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

Kompiliert am 11.04.2023 um 12:41:13 - Erstellt mit LaTeX

Inhaltsverzeichnis

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 Teilnehmen 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 vorraussetzen
- **A030** - Der Versuch setzt Vorwissen in Paketorientierten Datenübetragung 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 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 Übertragunsprotokoll 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 batteriebetriene Geräte nachteilig. Bei WLAN ist es problematisch, viele Geräte mit geringer Bandbreite mit einen Access-Point zu verbinden. Die nutzbare Bandbreite für Geräte wie Notebooks sinkt damit ab.

- **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.

- **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 entscheidentes 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.

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“, hautpsächlich 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 Kompatiblität 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 Kompatiblität anderer Geräte sicherzustellen, da offiziell meißt 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 Kompatiblität von Geräten verschiedener Hersteller möglich.

zigbee2mqtt

“zigbee2mqtt “ [**z2m**] ist ein quelloffenes Projekt auf GitHub. Die Anwendung kann anstelle von propietären “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 Kompatiblität 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 vorallem 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 Übertragungstandard im Bereich Heimautomatisierung voranzubringen. ZigBee basiert auf dem offenem 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.

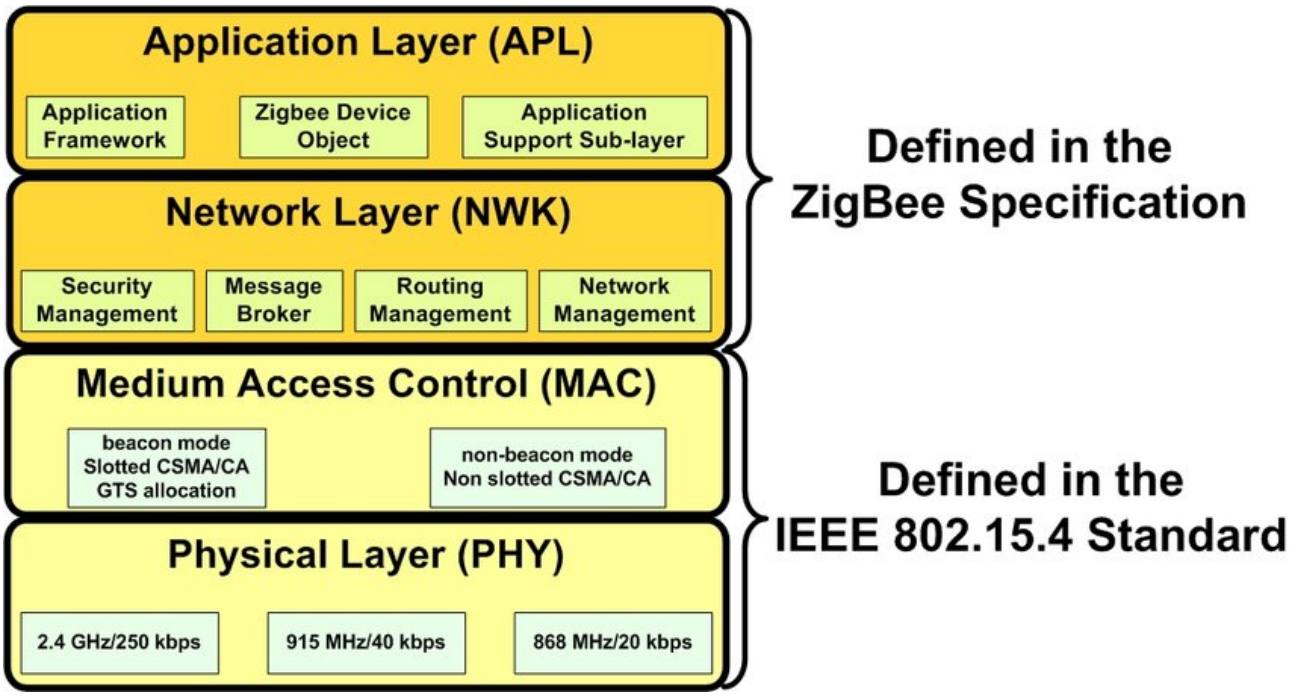


Abbildung 3.1: ZigBee Protocoll Stack
Bildquelle: `https://www.researchgate.net/figure/IEEE820154-ZigBee-protocol-stack-architecture_fig2_265150617`

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 vorallem 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: Test des Messagebrokers Mosquitto

Als Koordinator werden die Leistungsfähigeren Chips aus der 265X Reihe eingesetzt. ZigBee Geräte nutzen in einigen Anwendungen Bluetooth LE zur Koppelung, daher ist die Unterstüzung 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 Herstellern programmiert werden. Alternativ kann man günstig einen USB-Stick mit aufgelöteten 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 meißten 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 eingestzt.

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/andrebdo/w 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 beispielßweiser 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 RaspberryPis speziell auf die Bedürfnisse des Board angepasst ist. Es enthällt 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 meißt 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 taditioneller Virtualisierung, bei der meißt ein vollständiges Betriebssystem virtualisiert wird.

3.5.3 Docker-Compose

Docker-Compose ist ein Tool, um große Containerumgebungen im Textformart, 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 beispielweiße 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 Funktionsweiße wird später erläutert.

Alternativ zu der Docker-CLI lässt sich der Zielzustand auch beschreiben:

```
version: '3'
services:
  helloworld:
    container_name: helloworld
    image: hello-world
    ports:
      - 80:80
    volumes:
      - ./home:/home
    restart: unless-stopped
```

Mit einem

```
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 Funktionsweiße 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

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 Verfahrem zurückgesetzt werden, um wieder einem neuen Netzwerk beitreten zu können.

```
version: '3'
services:
  helloworld:
    container_name: helloworld
    image: hello-world
    ports:
      - 80:80
    volumes:
      - ./home:/home
    restart: unless-stopped
```

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.

state.json

Dies ist die zentrale Datenbank von zigbee2mqtt. Da SQLite eingesetzt wird, lässt sich auch hier der Inhalt wie bei einer Textdatei einfach auslesen. Hier sind alle Konfigurationen der Anwendung, aber auch für jeden Netzwerkteilnehmer Informationen hinterlegt.

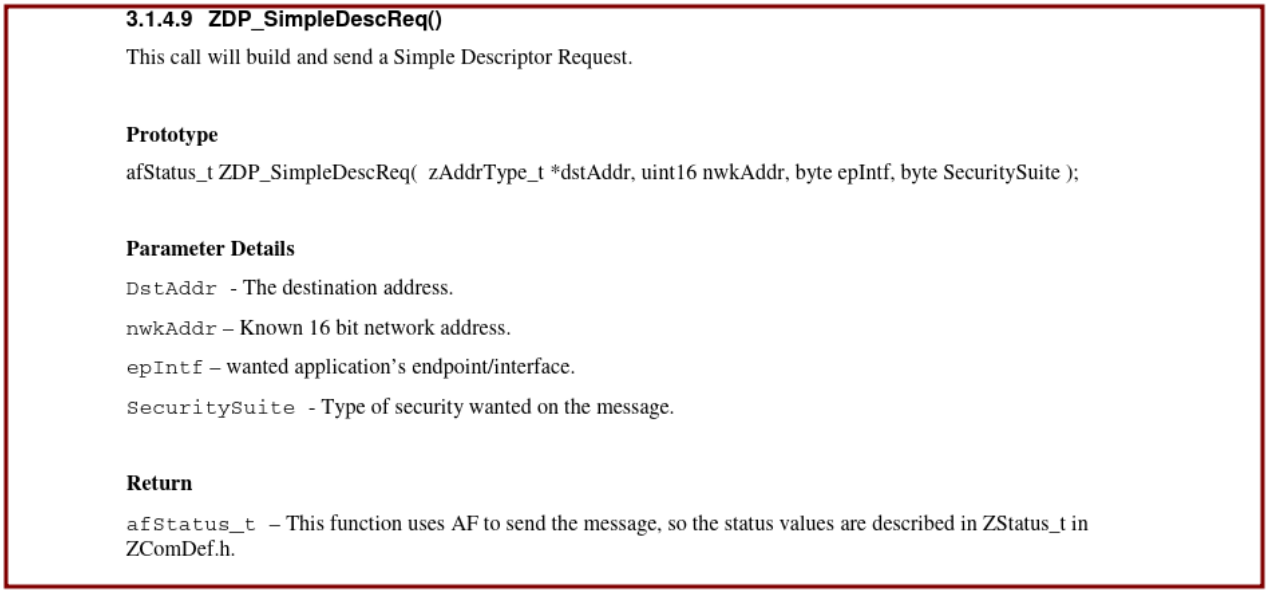
Auszug aus dem Konfigurationsteil:

Ein Datensatz eines Devices

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 Funktionsweiße, 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 verbindert sich direkt über einen seriellen Socket mit dem Koordinator. Über diese Schnittstelle spricht Herdsmann 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.

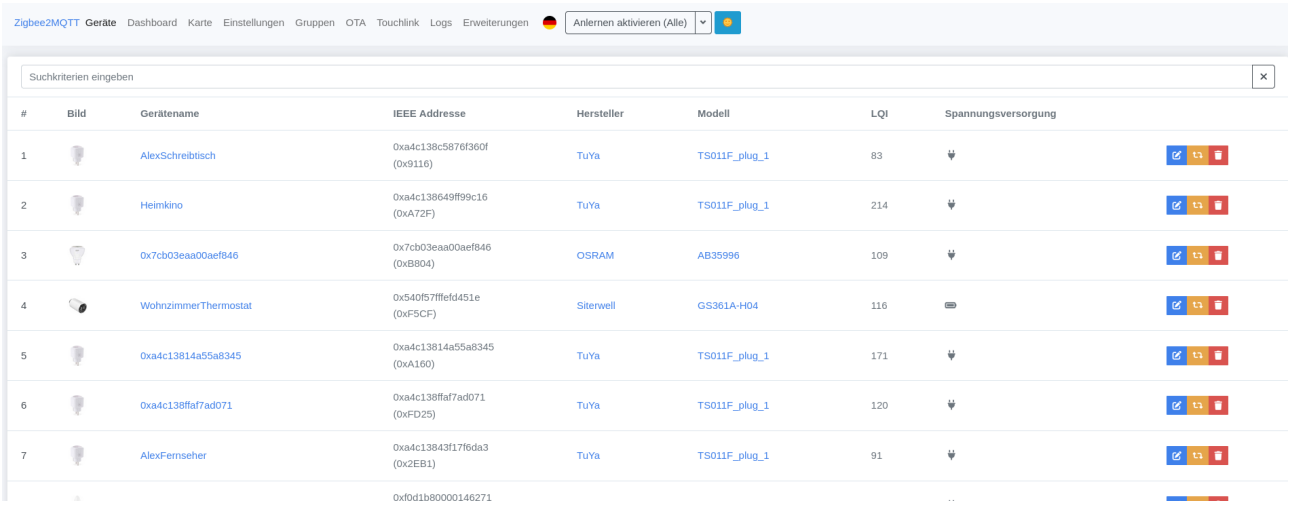


Abbildung 3.5: zigbee2mqtt Webfrontend

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

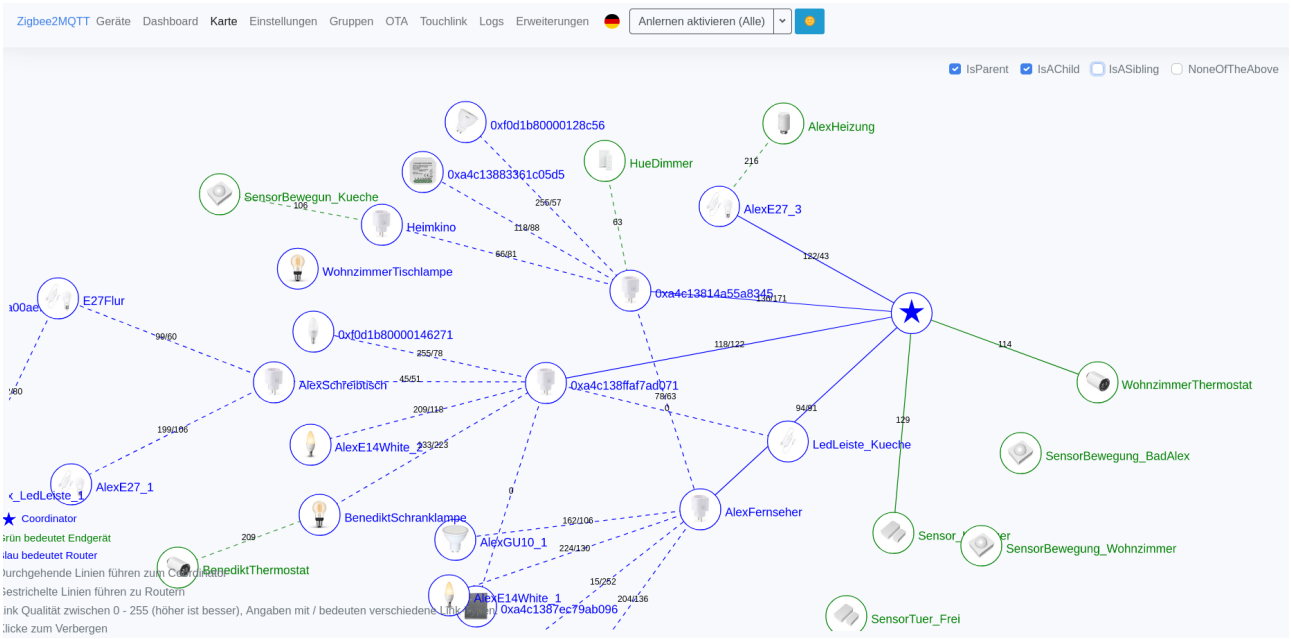


Abbildung 3.6: zigbee2mqtt Netzwerkvisualisierung

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

3.5.5 MQTT

MQTT [**mqtt**] ist ein Protokoll, um Nachrichten zwischen Teilnehmen 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 Teilneh-

mern sowie deren abonnierten “Topics “. Wird eine entsprechende Nachricht an den Broker “gepublisht “, werden alle “subscriber “ entsprechend informiert.

3.5.6 Wireshark

Wireshark ist eine quelloffene Anwendung um Datenstöme 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.

1.Screenshot Wireshark mit Zigbee Paketen

3.5.7 Ansible

Ansible ist ein Werkzeug zur Automatisierung. Arbeitsabläufe lassen sich strukturiert in YAML definieren. 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.

3.5.8

Kapitel 4

Versuchsaufbau

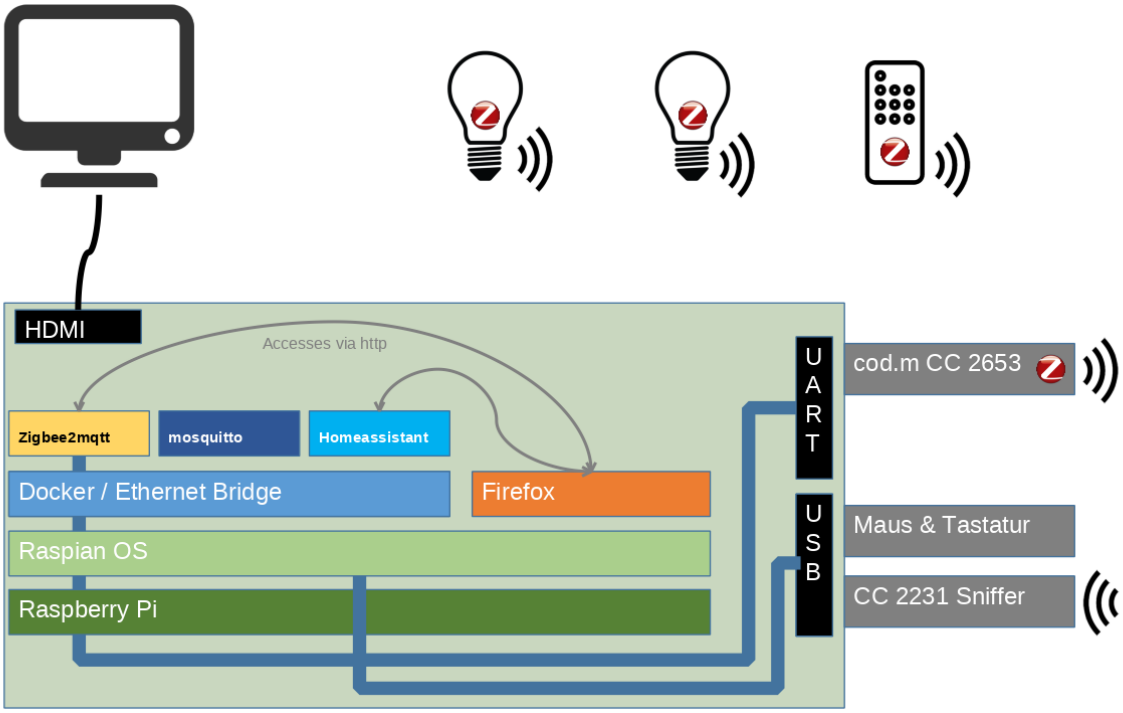


Abbildung 4.1: Versuchsaufbau

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 das Host Interface 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 Images 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, und mit welchen 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 gemountetn. Wenn der gemountete Pfad außerhalb

des Containers nicht existiert, wird der bestehende Ordner aus dem Container kopiert. Exisitert 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 erhällt.

Folgend die zentrale Konfigurationsdatei von “zigbee2mqtt “.

```
  homeassistant: true
permit_join: false
mqtt:
  base_topic: zigbee2mqtt
  server: mqtt://mosquitto:1883
serial:
  port: /dev/ttyACM0
frontend:
  port: 8080
  host: 0.0.0.0
  url: https://z2m.local
advanced:
  homeassistant_legacy_entity_attributes: false
  legacy_api: false
  legacy_availability_payload: false
device_options:
  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

```
  mosquitto:
  container_name: mosquitto
  image: eclipse-mosquitto:latest
  networks:
    - backbone
  restart: always
  deploy:
    resources:
      limits:
        memory: 125M
  volumes:
    - ./mosquitto/config:/mosquitto/config
    - ./mosquitto/data:/mosquitto/data
    - ./mosquitto/log:/mosquitto/log
```

Als MQTT Broker wird “mosquitto “ eingesetzt. Für diesen Container wurde der maximal erlaubte Ar-

beitsspeicher begrenzt. Die Konfigurationen wurden auch hier entsprechend auf den Host gemountet. Als Konfigurationsdatei wird das Standardtemplate verwendet, welches nur an zwei entsprechenden Stellen modifiziert ist.

```
...
allow_anonymous true
...

...
listener 1883
...
```

Es wird der Zugriff von nicht authentifizierten Geräten erlaubt. Dies stellt kein Risiko dar, da der Container nur innerhalb des “backbone “ Netwerkes 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.

```
- name: Add Hosts Entrys
  become: True
  lineinfile:
    path: /etc/hosts
    line: 127.0.0.1 z2m.local
```

4.0.3 Anwendungen

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

```
  - name: Install required Packages
  become: true
  apt:
    pkg:
      - wireshark
      - firefox
    state: latest
    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.

- RespberryPi 3 mit eingesetzter SD-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

Das 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.

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

```
1     > docker ps
```

Es sollten 3 Container im Status “Running“ sein.

ZigBee2Mqtt Einrichtung

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

```
1     https://zigbee2mqtt.local
```

d) Überprüfen Sie, dass keine Geräte mit dem Koordinator verbunden sind. Sollten Geräte in der Netzwerkübersicht ersichtlich sein, setzen sie den Versuch zurück. Dies wird in den FAQs beschrieben.

c) Stellen sie den Kanal entsprechend dem während dem Versuch gegebenen Wert sein. Dies verhindert eine gegenseitige Beeinflussung der verschiedenen Versuchsaufbauten. Zuhause können sie diesen Schritt überspringen.

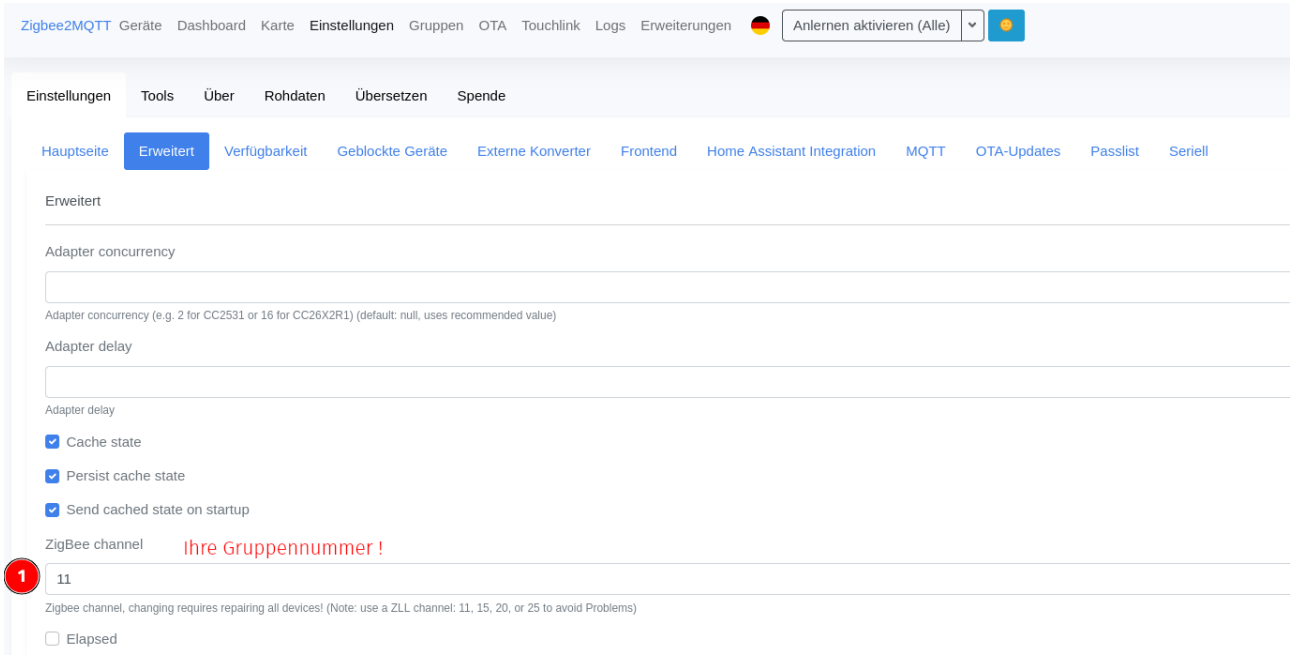


Abbildung 5.1: Zigbee Kanal Einstellung

d) Standardmäßig verwendet Zigbee2mqtt einen zufällig gewählten Netzwerkschlüssel. In diesem Versuch wird der Schlüssel vorgegeben, um die Pakete entschlüsseln zu können. Bitte setzen sie folgenden Schlüssel:

```
1     0x 00 00 ... 00 <Gruppennummer>
```

Die Einstellung finden sie unter der Kanal Einstellung als “Network key(string)“.

Achten sie darauf, im Anschluss die Einstellung am Ende der Seite zu bestätigen. Dafür klicken sie auf den “Submit “ Button am Ende der Seite

e) Starten sie Wireshark. Bei den verfügbaren Schnittstellen sollte sich eine ZigBee Schnittstelle finden. Über das vorangestelle 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 gesetzen 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 beschrieben Aktionen durchzuführen.

5.2.2 Aufgabe 2 - Joining einer Phillips Hue Lampe

a) Schalten sie eine der beiden Lampen ein. Die Lampe sollte leuchten, wenn sie in keinem Netzwerk registriert ist..
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 erfolgreiches Interview.

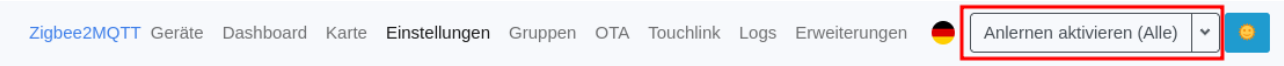


Abbildung 5.2: Zigbee Anlernen aktivieren

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:

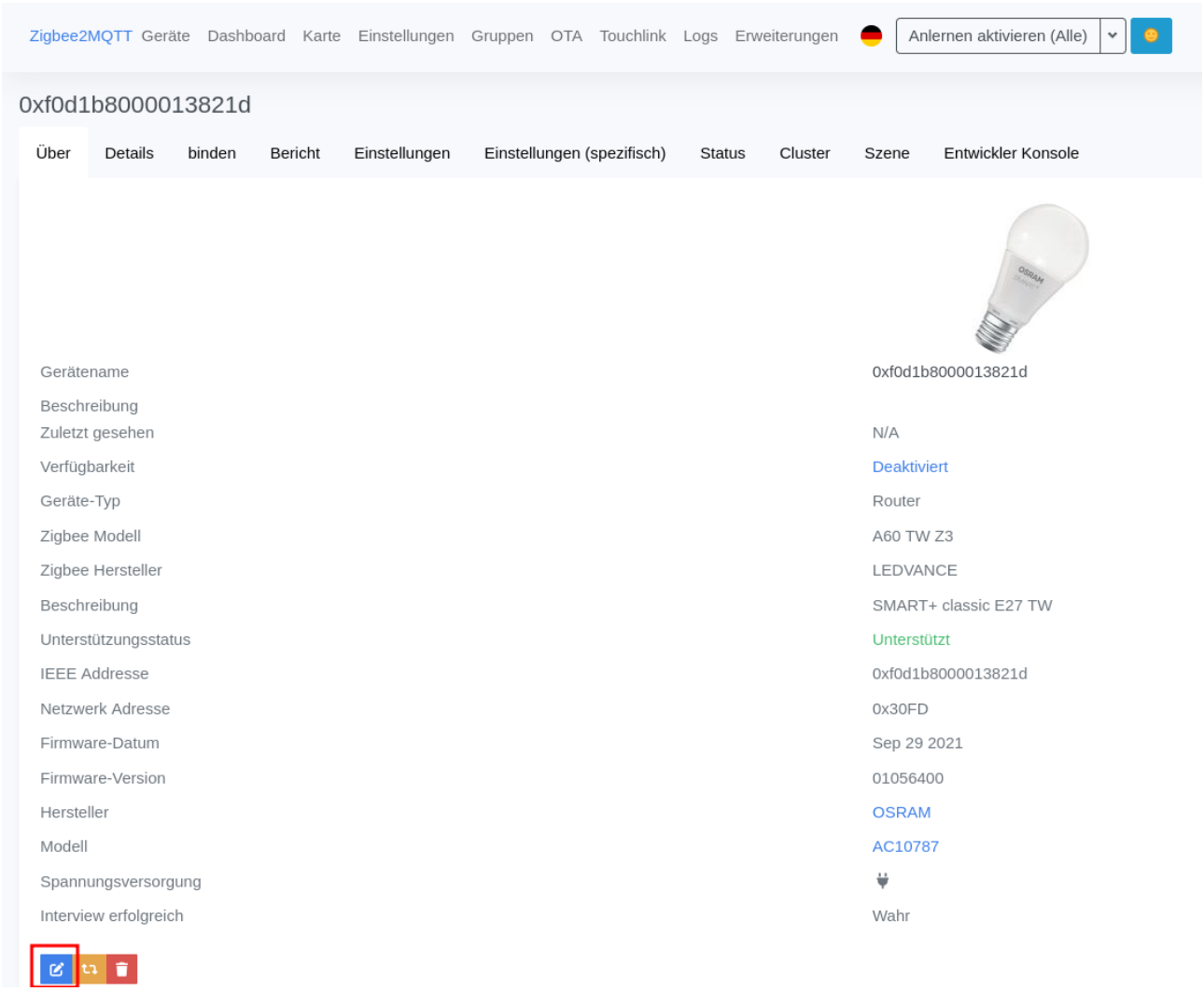


Abbildung 5.3: Zigbee Device Übersicht

d) Vergeben sie in der Übersichsseite 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.

Hinweis

Sollte die Lampe sich nicht Verbinden, kann es notwendig sein die Lampe zurückzusetzen. In den FAQs finden sie zwei Methoden dazu.

5.2.3 Aufgabe 3 - Joining einer Fernbedienung über die Lampe

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

a) Drücken sie alle 4 Tasten der Fernbedienung, die die LED in verschiedenen Farben blinkt.
b) 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.

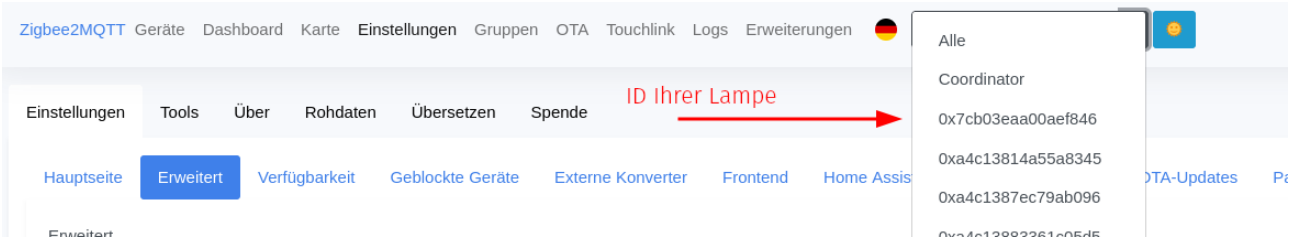


Abbildung 5.4: Zigbee Anlernen aktivieren - nur Lampe

Aufgabe

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

5.2.4 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 X Ihrer Lampe mit dem Endpunkt X Ihrer Fernbedienung.
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.

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.

d) Navigieren sie nun wieder zur Binding-Übersicht der Fernbedienung. Entfernen sie das Binding zu der Lampe. Binden sie die Fernbedienung nun mit der soeben angelegten Gruppe.

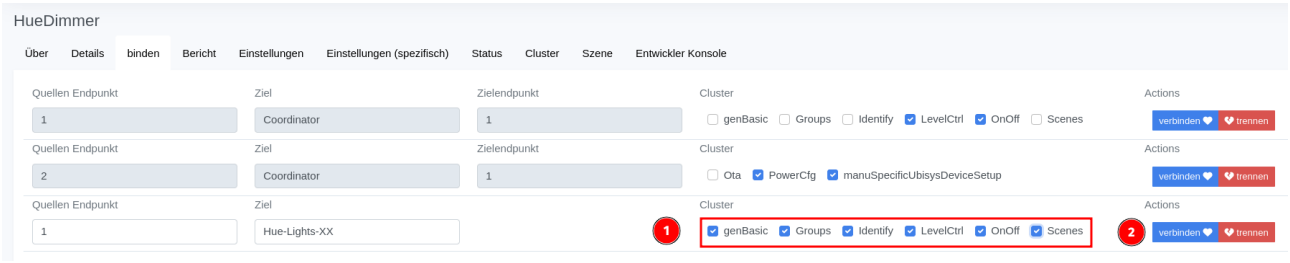


Abbildung 5.5: Zigbee Group Binding

Achten sie darauf, alle hier genannten Cluster anzuwählen.

Aufgabe

Speichern sie den Wireshark Capture ab als **“<Gruppe> - ZigbeeLab - Aufgabe 5“**.
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 Quelladdresse 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 **Tranport-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 Fernbedienung 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 Fernbedienung, 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 Fernbedienung. 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 Fernbedienung! Welche Informationen enthält diese Nachricht?

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 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?

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

Versuchsdurchführung

6.0.1 Aufgabe 2 - Joining einer Phillips Hue Lampe

6.0.2 Aufgabe 3 - Joining einer Fernbedienung über die Lampe

6.0.3 Aufgabe 4 - Binding der Fernbedienung

6.0.4 Aufgabe 5 - Gruppenbildung

6.0.5 Fragen

Aufgabe 2

Fragen

1. Untersuchen sie den **Beacon-Request**
Erläutern sie den Frametype und den Command Identifier.
Erläutern die Ziel- und Quelladdresse und was sie daraus erschließen können.

2. Wie teilt der Koordinator den umliegenden Geräten ab, 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 **Tranport-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 Fernbedienung 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 Fernbedienung, 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 Fernbedienung. 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 Fernbedienung! Welche Informationen enthält diese Nachricht?

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 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?

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 7

Life Cycle Management

In diesem Kapitel geht es um die Pflege, die Bereitstellung sowie die Zuücksetzung 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)

```
ansible-playbook DeployLab.yaml -K

#BECOME Password:
<secret>
```

7.0.2 Zurücksetzen des Versuchs

7.0.3 Update der eingesetzten Software

Abbildungsverzeichnis

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.