinationssensor: Temperatur & Feuchtigkeit

Für den Abdeckung der Feuchtigkeit und Temperaturmessung sind die günstigen DHT11 (Anhang 21) und DHT22 (Anhang 22) Sensoren gut geeignet. Die DHT11 und DHT22 Sensoren haben allerdings einen signifikanten Nachteil. Die Abtastrate beim DHT11 beträgt 1 Hz und beim DH 22 0.5Hz. Das bedeutet, dass alle ein bzw. zwei Sekunden neue Daten an die CPU geschickt werden. Dies könnte sich negativ auf die Datenerfassung nach einem Aufwachen der CPU aus dem Deep Sleep auswirken. Das Verhalten wird in der Praxis zu überprüfen und gegebenenfalls mittels Software zu Lösen sein. Die Kommunikation der DHT Sensoren erfolgt mittels eines «eigenen» Datenbus.

Tabelle 23: Eigenschaften DHT11 und DHT22

Sensor	**Messwert**	**Arbeitsbereich**	**Genauigkeit**
DHT11	Luftfeuchtigkeit	20 – 100 %	±5.0 %
	Temperatur	0 – +50 °C	±2.0 °C
DHT22	Luftfeuchtigkeit	0 – 100 %	2 – 5 %
	Temperatur	-40 – +80 °C	±0.5 °C

Im Zuge des Projektes fällt die Entscheidung auf den DHT22 Sensor, da dieser eine deutlich höhere Genauigkeit hat.

Tabelle 24: Entscheidungsübersicht Feuchtigkeit/Temperatur Sensor

Eigenschaften	**DHT11**	**DHT22**
Genauigkeit	Negativ	Positiv
Preis	Negativ	Positiv
Abtastrate	Positiv	Negativ

2.4.5.4.7 Kombinationssensor: Temperatur & Luftdruck

Der Bosch BMP280 (Anhang 23) vereint die Messung von Luftdruck und Temperatur. Als Alternative gibt es noch den MPL115A2 Chip (Anhang 24). Die beiden Sensoren unterscheiden sich in ihren Eigenschaften nur in dem Punkt ihrer Messtoleranz. Der MPL115A2 hat eine Ungenauigkeit von ±1 kPa der BMP280 von ±0.012kPa. Zudem ist der MPL deutlich teurer im Vergleich zum BMP280. Beide Varianten verfügen über die Möglichkeit der Kommunikation mittels des I²C Protokolls, was den Verdrahtungs- und Programmieraufwand deutlich minimiert.

Tabelle 25: Entscheidungsübersicht Druck/Temperatur Sensor

Eigenschaften	Bosch BMP 280	MPL115A2
Genauigkeit	Positiv	Negativ
Preis	Positiv	Negativ
I²C Protokoll	Positiv	Positiv

2.4.5.4.8 Kombinationssensor: Temperatur, Feuchtigkeit & Luftdruck

Für die Messung der Temperatur, Feuchtigkeit und Luftdruck gibt es auch kombiniert Sensoren. Es konnten zwei Modelle gefunden werden BME280 (Anhang 25) und MS8607 (Anhang 26). Beide Module unterscheiden sich nur minimal. Zur Kommunikation wird in beiden Fällen I²C verwendet.

Leider waren im Verlauf dieser Diplomarbeit keine der beiden Sensortypen innerhalb nützlicher Frist verfügbar. Daher wurde auf diese Variante verzichtet.

2.4.5.4.9 Kombinationssensor: Temperatur, Feuchtigkeit, Luftdruck & Luftqualität

Es gibt auch Module, die vier Raumdaten in einem Modul vereint. Die Module basieren auf dem Bosch BME680 (Anhang 27) und vereinen die Möglichkeiten Temperatur, Feuchtigkeit, Luftqualität und Luftdruck zu messen und die Werte mittels I²C an die CPU zu übermitteln.

2.4.5.4.10 Varianten Übersicht

In der folgenden Tabelle ist eine Übersicht aller ausgewählten Sensortypen und ihren ermittelbaren Werten dargestellt.

Tabelle 26: Auflistung der ausgewählten Sensoren / Erkennungswert

Sensor / Zweck	LHI778	LDR	10 kΩ	SGP30	NTC	DHT22	BMP280	BME680
Bewegungs-erkennung	X							
Helligkeit		X						
Ladestand			X					
Luftqualität				X				X
Temperatur					X	X	X	X
Luft-feuchtigkeit						X		X
Luftdruck							X	X

Die obere Tabelle zeigt auch auf, dass die Sensoren LHI778, LDR, und die Ladestandsanzeige in jeder möglichen Konfiguration verwendet werden kann. Die anderen Sensoren können sich untereinander ersetzen oder gegebenenfalls ergänzen. Bei parallelem Einsatz können Sensorwerte also mehrfach vorkommen. Diese doppelten Sensordaten können gegebenenfalls zur Plausibilitätskontrolle genutzt werden.

2.4.6 **Zusammenfassung Hardwarebestellung**

Die ausgewählte Hardware ist für die Diplomarbeit zweckmässig. Sie kann durch andere Module im Nachgang weiter ausgebaut werden. Eine hohe Modularität ist hergestellt.

2.5 Softwarekonzept

Zur besseren Darstellung des angedachten Programmablaufs wird zunächst ein vereinfachter Programmablaufplan (PAP) erstellt. Dieser bietet eine Orientierung, welche Programmeschritte zunächst entwickelt werden müssen. Die Einteilung bietet auch die Möglichkeit, das gesamte Programm in einzelne Funktionen aufzusplitten. Diese können entsprechend umgesetzt und unabhängig voneinander getestet werden.

2.5.1 Programmablaufpläne

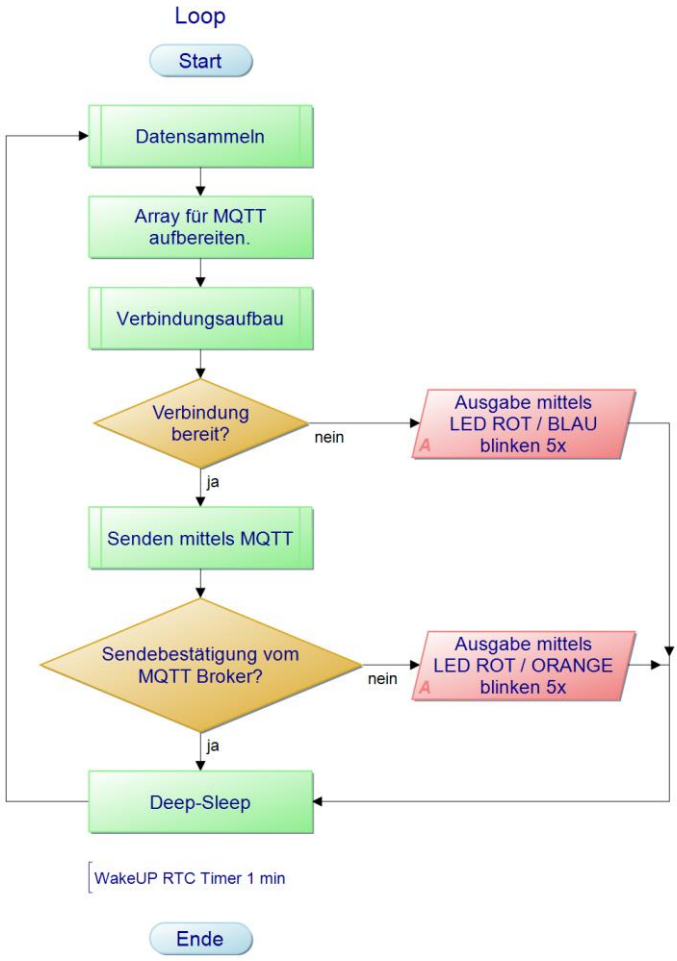


Abbildung 26: PAP Funktion Loop

Der Programablaufplan des «Loop» beschreibt den Ablauf des eigentlichen Programms. Das erste Mal wird er durch Start des ESP32 ausgelöst. Anders als in üblichen Programabläufen wird hier der Loop nicht mehrmals durchlaufen, sondern lediglich einfach nach dem Aufwachen abgearbeitet und endet wieder mit dem Start einer neuen Deep Sleep Periode.

Innerhalb eines Durchlaufes werden aus dem Loop heraus verschiedene Unterfunktionen aufgerufen die ihrerseits Abläufe beinhalten und im Folgenden kurz erklärt werden.

2.5.1.1 Datensammeln

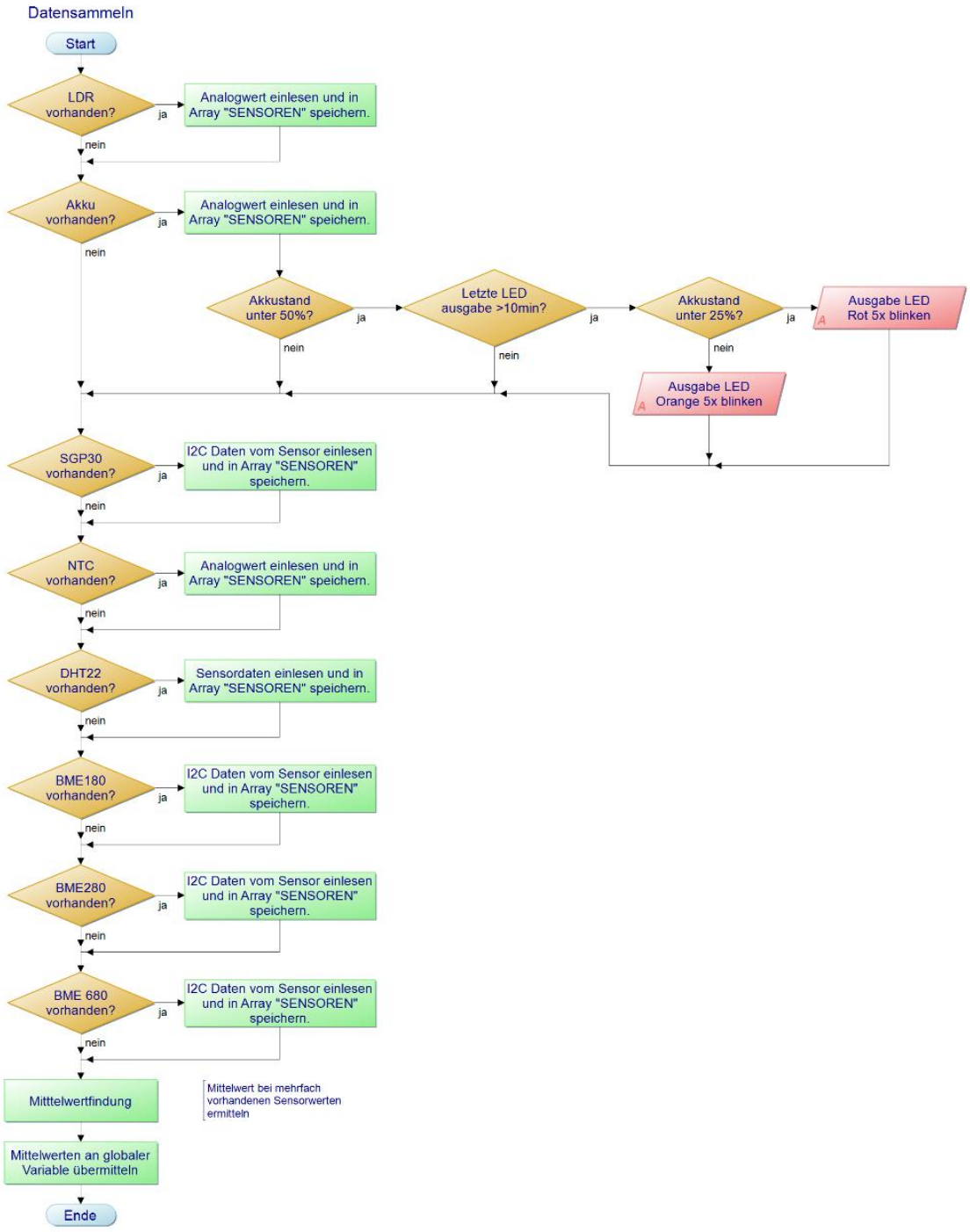


Abbildung 27: PAP Funktion Datensammeln

Die Funktion «Datensammeln» prüft zunächst ob laut Softwaredefinition der Sensor in der Programabfolge vorhanden ist, um dann die entsprechenden Informationen vom Sensor auszulesen und in einem Array zwischen zu speichern. Ein Sonderfall nimmt die Akkuladestandanzeige ein. Diese Abfrage steuert die LED an, um dem / der Nutzer*in ein visuelles Feedback zu geben. Nachdem alle vorhandenen Sensordaten ausgelesen wurden, wird bei doppelt vorhandenen Daten der Mittelwert gebildet.

2.5.1.2 Verbindungsaufbau

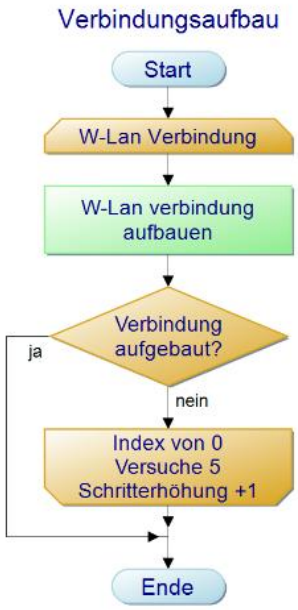


Abbildung 28: PSP Funktion Verbindungsaufbau

Mit der Funktion «Verbindungsaufbau» wird die WLAN-Verbindung zum Netzwerk hergestellt und so eine Kommunikation mit dem Home Assistenten ermöglicht. Da dieser Vorgang mitunter fehlschlagen kann, muss eine Fehlerabfrage realisiert werden, die nach einer bestimmten Anzahl von Fehlversuchen reagiert und die Sensoreinheit wieder in den Deep Sleep versetzt werden kann.

2.5.1.3 MQTT

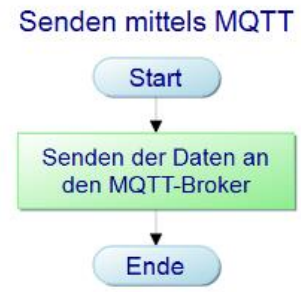


Abbildung 29: PAP Funktion Senden mittels MQTT

In der Funktion «MQTT» wird bei bestehender WLAN-Verbindung ein Kanal zum MQTT Broker hergestellt und die gesammelten Informationen an den MQTT Broker übermittelt. Gegebenenfalls wird auf eine Bestätigung des Empfangs der Daten gewartet.

2.5.1.4 Interrupt

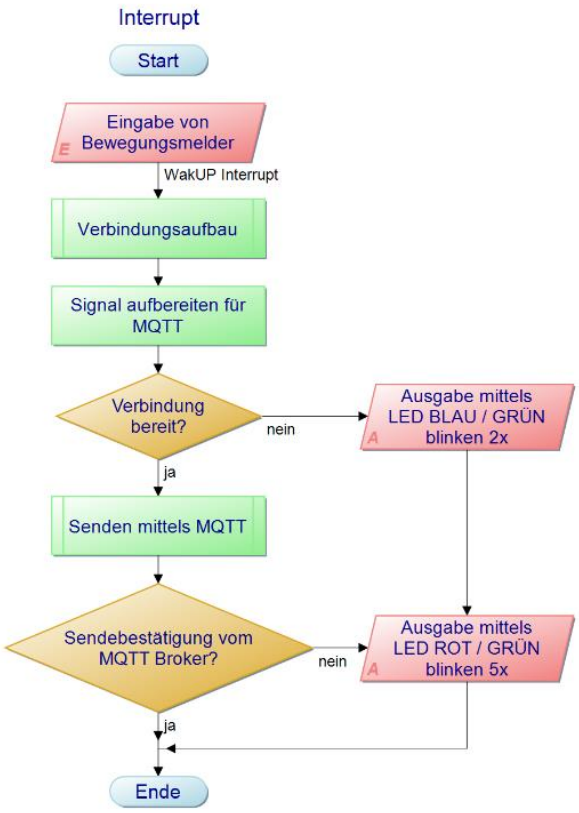


Abbildung 30: PAP Funktion Interrupt Bewegungsmelder

Ein Sonderfall ist das Interrupt, der den Deep Sleep Modus unabhängig vom Deep Sleep Timer unterbrechen kann. Dies ermöglicht das direkte Reagieren auf den Signaleingang, der vom Bewegungsmeder ausgelöst wird. Dadurch kann, direkt nach dem Aufwecken, der Home Assistant, darüber informiert werden, dass eine Bewegung registriert wurde bevor die Sensorphalanx wieder in den Deep Sleep Modus wechselt.

2.6 Abschluss Konzeptphase

Die umfassende Berücksichtigung von verschiedenen Faktoren wie Hard- und Softwareauswahl in der Konzeptphase, sowie das Ausarbeiten eines Softwarekonzeptes ermöglicht nun die Umsetzung eines ersten Prototypens der Sensorphalanx

3 Umsetzung

In folgendem Kapitel wird die Umsetzung des Projekts dokumentiert. Dies wird auch Erkenntnisse beinhalten, die im Zuge der Umsetzung gewonnen wurden und somit in den darauf aufbauenden Schritten berücksichtigt werden können.

3.1 Vorbereitung Raspberry Pi

Zunächst werden die im Reichelt Raspberry-Set enthaltenen Kühlkörper auf den Raspberry PI 3 B+ aufgeklebt. Dies ist zwar bei einem Modell 3 nicht zwingend notwendig, verbessert aber die Wärmeabfuhr. Das kann sich positiv auf die Langlebigkeit des Kleincomputers auswirken. Anschliessend wird das Board in das mitgelieferte Gehäuse eingebaut. Zur einfacheren Bedienung wird dann ein NexDock angeschlossen. Dieses ermöglicht es den Raspberry wie ein Notebook mit Mousepad, Tastatur, Bildschirm und Akkuversorgung zu bedienen.

Abbildung 31: Raspberry Pi 3 B+ in Reichelt-Set mit angeschlossenem NexDock

3.1.1 Home Assistant Installation

Die Installation von Home Assistant ist mit Hilfe des Programms Balena Etcher sehr einfach umzusetzen. Zunächst muss die im Raspberry Set enthalten Micro SD-Karte an ein bereits laufendes System, in diesem Fall ein Windows 10 Notebook mit SD-Kartenleser, angeschlossen werden. Nach dem Öffnen des Programmes kann direkt der Link (`https://github.com/home-assistant/operating-system/releases/download/6.6/haos_rpi3-64-6.6.img.xz`) zum Image, des passenden Raspberry Model, angegeben werden. Im nächsten Schritt wird dann die SD-Karte als Ziel ausgewählt und ein Klick auf «FLASH» leitet das Schreiben der Daten auf die SD-Karte ein. Nach Abschluss des Kopiervorgang werden die Daten noch vom Programm verifiziert. Im Anschluss ist die Micro SD-Karte bereit, um in den Raspberry Pi eingesteckt zu werden und die Einrichtung durch Starten des PIs zu beginnen. Damit der Raspberry PI im Zuge der Installation auch gleich die aktuellen Updates einholen kann, empfiehlt es sich beim ersten Start den Raspberry mittels Patchkabel an einen mit Internet ausgestatteten Netzwerkanschluss anzuschliessen.

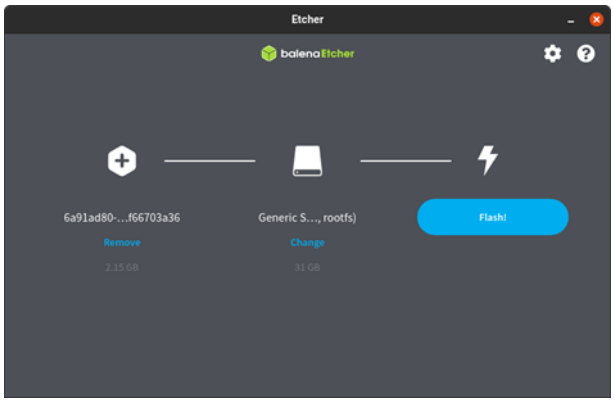


Abbildung 32: Balena Etche

3.1.2 Einrichten Home Assistant

Die grundsätzliche Einrichtung von Home Assistant läuft nach dem Start automatisch ab. Nach der ersten Initialisierung und dem Abschliessen des Startvorgangs ist Home Assistant im Netzwerk unter `homeassistant.local:8123` erreichbar und informiert bei dem ersten Besuch der Seite darüber, dass zunächst automatisch Updates installiert werden. Nachdem das Update durchgeführt wurden muss ein User Account angelegt werden. Danach kann der «Name» des Home-Assistant Server frei definiert und die ungefähre Position des Servers festgelegt werden.

3.1.2.1 Installation Mosquitto broker Addon

Abbildung 33: Mosquitto brok Einstellungen

Für die Kommunikation mit der Sensorphalanx muss zunächst noch der MQTT-Broker eingerichtet werden. Hierzu navigiert man nach der Anmeldung auf dem Webinterface (`homeassistant.local:8123`) in den Bereich «Einstellungen -> Add-ons, Backups & Supervisor» unten rechts auf den blaue Butten «ADD-ON STORE», sucht nach dem Mosquitto Broker und

installiert diesen mittels Klick auf den «INSTALL» Button. Im Anschluss sollten die Funktionen «Beim Booten starten» und «Watchdog» vom Mosquitto broker eingeschaltet werden, um einen reibungslosen Ablauf sicherzustellen.

Im oberen Teil befinden sich neben dem aktiven «Informationen» Reiter noch der Reiter «Dokumentation», diese beinhaltet alle wichtigen Informationen zum Einrichten des Mosquitto broker, «Konfiguration», dieser Reiter bietet die Möglichkeit den Mosquitto broker zu Parametrieren, und der «Protokoll» Reiter mit der Möglichkeit ein Protokoll des Mosquitto Brokers zu erhalten.

Im Reiter «Konfiguration» werden folgende Einstellungen im «Optionen» Fenster eingetragen:

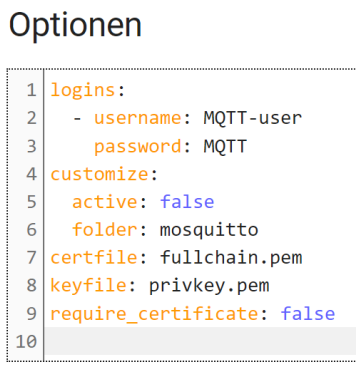


Abbildung 34: Einstellungen Mosquitto Broker

Als oberstes werden die Nutzerdaten für den MQTT Broker eingerichtet, dadurch kann kein Teilenemer ohne diese Informtion Daten an den Broker schicken. Alle weiteren Angaben sind orginal Einstellungen. Nähre Informationen dazu befinden sich im Reiter «Dokumentation». In der Kachel «Netzwerk» befinden sich die Portdefinitionen und ihre Bedeutung, diese können ebenfalls auf ihren Ursprungseinstellungen belassen werden.

Um sicher zu stellen, dass die Daten auch korrekt übermittelt werden kann im Bereich «Einstellungen > Geräte & Dienste» in der Kachel Mosquitto Broker die Funktion «Konfiguration» aufgerufen werden. Die Konfiguration besteht aus drei Kacheln und bietet in der Ersten, die Möglichkeit die MQTT Einstellungen zu bearbeiten. Hier muss beim Eintrag «Broker» die IP-Adresse des Raspberry Pis (192.168.0.200) eingetragen werden, zusätzlich werden die Benutzerdaten, die im oberen Teil definiert wurden, eingetragen. Danach ist es möglich selber MQTT Nachrichten zu verschicken. Nach einem Klick auf «Weiter» gibt es weitere MQTT-Optionen. Diese müssen aber nicht weiter beachtet werden.

Um zu schauen, ob der Broker arbeitet, kann nun im unteren Bereich ein Topic abonniert werden. Hierzu wird im Eingabefeld «Topic abonnieren» ein Topic eingetragen.

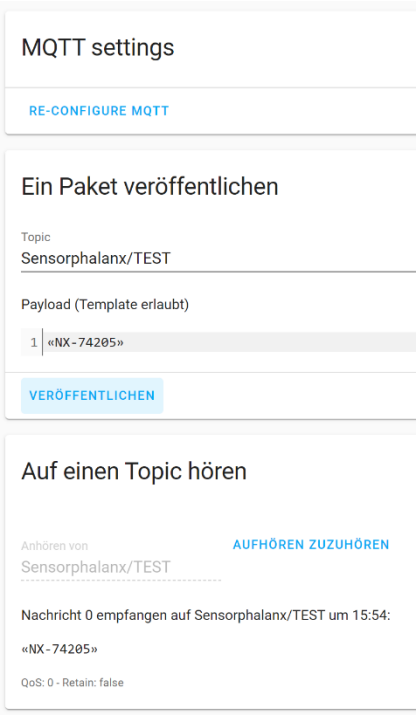


Abbildung 35: Beispiel des Tests des MQTT-brokers

Dies besteht beispielsweise aus einem Wort für den Sender und einem Namen eines Sensors, die durch einen Schrägstrich (/) getrennt werden z.B. «`Sensorphalanx/TEST`». Der hier als Beispiel verwendete Topic würde nach einem Klick auf «ANFANGEN ZUZUHÖREN» alle eingetroffenen Daten der Sensorphalanx des Sensors Test ausgeben. Alternativ könne auch durch Eingabe von «`Sensorphalanx/#`» alle Sensoren der Sensorphalanx angezeigt werden.

Um nun ein Datenpaket an für den Sensor «Test» an der Sensorphalanx zu erzeugen, kann man unter «Ein Paket veröffentlichen» ebenfalls den Topic «`Sensorphalanx/TEST`» eintragen. Im Anschluss definiert man noch den Payloade, hier kann ein beliebiger Text eingetragen werden, als Beispiel: «NX-74205» nach dem Absenden, durch Betätigen des «Veröffentlichen» Buttons, taucht die Nachricht unten auf. Das bestätigt das der MQTT-Broker arbeitet.

3.1.2.2 Installation File editor Addon

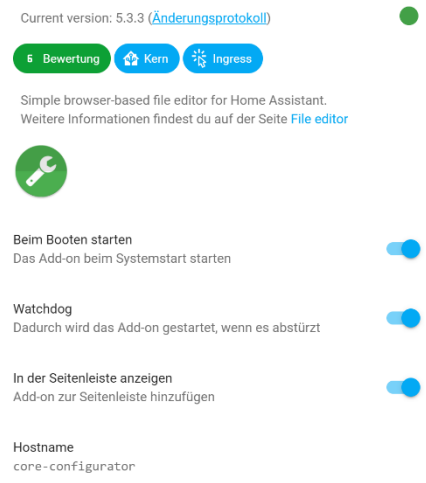


Abbildung 36: File Editor Einstellungen

Zusätzlich wird noch der «File editor» benötigt. Dieser erlaubt es Daten im Dashboard direkt zu editieren und wird benötigt, um die Sensoren einzurichten. Daher wird auch hier zunächst mittels «Einstellungen -> Add-ons, Backups & Supervisor» unten rechts auf den blauen Butten «ADD-ON STORE» navigiert und nach «File editor» gesucht und dieser installiert. Im Anschluss sollten die Funktionen «Beim Booten starten», «Watchdog» und «In der Seite anzeigen» eingeschaltet werden. Wenn der Editor nicht mehr benötigt wird, lässt es sich durch deaktivieren der Option «In der Seite anzeigen» Ausblenden, dadurch kann unbeabsichtigtes bearbeiten von Daten vermieden werden.

3.2 WLAN Access Point einrichten

Im nächsten Schritt wird ein Access Point eingerichtet um ein autonomes Netzwerk aufbauen zu können und für die später Präsentation ein unabhängiges und mobiles Netzwerk zur Verfügung zu haben.

Dazu wird sich mittels Laptops und Patchkabel mit dem Access Point verbunden ebenfalls wird der Raspberry mit einem Patchkabel an den Access Point angeschlossen. Nun meldet man sich mittels Webinterface am Access Point an und richtet zunächst einen DHCP Server, der die Verwaltung der im Netzwerk vorhandenen Teilnehmer mit Automatischer IP-Adresse übernimmt, ein.

Weiter wird dem Raspberry eine feste IP-Adresse zugewiesen, damit diese im weiteren Verlauf bei der Sensorphalanx hinterlegt werden kann, um eine Kommunikation mit dem System zu ermöglichen.

Im letzten Schritt wird das WLAN eingerichtet. Hierzu wird zunächst der WLAN-Name (SSID) mit letsbuildthings.ch definiert und Schlüssel vergeben.

3.3 Lochraster Prototyp

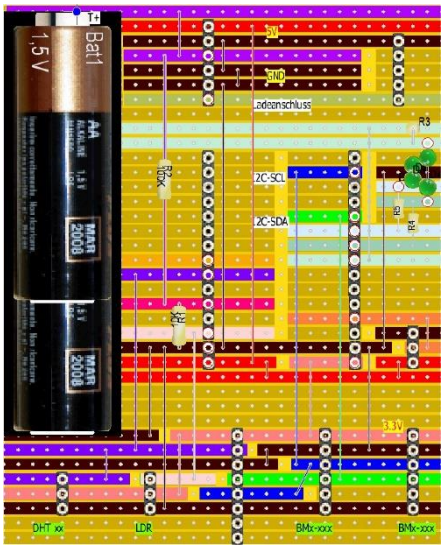


Abbildung 37: Lochrasterplatine

Um die ersten Versuche durchzuführen und das Programm für den ESP32 entwickeln zu können wird eine Lochrasterplatine aufgebaut. Dazu wird zunächst eine Zeichnung der Platine mittels «Loch Master 2» hergestellt (Anhang 28). Diese basiert auf den Datenblätter der verschiedenen Module und ermöglicht dessen Verbindung untereinander. Nach dem ersten Entwurf wird das Lochraster entsprechend der Zeichnung umgesetzt. Dabei muss darauf geachtet werden, dass die einzelnen Elemente genügend Platz haben.

Um Kurzschlüsse auszuschliessen, werden anschliessend alle Leiterbahnen durchgemessen und Kontaktfehler beseitigt. Nach Bestücken der Konnektoren mit den Modulen und dem Akku wird ein erster Test durchgeführt und mittels einer Powerbank der Akku vollgeladen. Nach Abziehen des Ladekabels hat sich ein Problem mit dem Laderegler gezeigt. Dieser schaltet nach 30 Sekunden ab, wenn nicht mindestens 50mA Laststrom vorhanden sind. Dieses Problem lässt sich zunächst durch Nutzung der Powerbank umgehen. Dies unterbindet das Abschalten. Als Möglichkeiten, das Problem in der PCB

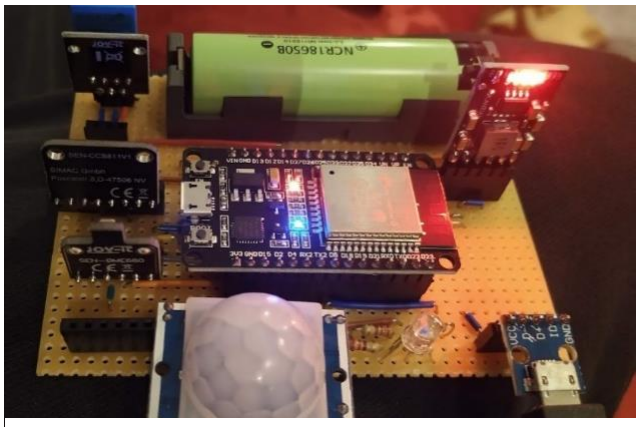


Abbildung 38: Lochraster bestückt

Version zu umgehen, wird ein anderer Regler verwendet. Zum einen gibt die Möglichkeit einen ähnlichen Regler zu verwenden der I²C unterstützt. Damit sollte es möglich sein mittels Befehles die Ausschaltfunktion abzuschalten. Zum anderen wäre als Alternative auch ein anderer Regler-Typ einsetzbar, der generell keine Abschaltung verwendet.

3.4 Software entwickeln

Zur Entwicklung der Software wird Visual Studio Code von Microsoft mit dem Addon PlattformIO verwendet. Mit dieser Erweiterung ist es möglich Arduino IDE kompatible Prozessoren zu programmieren. Bei der Verwendung von Visual Studio Code und PlatformIO ist darauf zu achten, dass die verschiedenen Funktionen vor dem Setup und Loop eingetragen sind, da sonst der Code nicht kompiliert werden kann.

Der gesamte verwendete Code ist in der Dateiablage dieser Arbeit (Anhang 29) zu finden. Im weiteren Verlauf der Dokumentation werden die Funktionsweisen der Programfunktionen näher beschreiben und mittels Auszüge aus dem Code einzelne besondere Abschnitte näher erläutert.

3.4.1 Funktionstest der Module mittels Software

Um die Sensoren zu testen, wurden zunächst die verschiedenen Bibliotheken der Module installiert und die zu den Bibliotheken gehörenden Beispiel-Programme auf den ESP32 übertragen. Im Zuge dieser Teste ist ein problematisches Verhalten mit den I²C Komponenten aufgetreten, dass sich wie folgt beschreiben lässt:

Wenn mehr als zwei I²C Komponenten am Bus angeschlossen sind, kommt es vor das einzelne Komponenten nicht zuverlässig erreicht werden können. Dies kann gut sichtbar gemacht werden durch Verwenden eines I²C-Scanners (22). Das Verhalten ist darauf zurück zu führen, dass jedes Modul mit fest verlötete Pull-up-Widerständen ausgestattet ist, dadurch wird mit jedem Modul ein weiterer Widerstand parallelgeschaltet, was dazu führt, dass bei drei Modulen der Gesamtwiderstand so absinkt das das Signal bei 3.3 V nicht mehr ausreichend stark ist, um alle Teilnehmer zu erreichen. Da dieses Problem mit den Modulen zusammenhängt und es bei den im Kapitel 2.4.5.4.10 aufgelisteten Modulen keine sinnvolle Variante mit mehr als zwei Modulen gibt, wird für dies Projekt die Anzahl, der gleichzeitigen Teilnehmer auf der I²C Schnittstellen, auf zwei Module beschränkt.

Sollte sich im Verlauf der Weiterentwicklung dieses Projektes abzeichnen das mehr Module benötigt werden, ist es möglich beim ESP32 eine zweite I²C Schnittstelle zu aktivieren oder alternativ bei den Modulen die Widerstände auszulöten.

3.4.2 Zusammenstellen der Bibliotheken

Während der Tests aus Kapitel 3.4.1 werden auch die für die Sensoren benötigten Bibliotheken zusammengetragen. Die Bibliotheken dienen zur einfacheren Programmierung und bieten die Möglichkeit mittels Funktionsaufruf direkt Abfragen an die Sensormodule zu schicken, ohne dass entsprechende Protokoll für den Senor aufbauen zu müssen. Bibliotheken haben aber auch einen Nachteil. Da sie in der Regel als Blackbox funktionieren, kann man nur schwer nachvollziehen was innerhalb der Bibliothek passiert. Für dieses Projekt werden aber nur offen einsehbare Bibliotheken verwendet, dadurch ist der Code einsehbar und gegebenenfalls auch anpassbar.

```
// Bibliotheken
#include <Arduino.h>
#include <Wire.h>
#include <WiFi.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <Battery18650Stats.h>
#include <Adafruit_BMP280.h>
#include <Adafruit_BME680.h>
#include <PubSubClient.h>
```

Abbildung 39: eingefügte Bibliotheken

Tabelle 27: Alle benötigten Bibliotheken

Bibliotheken	Zweck
Arduino.h	Beinhaltet alle Arduino-Variablendefinitionen und Arduino-Spezifischen Funktionen.
Wire.h	Für die Kommunikation mittels I²C
WiFi.h	Bibliothek zur WLAN-Kommunikation
Adafruit_Sensor.h	Vereint verschiedene Sensortypen und ermöglicht das Auslesen mittels eines Befehls
DHT.h	Beinhaltet die DHT Kommunikation
DHT_U.h	Wird für den DHT Sensor benötigt
Battery18650Stats.h	Für das Auslesen und Umrechnen des Ladezustand einer 18650 LI-ION Batterie
Adafruit_BMP280.h	Beinhaltet die BMP280 Kommunikation
Adafruit_BME680.h	Beinhaltet die BMP680 Kommunikation
PubSubClient.h	Beinhaltet die MQTT Kommunikation

3.4.3 Ablegen von Daten über den Deep Sleep hinaus

```
// Daten ablage in RTC Speicher
RTC_DATA_ATTR int64_t millis_DeepSleep;
RTC_DATA_ATTR bool BW_ON = 0;
RTC_DATA_ATTR bool BW_ST = 0;
```

Abbildung 40: Beispiel zum Speichern von Daten im Speicher der RTC

Da in diesem Projekt der Deep Sleep Modus des ESP32 genutzt wird, besteht Bedarf an der Möglichkeit Daten zu speichern, wenn der ESP im Deep Sleep Modus ist, dies ist mir üblichen Variablen nicht möglich. Wenn Variablen aber mittels RTC_DATA_ATTR definiert werden, werden die Informationen in einen kleinen Speicher der Real Time Clock des ESP32 verschoben und dort gespeichert, solange genügend Energie vorhanden ist. Das bedeutet die Daten gehen bei einem Spannungsverlust dennoch verloren.

3.4.4 Setup()

```
void setup()
{
  if (esp_sleep_get_wakeup_cause() != ESP_SLEEP_WAKEUP_EXT0)
  {
    gettimeofday(&tv_now, NULL);
    millis_DeepSleep = (int64_t)tv_now.tv_sec * 1000000L + (int64_t)tv_now.tv_usec;
  }
  Serial.begin(115200);
  Serial.println("  --  WakeUP  --  ");
  pinMode(LED_RO_PIN, OUTPUT);
  pinMode(LED_GR_PIN, OUTPUT);
  pinMode(LED_BL_PIN, OUTPUT);
  pinMode(IFR_BW_PIN, INPUT);
  digitalWrite(LED_GR_PIN, HIGH);
}
```

Abbildung 41: Auszug aus setup()

Im Setup wird als erstes kontrolliert, ob der ESP32 durch ein Interrupt des Bewegungsmelders aus dem Deep Sleep geweckt wird. In diesem Fall wird die aktuelle Zeit (Millisekunden nach letztem Neustart) aus der RTC (Real Time Clock) des ESP32 ausgelesen und zwischengespeichert. Weiter wird die Serielle Schnittstelle gestartet und der Pin-Mode für die LED und den Bewegungsmelder definiert, sowie die LED auf grün geschaltet um erkenntlich zu mache, dass die Sensorphalanx aufgewacht ist.

3.4.5 loop()

```
void loop()
{
  switch (esp_sleep_get_wakeup_cause())
  {
  case ESP_SLEEP_WAKEUP_EXT0:
    if (BW_ST == false)
    {
      doMoveAction();
        }
    else
    {
      BW_ON = true;
    }
    break;
  case ESP_SLEEP_WAKEUP_TIMER:
    doAction();
    break;
  default:;
    doAction();
    break;
  }
  StartWlan();
  MQTT();
  startDeepSleep();
}
```

Abbildung 42: Darstellung loop()

In der Loop-Funktion wird mittels eines Switchs abgefragt aus welchem Grund der ESP32 aus dem Deep Sleep geweckt wurde und anhand des Ergebnisses entsprechende Funktionen aufgerufen. So ist es bei einer Aktivierung durch den Bewegungsmelder nicht nötig alle Sensordaten zu sammeln, sondern lediglich dem Home Assistant mitzuteilen, dass dieser ausgelöst wurde.

Es werden aber immer die Funktionen «StartWlan», «MQTT» und «startDeepSleep» aufgerufen. Aufgrund dessen, dass der Deep Sleep immer am Ende des Loops gestartet wird, ist es nicht möglich, dass der Loop mehrmals durchläuft.

3.4.6 Daten Sammeln – doAction()

doAction() ist die aufwändigste Funktion. Ihre Aufgabe ist es alle Sensordaten einzusammeln und in ein Array zu schreiben. Aus diesen Daten werden dann die Mittelwerte aller gleichen Einheiten gebildet und in eine globale Variable abgelegt. Um dies zu ermöglichen wird am Anfang des Programms definiert welche Sensoren und welche Anschlüsse verwendet werden. Dies muss vom/von der Maker*in, vor dem Kompilieren des Programms definiert werden Auf diese Angaben bezieht sich dann die Funktion doAction().

```
// Sensorenauswahl
bool HC_SR50X = true;   // Infrarotbewegungsmelder
bool LDR = true;          // LDR Sensor vorhanden
bool SGP30 = true;        // SGP30 Sensor vorhanden
bool NTC = false;         // NTC Sensor vorhanden
bool BMP280 = false;      // BMP280 Sensor vorhanden
bool BME680 = true;       // BME680 Sensor vorhanden
bool Battery18650 = true; // Battery18650 Sensor vorhanden
bool DHT_11 = false;      // DHT Sensor vorhanden
bool DHT_22 = true;       // DHT Sensor vorhanden
#define DHTTYPE DHT22     // DHT Typ 11 oder 22

// Pindefinition
#define LED_RO_PIN 5  // LED Pin Rot
#define LED_GR_PIN 19 // LED Pin Grün
#define LED_BL_PIN 18 // LED Pin Blau
#define DHT_PIN 26    // DHT_IN Pin
#define LDR_PIN 13    // LDR Pin
#define NTC_PIN 13    // NTC Pin
#define AKKU_PIN 25   // AKKU Pin
#define IFR_BW_PIN 2  // IFR_In Pin / GPIO_NUM_"n" in startDeepSleep() anpassen
```

Abbildung 43: Sensor Bestimmung im Programm

Zunächst wird in der Funktion die Arraygrösse für die Sensorwerte festgelegt, dies passiert dynamisch auf Basis der ausgewählten Sensoren und deren definierten

Messeinheiten. Mithilfe der Funktion Array_size(). Nach Erstellen des Sensorarrays werden diese noch mit 0.00 bei Float- bzw. 0 bei Integerarrays initialisiert.

Im Anschluss erfolgt das Auslesen der Sensordaten, vorausgesetzt sie wurden am Anfang definiert. Auf Grund der Eigenheiten der Sensoren benötigt jeder einen eigenen Befehl. Im Folgenden wird anhand von vier unterschiedlichen Sensoren beschrieben, wie die Sensordaten gesammelt und verarbeitet werden.

3.4.6.1 NTC am Analog Eingang

```
// NTC Sensor Enlesen
if (NTC == true)
{

  int Vo = 0;
  float R1 = 10000;
  float logR2, R2, T, Tc;
  float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;
  for (int i = 0; i <= 100; i++)
  { // 100 Messungen zur Stabeliesierung des Messwertes
    Vo = Vo + analogRead(NTC_PIN);
  }
  Vo = Vo / 100;                        // Mittelwert aus 100 Messungen erzeugen
  R2 = R1 * (4095.0 / (float)Vo - 1.0);
  logR2 = log(R2);
  T = (1.0 / (c1 + c2 * logR2 + c3 * logR2 * logR2 * logR2));
  Tc = T - 273.15;

  int i = 0;
  while (arry_Temp[i] != 0.00)
  {
    i++;
  }
  arry_Temp[i] = Tc;
}
```

Abbildung 44: Auszug doAction() NTC Daten auslesen

Zunächst wird mittels `analogRead(NTC_PIN)` der Analogwert 100 mal eingelesen und die entstandene Summe durch 100 geteilt um den Mittelwert zu erhalten und so einen stabilen Messwert zu bekommen. Das Ergebnis wird dann logarithmisch umgerechnet, um den auf der NTC Widerstandskurve angepasst Temperaturwert zu errechnen, dieser Wert wird dann von Gradfahrenheit in Gradcelsius und in den nächsten freien Platz im Temperatur Array geschrieben.

3.4.6.2 DHT mit eigenem Kommunikationsprotokoll

```
// DHT Sensor Enlesen
if (DHT_11 || DHT_22 == true)
{
  DHT_Unified dht(DHT_PIN, DHTTYPE);
  dht.begin();
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (isnan(event.temperature))
  {
    Serial.println(F("Fehler beim einlesen der Temperatur!"));
  }
  else
  {
    int i = 0;
    while (arry_Temp[i] != 0.00)
    {
      i++;
    }
    arry_Temp[i] = event.temperature;
  }
  dht.humidity().getEvent(&event);
  if (isnan(event.relative_humidity))
  {
    Serial.println(F("Fehler beim einlesen der Luftfeuchtigkeit!"));
  }
  else
  {
    int i = 0;
    while (arry_Humidity[i] != 0.00)
    {
      i++;
    }
    arry_Humidity[i] = event.relative_humidity;
  }
}
```

Abbildung 45: Auszug aus doAction() für DHT Sensorn

Beim DHT Sensor wird zunächst definiert, ob es sich um ein DHT11 oder DHT22 Sensor handelt und im Anschluss wird die Verbindung zum Sensor aufgebaut, mittels `dht.temperature().getEvent(&event)` die Temperatur in Gradcelsius direkt ausgelesen und in den nächsten freien Platz im Temperatur Array abgelegt

Mit dem Befehl `dht.humidity().getEvent(&event)` wird direkt die relative Luftfeuchtigkeit ausgelesen und in den nächsten freien Platz im Array für die Luftfeuchtigkeit abgelegt.

3.4.6.3 BME680 mit I²C Kommunikation

```
// Setup oversampling und filter
bme.setTemperatureOversampling(BME680_OS_8X);
bme.setHumidityOversampling(BME680_OS_2X);
bme.setPressureOversampling(BME680_OS_4X);
bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
bme.setGasHeater(320, 150); // 320*C für 150 ms
```

Abbildung 46: Auszug aus doAction() BME680 Setup

Beim BME680 Sensormodul muss zunächst die Überabtastung (eng. Oversampling) eingestellt und aktiviert werden. Zusätzlich wird für den Gas-Sensor noch die Heiz-/Temperatur und Zeit eingestellt dies wird zur Ermittlung der Gas-Werte benötigt. Im Anschluss werden die Werte für Temperatur, Luftdruck, Luftfeuchtigkeit und Gaskonzentration direkt ausgelesen und in die dafür vorgesehenen Arrays geschrieben.

```
// Umrechnen der gaskonzentration zum IAQ index
float gas_lower_limit = 5000;  // Limit schlechte Luftquallität
float gas_upper_limit = 50000; // Limit gute Luftquallität
if (gas_reference > gas_upper_limit)
  gas_reference = gas_upper_limit;
if (gas_reference < gas_lower_limit)
  gas_reference = gas_lower_limit;

gas_score = (0.75 / (gas_upper_limit - gas_lower_limit) * gas_reference
- (gas_lower_limit * (0.75 / (gas_upper_limit - gas_lower_limit)))) * 100;
float airQuality = hum_score + gas_score;

if ((getgasreference_count++) % 10 == 0)
{
  int readings = 5;
  for (int i = 1; i <= readings; i++)
  {
    gas_reference += bme.readGas();
  }
  gas_reference = gas_reference / readings;
}

i = 0;
while (arry_Air_Quality_Score[i] != 0.00)
{
  i++;
}
arry_Air_Quality_Score[i] = (100 - airQuality) * 5;

String IAQ_text = "";
if (airQuality >= 301)
  IAQ_text += "Gefährlich";
else if (airQuality >= 201 && airQuality <= 300)
  IAQ_text += "Sehr ungesund";
else if (airQuality >= 176 && airQuality <= 200)
  IAQ_text += "Ungesund";
else if (airQuality >= 151 && airQuality <= 175)
  IAQ_text += "Ungesund für sensible Gruppen";
else if (airQuality >= 51 && airQuality <= 150)
  IAQ_text += "Mässig";
else if (airQuality >= 00 && airQuality <= 50)
  IAQ_text += "Gut";

Serial.print("Luftquallität = ");
Serial.println(IAQ_text);
```

Abbildung 47: Auszug aus doAction() BME680

Aus den Daten wird eine Berechnung der Luftqualität nach IAQ vorgenommen, diese Berechnung ist nötig, da die BME680 Bibliothek von Adafruit nicht in der Lage ist diese zu berechnen. Die Berechnung basiert auf einem Tutorial von Stefan Draeger (23) es hat sich aber nach der Inbetriebnahme des Sensors GSP30 herausgestellt, dass das Ergebnis nicht mit der Messung des eCO2 des GSP30 korrelieren, diese sollten sich aber ähnlich verhalten. Leider ist der GSP30 Sensor aber erst kurz vor Ende der Abgabefrist der Diplomarbeit eingetroffen, daher war es nicht mehr möglich diesem Problem nachzugehen. Für die Berechnung werden zunächst die Grenzwerte für Luftqualität festgelegt und entsprechend die Gasreferenzwerte bei Überschreiten überschrieben. Im Anschluss wird eine Gas Wert berechnet und mit einem weiter oben berechneten Humen Score addiert. Das ergibt nach einer Umrechnung einen Luft-Quallitätswert, der in das dafür vorgesehene Array gespeichert wird und in einem, bei angeschlossenem ESP32 lesbaren, Konsolentext umgewandelt und angezeigt.

3.4.6.4 GSP30 mit I²C Kommunikation

```
if (SGP30 == true && battery.getBatteryChargeLevel() > 50)
{
  if (!sgp.begin())
  {
    Serial.println("Sensor not found :(");
    while (1)
      ;
  }
  // sgp.setIAQBaseline(0x7FC7, 0x82A5);
  float sgp2temperature, sgp2humidity;
  if (DHT_11 || DHT_22 == true)
  {
    DHT_Unified dht(DHT_PIN, DHTTYPE);
    dht.begin();
    sensors_event_t event;
    // wenn ein DHT Sensor vorhanden ist kann dieser genutzt werden um
    dht.temperature().getEvent(&event);
    if (isnan(event.temperature))
    {
      sgp2temperature = 22.1; // [°C]
    }
    else
    {
      sgp2temperature = event.temperature;
    }
    dht.humidity().getEvent(&event);
    if (isnan(event.relative_humidity))
    {
      sgp2humidity = 45.2; // [%RH]
    }
    else
    {
      sgp2humidity = event.relative_humidity;
    }
  }
  else
  {
    sgp2temperature = 22.1; // [°C]
    sgp2humidity = 45.2;    // [%RH]
  }
  Serial.print("setHumidity() ");
  Serial.println(getAbsoluteHumidity(sgp2temperature, sgp2humidity));

  sgp.setHumidity(getAbsoluteHumidity(sgp2temperature, sgp2humidity));
  int i = 0;
  while (arry_TVOC[i] != 0)
  {
    i++;
  }
  for (int x = 0; x <= 30; x++)
  {
    if (!sgp.IAQmeasure())
    {
      Serial.println("IAQ Mesung fehlgeschlagen");
      return;
    }
    delay(1000);
  }
  arry_TVOC[i] = sgp.TVOC;
```

Abbildung 48: Auszug doAction() SGP30

Der GSP30 benötigt fast 30 Sekunden Anlaufzeit, um brauchbare Werte zu liefern, in dieser Zeit muss die Messung immer wieder angestossen werden. Da der Energieverbrauch laut Datenblatt in dieser Zeit 48.8 mA beträgt wird die Verwendung dieses Sensors noch abhängig vom Ladezustand des Akkus abhängig gemacht. Dadurch wird ab einem Ladezustand der kleiner als 50% ist die Messung nicht mehr vorgenommen. Weiter wird beim Wechsel in den Deep Sleep Modus bei Verwendung des Sensors die Zeit bis zum Erwachen von 30 auf 60 Sekunden angehoben. Dies ist nötig, da sonst eine permanente Messung statt findet und eine Verkürzung der Zeit, bei nichtgebrauch sicherstellt das der Spannungsregler von ersten Prototypen nicht abschaltet.

```
if (sizeof(arry_Temp) != 0)
{
  for (n = 0; n != (sizeof(arry_Temp) / sizeof(int)); n++)
  {
    Temperature += arry_Temp[n];
  }
  Temperature = Temperature / (sizeof(arry_Temp) / sizeof(int));
}
```

Abbildung 49: Auszug doAction() Mittelwert Findung

Im weiteren Verlauf der Funktion werden die Mittelwerte aus den gesammelten Daten errechnet und in Globale Variablen abgelegt und zur Kontrolle auf der Konsole angezeigt.

3.4.7 WLAN Verbindungsaufbau - StartWlan()

Die Netzwerkverbindung wird in der Funktion `StartWlan()` aufgebaut dies geschieht erst nach dem sammeln der Daten um eine unnötig lang aufgebaute Verbindung zu vermeiden, da dies mit einem hohem Energiebedarf verbunden ist. Für einen Verbindungsaufbau müssen am Anfang des Programms zunächst die WLAN Daten sowie die IP-Adresse der Sensorphalanx als auch des Gateways hinterlegt werden. Es ist grundsätzlich auch möglich die Sensorphalanx mit einer Automatischen IP-Adresse auszustatten dies hat aber den Nachteil das der Einwahlprozess länger dauert und dadurch mehr Energie benötigt.

```
// WLAN-Daten
const char *ssid = "letsbuildthings.ch";
const char *password = "XXXXXXXXXXX";

// IP Adresse der Sensorphalanx
IPAddress local_IP(192, 168, 0, 199);
// Gateway IP Adresse
IPAddress gateway(192, 168, 0, 1);
IPAddress subnet(255, 255, 255, 0);
```

Abbildung 50: W-LAN und Netzwerkdaten

```
while (WiFi.status() != WL_CONNECTED && n != 5)
{
  digitalWrite(LED_GR_PIN, LOW);
  delay(400);
  Serial.print("Verbindungsversuch...");
  digitalWrite(LED_GR_PIN, HIGH);
  delay(400);
  n++;
}

if (WiFi.status() == WL_CONNECTED)
{
  digitalWrite(LED_GR_PIN, LOW);
  Serial.println("");
  Serial.println("WiFi Verbunden.");

  // Berechnung der Verbindungsqualität
  long rssi = WiFi.RSSI();
  rssi = -rssi;

  if (rssi < 27) // Berechunung W-LAN Qualli···
  else if (rssi >= 27 && rssi < 33)···
  else if (rssi >= 33 && rssi < 36)···
  else if (rssi >= 36 && rssi < 40)···
  else if (rssi >= 40 && rssi < 80)···
  else if (rssi >= 80 && rssi < 90)···
  else if (rssi >= 90 && rssi < 99)···
  else···

  Serial.print("WIFI-Qualli: ");
  Serial.println(WiFiperct);
```

Abbildung 51: Auszug aus StartWlan()

In der Funktion `StartWlan()` wird versucht eine WLAN Verbindung zum definierten SSID aufzubauen,. Dies wird signalisiert durch dauerhaftes Leuchten der Blauen LED und das Blinken der Grünen LED bei jedem Verbindungsaufbau. Wenn nicht innerhalb von 5 Versuchen eine Verbindung aufgebaut werden konnte wird die Funktion beendet, dies wird angezeigt durch das Ansteuern der Roten LED für 1 Sekunde.

Bei erfolgreichem Verbindungsaufbau bleibt die Blaue LED angesteuert und es wird die Qualität der WLAN-Verbindung festgestellt um diese im späteren Verlauf an den Home Assistant zu übermittelt.

3.4.8 Datenübertragung mittels MQTT – MQTT()

```
void MQTT()
{
  WiFiClient espClient;
  PubSubClient client(espClient);
  client.setServer(mqtt_server, 1883);
  while (!client.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // verbindungs versuch
    if (client.connect(MQTT_Name, MQTT_User, MQTT_Pass))
    {
      Serial.println("connected");
      char tempString[8];
      if (esp_sleep_get_wakeup_cause() != ESP_SLEEP_WAKEUP_EXT0)
      {
        dtostrf(Temperature, 1, 2, tempString);
        client.publish("Sensorphalanx/temperature", tempString);
        Serial.print("Sensorphalanx/temperature: ");
        Serial.println(tempString);
        dtostrf(Brightness, 1, 2, tempString);
        client.publish("Sensorphalanx/brightness", tempString);
        Serial.print("Sensorphalanx/brightness: ");
        Serial.println(tempString);
        dtostrf(Pressure, 1, 2, tempString);
        client.publish("Sensorphalanx/Pressure", tempString);
        Serial.print("Sensorphalanx/Pressure: ");
```

Abbildung 52: Auszug aus MQTT()

Um den Datenaustausch mit der MQTT Funktion zu ermöglichen, müssen zunächst die für die Funktion wichtigen Parameter eingegeben werden, diese sind die IP Adresse vom MQTT-Broker `mqtt_server(192, 168, 0, 200)` sowie die MQTT Nutzer*innen-Daten zum Anmelden `MQTT_User = "MQTT-user"` & `MQTT_Pass = "MQTT".` Die Funktion sendet dann die Daten als String mit dem Topic des Sensornamens und der Zuordnung an den MQTT Brocker.

3.4.9 Deep Sleep Mode – startDeepSleep()

```
void startDeepSleep()
{
  if (SGP30 == true && Charge_LVL_TAB > 50)
  {
    TIME_TO_SLEEP = TIME_TO_SLEEP_AND_SGP30;
      }
  switch (esp_sleep_get_wakeup_cause())
  {
  case ESP_SLEEP_WAKEUP_EXT0:
    gettimeofday(&tv_now, NULL);
    time_us = (int64_t)tv_now.tv_sec * 1000000L + (int64_t)tv_now.tv_usec;
    esp_sleep_enable_timer_wakeup((TIME_TO_SLEEP * 1000000) - (time_us - millis_DeepS
    break;
  case ESP_SLEEP_WAKEUP_TIMER:
    gettimeofday(&tv_now, NULL);
    time_us = (int64_t)tv_now.tv_sec * 1000000L + (int64_t)tv_now.tv_usec;
    esp_sleep_enable_timer_wakeup((TIME_TO_SLEEP * 1000000) - (time_us - millis_DeepS
    if (LHI778 == true)
    {
      esp_sleep_enable_ext0_wakeup(GPIO_NUM_2, HIGH);
    }
    break;
  default:
    gettimeofday(&tv_now, NULL);
    time_us = (int64_t)tv_now.tv_sec * 1000000L + (int64_t)tv_now.tv_usec;
    esp_sleep_enable_timer_wakeup((TIME_TO_SLEEP * 1000000) - (time_us - millis_DeepS
    if (LHI778 == true)
    {
      esp_sleep_enable_ext0_wakeup(GPIO_NUM_2, HIGH);
    }
    break;
    }
  digitalWrite(LED_RO_PIN, LOW);
  digitalWrite(LED_GR_PIN, LOW);
  digitalWrite(LED_BL_PIN, LOW);
  Serial.println("  --  DeepSeep  --  ");
  esp_deep_sleep_start();
}
```

Abbildung 53: Auszug aus startDeepSleep()

Die letzte Funktion ist das Versetzen des ESP32 in den Tiefschlaf. Dazu wird zunächst, wie im Kapitel 3.4.5.4 berichtet festgelegt, ob die Sleeptime 30 oder 60 Sekunden beträgt. Anschliessend wird mithilfe eines Softwareschaltester entschieden unter welcher Bedingung die CPU zuletzt aufgeweckt wurde. Bei einer Aktivierung durch den Bewegungsmelder ist die Funktion `esp_sleep_enable_ext0_wakeup()` nicht nötig, da ein erneutes aktivieren durch den Bewegungsmelder erst nach einem normalen Auslesezyklus gebraucht wird. Zusätzlich wird noch berechnet wie lange der ESP geschlafen hat bis er durch den Bewegungsmelder geweckt wurde. Dies ermöglicht, das versetzen in den DeepSleep für den Rest der Intervallzeit.

3.5 Datenauswertung & Aufbereiten auf dem Raspberry

Im folgenden Kapitel werden die Anpassungen dokumentiert, die nötig sind, um die dem MQTT-Broker übermittelten Daten zu visualisieren und zu speichern. Dies setzt die Installation der Addons «MQTT Brokers» und «File Editor» aus Kapitell 3.1.2 voraus.

Um sicher zu stellen, dass die Daten auch korrekt übermittelt werden, kann im Bereich «Einstellungen > Geräte & Dienste» in der Kachel «Mosquitto broker» die Funktion «Konfiguration» erneut aufgerufen werden. Im Bereich «Auf ein Topic hören» wird «`Sensorphalanx/#` als Topic eintragen und aktiviert, nun sehen wir alle Nachrichten, die nach dem Absenden von der Sensorphalanx eintreffen.

3.5.1 Visualisierung Vorbereitung

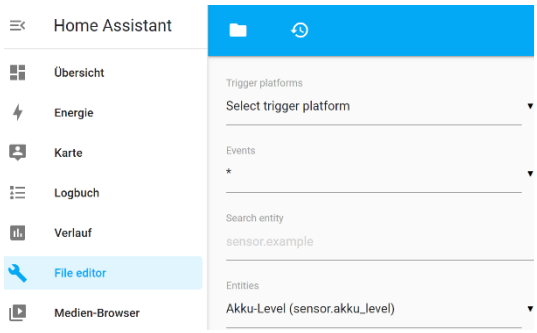


Abbildung 54: Auswahl configuration.yaml

Zur Visualisierung müssen die mittels MQTT-broker verfügbaren Werte zunächst klassifiziert werden. Erst dann weiss Home Assistant wie es die Daten zu behandeln hat. Hierzu wird im linken Menu der «File editor» geöffnet und das «Ordner» Symbol oben links aktiviert. In der sich einschiebenden Dateiübersicht wird die Datei `configuration.yaml` gesucht und geöffnet.

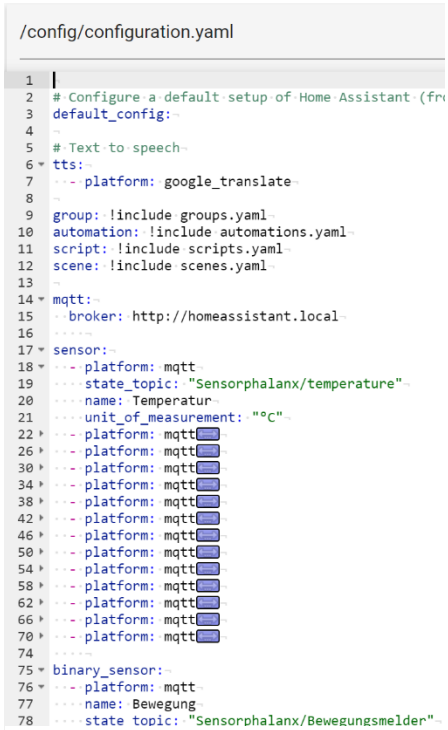


Abbildung 55: Anpassungen configuration.yaml

In dem Eingabefeld müssen verschiedene Einträge getätigt werden. Zunächst wird die Adresse des MQTT-brokers eingegeben, dass ist in der Abbildung 56 in der Zeile 14 und 15 ersichtlich. Daran anschliessend werden die Sensoren definiert, hier wird unterschieben zwischen binären Sensoren die nur ON und OFF unterscheiden und Sensoren mit grösserem Wertebereicht. Diese werden wie in Zeile 17 kenntlich gemacht. Darunter wird die Plattform mit «`mqtt`» festgelegt. In der nächsten Zeile wird der Topic der abgehört und definiert. Hier ist es wichtig exakt die gleiche Schreibweise zu verwenden, die auch in der MQTT Funktion der Sensoreinheit verwendet wird. Mit «`name:`» wird der angezeigte Name definiert und mittels «`unit_of_Masurement:`» die Einheit. Dieser Eintrag ist wichtig, wenn die Werte auch als Grafik

angezeigt werden sollen. Zu bedenken ist, das alle gleich definierten Einheiten im gleichen Grafen angezeigt werden. Bei Grössen die keine Einheit haben, wie zum Beispiel der LDR Sensor, kann der Bereich zwischen den Anführungszeichen leer bleiben.

Bei den binären Sensoren sind die Grundeinstellungen wie `platform`, `name` und `state_topic` identisch, zusätzlich muss noch die Nachricht, die erwartet wird mittels `payload_on` und `payload:off` entsprechend eingestellt werden. Nach Abschluss der Änderungen und wenn die Eingaben korrekt sind erfolgt automatisch eine Prüfung, ob der Code gelesen werden kann. Dies wird angezeigt durch einen grünen Haken oben rechts, bei einem roten Kreuz, kann durch Klicken auf das Symbol angezeigt werden welche Eingaben nicht korrekt sind. Wenn es plausible Änderungen am Dokument gibt, wird oben ebenfalls eine «Diskette» angezeigt, die bei Aktivierung das Speichern der Änderungen auslöst. Nach Änderungen an der `configuration.yaml` muss Home Assisitant zwingend neu gestartet werden, dies ist möglich durch öffnen des Menus hinter dem Zahnradsymbols und Auswahl der «Restart HASS» Option. Nach Abschluss des Neustarts stehen die Sensordaten zur Verfügung.

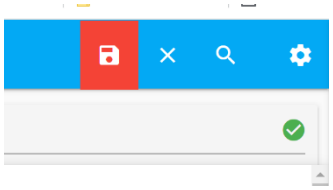


Abbildung 56: Darstellung der Korrekten Eingabe von Anpassungen

3.5.2 Einrichten der Übersicht

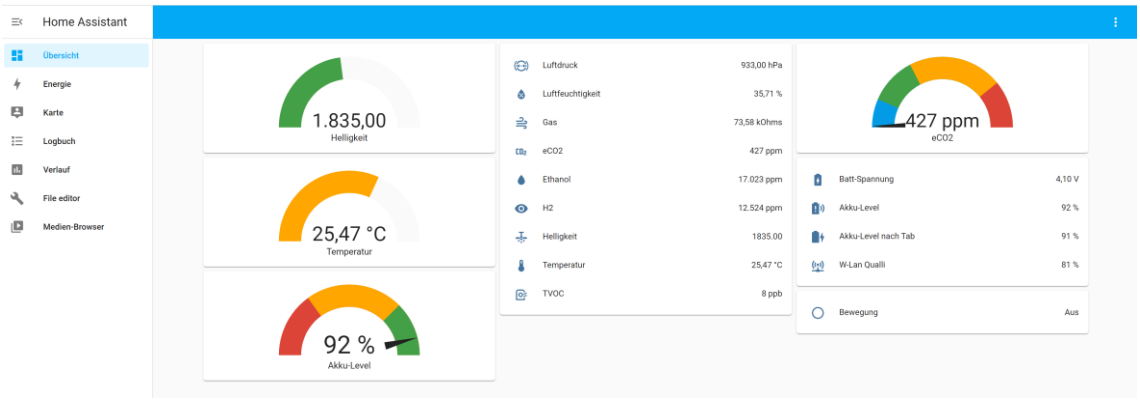


Abbildung 57: eingerichtete Ansicht aller möglichen Sensordaten vom Lochrasterprototypen

Um eine live Übersicht (wie in Abbildung 58) über alle Sensorwerte zu bekommen muss diese zunächst eingerichtet werden, hierzu geht man in die «Übersichte» Ansicht und klickt auf die Senkrecht angeordneten Punkte (⋮) hier erscheint nun ein Menu indem man die «Oberfläche konfigurieren» auswählt.

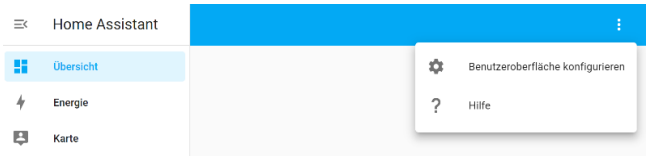


Abbildung 58:Auswahl «Oberfläche konfigurieren»

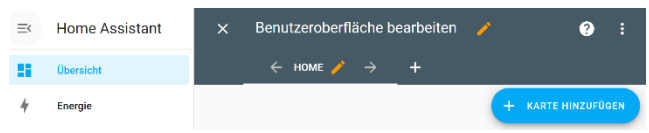


Abbildung 59: Ansicht Bearbeitungsmodus Übersicht

Nachdem die Ansicht in den Bearbeitungsmodus gewechselt hat, können mittels Klick auf «Karte hinzufügen» verschiedene Kartenansichten zu Auswahl eingeblendet werden.

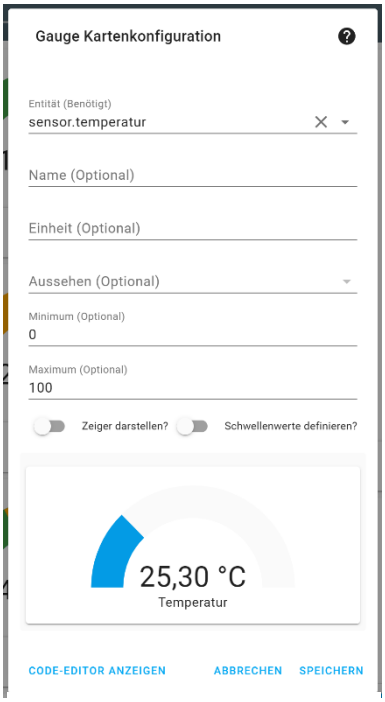


Abbildung 60: Anzeigeoption für GAUG Darstellung

Die Dokumentation beschränkt sich hier lediglich auf die Einrichtung von drei Karten, um eine schnelle Übersicht zu ermöglichen. Daher suchen wir zunächst die Karte mit dem Namen «Gauge» und richten diese für die Anzeige der Temperatur ein. Nachdem die Karte geöffnet wurde bieten sich verschiedene Einrichtungsoptionen an. In der ersten Zeile wird der Sensor ausgewählt. Der verwendete Name bezieht sich auf die Einstellung aus Kapitel 3.5.1 und kann mittels Dropdown Menu ausgewählt werden. Darunter kann ein Name und eine Einheit definiert werden, sollten diese abweichend der Einstellungen aus Kapitel 3.5.1 sein. Die Option «Aussehen» erlaubt das Einrichten von verschiedenen Anzeigethemen, da wir aber keine vorinstalliert haben, entfällt hier die Auswahl. Im folgenden Bereich kann noch die angezeigte Minimal- und Maximaltemperatur eingestellt werden. Mit dem Schalter kann, wie in der Abbildung 62, das Aussehen der Anzeige beeinflusst werden. Nachdem sich für die Anzeige mit Schwellwerten ohne Zeiger entschieden wurde, wird die Einstellung mit «Speichern» bestätigt.

Abbildung 61:Anzeigevarianten GAUGE

Im Anschluss wird die Anzeige automatisch in der Übersicht platziert. Für die Sensoren Akkulevel, Luftqualität Helligkeit & eCO2 wird genau gleich verfahren.

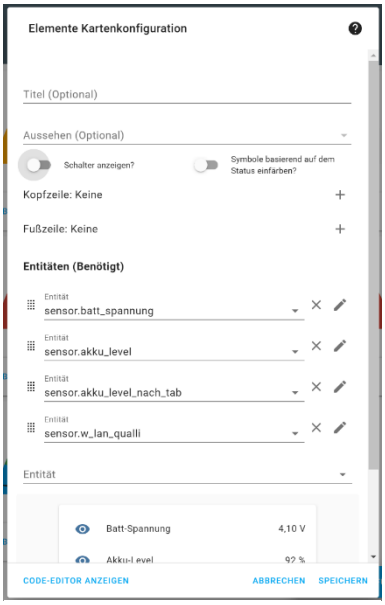


Abbildung 62: Anzeigeoption für Elemente Darstellung

Die nächste Karte ist die «Elemente» Karte, hier werden die Sensorwerte in eine Listenansicht dargestellt. Dazu müssen lediglich die Entitäten angepasst werden, hierzu wird mittels Dropdown Menu die Sensoren gesucht, die zusammengefasst werden sollen. In diesem Fall alle Informationen im Zusammenhang mit der Sensorphalanx, also Batterie Spannung, Akkulevel, Akkulevel nach Tabelle und WLAN Qualität. Nach der Auswahl werden diese unten in der Vorschau eingeblendet. Das dazugehörige Symbol kann noch durch betätigen des Stiftes angepasst werden.

Nach dem Speichern wird dieses Element in der Übersicht plaziert. Für die Anzeige der Umweltdaten werden die Einstellungen in einer gesonderten Karte gewählt.

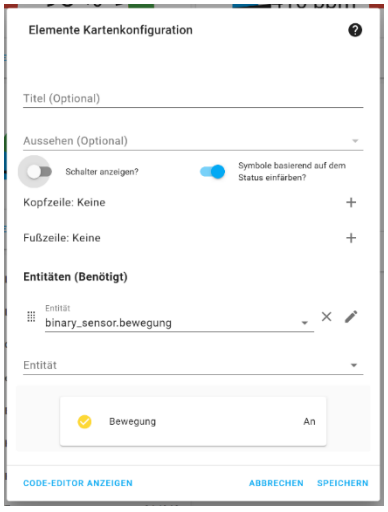


Abbildung 63:Anzeigeoption für Elemente Darstellung für binär Sensoren

Zuletzt wird der Bewegungsmelder hinzugefügt. Hierzu wird ebenfalls eine Elementekarte erstellt. Diese weicht von den bishereigen Einstellungen ab, denn hier wird Symbol als Zustandsanzeige verwendet, was mittels Auswahlschaters wird. Unten muss dann ledeglich noch der Sensor ausgewählt werden und im Anschluss wird alles gespeichert.

Wenn alle Elemente in der Übersicht enthalten sind, kann die Reihenfolge nach eigenem Wunsch verschoben werden. Dies geschieht über die, auf den Karten platzierten, Pfeilen. Wenn die Reihenfolge zufriedenstellend festgelegt wurde, kann die Bearbeitungsoberfläche mittels Klick auf das X oben rechts verlassen werden.

Damit ist die Einrichtung zur Anzeige aller Sensordaten abgeschlossen, wir können nun die Sensordaten live einsehen.

3.5.3 Bedienung: Verlauf / Historische Daten

Abbildung 65: Anzeige 24h Graph

Der Home Assistant speichert automatisch alle gesammelten Daten, dadurch bietet es den Benutzern*innen die Möglichkeit die Daten sowohl einzeln als auch im Vergleich miteinander anzuschauen und zusammenhänge festzustellen. Zum Aufrufen der historischen Daten gibt es zwei Möglichkeiten. Zum einen kann eine kleine 24 Stundenübersicht direkt durch klicken auf den Sensor Eintrag in der Übersicht aufgerufen werden. Mit einem Klick auf «Mehr anzeigen» wechselt dann die Anzeige in die Verlaufsanzeige, die uns den gewählten Sensor anzeigt. Alternativ kann direkt im linken Menu auf den Punkt «Verlauf» geklickt werden. Dies öffnet dann eine Übersicht über alle Sensorwerte.

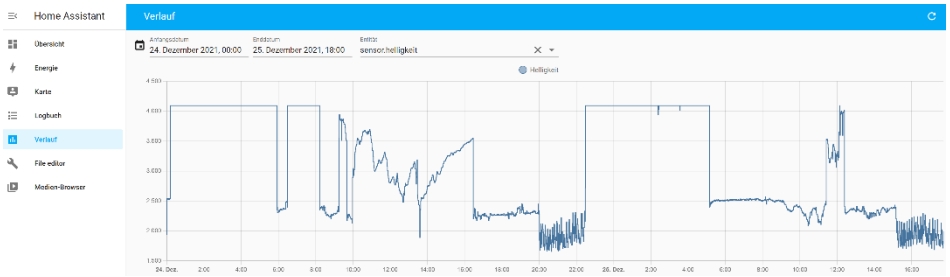


Abbildung 64: Anzeige historische Daten eines Sensors

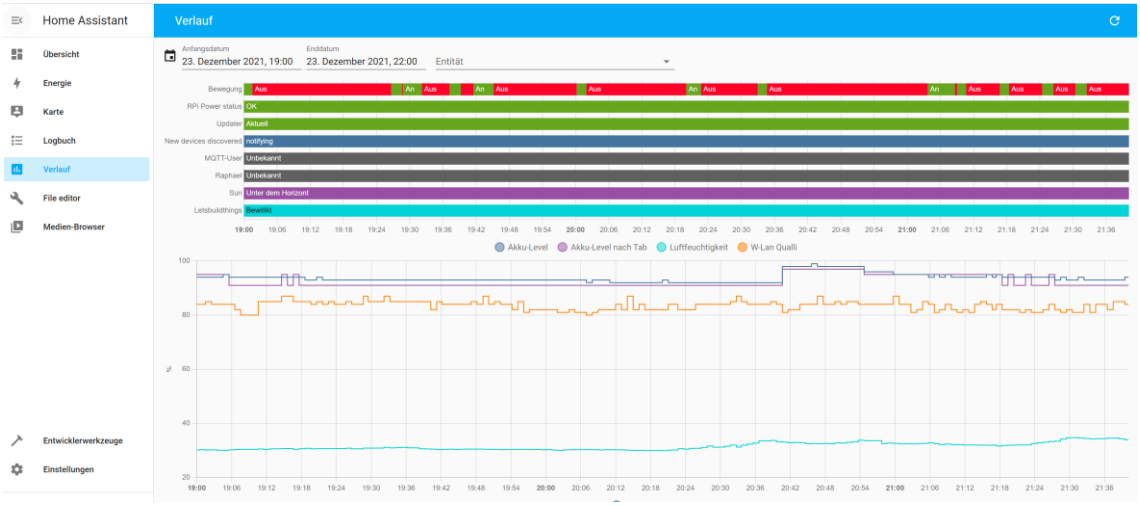


Abbildung 66: Anzeige aller Historischen Sensorwerte

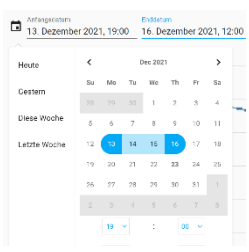


Abbildung 67: Zeitfenster zur Anzeige einstellen

Der angezeigte Zeitbereich kann mittels des Anfangs- und Enddatums eingegrenzt werden. Hierzu wird sowohl das Start- und Enddatum, als auch die Uhrzeit definiert. Nach einem Klick auf Auswählen werden die Daten aus der Datenbank gelesen und angezeigt. Wenn man nur einen einzelnen Sensor auslesen möchte, kann die entsprechende Entität in einem Dropdown Menu ausgewählt werden.

Grundsätzlich werden Werte als Balken angezeigt solange in der Datei `configuration.yaml` nicht, wie im Kapitel 3.5.1 beschreiben, die `unit_of_Masurement` defeniert wurde.

Abbildung 68: Balkenanzeige für Sensoren ohne Einheiten Definition

Wie in der Abbildung 67 zu erkennen ist werden Sensoren Werte mit gleicher Einheitendefinition in einem Grafen zusammengefasst. Nicht benötigte Anzeigen können aber oberhalb der Anzeige ausgeblendet werden.

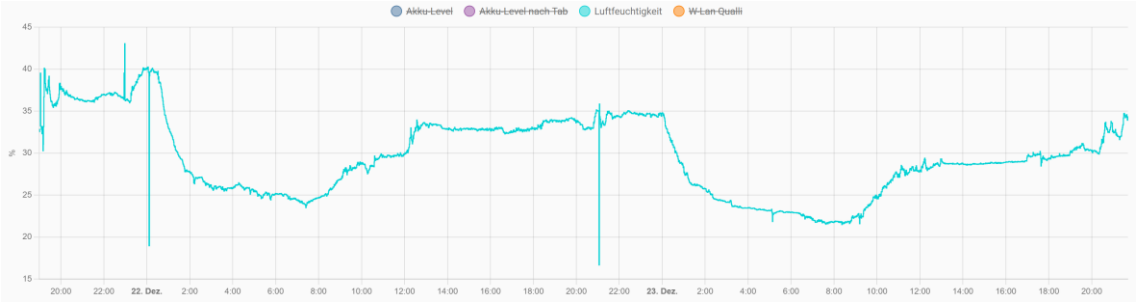


Abbildung 69: % Graf mit ausgeblendeten Werten

3.6 Auswertung Lochrasterprototyp

Durch die Möglichkeit der Einsicht in historische Daten besteht nun die Möglichkeit verschiedene Sensorwerte und Signalketten zu bewerten und daraus Schlüsse zu ziehen, um diese in die Weiterentwicklung der Sensorphalanx einfliessen zu lassen.

Zunächst wurde getestet wie lange der Lochrasterprototyp autonom in Betreib sein kann. Die folgende Messung zeigt den Spannungsverlauf des Lithium-Ionen-Akkus bis der Spannungsregler den Akku schütz und den Betrieb einstellt. Angeschlossen waren die Sensoren DHT22, BME680, BMP280, NTC und die Batterieauswertung.

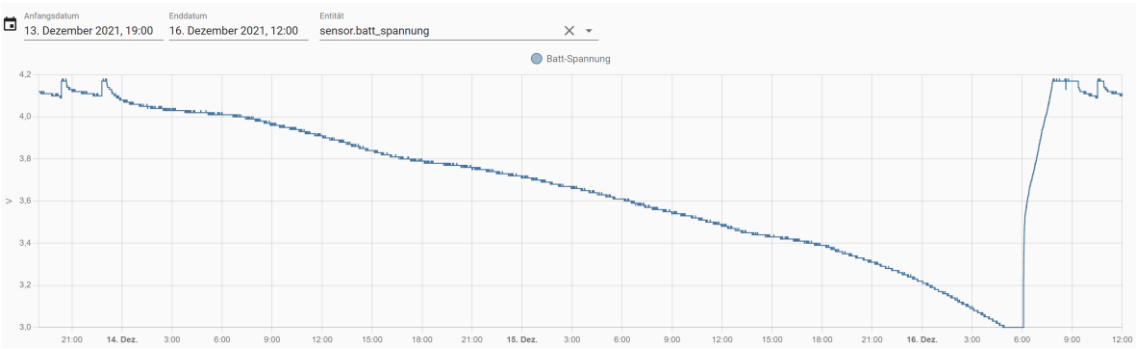


Abbildung 70: Spannungskurve des Lochrasterprototypen im Autonomen Akkubetrieb

Der Akkubetreib konnte bei einer Aktualisierungsrate von 30 Sekunden über 56h (2.33 Tage) aufrechterhalten werden. Wenn man diesen Wert als Ausgangswert nimmt, könnte man durch eine Verdopplung der Akkukapazität schon 112h (4,66 Tage)

erreichen. Wenn man die Zeitinterwall auf 60 Sekunden erhöht, ändert sich dieser Wert wieder um ca. Faktor 2 und kommt auf 224 h (9,33 Tage) wenn die Zeit auf 5 Minuten erhöht wird kommt ergibt das 1120 h (46,66 Tage = 6,6 Wochen) . Es gibt aber auch Potenzial um Energie einzusparen. Unter anderem kann man auf LEDs verzichten. Weiter ist bei der ESP32 Platine die Spannungsversorgung vom USB/Serial Wandler permanent angeschlossen und verbraucht auch bei nichtgebrauch Energie. Der Spannungsregler benötigt ebenfalls Energie für die LEDs. Durch die angeschlossenen Widerstände zur Batteriespannungsmessung fliest ebenfalls permanent Energie. Eine Möglichkeit ist unter Umständen auch die Sensoren nur bei Bedarf mit Energie zu versorgen.

Weiter ist beim Vergleich der Sensorwerte aufgefallen, dass die ausgegebenen Daten für die Luftqualität nicht mit den Gaswerten zusammenpassen. Hier ist eine Überarbeitung der Berechnung nötig. Da dies aber erst überprüfbar wurde, nachdem der GSP30 nach Lieferverzögerungen eingetroffen war, konnte dieses Problem nicht bis zur Abgabe der Diplomarbeit behoben werden.

Zusätzlich konnte die Verzögerung der Meldung des Bewegungsmelders bestimmt werden. Diese beruht darauf, dass nach Erkennen einer Bewegung zunächst eine WLAN Verbindung etabliert werden muss. Selbst bei optimalen Bedingungen dauert dies mindestens 3 Sekunden. Das macht den Bewegungsmelders als Schalter für Beleuchtung eher ungeeignet, da nur schnell auf die Bewegung reagiert werden kann, wenn auch ein Leuchtmitte direkt am ESP32 angeschlossen ist. Nichtsdestotrotz kann er zur Überwachung von Räumen und zum Aufrechterhalten von eingeschalteter Beleuchtung genutzt werden.

3.7 PCB Design

Die aus dem Lochrasterprototypen gewonnen Erkanntnisse fliessen nun in die Entwicklung des ersten PCB Prototypen. Zusätzlich zu den nötigen Bauteilen werden noch weitere Optionen eingeplant. Diese ermöglichen verschiedene Messungen vorzunehmen und gegebenenfalls Optionen offen zu halten für weitere Testaufbauten. Erstellt wird das PCB Layout mittels Autodesk Fusion 360. Dies bietet seit der Implementierung von Eagel im Januar 2020 die Möglichkeit PCB-Layouts zu erstellen und mit ein wenig mehr Aufwand auch ein 3D Modell zu erstellen. Da die Implementierung von Eagel ebenfalls bedeutet, dass eine Abwärtskompatibilität mit älteren Eagel Versionen besteht, sind auch ältere Bibliotheken mit einer grossen Zahl an PCB-Footprint und Symbolen von verschiedenen Bauelementen verfügbar. Damit diese auch in einem 3D Modell umgewandelt werden können, müssen lediglich noch 3d Steppdatein verknüpft werden. Diese können teilweise auf den Herstellerseiten oder Enthusiastenplattformen gefunden werden. Alternativ können sie auch mittels eines mitgelieferten Tools generiert werden, wenn sie auf Standartbauteilen basieren.

3.7.1 Anpassungen nach Lochrasterprototyp

Ursprünglich sollte für das PCB ein Laderegler auf Basis des IP5306 Battery Management ICs verwendet werden. Von diesem Vorhaben wird aber Abstand genommen, da es keine Arduino-IDE kompatiblen Bibliotheken für den IC gibt. Weiter wurden andere Laderegler mit I²C in Betracht gezogen, die über eine Arduino Bibliothek verfügen. Aufgrund der im Winter 2021 herrschenden Chipknappheit (15) wurde auch das Konzept verworfen, da alle möglichen Kandidaten beim Bestückungsanbieter einen unverhältnismässige Preisanstieg erfahren haben (günstigstes in Frage kommender IC lag bei 20USD / Stück). Daher wird auf einen Laderegler zurückgegriffen, der nicht mittels I²C managebar ist. Dadurch fielen auch die Battery-Manager Eigenschaften des IC weg. Daher wird die Batterie wieder mittels Messbrücke überwacht. Allerdings wird, um Energie zu sparen, diese Messbrücke schaltbar realisiert. Auf weitere Batterieschutzfunktionen wird in diesen Prototypen verzichtet. Sie sollen aber in einem möglichen Serienmodell einfliessen, wenn sich die Lage am Chipmarkt wieder normalisiert hat. Um auf ein bestehendes und bewehrtes Design zurückzugreifen wird als Grundlege das Adafruit HUZZAH32 – ESP32 Feather Board (Anhang 30) herangezogen aber mit verschiedenen Anpassungen für dieses Projekt modifiziert. Die Änderungen werden in folgendem Kapitel genauer erläutert.

3.7.2 PCB Schema

Das Schema (Anhang 31) ist in mehrere Bereiche unterteilt, um eine übersichtlichere Darstellung zu erreichen.

3.7.2.1 3.3V Power

Der Bereich 3.3V Power and Filter wurde vom Schema des EzSBC ESP32-01 Breakout and Development Board übernommen, da dieser eine sehr geringe Stromaufnahme für die Bereitstellung der 3.3V Spannung durch einen AP2112 Regulator (Anhang 32) erreicht. Zusätzlich wurde hier noch ein Batteriehalter für zwei 18650 Li-Ion Akkus verwendet, der mittels Jumper (JP12) von der gesamten Platine getrennt werden kann, dies ermöglicht sowohl ein Spannungsfreischalten als auch die Möglichkeit den Stromfluss des Akkus direkt zu messen.

3.7.2.2 5V Power

Für die 5V Spannungserzeugung wird ein AP3602AKTR-E1 (Anhang 33) verwendet. Dieser ist in der Lage bei einer Eingangsspannung von 2.7 bis 5V am Ausgang 5V und 100 mA zu erzeugen. Dies sollte ausreichend sein um sowohl einen «Neopixel» (Anhang 34) (Programmierbare RGB LED) als auch den Bewegungsmelder mit 5V zu versorgen.

3.7.2.3 Battery Guard

Zur Überwachung der Batteriespannung und somit der Akkukapazität wird eine mit Transistoren schaltbare Messbrücke verwendet. Diese basiert auf einer Schaltung aus einem Blogbeitrag von Andrey Ovcharov (24).

3.7.2.4 USB to Serial Converter

Der USB-Serial Converter wandelt die Signale, die mittels USB empfangen werden in Serial Befehle um und sendet sie an den ESP32. Abweichend vom HUZZAH32 wurden aber die VIO und VDD Anschlüsse mittels Kondensators auf Masse gelegt. Das verhindert, dass die 3.3V, die vom Converter erzeugt werden, vom ESP32 genutzt werden können. Zudem wird unterbunden, dass der Converter bei nicht angeschlossenem USB permanent mit 3.3 V versorgt wird.

3.7.2.5 Res. Terminals

Auf den Reserve Terminals sind alle Anschlüsse des ESP32 herausgeführt. Somit können auch noch Test gemacht werden, obwohl die Terminals auf der Platine verwendet werden. In einer möglichen Serienversion werden diese nur noch mit den nicht benutzten Anschlüssen des ESP32 ausgestattet.

3.7.2.6 5V Terminal

Dieses Terminal ist sowohl als Anschluss für den Bewegungsmelder, als auch eines «Neopixel», eine mit einem WS2812S Controller (Anhang 35) ausgestatteten RGB LED. Dies ermöglicht es mittels Bus Kommunikation direkt Farbwerte an die LED zu schicken. Dadurch können Anschlüsse am ESP32 eingespart werden, sowie die LED und der Bewegungsmelder mittels 5 V betreiben werden. Als Anschlüsse für die Einheiten werden hier PSS Konnektor (Anhang 36) vorgesehen.

3.7.2.7 A/Digit. Terminal

Dies Terminal dient dazu den DHT Sensor als auch weitere Analogwerte mittels PSS Anschluss aufzunehmen. Die PSS Kontakte ermöglichen es auch Module zu verwenden, die eine andere Anschlussreihenfolge auf ihrem Terminal haben. Dies ist möglich, da die Module mittels Kabel an einem PSK-Kontakt (Anhang 37) verbunden und dann auf den PSS-Kontakt aufgesteckt werden. Zusätzlich wurden Jumper eingesetzt, die es ermöglichen die 3.3V Versorgung direkt an das Modul anzuschliessen oder den geschalteten Ausgang des ESP32 als Versorgung des Moduls zu nutzen. Die Jumper sollen es zusätzlich ermöglichen den Strom, der durch die Module fliest, zu messen, um so einzuschätzen ob es überhaupt sinnvoll ist, diese schaltbar zu machen.

3.7.2.8 I²C Terminal

Das I²C Terminal bietet zusätzlich zu den Jumper Optionen aus dem A/D Terminal Anschlüsse für die I²C Kommunikation.

3.7.2.9 LIPO Charging

Der Bereich LIPO Charging wurde nahezu eins zu eins vom HUZZAH32 übernommen, lediglich die LED wurde ersetzt.

3.7.2.10Power LED

Die Power LED ist mittels Jumper an 3.3V verbunden, so dass auch diese zu Energiesparzwecken abgetrennt werden kann.

3.7.3 PBC Layout

Im nächsten Schritt wird die Form und Grösse des PCBs definiert. Um eine möglichst kompakte Bauform zu erreichen, wird sich an dem Batteriehalter der 18650 Akkus orientiert, da diese das grösste Objekt darstellt. Beim Platzieren der einzelnen Komponenten wird darauf wertgelegt, dass die Zuordnung der einzelnen Bereiche wie im Schema dargestellt werden um eine leichtere Fehlersuche zu ermöglichen. Im Anschluss werden die Beschriftungen angepasst und Orientierungspunkte (Potentiale an den Anschlüssen) angelegt. Das Routing der Leiterbahnen übernimmt die Software und gegebenenfalls wird manuel nachbearbeitet.

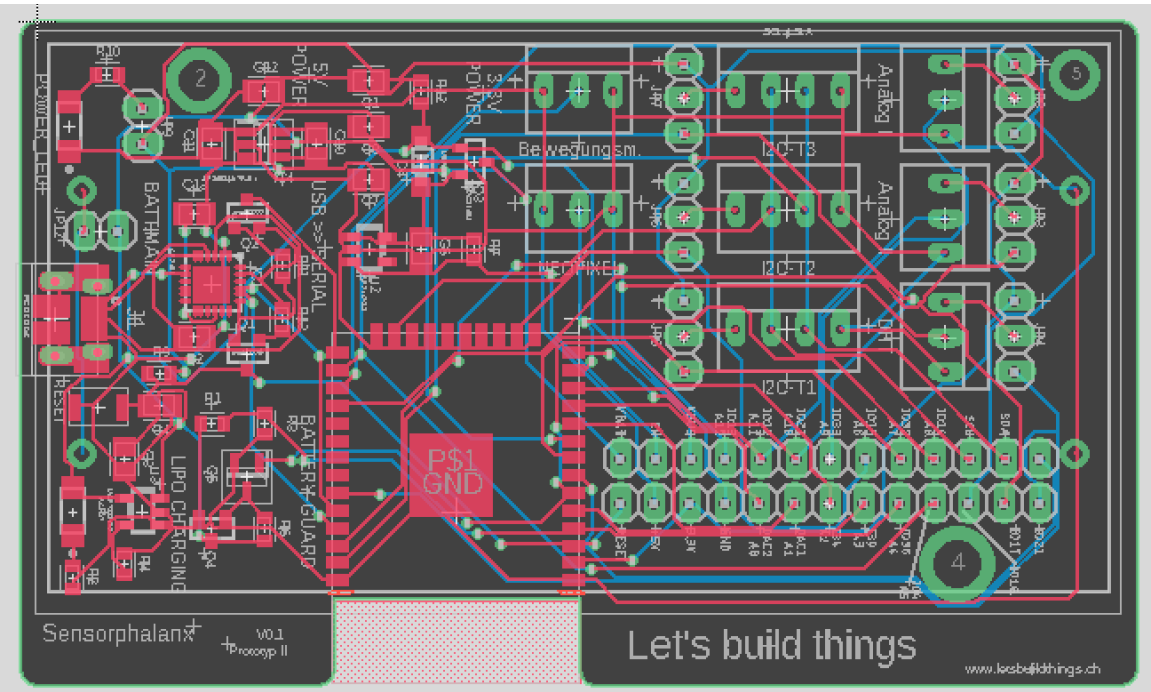


Abbildung 71: Platinen Layout des PTC Prototypen Rot=Oberseite, Blau = Unterseite, Grün = konventionelle Lütpunkte

Für die Bestellung bei PCBWay werden zum Abschluss die Gerberdatein exportiert. Diese beinhalten die Koordinaten für die Leiterbahnen, sowie die Bauteile als auch eine Stückliste der benötigten Komponenten. Bei der Stückliste werden noch die Komponenten, die keine SMD Bauform haben, entfernt, da diese manuell aufgelötet werden.

3.7.4 3D PCB

Als positiven Benefit der Nutzung von Fusion 360 für die PCB Layouterstellung ist die Möglichkeit ein 3D Modell zu erzeugen. Dies kann für die Erstellung eines Gehäuses genutzt werden.

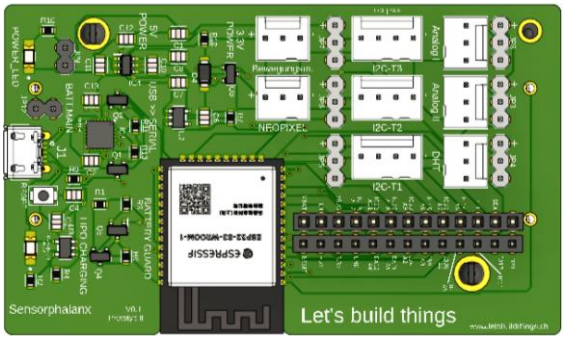


Abbildung 73: CAD Darstellung PCB Vorderseite

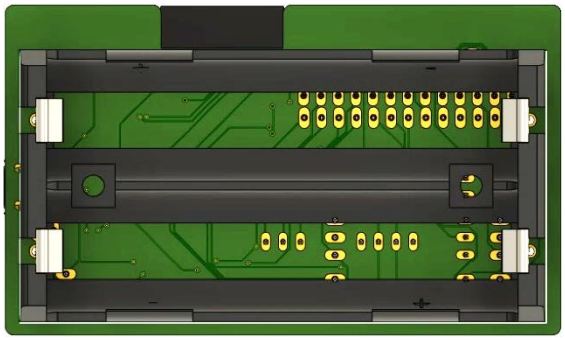


Abbildung 72: CAD PCB Rückseite

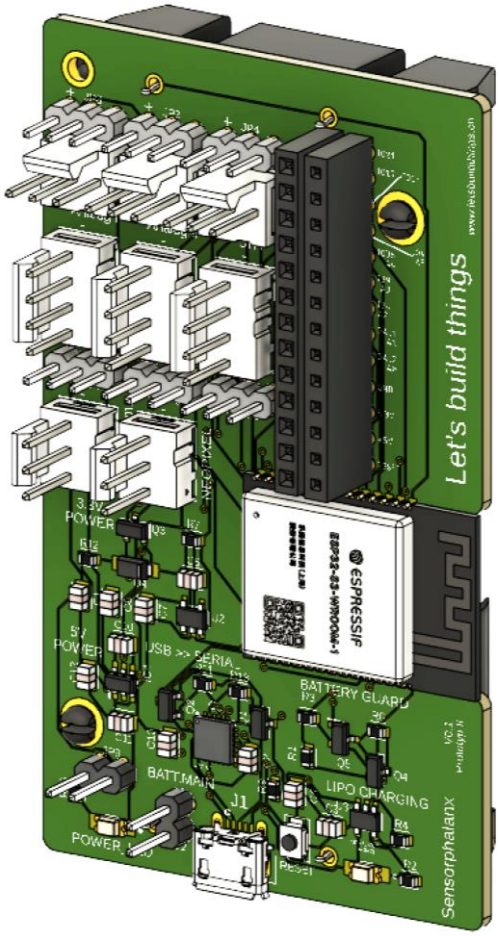


Abbildung 74: 3D CAD PCB

4 Zusammenfassung

Dieses Kapitel soll einen Überblick über die vorgestellte Diplomarbeit verschaffen. Zunächst gibt es ein zusammenfassendes Fazit im Kapitel 4.1. Im Kapitel 4.2 werden offene Punkte der aktuellen Version des Prototypens thematisiert. Das Kapitel 4.3 beschäftigt sich mit Ideen für die Weiterentwicklung des Projekts.

4.1 Fazit

In der Diplomarbeit wird ein Prototyp entwickelt, der in der Lage ist, mittels verschiedener, austauschbarer Module Raumdaten wie Luftfeuchtigkeit, Temperatur Helligkeit, Gas-Konzentration, Luftqualität usw. zu sammeln. Diese können gegebenenfalls zusammengeführt werden, bevor sie an ein das Home-Automations-System, Home Assistant, übertragen werden. Zusätzlich wird aufgezeigt, wie die gesammelten Daten mittel Home Assistant aufgezeichnet und visualisiert werden können.

Verschiedene Module werden für die Implementierung in den Prototypen gegeneinander abgewogen und acht davon ausgewählt. Weiter werden diverse andere Komponenten festgelegt, die für eine Umsetzung der Anforderungen nötig sind. Besonders hervorzuheben ist hier der ESP32, der das Herzstück des Prototypens bildet.

Nach Planung und Realisierung eines Lochrasterprototypen, wird dieser mit den ausgewählten Komponenten bestückt und im Anschluss eine Software mit verschiedenen Unterprogrammen entwickelt. Diese Unterprogramme ermöglichen das Sammeln und Zusammenfassen der Raumdaten sowie das Übermitteln dieser an das Home-Automations-System. Um Energie zu sparen wird im Anschluss an einen Sammel- und Sendezyklus der ESP32 wieder in den Tiefschlaf versetzt.

Nachdem das Programm entsprechend der Anforderungen funktioniert, wird mit der Planung eines PCB Prototypen begonnen. Dieser berücksichtigt verschiedene Erkenntnisse aus dem ersten Prototyp und wird von der Schemaerstellung über das PCB Layout bis zum 3D-Model dokumentiert.

Die Grundlage für eine Weiterentwicklung hin zu einem Bausatz, der für viele Maker*innen interessant ist, konnte mit dem Prototyp gestartet werden.

Das Optionale Ziel, die Veröffentlichung des Fortschritts der Diplomarbeit, konnte zunächst über neun Wochen realisiert werden, musste aber zugunsten des Projektfortschritts pausiert werden.

4.2 Zukünftige Arbeit

Die Software des Lochrasterprototypen enthält noch Aspekte, die eine Nachbesserung nötigt machen.

Ein wichtiger Aspekt ist hier die korrekte Berechnung der IAQ Werte und die entsprechende Darstellung im Home Assistant. Weiter ist die Möglichkeit, Daten bei nicht Erreichbarkeit vom Home Assistant zwischenzuspeichern und bei Erreichbarkeit des Home Assistant gesammelt zu übermitteln, aus Zeitgründen nicht umgesetzt worden.

Die erste PCB-Version des Prototyps befindet sich auf dem Weg aus Shenyang (PCB-Way) in die Schweiz und kann erst nach der Ankunft mit den fehlenden Bauteilen und der angepassten Software bestückt und in Home Assistant eingebunden werden.

Zusätzlich ist noch die Konstruktion eines Gehäuses für den PCB Prototypen nicht realisiert worden, da auch hierfür nicht mehr genügend Zeit vorhanden ist.

Das Wunschziel, die Veröffentlichung der Diplomarbeit und des Source code, wird voraussichtlich am 11.01.2022 auf letsbuildthings.ch erfolgen. Die Verzögerung entsteht durch die Produktion eines Videos.

Noch offene Arbeiten:

- Software Anpassungen IAQ Berechnung
- PCB-Prototyp bestücken und testen
- Gehäusekonstruktion und Druck
- Veröffentlichen der Ergebnisse inklusive Video

4.3 Ausblick

Der Sensorphalanxprototyp bietet eine gute Basis für eine weitere Entwicklung bis hin zu einem Bausatz. Auch die Weiterentwicklung zu einem Aktor-Modul mit dem dann beispielsweise Heizungen oder Beleuchtung geregelt werden können, ist denkbar.

Auch die Softwaremöglichkeiten des ESP32 sind noch nicht ansatzweise ausgereizt. Möglich wäre beispielsweise die Auswahl der installierten Sensoren nicht vor dem Laden der Software zu realisieren, sondern diese mittels Webserver direkt auf dem ESP32 einzurichten. Hier könnten auch weitere Einstellungen bequem vorgenommen werden.

Weiter bietet auch Home Assistant Potential für weitere Möglichkeiten der Anpassung und Erweiterung, die noch nicht in dieser Arbeit berücksichtigt wurden.

Literaturverzeichnis

1. **Krempl, Stefan.** 35C3: Über die "smarte" Glühbirne das Heimnetzwerk hacken. *heise online.* [Online] 29. 12 2018. [Zitat vom: 27. 12 2021.] https://www.heise.de/newsticker/meldung/35C3-Ueber-die-smarte-Gluehbirne-das-Heimnetzwerk-hacken-4259891.html.

2. **Wikipedia-Autoren.** Open Source. *Wikipedia – Die freie Enzyklopädie.* [Online] 06. 12 2021. [Zitat vom: 08. 12 2021.]

3. —. Proprietäre Software. *Wikipedia – Die freie Enzyklopädie.* [Online] 10. 09 2019. [Zitat vom: 01. 12 2021.] https://de.wikipedia.org/w/index.php?title=Propriet%C3%A4re_Software&oldid=192150907.

4. **contributors, Wikipedia.** OpenHAB. *Wikipedia.* [Online] Wikipedia, 11. 09 2021. [Zitat vom: 31. 10 2021.] https://en.wikipedia.org/w/index.php?title=OpenHAB&oldid=1043655906.

5. **e.V., openHAB Foundation.** openHAB. [Online] openHAB Foundation e.V. [Zitat vom: 31. 10 2021.] https://www.openhab.org.

6. **Wikipedia-Autoren.** Home Assistant. [Online] Wikipedia – Die freie Enzyklopädie, 12. 10 2021. [Zitat vom: 01. 11 2021.] https://de.wikipedia.org/w/index.php?title=Home_Assistant&oldid=216315742.

7. **Hedda.** community.home-assistant.io. *Home Assistant ist #2 und belegte den zweiten Platz beim GitHub "State of the Octoverse" 2020.* [Online] 03. 12 2020. [Zitat vom: 28. 12 2021.] https://www.reddit.com/r/homeassistant/comments/k5us3f/home_assistant_is_2_as_took_second_place_at/.

8. **Wikipedia-Autoren.** IoBroker. [Online] Wikipedia – Die freie Enzyklopädie, 18. 10 2021. [Zitat vom: 01. 11 2021.] https://de.wikipedia.org/w/index.php?title=IoBroker&oldid=216468214.

9. **iobroker.** iobroker. *iobroker.* [Online] [Zitat vom: 01. 11 2021.] https://www.iobroker.net/#de/documentation/README.md.

10. **Zimmer, Jonas.** *MQTT 5 als Anwendungsprotokoll iim Internet der Dinge.* Hochschulle Offenburg. Offenburg : Hochschulle Offenburg, 2019.

11. **Waschbusch, Thomas Joos / Lisa Marie.** HTTP, XMPP, CoAPP und MQTT: IoT-Protokolle für die Kommunikation. *industry-of-things.* [Online] 27. 03 2019. [Zitat vom: 25. 11 2021.] https://www.industry-of-things.de/http-xmpp-coapp-und-mqtt-iot-protokolle-fuer-die-kommunikation-a-813079/.

12. **Wikipedia-Autoren.** I²C. *Wikipedia – Die freie Enzyklopädie.* [Online] 31. 07 2021. [Zitat vom: 25. 11 2021.] https://de.wikipedia.org/w/index.php?title=I%C2%B2C&oldid=214379596.

13. —. Universal Asynchronous Receiver Transmitter. *Wikipedia – Die freie Enzyklopädie.* [Online] Wikipedia – Die freie Enzyklopädie, 13. 12 2020. [Zitat vom: 26. 11 2021.]

https://de.wikipedia.org/w/index.php?title=Universal_Asynchronous_Receiver_Transmitter&oldid=206516581.

14. **Assistant, Home.** Install Home Assistant. *home-assistant.io.* [Online] [Zitat vom: 03. 11 2021.] https://www.home-assistant.io/installation/.

15. **Cording, Stuart.** elektormagazine. *Was ist Raspberry Pi?* [Online] 26. 05 2021. [Zitat vom: 04. 11 2021.] https://www.elektormagazine.de/news/was-ist-raspberry-pi.

16. **Silvio Werner.** Chipkrise: Der Raspberry Pi bekommt eine saftige Preiserhöhung, Lieferbarkeit eingeschränkt. *Notebookcheck.com.* [Online] 20. 10 2021. [Zitat vom: 04. 11 2021.] https://www.notebookcheck.com/Chipkrise-Der-Raspberry-Pi-bekommt-eine-saftige-Preiserhoehung-Lieferbarkeit-eingeschraenkt.574162.0.html.

17. **Eicken, Thorsten von.** Running Wifi Microcontrollers on Battery. *TvE`s Blog.* [Online] 18. 11 2018. [Zitat vom: 07. 11 2021.] https://blog.voneicken.com/projects/low-power-wifi-intro/.

18. —. Low-Power ESP32 Boards. *TvE`s Blog.* [Online] 25. 08 2019. [Zitat vom: 07. 11 2021.] https://blog.voneicken.com/2018/lp-wifi-esp32-boards/.

19. —. ESP8266 vs. ESP32 on Battery Power. *TvE`s Blog.* [Online] 10. 12 2018. [Zitat vom: 07. 11 2021.] https://blog.voneicken.com/2018/lp-wifi-esp-comparison/.

20. **Wikipedia-Autoren.** Lithium-Ionen-Akkumulator. [Online] Wikipedia – Die freie Enzyklopädie., 07. 11 2021. [Zitat vom: 09. 11 2021.] https://de.wikipedia.org/w/index.php?title=Lithium-Ionen-Akkumulator&oldid=217053483.

21. —. Bewegungsmelder. *Wikipedia – Die freie Enzyklopädie.* [Online] 05. 05 2021. [Zitat vom: 11. 11 2021.] https://de.wikipedia.org/w/index.php?title=Bewegungsmelder&oldid=210585129.

22. **Heiko.** i2c-schnittstelle-esp32-arduino. *unsinnsbasis.de.* [Online] 23. 11 2020. [Zitat vom: 19. 12 2021.] https://unsinnsbasis.de/i2c-schnittstelle-esp32-arduino/.

23. **Draeger, Stefan.** Arduino Lektion #113: Umweltsensor BME680. *#113: Umweltsensor BME680.* [Online] 30. 11 2020. [Zitat vom: 07. 12 2022.] https://draeger-it.blog/arduino-lektion-113-umweltsensor-bme680-fuer-rel-luftfeuchtigkeit-luftdruck-temperatur-und-luftqualitaet/.

24. **Ovcharov, Andrey.** How to measure battery level with ESP32 microcontroller. *en.ovcharov.me.* [Online] 29. 02 2020. [Zitat vom: 2020. 12 26.] https://en.ovcharov.me/2020/02/29/how-to-measure-battery-level-with-esp32-microcontroller/.

25. **home-assistant, developers @.** Kernarchitektur. *developers.home-assistant.io.* [Online] 02. 11 2021. [Zitat vom: 23. 05 2021.] https://developers.home-assistant.io/docs/architecture/core.

26. **Reichelt.de.** RASPBERRY PI 3B+. *Reichelt.de.* [Online] [Zitat vom: 04. 11 2021.] https://www.reichelt.de/ch/de/raspberry-pi-3-b-4x-1-4-ghz-1-gb-ram-wlan-bt-raspberry-pi-3b--p217696.html?search=raspberry+pi+3+b%2B&&r=1.

27. —. RASP PI 4 B 4GB. *reichelt.de.* [Online] [Zitat vom: 04. 11 2021.] https://www.reichelt.de/ch/de/raspberry-pi-4-b-4x-1-5-ghz-4-gb-ram-wlan-bt-rasp-pi-4-b-4gb-p259920.html?&trstct=pos_2&nbc=1.

28. **Wikipedia-Autoren.** ESP8266. [Online] Wikipedia – Die freie Enzyklopädie, 09. 02 2021. [Zitat vom: 07. 11 2021.] https://de.wikipedia.org/w/index.php?title=ESP8266&oldid=208631060.

29. **Reichelt.** DEBO BMP280 . *Reichelt.de.* [Online] [Zitat vom: 10. 11 2021.] https://www.reichelt.com/ch/de/entwicklerboards-temperatur-und-drucksensor-bmp280-debo-bmp280-p266034.html?&trstct=pos_0&nbc=1.

30. **Microsoft.** GitHub ioBroker Insights. [Online] Microsoft. [Zitat vom: 01. 11 2021.] https://github.com/ioBroker/ioBroker/graphs/contributors.

Anhang

Die folgenden Dateien befinden sich im Ordner der Diplomarbeit

Anhang 1: Terminplan
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Ergebnisse\Terminplan.xlsx

Anhang 2: Raspberry Pi-3 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\raspberry-pi-3-datasheet.pdf

Anhang 3: Raspberry Pi-3+ - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\Raspberry-Pi-Model-Bplus-Product-Brief.pdf

Anhang 4: Raspberry Pi-4 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\raspberry-pi-4-datasheet.pdf

Anhang 5: ESP8266 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\ Datenblaetter\0a-esp8266ex_datasheet_en.pdf

Anhang 6: ESP32 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\esp32_datasheet_en.pdf

Anhang 7: EzSBC ESP32-01 Breakout and Development Board - Schema
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\Esp32_1D_s.pdf

Anhang 8: NodeMCU ESP32 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\SBC-NODEMCU-ESP32-DATASHEET_V1.2.pdf

Anhang 9: Lithium-Ionen-Akkus vom Typ 18650 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\PAN18650B_DS-EN.pdf

Anhang 10: IP5306 Battery Management ICs - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\IP5306-Injoinic.pdf

Anhang 11: HC-SR505 - Anleitung
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\HC-SR505-MINI-PIR-SENSOR-INFRAROT-BEWEGUNGSMELDER.pdf

Anhang 12: SEN-HC-SR501- Anleitung
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\SEN-HC-SR501-ANLEITUNG-23.09.2020.pdf

Anhang 13: LDR - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\SE012.pdf

Anhang 14: BH1750 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\adafruit-bh1750-ambient-light-sensor.pdf

Anhang 15: TSL2591 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\adafruit-tsl2591.pdf

Anhang 16: CCS811 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\CN04-2019_attachment_CCS811_Datasheet_v1-06.pdf

Anhang 17: SGP30 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\adafruit-sgp30-gas-tvoc-eco2-mox-sensor.pdf

Anhang 18: ST1147 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\datasheet-1485330-iduino-1485330-temperature-sensor-1-pcs.pdf

Anhang 19: NTC-MF52 9350 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\pj2-mf52type-1554.pdf

Anhang 20: DS18B20 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\DS18B20.pdf

Anhang 21: DHT11 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

Anhang 22: DHT22 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\DHT22.pdf

Anhang 23: Bosch BMP280 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\BST-BMP280-DS001-11.pdf

Anhang 24: MPL115A2 Chip - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\MPL115A2.pdf

Anhang 25: BME280 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\bst-bme280-ds002.pdf

Anhang 26: MS8607 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\ENG_DS_MS8607-02BA01_B3.pdf

Anhang 27: Bosch BME680 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\BME680.pdf

Anhang 28:Layout Lochraster
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Ergebnisse\Layout-Lochraster.pdf

Anhang 29: Quellcode der Diplomarbeit
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Ergebnisse\Quellcode_Sensorphalanx_Lochraster_Prototyp.pdf

Anhang 30: Adafruit HUZZAH32 – ESP32 Feather Board - Handbuch
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\adafruit-huzzah32-esp32-feather.pdf

Anhang 31: Elektronikschema Sensorphalanx PCB-Prototyp - Schema
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Ergebnisse\Elektronikschema_Sensorphalanx_PCB_Prototyp.pdf

Anhang 32: AP2112 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\AP2112.pdf

Anhang 33: AP3602AKTR-E1 - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\BCDSS01551_1-2512535.pdf

Anhang 34: Neopixel - Handbuch
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\adafruit-neopixel-uberguide.pdf

Anhang 35: WS2812S - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\WS2812S.pdf

Anhang 36: PSS Kontakt - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\IMP.pdf

Anhang 37: PSK-Kontakt - Datenblatt
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Datenblaetter\PSK_254-5W_DB.pdf

Besprechungsprotokolle 1
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Protokolle\Sitzungsprotokoll_I.pdf

Besprechungsprotokolle 2
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Protokolle\Sitzungsprotokoll_II.pdf

PDF-Version der Arbeit
Dateipfad: ..\DP_Raphael_Beckmann\DP_Raphael_Beckmann.pdf

Tagesprotokolle
Dateipfad: ..\DP_Raphael_Beckmann\Anhang\Protokolle\Tagesprotokolle.xlsx

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Flawil am 10.01.2022

Raphael Beckmann