



## Bachelorarbeit

---

# Entwurf und Implementierung eines modularen Multisensorknotens für Smart Home-Anwendungen

---

---

## Design and implementation of a modular multi-sensor node for smart home applications

---

Verfasser: Dennis Makein

Matrikelnummer: 420832

Abgabedatum: 24. Mai 2025

Entwurf Mikroelektronischer Systeme (EMS)

Prüfer: Prof.Dr.-Ing. Norbert Wehn

Betreuer: Dr.Ing. Christian De Schryver

## **Eidesstattliche Erklärung**

Ich versichere hiermit an Eides statt, die vorliegende Arbeit gemäß BPO/MPO Elektrotechnik und Informationstechnik bis auf die durch meinen Betreuer gewährte Unterstützung ohne Hilfe Dritter selbstständig angefertigt, alle benutzten Quellen und Hilfsmittel einschließlich des Internets vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert, mit Abkürzung oder sinngemäß übernommen wurde.

Kaiserslautern, den 24. Mai 2025

Dennis Makein

## **Kurzfassung**

hallo

## **Abstract**

hal

# Inhaltsverzeichnis

<b>Eidesstattliche Erklärung</b>	II
<b>Kurzfassung   Abstract</b>	III
<b>1 Einleitung</b>	1
<b>2 State of the Art</b>	2
2.1 Mikrocontroller . . . . .	2
2.2 SmartHome Standards . . . . .	3
2.2.1 ZigBee . . . . .	3
2.2.2 Z-Wave . . . . .	4
2.2.3 BLE . . . . .	4
2.2.4 Wi-Fi . . . . .	4
2.3 SmartHome Produkte . . . . .	5
2.3.1 Homematic . . . . .	5
<b>3 Theoretische Grundlagen</b>	6
3.1 Sensorenauswahl . . . . .	6
3.2 Grundlagen von I2C . . . . .	7
3.3 Sensorenbeschreibung . . . . .	8
3.3.1 IST HYT 271 . . . . .	8
3.3.2 Joy-it SEN-CCS811V1 . . . . .	10
3.3.3 Omron D6T1A01 . . . . .	12
3.4 MQTT . . . . .	14
3.5 Matter . . . . .	14
3.5.1 Weitere Begriffe . . . . .	16
3.5.2 Thread-Protokoll . . . . .	17
<b>4 Durchführung</b>	18
4.1 Aufbau auf dem Breadboard . . . . .	18
4.2 Einrichtung VS Code mit der ESP-IDF . . . . .	19
4.3 Matter Sensors . . . . .	19
4.3.1 Aufbau . . . . .	19

## Inhaltsverzeichnis

---

4.3.2 Drivers . . . . .	19
4.3.3 app_main . . . . .	20
4.4 PCB . . . . .	21
4.5 SmartHome Server . . . . .	21
<b>5 Ergebnisse und Auswertung</b>	<b>22</b>
<b>6 Schlussfolgerung und Ausblick</b>	<b>23</b>
<b>7 Kapitel Beispiel</b>	<b>24</b>
7.1 Kapitel bsp.1 . . . . .	24
7.1.1 Formeln . . . . .	26
7.1.2 Grafiken . . . . .	26
7.1.3 Literatur . . . . .	27
7.1.4 Deckblatt . . . . .	27
7.1.5 Übersicht . . . . .	27
<b>Literaturverzeichnis</b>	<b>32</b>
<b>Abkürzungsverzeichnis</b>	<b>33</b>
<b>Symbolverzeichnis</b>	<b>34</b>
<b>Glossar</b>	<b>35</b>
<b>Tabellenverzeichnis</b>	<b>36</b>
<b>Abbildungsverzeichnis</b>	<b>37</b>
<b>Anhang</b>	<b>38</b>
A.1 Anhang 1 . . . . .	38

---

# **1 Einleitung**

---

## 2 State of the Art

### 2.1 Mikrocontroller

Im Folgenden wird ein kurzer Überblick über verschiedene Mikrocontroller geschaffen. Darunter auch der in diesem Projekt verwendete ESP32-C6:

Der **ESP32-C6** ist ein Mikrocontroller der Firma Espessif, welcher in vielen Bereichen seinen Einsatz finden kann. Gerade in neuartigen IoT-basierten Anwendungen kann der ESP32-C6 aufgrund von seiner Vielseitigkeit überzeugen. Der Controller unterstützt Wi-Fi 6(2.2.4), Bluetooth 5 (/ BLE(2.2.3)), ZigBee(2.2.1) und auch Thread3.5.1. Anwendungsbeispiele mit dem ESP können z.B. im SmartHome-Bereich, der Gesundheitsüberwachung, einem POS-System (Point of Sale) oder auch generellen Einsatz in der Industrie finden. Die Software des ESP ist open-source, was für die Entwicklung und Innovation einen großen Vorteil bietet, da diese durch die Zusammenarbeit vieler schneller vorangetrieben werden können. Die Hardware des Chips hingegen ist nicht open-source und genauere Details sind unter Verschluss gehalten. Bekannt ist z.B., dass der Hauptbaustein dieses Chips ein 32 Bit RISC-V Prozessor ist. Dieser ist im ESP32-C6-WROOM verbaut. Der WROOM kann Teil eines DevKits sein, mit welchem auch in diesem Projekt gearbeitet wird.

Als weiterer großer Vertreiber im Markt für Mikrocontroller ist vor allem noch **Arduino** zu nennen. Arduino ist bekannt für die große Anzahl an möglichen Boards, wie z.B. die bekannten NANO oder UNO. Viele der Arduino-Boards sind gerade für die IoT-Benutzung ausgelegt. Sie sind mit BLE(2.2.3) und Wi-Fi(2.2.4) ausgestattet, was einen vielseitigen Einsatz z.B. im Heimnetzwerk, zur Automatisierung oder auch mit Robotern ermöglicht. Hier ist ebenfalls die Software, wie beim ESP(2.1), open-source. Als Chips der Arduino-Boards werden vermehrt Atmel AVR Mikrocontroller (z.B. die ATmega-Reihe) verwendet. Zudem gibt es auch Erweiterungen für die Boards, Shields genannt, die noch einmal speziellere Funktionen erfüllen können. Des Weiteren ist die eigene integrierte IDE (Entwicklungsumgebung) zu erwähnen, in welcher die Programmierung in einer einfacher gehaltenen Version von C++ erfolgt.

Ein anderer bekannter Hersteller ist **Raspberry Pi**. Hier gibt es ebenfalls viele verschiedene vorgefertigte Boards, Module oder auch Erweiterungen, aus denen anhand der speziellen Anforderungen ausgewählt werden kann. Diese werden ebenso wie der ESP(2.1) und Arduino(2.1) in vielen Gebieten, wie z.B. IoT, SmartHome oder in der Industrie, benutzt. Dabei können die Schnittstellen zwischen den Boards variieren, aber die meisten sind Bluetooth(2.2.3), Wi-

Fi(2.2.4), HDMI und USB-fähig. Die Schaltpläne der Boards sind teilweise dokumentiert und viele Zusatzplatinen sind auch open-source. Die Prozessoren hingegen sind proprietär. Das Betriebssystem des Raspberry Pi ist hingegen vollständig open-source. Es ist eine auf Debian basierende Linux Distribution, welche viele Programme wie z.B. Python, HomeAssistant oder Docker unterstützt.

## 2.2 SmartHome Standards

Zwar werden hier in diesem Projekt nur Matter und die Charakteristiken von MQTT benutzt, jedoch werden im Folgenden noch diverse andere Standards für Multi-Sensor Applikationen in Smart-Home Netzwerken aufgezeigt, um einen allgemeinen Überblick über das Thema zu verschaffen.

### 2.2.1 ZigBee

ZigBee ist mit dem IEEE 802.15.4 Standard eingeführt worden. Zu ZigBee gehören Unternehmen wie Invensys, Honeywell, Mitsubishi Electric, Motorola, Philips und viele mehr. ZigBee ist sehr stark für verbindungslose Sensoren ausgelegt. Der Standard ist kosteneffizient, stromsparend, hat eine geringe Latenz. Dabei ist die Datenrate im Vergleich zu Bluetooth2.2.3 oder Wi-Fi2.2.4 um ein Vielfaches niedriger, wohingegen bei ZigBee die mögliche Anzahl an Nodes pro Netzwerk positiv herausragt (653356 Nodes). Die Distanz zwischen 2 Geräten kann bis zu 50 Meter betragen. Dabei kann jedes Gerät die Daten an ein anderes Gerät weiterleiten. Es gibt 2 verschiedenen Arten von Geräten in dem Netzwerk: FFD = fully functional device und RFD = reduced functional device. Ein FFD kann mit jedem Gerät kommunizieren, neben der Endgerätfunktion auch als Netzwerk Koordinator (network coordinator) oder auch Router dienen und in jeder Topologieart eingesetzt werden. Ein RFD hingegen kann nur mit einem Netzwerk Koordinator kommunizieren und nur in einer Stern-Topologie vernetzt sein. Ein RFD kann z.B. ein klassischer Sensor sein, welcher sich auch, typisch für ZigBee, in einen Schlaf-Modus (snooz) versetzen kann, wenn dieser keine Anfrage erhält, um somit Energie zu sparen. Außerdem ist im Rahmen des Sicherheitsaspektes zu erwähnen, dass Geräte in einem ZigBee-Netzwerk eine Liste an vertrauenswürdigen Geräten innerhalb des Netzwerks besitzen. Hier wird ein kryptischer Schlüssel benutzt, welcher den Geräten bei der Aufnahme in das Netzwerk mitgegeben wird (ähnlich wie bei Matter3.5). [1, 2]

### **2.2.2 Z-Wave**

Z-Wave ist ein Funkstandard mit Niedrigenergie-Funkwellen. Das System benutzt im Vergleich zu den anderen aufgelisteten Protokollen eine Frequenz von 868,42 MHz, sodass keine Interferenz mit diesen entstehen. Die Manchester-Kodierung, die bei diesem Standard verwendet wird, ist eine Form der Phasenmodulation. Ein Z-Wave Netzwerk ist in der Mesh-Netzwerktopologie aufgebaut. Jedes Netzwerk besitzt seine eigene Netzwerk-ID. Dabei kann jedes Netzwerk bis zu 232 Knotenpunkte besitzen. Nur Knoten mit gleicher Netzwerk-IDs können miteinander kommunizieren. Zur Einrichtung und Administrierung eines Netzwerkes wird ein zentraler Controller benötigt. Je mehr Geräte in einem Netzwerk vorhanden sind, desto stärker und weiter reicht das gesendete Signal. Zudem ist eine Erweiterung der Reichweite durch einen Repeater möglich. Z-Wave besitzt eine Low-Power-RF-Kommunikationstechnologie für Mesh-Netzwerke ohne Koordinatorknoten. Für die Sicherheit in den Netzwerken sorgt das AES-128 Verschlüsselungssystem. Negativ zu notieren ist, dass Z-Wave nur mit einer geringen Datenübertragungsgeschwindigkeit von 100 Kbit/s arbeitet, da das System voll und ganz auf auf den Gebrauch im Smart-Home Bereich ausgelegt ist. [3]

### **2.2.3 BLE**

Bluetooth ist ein verbindungsloser Standard, der im alltäglichen Leben Gebrauch findet. BLE (Bluetooth low energy) wurde in der Version Bluetooth 4 mit eingeführt. Wie LE schon sagt, war es das Ziel Bluetooth energieeffizienter zu gestalten. Dies wurde vor allem durch die immer größere und auch ständige bzw. länger andauernde Vernetzung von Geräten, z.B. IoT (Internet of Things), miteinander ein wichtiger Vorteil. Ein Beispiel hierzu kann der Einsatz von BLE zur Gerätortung sein. Dabei arbeitet BLE, genauso wie das klassische Bluetooth im 2,4 GHz Frequenzband. Ein Gerät kann neben Punkt-zu-Punkt-, wie beim herkömmlichen Bluetooth, auch in Broadcast- oder Mesh-Topologie angebunden sein. [4, 5]

### **2.2.4 Wi-Fi**

Wi-Fi (Wireless Fidelity) ist neben Bluetooth 2.2.3 die wohl am weitesten bekannte und verbreitete Technologie. Bei Wi-Fi gibt es zwei verschiedene Frequenzbänder, das 2,4 GHz und das 5 GHz Band. Wi-Fi basiert auf der IEEE 802.11 und der aktuell neuste Standard ist Wi-Fi 7. WPA3 (Wi-Fi Protected Access 3) ist für die Sicherheit der Verbindung zuständig. Dabei gibt es wiederum zwei Arten, WPA3-Personal, welches sich nach dem klassischen Verbraucher richtet,

und WPA3-Enterprise, welches auf die Sicherheit für Unternehmen spezialisiert ist. Wi-Fi ist der Standard, der mit Abstand am meisten Stromleistung benötigt. Dafür kann es einen hohen Datenverkehr bei niedriger Latenz handeln und die Verbindung auch bei einer größeren Reichweite aufrecht erhalten.

## 2.3 SmartHome Produkte

### 2.3.1 Homematic

Homematic ist ein Produkt der Firma eQ-3.

funk- und kabelgebundene Kommunikation möglich, auch miteinander (Advanced Routing)  
eigene App für die Einrichtung und Steuerung; Steuerung auch mit Sprachassistenten (Alexa und Google Assistant möglich)

Protokoll IPv6; 868 MHz Frequenz -> Kommunikation über Funk (resistent gegen Internetausfall)

Redundanz (wenn mehrere Access Points) VDE- und AV-Test-geprüft; Verschlüsselungsstandard AES128; Protokoll Bidirectional Communication Standard (BidCoS); nur berechtigte Sensoren können Befehle an Geräte senden (Challenge-Response-Authentifizierungsverfahren)

Shelly

Bosch Smart Home

Eve Home

Aquara

Tab. 2.1: Smart-Home Systems

<b>Homematic</b>	
<b>Kommunikation</b>	Funk und Kabel
<b>Einrichtung</b>	
<b>Sicherheit</b>	AES128

---

### 3 Theoretische Grundlagen

#### 3.1 Sensorenauswahl

Zu Beginn der Arbeit wurde anhand der Aufgabenstellung eine Vorauswahl für die Arten der zu verwendeten Sensortypen des Multisensors herausgesucht. Diese sind ein Temperatur-, ein Feuchtigkeits-, ein Luftqualitätssensor, sowie ein Standardpräsenzmelder.

Tab. 3.1: Multisensor (Temperatur- und Feuchtigkeitssensor)

<b>Bezeichnung</b>	<b>Interface</b>	<b>Betriebsspannung</b>	<b>Messbereich</b>	<b>Genauigkeit</b>	<b>Kosten</b>
IST Sensor Feuchte- und Temperatur-Sensor HYT 271 [6]	I2C	2,7V bis 5,5V	0 bis 100%rF, -40 bis 125°C	±1,8%rF, ±0,2°C	36,99€
Amphenol Digital Temperatur- und Feuchtigkeits- sensor CC2D23-SIP [7]	I2C	2,3V bis 5,5V	0 bis 100%RH, -40 bis 125°C	±2%rF, ±0,3°C	13,80€

Tab. 3.2: Luftqualitätssensor

<b>Bezeichnung</b>	<b>Interface</b>	<b>Betriebsspannung</b>	<b>Messung</b>	<b>Kosten</b>
Joy-it SEN-CCS811V1 Sensor-Modul [8]	I2C	3V bis 5V	0 - 1187 Teile/Milliarde (TVOCs) 400 - 8192 Teile/Million (eCO2)	36,99€

Tab. 3.3: Standardpräsenzmelder

<b>Bezeichnung</b>	<b>Interface</b>	<b>Betriebsspannung</b>	<b>Messbereich</b>	<b>Kosten</b>
Omron D6T1A01 Thermischer MEMS-Präsenzsensor [9]	I2C	4,5V bis 5,5V (3,5mA)	Kreisdurchmesser: 222cm (2m Höhe)	13,95€
Omron D6T-44L-06H Wärmesensor IC-Näherungssensor [10]	I2C	4,5V bis 5,5V (5mA)	162cm x 169cm (2m Höhe)	49,78€

Anschließend wurden, durch Auswahl des Betreuers, die Sensoren IST HYT 271, Joy-it SEN-CCS811V1 und Omron D6T1A01, als die zu verwendeten Komponenten bestimmt, welche im Anschluss zum I2C-Kapitel noch einmal genauer dargestellt werden.

## 3.2 Grundlagen von I2C

Das I2C-Interface ist ein Bussystem, das auf der grundlegenden Master-Slave-Beziehung funktioniert. D.h. dass ein Master (z.B. ein Mikrocontroller) und viele Slaves (z.B. Sensoren) existieren. Hierbei bestimmt der Master die Art der Datenübertragung, ob dieser Daten an die Slaves sendet oder diese von ihnen empfängt. Der I2C-Bus ist ein synchroner serieller Zweidrahtbus, was für die Datenübertragung bedeutet, dass es eine Daten- und eine Taktleitung gibt (SDA und SCL). Zusätzlich wird noch eine GND-Leitung benötigt.

Die Datenübertragung erfolgt über ein I2C-Protokoll, welchem alle I2C-fähige Komponenten unterliegen. Dar grundlegende Aufbau des Protokolls lässt sich in 7 Teile gliedern:

Bei der Startbedingung wird ein Aufwecksignal vom Master an alle Slaves geschickt, wodurch die Slaves auf die darauf folgenden weitere Teile aufmerksam gemacht werden.

Anschließend folgt die 7-Bit Adresse des Slaves, mit welchem der Master die Datenübertragung durchführen möchte.

Als nächstes folgt ein Read- oder Write-Bit, mit welchem angegeben wird, ob der Slave vom Master Informationen empfängt (liest) oder im Informationen bereitstellt (schreibt).

Auf dieses folgt ein anschließendes ACK-Bit des Slaves.

Aufgrund der R/W-Unterscheidung entscheidet sich auch im Folgendem welcher von beiden die Daten sendet. Diese Daten werden in Paketen von 8 Bit, also einem Byte, versendet.

Jedes gesendete Datenpaket wird mit einem ACK-Bit vom Gegenspieler bestätigt, es sei denn dieser möchte die Verbindung. Hier wird dann ein negiertes ACK-Bit gesendet.

Abschließend beendet der Master die Datenübertragung mit einer Stoppbedingung. [11, 12]

### 3.3 Sensorenbeschreibung

#### 3.3.1 IST HYT 271



Abb. 3.1: IST HYT 271

Der HYT 271 der IST-Familie ist Temperatur- und Feuchtigkeitssensor. Diese generieren anhand des polymerbasierten Sensorelements Daten, welche durch Umrechnung in relative Feuchtigkeits- und Temperaturwerte umgerechnet werden können.

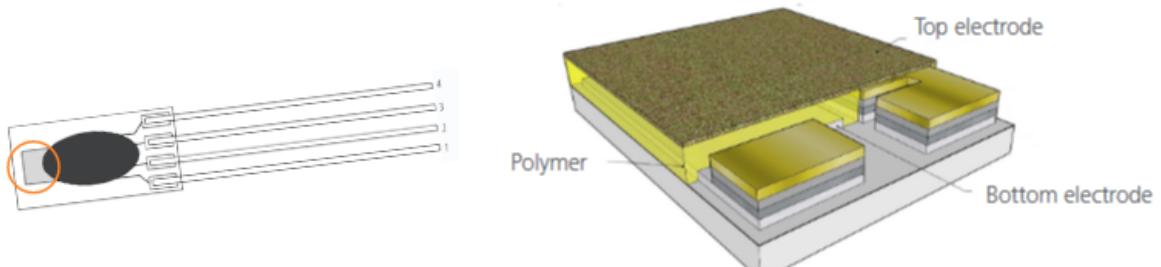


Abb. 3.2: aktiver Sensor des HYT271

Abb. 3.3: Aufbau des Sensors

Dieser Sensor beruht auf dem beschriebenen Protokoll im I2C-Kapitel 3.2. Der Sensor hat 2 verschiedene Kommandos: DF = Data Fetch und MR = Measurement Request. Bei beiden Kommandos ist der Anfang des Protokolls gleich: Zuerst wird das Start-Bit übermittelt, gefolgt von der 7 Bit Geräteadresse. Danach folgt das Write/Read-Bit, welches beim Measurement Request als Write und beim Data Fetch als Read ausgeführt wird. Nach dem Write/Read-Bit folgt ein ACK-Bit des Sensors. Beim MR schließt das Stopp-Bit das Protokoll ab, wohingegen beim DF jetzt erst die Daten vom Sensor übermittelt werden. Dabei werden in 4 Byte-Blöcken die Luftfeuchtigkeitswerte (1. und 2. Byte) und die Temperaturwerte (3. und 4. Byte) gesendet. Nach den ersten 3 Bytes folgt jeweils ein ACK-Bit des Masters. Nach dem 4. Byte folgt ein NACK-Bit mit dem anschließenden Stopp-Bit.

Um nun aus diesen Bytes reale Datenwerte generieren zu können, müssen zuerst die Luft-

### 3 Theoretische Grundlagen

---



Abb. 3.4: HYT271 Measurement Request Protokoll

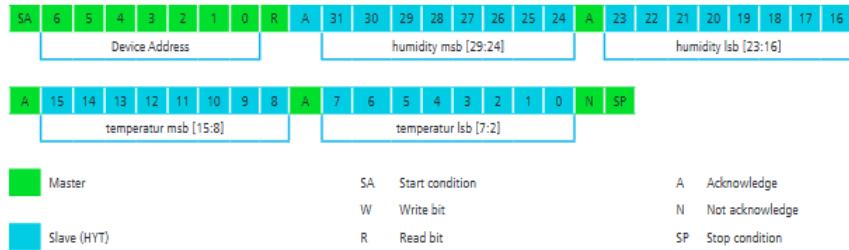


Abb. 3.5: HYT271 Data Fetch Protokoll

feuchtigkeitsblöcke, 1 und 2, sowie die Temperaturblöcke, 3 und 4, zusammengeführt werden. Bei den 16 Bits der Luftfeuchtigkeit muss man nun die ersten 2 Bits, die Bits die am weitesten links liegen, zu Nullen maskieren und diese 14 Bit dann in eine Dezimalzahl umwandeln. Abschließend wird diese Zahl mit 100 multipliziert und durch 16383 geteilt. Somit erhält man den Luftfeuchtigkeitswert in der Einheit %RH. Dabei kann die Luftfeuchtigkeit die Werte 0-100%RH annehmen. Der dezimale Wert 0 entspricht auch 0%RH und der dezimale Wert 16383 (hexadezimal: 3FFF) entspricht 100%RH.

Um den Temperaturwert aus den Byteblöcken 3 und 4 zu erhalten, muss man diese ebenfalls von 16 Bit auf 14 Bit verkleinern. Hier werden die letzten 2 Bits, die Bits die am weitesten rechts liegen, gestrichen. Die aus den 14 Bit entstandene Dezimalzahl muss noch mit dem Faktor 165/16383 multipliziert werden und abschließend wird hier noch der Wert 40 abgezogen. Somit erhält man den Temperaturwert in °C. Dieser kann die Werte im Bereich von -40°C bis zu 125°C annehmen. Der dezimale Wert 0 entspricht -40°C und der dezimale Wert 16383 (hexadezimal: 3FFF) entspricht 125°C.

[13]

### 3 Theoretische Grundlagen

---

Example:

	Byte 1	Byte 2	Byte 3	Byte 4
	31 dec	109 dec	96 dec	72 dec
bin	0001.1111	0110.1101	0110.0000	0100.1000
	Humidity 14 bit right-adjusted		Temperature 14 bit left-adjusted	
hex	1F6D		1812	
dec	8045 x 100/16383 =		6162 x 165/16383 - 40 =	
	49.1 %RH		22.06 °C	

Abb. 3.6: HYT271 Umrechnung in reale Werte

#### 3.3.2 Joy-it SEN-CCS811V1

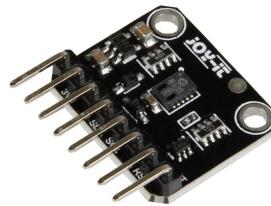


Abb. 3.7: Joy-it SEN-CCS811V1

Der Joy-it SEN-CCS811V1 ist ein Gassensor, mit welchem die Qualität der Luft gemessen werden kann. Dabei kann man die flüchtigen organischen Verbindungen (VOC) messen, welche man als TVOC oder eCO<sub>2</sub> ausgeben lassen kann. Dieser Sensor beruht ebenfalls auf dem I2C-Protokoll, welches im vorherigen Kapitel beschrieben wurde.

Hier werden ebenfalls eine Write- und anschließend eine Read-Operation, wie beim HYT271 durchgeführt. Der Aufbau der Operationen ist zu Beginn wieder gleich: Sie starten mit einem Start-Bit, gefolgt von der Geräteadresse. Anschließend folgt wieder das Write/Read-Bit, welches dazu führt, dass die beiden Operationen im folgenden, nach dem folgenden ACK-Bit des Slaves, unterschiedlich ablaufen. Bei der Write Operation folgt die Registeradresse, auf welche wieder ein ACK des Slaves und abschließend noch das Stopp-Bit folgt. Die Read Operation lässt nach dem ACK-Bit die Daten von der gegebenen Registeradresse. Abschließend folgt ein NACK des Masters und das Stopp-Bit. Mit der Registeradresse wird festgelegt, aus welchem Register die Daten der Read Operation gelesen werden. Um die TVOC und eCO<sub>2</sub> Daten auszulesen müssen die Daten aus dem ALG\_RESULT\_DATA Register mit der Adresse 0x02 ausgelesen werden. Das Register besteht aus 8 Bytes, wobei Byte 0 und 1 den eCO<sub>2</sub>-Wert und Byte 2 und 3 den TVOC-Wert beinhalten.

### 3 Theoretische Grundlagen

---

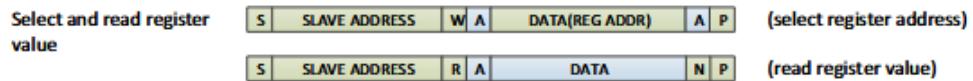


Abb. 3.8: CCS811 Protokoll

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6 & 7
eCO <sub>2</sub> High Byte	eCO <sub>2</sub> Low Byte	TVOC High Byte	TVOC Low Byte	STATUS	ERROR_ID	See RAW_DATA

Abb. 3.9: CCS811 Aufbau des Algorithm Results Registers

Der eCO<sub>2</sub>-Wert kann Werte im Bereich von 400 ppm bis zu 8192 ppm annehmen. Der TVOC-Wert hingegen kann Werte im Bereich von 0 ppb bis zu 1187 ppb annehmen.

[14]

### 3.3.3 Omron D6T1A01



Abb. 3.10: Omron D6T1A01

Der Omron D6T1A01 gehört zur Gruppe der D6T Thermalsensoren. Alle D6T Thermalsensoren bestehen aus einer Platine, einer Siliziumlinse, einem Thermopile-Sensor und einem speziellen Adapter. Dabei bündelt die Siliziumlinse die infrarote Strahlung, welche von umliegenden Objekten abgestrahlt wird, und bündelt diese auf den unter ihr liegenden Thermopile-Sensor. Dieser wandelt diese infrarote Strahlung dann in elektromotorische Kraft um, aus welcher dann im folgenden die Temperaturwerte der Objekte berechnet werden. Diese Werte können dann über den Adapter via I2C erhalten werden.

Das I2C-Protokoll ist bei der Gruppe der D6T Thermalsensoren unterschiedlich. Bei dem ausgewählten Typ, dem D6T1A01 beginnt das Protokoll wie gewöhnlich mit dem Start-Bit und der Geräteadresse des Sensors. Anschließend folgt das Write-Bit, welches im Folgenden durch ein ACK des Slaves bestätigt wird. Danach wird das Register angegeben, welches für die spätere Read-Operation ausgelesen werden soll. Beim D6T1A01 ist dieses das 0x4C. Als nächster Teil des Protokolls steht die Repeat Start Condition (Sr) an. Durch diese kann eine Write- und Read-Operation in einem Protokollablauf stattfinden. Der erwähnte Read-Befehl folgt nach der erneuten Geräteadresse mit dem dazugehörigen ACK. Nun stehen die angeforderten Daten in Form von 2 PTAT Byte (Low und High), 2 weitere Byte für das PO und ein Byte für das PEC zur Verfügung. Hinter jedem dieser Bytes wird ein ACK zur Empfangsbestätigung gesendet. Das Protokoll wird vom bekannten Stopp-Bit (NACK) abgeschlossen. Generell bestehen PTAT und PO aus 2 Byte. Um aus diesen Bytes einen realen Temperaturwert zu erhalten, müssen diese miteinander verrechnet werden:

$$PTAT = \frac{256 \cdot PTAT(Hi) + PTAT(Lo)}{10} \quad (3.1)$$

$$T0 = \frac{256 \cdot T0(Hi) + T0(Lo)}{10} \quad (3.2)$$

### 3 Theoretische Grundlagen

---

Das PEC-Byte dient lediglich zur Fehlererkennung.

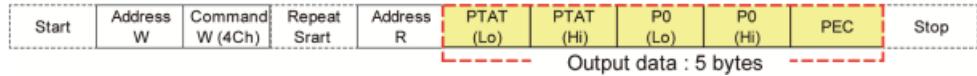
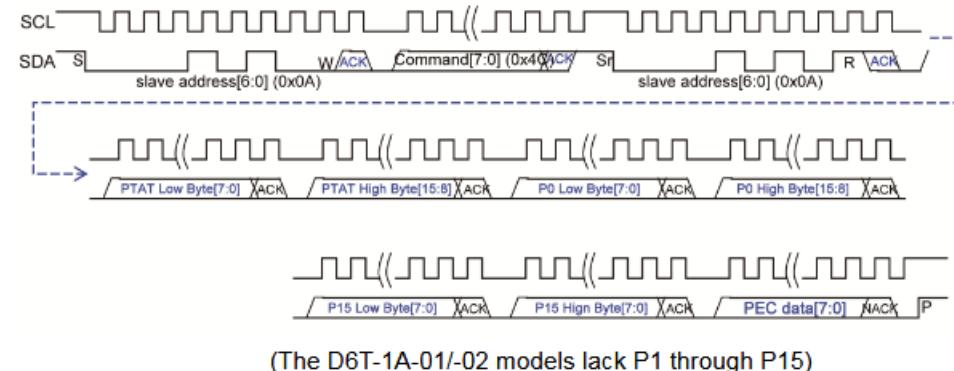


Abb. 3.11: D6T1A01 Protokoll

#### Signal Chart



(The D6T-1A-01/-02 models lack P1 through P15)

Fig. 16. Signal Terminal Flow (D6T-1A-01/-02/44L-06)

"S"	: Start Condition
"Sr"	: Repeat Start Condition
"P"	: Stop Condition
"W/R"	: Write (Lo) / Read (Hi)
"ACK"	: Acknowledge reply
"NACK"	: No-acknowledge reply

\* Refer to the I<sup>2</sup>C bus specifications for the definitions of these I<sup>2</sup>C terms.

Abb. 3.12: D6T1A01 Signalverlauf

[15, 16]

### 3.4 MQTT

Das MQTT-Modell ist ein Client-Server-Protokoll, bei welchem bidirektional, d.h. vom Client zum Server und vom Server zum Client, kommuniziert werden kann. Das Modell besteht aus 2 verschiedenen Gerätearten, dem Client und dem Broker. Diese Modell basiert auf Themen (Topics), welche die Clients abonnieren können(to subscribe). Clients können beide Rollen, die des Empfängers und die des Senders, erfüllen. Eine Art Mittelsmann ist dabei der Broker, welcher Daten zu einem Thema empfängt und diese dann für andere Clients, die dieses Thema abonniert haben, zur Verfügung stellt (Publish).

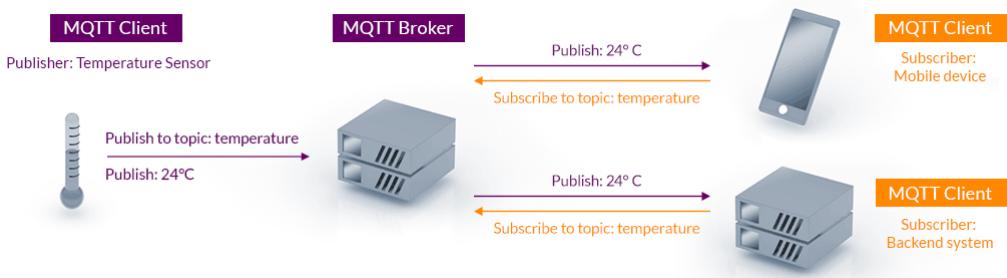


Abb. 3.13: MQTT Modell

Im MQTT System sind 3 QoS-Level definiert. Diese unterscheiden sich nach der Zuverlässigkeit über die Ankunft der gesendeten Nachricht. Level 0, at most once, ist nicht sehr zuverlässig, da hier nur die Nachricht versendet wird und nicht geprüft wird, ob diese auch vom Broker empfangen wurde. Bei Level 1, at least once, wird vom Client die Nachricht solange immer wieder verschickt, bis ein ACK vom Broker bei Client wieder ankommt. Hierbei ist zwar gesichert, dass der Datenaustausch funktioniert hat, jedoch wird die Nachricht vielleicht öfter, als dies benötigt ist, verschickt. Bei Level 2, exactly once, wird während der Kommunikation ein Handshake zwischen Client und Server ausgeführt. Hierbei können Client und Server anhand der Handshakeprozedur sicher sein, dass die Nachricht tatsächlich angekommen ist.

[17, 18]

### 3.5 Matter

Matter ist ein Verbindungsstandard, welcher über eine lokale Verbindung

im Heimnetzwerk funktioniert. Ein großer Vorteil dieses Verbindungsstandards ist nicht nur die Vereinigung vieler verschiedener Standards, wie z.B. Amazon, Google oder anderer SmartHome-Assistenten, sondern die, im selben Zug erreichte, Trennung von der Cloud hin zur **lokalen**

**Kommunikation** ohne das Internet. Dazu wird nur ein matter-kompatibles Gerät benötigt, welches für Apps und SmartHome-Steuerzentralen erreichbar ist. [19] Das grundlegende **Sicherheitsprinzip** bei Matter gleicht dem aus dem Internet. Dieses wird Public Key Infrastructure-Verfahren (PKI) genannt. Anhand von gültigen Zertifikaten, die durch die Hersteller vergeben werden (DAC), werden Verbindungen aufgebaut. Bei diesen ist theoretisch das Aufzeichnen durch Dritte möglich, jedoch können diese das Aufgezeichnete, aufgrund des digitalen Schlüssels, nicht entschlüsseln. Wird ein Gerät in Betrieb genommen sendet dieses automatisch Signale aus, mit welchen es anzeigt, dass es ins Netzwerk aufgenommen werden möchte. Der Controller startet bei Erhalt des Signals einen PASE-Prozess mit dem neuen Gerät, welcher durch das gerätespezifische Passwort / QR-Code als sicher gilt.

Nach Prüfung des DAC erhält das Gerät einen digitalen Ausweis, das OpCert, ein Stammzertifikat (Root Certificat), den digitalen Schlüssel und eine Liste aller Funktionen (ACL). Bei gleichem Stammzertifikat vertrauen sich die Geräte gegenseitig und die ACL gibt dem Gerät an, welche anderen Geräte es steuern oder von welchen es gesteuert werden darf. Des Weiteren gibt es ein digitales Register, das Distributed Compliance Ledger (DCL) oder auch übersetzt verteiltes Hauptbuch für Konformität. Es ist ein Netzwerk aus unabhängigen Servern, die entweder unternehmensintern oder durch das CSA (Connectivity Standard Allliance) betrieben werden. Ein wichtiger Punkt hierbei ist die Dezentralisierung, die vor Angriffen auf das Netzwerk schützt und die Kommunikation untereinander, die über ein kryptografisch gesichertes Protokoll abläuft. In dem Register werden Daten, wie z.B. Informationen zum Anbieter, Liste der Stammzertifikate, IDs, Informationen zur aktuellen Softwareversion, etc., gespeichert. Um ein Update für ein Produkt bereitzustellen oder sogar ein neues Gerät auf den Markt zu bringen, gibt es spezielle Prüfinstitute. Bei erfolgreichem Zertifizierungsprozess, durch das Institut, wird die CSA benachrichtigt, die das Produkt oder das Update dann als zertifiziert eintragen kann. Anhand dieses Statuses können die verschiedenen Matter-Ökosystem (z.B. Android, Apple, etc.) dann bedenkenlos das Gerät zu ihrem System hinzufügen. [20] Ein weiterer Vorteil von Matter nennt sich **Multi Admin**. Multi Admin bedeutet, dass ein Gerät in verschiedenen Matter Fabrics (nach Standard: mindestens 5 gleichzeitig möglich) installiert werden kann. Das heißt z.B., dass nicht jede Person in einem Haushalt das selbe Ökosystem benutzen muss. Ein Licht oder eine Steckdose kann somit über mehrere verschiedene Systeme gesteuert werden. [21] Das, im Kontext von Multi Admin erwähnte, **Matter Fabric** ist ein aufgespanntes Mesh-Netz von Matter Nodes. Eine oder auch mehrere (bei Unterfunktionen eines Gerätes) Nodes

sind physikalische Geräte, die das Mesh-Netz bilden. Durch Erhalt eines eindeutigen Sicherheitszertifikates, das nur für dieses eine Fabric gilt, wird eine Node in dieses aufgenommen. Dieses Zertifikat erhält das Node durch den Commissioner. Ein **Matter Commissioner** richtet die neue Node ein bzw. gibt ihr Informationen mit und weißt sie ins neue Fabric ein. Der Commissioner ist oft mit dem Matter Controller in einem Gerät kombiniert. Ein **Matter Controller** ist für die Steuerung installierter Geräte zuständig. Ein Controller kann vieles sein, z.B. eine App, ein Smart-TV oder auch spezielle SmartHome-Zentralen. Dabei muss es nicht nur einen Controller geben, sondern man kann von mehreren Geräten aus in einem Matter-Fabric die Nodes ansteuern bzw. kontrollieren. [22, 23]

#### 3.5.1 Weitere Begriffe

Das Konzept des **Matter Binding** ist ein großer Schritt zur Automatisierung im Smart Home Bereich, speziell in der Kategorie der Funkkommunikation. Hierbei gehen Geräte wie z.B. ein Temperatursensor und eine Heizung eine Verbindung ein, durch welche die Heizung die Werte des Sensors direkt durch eine Art Abonnementmodell (siehe MQTT) erhält. Anhand von zuvor definierten Richtwerten zur Temperatur in einem Raum, könnte sich die Heizung somit selbst regeln. Leider haben die meisten Ökosysteme bisher noch keine Bindings in ihren Systemen vorgesehen oder es fehlt an Bedienelementen mit geeigneten Einstellungen für die entsprechenden Bindings. [24]

Der Begriff **Matter Casting** stammt von Amazon. Amazon benutzt ihn als Synonym für Medien-Streaming mit Prime Video. Man kann das Smartphone durch Drücken des Casting Symbols einfach mit dem Endgerät verbinden. Dabei kann man sich Infos zur aktuellen Serie / Film über das Smartphone anzeigen lassen oder dies einfach als eine Art von Fernbedienung benutzen. Man kann sogar die App auf dem Smartphone nach Starten der Wiedergabe schließen, da das Endgerät mit dem Internet verbunden ist. Beim Thema Matter Casting wird außerdem sehr zukunftsweisend gedacht. Es soll möglich sein nicht nur Videos, sondern auch noch Benachrichtigungen, wie z.B. dass die Waschmaschine fertig oder dass der Feuermelder ausgelöst hat, anzeigen zu lassen. [25]

Ein weiteres, bisher noch nicht erwähntes Gerät, ist die **Matter Bridge**. Sie hat die Aufgabe Geräte, die nicht dem Matter Standard entsprechen, in das Matter-Netzwerk einzubinden. Die Matter Bridge ist also eine Art Übersetzer, der die fremden Systeme matterfähig macht. [26]

### 3.5.2 Thread-Protokoll

Thread ist ein Funkprotokoll, welches im Matter-Standard als Alternative zum WLAN vorgesehen ist. Dabei bietet es zwei Vorteile: Das Protokoll benötigt wenig Energie zum Empfangen und Versenden von Daten und ist somit sehr stromsparend. Des Weiteren ist Thread Mesh-fähig. Das bedeutet, dass die mit der Stromleitung verbundenen Geräte als Repeater fungieren und sie somit das Netzwerk flächenmäßig vergrößern, wodurch auch weiter entferntere Geräte eingebunden werden können. Die mit der Stromleitung verbundenen Geräte werden im Thread-Netzwerk als **Router** bezeichnet. Sie machen das Mesh-Netzwerk robuster und stabiler. Beispiele für Router sind z.B. Lampen oder Zwischenstecker an Steckdosen. Neben den **Endgeräten** gibt es noch als 3. Gerätekategorie den **Border-Router**. Vom diesem Gerätetyp wird in jedem Thread-Netzwerk immer mindestens ein Exemplar benötigt. Sie sind die Verbindung zum IP-Netzwerk, wodurch das Thread-Netzwerk dann via Internet oder über Steuerung per Smartphone zugänglich wird. Die Router und Endgeräte erhalten eine IP-Adresse (IPv6), was sie leichter adressierbar macht. [27, 28]

---

## 4 Durchführung

### 4.1 Aufbau auf dem Breadboard

Bevor man mit dem eigentlichen Programmcode beginnen kann, muss man zuerst die Hardware aufbauen und korrekt anschließen. Hierfür standen zusätzlich, neben den Sensoren und dem ESP32-C6 Board, ein Breadboard, Jumper-Wires und verschiedene Widerstände bereit. Zuerst wurde der ESP Controller auf das Board gesteckt, wobei zu beachten war, dass jeweils auf beiden Seiten des Boards, an den Anschlusspinns eine Reihe Platz gelassen werden musste, um mögliche Anschlüsse zu legen.

Anhand des Layouts [29] kann man den Grundaufbau und die speziellen Funktionen der einzelnen Pins erkennen. Da es sich bei jedem der 3 verwendeten Sensoren um I2C-fähige Sensoren handelt, liegt es nahe eine 'SDA-Kette' und eine 'SCL-Kette' mit den Sensoren zu bilden.

Als SDA-Pin wurde der GPIO-Pin 18 und als SCL-Pin wurde der GPIO-Pin 19 gewählt. Von diesen Pins liegt jeweils eine Leitung zu einer neuen Breadboard-Reihe, in welcher die jeweiligen Sensoren der entsprechenden Leitungsart angeschlossen sind (I2C-Bussystem). Dabei ist zu beachten, dass noch 2,2 kOhm Widerstände zwischen den Reihen und der VCC-Verbindung des HYT 271 und des Joy-It CCS811 liegen, welche als Pull-up Widerstände fungieren. Zudem liegen von dem Ground-, dem 3V- und dem 5V-Pin diverse Leitungen zur Spannungsversorgung. Die 3V-Spannungsversorgung wird für den HYT 271 und den Joy-It CCS811 benötigt, während die 5V-Spannungsversorgung für den Omron D6T1A benötigt wird.

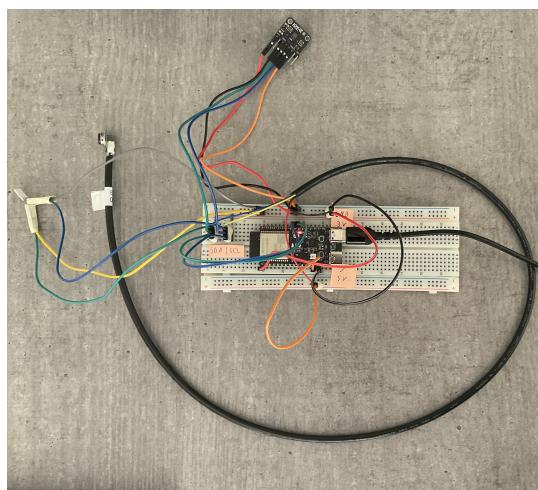


Abb. 4.1: Hardwareaufbau

## 4.2 Einrichtung VS Code mit der ESP-IDF

Für die grundlegende Umgebung zum Schreiben des Programms wurde VS Code und die zum ESP-32 benötigte Software ESP-IDF benutzt. Diese wurde mithilfe der offiziellen Webseite des ESP32 von Espressif durchgeführt. [30]

Zuerst wurde das WSL-System (Windows Subsystem for Linux) aufgesetzt. Hierzu wurde die offizielle Seite von Espressif genutzt. Dabei ist zu erwähnen, dass die Version Ubuntu-22.04 benutzt wurde. [31] Ebenso wurden die grundlegenden Schritte des Standard Toolchain Setups für diese WSL durchgeführt. [32] Um nun das erforderliche Setup zu komplettieren, wurde ESP-Matter konfiguriert. [33]

## 4.3 Matter Sensors

Als Grundgerüst des Projekts wurde die Vorlage aus dem Esp-Matter Ordner 'esp-matter/examples/sensors' [?](<https://github.com/espressif/esp-matter/tree/main/examples/sensors>) benutzt. Das Sensor-Projekt zielte darauf ab 2 Sensoren, einen SHTC3- und einen PIR-Sensor einzubinden. Im Folgenden werden die Änderungen am Projekt dargelegt und die wichtigsten Bausteine erläutert.

### 4.3.1 Aufbau

Der anfängliche Aufbau des Projekts wird in der folgenden Abbildung kurz dargestellt:

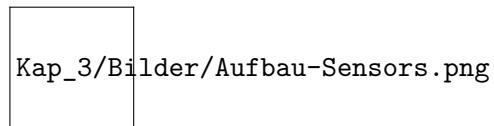


Abb. 4.2: Sensors-Struktur

In den grundlegenden Dateien, wie z.B. CMakeLists, der idf\_component oder der app\_openthread\_config wurden keine Änderungen vorgenommen.

### 4.3.2 Drivers

Da der in der Vorlage verwendete Sensor SHTC3 über ein I2C-Protokoll kommuniziert, wurde sich im Folgenden, bei dem Erstellen der neuen Dateien, an den beiden bestehenden Dateien shtc3.cpp und shct3.h orientiert. In der neuen .h-Datei werden dann zuerst 3 'Callbacks' für die

3 Sensoren erstellt. Im Anschluss werden die Strukturen der Sensoren mit ihren jeweiligen Parametern, wie z.B. beim HYT271 Luftfeuchtigkeit und Temperatur, definiert. Abschließend werden die 3 'init' Funktionen, die in der .cpp-Datei definiert sind, noch deklariert.

In dieser wurde zunächst nichts geändert: die Strukturen aus der .h-Datei sind um einen Timer erweitert. Des Weiteren wird der I2C-Master mit der 'i2c\_config\_t' konfiguriert. Die weitere Struktur der Datei beinhaltet Funktionen für die Sensoren, die verschachtelt aufgerufen werden. Diese Funktionen sind wie folgt aufgebaut:

```
sensor_init -> timer_cb_internal -> convert -> read
```

Die read-Funktion orientiert sich stark am I2C-Protokoll des jeweiligen Sensors. Der Aufbau des Codes wurde somit mit den Datenblättern der Sensoren (Kapitel 2.3) und der Anleitung zum ESP32-C6 für I2C[34] geschrieben. In der convert-Funktion wird dann die read-Funktion ausgeführt und die ausgelesenen Daten des Sensors in einem array gespeichert. Diese Daten müssen dann anhand einer sensorspezifischen Umrechnung, welche wieder aus den Datenblättern hervorgeht, in das entsprechende Format der gemessenen Variable konvertiert werden. Im timer\_cb\_internal werden zum einen ein Zeiger (arg) erstellt, der in einen \_cxt\_t \*-Typ umgewandelt wird und somit auf Konfiguration zugreifen kann. Des Weiteren wird ein Callback definiert, welchem die 3 Argumente, endpoint\_id, der jeweilige errechnete Wert und die user.data, zugewiesen werden. In der letzten Funktion, der init, wird der Sensor initialisiert und die periodische Abfrage gestartet. Hierzu muss zuerst die I2C-Schnittstelle initialisiert werden. Das Argument der Funktion wird unter der .config des Sensors abgespeichert. Im Anschluss wird der Timer erstellt. Dabei wird die timer\_cb\_internal-Funktion dem .callback zugewiesen, also immer dann ausgeführt wenn der Timer ausläuft. Zudem wird ein Zeiger auf den Sensor als Argument übergeben, sodass die Callback-Funktion auf die Konfiguration des Sensors zugreifen kann. Des Weiteren gibt es noch, spezifisch für den CCS811, eine write-Funktion, mit der im entsprechenden sensor\_init der Modus gesetzt und der APP\_START ausgeführt werden müssen.

### 4.3.3 app\_main

Die app\_main besitzt 5 'sensor\_notification'-Funktionen, Temperatur, Luftfeuchtigkeit, eCO<sub>2</sub>, TVOC und Anwesenheit. Bei diesen werden jeweils die 3 Argumente aus der timer\_cb\_internal-Funktion benötigt.

Das Attribut 'MeasuredValue' des jeweiligen Clusters wird im entsprechenden Endpunkt gespeichert. Nur bei der Anwesenheitsbenachrichtigung ist der 'MeasuredValue' durch 'Occupancy' ersetzt, da die Anwesenheit in einem bool mit wahr oder falsch gespeichert wird.

In der 'extern 'C' void app\_main' wird zuerst das NVS (Non-Volatile Storage) initialisiert. Der oben genannte (Hardware-)Reset-Button wird ebenfalls registriert. Folgend wird eine zentrale Node und 5 weitere Nodes für die 5, oben schon genannten, Variablen erstellt. Bei diesen ist der Aufbau im grundlegenden der gleiche: eine entsprechende config (esp\_endpoint-Dateien) wird verwendet um danach einen Endpunkt zu erstellen. Im Weiteren werden die 3 Sensortreiber mit den eben angesprochenen Endpunkten und der oben genannten Benachrichtigung initialisiert. Abschließend wird Matter gestartet.

### 4.4 PCB

Aus dem Breadboardaufbau (3.1), welcher vor allem für Testzwecke sehr gut geeignet ist, soll eine PCB entstehen. Diese ist dann im Vergleich zum Testaufbau strukturierter und geeigneter für einen realitätsnahen Einsatz. Ebenso kann aufgrund kürzerer Leiterbahnen im Vergleich zu den Jumper-Wires der Energieverbrauch, wenn auch nur leicht, optimiert werden. Zudem sind die Leitungsverbindungen aufgrund des erstellten Plans und der Label gut nachvollziehbar.

Zur Erstellung einer PCB gehören mehrere Schritte. Im 1. Schritt wird der grundlegender Plan mit den Komponenten erstellt, auf dem diese durch Anschlusslabel oder auch direkt miteinander verbunden werden. Dabei ist darauf zu achten, dass im Sinne der zukünftigen Erweiterbarkeit der Platine wenige Anschlüsse offen bleiben. Aus diesem Grund werden sogenannte Pin Header verwendet, mit welchem freie Anschlüsse verbunden werden können. Zu jeder Komponente muss außerdem ein Footprint, welcher die genauen Abmaße des Bauteils, vor allem die Abstände und Dimensionen zwischen den einzelnen Pads oder Durchgangslöchern, beschreibt. Anhand dieser Footprints und der angegebenen Verbindungen im zuvor erstellten Schaltplan kann dann ein Entwurf der Leiterplatine erstellt werden. Wichtig zu beachten ist hierbei die Anzahl an verfügbaren Layern. In diesem Fall wurde eine 2 Layer PCB entworfen.

### 4.5 SmartHome Server

installation von VirtualBox mit dem Extension Pack (jeweils Version 7.1.8)

---

## **5 Ergebnisse und Auswertung**

hallo

---

## **6 Schlussfolgerung und Ausblick**

---

# 7 Kapitel Beispiel

## 7.1 Kapitel bsp.1

Diese Vorlage entspricht den neuen Regeln des Brand Designs der RPTU. Hierzu wurde die Vorlage von <https://github.com/RPTU-EIT/report-eit-latex> modifiziert. Im Unterordner style-eit-latex ist die Datei EIT.sty hinterlegt. In dieser Style-Datei wurden die Farben und die Schriftart auf das RPTU-Design umgestellt. Aufgrund der notwendigen Verwendung bestimmter Packages wird diese Vorlage nur in LuaLaTeX unterstützt!

Folgende Schriftarten wurden als Standard integriert:

Tab. 7.1: RedHatText-Standardschriftart und ihre Formen

Form	Name der Schriftart	Befehl
normal	RedHatText-Regular	-
fett	<b>RedHatText-SemiBold</b>	\bfseries oder \textbf{}
kursiv	<i>RedHatText-Italic</i>	\itshape oder \textit{}
fett & kursiv	<b>RedHatText-SemiBoldItalic</b>	\bfseries\itshape oder \textbf{\textit{}}

Des Weiteren wurden die anderen verfügbaren RedHatText-Schriftarten integriert:

Tab. 7.2: Weitere RedHatText-Schriftarten und ihre Formen

Form	Name der Schriftart	Befehl
light	RedHatText-Light	\fontseries{li}\fontshape{n}\selectfont
light & kursiv	<i>RedHatText-Lightitalic</i>	\fontseries{li}\fontshape{it}\selectfont
medium	<b>RedHatText-Medium</b>	\fontseries{md}\fontshape{n}\selectfont
medium & kursiv	<b>RedHatText-MediumItalic</b>	\fontseries{md}\fontshape{it}\selectfont
extra fett	<b>RedHatText-Bold</b>	\fontseries{bf}\fontshape{n}\selectfont
extra fett & kursiv	<b>RedHatText-BoldItalic</b>	\fontseries{bf}\fontshape{it}\selectfont

## 7 Kapitel Beispiel

---

Die Schriftfarbe kann über den `\textcolor{<Farbe>}{<Text>}` Befehl eingestellt werden.  
Folgende Farben stehen zur Verfügung:

Tab. 7.3: RPTU-Farben

Farbe	Befehl
<b>blaugrau</b>	<code>\textcolor{rptublaugrau}{blaugrau}</code>
<b>gruengrau</b>	<code>\textcolor{rptugruengrau}{gruengrau}</code>
<b>dunkelblau</b>	<code>\textcolor{rptudunkelblau}{dunkelblau}</code>
<b>hellblau</b>	<code>\textcolor{rptuhellblau}{hellblau}</code>
<b>dunkelgruen</b>	<code>\textcolor{rptudunkelgruen}{dunkelgruen}</code>
<b>hellgruen</b>	<code>\textcolor{rptuhellgruen}{hellgruen}</code>
<b>violett</b>	<code>\textcolor{rptuviolett}{violett}</code>
<b>pink</b>	<code>\textcolor{rptupink}{pink}</code>
<b>rot</b>	<code>\textcolor{rpturot}{rot}</code>
<b>orange</b>	<code>\textcolor{rptuorange}{orange}</code>
<b>schwarz</b>	<code>\textcolor{rptuschwarz}{schwarz}</code>
<b>weiss</b>	<code>\textcolor{rptuweiss}{weiss}</code>

### 7.1.1 Formeln

Es empfiehlt sich, Formeln in der folgenden Form anzugeben, damit alle verwendeten Größen nachvollziehbar sind:

$$y = a_0 + a_1 x \quad (7.1)$$

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \quad (7.2)$$

$y$  Funktionswert

$x$  Datenpunkt

$a_i$  Koeffizienten

### 7.1.2 Grafiken

Grafiken können mittels `\includegraphics` eingefügt werden.



Abb. 7.1: RPTU Logo als .png

Alternativ können auch Vektordateien eingefügt werden. Grafiken aus Inkscape können auch als `.tex` Datei hinterlegt werden. Dazu wird eine Vektorgrafik aus Inkscape über die `pdf_tex` Option exportiert. Inkscape exportiert die Grafik als `.pdf` und hinterlegt den Text in einer `.pdf_tex` Datei. Letztere kann auch als `.tex` Datei in L<sup>A</sup>T<sub>E</sub>X eingebunden werden, wie das folgende Beispiel zeigen soll:

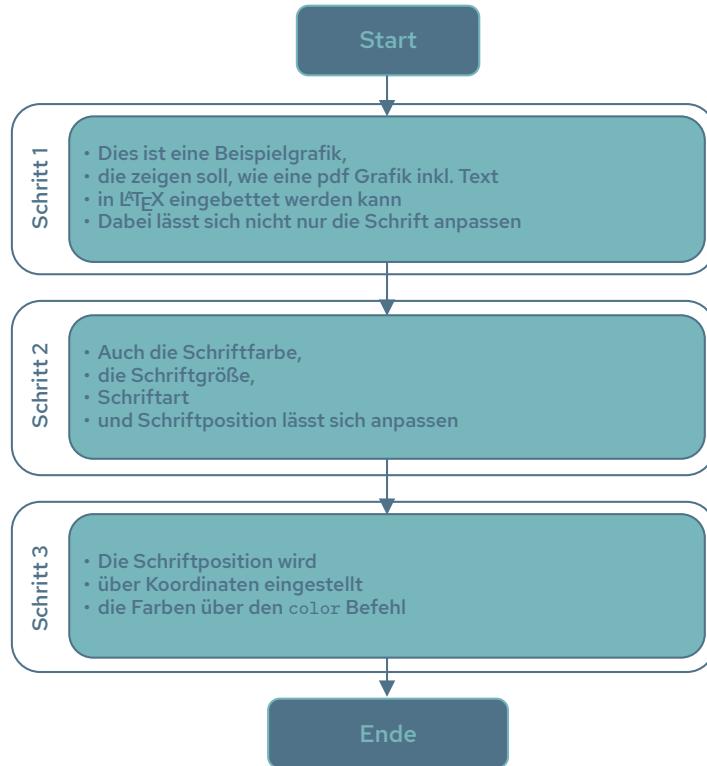


Abb. 7.2: Beispielgrafik mit eingebetteten Text

### 7.1.3 Literatur

Literatur wird in der Datei `Bibliografie.bib` hinterlegt. Als Zitierstil ist der IEEE-Stil voreingestellt. Zitiert wird mittels `\cite{}` Befehl. Dabei lassen sich Seitenangaben über die Optionen dieses Befehls realisieren. Zum Beispiel [35, S.416ff]: `\cite[S.416ff]{Brokate.2015}`

### 7.1.4 Deckblatt

Auf dem Deckblatt kann das entsprechende Lehrstuhllogo eingepflegt werden. Dazu muss das Logo im Unterordner `style-eit-latex/backgrounds` hinterlegt werden. Die Einstellungen zum Einfügen des Logos finden sich in der `Main.tex` in Zeile 35.

### 7.1.5 Übersicht

In der folgenden Tabelle werden die zuvor aufgelisteten Standards mit ihren wichtigsten Eigenschaften für den Smart-Home-Gebrauch noch einmal aufgelistet.

Tab. 7.4: Smart-Home Standards

	<b>Matter</b>	<b>Thread</b>	<b>ZigBee</b>	<b>Z-Wave</b>	<b>BLE</b>	<b>Wi-Fi</b>
<b>Stromverbrauch</b>			100mW	1mW	10mW	hoch
<b>Frequenz</b>			2,4GHz	868,42MHz	2,4GHz	2,4 / 5GHz
<b>Reichweite</b>			100m	30m	50m	1000m
<b>Kompatibilität</b>		gleicher Hersteller	verschiedene Hersteller möglich		BLE Kompatibilität	Wi-Fi Kompatibilität
<b>max. Nodeanzahl</b>			653356			

---

## Literaturverzeichnis

- [1] S. Safaric und K. Malaric, "Zigbee wireless standard," in *Proceedings ELMAR 2006*. IEEE, 2006, Seite 259–262.
- [2] P. Dhillon und H. Sadawarti, "A review paper on zigbee (ieee 802.15. 4) standard," *International journal of engineering research and technology*, Band 3, 2014.
- [3] admin, "Z-wave | z-wave europe - the leading european distributor for smart home products." [Online]. Verfügbar: <https://zwave.eu/de/zwave/> [Zugriff am: 2025-04-13]
- [4] M. R. Ghori, T.-C. Wan, und G. C. Sodhy, "Bluetooth low energy mesh networks: Survey of communication and security protocols," *Sensors*, Band 20, Nr. 12, Seite 3590, 2020.
- [5] "Bluetooth technologie Übersicht | bluetooth® technologie website." [Online]. Verfügbar: <https://www.bluetooth.com/de/learn-about-bluetooth/tech-overview/> [Zugriff am: 2025-04-10]
- [6] "IST Sensor Feuchte- und Temperatur-Sensor 1 St. HYT 271 Messbereich: 0 - 100 % rF (L x B x H) 10.2 x 5.1 x 1.8 mm kaufen." [Online]. Verfügbar: <https://www.conrad.de/de/p/ist-sensor-feuchte-und-temperatur-sensor-1-st-hyt-271-messbereich-0-100-rf-l-x-b-x-h-10-2-x-5-1-x-1-8-mm-505675.html> [Zugriff am: 2025-01-25]
- [7] "CC2D23-SIP | Amphenol Digital Temperatursensor und Feuchtigkeitssensor ±2% THT, 4-Pin, I2C -40 bis 125 °C. | RS." [Online]. Verfügbar: <https://de.rs-online.com/web/p/temperatur-und-luftfeuchtigkeit-sensoren/2105091> [Zugriff am: 2025-01-28]
- [8] "Joy-it SEN-CCS811V1 Sensor-Modul 1 St. kaufen." [Online]. Verfügbar: <https://www.conrad.de/de/p/joy-it-sen-ccs811v1-sensor-modul-1-st-1884871.html> [Zugriff am: 2025-01-25]
- [9] r. e. G. I. T. webmaster@reichelt.de, "OMRON Thermischer MEMS-Präsenzsensor, 1x1 Element, 5...50 °C | Bewegungs-/Präsenzsensoren günstig kaufen | reichelt elektronik." [Online]. Verfügbar: [https://www.reichelt.de/de/de/shop/produkt/thermischer\\_memspresenzsensor\\_1x1\\_element\\_5\\_50\\_c-293436](https://www.reichelt.de/de/de/shop/produkt/thermischer_memspresenzsensor_1x1_element_5_50_c-293436) [Zugriff am: 2025-01-25]

- [10] "D6T-44L-06H | Omron D6T Wärmesensor IC-Näherungssensor 3m, Modul 4-Pin | RS." [Online]. Verfügbar: <https://de.rs-online.com/web/p/naherungssensoren-ics/2043349> [Zugriff am: 2025-01-28]
- [11] R. Sathiyamoorthy, "I2c-bus," Klasse E4p, Embedded Control, Hr.Felser, HTI Burgdorf. [Online]. Verfügbar: [https://www.mikrocontroller.net/attachment/372171/I2C\\_bus.pdf](https://www.mikrocontroller.net/attachment/372171/I2C_bus.pdf) [Zugriff am: 2025-01-25]
- [12] "I2C-Bus - einfach und verständlich erklärt!" [Online]. Verfügbar: <http://fmh-studios.de/theorie/informationstechnik/i2c-bus/> [Zugriff am: 2025-01-28]
- [13] "Application note humidity modules hyt." [Online]. Verfügbar: [https://www.ist-ag.com/sites/default/files/downloads/AHHYTM\\_E.pdf](https://www.ist-ag.com/sites/default/files/downloads/AHHYTM_E.pdf) [Zugriff am: 2025-01-25]
- [14] "Ccs811\_datasheet-ds000459.pdf." [Online]. Verfügbar: [https://cdn.sparkfun.com/assets/learn\\_tutorials/1/4/3/CCS811\\_Datasheet-DS000459.pdf](https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf) [Zugriff am: 2025-01-25]
- [15] "Mems thermal sensors d6t user's manual." [Online]. Verfügbar: [https://cdn-reichelt.de/documents/datenblatt/B400/D6T\\_MANUAL-ENPDF.pdf](https://cdn-reichelt.de/documents/datenblatt/B400/D6T_MANUAL-ENPDF.pdf) [Zugriff am: 2025-01-25]
- [16] "Mems thermal sensors d6t." [Online]. Verfügbar: [https://cdn-reichelt.de/documents/datenblatt/B400/D6T\\_DB-EN.pdf](https://cdn-reichelt.de/documents/datenblatt/B400/D6T_DB-EN.pdf) [Zugriff am: 2025-01-25]
- [17] "MQTT - The Standard for IoT Messaging." [Online]. Verfügbar: <https://mqtt.org/> [Zugriff am: 2025-03-03]
- [18] C. de Schryver, "Vorlesung netzwerk- und bustechnik (nubt): T09 metagateways onlyoff-ice," 2024.
- [19] "Matter-Vorteile #1: Lokale Verbindung." [Online]. Verfügbar: <https://matter-smarthome.de/vorteile/matter-vorteile-1-die-lokale-verbindung/> [Zugriff am: 2025-03-04]
- [20] "Matter-Vorteile #4: Sicherheit und Verschlüsselung." [Online]. Verfügbar: <https://matter-smarthome.de/vorteile/matter-vorteile-4-sicherheit-verschluesselung/> [Zugriff am: 2025-03-10]
- [21] "Matter-Vorteile #5: Multi-Admin-Betrieb." [Online]. Verfügbar: <https://matter-smarthome.de/vorteile/matter-vorteile-5-multi-admin/> [Zugriff am: 2025-03-05]

- [22] F.-O. Grün, "Was ist ein Matter Fabric?" May 2023. [Online]. Verfügbar: <https://matter-smarthome.de/wissen/was-ist-ein-matter-fabric/> [Zugriff am: 2025-03-11]
- [23] --, "Was ist ein Matter Controller?" Jan. 2024. [Online]. Verfügbar: <https://matter-smarthome.de/wissen/was-ist-ein-matter-controller/> [Zugriff am: 2025-03-04]
- [24] --, "Was ist ein Matter Binding?" Sep. 2023. [Online]. Verfügbar: <https://matter-smarthome.de/wissen/was-ist-ein-binding-in-matter/> [Zugriff am: 2025-03-11]
- [25] --, "Was ist Matter Casting?" Jul. 2024. [Online]. Verfügbar: <https://matter-smarthome.de/wissen/was-ist-matter-casting/> [Zugriff am: 2025-03-11]
- [26] --, "Was ist eine Matter Bridge?" Jun. 2023. [Online]. Verfügbar: <https://matter-smarthome.de/wissen/was-ist-eine-matter-bridge/> [Zugriff am: 2025-03-04]
- [27] "Matter-Vorteile #2: das Funkprotokoll Thread." [Online]. Verfügbar: <https://matter-smarthome.de/vorteile/matter-vorteile-2-das-funkprotokoll-thread/> [Zugriff am: 2025-03-11]
- [28] F.-O. Grün, "Was ist ein Thread Border Router?" Jul. 2023. [Online]. Verfügbar: <https://matter-smarthome.de/wissen/was-ist-ein-thread-border-router/> [Zugriff am: 2025-03-11]
- [29] "esp32-c6-devkitc-1-pin-layout.png (1600×1120)." [Online]. Verfügbar: [https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32c6/\\_images/esp32-c6-devkitc-1-pin-layout.png](https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32c6/_images/esp32-c6-devkitc-1-pin-layout.png) [Zugriff am: 2025-01-25]
- [30] "vscode-esp-idf-extension/README.md at master · espressif/vscode-esp-idf-extension." [Online]. Verfügbar: <https://github.com/espressif/vscode-esp-idf-extension/blob/master/README.md> [Zugriff am: 2025-01-25]
- [31] "Using WSL in Windows - - - ESP-IDF Extension for VSCode latest documentation." [Online]. Verfügbar: <https://docs.espressif.com/projects/vscode-esp-idf-extension/en/latest/additionalfeatures/wsl.html> [Zugriff am: 2025-04-05]
- [32] "Standard toolchain setup for linux and macOS - ESP32-c6 - - ESP-IDF programming guide latest documentation." [Online]. Verfügbar: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32c6/get-started/linux-macos-setup.html> [Zugriff am: 2025-04-09]

- [33] "2. Developing with the SDK - ESP32-C6 - – Espressif's SDK for Matter latest documentation." [Online]. Verfügbar: <https://docs.espressif.com/projects/esp-matter/en/latest/esp32c6/developing.html#developing-your-product> [Zugriff am: 2025-04-04]
- [34] "Inter-integrated circuit (i2c) - ESP32-c6 - – ESP-IDF programming guide v5.1 documentation." [Online]. Verfügbar: <https://docs.espressif.com/projects/esp-idf/en/v5.1/esp32c6/api-reference/peripherals/i2c.html> [Zugriff am: 2025-04-14]
- [35] M. Brokate, N. Henze, F. Hettlich, A. Meister, G. Schranz-Kirlinger, T. Sonar, und D. Rademacher, *Grundwissen Mathematikstudium: Höhere Analysis, Numerik und Stochastik*, 1. Auflage. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

---

## Abkürzungsverzeichnis

Abkürzung	Bedeutung
ggf.	gegebenenfalls
z.B.	zum Beispiel
d.h.	das heißt
I2C	inter-integrated-circuit-bus
SDA	shift data-Leitung
SCL	shift clock-Leitung
GND	ground
R/W	read oder write
ACK	acknowledge
VOC	volatile organic compounds
TVOC	totale volatile organic compounds
CO <sub>2</sub>	carbon dioxide
eCO <sub>2</sub>	equivalent carbon dioxide
ppm	parts per million
ppm	parts per billion
PTAT	proportional to absolut temperature
PEC	packet error check code
MQTT	message queueing telemetry transport
Qos	quality of service
DAC	device attestation certificate
PASE	password authenticated session establishment
ACL	access control list
IDE	integrated development environment

---

## Symbolverzeichnis

Variable	Bedeutung	Basiseinheit
$I$	Strom	A
$U$	Spannung	V

---

## Glossar

Begriff	Beschreibung
anthropogen	Der Begriff bezeichnet etwas, was durch den Menschen verursacht oder beeinflusst wird. Im Kontext des Klimawandels beschreibt der Begriff den Einfluss des Menschen auf das Klima der Erde.
Breadboard-Reihe	Das Breadboard ist in mehrere Teile aufgeteilt. Dabei ist der größte Teil in ein Raster von Spalten und Reihen aufgeteilt, wobei die Spalten von a-j und die Reihen von 1-63 gegliedert sind.
Thermopile-Sensor	Ein Gerät, das an ein Thermoelement kaskadiert wird, um die Spannung zu erhöhen. Thermoelemente werden so angeordnet, dass die heißen Verbindungen nebeneinander liegen.[15]
Scatternet	Ein Scatternet ist eine Gruppe von maximal 10 kleinerer Netze (Piconetze). Diese sind über Bluetooth miteinander verbunden und können auch über größere Entfernung miteinander Daten austauschen.
AES-128	Das AES-128 ist ein Verschlüsselungssystem. Dabei steht AES für Advanced Encryption Standard und die Zahl 128 für die Anzahl von Bits, welche für den Schlüssel verwendet wurde.

---

## **Tabellenverzeichnis**

2.1	Smart-Home Systems . . . . .	5
3.1	Multisensor (Temperatur- und Feuchtigkeitssensor) . . . . .	6
3.2	Luftqualitätssensor . . . . .	6
3.3	Standardpräsenzmelder . . . . .	6
7.1	RedHatText-Standardschriftart und ihre Formen . . . . .	24
7.2	Weitere RedHatText-Schriftarten und ihre Formen . . . . .	24
7.3	RPTU-Farben . . . . .	25
7.4	Smart-Home Standards . . . . .	28

---

## **Abbildungsverzeichnis**

3.1	IST HYT 271 . . . . .	8
3.2	aktiver Sensor des HYT271 . . . . .	8
3.3	Aufbau des Sensors . . . . .	8
3.4	HYT271 Measurement Request Protokoll . . . . .	9
3.5	HYT271 Data Fetch Protokoll . . . . .	9
3.6	HYT271 Umrechnung in reale Werte . . . . .	10
3.7	Joy-it SEN-CCS811V1 . . . . .	10
3.8	CCS811 Protokoll . . . . .	11
3.9	CCS811 Aufbau des Algorithm Results Registers . . . . .	11
3.10	Omron D6T1A01 . . . . .	12
3.11	D6T1A01 Protokoll . . . . .	13
3.12	D6T1A01 Signalverlauf . . . . .	13
3.13	MQTT Modell . . . . .	14
4.1	Hardwareaufbau . . . . .	18
4.2	Sensors-Struktur . . . . .	19
7.1	RPTU Logo als .png . . . . .	26
7.2	Beispielgrafik mit eingebetteten Text . . . . .	27

---

## **Anhang**

### **A.1 Anhang 1**