

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

1	Einführung	2
1.1	Anforderungen an die Praktikumsarbeit	2
2	Marktübersicht Technologien	3
2.1	Funkprotokolle	3
2.2	Zigbee Anwendungen	4
2.2.1	Kommerzielle Anwendungen	4
2.2.2	Nicht kommerzielle Anwendungen	5
3	Grundlagen	7
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	10
3.4.4	Phillips Hue Komponenten	11
3.5	Eingesetzte Software	11
3.5.1	Raspbian OS	11
3.5.2	Docker	11
3.5.3	Docker-Compose	11
3.5.4	zigbee2mqtt	12
3.5.5	MQTT	17
3.5.6	Wireshark	17
3.5.7	Ansible	18
4	Versuchsaufbau	20
4.0.1	Containerverwaltung	21
4.0.2	Namensauflösung	24
4.0.3	Anwendungen	24
5	Versuchsdurchführung	25
5.1	Versuchsaufbau	25
5.2	Aufgabenstellungen	25

5.2.1	Aufgabe 1 - Vorbereitungen	26
5.2.2	Aufgabe 2 - Joining einer Phillips Hue Lampe	27
5.2.3	Aufgabe 3 - Joining eines Gerätes über ein anderes Gerät	28
5.2.4	Aufgabe 4 - Binding der Fernbedienung	29
5.2.5	Aufgabe 5 - Gruppenbildung	30
5.2.6	Fragen	31
6	Musterlösung	35
6.0.1	Aufgabe 2.1 - Joining einer Phillips Hue Lampe	35
6.0.2	Aufgabe 2.2 - Schalten einer Lampe	36
6.0.3	Aufgabe 3 - Joining einer Fernbedienung über die Lampe	36
6.0.4	Aufgabe 4 - Binding der Fernbedienung	37
6.0.5	Aufgabe 5 - Gruppenbildung	37
6.0.6	Fragen	37
7	Life Cycle Management	38
7.0.1	Deployment	38
7.0.2	Zurücksetzen des Versuchs	39
7.0.3	Update der eingesetzten Software	39
8	Anhänge	I
.1	Texas Instruments SimpleLink	I
	Abbildungsverzeichnis	V
	Literatur	VI

Kapitel 1

Einführung

In diesem Projekt soll ein Praktikumsversuch für die Vorlesung Internet of Things für Professor Dr. Jürgen Winter entwickelt werden. In dem Versuch soll die Funktionsweise des Funkprotokolls ZigBee untersucht werden. Es wird ein kleines ZigBee Netz mit mehreren Teilnehmern aufgebaut und die Kommunikation zwischen diesen aufgezeichnet und untersucht. Es wird eine Versuchsanleitung entwickelt, die Schritt für Schritt durch den Versuch führt.

1.1 Anforderungen an die Praktikumsarbeit

Die Anforderungen an der Versuch werden an dieser Stelle definiert, um in dieser Dokumentation darauf Bezug nehmen zu können.

- **A010** - Der Versuch soll an einem Tag durchführbar sein.
- **A020** - Der Versuch soll kein Vorwissen in Linux voraussetzen
- **A030** - Der Versuch setzt Vorwissen in Paketorientierten Datenübetragung voraus.
- **A040** - Der Versuch setzt Vorwissen in der Bedienung von Wireshark voraus.
- **A050** - Der Versuch soll zu Hause und in der Hochschule durchführbar sein.
- **A100** - Der Versuch soll automatisch auf den Raspberry 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 sich die Außentemperatur aus dem Wetterbericht beziehen. Viele Hersteller haben mittlerweile ein breites Portfolio an sogenannten “Smart Devices“, den dazu je nach Übertragungsprotokoll notwendigen “Bridge“ und einer entsprechenden App zur Steuerung. Hier ist der Hersteller Phillips mit seiner Produktmarke “Hue“ als Beispiel zu nennen. Die Produktgruppe umfasst eine Bridge mit zugehöriger 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 einzubinden. Wlan ist allerdings optimiert für hohe Übertragungsraten und nicht für leistungsschwache Endgeräte. Dies ist insbesondere für batteriebetriebene Geräte nachteilig. Bei WLAN ist es problematisch, viele Geräte mit geringer Bandbreite mit einem Access-Point zu verbinden. Die nutzbare Bandbreite für Geräte wie Notebooks sinkt damit ab.

- **Bluetooth**

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.

- **Z-Wave**

Z-Wave 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.

- **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.

- **Thread**

Thread ist ein Funkprotokoll welches ebenfalls auf den 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 adressierbar.

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 dienen 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 Steuerungssoft-

ware “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:

- **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 meist nur die Geräte aus dem eigenem Haus supported 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 “ [**z2m**] 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 die Gerätefunktionen 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 mehrere Hersteller unterstützt werden. ZHA unterstützt neben Texas Instruments auch Hardware von Dresden Elektronik, Silicon Labs, DIGI und ZiGate. ZHA ist für den Anwender extrem vereinfacht, 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 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 ist damit die Grundlage von ZigBee, Thread und 6LoWPAN. Im Standard sind mehrere Modulationsverfahren sowie Frequenzbereiche definiert. Peer-to-Peer ist Teil des Standards. 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 Heimautomatisierung voranzubringen. ZigBee basiert auf dem offenen 802.15.4 Standard, bringt allerdings zusätzliche Komponenten mit, die nicht in einem IEEE Standard definiert sind. ZigBee ist in Form von weiteren Protokollschichten 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.

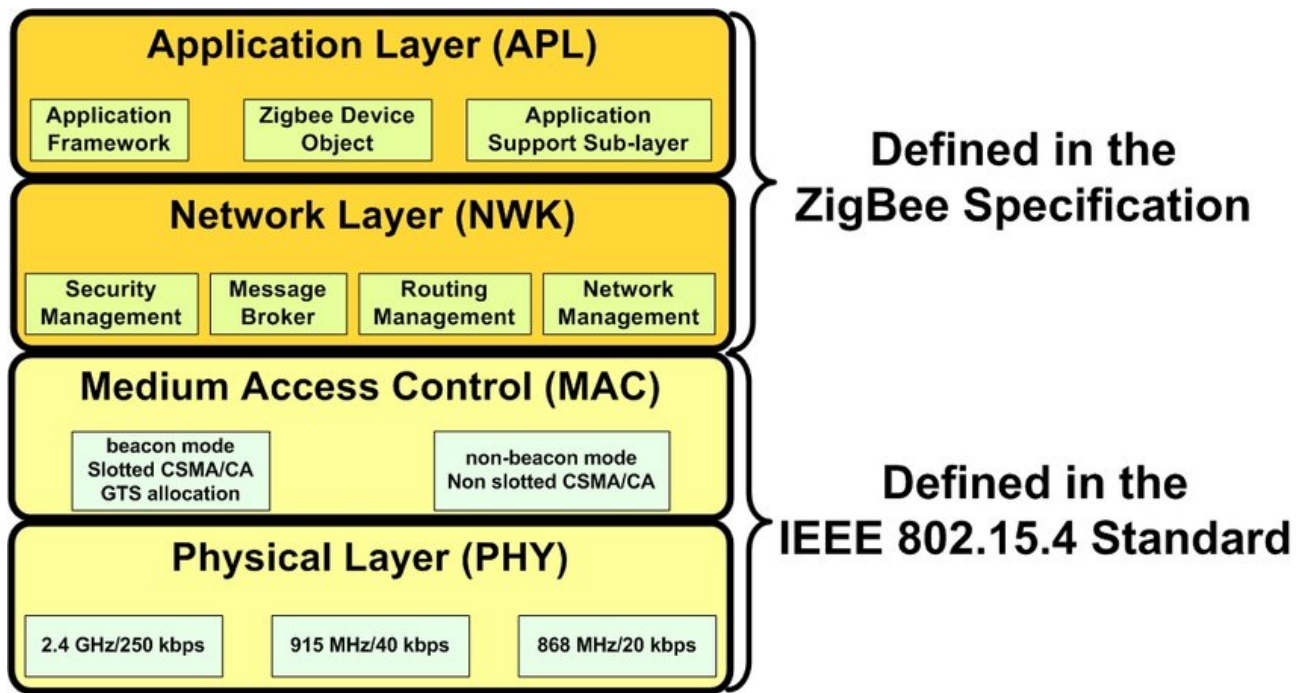


Abbildung 3.1: ZigBee Protocoll Stack

Bildquelle: 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 ZigBee Geräte nutzen lassen. 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 ⁽¹⁾
CC2650F128xxx	Multi-Protocol ⁽²⁾	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. ZigBee Geräte nutzen in einigen Anwendungen Bluetooth LE zur Koppelung, daher ist die Unterstützung diesen Protokolls sinnvoll.

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							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		704	144 + 8	31				X
CC2652P		X				X	X	X	X	X	352	80 + 8	26				X
CC2652P7		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 Z-Stack Bibliothek zur Verfügung. Diese stellt alle grundlegenden Funktionen um das ZigBee Protokoll zu implementieren. Mit Texas In-

struments 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 Herstellers programmiert werden. Alternativ kann man günstig einen USB-Stick mit aufgelöteten CC Chip erwerben, und auch diesem mit entsprechenden Tools programmieren.

Weitere 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 per Docker Containerisierung ausgerollt.

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

Auf dem RaspberryPi wird das Linux-basierte Betriebssystem RaspbianOS. Dies ist eine von den Entwicklern des RaspberryPis eigenentwickelte und für den RaspberryPis angepasste Linux-Distribution. Es baut auf Ubuntu auf.

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.

3.4.3 CC2531 Sniffer Stick

Mit diesem Stick wird die ZigBee Kommunikation in des Versuchsnetzwerkes mitgeschnitten. Der Stick basiert auf einem leistungsschwachen Chip, der mit entsprechender Firmware Pakete 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/andrebd/cc2531>. Hier ist auch eine Anleitung zum kompilieren. Die hieraus entstehende ausführbare Datei muss in entsprechenden Wireshark Ordner kopiert werden, und kann anschließend als Interface ausgewählt werden. Die Funktion nennt sich bei Wireshark “extcap”.

todo: Screenshot wireshark

3.4.4 Phillips Hue Komponenten

Unter dem Namen “Hue “ vertreibt Phillips eine Reihe intelligenten Endgeräten sowie entsprechenden 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 der ZigBee Kommunikation zwischen den Geräten und der IP Kommunikation zu beispielsweise einer Smartphone App. Die Geräte sind kompatibel zu dem Software-Gateway zigbee2mqtt, benutzen also keine speziellen Schlüssel oder ähnliches. 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 RaspberryPi speziell auf die Bedürfnisse des Board angepasst ist. Es enthält eine Desktop Umgebung sowie grundlegende Pakete. Es basiert auf Debian, damit sind entsprechenden reichhaltige 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. Da alle notwendigen Abhängigkeiten mit einem Container mitgeliefert werden, ist eine Installation meist komplikationsfrei. 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 meist ein vollständiges Betriebssystem virtualisiert wird.

3.5.3 Docker-Compose

Docker-Compose ist ein Tool, um große Containerumgebungen im Textformat, hier “YAML “ zu definieren. Ein Container kann entweder 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 ! “ zurückgibt 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.

Alternativ zu der Docker-CLI lässt sich der Zielzustand auch 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

```

errhält man das selbe Ergebniss wie mit dem vorher gezeigtem CLI Befehl.

3.5.4 zigbee2mqtt

zigbee2mqtt ist ein offenes Softwareprojekt und am besten mit “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 ihren Apps zu kommunizieren, macht zigbee2mqtt die Geräte per mqtt nach außen verfügbar. Auf abstrakter Ebene bedeutet dies, das 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 ein 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 gibt nach erfolgreichem Interview alle Cluster an.

todo: Screenshot MQTT

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 gezeigten 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

Wie der Name sagt, sind hier 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     ...

```

Hier liegen die wichtigsten Parameter über das ZigBee Netzwerk sowie Eckdaten über alle Teilnehmer ab.

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      "0xbc33acfffe9587ed": {
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","nwAddr":0,"manufId":
   :0,"epList":[1,2,3,4,5,6,8,10,11,12,13,47,110,242],
2  "endpoints":{"1":{"profId":260,"epId":1,"devId":5,"inClusterList":[],"outClusterList":
   :[],"clusters":{},"binds":[],"configuredReportings":[]},
3  "meta":{}}, {"2":{"profId":257,"epId":2,"devId":5,"inClusterList":[],"outClusterList":
   :[],"clusters":{},"binds":[],"configuredReportings":[]},
4  "meta":{}}, {"3":{"profId":260,"epId":3,"devId":5,"inClusterList":[],"outClusterList":
   :[],"clusters":{},"binds":[],"configuredReportings":[]},
5  "meta":{}}, {"4":{"profId":263,"epId":4,"devId":5,"inClusterList":[],"outClusterList":
   :[],"clusters":{},"binds":[],"configuredReportings":[]},
6  "meta":{}}, {"5":{"profId":264,"epId":5,"devId":5,"inClusterList":[],"outClusterList":
   :[],"clusters":{},"binds":[],"configuredReportings":[]},
7  "meta":{}}, {"6":{"profId":265,"epId":6,"devId":5,"inClusterList":[],"outClusterList":
   :[],"clusters":{},"binds":[],"configuredReportings":[]},
8  "meta":{}}, {"8":{"profId":260,"epId":8,"devId":5,"inClusterList":[],"outClusterList":
   :[],"clusters":{},"binds":[],"configuredReportings":[]},
9  "meta":{}}, {"10":{"profId":260,"epId":10,"devId":5,"inClusterList":[],"outClusterList":
   :[],"clusters":{},"binds":[],"configuredReportings":[]},
10 "meta":{}}, {"11":{"profId":260,"epId":11,"devId":1024,"inClusterList":[1281,10],
   "outClusterList":[1280,1282], "clusters":{},"binds":[],"configuredReportings":[]},
11 "meta":{}}, {"12":{"profId":49246,"epId":12,"devId":5,"inClusterList":[],"
   "outClusterList":[],"clusters":{},"binds":[],"configuredReportings":[]},
12 "meta":{}}, {"13":{"profId":260,"epId":13,"devId":5,"inClusterList":[25],
   "outClusterList":[],"clusters":{},"binds":[],"configuredReportings":[]},
13 "meta":{}}, {"47":{"profId":260,"epId":47,"devId":5,"inClusterList":[],"outClusterList":
   :[],"clusters":{},"binds":[],"configuredReportings":[]},
14 "meta":{}}, {"110":{"profId":260,"epId":110,"devId":5,"inClusterList":[],"
   "outClusterList":[],"clusters":{},"binds":[],"configuredReportings":[]},
15 "meta":{}}, {"242":{"profId":41440,"epId":242,"devId":5,"inClusterList":[],"
   "outClusterList":[],"clusters":{},"binds":[],"configuredReportings":[]},
16 "meta":{}},"interviewCompleted":true,"meta":{"lastSeen":1671278654240,"
   defaultSendRequestWhen":"immediate"}

```

In diesem Datensatz ist jedes direkt verbundene Gerät vermerkt sowie die Beziehung zu diesem.

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 3.4: 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.

zigbee-herdsman

Der Herdsman ist die eigentliche Kernanwendung von zigbee2mqtt. Diese Modul verbindet sich direkt über einen seriellen Socket 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.

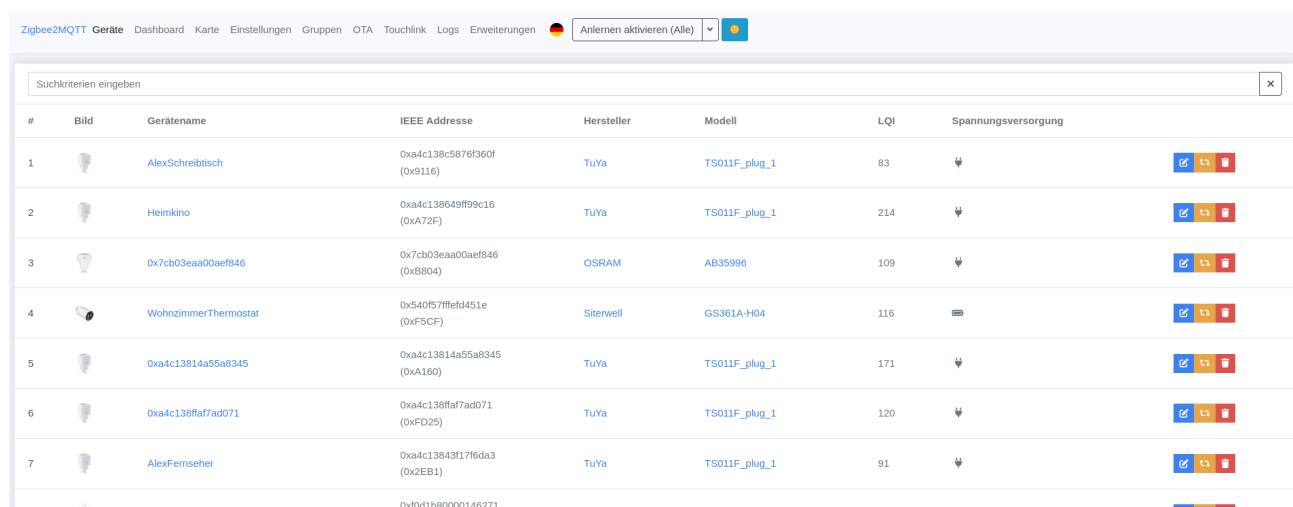
Die API von Herdman wird im entsprechenden GitHub Repository dokumentiert. <https://github.com/Koenkk/zigbee-herdsman>

zigbee-herdman-converters

Dieser Konverter kann proprietäre Cluster die von selbstentwickelten Devices oder manch Devices von Drittherstellern. Mit diesem Converter lassen sich proprietäre Cluster von Geräte so adaptieren, dass sie nach Wunsch gesteuert und ausgelesen werden können.

zigbee2mqtt

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



The screenshot shows the zigbee2mqtt web interface. At the top, there is a navigation bar with links: Zigbee2MQTT, Geräte, Dashboard, Karte, Einstellungen, Gruppen, OTA, Touchlink, Logs, Erweiterungen. There is also a language selector (German flag) and a button 'Anlernen aktivieren (Alle)'. Below the navigation bar is a search bar with the placeholder text 'Suchkriterien eingeben'. The main content is a table listing devices. The table has columns: #, Bild, Gerätename, IEEE Adresse, Hersteller, Modell, LQI, and Spannungsversorgung. There are 7 devices listed, each with a small icon, a name, an IEEE address, a manufacturer, a model, an LQI value, and a power supply status. Each row also has three action buttons: a blue square with a white icon, an orange square with a white icon, and a red square with a white icon.

#	Bild	Gerätename	IEEE Adresse	Hersteller	Modell	LQI	Spannungsversorgung
1		AlexSchreibtisch	0xa4c138c5876f360f (0x9116)	TuYa	TS011F_plug_1	83	
2		Heimkino	0xa4c138649ff99c16 (0xA72F)	TuYa	TS011F_plug_1	214	
3		0x7cb03eaa00aef846	0x7cb03eaa00aef846 (0xB804)	OSRAM	AB35996	109	
4		WohnzimmerThermostat	0x540f57ffef4d451e (0xF5CF)	Siterwell	GS361A-H04	116	
5		0xa4c13814a55a8345	0xa4c13814a55a8345 (0xA100)	TuYa	TS011F_plug_1	171	
6		0xa4c138ffa7ad071	0xa4c138ffa7ad071 (0xFD25)	TuYa	TS011F_plug_1	120	
7		AlexFernseher	0xa4c13843f17f6da3 (0x2EB1)	TuYa	TS011F_plug_1	91	

Abbildung 3.5: zigbee2mqtt Webfrontend

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

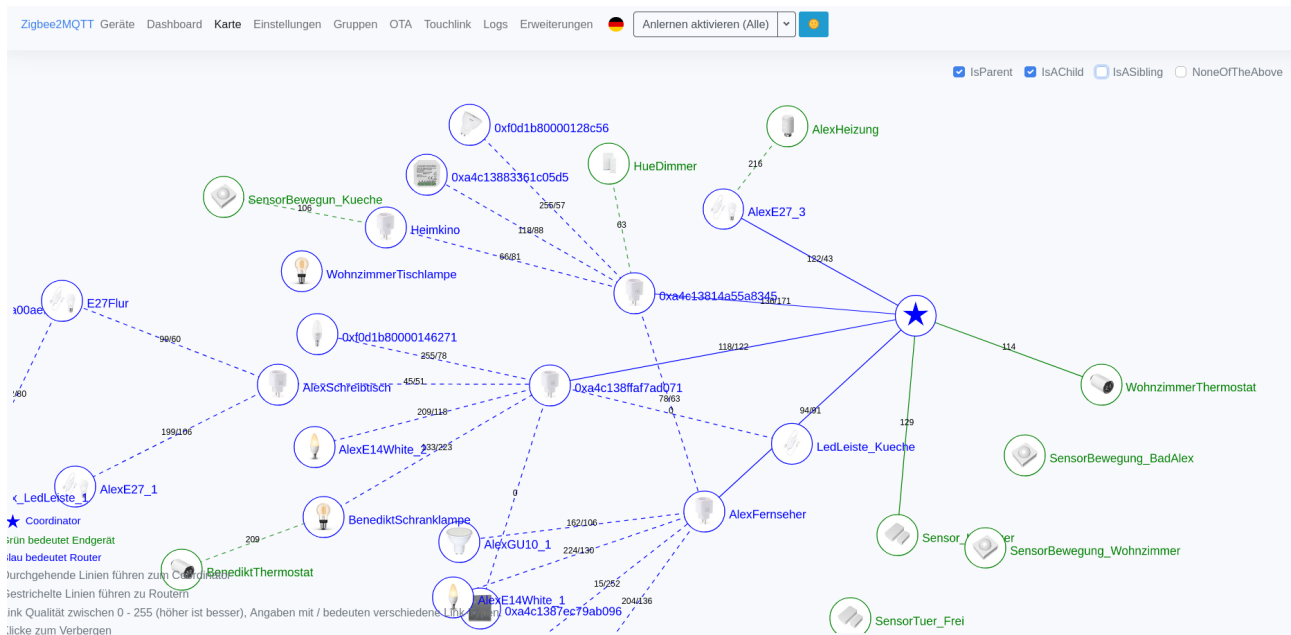


Abbildung 3.6: zigbee2mqtt Netzwerkvisualisierung

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

3.5.5 MQTT

MQTT [**mqtt**] ist ein Protokoll, um Nachrichten zwischen Teilnehmern in einem 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 “gepublizt“, werden alle “subscriber “ entsprechend informiert.

3.5.6 Wireshark

Wireshark ist eine quelloffene Anwendung um Datenströme 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 Scripte in den Ordner “.../extcap “ abgelegt werden, welche Paketströme zurückliefern. Diese Technik wird in diesem Versuch eingesetzt.

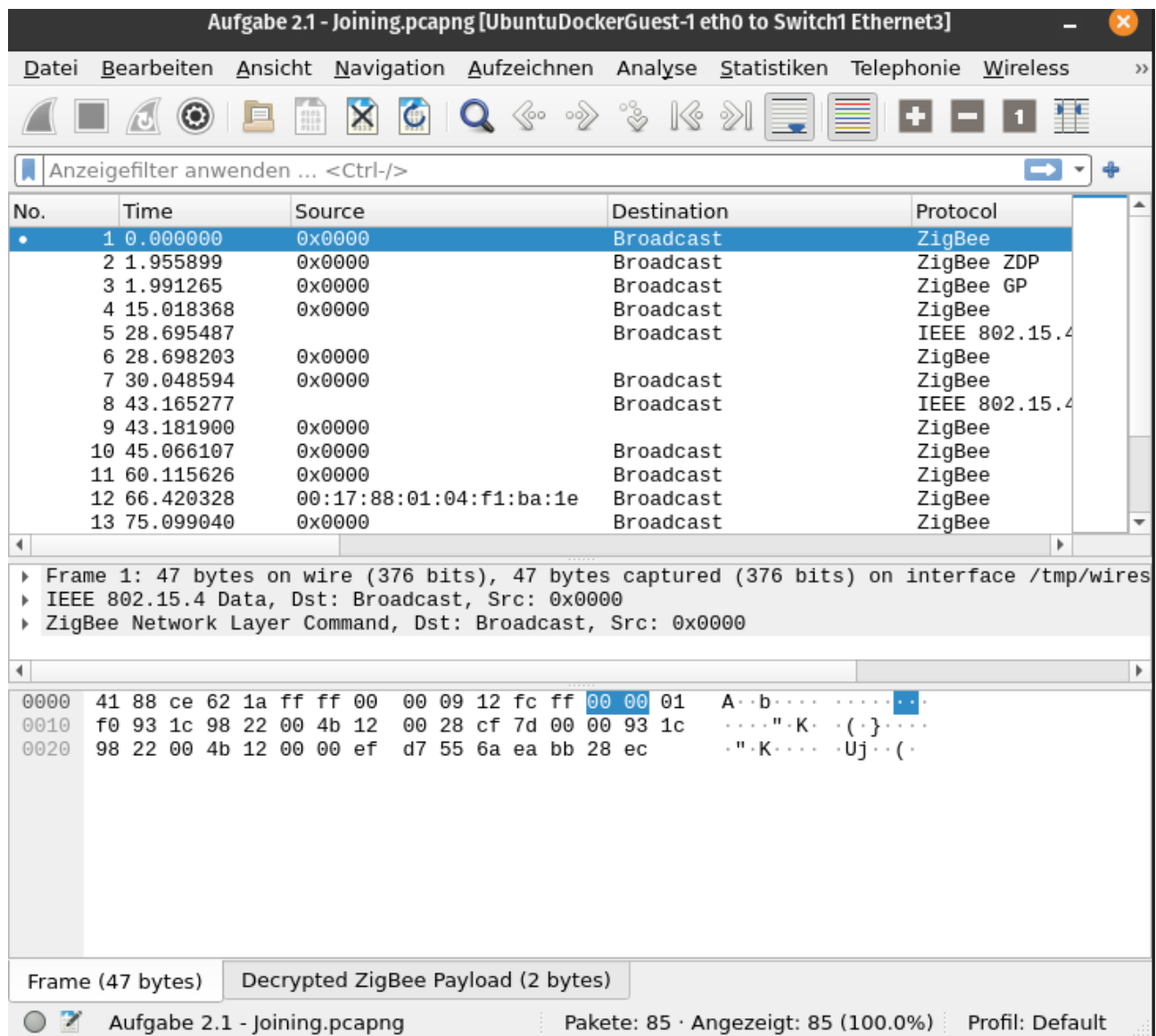


Abbildung 3.7: Wireshark

3.5.7 Ansible

Ansible ist ein Werkzeug zur Automatisierung. Arbeitsabläufe lassen sich strukturiert in YAML (yet another markup language) definieren. Ansible kann Aufgaben auf dem lokalen 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 parametrisierte Pythonmodule auf den jeweiligen Host per SCP und führt diese dort aus.

Kapitel 4

Versuchsaufbau

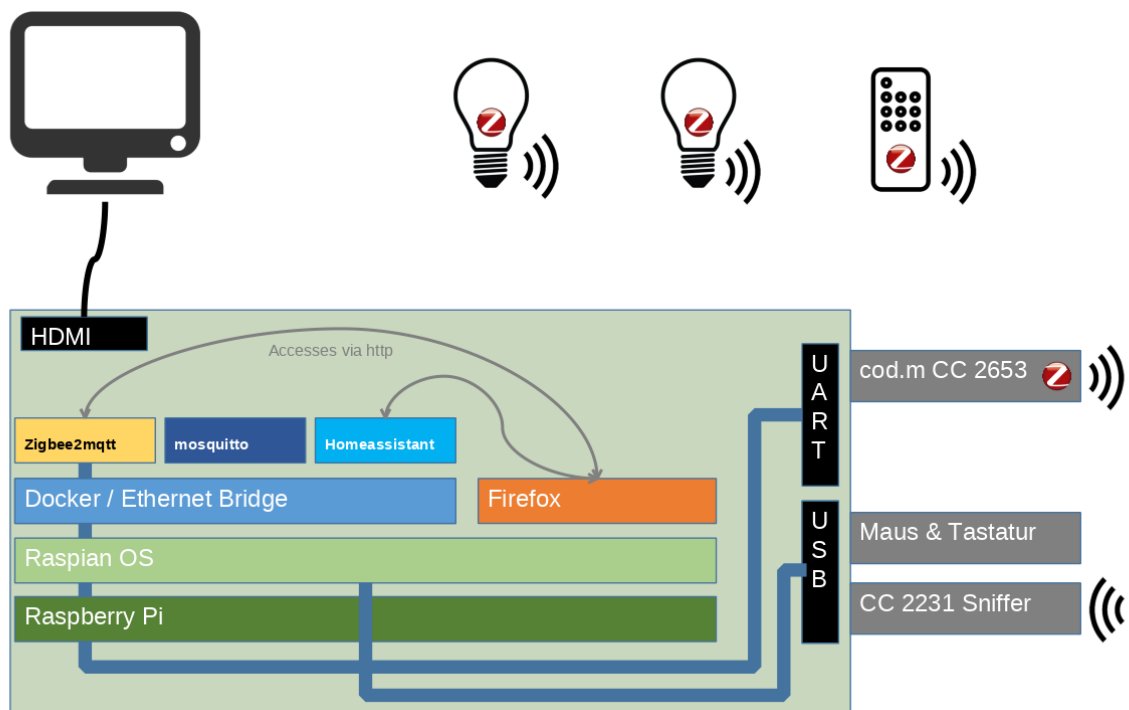


Abbildung 4.1: Versuchsaufbau

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 Domainnamen auflösen zu können.

Der cod.m Zigbeecontroller wird direkt an den Docker Container durchgereicht. Der Sniffer Stick ist

regulär am Host angeschlossen.

4.0.1 Containerverwaltung

Die Container werden mit “Docker-Compose “ verwaltet. Im folgenden werden die Definitionen der einzelnen Services erläutert.

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 anpasst. Dafür wird bei einem Service eine entsprechende Umgebungsvariablen gesetzt. Über den “docker.sock“ erfährt der Proxy ob Container gestartet werden sowie deren Umgebungsvariablen. Wird ein Container mit der Umgebungsvariable “VIRTUAL_HOST=z2m.local “ gestartet, wird automatisch eine Weiterleitung für alle Anfragen mit dem Header “z2m.local“ eingerichtet auf den entsprechenden Container. 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. Dafür werden von Docker Regeln in die “iptables “ Tabellen geschrieben.

```

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.

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/ttyUSB0:/dev/ttyACM0
10 restart: always
11 environment:
12   - VIRTUAL_HOST=z2m.local
13   - VIRTUAL_PORT=8080
14 group_add:
15   - dialout

```

Dem Container wird ebenfalls 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.

Folgend 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 aus und muss explizit erlaubt werden. Desweiteren wird ein MQTT Server angegeben, sowie ein “base-topic“ definiert. Docker löst Containernamen in Dockernetzwerken zu IP-Adressen auf, sodass hier als Server einfach der entsprechende Containernamen angegeben werden kann. Im weiteren wird der Pfad angegeben, auf den der cod.m Adapter gemountet wurde, sowie entsprechende Einstellung für den Webserver gesetzt.

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 “ Netzwerkes erreichbar ist. Zusätzlich wird der Port definiert, auf dem der MQTT Service läuft. 1883 ist der standardmäßige Port für MQTT.

4.0.2 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 Entrys
2 |   become: True
3 |   lineinfile:
4 |     path: /etc/hosts
5 |     line: 127.0.0.1 z2m.local
```

4.0.3 Anwendungen

Für den Versuch wird weiterhin ein Webbrowser 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

Versuchsdurchführung

In diesem Kapitel wird der Versuchsaufbau beschrieben sowie eine Versuchsanleitung gegeben.

5.1 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 Modul sollte bereits auf Ihrem Raspberry montiert sein.

5.2 Aufgabenstellungen

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

5.2.1 Aufgabe 1 - Vorbereitungen

Vorbereitung

a) Schließen sie an den RaspberryPi Monitor, Tastatur, Maus sowie den Sniffer-Stick an. Durch Anschluss der Stromversorgung startet der Raspberry automatisch. Melden sie sich mit folgenden Zugangsdaten an:

- User: student
- Password: zigbeelab

b) Starten sie den Versuch, in dem sie ein Konsolenfenster öffnen und folgenden Befehl absetzen.

```
1 | > docker ps
```

Setzen sie den gewünschten Kanal ein. (Gruppennummer)

c) Starten sie ein Konsolenfenster und überprüfen mit folgendem Befehl, ob die Container ausgeführt werden:

```
1 | > docker ps
```

Es sollten 3 Container im Status "Running" sein.

ZigBee2Mqtt Einrichtung

d) Starten sie den Webbrowser Firefox und Navigieren zu der Seite:

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

e) Starten sie Wireshark über das Symbol auf dem Desktop. Bei den verfügbaren Schnittstellen sollte sich eine "TI CC22531" Schnittstelle finden. Über das vorangestellte Zahnrad-Symbol können sie den abzuhörenden Kanal einstellen. Stellen sie den eben gewählten Zigbee Kanal ein. (Gruppennummer)

Starten sie einen Capture Vorgang und warten einigen Sekunden ab.

Beenden sie den gestarteten Capture Vorgang. Gehen sie in das Menü: Bearbeiten > Einstellungen > Protokolle > ZigBee > Edit (Pre-configured Keys) und tragen hier den "TC-Link Key" und den "Network Key" ein. Als "Network Key" verwenden sie den in Zigbee2Mqtt gesetzten Key. Der "TC-Link Key" ist ein Standard-Key, der verwendet werden muss.

```
1 | 0x 5A 69 67 42 65 65 41 6C 6C 69 61 6E 63 65 30 39 (ZigBeeAlliance09)
```

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 so kurz wie möglich zu halten, und in dieser Zeit nur die in der Aufgabenstellung explizit beschriebenen Aktionen durchzuführen.

5.2.2 Aufgabe 2 - Joining einer Phillips Hue Lampe

a) Schalten sie eine der beiden Lampen ein. Setzen sie die Lampe zurück, indem sie auf der Fernbedienung die beiden äußeren Tasten drücken während sie diese dicht an die Lampe halten. Die Lampe muss mehrmals blinken.

b) Starten sie nun ein Wireshark Mitschnitt und erlauben 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 einen erfolgreichen Interview.

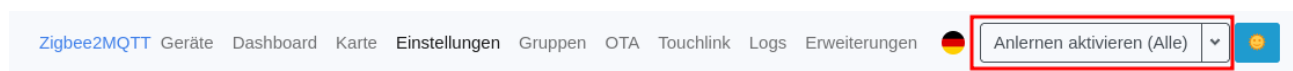


Abbildung 5.1: Zigbee Anlernen aktivieren

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:

Oxf0d1b8000013821d	
Geräte-Name	Oxf0d1b8000013821d
Beschreibung	N/A
Zuletzt gesehen	N/A
Verfügbarkeit	Deaktiviert
Geräte-Typ	Router
Zigbee Modell	A60 TW Z3
Zigbee Hersteller	LEDVANCE
Beschreibung	SMART+ classic E27 TW
Unterstützungsstatus	Unterstützt
IEEE Adresse	Oxf0d1b8000013821d
Netzwerk Adresse	0x30FD
Firmware-Datum	Sep 29 2021
Firmware-Version	01056400
Hersteller	OSRAM
Modell	AC10787
Spannungsversorgung	⚡
Interview erfolgreich	Wahr

Abbildung 5.2: 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 “Details” 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.

5.2.3 Aufgabe 3 - Joining eines Gerätes über ein anderes Gerät

Für diese Aufgabe sollte nur 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.

- Setzen sie die zweite Lampe zurück, indem sie auf der Fernbedienung die beiden äußeren Tasten drücken während sie diese dicht an die Lampe halten. Die Lampe muss mehrmals blinken.
- 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 grünen LED signalisiert.

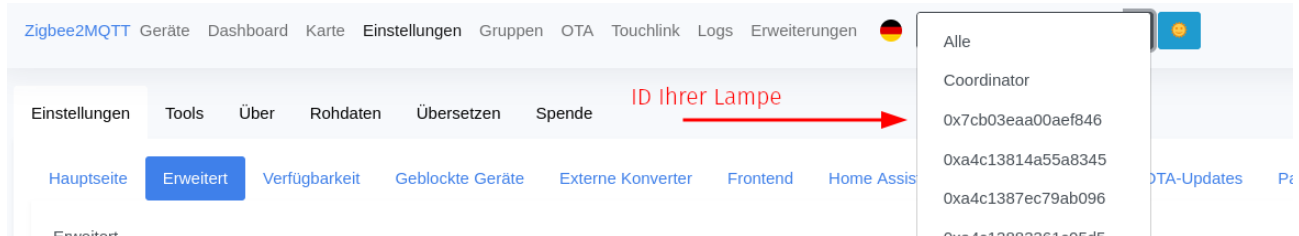


Abbildung 5.3: Zigbee Anlernen aktivieren - nur Lampe

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. Die Fernbedienung lässt sich durch das drücken aller 4 Tasten gleichzeitig zurücksetzen. Drücken sie diese solange bis die LED Grün/Orange blinkt Die Fernbedienung lässt nun zwar ein neues Netzwerk zu, allerdings sucht sie nicht auf neuen Kanälen. Bei einem Kanalwechsel kann es notwendig sein, die Batterie längere Zeit zu entfernen oder den “Setup “ Knopf auf der Rückseite zu betätigen.

Ihre Übersichtsseite sollte wie folgt aussehen.

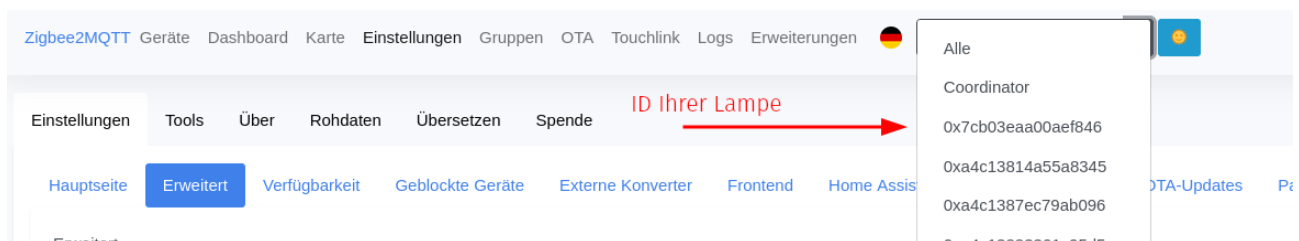


Abbildung 5.4: Zigbee Anlernen aktivieren - nur Lampe

5.2.4 Aufgabe 4 - Binding der Fernbedienung

- 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 11 Ihrer Lampe mit dem Endpunkt Ihrer Fernbedienung. 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. Die Hersteller gehen davon aus, dass die Fernbedienung gewöhnlich per Touchlink gebunden wird. Warten sie bis zigbee2mqtt ein erfolgreiches binding meldet.

Abbildung 5.5: Zigbee Group Binding

Achten sie darauf alle hier 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.

5.2.5 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) 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.

5.2.6 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.

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?

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?

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

Erläutern Sie die Funktion dieser Nachricht.

Mit welchem Kommandoframe antwortet der Koordinator auf den Data-Request?

Welche Werte besitzen die Felder „Short Address“ und „Association Status“?

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 und 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!

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

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!

2. Welchem Zweck dient die **Update Device** Nachricht? Wer ist Absender und wer ist Empfänger? Welche Adresse steht im Feld „Device Address“?

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?

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?

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?

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

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?

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?

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.

Aufgabe 5

Fragen

1. Verdeutlichen Sie sich den Vorgang in dem Sie die vom Koordinator versendeten Nachrichten **Get Group Membership** bzw. **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 Zielpunkt 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 Zielpunkt ist vorhanden?
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?

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.

Kapitel 6

Musterlösung

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

6.0.1 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 15.018368	0x0000	Broadcast	ZigBee	47 Link Status
5 28.695487	0x0000	Broadcast	IEEE 802.15.4	10 Beacon Request
6 28.698203	0x0000	Broadcast	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	10 Beacon Request
9 43.181900	0x0000	Broadcast	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 66.428328	00:17:88:01:04:f1:b...	Broadcast	ZigBee	35 ZCL Touchlink: Scan Request, Seq: 0
13 75.099040	0x0000	Broadcast	ZigBee	47 Link Status
14 98.832344	00:17:88:01:04:f1:b...	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	Broadcast	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	00:17:88:01:04:b9:3...	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	00:12:4b:00:22:98:1...	00:17:88:01:04:b9:3...	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	0x1ed6	IEEE 802.15.4	5 Ack
25 118.851927	0x0000	0x1ed6	ZigBee	73 Transport Key
26 118.852013	0x0000	0x1ed6	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
28 119.181401	0x0000	0x1ed6	ZigBee ZDP	48 Node Descriptor Request, Nwk Addr: 0x1ed6

Abbildung 6.1: Wireshark Ausschnitt - Joining einer Lampe Teil 1

Im ersten Teil ist der Physikalische Verbindungsaufbau 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			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	119.181401	0x0000	0x1ed6	ZigBee ZDP	48 Node Descriptor Request, Nwk Addr: 0x1ed6
29	119.181754			IEEE 802.15.4	5 Ack
30	119.194677	0x1ed6	0x0000	ZigBee ZDP	62 Node Descriptor Response, Nwk Addr: 0x1ed6, Status: Success
31	119.194780			IEEE 802.15.4	5 Ack
32	119.217496	0x0000	0x1ed6	ZigBee	45 APS: Ack, Dst Endpt: 0, Src Endpt: 0
33	119.217934			IEEE 802.15.4	5 Ack
34	119.234410	0x0000	0x1ed6	ZigBee ZDP	48 Active Endpoint Request, Nwk Addr: 0x1ed6
35	119.234767			IEEE 802.15.4	5 Ack
36	119.245166	0x1ed6	0x0000	ZigBee ZDP	52 Active Endpoint Response, Nwk Addr: 0x1ed6, Status: Success
37	119.246083			IEEE 802.15.4	5 Ack
38	119.249120	0x0000	0x1ed6	ZigBee	45 APS: Ack, Dst Endpt: 0, Src Endpt: 0
39	119.249546			IEEE 802.15.4	5 Ack
40	119.264849	0x0000	0x1ed6	ZigBee ZDP	49 Simple Descriptor Request, Nwk Addr: 0x1ed6, Endpoint: 11
41	119.266001			IEEE 802.15.4	5 Ack
42	119.276379	0x1ed6	0x0000	ZigBee ZDP	74 Simple Descriptor Response, Nwk Addr: 0x1ed6, Status: Success
43	119.276723			IEEE 802.15.4	5 Ack
44	119.315762	0x0000	0x1ed6	ZigBee	45 APS: Ack, Dst Endpt: 0, Src Endpt: 0
45	119.316512			IEEE 802.15.4	5 Ack

Abbildung 6.2: 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“ interviewed und per “Simple Descriptor“ Nachrichten beschrieben.

6.0.2 Aufgabe 2.2 - Schalten einer Lampe

1	0.000000	0x1ed6	Broadcast	ZigBee	50 Link Status
2	15.089857	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: 0xffff, Src: 0x0000
7	41.373276	0x0000	Broadcast	ZigBee	51 Many-to-One Route Request, Dst: 0xffff, 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	50 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 6.3: Wireshark Ausschnitt - Schalten einer Lampe

Hier ein ein- und Auschaltvorgang einer Lampe zu sehen. Zum Schalten wird jeweils ein “ZigBee Kommando Frame“ versendet, welches erst Physikalisch und anschließend auf ZigBee Ebene bestätigt wird.

6.0.3 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 6.4: Wireshark Ausschnitt - Schalten einer Lampe

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

22	64.684235		Broadcast	IEEE 802.15.4	10 Beacon Request
23	64.687071	0x1ed6		ZigBee	28 Beacon, Src: 0x1ed6, EPID: TexasIns_00:22:98:1c:93
24	65.823693	00:17:88:01:04:a5:32:74	0x1ed6	IEEE 802.15.4	21 Association Request, FFD
25	65.823921			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			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			IEEE 802.15.4	5 Ack
30	66.338862	0x1ed6	0x0000	ZigBee	55 Route Record, Dst: 0x0000
31	66.339195			IEEE 802.15.4	5 Ack
32	66.428475	0x1ed6	0x0000	ZigBee	60 Update Device
33	66.428570			IEEE 802.15.4	5 Ack
34	66.460232	0x0000	0x1ed6	ZigBee	102 Transport Key
35	66.460313			IEEE 802.15.4	5 Ack
36	66.471942	0x1ed6	0xabf3	ZigBee	73 Transport Key
37	66.483953			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	79.773596	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	80.061515	0x0000	Broadcast	ZigBee	59 Route Request, Dst: 0xabf3, Src: 0x0000

Abbildung 6.5: Wireshark Ausschnitt - Schalten einer Lampe 1

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.

6.0.4 Aufgabe 4 - Binding der Fernbedienung

67	3.100659	0x0000	0xb9f7	ZigBee ZDP	71 Bind Request, On/Off (Cluster ID: 0x0000) 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: 0x0000) Src: PhilipsL_01:04:f1:ba:1e, Dst: PhilipsL_01:04:b9:35:9d

Abbildung 6.6: Wireshark Ausschnitt - Binden einer Lampe 1

Die Bindung wird durch ein Bind-Request von dem Koordinator an die Fernbedienung initiiert. Hier 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 6.7: 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.

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 6.8: Wireshark Ausschnitt - Schalten einer Lampe mit Binding

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

6.0.5 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 6.9: Wireshark Ausschnitt - Schalten einer Lampe mit Binding

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

```

Frame 18: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface /tmp/wireshark_extcap_cc25315U7E41, 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 6.10: Wireshark Ausschnitt - Schalten einer Lampe mit Binding

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 um 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.

6.0.6 Fragen

Aufgabe 2

Kapitel 7

Life Cycle Management

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

Um ein Raspberry automatisch für einen Versuch vorzubereiten, muss dessen Ip-Adresse in der “hosts“ Datei eingetragen werden. Anschließend muss der SSH Schlüssel des Ansible Servers auf dem Raspberry hinterlegt werden. Auf dem Raspberry muss ein User “ansible“ angelegt.

Folgende Schritte müssen ausgeführt werden, um eine Raspberry vorzubereiten.

- Installation von Raspbian OS
- Anlegen User ansible / <secret>
- Hinterlegen SSH-Key von Ansible Server für ansibe user
- Eintragen Raspberry IP in hosts in Ansible Root Verzeichnis
- Konnektivität Ansibe Server und Raspberry herstellen (ping)

7.0.1 Deployment

Zum ausrollen folgenden Befehl auf dem Ansible Host absetzen.

Anschließend wird

- Docker Installiert
- Der User Student angelegt
- Die config Files ausgerollt
- Die /etc/hosts Einträge gesetzt
- zbwireshark Installiert
- Die Docker Services gestartet (zigbee2mqtt)

```
1  ansible-playbook DeployLab.yaml -K
2
3  #BECOME Password:
4  <secret>
```

7.0.2 Zurücksetzen des Versuchs

7.0.3 Update der eingesetzten Software

Kapitel 8

Anhänge

.1 Texas Instruments SimpleLink

SimpleLink™ Microcontroller Platform



Key benefits

- **Broadest 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](#).

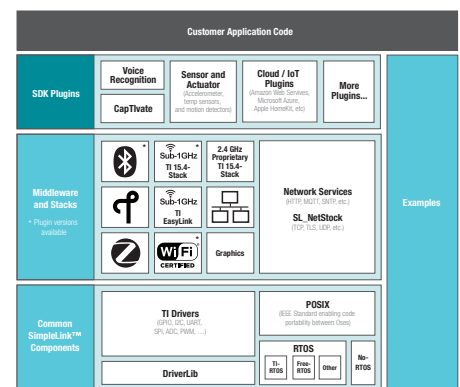
Software Tools

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, or Thread	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	CC3120B00ST	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

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/sampters.htm>).

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

Abbildungsverzeichnis

3.1	ZigBee Protocoll Stack Bildquelle: https://www.researchgate.net/figure/IEEE820154-ZigBee-protocoll-stack-fig2_265150617	8
3.2	TI Device Family	9
3.3	TI CC 265X Serie	9
3.4	Z-Stack API Auszug	15
3.5	zigbee2mqtt Webfrontend	16
3.6	zigbee2mqtt Netzwerkvisualisierung	17
3.7	Wireshark	18
4.1	Versuchsaufbau	20
4.2	Raspberry iptables	22
4.3	Raspberry netstat	22
5.1	Zigbee Anlernen aktivieren	27
5.2	Zigbee Device Übersicht	28
5.3	Zigbee Anlernen aktivieren - nur Lampe	29
5.4	Zigbee Anlernen aktivieren - nur Lampe	29
5.5	Zigbee Group Binding	30
6.1	Wireshark Ausschnitt - Joining einer Lampe Teil 1	35
6.2	Wireshark Ausschnitt - Joining einer Lampe Teil 2	36
6.3	Wireshark Ausschnitt - Schalten einer Lampe	36
6.4	Wireshark Ausschnitt - Schalten einer Lampe	36
6.5	Wireshark Ausschnitt - Schalten einer Lampe	37

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>.
- [Wil22] Jason Wilder. *nginx-proxy*. [Online; Stand 03. Oktober 2022]. 2022. URL: <https://github.com/nginx-proxy/nginx-proxy>.