

Hochschule RheinMain  
Fachbereich ITE  
Studiengang EE-CS

## **Scientific Project**

# **Entwicklung eines Zigbee Praktikums mit quelloffenem Software-Gateway**

verfasst von

**Benedikt HEUSER**  
Matrikelnummer 105320

am

25.04.2022

# Inhaltsverzeichnis

<b>1 Einführung</b>	<b>2</b>
1.1 Anforderungen an die Praktikumsarbeit . . . . .	2
<b>2 Marktübersicht Technologien</b>	<b>3</b>
2.1 Funkprotokolle . . . . .	3
2.2 Zigbee Anwendungen . . . . .	5
2.2.1 Kommerzielle Anwendungen . . . . .	5
2.2.2 Nicht kommerzielle Anwendungen . . . . .	6
<b>3 Grundlagen</b>	<b>7</b>
3.1 LR-WPAN - IEEE 802.15.4 . . . . .	7
3.2 ZigBee . . . . .	7
3.3 Texas Instruments CC Chips . . . . .	8
3.4 Versuchshardware . . . . .	10
3.4.1 RaspberryPi . . . . .	10
3.4.2 RaspberryPi Zigbee Hat . . . . .	10
3.4.3 CC2531 Sniffer Stick . . . . .	11
3.4.4 Phillips Hue Komponenten . . . . .	12
3.5 Eingesetzte Software . . . . .	12
3.5.1 Raspbian OS . . . . .	12
3.5.2 Docker . . . . .	12
3.5.3 Docker-Compose . . . . .	12
3.5.4 zigbee2mqtt . . . . .	13
3.5.5 MQTT . . . . .	20
3.5.6 Wireshark . . . . .	20
3.5.7 Ansible . . . . .	21
<b>4 Versuchsaufbau</b>	<b>23</b>
4.1 Einleitung . . . . .	23
4.2 Containerverwaltung . . . . .	24
4.2.1 Nginx Proxy . . . . .	24
4.2.2 zigbee2mqtt . . . . .	25
4.2.3 mosquitto . . . . .	26
4.3 Namensauflösung . . . . .	27

---

4.4 Anwendungen . . . . .	27
<b>5 Musterlösung</b>	<b>28</b>
5.1 Einleitung . . . . .	28
5.2 Aufgabe 2.1 - Joining einer Phillips Hue Lampe . . . . .	28
5.3 Aufgabe 2.2 - Schalten einer Lampe . . . . .	29
5.4 Aufgabe 3 - Joining einer Fernbedienung über die Lampe . . . . .	29
5.5 Aufgabe 4 - Binding der Fernbedienung . . . . .	30
5.6 Aufgabe 5 - Gruppenbildung . . . . .	31
<b>6 Life Cycle Management</b>	<b>33</b>
6.1 Deployment . . . . .	33
6.2 Zurücksetzen des Versuchs . . . . .	34
6.3 Update der eingesetzten Software . . . . .	35
6.4 Troubleshooting . . . . .	35
<b>7 Anhänge</b>	<b>I</b>
.1 LabGuide . . . . .	I
.2 Texas Instruments SimpleLink . . . . .	XVII
.3 Raspbe Modell 3B . . . . .	XXI
.4 cod.m ZigBee RaspberryPi Modul . . . . .	XXVII
.5 Phillips Hue Dimmer Switch . . . . .	XXX
<b>Abbildungsverzeichnis</b>	<b>XXXIII</b>
<b>Literatur</b>	<b>XXXIV</b>

# Kapitel 1

## Einführung

In dieser Projektarbeit soll ein Praktikumsversuch für das Modul “Internet of Things“ von Professor Dr. Jürgen Winter entwickelt werden. Bei dem Versuch soll die Funktionsweise des Funkprotokolls ZigBee untersucht werden. Es wird ein ZigBee Netzwerk mit mehreren Teilnehmern aufgebaut und die Kommunikation zwischen diesen aufgezeichnet und untersucht. Zusätzlich wird eine Versuchsanleitung erstellt, welche den Studenten durch den Versuch führt.

### 1.1 Anforderungen an die Praktikumsarbeit

Die Anforderungen an der Versuch werden an dieser Stelle definiert.

- **A010** - Der Versuch soll an einem Tag durchführbar sein.
- **A020** - Der Versuch soll nur wenig Vorwissen in Linux vorraussetzen.
- **A030** - Der Versuch setzt Vorwissen in Paketorientierten Datenübertragung vorraus.
- **A040** - Der Versuch setzt Vorwissen in der Bedienung von Wireshark vorraus.
- **A050** - Der Versuch soll zu Hause und in der Hochschule durchführbar sein.
- **A100** - Der Versuch soll automatisch auf dem RaspberryPi ausgerollt werden können.
- **A120** - Es soll aktuelle und quelloffene Software zum Einsatz kommen.
- **A210** - Es soll die Funktionsweise des Joinings, des Routings, des Bindings sowie der Gruppenbildung untersucht werden.
- **A210** - Es sollen die implementierten Sicherheitsmechanismen untersucht und bewertet werden.

## Kapitel 2

# Marktübersicht Technologien

“Internet of things“ beschreibt die Befähigung von Endgeräten mit Datennetzen zu kommunizieren. So können Waschmaschinen einen fertigen Waschgang kommunizieren, oder ein Heizungsthermostat die Außentemperatur aus dem Wetterbericht beziehen. Viele Hersteller haben mittlerweile ein breites Portfolio an sogenannten “Smart Devices“ und der dazu je nach Übertragungsprotokoll notwendigen “Bridge“ sowie einer entsprechenden App zur Steuerung. Der Hersteller Phillips mit seiner Produktmarke “Hue“ kann als Beispiel genannt werden. Die Produktgruppe umfasst eine Bridge mit zugehöriger Smartphone App, sowie den klassischen Komponenten wie Lampen, Steckdose und Schalter. Der Markt wurde durch Heimassistenten stark belebt. Amazons Alexa, der Google Echo Dot und die Pendanten von Apple und Microsoft sind in immer mehr Haushalten zu finden. Diese Heimassistenten können sich entweder mit den herstellerspezifischen Bridges verbinden, oder können direkt an PANs (Personal-Area-Networks) wie Zigbee teilnehmen und die Geräte steuern. Dieses Kapitel soll einen Überblick über die relevanten Technologien am Markt verschaffen.

### 2.1 Funkprotokolle

Aktuell gibt es mehrere Funkprotokolle, welche im Bereich IoT relevant sind. Dazu gehören:

- **Wlan**

Wlan ist ein verbreiteter und etablierter Standard, der überwiegend für die Anbindung mobiler Geräte an den Internetrouter dient. Dies macht es naheliegend, auch smarte Geräte per WLAN zu vernetzen. Wlan ist allerdings optimiert für hohe Übertragungsraten und nicht für leistungsschwache Endgeräte. Dies ist insbesondere für batteriebetriebene Geräte nachteilig. Zusätzlich benötigt jedes Endgerät Zeitslots um Daten zu Senden und zu Empfangen. Diese müssten auch für Geräte freigehalten werden, die eigentlich sehr wenig Bandbreite benötigen. Viele Geräte können sich also nachteilig auf die Performance eines Wlan Netzwerks auswirken. Dieses Problem ist seit WIFI6 mit OFDMA durch mehrere Subcarrier zumindest weniger stark ausgeprägt.

- **Blueooth**

Ebenso wie Wlan hat Bluetooth eine weite Verbreitung. Durch Implementierung des Standard

Bluetooth LE ist es möglich leistungsschwache sowie batteriebetriebene Geräte mit Bluetooth auszustatten. Bluetooth ist allerdings nicht für hohe Reichweiten oder für Netzwerke mit vielen Teilnehmern konzipiert. Primäre Anwendungsfall ist zum Beispiel das Verbinden eines Headsets mit einem Handy. Seit 2017 existiert der Standard "Bluetooth Mesh". Dieser löst Probleme des normalen Bluetooths um es für IoT einzusetzen. Bisher gibt es keine bekannten größeren Ecosysteme in denen es eingesetzt wird. Ein Benchmark zeigt keine Vorteile gegenüber ZigBee und Thread. [sila]

- **Z-Wave**

Z-Wave [zwa23] ähnelt technologisch Zigbee. Das Protokoll ist proprietär. Der Hauptunterschied ist, dass Z-Wave in einem frei nutzbaren Low-Frequency Band arbeitet und nicht wie ZigBee im 2,4 Ghz Band. Die Reichweite ist durch die geringere Trägerfrequenz höher. Die Verbreitung am Markt ist allerdings geringer.

- **ZigBee**

Zigbee [All15] ist ein auf den 802.40.5 Standard aufbauendes Protokoll, welches grundlegend für die Anbindung vieler leistungsschwacher Geräte in einem großen räumlichen Areal konzipiert ist. Ein großer konzeptioneller Vorteil ist, dass bei ZigBee ein Mesh-Netzwerk aufgebaut wird. Es können auch Geräte angebunden werden, die keine direkte Funkverbindung zum Koordinator haben. Zusätzlich sind Funktionen implementiert, welche das Management einer hohen Anzahl von Devices erleichtert. Ein großer Nachteil der bei ZigBee oft angeführt wird ist, dass die Geräte nicht direkt per IPv6 adressierbar sind. 2013 wurde ein Standard mit dem Namen "ZigBee IP" veröffentlicht, der IPv6 supportet. Auf diesem wiederum basiert ZigBee Smart Energy, welches in intelligenten Stromzählern eingesetzt wird. In Großbritannien sind diese bereits im Einsatz[Neh23], in Deutschland sollen zumindest bis 2032 die alten Stromzähler durch vernetzte ersetzt werden. [BMW23]

- **Thread**

Thread [Goo23] ist ein Funkprotokoll welches ebenfalls auf dem 802.15.4 Standard basiert. Ebenso wie ZigBee ist es Meshfähig. Ein entscheidendes Unterscheidungsmerkmal ist allerdings, dass die Geräte per IPv6 adressiert werden. Daher sind die Geräte theoretisch ohne die Verwendung einer Bridge aus einem herkömmlichen Ethernet Netzwerk addresierbar. Dies ist vor allem für die Anbindung von Geräten im öffentlichen Raum wie Parkuhren extrem vorteilhaft. Bisher setzt nur Google mit dem Heimassistenten Nest im größeren kommerziellen Umfeld ein. Es gibt erste Hersteller wie eve und nanoleaf welche entsprechende Smart Devices anbieten. Eine Liste mit zertifizierten Produkten ist hier zu finden: <https://www.threadgroup.org/What-is-Thread/Thread-Benefits#certifiedproducts>

## 2.2 Zigbee Anwendungen

### 2.2.1 Kommerzielle Anwendungen

#### Amazon Echo

Der Heimassistent Amazon Echo ab Generation 4 ist der einzige seiner Art der eine Zigbee Integration hat und damit als Gateway und Koordinator fungieren kann. Die Pendanten der Firmen Google, Microsoft und Apple benötigen ein dediziertes Zigbee Gateway.

#### Phillips Hue

Phillips vertreibt unter dem Namen eine Zigbee Bridge und eine Vielzahl von Devices aus dem Segment Beleuchtung und Steckdosen.

#### Dresden Electronic

Dresden Electronic bietet Software und Hardware zum Aufbau von Zigbee Netzwerken an. Es werden Zigbee USB Adapter und RaspberryPi Hats mit ATMega Chips angeboten, sowie eine Steuerungssoftware “deCONZ“. Als komplette Produktlinie für den Endanwender gibt es die Produktsparte “Phoscon“, hauptsächlich zur smarten Beleuchtung.

#### Weitere Hersteller

Weitere bekannte Hersteller/Marken mit Zigbee Devices und Gateways:

- **Telekom** - QIVICON
- **Logitech** - Harmony Hub
- **LIDL** - Silvercrest
- **TUYA** - Smart Life
- **Innr** - ZigBee Bridge
- **SONOFF** - Günstige Hardware jeder Art
- **homee** - modular Smart Home Central
- **Osram** - Lightify
- **Ledvance** - Zigbeefähige Steckdosen und Lampen “Smart+“

Nachteil dieser Lösungen ist, dass die Kompatibilität zu Geräten von Drittherstellern vollständig in der Hand des Herstellers ist. In der Regel ist aus wirtschaftlichen Gründen die Unterstützung konkurrierender Hersteller nicht gewünscht. Es ist schwierig, bei Anschaffung eines dieser Systeme die Kompatibilität anderer Geräte sicherzustellen, da offiziell heißt nur die Geräte aus dem eigenen Haus unterstützt sind.

### 2.2.2 Nicht kommerzielle Anwendungen

Vorteil von quelloffenen Anwendungen ist, dass diese durch eine Community gepflegt und Geräte von drittherstellern beliebig integriert werden können. Grundlegend ist der Zigbee Standard universell, und die Kompatibilität von Geräten verschiedener Hersteller möglich.

#### **zigbee2mqtt**

zigbee2mqtt [Kan22] ist ein quelloffenes Projekt auf GitHub. Die Anwendung kann anstelle von proprietären Bridges als ZigBee-Gateway eingesetzt werden. Die Anwendung kann ein ZigBee Netzwerk in der Rolle des Koordinators verwalten und Funktionen per MQTT nach außen verfügbar machen.

#### **ZHA**

ZHA ist ein direkt in HomeAssistant integriertes Plugin, um Zigbee Koordinatoren direkt in HomeAssistant einzubinden. Vorteil von ZHA ist, dass ZigBee Chips mehrerer Hersteller unterstützt werden. ZHA unterstützt neben Texas Instruments auch Hardware von Dresden Elektronik, Silicon Labs, DIGI und ZiGate. ZHA ist für den Anwender komfortabler als zigbee2mqtt, es sind wenige technische Informationen ersichtlich oder konfigurierbar. Die Endgeräte Kompatibilität ist derzeit schlechter als bei zigbee2mqtt.

# **Kapitel 3**

## **Grundlagen**

In diesem Kapitel werden alle verwendeten Komponenten und Technologien kurz erläutert.

### **3.1 LR-WPAN - IEEE 802.15.4**

LR-WPAN [IEE23] steht für “Low Rate - Wireless personal area network“. Es handelt sich um ein drahtloses geschlossenes Netzwerk, welches für niedrige Datenraten ausgelegt ist. Der Standard definiert den Physical Layer sowie den Media-Access Layer und bildet die Grundlage von ZigBee, Thread und 6LowPAN. Im Standard sind mehrere Modulationsverfahren sowie Frequenzbereiche definiert. Der Standard beinhaltet sowohl die klassische Stern Topologie, als auch die bei ZigBee verwendete Peer-to-Peer Topologie. Weiteres Schlüsselfeatures sind Kollisionsvermeidung sowie Echtzeifunktionalitäten. Im Vergleich zum 802.1d Standard fallen vor allem die kürzeren Adressen auf. Dadurch kann die für den Anwendungsbereich wertvolle Bandbreite und Rechenleistung reduziert werden.

### **3.2 ZigBee**

Die ZigBee Alliance wurde durch ein Konsortium von Herstellern gegründet, um einen einheitlichen Übertragungsstandard im Bereich Heiamtatisierung voranzubringen. ZigBee basiert auf dem offenen 802.15.4 Standard, bringt allerdings zusätzliche Komponenten mit. ZigBee ist in Form von weiteren Protokollsichten implementiert, welche auf IEEE 802.15.4 aufsetzen. ZigBee nutzt DSSS, also Frequenzspreizung als Modulationsverfahren. Die genutzten Kanäle, 11 bis 26, liegen im 2,4 Ghz Band. Zigbee interferiert damit mit WLAN und Bluetooth.

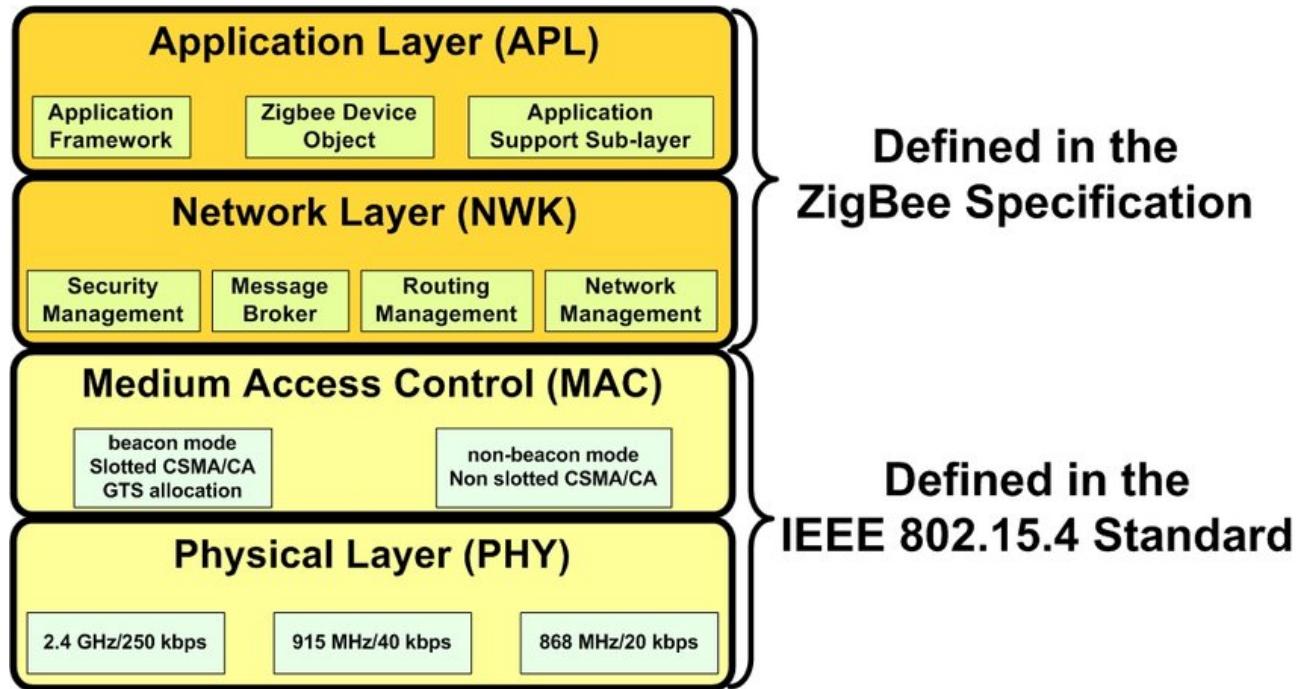


Abbildung 3.1: ZigBee Protocoll Stack

Bildquelle: [https://www.researchgate.net/figure/IEEE820154-ZigBee-protocol-stack-architecture\\_fig2\\_265150617](https://www.researchgate.net/figure/IEEE820154-ZigBee-protocol-stack-architecture_fig2_265150617)

Der Anwendungsbereich für ZigBee ist die Heimautomatisierung. Geräte können zentral gesteuert und überwacht werden. Markante Eigenschaft von ZigBee ist, dass die Geräte keine direkte Funkverbindung zu einem zentralen Controller brauchen. Andere Geräte können als Router fungieren, und damit die Reichweite erhöhen. Sende- und Empfangsleistung ist vor allem bei kleinen batteriebetriebenen Geräten oft der einschränkende Faktor.

### 3.3 Texas Instruments CC Chips

Texas Instruments bietet ein Spektrum von Microcontrollern, die sich mit entsprechender Firmware für Devices als ZigBee SoC nutzen lassen. Alternativ kann mit so einem Microcontroller auch ein Koordinator betrieben werden. Kleinere Varianten können in Endgeräten wie Lampen und Thermostate, größere als Koordinator selbst verwendet werden.

Die aktuelle Chipfamilie TexasInstruments CC26XX:

**Table 3-1. Device Family Overview**

DEVICE	PHY SUPPORT	FLASH (KB)	RAM (KB)	GPIO	PACKAGE <sup>(1)</sup>
CC2650F128xxx	Multi-Protocol <sup>(2)</sup>	128	20	31, 15, 10	RGZ, RHB, RSM
CC2640F128xxx	Bluetooth low energy (Normal)	128	20	31, 15, 10	RGZ, RHB, RSM
CC2630F128xxx	IEEE 802.15.4 Zigbee(/6LoWPAN)	128	20	31, 15, 10	RGZ, RHB, RSM
CC2620F128xxx	IEEE 802.15.4 (RF4CE)	128	20	31, 10	RGZ, RSM

(1) Package designator replaces the xxx in device name to form a complete device name, RGZ is 7-mm × 7-mm VQFN48, RHB is 5-mm × 5-mm VQFN32, and RSM is 4-mm × 4-mm VQFN32.

(2) The CC2650 device supports all PHYs and can be reflashed to run all the supported standards.

Abbildung 3.2: TI Device Family

Als Koordinator werden die Leistungsfähigeren Chips aus der 265X Reihe eingesetzt. Touchlink ist eine Technologie, um ZigBee Geräte einfach zu koppeln. Dafür setzt Touchlink Bluetooth LE ein. Die Unterstützung von diesem Protokoll ist daher vorteilhaft.

## 6 Device Comparison

Device	RADIO SUPPORT										FLASH (KB)	RAM + Cache (KB)	GPIO	PACKAGE SIZE			
	Sub-1 GHz Prop.	2.4 GHz Prop.	Wireless M-Bus	Wi-SUN®	Sidewalk	Bluetooth® LE	ZigBee	Thread	Multiprotocol	+20 dBm PA				4 x 4 mm VQFN (32)	5 x 5 mm VQFN (32)	5 x 5 mm VQFN (40)	7 x 7 mm VQFN (48)
CC1310	X		X								32-128	16-20 + 8	10-30	X	X		X
CC1311R3	X		X								352	32 + 8	22-30			X	X
CC1311P3	X		X								352	32 + 8	26				X
CC1312R	X		X X								352	80 + 8	30				X
CC1312R7	X		X X	X					X		704	144 + 8	30				X
CC1352R	X	X	X X		X	X	X	X			352	80 + 8	28				X
CC1352P	X	X	X X		X	X	X	X	X		352	80 + 8	26				X
CC1352P7	X	X	X X	X	X	X	X	X	X		704	144 + 8	26				X
CC2640R2F					X						128	20 + 8	10-31	X	X		X
CC2642R						X					352	80 + 8	31				X
CC2642R-Q1						X					352	80 + 8	31				X
CC2651R3		X			X	X					352	32 + 8	23-31			X	X
CC2651P3		X			X	X			X		352	32 + 8	22-26			X	X
CC2652R		X			X	X	X	X			352	80 + 8	31				X
CC2652RB		X			X	X	X	X			352	80 + 8	31				X
CC2652R7		X			X	X	X	X	X		704	144 + 8	31				X
CC2652P		X			X	X	X	X	X	X	352	80 + 8	26				X
CC2652P7		X			X	X	X	X	X	X	704	144 + 8	26				X

Abbildung 3.3: TI CC 265X Serie

In der Tabelle sind die unterstützten Protokolle der einzelnen Modelle sowie deren Leistungsfähigkeit aufgeführt. Es ist anzumerken, dass die größeren Modelle schon den Standard Thread unterstützen, der vermutlich durch das Projekt "Matter" erheblich an Bedeutung gewinnen wird.

Texas Instruments stellt als Basis für ZigBee Anwendungen eine Bibliothek "Z-Stack" zur Verfügung.

Diese stellt grundlegenden Funktionen um das ZigBee Protokoll zu implementieren zur Verfügung. Mit Texas Instruments Code Composer Studio steht eine IDE bereit, um den Entwicklungsprozess zu unterstützen. Auf den entsprechend Leistungsfähigeren Chips lassen sich in freie Speicherbereiche noch zusätzliche Funktionalitäten einprogrammieren. Die Chips können mit Programmierboards des Herstellern programmiert werden. Alternativ kann man günstig einen USB-Stick mit aufgelöten CC Chip erwerben, und auch diesem mit entsprechenden Tools programmieren.

Weiter Informationen: <https://www.ti.com/tool/Z-STACK#overview>

In dem OpenSource Projekt zigbee2mqtt werden ausschließlich Chips von Texas Instruments unterstützt. Die meisten gängigen Anbieter von Microchips haben entsprechende Modelle im Angebot.

## 3.4 Versuchshardware

### 3.4.1 RaspberryPi

Der RaspberryPi ist ein ARM basierter Computer im Mini-Format. Er dient in diesem Versuch als Applikationsserver und gleichzeitig als Versuchs-PC, auf dem der Versuch durchgeführt wird. Die eingesetzten Anwendungen sind als Webservice implementiert und werden als Container mit Docker betrieben.

Der RaspberryPi besitzt die PC typischen Schnittstellen wie Ethernet, HDMI, und USB. Als Speicher wird eine SD-Karte eingesetzt.

### 3.4.2 RaspberryPi Zigbee Hat

Als Zigbee Koordinator wird ein auf dem TI CC2652 basierendem RaspberryPi Hat vom Hersteller cod.m eingesetzt. Dieser wurde vom Hersteller für den Einsatz mit "homegear" oder "zigbee2Mqtt" entwickelt. Ein Datenblatt und Bedienungsanleitung sind im Anhang.



Abbildung 3.4: ZigBee cod.m Koordinator

Bildquelle: cod.m Produktfoto

Durch einen Lötjumper kann zwischen einer aufgedruckten und einer per "ufl" angeschlossenen Antenne gewechselt werden. Im Auslieferungszustand ist es notwendig, eine externe Antenne anzuschließen.

### 3.4.3 CC2531 Sniffer Stick

Mit diesem Stick wird die ZigBee Kommunikation des Versuchsnetswerkes mitgeschnitten. Der Stick basiert auf einem leistungsschwachen Chip, der mit entsprechender Firmware ZigBee Kommunikation mitschneiden kann.

Als Treiber wird ein in C geschriebenes Programm verwendet, welches es ermöglicht den Stick direkt als Interface in Wireshark hinzuzufügen. Der Quellcode findet sich in GitHub unter <https://github.com/andrebdw/wireshark-cc2531>. Eine Anleitung zum kompilieren ist im Repository enthalten. Die hieraus entstehende ausführbare Datei muss in ein entsprechendes Verzeichnis kopiert werden, und kann anschließend als Interface in Wireshark ausgewählt werden. Die Funktion nennt sich bei Wireshark "extcap".

### 3.4.4 Phillips Hue Komponenten

Unter dem Namen “Hue“ vertreibt Phillips eine Reihe von vernetzten Endgeräten sowie entsprechende Komponenten um diese zu steuern. Die Phillips Hue Serie setzt auf ZigBee sowie Bluetooth LE. Unter anderem sind Lampen, Steckdosen, eine Bridge sowie eine App verfügbar. Die Bridge stellt bei traditionellen Lösungen die Schnittstelle zwischen dem ZigBee Netzwerk und der IP Kommunikation zu einer Smartphone App oder ähnlichem. Die Endgeräte sind Kompatibel zu dem Software-Gateway zigbee2mqtt, benutzen also keine speziellen Schlüssel. Die Lampen werden in dem Versuch als Demonstrationsobjekte eingesetzt. Sie können Ein- und Ausgeschaltet werden, sowie gedimmt werden. Zusätzlich wird eine Phillips Hue Fernbedienung verwendet, die zur Steuerung der Lampen genutzt wird.

## 3.5 Eingesetzte Software

### 3.5.1 Raspbian OS

RaspbianOS ist eine leichtgewichtige Linux Distribution, welche direkt vom Hersteller des RaspberryPis speziell auf die Bedürfnisse des Board angepasst ist. Es enthält eine Desktop Umgebung sowie grundlegende Pakete. Es basiert auf Debian, damit sind entsprechende gut gepflegte Paketquellen verfügbar.

### 3.5.2 Docker

Docker ist eine Containerisierungslösung, um Anwendungen containerisiert auf Linux-Servern ausführen zu können. Docker reduziert erheblich den Aufwand Anwendungen zu betreiben. Docker Container beinhalten alle Abhängigkeiten um die Anwendung im Container lauffähig zu machen. Prozesse laufen in eigenen Namespaces und sind dadurch abgekoppelt von anderen Containern sowie dem Hostbetriebssystem. Im Unterschied zur Virtualisierung werden einige Ressourcen gemeinsam genutzt. Dadurch ist die Effizienz höher als bei traditioneller Virtualisierung, bei der ein vollständiges Betriebssystem virtualisiert wird.

### 3.5.3 Docker-Compose

Docker-Compose ist ein Tool, um Containerumgebungen im Textformat, hier YAML, zu beschreiben. Ein Container kann im einfachen Fall per Docker-CLI mit entsprechenden Parametern gestartet werden:

```
1 | docker run hello-world -v ./home:/home -p 80:80
```

Durch diesen Befehl wird der Container “hello-world“ aus dem Docker Repository geladen und anschließend gestartet. In diesem ist ein einfacher Webserver der bei Aufruf ein “Hello world !“ zurück-

gibt implementiert. Zusätzlich wird der Ordner "home" in den Container gemountet. Dieser bleibt auch bei einem erneuten Laden des Containers persistent. Dies wird beispielweise für Konfigurationsdateien oder andere persistente Dateien genutzt. Um den Container auch auf der Schnittstelle des Host-Systems verfügbar zu machen, wird der Port 80 auf den Container Port 80 gemappt. Die Funktionsweise wird später erläutert.

Als handliche Alternative zu der Docker-CLI lässt sich der Zielzustand auch in einer Textdatei beschreiben:

```

1 | version: '3'
2 | services:
3 |   helloworld:
4 |     container_name: helloworld
5 |     image: hello-world
6 |     ports:
7 |       - 80:80
8 |     volumes:
9 |       - ./home:/home
10 |    restart: unless-stopped

```

Mit einem

```
1 | docker-compose up -d
```

gelangt man zum selben Ergebniss wie mit dem vorher gezeigtem CLI Befehl. Durch die Beschreibung in der Textdatei ist das Ergebniss allerdings reproduzierbar.

### 3.5.4 zigbee2mqtt

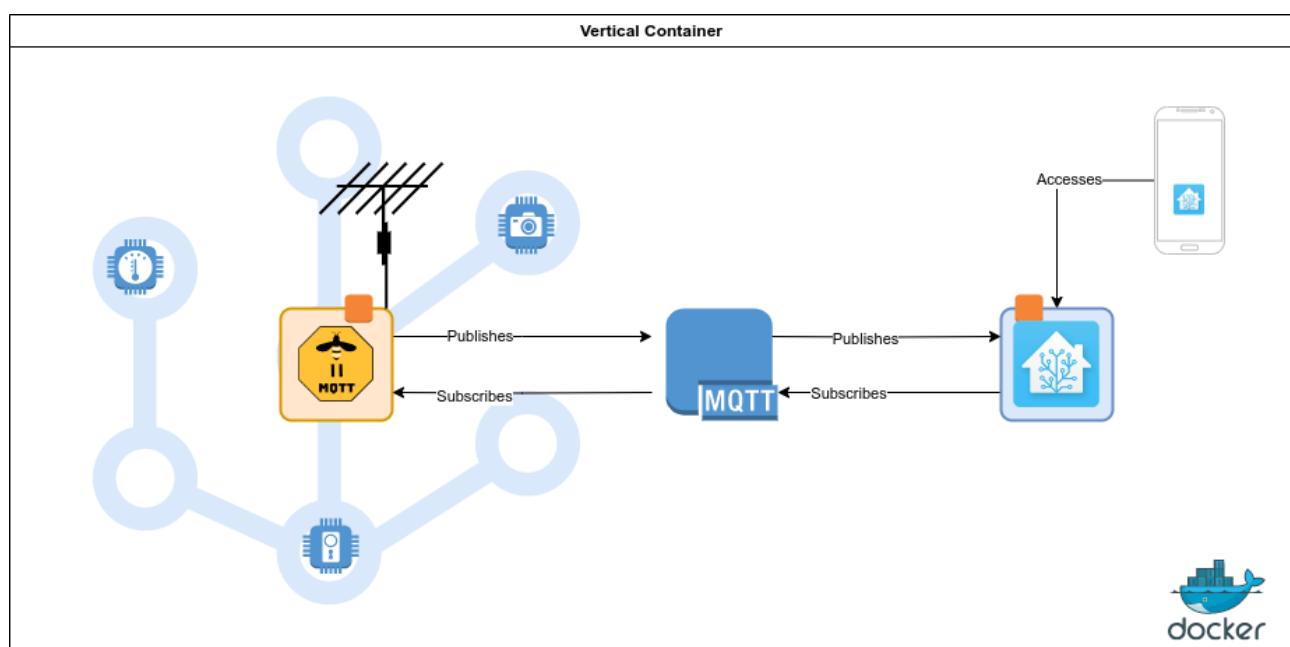


Abbildung 3.5: Zigbee2Mqtt Kontext

zigbee2mqtt ist ein offenes Softwareprojekt und kann am besten mit dem Begriff “Software-Zigbee-Gateway“ beschrieben werden. Es übernimmt die Funktionalität, die normalerweise entsprechende Bridges der Hersteller übernehmen. Während traditionelle Bridges, wie zum Beispiel die Phillips Hue Bridge eine REST API zur Verfügung stellen um mit entsprechenden Apps zu kommunizieren, macht zigbee2mqtt die Geräte per MQTT nach außen verfügbar. Auf abstrakter Ebene heißt das, dass es ein Gateway zwischen einem Zigbee Netzwerk und einem traditionellen IPv4 Netzwerk ist. Zur Steuerung und Visualisierung lassen sich per MQTT Anwendungen wie “Homeassistant“ oder “OpenHUB“ oder auch entsprechende Eigenentwicklungen einsetzen. Zigbee2mqtt greift direkt auf den cod.m ZigBee Adapter zu.

Quellcode und Dokumentation: <https://github.com/Koenkk/zigbee2mqtt>

Homepage: <https://www.zigbee2mqtt.io/>

Zigbee2mqtt verwaltet das Zigbee Netzwerk und ermöglicht es Drittanwendungen, die Geräte in diesem ZigBee Netz zu steuern. Wird ein neues Device ins das Netzwerk eingefügt, kündigt zigbee2mqtt das Gerät per MQTT an, und teilt der Anwendung nach erfolgreichem Interview die verfügbaren Cluster des neuen Teilnehmers mit.

Zigbee2Mqtt verwaltet drei Datenbanken, welche die Funktionsweise deutlich machen. Viele Funktionen, wie zum Beispiel die Verwaltung von Routingtabellen und Verschlüsselung der Kommunikation sind direkt in der Hardware implementiert. Diese Funktionen lassen sich wie in der im Punkt TI CC Firmware gezeigen API steuern und abfragen. Zigbee2mqtt verwaltet in einer eigenen Datenbank die Geräte im Netzwerk sowie deren Eigenschaften. Folgende Datensätze finden sich in der Anwendung: **coordinator-backup.json**

Hier sind die für die Initialisierung beim Start des Koordinators wichtigen Informationen abgelegt. Dies beinhaltet alle dem Netzwerk zugehörigen Geräte. Durch löschen dieser Datei wird das Netzwerk vollständig zurückgesetzt. Die einzelnen Teilnehmer müssen dann manuell per Touchlink oder nach herstellerspezifischem Verfahren zurückgesetzt werden, um wieder einem neuen Netzwerk beitreten zu können.

```

1  {
2      "metadata": {
3          "format": "zigpy/open-coordinator-backup",
4          "version": 1,
5          "source": "zigbee-herdsman@0.14.103",
6          "internal": {
7              "date": "2023-05-04T19:48:28.936Z",
8              "znpVersion": 1
9          }
10     },
11     "stack_specific": {
12         "zstack": {
13             "tclk_seed": "69a6670050d8354347405537724e1a81"
14         }
15     },
16     "coordinator_ieee": "00124b0026b748c8",
17     "pan_id": "1a62",
18     "extended_pan_id": "00124b0026b748c8",
19     "nwk_update_id": 0,

```

```

20     "security_level": 5,
21     "channel": 11,
22     "channel_mask": [
23       11
24     ],
25     "network_key": {
26       "key": "01030507090b0d0f00020406080a0c0d",
27       "sequence_number": 0,
28       "frame_counter": 5552161
29     },
30     "devices": [
31       {
32         "nwk_address": "9a3b",
33         "ieee_address": "0017880104b9359d",
34         "is_child": false,
35         "link_key": {
36           "key": "87b4d0a2668847d8a876fe4454bf654e",
37           "rx_counter": 0,
38           "tx_counter": 121
39         }
40       },
41     ...

```

**state.json**

In dieser Datei sind alle aktuellen Zustände Geräte im Netzwerk hinterlegt. Sie dient dazu, bei einem Neustart des Koordinators den letzten Zustand wieder herzustellen.

```

1   ...
2   "0xbc33acffffe9587ed": {
3     "brightness": 15,
4     "state": "OFF",
5     "color_mode": "color_temp",
6     "color_temp": 360,
7     "linkquality": 40,
8     "color": {
9       "x": 0.4542,
10      "y": 0.4092
11    },
12    "do_not_disturb": false
13  ...

```

**database.db**

Dies ist die zentrale Datenbank von zigbee2mqtt. Da SQLite eingesetzt wird, lässt sich auch hier der Inhalt wie bei einer Textdatei einfach auslesen. Es liegt für jedes Device ein Datensatz ab.

Datensatz des Koordinators:

```

1 { "id":1,"type":"Coordinator","ieeeAddr":"0x00124b0026b748c8","nwkAddr":0,"manufId"
2   :0,"epList":[1,2,3,4,5,6,8,10,11,12,13,47,110,242],
3   "endpoints": {"1": {"profId":260,"epId":1,"devId":5,"inClusterList":[],"outClusterList"
4     :[],"clusters":{},"binds":[],"configuredReportings":[]},
5   "meta":{}}, "2": {"profId":257,"epId":2,"devId":5,"inClusterList":[],"outClusterList"
6     :[],"clusters":{},"binds":[],"configuredReportings":[]},
7   "meta":{}}, "3": {"profId":260,"epId":3,"devId":5,"inClusterList":[],"outClusterList"
8     :[],"clusters":{},"binds":[],"configuredReportings":[]},
9   "meta":{}}, "4": {"profId":263,"epId":4,"devId":5,"inClusterList":[],"outClusterList"
10    :[],"clusters":{},"binds":[],"configuredReportings":[]},
11  ...

```

```

6 "meta":{}}, "5": {"profId":264, "epId":5, "devId":5, "inClusterList":[], "outClusterList"
7 :[], "clusters":{}, "binds":[], "configuredReportings":[]},
8 "meta":{}}, "6": {"profId":265, "epId":6, "devId":5, "inClusterList":[], "outClusterList"
9 :[], "clusters":{}, "binds":[], "configuredReportings":[]},
10 "meta":{}}, "8": {"profId":260, "epId":8, "devId":5, "inClusterList":[], "outClusterList"
11 :[], "clusters":{}, "binds":[], "configuredReportings":[]},
12 "meta":{}}, "10": {"profId":260, "epId":10, "devId":5, "inClusterList":[], "outClusterList"
13 :[], "clusters":{}, "binds":[], "configuredReportings":[]},
14 "meta":{}}, "11": {"profId":260, "epId":11, "devId":1024, "inClusterList": [1281, 10], "outClusterList": [1280, 1282], "clusters":{}, "binds":[], "configuredReportings":[]},
15 "meta":{}}, "12": {"profId":49246, "epId":12, "devId":5, "inClusterList": [], "outClusterList": [], "clusters":{}, "binds":[], "configuredReportings":[]},
16 "meta":{}}, "13": {"profId":260, "epId":13, "devId":5, "inClusterList": [25], "outClusterList": [], "clusters":{}, "binds":[], "configuredReportings":[]},
17 "meta":{}}, "47": {"profId":260, "epId":47, "devId":5, "inClusterList": [], "outClusterList": [], "clusters":{}, "binds":[], "configuredReportings":[]},
18 "meta":{}}, "110": {"profId":260, "epId":110, "devId":5, "inClusterList": [], "outClusterList": [], "clusters":{}, "binds":[], "configuredReportings":[]},
19 "meta":{}}, "242": {"profId":41440, "epId":242, "devId":5, "inClusterList": [], "outClusterList": [], "clusters":{}, "binds":[], "configuredReportings":[]},
20 "meta":{}}, "interviewCompleted":true, "meta":{}, "lastSeen":1671278654240, "defaultSendRequestWhen": "immediate"

```

In dieser Datenbank sind alle Geräte, welche direkt mit dem Koordinator verbunden sind vermerkt. Zu jedem Gerät werden die gebundenen Cluster vermerkt.

### **zigbee-herdsman**

Der Herdsman ist die eigentliche Kernanwendung von zigbee2mqtt. Dieses Modul verbindet sich direkt über einen serielle Schnittstelle mit dem Koordinator. Über diese Schnittstelle spricht Herdsman die API des Koordinators an um das Netzwerk zu verwalten. Herdsman verwaltet die Datenbank und damit den Zustand des Netzwerkes. Das Modul stellt nach außen eine API zur Verfügung, mit der sich das Netzwerk verwalten lassen kann. Auf diese API greift auch die integrierte WebGui zu.

Die API von zigbee-herdsman wird im entsprechenden GitHub Repository dokumentiert.

<https://github.com/Koenkk/zigbee-herdsman>

### **zigbee-herdman-converters**

Dieses Modul dient zur Abstraktion verschiedenster Geräte für zigbee2mqtt. Das Modul hat ein eigenes Repository. Es liegt für jeden Hersteller von unterstützten Smart Devices eine Typescript Datei ab, in der die Geräte definiert werden. Die Geräte werden Anhang ihrer ID erkannt. Anschließend werden die Cluster beschrieben sowie die gültigen parametrierbaren Werte.

```

1 const definitions: Definition[] = [
2   {
3     fingerprint: [{modelID: 'TS130F', manufacturerName: '_TZ3000_vd43bbfq'}, {
4       modelID: 'TS130F', manufacturerName: '_TZ3000_fccpjz5z'},
5       model: 'QS-Zigbee-C01',
6       vendor: 'Lonsonho',
7       description: 'Curtain/blind motor controller',
8       fromZigbee: [fz.cover_position_tilt, fz.tuya_cover_options],
9       toZigbee: [tz.cover_state, tz.cover_position_tilt, tz.tuya_cover_calibration,
          , tz.tuya_cover_reversal],
       meta: {coverInverted: true},

```

```

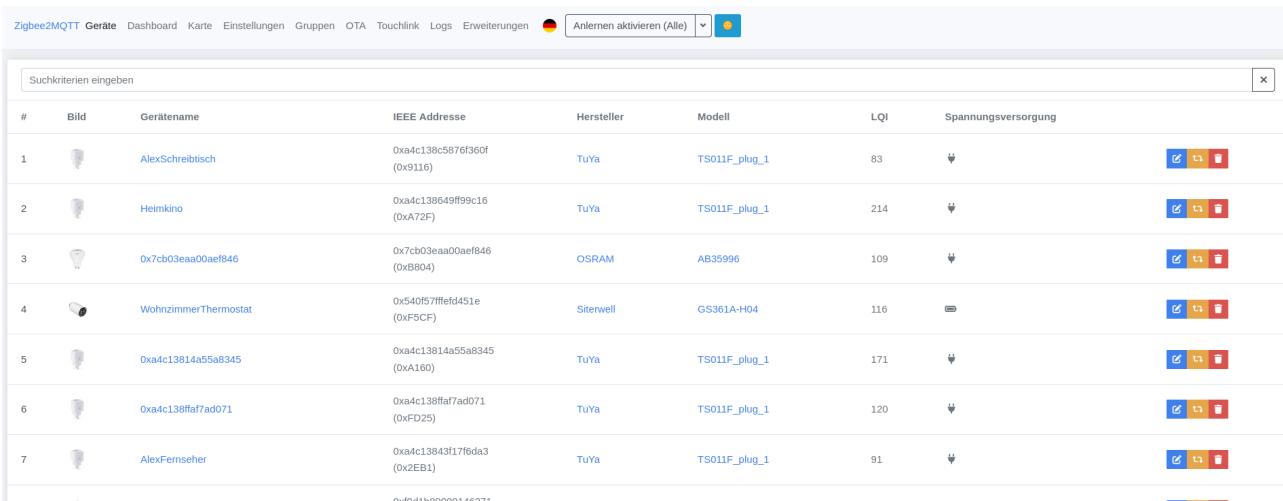
10     exposes: [e.cover_position(), e.enum('moving', ea.STATE, ['UP', 'STOP', 'DOWN']),
11         e.binary('calibration', ea.ALL, 'ON', 'OFF'), e.binary('motor_reversal',
12             ea.ALL, 'ON', 'OFF'),
13         e.numeric('calibration_time', ea.STATE).withUnit('S').withDescription(
14             'Calibration time')],
15     },

```

An dieser Stelle wird ein Rolladenmotor von Lonsonho definiert. Damit kann dieser durch zigbee2mqtt erkannt und gesteuert werden. In weiten Modulen werden dann die Werte die von den Geräten kommen beziehungsweise an diese gesendet werden auf einen globalen Standard konvertiert. Damit können Rolladenmotoren von verschiedenen Herstellern mit einer Anwendung identisch gesteuert werden. Das Repository findet sich unter <https://github.com/Koenkk/zigbee-herdsman-converters>. Durch den quelloffenen Ansatz ließe sich ein Gerät, welches bisher noch nicht unterstützt wird, hier nachpflegen.

## zigbee2mqtt

Dieses Modul umschreibt die beiden vorher beschriebenen Module und umfasst noch eine Weboberfläche. Die Weboberfläche dient zur Verwaltung und Visualisierung des Netzwerkes.



The screenshot shows a web-based interface for managing a zigbee2mqtt setup. At the top, there's a navigation bar with links for 'Zigbee2MQTT', 'Geräte' (Devices), 'Dashboard', 'Karte', 'Einstellungen', 'Gruppen', 'OTA', 'Touchlink', 'Logs', and 'Erweiterungen'. There are also buttons for 'Anlernen aktivieren (Alle)' and a language switch. Below the navigation is a search bar labeled 'Suchkriterien eingeben'. The main area is a table listing seven devices:

#	Bild	Gerätename	IEEE Adresse	Hersteller	Modell	LQI	Spannungsversorgung	Aktionen
1		AlexSchreibtisch	0xa4c138c5876f360f (0x9116)	TuYa	TS011F_plug_1	83		
2		Heimkino	0xa4c138649ff99c16 (0xA72F)	TuYa	TS011F_plug_1	214		
3		0x7cb03ea00aaef846	0x7cb03ea00aaef846 (0xB804)	OSRAM	AB35996	109		
4		WohnzimmerThermostat	0x540f57ffefd451e (0xF5CF)	Siterwell	GS361A-H04	116		
5		0xa4c13814a55a8345	0xa4c13814a55a8345 (0xA160)	TuYa	TS011F_plug_1	171		
6		0xa4c138ffa7ad071	0xa4c138ffa7ad071 (0xFD25)	TuYa	TS011F_plug_1	120		
7		AlexFernseher	0xa4c13843f17fdas3	TuYa	TS011F_plug_1	91		

Abbildung 3.6: zigbee2mqtt Webfrontend

Die Weboberfläche biete die Möglichkeit alle Endpunkte von Herdsman abzufragen und entsprechend zu steuern.

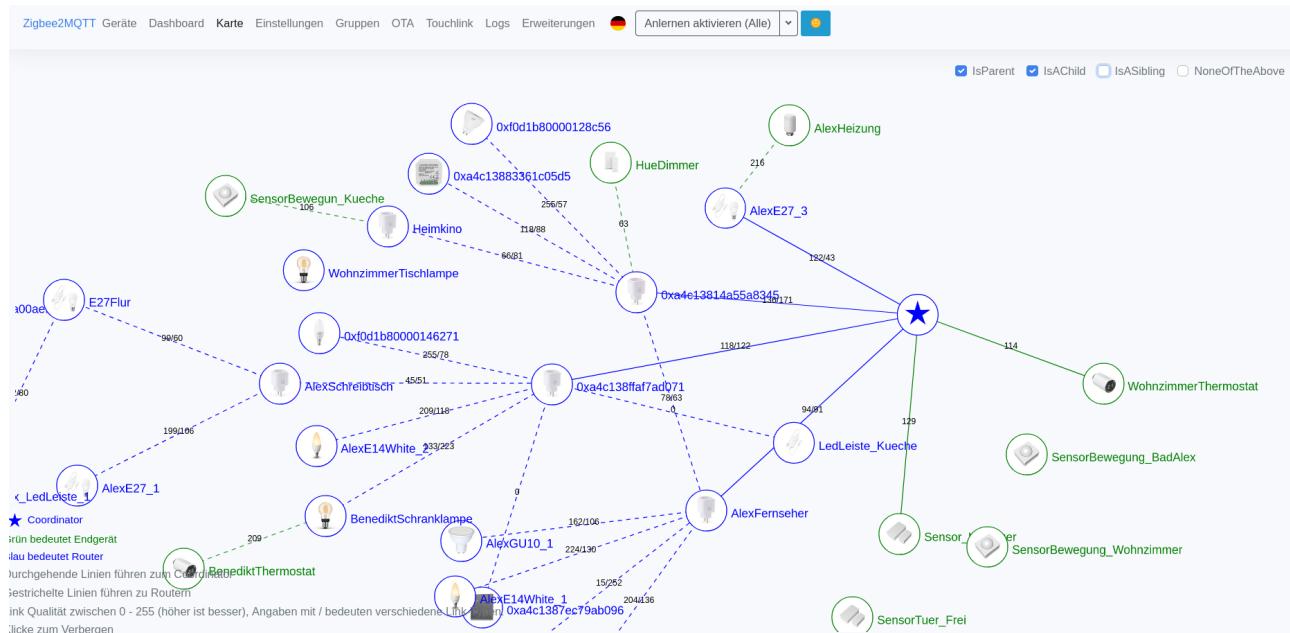


Abbildung 3.7: zigbee2mqtt Netzwerkvisualisierung

Das Netzwerk lässt sich in einer dynamischen Übersicht visualisieren. Hier die aktiv genutzten Verbindung zwischen den Geräten.

### TI CC Firmware

Eine Firmware für die Texas Instruments Chips, um diese als Koordinator einsetzen zu können. Die Firmware basiert auf dem Z-Stack von Texas Instruments. Sie wird fertig kompiliert in dem Git-Repo von zigbee2mqtt angeboten. Sie kann auf die USB-Koordinatoren per USB geflasht werden, der Einsatz eines Launchpads ist nicht notwendig. Eine Anleitung findet sich auf der Homepage von zigbee2mqtt.

Zur Veranschaulichung der Funktionsweise, ein Ausschnitt aus der API Dokumentation:

**3.1.4.9 ZDP\_SimpleDescReq()**

This call will build and send a Simple Descriptor Request.

**Prototype**

```
afStatus_t ZDP_SimpleDescReq( zAddrType_t *dstAddr, uint16 nwkAddr, byte epIntf, byte SecuritySuite );
```

**Parameter Details**

**DstAddr** - The destination address.

**nwkAddr** – Known 16 bit network address.

**epIntf** – wanted application's endpoint/interface.

**SecuritySuite** - Type of security wanted on the message.

**Return**

**afStatus\_t** – This function uses AF to send the message, so the status values are described in ZStatus\_t in ZComDef.h.

Abbildung 3.8: Z-Stack API Auszug

In diesem Beispiel wird beschrieben, wie man einen SimpleDescriptor-Request an ein Zigbee-Device versendet. Dieser Aufruf ist entsprechend parametrierbar, und wird zur Abfrage der verfügbaren Endpunkte eines Gerätes nach dessen Beitritt in das Netzwerk abgefragt.

Die Firmware, die auf dem Koordinator zum Einsatz kommt basiert auf einem Beispiel Projekt für einen ZigBee Koordinator von Texas Instruments. Dieses ist im SDK Kit des TI CC2652 Chips enthalten, und kann mit dem Texas Instruments Code Compose Studio bearbeitet werden. Die Firmware kann selbst kompiliert werden. Dazu kann ein Git Patch von den Maintainern von zigbee2mqtt angewendet werden. Zusätzlich können nun eigene Änderungen in den Code einfließen. Der Patch ist unter folgender URL zu finden. [https://github.com/Koenkk/Z-Stack-firmware/blob/master/coordinator/Z-Stack\\_3.x.0/firmware.patch](https://github.com/Koenkk/Z-Stack-firmware/blob/master/coordinator/Z-Stack_3.x.0/firmware.patch). Die Änderungen, die vorgenommen werden können hier nachgelesen werden. Eine Anleitung zum kompilieren liegt ebenfalls im Git Repository ab.

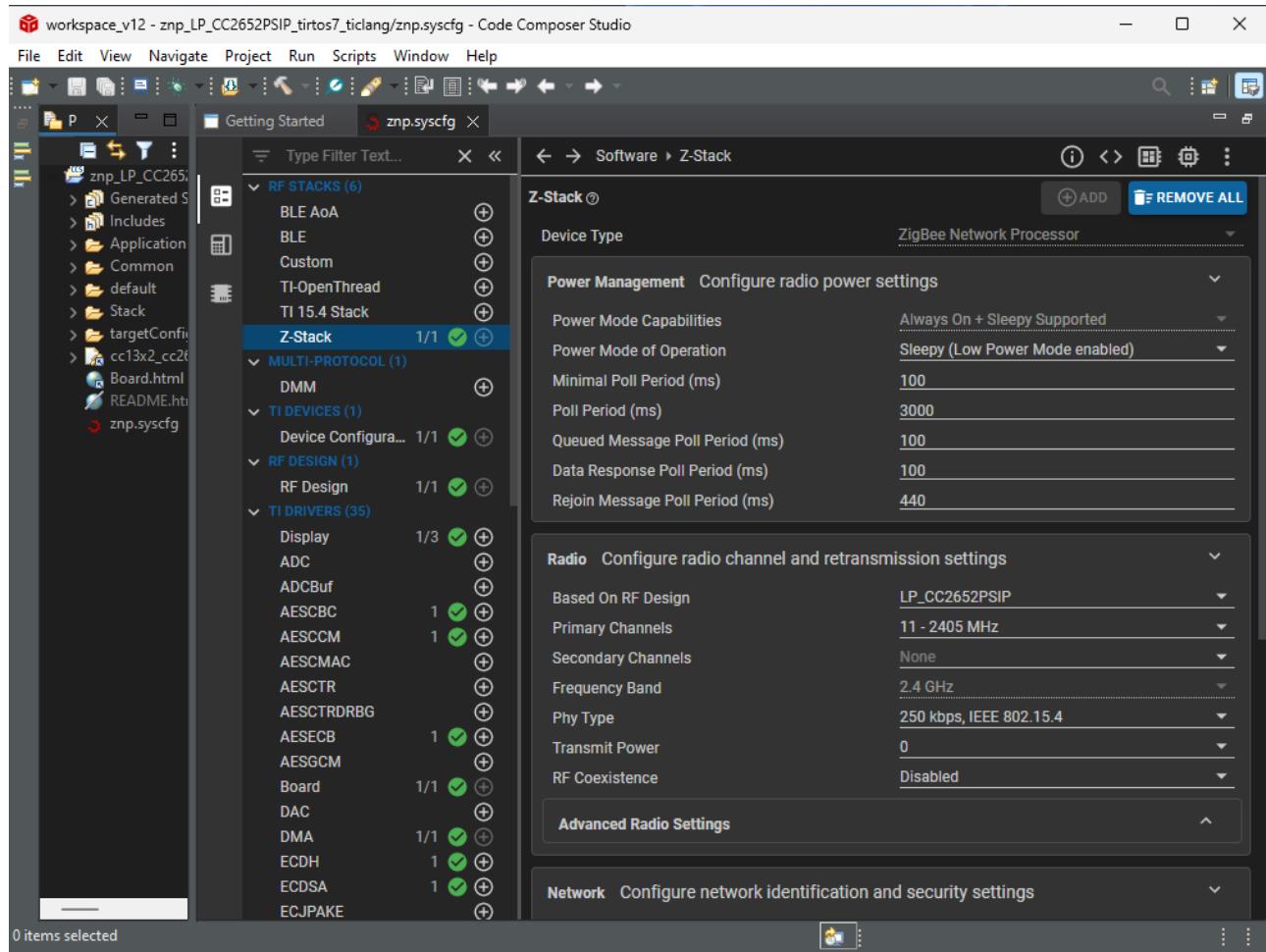


Abbildung 3.9: TI Code Composer Studio

In der Abbildung ist das geöffnete Projekt im Code Composer Studio mit den verschiedenen Konfigurationen die vorgenommen werden können.

### 3.5.5 MQTT

MQTT [Com19] ist ein Protokoll, um Nachrichten zwischen Teilnehmern in einem IP Netzwerk auszutauschen. Alle Nachrichten werden unter einem definierten Topic an einen zentralen Messagebroker gesendet. Teilnehmer können Topics abonnieren. Der Broker verwaltet eine Liste mit allen Teilnehmern sowie deren abonnierten Topics. Wird eine entsprechende Nachricht an den Broker gesendet, werden alle Abonennten des Topics entsprechend informiert. In dem Versuch wird der quelloffene MQTT Broker "mosquitto" eingesetzt.

### 3.5.6 Wireshark

Wireshark ist eine quelloffene Anwendung um Datenstöße mitzuschneiden und zu untersuchen. Wireshark selbst nutzt standardmäßig "npcap" um Datenverkehr auf Netzwerkkarten aufzuzeichnen. Es ist möglich über andere Schnittstellen Wireshark Datenströme zur Verfügung zu stellen. Zu diesem

Zweck können ausführbare Dateien in einen Ordner „.../extcap“ abgelegt werden, welche Paketströme zurückliefern.

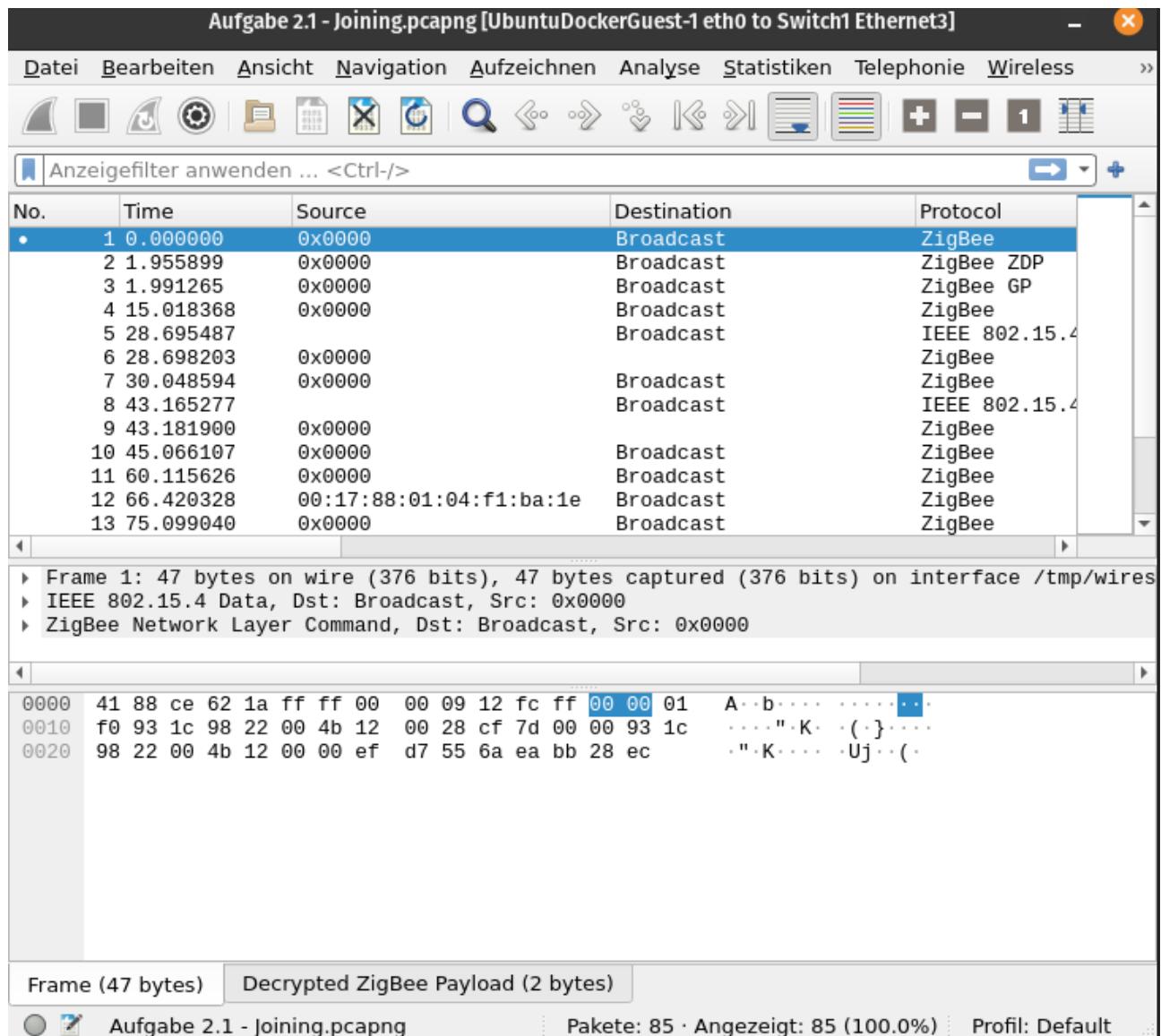


Abbildung 3.10: Wireshark

### 3.5.7 Ansible

Ansible ist ein Werkzeug zur Automatisierung. Arbeitsabläufe lassen sich strukturiert in YAML (yet another markup language) definieren. Dies ist eine alternative zum schreiben von Shell Scripten. Ansible kann Aufgaben auf dem lokalem System und auf Remotesystemen ausführen. Aufgaben können in Rollen zusammengefasst werden. Eine Rolle kann einem Host wie folgt zugewiesen werden:

```

1 - name: Deploy the Lab
2   hosts: localhost
3   roles:
4     - DeployDocker
5     - DeployLabUtils

```

Die Rollen “DeployDocker“ und “DeployLabUtils“ umfassen eine Menge von Aufgaben zur Installation notwendiger Komponenten und weitere Vorbereitungen für den Praktikumsversuch. Diese Rollen werden “localhost“, also dem ausführendem System selbst zugewiesen. Aus technischer Sicht lädt Ansible parametrierte Pythonmodule auf den jeweiligen Host per SCP und führt diese dort aus.

# Kapitel 4

## Versuchsaufbau

### 4.1 Einleitung

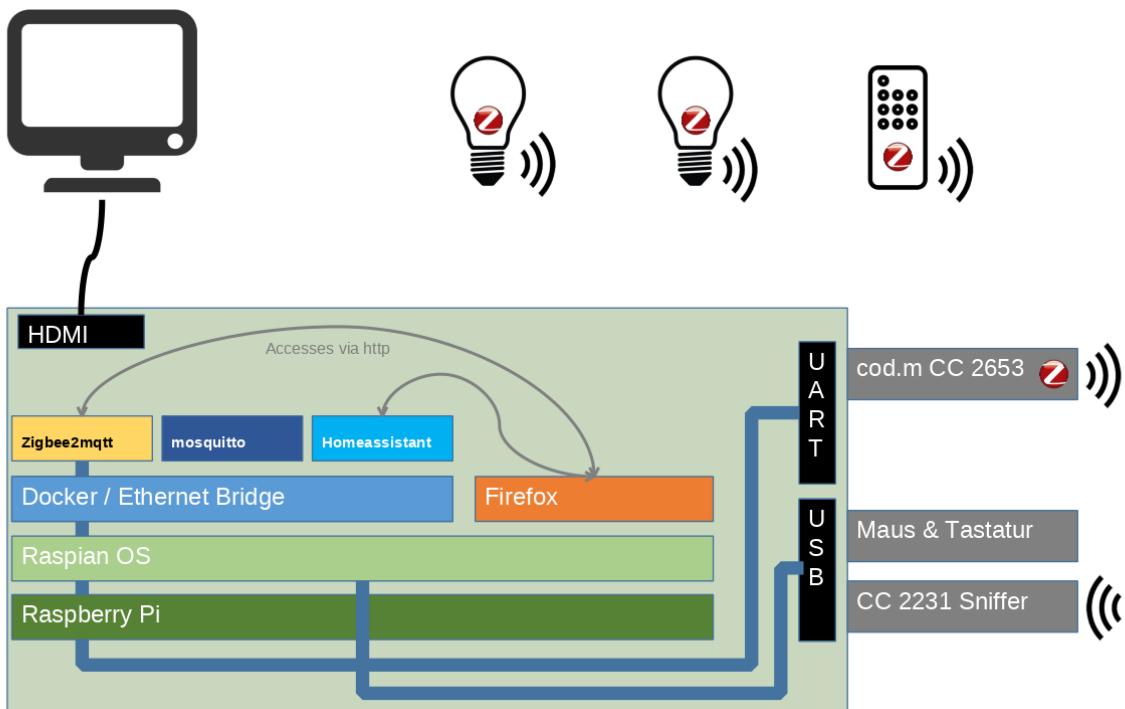


Abbildung 4.1: Versuchsaufbau

In dieser Abbildung wird der schematische Versuchsaufbau gezeigt. Die drei Anwendungen werden als Docker Container ausgeführt. Sie kommunizieren untereinander über ein eigenes Docker Netzwerk. Dies ist eine von Docker verwaltete Linux Bridge. Nur der NGINX Reverse Proxy hat zwei Ports die auf die Host Schnittstelle gemappt werden. Die Webfrontends sind damit über den lokal installierten Browser über das Loopback-Interface erreichbar. Damit ist der Raspberry Applikationsserver und Versuchs-PC zugleich.

Es werden entsprechende Namen in der lokalen "hosts" Konfigurationsdatei hinterlegt, um lokal Do-

mainnamen auflösen zu können.

Der cod.m Zigbeecontroller wird direkt an den Docker Container durchgereicht. Der Sniffer Stick ist regulär am Host angeschlossen.

## 4.2 Containerverwaltung

### 4.2.1 Nginx Proxy

```

1 proxy:
2   container_name: nginx
3   image: jwilder/nginx-proxy:alpine
4   networks:
5     - backbone
6   ports:
7     - 80:80
8     - 443:443
9   volumes:
10    - ./NGINX/proxy/conf.d:/etc/nginx/conf.d:rw
11    - ./NGINX/proxy/vhost.d:/etc/nginx/vhost.d:rw
12    - ./NGINX/proxy/html:/usr/share/nginx/html:rw
13    - ./NGINX/proxy/certs:/etc/nginx/certs:ro
14    - /etc/localtime:/etc/localtime:ro
15    - /var/run/docker.sock:/tmp/docker.sock:ro
16   restart: unless-stopped

```

Der Proxy basiert auf einem Image des Proxys NGINX [Wil22]. Vorteil dieses erweiterten NGINX ist es, dass dieser automatisiert seine Konfigurationen verwaltet. Dafür wird bei einem Service eine Umgebungsvariable gesetzt. Über den “docker.sock“ erfährt der Proxy ob Container gestartet werden und kann auf deren Umgebungsvariablen zugreifen. Wird ein Container mit der Umgebungsvariable “VIRTUAL\_HOST=z2m.local“ gestartet, wird automatisch eine Weiterleitung für alle Anfragen mit dem Header “z2m.local“ auf den Container eingerichtet. Alle Servicecontainer sind in einer eigenen L2-Domäne und von außen nicht direkt erreichbar. Der Proxy Container ist der einzige, dem externe Ports zugewiesen werden. Dies wird durch entsprechende “iptables“ Einträge umgesetzt.

```
ansible@raspberrypi:~ $ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD DROP
-P OUTPUT ACCEPT
-N DOCKER
-N DOCKER-ISOLATION-STAGE-1
-N DOCKER-ISOLATION-STAGE-2
-N DOCKER-USER
-A FORWARD -j DOCKER-USER
-A FORWARD -j DOCKER-ISOLATION-STAGE-1
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o docker0 -j DOCKER
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
-A FORWARD -o br-89a3bb3d47e4 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o br-89a3bb3d47e4 -j DOCKER
-A FORWARD -i br-89a3bb3d47e4 ! -o br-89a3bb3d47e4 -j ACCEPT
-A FORWARD -i br-89a3bb3d47e4 -o br-89a3bb3d47e4 -j ACCEPT
-A DOCKER -d 172.18.0.3/32 ! -i br-89a3bb3d47e4 -o br-89a3bb3d47e4 -p tcp -m tcp --dport 443 -j ACCEPT
-A DOCKER -d 172.18.0.3/32 ! -i br-89a3bb3d47e4 -o br-89a3bb3d47e4 -p tcp -m tcp --dport 80 -j ACCEPT
-A DOCKER-ISOLATION-STAGE-1 -i docker0 ! -o docker0 -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -i br-89a3bb3d47e4 ! -o br-89a3bb3d47e4 -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -j RETURN
-A DOCKER-ISOLATION-STAGE-2 -o docker0 -j DROP
-A DOCKER-ISOLATION-STAGE-2 -o br-89a3bb3d47e4 -j DROP
-A DOCKER-ISOLATION-STAGE-2 -j RETURN
-A DOCKER-USER -j RETURN
```

Abbildung 4.2: Raspberry iptables

Die Regeln in der Tabelle “DOCKER“ werden durch Docker geschrieben. Eigene Regeln können in der Tabelle “DOCKER-USER“ definiert werden.

```
ansible@raspberrypi:~ $ sudo netstat -tulpn | grep docker
tcp        0      0 0.0.0.0:443           0.0.0.0:*               LISTEN      7726/docker-proxy
tcp        0      0 0.0.0.0:80            0.0.0.0:*               LISTEN      7764/docker-proxy
tcp6       0      0 :::::443             ::::*                  LISTEN      7736/docker-proxy
tcp6       0      0 :::::80              ::::*                  LISTEN      7777/docker-proxy
```

Abbildung 4.3: Raspberry netstat

In dieser Ausgabe ist erkennbar, dass Docker auf die Ports 80 und 443 lauscht.

### 4.2.2 zigbee2mqtt

```
1  zigbee2mqtt:
2    container_name: zigbee2mqtt
3    image: koenkk/zigbee2mqtt
4    networks:
5      - backbone
6    volumes:
7      - ./Z2M/data:/app/data
8    devices:
9      - /dev/ttyAMA0:/dev/ttyAMA0
10   restart: always
11   environment:
12     - VIRTUAL_HOST=z2m.local
13     - VIRTUAL_PORT=8080
14   group_add:
15     - dialout
```

Dem Container wird ebenfalls das Netzwerk “backbone“ zugewiesen. Ein Verzeichnis mit Konfigurationen und persistenten Datensätzen wird auf den Host gemountet. Wenn der gemountete Pfad

außerhalb des Containers nicht existiert, wird der bestehende Ordner aus dem Container kopiert. Existiert der Pfad, wird der Ordner vom Host in den Container gemountet. Bei Linux können Geräte über das selbe Verfahren wie Verzeichnisse adressiert werden. “- /dev/ttyUSB0:/dev/ttyACM0“ reicht den ZigBee Adapter an den Docker Container weiter. In den Umgebungsvariablen wird dem Proxy noch mitgeteilt, unter Welcher URL er erreichbar sein soll und auf welchem Port der Webserver läuft. Die Gruppe “dialout“ ist notwendig, damit der Container Zugriffsrechte auf die serielle Schnittstelle des Hosts erhält.

Die zentrale Konfigurationsdatei von “zigbee2mqtt“:

```

1  homeassistant: true
2  permit_join: false
3  mqtt:
4    base_topic: zigbee2mqtt
5    server: mqtt://mosquitto:1883
6  serial:
7    port: /dev/ttyACM0
8  frontend:
9    port: 8080
10   host: 0.0.0.0
11   url: https://z2m.local
12 advanced:
13   homeassistant_legacy_entity_attributes: false
14   legacy_api: false
15   legacy_availability_payload: false
16 device_options:
17   legacy: false

```

Es wird der “Homeassistant“ Modus aktiviert. Damit werden die Nachrichten an den MQTT Broker für Homeassistant verständlich gestaltet. Das Beitreten neuer Geräte ist standardmäßig deaktiviert und muss explizit erlaubt werden. Des Weiteren wird ein MQTT Server angegeben, sowie ein “base-topic“ definiert. Docker löst Containernamen in Dockernetzwerken zu IP-Adressen auf, sodass hier als Server der entsprechende Containernamen angegeben werden kann. Im weiteren wird der Pfad angegeben, auf den der Zigbee2MQTT Adapter gemountet wurde, sowie entsprechende Einstellung für den Webserver gesetzt.

### 4.2.3 mosquitto

```

1  mosquitto:
2    container_name: mosquitto
3    image: eclipse-mosquitto:latest
4    networks:
5      - backbone
6    restart: always
7    deploy:
8      resources:
9        limits:
10       memory: 125M
11    volumes:
12      - ./mosquitto/config:/mosquitto/config
13      - ./mosquitto/data:/mosquitto/data
14      - ./mosquitto/log:/mosquitto/log

```

Als MQTT Broker wird “mosquitto“ eingesetzt. Die Konfigurationen wurden auch hier entsprechend auf den Host gemountet. Als Konfigurationsdatei wird das Standardtemplate verwendet, welches nur an zwei entsprechenden Stellen modifiziert ist.

```

1 | ...
2 | allow_anonymous true
3 | ...
4 |
5 | ...
6 | listener 1883
7 | ...

```

Es wird der Zugriff von nicht authentifizierten Geräten erlaubt. Dies stellt kein Risiko dar, da der Container nur innerhalb des “backbone“ Netwerkes sowie vom Host selber erreichbar ist. Zusätzlich wird der Port definiert, auf dem der MQTT Service läuft. 1883 ist der standard Port für MQTT.

## 4.3 Namensauflösung

Für eine lokale Namensauflösung werden die Hosts in die “\etc\hosts“ eingetragen. Dies wird automatisch in der Ansible Rolle “DeployLab“ gemacht.

```

1 | - name: Add Hosts Entries
2 |   become: True
3 |   lineinfile:
4 |     path: /etc/hosts
5 |     line: 127.0.0.1 z2m.local

```

## 4.4 Anwendungen

Für den Versuch wird weiterhin ein Webbrower sowie Wireshark benötigt. Die beiden Anwendungen werden per Ansible installiert:

```

1 | - name: Install required Packages
2 |   become: true
3 |   apt:
4 |     pkg:
5 |       - wireshark
6 |       - firefox
7 |     state: latest
8 |     update_cache: true

```

# Kapitel 5

## Musterlösung

### 5.1 Einleitung

Alle Wireshark Mitschnitte liegen als “\*.pcapng“ Datei bei.

### 5.2 Aufgabe 2.1 - Joining einer Phillips Hue Lampe

1 0.000000	0x0000	Broadcast	ZigBee	47 Link Status
2 1.955899	0x0000	Broadcast	ZigBee ZDP	48 Permit Join Request
3 1.991265	0x0000	Broadcast	ZigBee GP	51 ZCL Green Power: GP Proxy Commissioning Mode, Seq: 3
4 15.018265	0x0000	Broadcast	ZigBee	47 Link Status
5 28.695487	0x0000	Broadcast	IEEE 802.15.4	10 Beacon Request
6 28.698293	0x0000		ZigBee	28 Beacon, Src: 0x0000, EPID: TexasIns_00:22:98:1c:93
7 30.048594	0x0000	Broadcast	ZigBee	47 Link Status
8 43.165277	0x0000	Broadcast	IEEE 802.15.4	19 Beacon Request
9 43.181990	0x0000		ZigBee	28 Beacon, Src: 0x0000, EPID: TexasIns_00:22:98:1c:93
10 45.066107	0x0000	Broadcast	ZigBee	47 Link Status
11 60.115626	0x0000	Broadcast	ZigBee	47 Link Status
12 66.420328	00:17:88:01:04:f1:..	Broadcast	ZigBee	35 ZCL Touchlink: Scan Request, Seq: 0
13 75.099640	0x0000	Broadcast	ZigBee	47 Link Status
14 98.832344	00:17:88:01:04:f1:..	Broadcast	ZigBee	35 ZCL Touchlink: Scan Request, Seq: 0
15 105.168305	0x0000	Broadcast	ZigBee	47 Link Status
16 106.996289	0x0000	Broadcast	IEEE 802.15.4	10 Beacon Request
17 117.075472	0x0000	Broadcast	IEEE 802.15.4	10 Beacon Request
18 117.081943	0x0000		ZigBee	28 Beacon, Src: 0x0000, EPID: TexasIns_00:22:98:1c:93
19 118.217271	00:17:88:01:04:b9:3.. 0x0000		IEEE 802.15.4	21 Association Request, FFD
20 118.217621	0x0000		IEEE 802.15.4	5 Ack
21 118.715359	00:17:88:01:04:b9:3.. 0x0000		IEEE 802.15.4	18 Data Request
22 118.716879	0x0000		IEEE 802.15.4	5 Ack
23 118.734622	00:12:4b:00:22:98:1.. 00:17:88:01:04:b9:3..	IEEE 802.15.4	27 Association Response, PAN: 0x1a62 Addr: 0x1ed6	
24 118.734965	0x0000		IEEE 802.15.4	5 Ack
25 118.851927	0x0000	0x1ed6	ZigBee	73 Transport Key
26 118.852013	0x0000		IEEE 802.15.4	5 Ack
27 118.883596	0x1ed6	Broadcast	ZigBee ZDP	57 Device Announcement, Nwk Addr: 0x1ed6, Ext Addr: PhilipsL_01:04:b9:35:9d
28 119.181461	0x0000		ZigBee ZDP	48 Node Descriptor Request, Nwk Addr: 0x1ed6

Abbildung 5.1: Wireshark Ausschnitt - Joining einer Lampe Teil 1

Im ersten Teil ist der Verbindungsauflauf zu sehen. Mit Paket 2 erlaubt der Koordinator allen Mitgliedern die Aufnahme weiterer Geräte. Paket 5 ist ein Beacon der Lampe, welcher den Beitrittswunsch signalisiert. Auf diesen Antwortet der Koordinator jeweils mit einem Beacon Response. Nun folgt ein Association Request sowie Data Request der Lampe (19). In der Association Response (23) bekommt die Lampe Ihre kurze Adresse zugewiesen. In Paket 25 wird der Transport Key zur verschlüsselten Kommunikation übermittelt.

26 118.852013	0x1ed6	Broadcast	IEEE 802.15.4	5 Ack
27 119.883596	0x1ed6	ZigBee ZDP	57 Device Announcement, Nwk Addr: 0x1ed6, Ext Addr: PhilipsL_01:04:b9:35:9d	
28 119.181401	0x0000	ZigBee ZDP	48 Node Descriptor Request, Nwk Addr: 0x1ed6	
29 119.181754		IEEE 802.15.4	5 Ack	
30 119.194677	0x1ed6	0x0000	62 Node Descriptor Response, Nwk Addr: 0x1ed6, Status: Success	
31 119.194780		ZigBee ZDP	5 Ack	
32 119.217496	0x0000	0x1ed6	45 APS: Ack, Dst Endpt: 0, Src Endpt: 0	
33 119.217934		ZigBee	5 Ack	
34 119.234410	0x0000	0x1ed6	48 Active Endpoint Request, Nwk Addr: 0x1ed6	
35 119.234767		ZigBee ZDP	5 Ack	
36 119.245166	0x1ed6	0x0000	52 Active Endpoint Response, Nwk Addr: 0x1ed6, Status: Success	
37 119.246083		ZigBee	5 Ack	
38 119.249120	0x0000	0x1ed6	45 APS: Ack, Dst Endpt: 0, Src Endpt: 0	
39 119.249546		IEEE 802.15.4	5 Ack	
40 119.264849	0x0000	0x1ed6	49 Simple Descriptor Request, Nwk Addr: 0x1ed6, Endpoint: 11	
41 119.266061		ZigBee ZDP	5 Ack	
42 119.276379	0x1ed6	0x0000	74 Simple Descriptor Response, Nwk Addr: 0x1ed6, Status: Success	
43 119.276723		ZigBee ZDP	5 Ack	
44 119.315102	0x0000	0x1ed6	45 APS: Ack, Dst Endpt: 0, Src Endpt: 0	
45 119.316512		ZigBee	5 Ack	

Abbildung 5.2: Wireshark Ausschnitt - Joining einer Lampe Teil 2

Im Anschluss fragen die Geräte gegenseitig ihre aktiven Endpunkte über “Node Descriptor“ Nachrichten ab. Die einzelnen Endpunkte werden im Anschluss per “Active Endpoint Requests“ abgefragt und per “Simple Descriptor“ Nachrichten beschrieben.

### 5.3 Aufgabe 2.2 - Schalten einer Lampe

1 0.000000	0x1ed6	Broadcast	ZigBee	50 Link Status
2 15.080857	0x1ed6	Broadcast	ZigBee	50 Link Status
3 17.077725	0x0000	Broadcast	ZigBee	50 Link Status
4 30.244635	0x1ed6	Broadcast	ZigBee	50 Link Status
5 32.127576	0x0000	Broadcast	ZigBee	50 Link Status
6 41.051630	0x0000	Broadcast	ZigBee	51 Many-to-One Route Request, Dst: 0xffffc, Src: 0x0000
7 41.373276	0x0000	Broadcast	ZigBee	51 Many-to-One Route Request, Dst: 0xffffc, Src: 0x0000
8 41.942384	0x0000	0x1ed6	ZigBee HA	48 ZCL OnOff: On, Seq: 3
9 41.943854			IEEE 802.15.4	5 Ack
10 41.952768	0x1ed6	0x0000	ZigBee	55 Route Record, Dst: 0x0000
11 41.952973			IEEE 802.15.4	5 Ack
12 42.045627	0x1ed6	0x0000	ZigBee HA	58 ZCL: Default Response, Seq: 3
13 42.045848			IEEE 802.15.4	5 Ack
14 42.592298	0x0000	0x1ed6	ZigBee HA	48 ZCL OnOff: Off, Seq: 4
15 42.592633			IEEE 802.15.4	5 Ack
16 42.601640	0x1ed6	0x0000	ZigBee HA	50 ZCL: Default Response, Seq: 4
17 42.603117			IEEE 802.15.4	5 Ack
18 45.323498	0x1ed6	Broadcast	ZigBee	50 Link Status
19 47.112757	0x0000	Broadcast	ZigBee	50 Link Status

Abbildung 5.3: Wireshark Ausschnitt - Schalten einer Lampe

Hier ein ein Ein- und Ausschaltvorgang einer Lampe zu sehen. Zum Schalten wird jeweils ein “ZigBee Kommando Frame“ versendet.

### 5.4 Aufgabe 3 - Joining einer Fernbedienung über die Lampe

1 0.000000	0x0000	0x1ed6	ZigBee ZDP	48 Permit Join Request
2 0.000388			IEEE 802.15.4	5 Ack
3 0.010651	0x1ed6	0x0000	ZigBee ZDP	47 Permit Join Response, Status: Success
4 0.010981			IEEE 802.15.4	5 Ack

Abbildung 5.4: Wireshark Ausschnitt - Permit Join über Lampe

Hier erteilt der Koordinator der Lampe die Erlaubnis neue Geräte in das Netzwerk aufzunehmen.

22 64.684235		Broadcast	IEEE 802.15.4	10 Beacon Request
23 64.687071	0x1ed6	ZigBee	IEEE 802.15.4	28 Beacon, Src: 0x1ed6, EPID: TexasIns_00:22:98:1c:93
24 65.623693	00:17:88:01:04:a5:32:74	0x1ed6	IEEE 802.15.4	21 Association Request, FFD
25 65.823921		ZigBee	IEEE 802.15.4	5 Ack
26 66.322839	00:17:88:01:04:a5:32:74	0x1ed6	IEEE 802.15.4	18 Data Request
27 66.326754		ZigBee	IEEE 802.15.4	5 Ack
28 66.327112	00:17:88:01:04:b9:35:9d	00:17:88:01:04:a5:32:74	IEEE 802.15.4	27 Association Response, PAN: 0x1a62 Addr: 0xabf3
29 66.328573		ZigBee	IEEE 802.15.4	5 Ack
30 66.338862	0x1ed6	0x0000	ZigBee	55 Route Record, Dst: 0x0000
31 66.339195		ZigBee	IEEE 802.15.4	5 Ack
32 66.428475	0x1ed6	0x0000	ZigBee	60 Update Device
33 66.428570		ZigBee	IEEE 802.15.4	5 Ack
34 66.460232	0x0000	0x1ed6	ZigBee	102 Transport Key
35 66.460313		ZigBee	IEEE 802.15.4	5 Ack
36 66.471942	0x1ed6	0xabf3	ZigBee	73 Transport Key
37 66.483953		ZigBee	IEEE 802.15.4	5 Ack
38 66.498071	0xabf3	Broadcast	ZigBee ZDP	57 Device Announcement, Nwk Addr: 0xabf3, Ext Addr: PhilipsL_01:04:a5:32:74
39 66.581020	0xabf3	Broadcast	ZigBee ZDP	57 Device Announcement, Nwk Addr: 0xabf3, Ext Addr: PhilipsL_01:04:a5:32:74
40 67.225573	0xabf3	Broadcast	ZigBee ZDP	57 Device Announcement, Nwk Addr: 0xabf3, Ext Addr: PhilipsL_01:04:a5:32:74
41 79.773506	0x0000	Broadcast	ZigBee	59 Route Request, Dst: 0xabf3, Src: 0x0000
42 79.785345	0xabf3	0x0000	ZigBee	77 Route Reply, Responder: 0xabf3, Originator: 0x0000
43 79.785466			IEEE 802.15.4	5 Ack
44 79.968391	0xabf3	Broadcast	IEEE 802.15.4	53 Data, Dst: Broadcast, Src: 0xabf3, Bad FCS
45 88.061515	0x0000	Broadcast	ZigBee	59 Route Request, Dst: 0xabf3, Src: 0x0000

Abbildung 5.5: Wireshark Ausschnitt - Beitritt über Lampe

Die hat zur Folge, dass die Lampe auf die Beacon-Requests der Lampe mit Beitrittswunsch reagiert. Diese Beacons dienen dazu, das Netzwerk der neuen Lampe bekannt zu machen. Mit Nachricht 24 erfragt die neue Lampe einen Beitritt in das Netzwerk. Der Lampe wird eine Kurze Adresse sowie die PAN-ID von der Lampe im Netzwerk zugewiesen. Anschließend macht die bestehende Lampe das neue Gerät dem Koordinator bekannt. Dieser versendet im Anschluss den Netzwerk Schlüssel. Dieser wird in einem Tunnel verschlüsselt bis zu dem letzten Gerät vor dem neuen Teilnehmer übermittelt. Im Anschluss erfolgt das Interview des neuen Gerätes, dies ist identisch zu Aufgabe 2.

## 5.5 Aufgabe 4 - Binding der Fernbedienung

67 3.100659	0x0000	0xb9f7	ZigBee ZDP	71 Bind Request, On/Off (Cluster ID: 0x0006) Src: PhilipsL_01:04:f1:ba:1e, Dst: PhilipsL_01:04:b9:35:9d
68 3.100857			IEEE 802.15.4	5 Ack
69 3.105976	0xb9f7	0x1ed6	IEEE 802.15.4	12 Data Request
70 3.106212			IEEE 802.15.4	5 Ack
71 3.111593	0x0000	0xb9f7	ZigBee ZDP	71 Bind Request, Level Control (Cluster ID: 0x0008) Src: PhilipsL_01:04:f1:ba:1e, Dst: PhilipsL_01:04:b9:35:9d

Abbildung 5.6: Wireshark Ausschnitt - Binden einer Lampe 1

Die Bindung wird durch ein Bind-Request von dem Koordinator an die Fernbedienung initiiert. Dies erfolgt durch Paket 67 und 71.

91 3.384860	0xb9f7	0x0000	ZigBee ZDP	47 Bind Response, Status: Success
92 3.385329			IEEE 802.15.4	5 Ack
93 3.629394	0xb9f7	0x1ed6	IEEE 802.15.4	12 Data Request
94 3.631014			IEEE 802.15.4	5 Ack
95 4.130406	0xb9f7	0x1ed6	IEEE 802.15.4	12 Data Request
96 4.130760			IEEE 802.15.4	5 Ack
97 4.659775	0xb9f7	0x1ed6	IEEE 802.15.4	12 Data Request
98 4.660101			IEEE 802.15.4	5 Ack
99 5.161620	0xb9f7	0x1ed6	IEEE 802.15.4	12 Data Request
100 5.161831			IEEE 802.15.4	5 Ack
101 5.662359	0xb9f7	0x1ed6	IEEE 802.15.4	12 Data Request

Abbildung 5.7: Wireshark Ausschnitt - Binden einer Lampe 2

Die Fernbedienung bestätigt eine erfolgreiche Bindung durch eine Bind-Response Nachricht mit einem "Success" im Payload. Im Anschluss beginnt die Fernbedienung mit Polling, um zu überprüfen ob weitere Nachrichten für sie verfügbar sind.

4 0.011087			IEEE 802.15.4	5 Ack
• 5 0.178557	0xa19f	0x1ed6	ZigBee HA	48 ZCL OnOff: Off, Seq: 39
6 0.178859			IEEE 802.15.4	5 Ack
<hr/>				
▶ Frame 5: 48 bytes on wire (384 bits), 48 bytes captured (384 bits) on interface /tmp/wireshark_extcap_cc2531P0QS41, id 0				
▶ IEEE 802.15.4 Data, Dst: 0x6af3, Src: 0xa19f				
▶ Frame Control Field: 0x8861, Frame Type: Data, Acknowledge Request, PAN ID Compression, Destination Addressing Mode: Short/16-bit, Frame Version: Sequence Number: 232				
Destination PAN: 0x1a62				
Destination: 0x6af3				
Source: 0xa19f				
[Extended Source: Ember_00:13:f4:e1:d2 (00:0d:6f:00:13:f4:e1:d2)]				
[Origin: 5]				
▶ TI CC24xx-format metadata: FCS OK				
▼ ZigBee Network Layer Data, Dst: 0x1ed6, Src: 0xa19f				
▶ Frame Control Field: 0x0248, Frame Type: Data, Discover Route: Enable, Security Data				
Destination: 0x1ed6				
Source: 0xa19f				
Radius: 30				
Sequence Number: 183				
▶ ZigBee Security Header				
▼ ZigBee Application Support Layer Data, Dst Endpt: 11, Src Endpt: 2				
▶ Frame Control Field: Data (0x40)				
.... .00 = Frame Type: Data (0x0)				
.... 00.. = Delivery Mode: Unicast (0x0)				
..0. .... = Security: False				
.1.. .... = Acknowledgement Request: True				
0... .... = Extended Header: False				
Destination Endpoint: 11				
Cluster: On/Off (0x0006)				
Profile: Home Automation (0x0104)				
Source Endpoint: 2				
Counter: 97				
▼ ZigBee Cluster Library Frame				
▶ Frame Control Field: Cluster-specific (0x11)				
Sequence Number: 39				
Command: Off (0x00)				

Abbildung 5.8: Wireshark Ausschnitt - Schalten einer Lampe mit Binding

Die ZCL Nachricht wird nun direkt von der Fernbedienung an die Lampe als Unicast geschickt.

## 5.6 Aufgabe 5 - Gruppenbildung

1 0.000000	0x0000	0x1ed6	ZigBee HA	51 ZCL Groups: Add Group, Seq: 20
2 0.000256			IEEE 802.15.4	5 Ack
3 0.010438	0x1ed6	0x0000	ZigBee HA	51 ZCL Groups: Add Group Response, Seq: 20
4 0.010741			IEEE 802.15.4	5 Ack
5 0.504999	0x1ed6	Broadcast	ZigBee	53 Link Status
6 2.794740	0x0000	0x6af3	ZigBee HA	51 ZCL Groups: Add Group, Seq: 21
7 2.796357			IEEE 802.15.4	5 Ack
8 2.805364	0x6af3	0x0000	ZigBee HA	51 ZCL Groups: Add Group Response, Seq: 21
9 2.805577			IEEE 802.15.4	5 Ack

Abbildung 5.9: Wireshark Ausschnitt - Hinzufügen zu einer Gruppe

Zuerst wird die Gruppe angelegt. Dafür wird an die jeweiligen Geräte eine Add Group Nachricht gesendet. Diese wird mit einer jeweiligen Response Nachricht bestätigt.

18 2.937848	0x0000	0xfd47	ZigBee ZDP	60 Bind Request, Level Control (Cluster ID: 0x0008) Src
19 2.939253			IEEE 802.15.4	5 Ack
20 2.946993	0xfd47	0x0000	ZigBee ZDP	47 Bind Response, Status: Success
21 2.947222			IEEE 802.15.4	5 Ack
<hr/>				
▶ Frame 18: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface /tmp/wireshark_extcap_cc2531U7E41, id 0				
▶ IEEE 802.15.4 Data, Dst: 0xfd47, Src: 0x0000				
▶ ZigBee Network Layer Data, Dst: 0xfd47, Src: 0x0000				
▶ ZigBee Application Support Layer Data, Dst Endpt: 0, Src Endpt: 0				
▼ ZigBee Device Profile, Bind Request, Level Control (Cluster ID: 0x0008) Src: Ember_00:14:17:da:50, Dst: 0x000f				
Sequence Number: 163				
Source: Ember_00:14:17:da:50 (00:0d:6f:00:14:17:da:50)				
Source Endpoint: 2				
Cluster: 0x0008 (Level Control)				
Address Mode: Group (1)				
Destination: 0x000f				

Abbildung 5.10: Wireshark Ausschnitt - Schalten einer Lampe über eine Gruppe

Im Anschluss wird die Gruppe an den Endpunkt der Fernbedienung gebunden. Der Fernbedienung wird diesmal als Ziel die Gruppen Broadcast Adresse 0x000F sowie der Adressmodus Gruppe mitgeteilt. Auch dies wird von der Fernbedienung wieder bestätigt. Das Reporting welches im Anschluss konfiguriert wird ist eine Eigenart von zigbee2mqtt. Damit wird der Koordinator bei Zustandsänderungen der Lampe informiert, um diese Informationen beispielweise an eine zentrale Heimautomatisierung weiterzugeben.

Beim Schalten wird das ZCL Kommando nun an die Gruppenadresse 0x000F versendet. Die zu schaltenden Geräte überprüfen selbst ob sie in der Gruppe sind oder nicht. Prinzipiell erhält jedes Gerät im Netzwerk das Kommando.

# Kapitel 6

## Life Cycle Management

### 6.1 Deployment

In diesem Kapitel geht es um die Pflege, die Bereitstellung sowie das Zurücksetzen des Praktikumversuchs. Sämtliche Schritte wurden mit Ansible-Playbooks automatisiert.

Folgende Schritte müssen ausgeführt werden, um ein RaspberryPi vorzubereiten.

- Installation von Raspbian OS 32 Bit
- Anlegen eines Users
- Klonen des GitHub Repositorys
- Installation von Ansible
- UART auf Raspberry vorbereiten
- Ansible Playbook Ausführen

Das Repository kann mit Git und folgendem befehl geklont werden:

```
1 | git clone https://github.com/nlab4hsrm/ZigbeeHSRM.git
```

Am besten wird das Repository direkt in den Homefolder des Users geklont, der später den Versuch durchführt. Damit kann dieser selbständig den Versuch zurücksetzen. Im Anschluss liegt ein Ordner “ZigbeeHSRM“ ab. In diesem sind die Playbooks und der LabGuide enthalten.

Zusätzlich muss noch Ansible installiert werden:

```
1 | sudo apt install ansible
```

Bevor der Versuch ausgerollt werden kann, ist es zusätzlich notwendig manuell die UART Schnittstelle des Raspberries zu konfigurieren. Dazu wird zuerst das Tool rasp-config aufgerufen.

```
1 | sudo raspi-config
```

Hier muss unter dem Menüpunkt “3 Interface Options“ die Konfiguration für “I6 Serial Port“ gestartet werden. Im ersten Schritt wird der Zugriff auf die Shell über den seriellen Port deaktiviert. In zweitem Schritt wird die Serielle Schnittstelle dann aktiviert.

Anschließend wird Bluetooth deaktiviert. Dies passiert durch hinzufügen folgender Zeile in die Datei “/boot/config.txt“

```
1 | dtoverlay=disable-bt
```

Im letzten Schritt muss noch der Dienst “hciuart“ deaktiviert werden, da dieser ansonsten auf die Schnittstelle zugreift

```
1 | systemctl disable hciuart
```

Danach muss der RaspberryPi neu gestartet werden!

Zum Ausrollen werden die Rollen “Docker“ und “ZigbeeLab“ zugewießen. Dies geschieht durch die DeployLab.yaml.

```
1 | ansible-playbook ~/ZigbeeHSRM/playbooks/DeployLab.yaml -K -e "channel=<channelnumber>"
```

Im Anschluss muss das jeweilige Passwort des Users angegeben werden. Der User benötigt Root Rechte.

Anschließend wird

- Docker installiert.
- Die /etc/hosts Einträge gesetzt.
- Die Docker Umgebung vorbereitet.
- Die Versuchsumgebung mit allen Komponenten gestartet.

## 6.2 Zurücksetzen des Versuchs

Zum zurücksetzen wird das Playbook “ResetLab“ ausgeführt.

```
1 | ansible-playbook ~/ZigbeeHSRM/playbooks/ResetLab.yaml -K -e "channel=<channelnumber>"
```

Hinweis: Das Playbook funktioniert nur in der 32-Bit Version von RaspbianOS. Für 64-Bit muss ein anderes Docker Repository im Playbook angegeben werden. Es wird dann arm64 anstelle von armhf benötigt. Dies ist allerdings nicht getestet.

## 6.3 Update der eingesetzten Software

Die RaspberryOS Pakete können mit dem Paketmanager apt aktuell gehalten werden. Docker zieht initial die Container aus Dockerhub. Mit folgendem Befehl überprüft Docker das Repository auf aktuelle Container.

```
1 | docker compose pull /srv/docker-compose.yaml
```

Anschließend können die Container mit folgendem Befehl neu gestartet werden.

```
1 | docker compose up -d /srv/docker-compose.yaml
```

Sämtliche Software wird onehin bei jedem Zurücksetzen des Versuchen aktualisiert.

## 6.4 Troubleshooting

Es empfiehlt sich bei Problemen sich den Log der Container anzuschauen.

```
1 | docker attach <container-name>
```

In der Regel ist es am sinnvollsten, mit dem Log von zigbee2mqtt zu starten. Dieser ist erfahrungsgemäßig bei einer Fehlersuche sehr hilfreich.

Ein gängiger Fehler ist, dass der Container zigbee2mqtt regelmäßig neustartet. Diese Information erhält man mit einem

```
1 | docker ps
```

Die Gründe dafür können vielfältig sein. Am häufigsten ist ein Problem mit dem Zugriff auf die UART Schnittstelle. Ebenfalls kann ein belegter Funkkanal oder eine doppelte PAN-ID in der Umgebung Grund für Probleme sein. Der Grund kann aus der Log des Containers ermittelt werden:

```
1 | docker attach zigbee2mqtt
```



# **Kapitel 7**

## **Anhänge**

### **.1 LabGuide**

Hochschule RheinMain  
Fachbereich ITE  
Studiengang EE-CS

## **LabGuide**

# **ZigBee Versuch auf Basis von zigbee2mqtt**

verfasst von

**Benedikt HEUSER**  
Matrikelnummer 105320

am

25.04.2022

# **Inhaltsverzeichnis**

# Kapitel 1

## Versuchsdurchführung

### 1.1 Versuchsaufbau

Folgende Hardware sollte sich in Ihrer Versuchskiste befinden. Bitte überprüfen sie dies vor Beginn des Versuches.

- RaspberryPi 3 mit eingesetzter microSD-Karte
- CC2531 Sniffer Stick
- cod.m ZigBee CC2652P2 Raspberry Pi Module
- 2 x Phillips Hue White E27
- 1 x Phillips Hue dimmer switch
- HDMI Kabel
- Ethernet Kabel

Ein cod.m Zigbee Modul sollte bereits auf Ihrem Raspberry montiert sein.

In diesem Praktikumsversuch wird ein kleinen ZigBee Netzwerk errichtet. Verschiedene Funktionen des Protokolls werden getestet und aufgezeichnet.

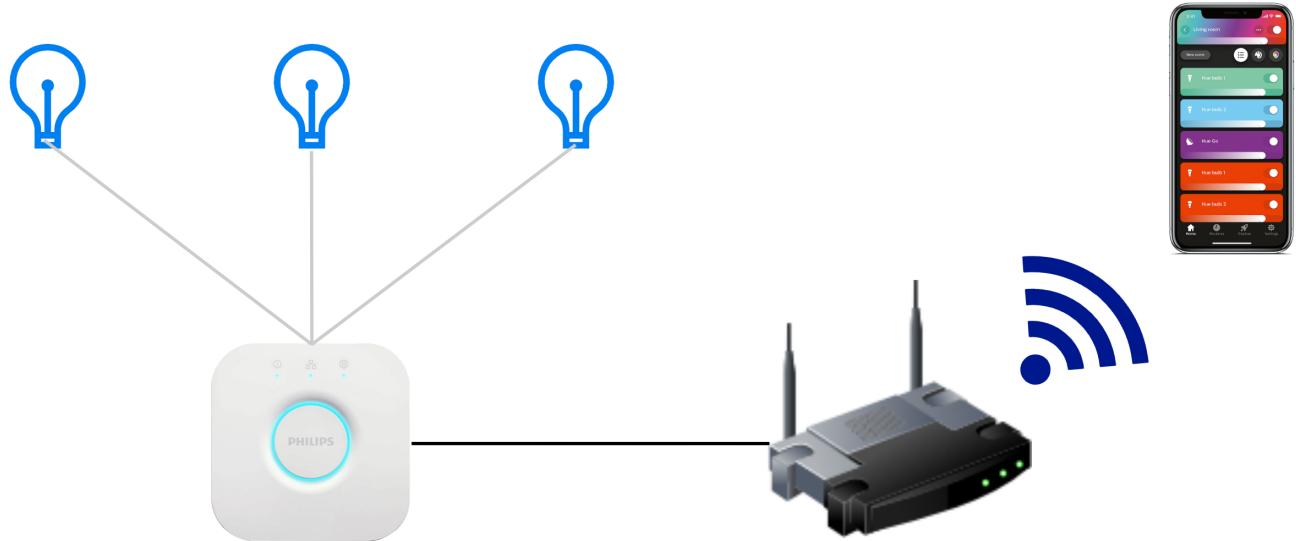


Abbildung 1.1: Phillips Hue

Das ZigBee Protokoll, welches in diesem Versuch untersucht wird, kommt klassischerweise in hier gezeigtem Szenario zum Einsatz. Eine Bridge, hier von Phillips, verbindet sich per ZigBee mit Smarthome Komponenten wie Lampen. Diese Bridge fungiert als ZigBee Koordinator. Eine Smartphone App greift per REST-API auf die Bridge zu. Mit dieser App können nun die Lampen gesteuert werden.

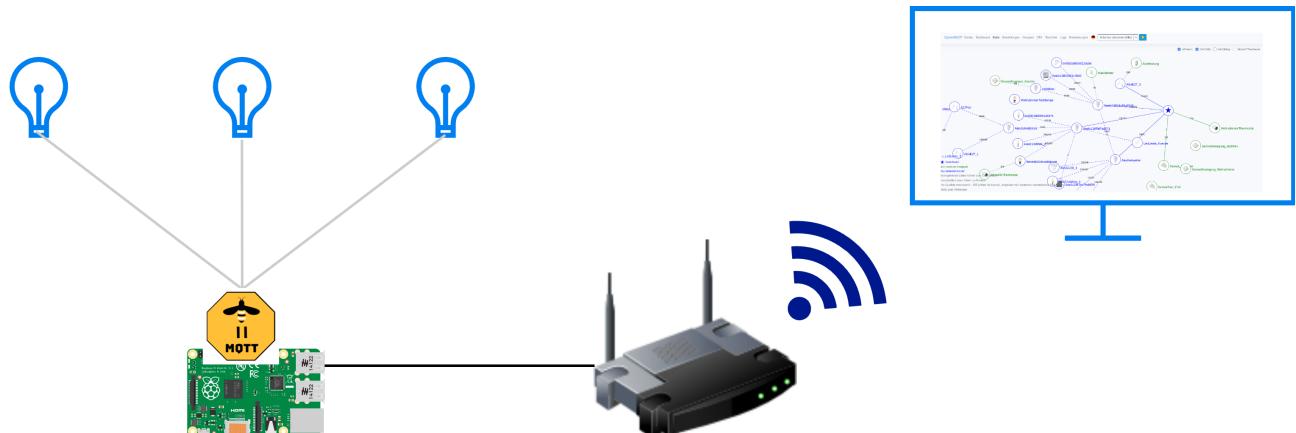


Abbildung 1.2: Versuchsaufbau

In diesem Versuch wird als Koordinator anstelle der Phillips Bridge ein RaspberryPi eingesetzt. Dieser benötigt dafür das cod.m Zigbee Modul sowie eine entsprechende Software. In dem Versuch wird zigbee2mqtt eingesetzt. Anstelle einer App wird zigbee2mqtt über ein Webinterface administriert. Dieses lässt sich wie in diesem Versuch gezeigt lokal aufrufen oder aber auch nach außen verfügbar machen. In der Regel wird zigbee2mqtt an einer weitere Anwendung wie Homeassistant gekoppelt, um Lampen und andere Geräte ansprechend steuern zu können. Die Kommunikation zwischen diesen beiden Applikation würde per MQTT stattfinden.

Der RaspberryPi dient in diesem Versuch als Koordinator und Versuchs-PC zugleich.

## 1.2 Aufgabenstellungen

Bitte arbeiten sie die folgenden Aufgabenstellungen durch. Fertigen sie im Anschluss einen Versuchsbericht an. Beantworten sie die anhängenden Fragen implizit oder explizit.

### 1.2.1 Aufgabe 1 - Vorbereitungen

#### Vorbereitung

- Schließen sie an den RaspberryPi Monitor, Tastatur, Maus sowie den Sniffer-Stick an. Durch Anschluss der Stromversorgung startet der RaspberryPi automatisch.
- Starten sie den Versuch, in dem sie ein Konsolenfenster öffnen und folgenden Befehl absetzen.

```
1 |      > ansible-playbook playbooks/reset-lab.yaml -e "channel=<Gruppennummer>"
```

Setzen sie den gewünschten Kanal für <Gruppennummer> ein. Der Start des Labs kann einige Minuten dauern. Dieser Befehl muss nur einmal zu Beginn des Praktikums ausgeführt werden. Bei einem Neustart des RaspberryPis startet der Versuch automatisch.

- Starten sie ein Konsolenfester und überprüfen mit folgendem Befehl, ob die Container ausgeführt werden:

```
1 |      > docker ps
```

Es sollten 3 Container im Status “Running“ sein. Sollte dies nicht der Fall sein, beispielsweise ein periodisches Neustarten des Containers “zigbee2mqtt“, lesen sie die weiteren Schritte in den FAQs nach.

- Starten sie den Webbrowser Firefox und Navigieren zu der Seite:

```
1 |      https://z2m.local
```

- Starten sie Wireshark über das Symbol auf dem Desktop oder alternativ per “sudo wireshark“. Bei den verfügbaren Schnittstellen sollte sich eine “TI CC2231“ Schnittstelle finden. Über das vorangestellte Zahnrad-Symbol können sie den abzuhörenden Kanal einstellen. Stellen sie den eben gewählten Zigbee Kanal ein. (Gruppennummer)

- Setzen sie alle ZigBee Komponenten des Versuchs zurück. Dafür drücken sie auf der Rückseite der Fernbedienung die Reset Taste. Die Phillips Hue Lampen können ausschließlich per Touchlink zurückgesetzt werden. Drücken sie dazu die äußeren Taste der Fernbedienung und halten diese direkt an die Lampe. Warten sie bis die Lampe mehrmals aufblinkt. Dies kann mehrere Versuche mit verschiedenen Fernbedienungspositionen benötigen. Schalten sie im Anschluss die beiden Lampen wieder aus.

Alternativ bietet zigbee2mqtt ebenfalls eine Touchlink Funktionalität. Bringen sie dafür die Lampe

direkt an den Koordinator und Scannen in dem zigbee2mqtt Webinterface nach Touchlink Geräten. Im Anschluss kann die Lampe hier zurückgesetzt werden. Auch hier wird das durch mehrmaliges blinken der Lampe signalisiert.

The screenshot shows the Zigbee2MQTT dashboard with the following details:

- Header: Zigbee2MQTT, Gerät, Dashboard, Karte, Einstellungen, Gruppen, OTA, Touchlink, Logs, Erweiterungen, German flag, An.
- Middle section: "1 Touchlink Gerät entdeckt." with a refresh icon.
- Search bar: Suchkriterien eingeben.
- Table header: #, IEEE Adresse, Gerätename, Kanal.
- Table data:
 

#	IEEE Adresse	Gerätename	Kanal
1	0x0017880104b9359d	0x0017880104b9359d	20
- Action icons: A blue warning icon and a red edit icon.

Abbildung 1.3: Touchlink - Lampen zurücksetzen

### Hinweis

Alle Aufgaben sollen mit Wireshark mitgeschnitten werden. Lesen sie die Aufgabenstellung erst durch und machen sie sich den Ablauf klar. Versuchen sie das Zeitfenster des Wireshark Mitschnitts kurz halten, und in dieser Zeit nur die in der Aufgabenstellung explizit beschriebenen Aktionen durchzuführen.

## 1.2.2 Aufgabe 2 - Joining einer Phillips Hue Lampe

- Schalten sie eine der beiden Lampen ein.
- Starten sie nun ein Wireshark Mitschnitt und erlauben anschließend in zigbee2mqtt das Anlernen von Geräten. Sobald zigbee2mqtt ein erfolgreiches Interview gemeldet hat, beenden sie den Capture Vorgang. Die Lampe signalisiert durch ein kurzes blinken ein erfolgreiches Interview.

The screenshot shows the Zigbee2MQTT dashboard with the following details:

- Header: Zigbee2MQTT, Gerät, Dashboard, Karte, Einstellungen, Gruppen, OTA, Touchlink, Logs, Erweiterungen, German flag.
- Action bar: Anlernen aktivieren (Alle) (highlighted with a red box), a dropdown menu, and a blue button.

Abbildung 1.4: Zigbee Anlernen aktivieren

**Aufgabe**

Speichern sie den Wireshark Capture ab als “<Gruppe> - ZigbeeLab - Aufgabe 2.1“. Beantworten sie die Fragen in Ihrem Versuchsbericht.

c) Navigieren sie nun zur Übersichtsseite der Lampe. Diese sollte ähnlich wie folgende Seite aussehen:

The screenshot shows the Zigbee2MQTT interface with the following details:

Attribut	Wert
Gerätename	0xf0d1b8000013821d
Beschreibung	N/A
Zuletzt gesehen	Deaktiviert
Verfügbarkeit	Router
Geräte-Typ	A60 TW Z3
Zigbee Modell	LEDVANCE
Zigbee Hersteller	SMART+ classic E27 TW
Beschreibung	Unterstützt
Unterstützungsstatus	0xf0d1b8000013821d
IEEE Adresse	0x30FD
Netzwerk Adresse	Sep 29 2021
Firmware-Datum	01056400
Firmware-Version	OSRAM
Hersteller	AC10787
Modell	Wahr
Spannungsversorgung	
Interview erfolgreich	

At the bottom left, there are three icons: a blue square with a white checkmark, an orange square with a white circular arrow, and a red square with a white trash can. The first icon is highlighted with a red box.

Abbildung 1.5: Zigbee Device Übersicht

- d) Vergeben sie in der Übersichtsseite der Lampe einen nutzerfreundlichen Namen. Dies geschieht über den blauen Button im unteren Teil der Übersicht.  
e) Dimmen und schalten sie die Lampe über die Weboberfläche. Die ist unter dem Reiter “Dashboard“ möglich. Starten sie einen weiteren Capture Vorgang und schneiden in diesem einen Schaltvorgang mit.

**Aufgabe**

Speichern sie den Wireshark Capture ab als “<Gruppe> - ZigbeeLab - Aufgabe 2.2“.  
Beantworten sie die Fragen in Ihrem Versuchsbericht.

### 1.2.3 Aufgabe 3 - Joining eines Gerätes über ein anderes Gerät

Für diese Aufgabe sollte eine Lampe mit dem Koordinator verbunden sein. Die zweite Lampe wird nun über die Lampe dem Netzwerk hinzugefügt. Aus diesem Grund wird es nur der Lampe erlaubt ein neues Gerät aufzunehmen.

- Schalten sie die zweite Lampe ein und starten einen Wireshark Mitschnitt.
- Erlauben sie den Beitritt neuer Geräte explizit für die bereits verbundene Phillips Lampe. Ein erfolgreiches anlernen wird auch hier in der Weboberfläche und durch ein blinken der Lampe signalisiert.

The screenshot shows the Zigbee2MQTT dashboard with the 'Erweitert' tab selected. On the right, a sidebar displays a list of devices under the heading 'Alle'. The first device listed is 'Coordinator' with the ID '0x7cb03eaa00aeef846'. Below it are three other device IDs: '0xa4c13814a55a8345', '0xa4c1387ec79ab096', and '0xa4c1380336100e4f'. A red arrow points from the text 'ID Ihrer Lampe' to the input field where the device ID is entered.

Abbildung 1.6: Zigbee Anlernen aktivieren - nur Lampe

**Aufgabe**

Speichern sie den Wireshark Mitschnitt ab als “<Gruppe> - ZigbeeLab - Aufgabe 3“.  
Beantworten sie die Fragen in Ihrem Versuchsbericht.

- Sehen sie sich die Netzwerkübersicht unter dem Reiter “Karte“ an. Aktivieren sie nur den Haken “isParent“

- Fügen sie die Fernbedienung Ihrem Netzwerk hinzu. Gehen sie dabei wie bisher vor. Setzen sie die Batterie in die Lampe ein und erlauben die Aufnahme neuer Geräte.

Ihre Übersichtsseite sollte wie folgt aussehen.

The screenshot shows the Zigbee2MQTT dashboard with the 'Erweitert' tab selected. On the right, a sidebar displays a list of devices under the heading 'Alle'. The first device listed is 'Coordinator' with the ID '0x7cb03eaa00aeef846'. Below it are three other device IDs: '0xa4c13814a55a8345', '0xa4c1387ec79ab096', and '0xa4c1380336100e4f'. A red arrow points from the text 'ID Ihrer Lampe' to the input field where the device ID is entered.

Abbildung 1.7: Zigbee Anlernen aktivieren - nur Lampe

### 1.2.4 Aufgabe 4 - Binding der Fernbedienung

- a) Navigieren sie in der Weboberfläche zu der Übersicht Ihrer Lampe. Dort finden sie einen Reiter "binden".
- b) Starten sie ein Wireshark Mitschnitt. Binden sie den Endpunkt der Fernbedienung mit dem Endpunkt Ihrer Fernbedienung. Binden sie die Cluster "OnOff" und "LevelCtrl". Betätigen sie direkt nach dem Anlegen der Bindung eine Taste auf der Fernbedienung. Zum Batteriesparen "schläft" diese periodisch und kann in dieser Zeit keine Bindungsanfragen annehmen. Der Hersteller geht davon aus, das die Fernbedienung gewöhnlich per Touchlink gebunden wird. Warten sie bis zigbee2mqtt ein erfolgreiches Binding meldet.

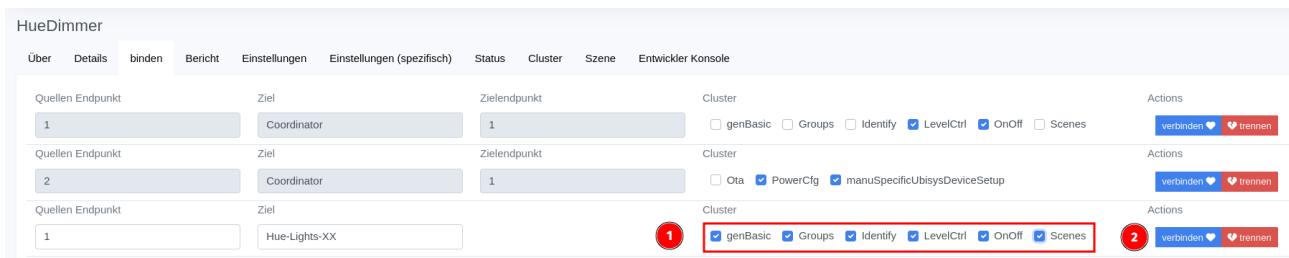


Abbildung 1.8: Zigbee Group Binding

Achten sie darauf alle die gezeigten Cluster anzuwählen. Jedes Cluster wird einzeln gebunden.

- c) Schalten sie nun die Lampe mit der Fernbedienung ein und aus.

#### Hinweis

Speichern sie den Wireshark Capture ab als "<Gruppe> - ZigbeeLab - Aufgabe 4".

Beantworten sie die Fragen in Ihrem Versuchsbericht.

- d) Entfernen sie im Anschluss dieses Binding. Drücken sie auch hier eine Taste der Fernbedienung damit diese geweckt wird.

### 1.2.5 Aufgabe 5 - Gruppenbildung

- a) Navigieren sie in der Weboberfläche zu dem Reiter "Groups".
- b) Legen sie eine Gruppe mit dem Namen "Hue-Lights-<Gruppe>" an.
- c) Starten sie einen Wireshark Mitschnitt. Editieren sie nun die Gruppe. Fügen sie die Endpunkte der beiden Lampen, die zum Steuern verwendet werden, der Gruppe hinzu.

#### Aufgabe

Speichern sie den Wireshark Capture ab als "<Gruppe> - ZigbeeLab - Aufgabe 5".

Beantworten sie die Fragen in Ihrem Versuchsbericht.

d) Navigieren sie nun wieder zur Binding-Übersicht der Fernbedienung. Binden sie die Fernbedienung nun mit der soeben angelegten Gruppe. Achten sie auch hier darauf eine Taste der Fernbedienung zu betätigen. e) Starten sie einen Wireshark Mitschnitt. Schalten sie die Gruppe mit der Fernbedienung ein und aus.

#### Aufgabe

Speichern sie den Wireshark Capture ab als “<Gruppe> - ZigbeeLab - Aufgabe 5.1“.

Beantworten sie die Fragen in Ihrem Versuchsbericht.

### 1.2.6 Fragen

#### Aufgabe 2

##### Fragen

1. Untersuchen sie den **Beacon-Request**

- Erläutern sie den Frametype und den Command Identifier.
- Erläutern die Ziel- und Quelladresse und was sie daraus erschließen können.

Hinweis: Wireshark Filter: `frame.protocols == "wpan"`

2. Wie teilt der Koordinator den umliegenden Geräten mit, dass er dem Netzwerk den Beitritt weiterer Geräte erlaubt ?

- Zu welchem Frametype gehört der Beacon und durch welchen Wert wird er spezifiziert?
- Welche Ziel- und Quelladressen werden verwendet?

Hinweis: Wireshark Filter: `frame.protocols == "wpan:zbee_beacon"`

4. Welchen Wert hat das Feld „Association Permit“ im letzten Beacon des Koordinators und wie ist dieser Wert zu interpretieren?

5. Untersuchen Sie den **Association Request** der Lampe an den Koordinator.

- Welchen Wert hat das Feld “Allocate Address “ und wie ist dieser zu interpretieren?
- Welchen Wert hat das Feld “Device Type“ und wie ist dieser zu interpretieren?

Hinweis: Wireshark Filter: `frame.protocols == "wpan"`

7. Untersuchen die den **Data Request** von der Lampe an den Koordinator.

- Erläutern Sie die Funktion dieser Nachricht.
- Mit welchem Kommando frame antwortet der Koordinator auf den Data-Request?
- Welche Werte besitzen die Felder „Short Address“ und „Association Status“?

Hinweis: Wireshark Filter: `frame.protocols == "wpan"`

**8. Untersuchen Sie einen IEEE802.15.4. Ack-Frame.**

-Welche Adressfelder werden benutzt?

-Wie findet die Zuordnung zum Daten- oder Kommandoframe statt, der durch die Ack bestätigt wird?

-Werden grundsätzlich alle Frames bestätigt?

**10. Wie lautet die letzte Nachricht, bei der 64-bit-MAC-Adressen verwendet werden?**

-Wie lautet die erste Nachricht, bei der die 16-Bit-Kurzadresse der beigetretenen Lampe verwendet wird?

**11. Was ist die letzte Nachricht, die auf dem NWL-Layer unverschlüsselt übertragen wird?****12. Erläutern Sie den Zweck der Transport-Key-Nachricht.**

-Wie lautet der Frametype des 802.15.4-Frames, in dem die Transport-Key-Nachricht transportiert wird?

-Wie lautet der ZigBee-NWK Frametype des Frames?

-Treffen Sie möglichst genaue Aussagen zum in der Transport-Key übertragenen Schlüssel (Schlüsseltype).

-Erläutern Sie, wie die Transport-Key-Nachricht kryptographisch gesichert ist.

-Interpretieren Sie den Inhalt des Radius Feldes im NWK-Frame, das die TransportKeyNachricht enthält!

Hinweis: Wireshark Filter: zbee\_aps.cmd.id == 0x05

**13. Erläutern Sie, den Zweck des versendeten Active-Endpoint-Requests und des SimpleDescriptor-Requests.**

-Beschreiben Sie die Information, die in den entsprechenden Response-Nachrichten enthalten ist.

-Wie stellt zigbee2mqtt die Information dar?

-Welche Endpoints werden für den Austausch der untersuchten Request- und ResponseNachrichten verwendet? Interpretieren Sie dies!

Hinweis: Wireshark Filter: frame.protocols == "wpan:zbee\_nwk:zbee\_aps:zbee\_zdp"

**14. Durch welche ZigBee-Frames werden die Schaltvorgänge übertragen?****15. Beschreiben Sie möglichst genau, durch welche Headerfelder die Schaltvorgänge definiert sind!****16. Welche Endpoints werden für die Schaltvorgänge benutzt? Woher hat der Koordinator Kenntnis über die in der Lampe verwendeten Endpoints?**

**Aufgabe 3****Fragen**

1. Erläutern Sie den Zweck der **Permit-Join-Request** Nachricht.

-An welche ZigBee-NWKZieladresse wird die Nachricht versendet?

-Erläutern Sie das wichtigste Headerfeld!

Hinweis: Wireshark Filter: zbee\_aps.profile == 0x0000

2. Welchem Zweck dient die **Update Device** Nachricht?

-Wer ist Absender und wer ist Empfänger?

-Welche Adresse steht im Feld „Device Address“?

Hinweis: Wireshark Filter: zbee\_aps.type == 0x1

3. Von welchem Device erhält die zweite Lampe ihre 16 Bit Kurzadresse und wie lautet sie?

4. Wie viele **Transport-Key** Nachrichten wurden ausgetauscht?

-Erläutern Sie wer jeweils der Absender und wer der Empfänger ist.

-Versuchen Sie die den Vorgang zu erklären und gehen Sie dabei auf das Kommandoframe „Tunnel“ ein. -Wie sind die Transport-Key Nachrichten kryptographisch gesichert? -Was sind die wichtigsten Headerfelder des Tunnel-Kommandoframes?

Hinweis: Wireshark Filter: zbee\_aps.type == 0x1

5. Untersuchen Sie die **Device Announcement** Nachricht der zweiten Lampe.

-Welchen Zweck hat sie?

-Schauen Sie sich die „Capability Information“ an. Handelt es sich um ein Full-Function-Device?

-Welcher Wert steht im Feld „AC Power“ und was sagt dieser Wert aus?

Hinweis: Wireshark Filter: zbee\_aps.profile == 0x0000

6. Untersuchen Sie die **Active-Endpoint-Request** Nachricht und ihren Weg vom Koordinator bis zur zweiten Lampe.

-Vergleichen Sie die Adressen im ZigBee-NWKLAYER und im IEEE-Layer und erklären Sie den Zusammenhang.

-An welchem Headerfeld können Sie zweifelsfrei identifizieren, dass es die gleiche Nachricht ist, die nur weitergeleitet wird?

Hinweis: Wireshark Filter: zbee\_aps.profile == 0x0000

7. Untersuchen Sie die **Simple Descriptor Response**- Nachrichten der zweiten Lampe! -Welche Informationen enthält diese Nachricht?

Hinweis: Wireshark Filter: zbee\_aps.profile == 0x0000

8. Erklären sie die Zahlen, welche in der Kartenansicht an den Verbindungen notiert sind.

## Aufgabe 4

### Fragen

1. Untersuchen Sie die **Bind Request**- und die **Bind Response**-Nachricht.

-Was sind jeweils die NWK-Quell- und NWK-Zieladressen? Erläutern Sie, den Inhalt der BindRequest Nachricht.

-Was genau bewirkt die Nachricht? In der Nachricht sind nur 64-Bit Adressen enthalten. Stellt das ein Problem dar?

Hinweis: Wireshark Filter: zbee\_aps.profile == 0x0000

2. Warum wird vom Koordinator kein Bind-Request an die Lampe gesendet?

3. Betrachten Sie die **ZCL: OnOff** Nachricht. Geben Sie die NWK-Quell- und Zieladresse an.

-Welchen Wert hat das On/OFF-Cluster und welches Kommando wird zum Schalten verwendet?

Hinweis: Wireshark Filter: zbee\_zcl.type == 0x1

4. Interpretieren Sie die von der Fernbedienung gesendeten **Data Request** Nachrichten? -Wie groß ist der zeitliche Abstand zwischen zwei Data-Requests? -Was löst eine DataRequest-Nachricht beim Empfänger aus? Geben Sie ein Beispiel. Hinweis: Wireshark Filter: wpan.cmd == 0x04

## Aufgabe 5

### Fragen

1. Verdeutlichen Sie sich den Vorgang in dem Sie die vom Koordinator versendeten Nachricht **Add Group** untersuchen.

-Fassen Sie die wichtigsten Informationen der Nachrichten zusammen.

2. Betrachten Sie die **Add Group Response** Nachricht des Empfängers.

-Welche Information ist enthalten?

-Welcher Zielendpunkt wird im APS-Frame des AddGroup-Befehles verwendet?

-Geben Sie eine Erklärung!

3. Wie lautet die Destination Adresse im ZDP-Header der **Bind Request** Nachricht?

-Welcher Zielendpunkt ist vorhanden?

Hinweis: Wireshark Filter: zbee\_aps.profile == 0x0000

4. Was ist die NWK-Zieladresse der **ZCL-OnOff** Nachricht?

-Um welche Art von Nachricht handelt es sich hierbei?

-Finden Sie die von Ihnen eingestellte Gruppen-ID in der „ZCL OnOff“-Nachricht wieder?

Hinweis: Wireshark Filter: zbee\_zcl.type == 0x1

## Weiterführende Fragen

### Fragen

1. Welchem Zweck dienen die **Link Status** Nachrichten?

-Über welche Anzahl von **Hops** wird diese Nachricht übertragen? -Analysieren Sie exemplarisch einige Link-StatusNachrichten und Interpretieren Sie diese!

2. Untersuchen Sie die **Link Quality Request-** bzw. **Link Quality Response**-Nachrichten.

-Gehen Sie auf den LQI-Wert in der **Link Quality Response** Nachricht und was bedeutet dieser?

3. Interpretieren Sie **Route Request** Nachrichten und zugehörige **Route-Response**-Nachrichten.



## .2 Texas Instruments SimpleLink

# SimpleLink™ Microcontroller Platform



## Key benefits

- Broader portfolio of differentiated 32-bit Arm®-based MCUs
  - Industry-leading low-power consumption
  - On-chip dedicated, integrated security to reduce external security threats
  - Supports more than ten differentiated wired and wireless protocols
- Single software foundation for 100 percent software reuse
  - SDK based on common foundation of drivers, frameworks and libraries
  - Pre-integrated TI-RTOS kernel already deployed in thousands of products across multiple applications
  - POSIX-compliant API ensures compatibility with numerous third-party software components
- Faster development with unified tool chain, training and resources
  - One development environment across the whole portfolio of SimpleLink MCUs
  - Free cloud and offline tools with Code Composer Studio™ Cloud and Desktop
  - SimpleLink Academy offers free, hands-on training
  - 24/7 support through the TI E2E™ community

TI's SimpleLink™ platform offers the broadest portfolio of 32-bit Arm-based microcontrollers (MCUs) with industry-leading features including low power and robustness, and integrated security to support more than ten differentiated wired and wireless protocols. Each device offers developers a number of features to uniquely solve their problems, whether capturing high-precision 16-bit analog signal, enabling more security, achieving a range of over 20 kilometers or making a coin cell last for several years.

However, when it comes to software development, these devices are developed around a single software foundation providing 100 percent code compatibility within the SimpleLink software development kits (SDKs). The SimpleLink MCU SDKs feature common components and device-specific middleware to speed up your time-to-market and provide a unified development experience across the entire SimpleLink MCU portfolio of wired and wireless devices. Intuitive and standardized APIs enable 100% application code portability.

The SDK common foundation of drivers, frameworks and libraries enables developers to both access peripherals via portable and easy-to-use TI Drivers API as well as optimize via lower-level access with DriverLib hardware abstraction layer (HAL). Developers can leverage real-time and multi-tasking operations with the integrated TI-RTOS kernel, or tap into other APIs and OS/kernels with POSIX-compliant APIs. A wide range of plug-ins help developers realize additional connectivity and external functionalities. More information about [SimpleLink SDK and code portability](#).

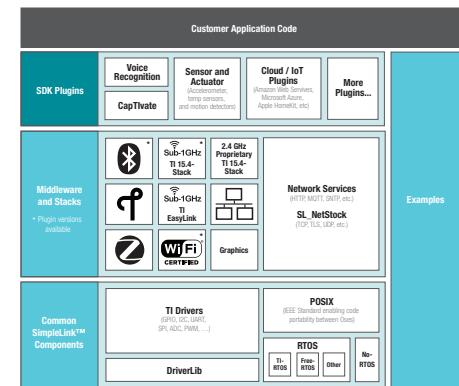
## Software Tools

TI's [unified tool chain](#) delivers faster development with free software tools, training and support to get designs started immediately.

### [Code Composer Studio™ integrated development environment \(IDE\)](#)

- Free, unlimited license
- Powerful and efficient IDE
- Additional support for third-party IDEs and tools including IAR®, Keil and SEGGER.

## SimpleLink MCU SDK



## Cloud-based tools

- Free access
- Seamless integration to traditional offline tools

## SimpleLink Academy

- Dozen of highly curated workshops and chapters
- Help developers ramp quickly on SimpleLink Platform

## SimpleLink MCU portfolio

	SimpleLink™ Ethernet microcontroller	SimpleLink™ Host microcontroller	SimpleLink™ Wi-Fi® Wireless network processor	SimpleLink™ Wi-Fi® Wireless microcontroller	SimpleLink™ Bluetooth® low energy Wireless microcontroller	SimpleLink™ Sub-1 GHz + 2.4-GHz concurrency Wireless microcontroller	SimpleLink™ Sub-1 GHz Wireless microcontroller	SimpleLink™ 2.4-GHz Multi-standard Wireless microcontroller
Product	MSP432E4	MSP432P4	CC3120	CC3220	CC2642R CC2640R2F	CC1352P CC1352R CC1350	CC1312R CC1310	CC2652R
MCU type	Wired MCU	Host MCU	Network processor	Wireless MCU	Wireless MCU	Wireless MCU	Wireless MCU	Wireless MCU
Application	✓	✓	-	✓	✓	✓	✓	✓
Wireless stack + RF	-	-	✓	✓	✓	✓	✓	✓
Wireless and wired technology	Ethernet, USB, CAN and wireless connectivity with SDK plugins	Connectivity with SDK plugins	Wi-Fi®	Wi-Fi	Bluetooth® low energy	Sub-1 GHz + Bluetooth low energy, Zigbee, Thread, or 2.4-GHz proprietary	Sub-1 GHz	Bluetooth low energy, Zigbee, Thread, or 2.4-GHz proprietary
Key differentiation	Integrated Ethernet MAC + PHY and advanced cryptography accelerators	Capture analog signals at up to 16 ENOB using ADC	Network processor with integrated Wi-Fi and Internet protocols	Wi-Fi CERTIFIED™ single-chip MCU with enhanced security	Lowest power BT4.2 and BT5 Flash-based solution	Option for integrated 20 dBm PA enabling longer range	Combine low power and longest range to achieve 20 kms on a coin cell	Future-proof designs with the lowest-power 2.4-GHz multi-standard platform
SimpleLink SDK compatible	✓	✓	✓	✓	✓	✓	✓	✓

## SimpleLink portfolio development kits

	Ethernet MCU	Host MCU	Wi-Fi Wireless network processor	Wi-Fi Wireless MCU	Bluetooth low energy	Sub 1-GHz + 2.4-GHz concurrency	Sub 1-GHz	2.4-GHz multi-standard
Product	MSP432E4	MSP432P4	CC3120	CC3220	CC2642R CC2640R2F	CC1352P CC1352R CC1350	CC1312R CC1310	CC2652R
LaunchPad™	MSP-EXP432E401Y	MSP-EXP432P401R	CC3120BOOST	CC3220SF-LAUNCHXL	LAUNCHXL-CC26X2R1 LAUNCHXL-CC2640R2	LAUNCHXL-CC1352R1 LAUNCHXL-CC1350	LAUNCHXL-CC1312R1 LAUNCHXL-CC1310	LAUNCHXL-CC26X2R1

TI Resource Explorer	GUI Composer	PinMux	EnergyTrace™ technology	SmartRF™ Studio
Cloud-based repository featuring code examples, documentation, training and more	Build graphical interfaces to complement your application	Code-gen utility to generate pin, peripheral and driver configuration code	Optimize your application for the lowest possible power consumption	Utility to configure wireless SimpleLink devices

For more information on TI's SimpleLink MCU platform as well as software and hardware tools, please visit [www.ti.com/simplelink](http://www.ti.com/simplelink)

The platform bar, Code Composer Studio, E2E, EnergyTrace, LaunchPad, SimpleLink and SmartRF are trademarks of Texas Instruments.  
All other trademarks are the property of their respective owners.

## **IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES**

Texas Instruments Incorporated ("TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

**TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.**

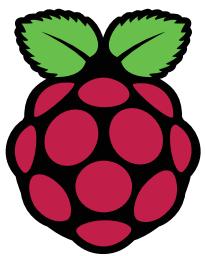
TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

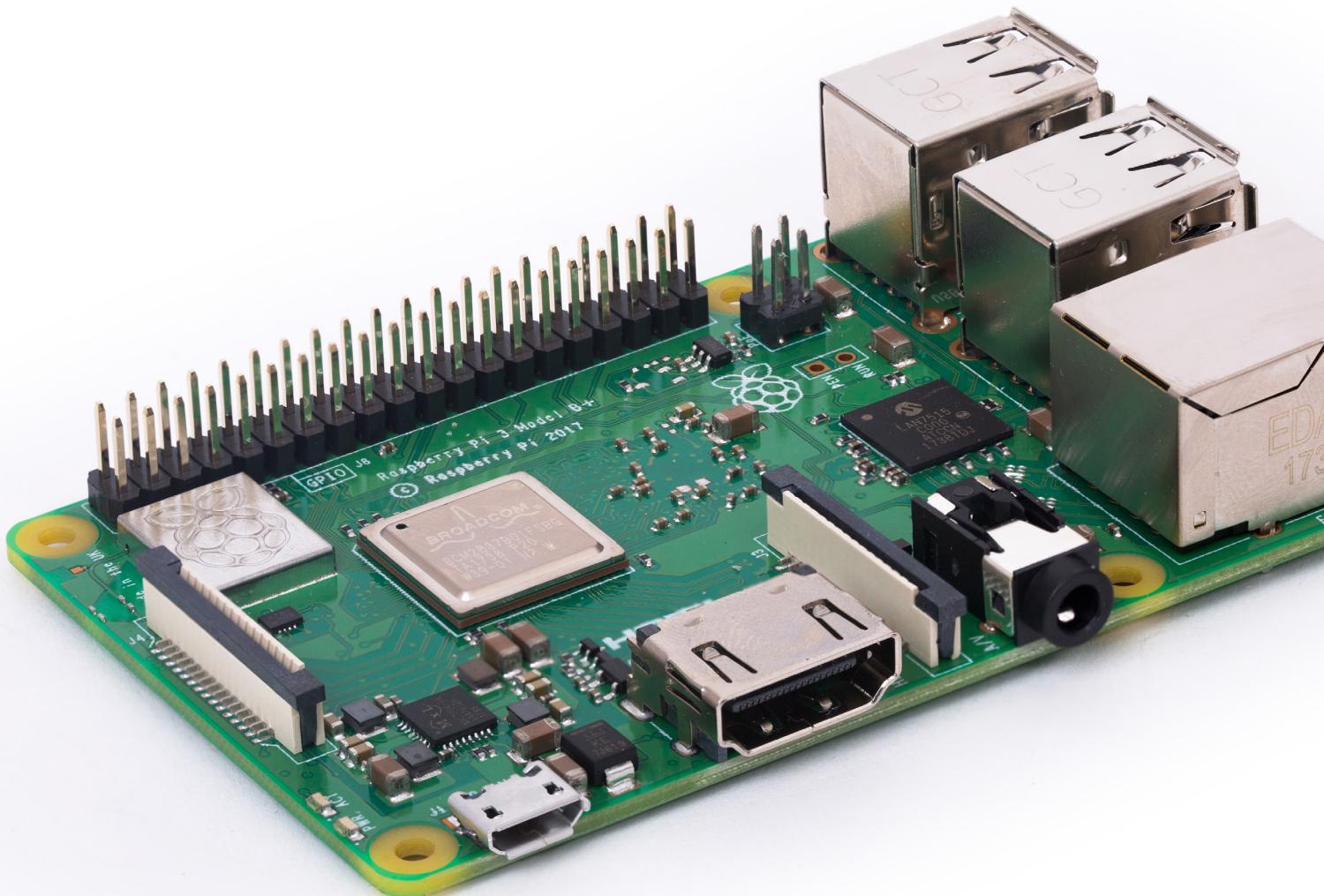
This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products (<http://www.ti.com/sc/docs/stdterms.htm>), **evaluation modules**, and samples (<http://www.ti.com/sc/docs/samptersms.htm>).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2018, Texas Instruments Incorporated

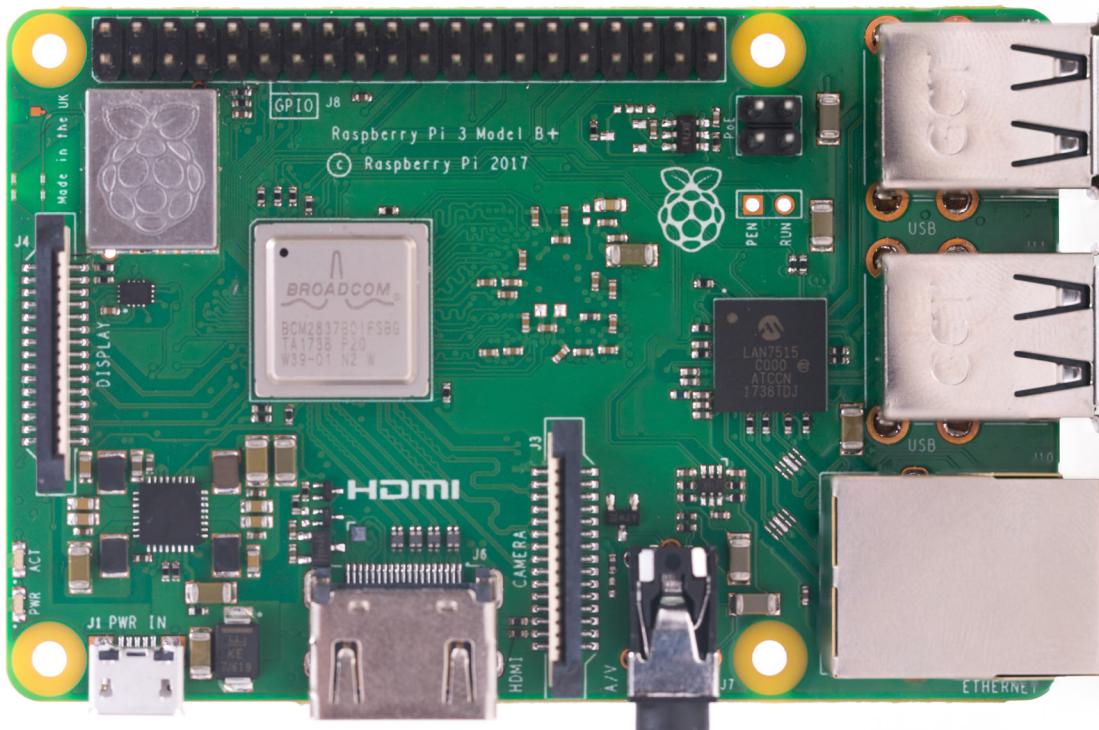
### **.3 Raspbe Modell 3B**



# Raspberry Pi 3 Model B+



# Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4 GHz, dual-band 2.4 GHz and 5 GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT.

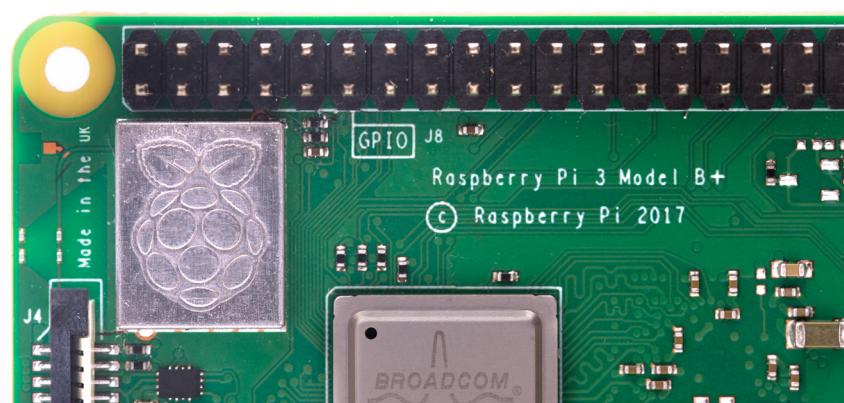
The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

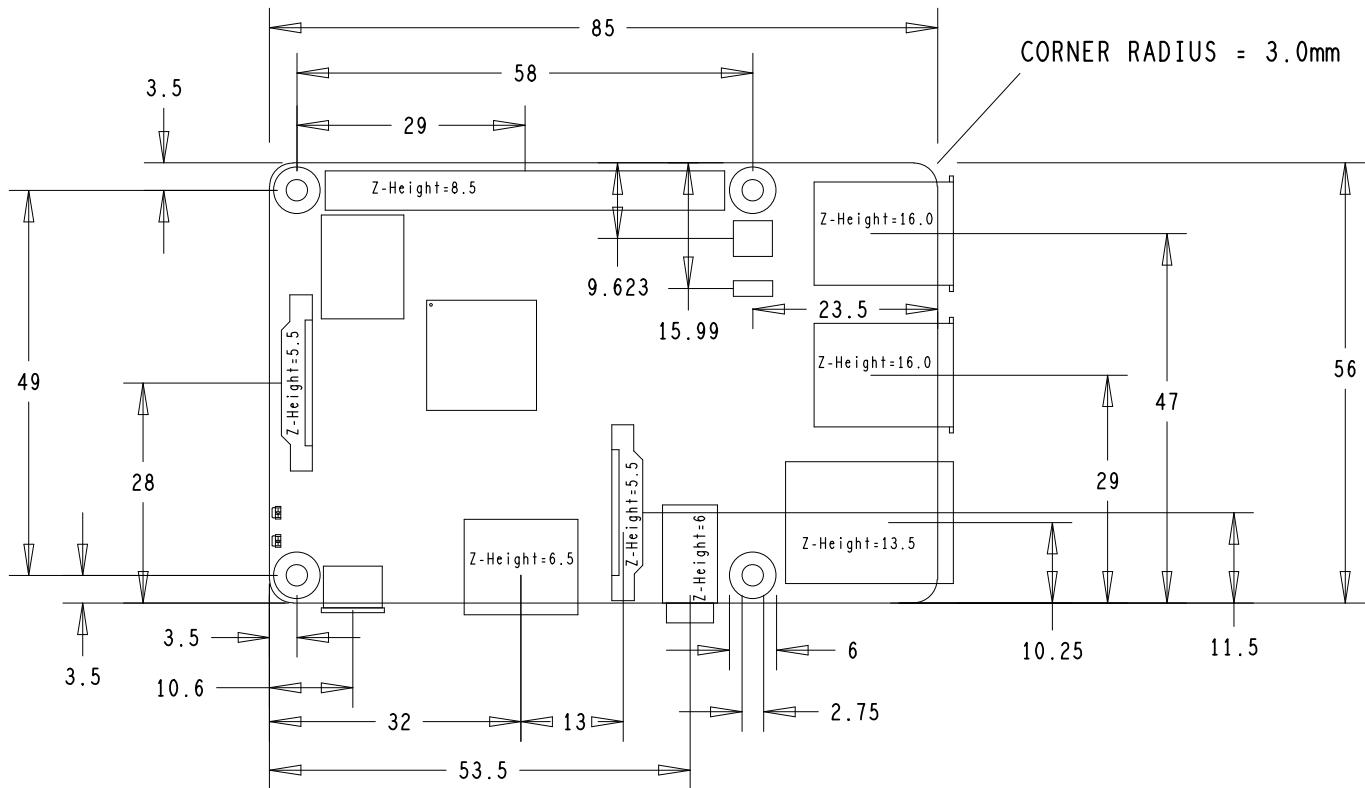


# Specifications

<b>Processor:</b>	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
<b>Memory:</b>	1GB LPDDR2 SDRAM
<b>Connectivity:</b>	<ul style="list-style-type: none"> <li>■ 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE</li> <li>■ Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)</li> <li>■ 4 × USB 2.0 ports</li> </ul>
<b>Access:</b>	Extended 40-pin GPIO header
<b>Video &amp; sound:</b>	<ul style="list-style-type: none"> <li>■ 1 × full size HDMI</li> <li>■ MIPI DSI display port</li> <li>■ MIPI CSI camera port</li> <li>■ 4 pole stereo output and composite video port</li> </ul>
<b>Multimedia:</b>	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
<b>SD card support:</b>	Micro SD format for loading operating system and data storage
<b>Input power:</b>	<ul style="list-style-type: none"> <li>■ 5V/2.5A DC via micro USB connector</li> <li>■ 5V DC via GPIO header</li> <li>■ Power over Ethernet (PoE)-enabled (requires separate PoE HAT)</li> </ul>
<b>Environment:</b>	Operating temperature, 0–50°C
<b>Compliance:</b>	For a full list of local and regional product approvals, please visit <a href="http://www.raspberrypi.org/products/raspberry-pi-3-model-b+">www.raspberrypi.org/products/raspberry-pi-3-model-b+</a>
<b>Production lifetime:</b>	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.



# Physical specifications



## Warnings

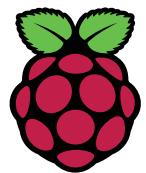
- This product should only be connected to an external power supply rated at 5V/2.5A DC. Any external power supply used with the Raspberry Pi 3 Model B+ shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors, and mice when used in conjunction with the Raspberry Pi.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

## Safety instructions

**To avoid malfunction of or damage to this product, please observe the following:**

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi 3 Model B+ is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the edges to minimise the risk of electrostatic discharge damage.





# Raspberry Pi

HDMI is a trademark of HDMI Licensing, LLC  
Raspberry Pi is a trademark of the Raspberry Pi Foundation

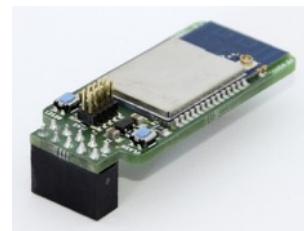
#### **.4 cod.m ZigBee RaspberryPi Modul**

# ZigBee CC2652P2 Raspberry Pi Funkmodul

V0.2

**cod.m**  
digitale Individuelllösungen

- 2,4 GHz CC2652P-ZigBee-Funkmodul mit u.FL-Antennenbuchse zur Verwendung mit einer UART-Schnittstelle
- Ausgelegt für den direkten Anschluss an die GPIO-Leiste aller Raspberry Pi Modelle
- Entworfen für die Verwendung mit Homegear ([www.homegear.eu](http://www.homegear.eu)) oder zigbee2mqtt ([www.zigbee2mqtt.io](http://www.zigbee2mqtt.io))
- Für die ZigBee-Coordinator-Funktionalität kommt Z-Stack-Firmware 3.x (Koenkk) zum Einsatz
- Aktuellste Version des Texas-Instruments ZigBee CC2652P Cortex-M4F Microcontrollers mit integriertem power amplifier (+20dBm)
- Integrierter LDO-Spannungswandler (5V/3.3V)
- Durch Änderung eines Lötjumpers kann die vorhandene PCB-Antenne genutzt werden



## Vorbereitung Raspberry Pi und Serial/UART

1. Den U.FL-Antennenadapter an die Antennenbuchse des Moduls anschließen (orange Markierung).
2. Das Modul auf die GPIO-Leiste des ausgeschalteten Raspberry Pi's aufstecken (blaue Markierung).
3. Raspberry Pi mit Strom versorgen und starten
4. Raspberry Pi UART-Konfiguration: Das Modul ist für die Verwendung mit der UART-Schnittstelle ausgelegt. Um diese nutzen zu können, sind einige Konfiguration notwendig. Es muss die serielle Schnittstelle verfügbar gemacht und der UART aktiviert werden.
  - a. Da `ttyAMA0` oder `serial0` von der seriellen Konsole verwendet werden können, muss man sie zunächst freigeben. Dafür löscht man alle Referenzen zu `ttyAMA0` (Bsp.: `console=ttyAMA0,115200`) und `serial0` (Bsp.: `console=serial0,115200`) in der Datei `/boot/cmdline.txt`, falls dort Einträge vorhanden sind.
  - b. Ab Raspberry Pi 3 wird `/dev/ttyAMA0` für die Wifi und Bluetooth-Schnittstelle verwendet. Da man diese Schnittstelle am besten für das ZigBee-Modul verwenden sollte, muss man Bluetooth deaktivieren. Dafür ergänzt man folgende Zeile in der Datei `/boot/config.txt`:

```
dtoverlay=disable-bt
```

Alternativ kann man auch das Bluetooth-Modul auf `mini UART` umstellen:

```
dtoverlay=miniuart-bt
```

- c. In der Datei `/boot/config.txt` muss der UART aktiviert sein:

```
enable_uart=1
```

- d. Wenn man Bluetooth deaktiviert hat, muss man alle Betriebssystemdienste deaktivieren, die darauf zugreifen wollen. Deshalb muss der `Modem System Service` deaktiviert werden:

```
sudo systemctl disable hcuart
```

5. Raspberry Pi neustarten

Siehe auch: [https://www.zigbee2mqtt.io/information/connecting\\_cc2530.html#to-a-raspberry-pi-zero](https://www.zigbee2mqtt.io/information/connecting_cc2530.html#to-a-raspberry-pi-zero)  
<https://doc.homegear.eu/homegear-zigbee/configuration.html#free-up-serial-line-and-enable-uart>

## Konfiguration Homegear

Nach der Installation von `homegear-zigbee`, muss in der Datei `zigbee.conf` folgende Einstellungen vorgenommen werden. Dadurch wird innerhalb von Homegear die Bezeichnung `CC2652` festgelegt und die seriellen Schnittstelle `ttyAMA0` verwendet.



```
[Serial]  
  
id = CC2652  
deviceType = serial  
  
#use your own 16 bytes hexadecimal key!  
password = AABBCCDDEEFF11223344556677889900  
  
device = /dev/ttyAMA0
```

Siehe auch: <https://doc.homegear.eu/homegear-zigbee/configuration.html#serial>

## Konfiguration zigbee2mqtt

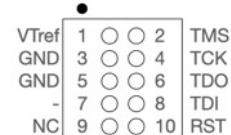
Wenn man `zigbee2mqtt` installiert hat, muss die nachfolgende Konfiguration in der Datei `data/configuration.yaml` eingestellt werden. Es wird `ttyAMA0` mit einer Baudrate von 115200 und deaktiviertem RTS/CTS verwendet.



```
serial:  
  port: /dev/ttyAMA0  
advanced:  
  baudrate: 115200  
  rtscts: false
```

## Firmware Update

An der 2x5-Stifteleiste auf der Oberseite (JTAG) kann zur Firmwareaktualisierung ein JTAG-Debugger angeschlossen werden. Alternativ kann die Firmware direkt vom Raspberry Pi aus mittels Serial-Bootloader und `cc3538-prog` aktualisiert werden.



[https://github.com/Koenkk/Z-Stack-firmware/tree/master/coordinator/Z-Stack\\_3.x.0/bin](https://github.com/Koenkk/Z-Stack-firmware/tree/master/coordinator/Z-Stack_3.x.0/bin) (ZigStar)  
<https://github.com/codm/cc2652-raspberry-pi-module#firmware>

## Bestimmungsgemäße Verwendung

Dieses Modul ist dazu bestimmt, mittels der GPIO-Leiste (UART) am Raspberry Pi angeschlossen zu werden und diesem eine ZigBee-Schnittstelle (Coordinator) zur Verfügung zu stellen.

Ausschließlich die genannte Bestimmungsgemäße Verwendung ist zulässig. Eine andere Verwendung führt zu Gewährleistungs- und Haftungsausschluss.

## Sicherheitshinweise

Halten Sie das Modul von Wärme und Sonnenstrahlung fern. Vermeiden Sie den Kontakt mit Staub und den Einfluss von Flüssigkeiten. Verwenden Sie das Modul nur in Innenräumen. Schützen Sie das Modul vor elektrostatischer Entladung.

## Technische Daten

<b>Kurzbezeichnung:</b>	CC2652 RPi Serial Module V0.2, 90084	<b>Abmessung:</b>	45 x 20mm
<b>Versorgungsspannung:</b>	5V	<b>Gewicht:</b>	4g
<b>Umgebungstemperatur:</b>	+5 bis +45°C	<b>Standard:</b>	IEEE 802.15.4 (ZigBee Coordinator)

Open-Source Projekt: Support im Homegear-Forum oder über [shop@codm.de](mailto:shop@codm.de), kein Telefon support!

cod.m GmbH  
Allendorfer Straße 56  
35708 Haiger

Geschäftsführer: Patrik Mayer  
Amtsgericht Wetzlar, HRB 6686

+49 2773 91878-0  
<https://www.codm.de>  
<https://shop.codm.de>

UST-ID: DE815516311  
WEEE-Reg.-Nr.: DE78677954



RoHS



## .5 Phillips Hue Dimmer Switch

Philips Hue  
Dimmer switch



#### Wireless installation

Battery powered  
Easy access to light recipes  
Use as a remote control



8718696743157



## Control your lights

anywhere in your house

Control and dim your lights with a Hue dimmer switch. Both a wall switch and remote control, this smart switch attaches easily to walls with the included adhesive and can be removed from the magnetic base to use as a remote control.

### Limitless possibilities

- Switch between light scenes
- Brighten and dim your smart lights
- Mount the dimmer switch anywhere
- Use as a remote control dimmer
- Connect up to 10 smart lights

**PHILIPS**

**Dimmer switch**

Wireless installation Battery powered, Easy access to light recipes, Use as a remote control

8718696743157

# Highlights

## Brighten and dim your lights



The Philips Hue Dimmer switch allows you to wirelessly turn your lights up high or down low.

## Connect up to 10 smart lights



With the Philips Hue wireless dimming kit for smart lights, you can connect up to 10 lights to control simultaneously from a single dimmer switch.

## Mount anywhere



The Philips Hue Dimmer switch functions as a normal wall switch and dimmer — but better. In

addition to mounting with screws or strong adhesive tape, you can remove the magnetic control and carry it with you anywhere.

## Switch between light scenes



The smart Dimmer Switch allows you to toggle through four light recipes by simply pressing the "on" switch. Energize, Concentrate, Read, and Relax are programmed by default and you customize your selection in the Hue app.

## Use as a remote control dimmer



The Hue Dimmer switch is magnetic and can be removed from the base plate, allowing you to use the dimmer switch as a handy remote control.

# Specifications

## The switch

- Batteries included: 1 x CR2450
- Frequency band: 2400 – 2483.5 MHz
- Lifetime: 50000 clicks
- Max. lights per switch: 10 if not linked to Hue bridge
- Minimal battery lifetime: 3 year(s)
- Minimal range indoor: 500 inch
- Software upgradeable: when connected to Hue bridge
- Switch depth: 11 mm
- Switch height: 92 mm
- Switch width: 35 mm
- Weight of switch: 37 g
- Wall plate depth: 14 mm
- Wall plate height: 115 mm
- Wall plate width: 70 mm
- Weight including wall plate: 67 g
- Zigbee Light link: protocol IEEE 802.15.4

## What's in the box

- Hue dimmer switch: 1

## Environmental

- Operational humidity: 5% < H < 95% (non condensing)
- Operational temperature: -10°C – 45°C

## Guarantee

- 2 years



Issue date 2018-12-17

Version: 5.0.1

12 NC: 9290 011 73761  
EAN: 87 18696 74315 7

© 2018 Philips Lighting Holding B.V.  
All Rights reserved.

Specifications are subject to change without notice.  
Trademarks are the property of Philips Lighting Holding B.V. or their respective owners.

[www.philips.com](http://www.philips.com)

# Abbildungsverzeichnis

3.1	ZigBee Protocoll Stack Bildquelle: <a href="https://www.researchgate.net/figure/IEEE820154-ZigBee-protocol-stack-architecture-1024x480_265150617">https://www.researchgate.net/figure/IEEE820154-ZigBee-protocol-stack-architecture-1024x480_265150617</a> . . . . .	8
3.2	TI Device Family . . . . .	9
3.3	TI CC 265X Serie . . . . .	9
3.4	ZigBee cod.m Koordinator Bildquelle: cod.m Produktfoto . . . . .	11
3.5	Zigbee2Mqtt Kontext . . . . .	13
3.6	zigbee2mqtt Webfrontend . . . . .	17
3.7	zigbee2mqtt Netzwerkvisualisierung . . . . .	18
3.8	Z-Stack API Auszug . . . . .	19
3.9	TI Code Compose Studio . . . . .	20
3.10	Wireshark . . . . .	21
4.1	Versuchsaufbau . . . . .	23
4.2	Raspberry iptables . . . . .	25
4.3	Raspberry netstat . . . . .	25
5.1	Wireshark Ausschnitt - Joining einer Lampe Teil 1 . . . . .	28
5.2	Wireshark Ausschnitt - Joining einer Lampe Teil 2 . . . . .	29
5.3	Wireshark Ausschnitt - Schalten einer Lampe . . . . .	29
5.4	Wireshark Ausschnitt - Permit Join über Lampe . . . . .	29
5.5	Wireshark Ausschnitt - Beitritt über Lampe . . . . .	30
5.6	Wireshark Ausschnitt - Binden einer Lampe 1 . . . . .	30
5.7	Wireshark Ausschnitt - Binden einer Lampe 2 . . . . .	30
5.8	Wireshark Ausschnitt - Schalten einer Lampe mit Binding . . . . .	31
5.9	Wireshark Ausschnitt - Hinzufügen zu einer Gruppe . . . . .	31
5.10	Wireshark Ausschnitt - Schalten einer Lampe über eine Gruppe . . . . .	31

# Literatur

- [All15] ZigBee Alliance. *Zigbee Standard*. [Online; Stand 10. April 2023]. 2015. URL: <https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf>.
- [BMW23] BMWK. *Smart Meter-Gesetz*. [Online; Stand 30. Juli 2023]. 2023. URL: <https://www.bmwk.de/Redaktion/DE/Pressemitteilungen/2023/05/20230512-smart-meter-gesetz-final-beschlossen.html>.
- [Com19] OASIS MQTT Technical Committee. *Z-Stack Monitor and Test API*. [Online; Stand 11. April 2023]. 2019. URL: <https://github.com/koenkk/zigbee-herdsman/raw/master/docs/Z-Stack%20Monitor%20and%20Test%20API.pdf>.
- [Goo23] Google. *Thread Standard*. [Online; Stand 12. April 2023]. 2023. URL: <https://openthread.io/guides/thread-primer>.
- [IEE23] IEEE. *802.15.4 Standard*. [Online; Stand 12. April 2023]. 2023. URL: <https://ieeexplore.ieee.org/document/9144691>.
- [Kan22] Koen Kelters. *zigbee2mqtt*. [Online; Stand 03. Oktober 2022]. 2022. URL: <https://www.zigbee2mqtt.io/>.
- [Neh23] Nicolas Nehde. *UK Smart Meter Rollout*. [Online; Stand 30. Juli 2023]. 2023. URL: <https://www.smart-energy.com/regional-news/europe-uk/zigbee-alliance-uk-certification/>.
- [Wil22] Jason Wilder. *nginx-proxy*. [Online; Stand 03. Oktober 2022]. 2022. URL: <https://github.com/nginx-proxy/nginx-proxy>.
- [zwa23] zwave. *Z-WAVE Standard*. [Online; Stand 12. April 2023]. 2023. URL: <https://z-wavealliance.org/technology-overview/>.