Hochschule RheinMain
Fachbereich ITE
Studiengang EE-CS

Scientific Project

Entwicklung eines Zigbee Praktikums unter Verwendung von OpenSource Technologie

verfasst von **Benedikt HEUSER**
Matrikelnummer 105320

am 25.04.2022

Kompiliert am 11.12.2022 um 14:22:16 - Erstellt mit LaTeX

Inhaltsverzeichnis

Kapitel 1

Einführung

In diesem Projekt soll ein Praktikumsversuch für die Vorlesung Internet of Things für Professor Jürgen Winter entwickelt werden. In dem Versuch soll das Verhalten des ZigBee Protokolles untersucht werden. Es wird ein kleines ZigBee Netz mit einigen wenigen Komponenten aufgebaut und die Kommunikation mitgeschnitten. Anschließend wird die Kommunikation mit dem Protokollanalyse Werkzeug Wireshark untersucht. Es wird ein "LabGuide " geschrieben, dass 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** - De 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.
- **A060** - Studenten sollen eine Versuchsbeschreibung sowie alle nötigen Utensilien erhalten. Im Heimversuch müssen KVM-Komponenten von den Studenten selbst gestellt werden.
- **A100** - Der Versuch soll automatisch auf den Raspberry ausgerollt werden können.
- **A110** - Es wird, bis auf den Ausrollvorgang, keine Internetverbindung benötigt.
- **A120** - Es soll ausschließlich gewartete und quelloffene Software zum Einsatz kommen.
- **A200** - Der Versuch soll die Grundlagen eines Mesh-Netwerkes vermitteln.

- **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

IoT für den Heimanwender befindet sich größtenteils in der Hand etablierter Technologie-Unternehmen. Diese entwickeln meißt ein eigenes Öko-System, welcher mehr oder weniger Kompatibel mit Produkten anderer Hersteller sind. Hier ist der Hersteller Phillips mit seiner Produktmarke “Hue “ als gutes Beispiel zu nennen. Die Produktgruppe umfasst eine Bridge mit zugehöriger App, sowie den klassischen Komponenten wie Lampen, Steckdose und Schalter. Der Markt wurde durch sogenannte Heimassistenten stark belebt. Diese zumeißt sprachgesteuerten Geräte erfreuen sich starker beliebtheit. Zu diesen Geräten gehört beispielweiße Amazon Alexa. Diese Heimassistenten verfügen Anbindungen zu den proprietären Gatewys, oder können teilweise sogar direkt an PANs wie Zigbee teilnehmen und die Geräte direkt steuern. Dieses Kapitel soll einen Überblick über die Technologien am Markt verschaffen.

2.1 IoT Funkprotokolle

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

- **Wlan** Wlan ist ein weit 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 optimier für hohe Übertragungsraten, und nicht für besonders leistungsschwache Endgeräte. Die ist gerade für Batteriebetriene Geräte ein enormer Nachteil. Zusätzlich ist es oft nicht gewünscht, Smarte Devices an ein Netzwerk mit Internetzugang anzuschließen, da diese sobald sie mit dem Internet kommunizieren auch entsprechende Sicherheitsrisiken darstellen.
- **Bluetooth** Ebenso wie Wlan hatte Bluetooth auch schon vor dem IoT Boom eine erhebliche Verbreitung. Durch Implementierung des Standard Bluetooth LE ist auch die Leistungsfähigkeit von Devices hier eine geringere Problematik. Bluetooth ist allerdings nicht für hohe Reichweite oder für viele Geräte konzipiert. Der geographische Raum, den eine Bluetooth Bridge abdecken kann, ist also eher gering.
- **Z-Wave**

- **ZigBee** Zigbee ist ein auf den 802.40.5 Standard aufbauendes Protokoll, welches grundlegend für die Anbindung vieler 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 Newcomer. Es basiert ebenfalls auf den 802.15.4 Standard. 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 erreichbar und addresierbar.

2.2 Zigbee Anwendungen

2.2.1 Kommerzielle Anwendungen

Amazon Echo

Der Heimassistent Amazon Echo 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. [Ama12]

Phillips Hue

Phillips stellt eine Zigbee Bridge und eine Vielzahl an Devices aus dem Segment Beleuchtung und Steckdosen.

Dresden Electronic

Dresden Electronic bietet Software und Hardware zum Aufbau von Zigbee Netzwerken an. Es gibt Zigbee USB Adapter und RaspberryPi Hats mit ATMega Chips, sowie eine Steuerungssoftware “deCONZ “. Als komplette Produktlinie für den nicht technisch visierten Endkungenmarkt 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**
- **homee** - modular Smart Home Central

Nachteil all 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 sehr mühsam, bei Anschaffung eines dieser Systeme die Kompatiblität anderer Geräte sicherzustellen.

2.2.2 Nicht kommerzielle Anwendungen

Ein großer Vorteil von OpenSource 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 ist ein quelloffenes Projekt auf GitHub, welches aus einer Serveranwendung mit Web-GUI, und einer Firmware für diverse Texas Instruments Chips besteht. Grundlegende Koordinator Fähigkeiten sind auf der Hardware implementiert, Hardware Abstraktionen sowie die Weiterreichung von Nachrichten an ein MQTT Broker sind in der Webanwendung implementiert. Auf der anderen Seite des MQTT Brokers, zur Visualisierung und Steuerung der Devices kann dann Homeassistant oder ioHAB eingesetzt werden. zigbee2Mqtt bietet eine Menge Möglichkeiten, Informationen zu sammeln und direkt Einflussnahme auf die Devices zu nehmen.

ZHA

ZHA ist ein direkt in HomeAssistant integriertes Plugin, um Zigbee Koordinatoren direkt in HomeAssistant einzubinden. Vorteil von ZHA ist, dass die Liste von unterstüztzter Zigbee-Chips deutlich länger ist. ZHA unterstützt neben Texas Instruments auch Hardware von Dresden Elektronik, Silicon Labs, DIGI und ZiGate. ZHA ist für den Anwender extrem vereinfacht, es sind kaum technische Informationen ersichtlich oder konfigurierbar.

Kapitel 3

Grundlagen

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

3.1 ZigBee

Die ZigBee Alliance wurde durch ein Konsortium von Herstellern gegründet, um einen einheitlichen Übertragungstandard im Bereich Heimautomatisierung voranzubringen. ZigBee basiert zwar auf einem offenem IEEE Standard, bringt aber proprietäre Komponenten mit. ZigBee ist in Form von weiteren Protokollschichten implementiert, welche auf IEEE 802.15.4 aufsetzen.

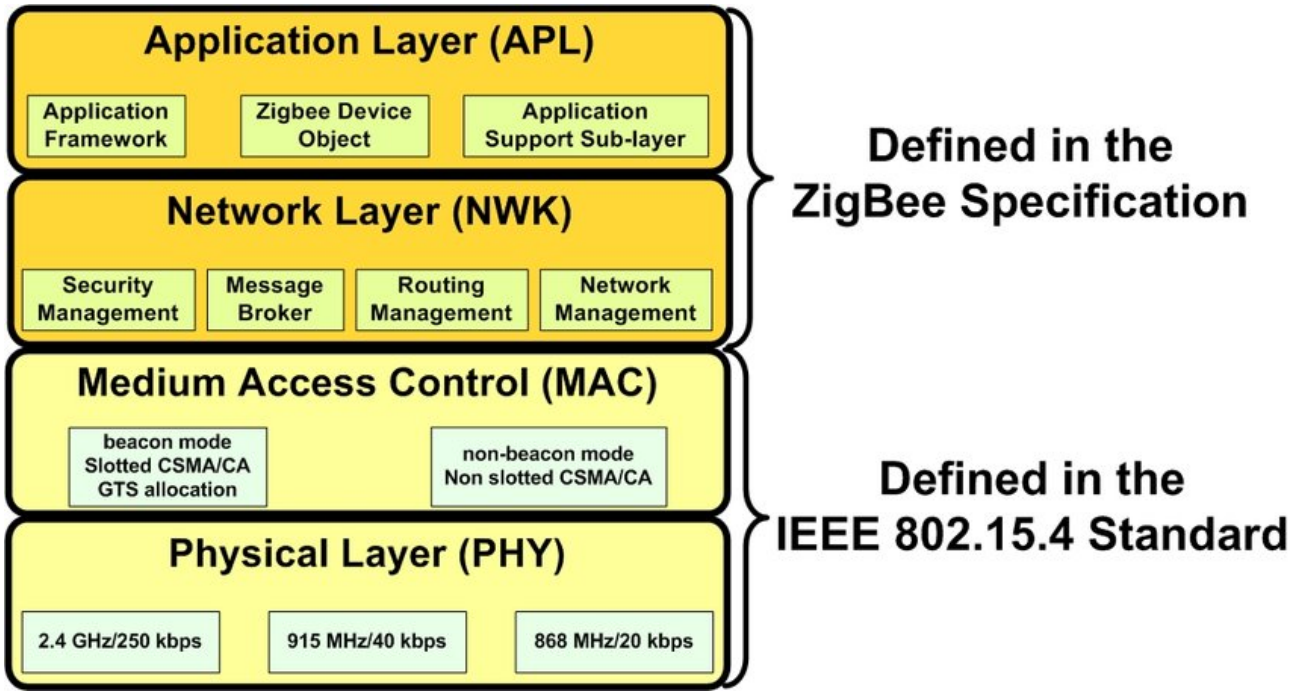


Abbildung 3.1: ZigBee Protocoll Stack
Bildquelle: `https://www.researchgate.net/figure/IEEE820154-ZigBee-protocol-stack-architecture_fig2_265150617`

Genutzt wird es, um Geräte, wie zum Beispiel Lampen, Schalter oder Thermostate zur Kommunikation zu befähigen. Dies kann genutzt werden um die Geräte zu steuern, sowie aktuelle Zustandsdaten abzugragen. Markantes Merkmal am Protokoll ist, dass die Geräte keine direkte Funkverbindung zu einem zentralen Controller brauchen. Die einzelnen Geräte können als Router fungieren. Dies sorgt dafür, dass mir einer geringen Sendeleistung ein geografisch großes Gebiet abgedeckt werden kann. Sende- und Empfangsleistung ist vorallem bei kleinen Batteriebetriebenen Geräten der einschränkende Faktor.

3.2 Texas Instruments CC Chips

Texas Instruments bietet ein Spektrum von Microcontrollern, die sich mit entsprechender Firmware als ZigBee Device nutzen lassen können.

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 manchen Fällen Bluetooth LE zur Koppelung, daher ist die simultane 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

Hier in der Tabelle sind die unterstützten Protokolle der einzelnen Modelle sowie deren Leistungsfähigkeit aufgeführt. Spannenderweise ist zu sehen, dass die Top Modelle schon den Standard Thread unterstützen, der vermutlich durch das Projekt "Matterërheblich an bedeutung gewinnt.

Texas Instruments bietet als Basis für ZigBee Eigententwicklungen eine Z-Stack Bibliothek. Diese stellt alle grundlegende Funktionen um das ZigBee Protokoll zu implementieren. Mit TexasInstruments Code Composer Studio steht eine IDE bereit, um den Entwicklungsprozess zu unterstützen. Auf den entsprechend Leistungsfähigeren Chips lassen sich so noch zusätzliche Funktionalitäten einprogrammieren.

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

In dem OpenSource Projekt ßigbee2mqtt"werden ausschließlich Chips von Texas Instruments unterstützt. Es sei erwähnt, dass die meißten gängigen Anbieter von Microchips entsprechende Modelle im Angebot haben.

3.3 Versuchshardware

3.3.1 RaspberryPi

Der RaspberryPi ist ein ARM basierter Computer im Mini-Format. Er dient in diesem Versuch als Server, der die Applikationen hostet und gleichzeitig als Versuchs PC, von dem der Versuch aus ausgeführt wird. Die eingesetzten Dienste sind alle als Webservice implementiert, und daher vollständig auf Kommandozeile parametrierbar, sowie mit WebGUI bedienbar.

Der RaspberryPi besitzt die Nutzer PC typischen Schnittstellen wie Ethernet, HDMI, sowie USB. Als Hauptspeicher wird eine SD-Karte eingestzt. Dies ist ein erheblicher Vorteil beim vorbereiten mehrerer RaspberryPis für den Versuch.

Auf dem RaspberryPi wird das Linux-basierte Betriebssystem RaspbyanOS eingesetzt. Durch die enorme Verbreitung ist hier mit regelmäígen Updates in Zukunft zu rechnen.

3.3.2 RaspberryPi Zigbee Hat

Als Zigbee Koordinator kommt ein auf dem TI CC2652 basierendem RaspberryPi Hat vom Hersteller “cod.m “ zum Einsatz. Dieser wurde vom Hersteller zum Einsatz mit “homegear “ oder “zigbee2Mqtt “. Ein Datenblatt sowie ein Manual ist im Anhang.

3.3.3 CC2235 Sniffer Stick

Mit diesem Stick wird die ZigBee Kommunikation zwischen dein einzelnen Devices sowie dem Koordinator mitgeschnitten auf einem bestimmten Kanal mitgeschnitten. Es lässt sich beinahe jeder Texas Instruments ZigBee Chip bereits auf einen USB-Stick aufgelötet günstig aus China beziehen.

`https://github.com/virtualabs/cc2531-killerbee-fw`

3.3.4 Phillips Hue Komponenten

Die Phillips Hue Serie setzt vollständig auf den ZigBee Standard auf. Die Produkte haben ein eigenes Öko-System mit entsprechender App und Gateway. Sie sind Kompatibel zu dem Software-Gateway zigbee2mqtt. Die Lampen werden in dem Versuch als Demonstrationsobjekte eingesetzt. Sie können Ein- und Ausgeschaltet werden, sowie gedimmt werden. Zusätzlich wird einer Phillips Hie Fernbedienung verwendet, die zur Steuerung der Lampen dient.

3.4 Eingesetzte Software

3.4.1 Raspbian OS

RaspbianOS ist eine leichtgewichtige Linux Distribution, welche direkt vom Hersteller des Raspberry-Pis speziell auf die Bedürfnisse des Board angepasst werden. Es enthällt eine Desktop Umgebung sowie die grundlegende Paketen. Es basiert auf Debian, damit sind auch die entsprechenden Paketquellen verfügbar.

3.4.2 Docker

Docker ist eine Container Umgebung, um Anwendungen containerisiert auf Linux-Servern laufen lassen zu können. Docker reduziert erheblich den Aufwand, Anwendungen auf Servern auszurollen. Alle Abhängigkeiten sind im Container enthalten, sodass hier keine Komplikationen mit anderen Anwendungen zu befürchten sind. Prozesse laufen in eigenen Namespaces und sind dadurch abgekoppelt vom Host Betriebssystem. Im Unterschied zur Virtualisierung werden Systemprozesse gemeinsam genutzt. Dadurch ist die Effizienz höher, als bei taditioneller Virtualisierung.

3.4.3 Docker-Compose

Docker-Compose ist ein Tool, um große Containerumgebungen per YAML zu definieren und automatisch deployen zu lassen. Klassisch kann ein Container per CLI Befehl mit entsprechenden Parametern gestartet werden:

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

Dies ist allerdings für große Umgebungen unhandlich. Alternativ werden Container also Services in einer YAML definiert.

```
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
```

können dann alle definierten Services gestartet werden. Dies folgt der allgemeinen Container Idee, Umgebungen zu definieren und beliebig ausrollen zu können, anstelle den Zielzustand manuell zu

konfigurieren.

3.4.4 zigbee2mqtt

zigbee2mqtt ist ein offenes Softwareprojekt und stellt ein “Software-Zigbee-Gateway “dar. Es übernimmt die Funktionalität, die normalerweise entsprechende “Bridges “übernehmen. Während traditionelle Bridges, wie zum Beispiel die Phillips Hue Bridge eine REST API zur Verfügung stellen um mit Ihren entsprechenden Apps zu kommunizieren, macht zigbee2mqtt die Geräte per mqtt nach außen verfügbar. Auf abstrakterer 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 “ bzw. entsprechende Eigenentwicklungen einsetzen.

Quellcode und Dokumentation: `https://github.com/Koenkk/zigbee2mqtt` Homepage: `https://www.zigbee2mqtt.io/`

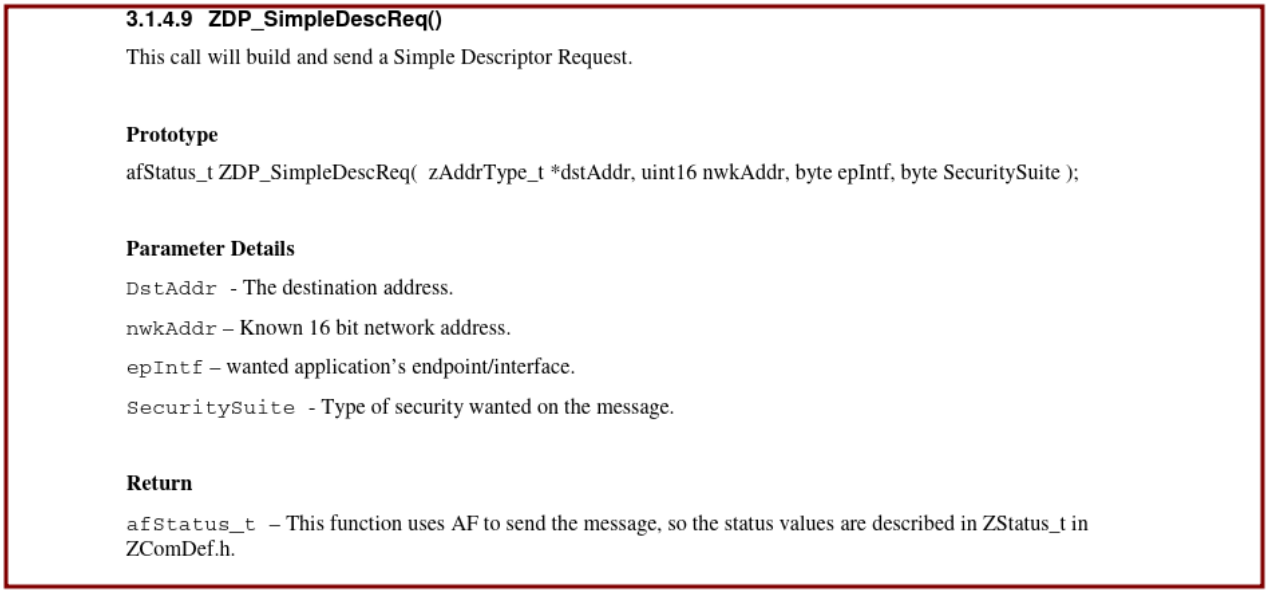
TI CC Firmware

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 einem Git-Repo angeboten.

zigbee-herdman

Dieses Modul verbindet sich direkt mit dem Zigbee Adapter, und steuert Ihn über die “TI zStack monitoring and test API “. [Ins12]

Zur verdeutlichung, wie dies in Realität aussieht, hier 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-herdman-converters

Dieser Konverter kann proprietäre Cluster die durch Geräte exposed werden umwandeln in Standard Cluster. Mit diesem Converter lassen sich proprietäre Cluster von Geräte so adaptieren, dass sie nach Wunsch gesteuert und ausgelesen werden können.

zigbee2mqtt

Das Hauptmodul stellt die WebGui sowie eine Webanwendung mit einer SQLite Datenbank. Die Webanwendung und die Datenbank verwalten den Zustand des Netzwerkes und die angebundenen Geräte. Die WebGUI dient zur Administration des Koordinators.

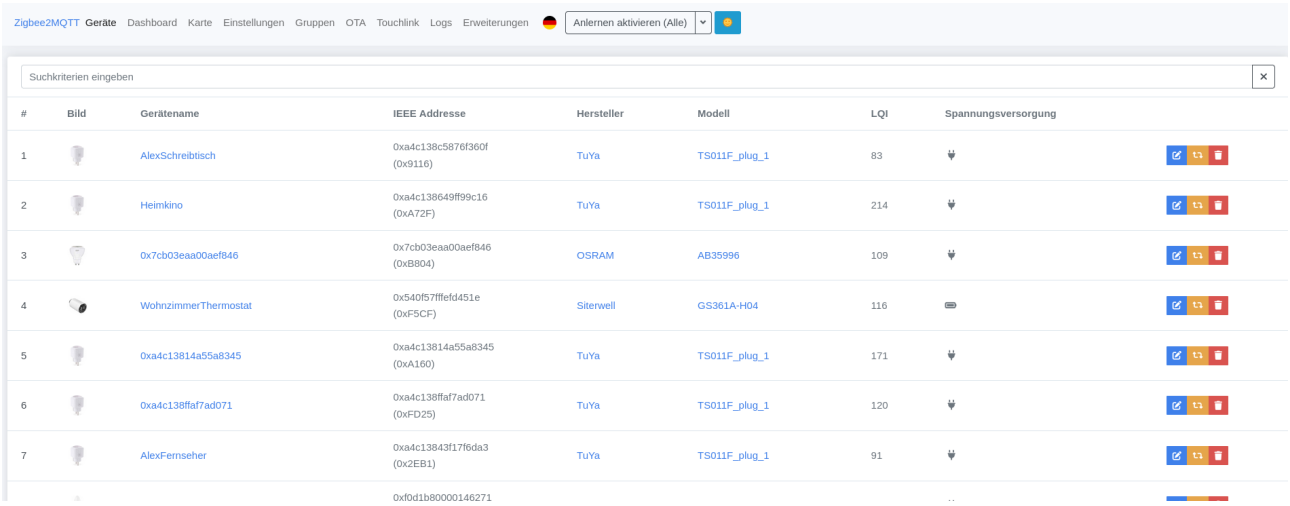


Abbildung 3.5: zigbee2mqtt Webfrontend

Die WebGUI enthällt eine große Anzahl von Funktionen, die weitaus tiefer reichen als für die Nutzung notwendig sind. Prinzipiell sind die meißten ZigBee Geräte Kompatibel, wenn ein Community Mitglied dieses bereits in der Anwendung angelegt hat. Es ist auch möglich, eigene Beschreibungen für nich nicht unterstützte Geräte zu erstellen.

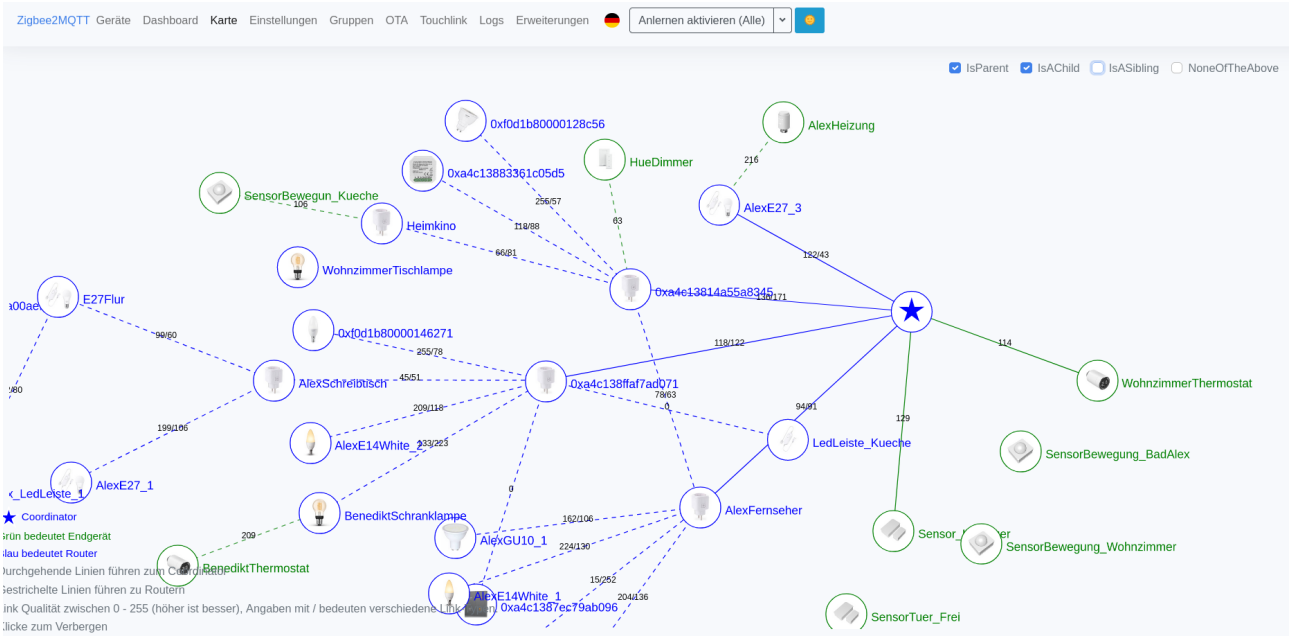


Abbildung 3.6: zigbee2mqtt Netzwerkvisualisierung

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

3.4.5 Wireshark

Wireshark ist eine quelloffene Anwendung um Datenstöme Mitzuschneiden und zu Untersuchen. Es kann durch Verwendung von Packetsniffern wie nPcap verschiedenste Medien wie zum Beispiel Ethernet und USB mit entsprechenden Protokollen verarbeiten.

3.4.6 zbWireshark

Dies ist ein Script aus einem Git-Hub Repository. Mit diesem Script lässt sich Wireshark starten und anschließend Pakete über den ZigBee Sniffer Stick mitschreiben.

`https://github.com/silicrax/killerbee/tree/master/tools`

3.4.7 Ansible

Ansible ist ein Tool zur IT-Automatisierung. Arbeitsabläufe lassen sich ebenfall gut strukturiert in YAML definieren. Ansible verbindet sich per SSH mit den zu konfigurierenden Hosts und führt Python Module aus. Hosts lassen sich in Gruppen, und Tasks in sogenannten Rollen sammeln. So ist es möglich, eine Gruppe von Hosts die Rolle “Webapplikation “, und einer Gruppe von anderen Hosts der Gruppe “Datenbank “ zuweisen. Eine Zuweisung von Rollen sihet folgendermaßen aus:

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

In diesem Beispiel wird auf dem lokalen Host Docker installiert, und das ZigBee Praktikum vorbereitet.

3.4.8

Kapitel 4

Versuchsaufbau

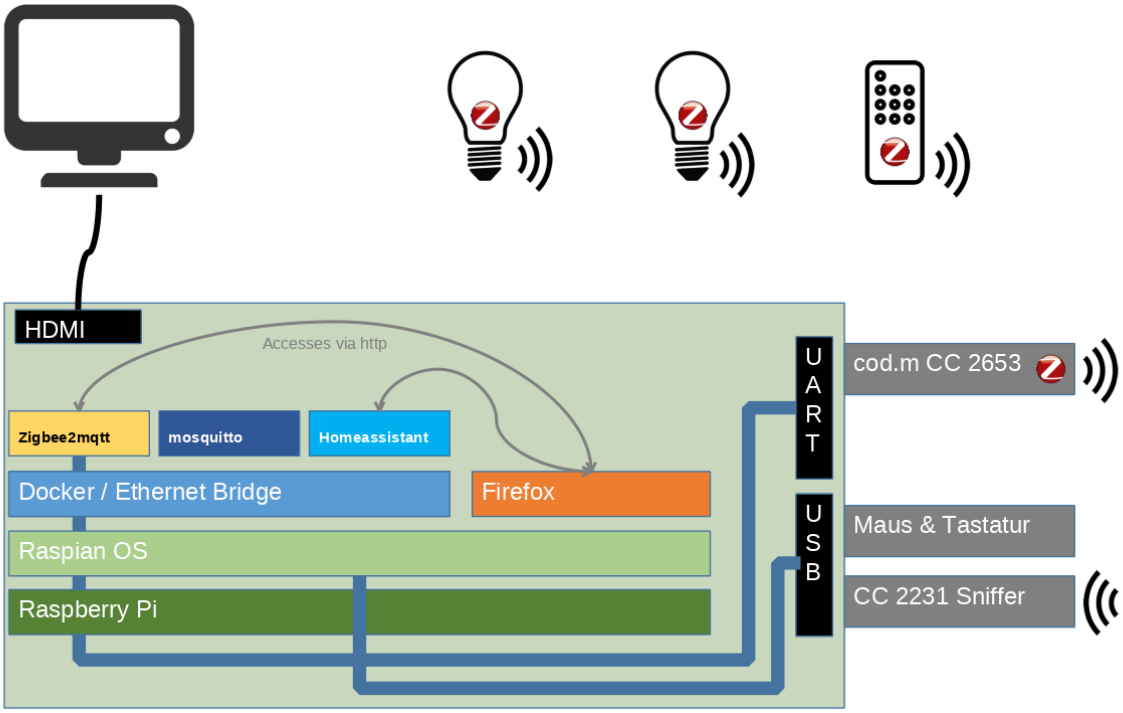


Abbildung 4.1: Versuchsaufbau

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

Die Hostnames werden in der lokalen hosts Datei angelegt, damit die Namen Lokal ausfgelöst werden.

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

4.0.1 Containerverwaltung

Die Container werden mit dem CLI Tool “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 Service Proxy basiert auf einem Image von des Reverse Proxys NGINX [Wil22], erweitert durch “jwilder “. Vorteil dieses erweiterten Images ist es, dass dieser automatisiert seine Konfigurationen anpasst. Zu diesem Zweck ist der “docker.sock “ in den Container gemountet. Hierüber erfährt der Proxy, ob Container gestartet worden sind, und welche Umgebungsvariablen gesetzt worden sind. Wird ein weiterer Container mit der Umgebungsvariable “VIRTUAL_HOST=z2m.local “ gestartet, wird automatisch eine Weiterleitung für alle Anfragen auf “z2m.local“ eingerichtet auf den entsprechenden Container. Dies erlaubt es beliebig viele Services auf einfache Art und weiße auf einem Host verfügbar zu machen. Zusätzlich sind alle Servicecontainer in einer eigenen L2-Domäne und sind von außen nicht erreichbar. Dies erhöht die Sicherheit, da Zugriff von außen nur über die beiden gemappten Ports des Proxys erfolgen kann. Dem Proxy wird daher entsprechend das Netzwerk ’backbone’ zugewiesen. Im Abschnitt “Ports “ können Ports nach außen gemappt werden. In diesem Fall wird der Port 80 und 443 des Proxys auf die öffentlichen Ports 80 und 443 des Hosts gemappt. Hierfür werden von Docker Regeln in die “iptables “ 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 im Table “DOCKER “ werden durch Docker geschrieben, diese sollen nicht modifiziert werden. Eigene Regeln können in den Table “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 diesem Output ist erkennbar, dass Docker einen Listener auf den Ports 80 und 443 laufen hat.

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     # CC251
10    # - /dev/ttyUSB0:/dev/ttyACM0
11    # CC2530 / GBAN GB2530S
12    #- /dev/ttyUSB_cc2530:/dev/ttyACM0
13  restart: always
14  environment:
15    - VIRTUAL_HOST=z2m.local
16    - VIRTUAL_PORT=8080
17  group_add:
18    - dialout
```

Der Container wird ebenfalls an das Netzwerk “Backbone “ angeschlossen. Der Ordner mit den Konfigurationen wird auf den Host gemountet und bleiben damit konsistent, auch wenn der Container als solches getauscht wird. Wenn der gemountete Pfad außerhalb des Containers nicht existiert, wird der Ordner aus dem Containergemountet. Exisitert der Pfad, wird der Ordner vom Host in den Container gemountet. Durch das Linux-Mantra “Everything is a file “ ist es auch möglich Geräte über die selbe Art und Weise in einen Container zu mounten wie einen regulären Dateipfad. **** ist der Socket zu dem cod.m Modul. 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. Der Container Prozess läuft mit entsprechenden Rechten.

Im Playbook “DeplayLabutils “ wird folgende Konfiguration eingespielt:

```
  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 Modus “homeassistant “ aktiviert, damit werden die Nachrichten an den MQTT Broker so gestaltet, dass homeassistant diese versteht. Das Beitreten neuer Geräte ist im Standard aus und muss explizit erlaubt werden. Desweiteren wird ein MQTT Server angegeben, sowie ein “base-topic “ definiert. Docker löst Containernamen in Dockernetzwerken auf zu IP-Adressen, 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
```

```
14      - ./mosquitto/log:/mosquitto/log
```

Als MQTT Broker wird “mosquitto “ eingesetzt. Für diesen Container wurde der erlaubte genutzte Arbeitsspeicher begrenzt. Die Konfgurationen wurden auch hier entsprechend auf den Host gemountet damit diese Persistent bleiben. Als Konfiguration wird ein Template verwendet, welches nur an entsprechenden Stellen modifiziert worden 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 Problem 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 Standardort 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 “ eingetragen.

```
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 lediglich 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
```

Zusätzlich wird das Binary Package “zbwireshark “in einen Pfad kopiert, der als Pfad in den Umgebungsvariablen definiert ist. Somit kann “zbwireshark “ lokal aufgerufen werden.

Kapitel 5

Versuchsdurchführung

In diesem Kapitel wird der Versuchsaufbau beschrieben, der LabGuide eingebettet, sowie eine Musterlösung gegeben.

5.1 Versuchsaufbau

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

- RespberryPi 3
- 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. Bitte stellen sie selbst einen Monitor mit HDMI Anschluss, sowie Maus und Tastatur bereit.

5.2 Aufgabenstellungen

Bitte arbeiten sie alle folgenden Aufgabenstellungen ab. Fertigen sie im Anschluss einen Versuchsbericht an. Es reicht, wenn die gestellten Fragen implizit beantwortet werden.

5.2.1 Aufgabe 1 - Vorbereitungen

Vorbereitung

Schließen sie an den RaspberryPi den Monitor, eine Tastatur und Maus, sowie den Sniffer Stick an. Durch Anschluss der Stromversorgung startet der Raspberry automatisch. Melden sie nach starten des Betriebssystemen mit folgenden Zugangsdaten an:

- User: student
- Password: zigbeelab

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

```
1     > docker ps
```

Es sollten 3 Container im Status “Running“ sein. Beschreiben sie in eigenen Worten welche Container sie hier sehen.

ZigBee2Mqtt Einrichtung

Starten sie den Webbrowser Firefox und besuchen die Webseite:

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

Überprüfen Sie dass keine Geräte mit dem Koordinator verbunden sind. Im Zweifelsfall können sie den Versuch zurücksetzen, oder die Geräte per Hand herauslöschen. Dies wird in den FAQs beschrieben.

Stellen sie den Kanal, den der Zigbee Koordinator nutzen soll nun auf den durch Ihren Professor vorgegeben Wert. Dies verhindert, dass sich die verschiedenen ZigBee-Netze gegenseitig beeinflussen. Zuhause können sie diesen Schritt überspringen.

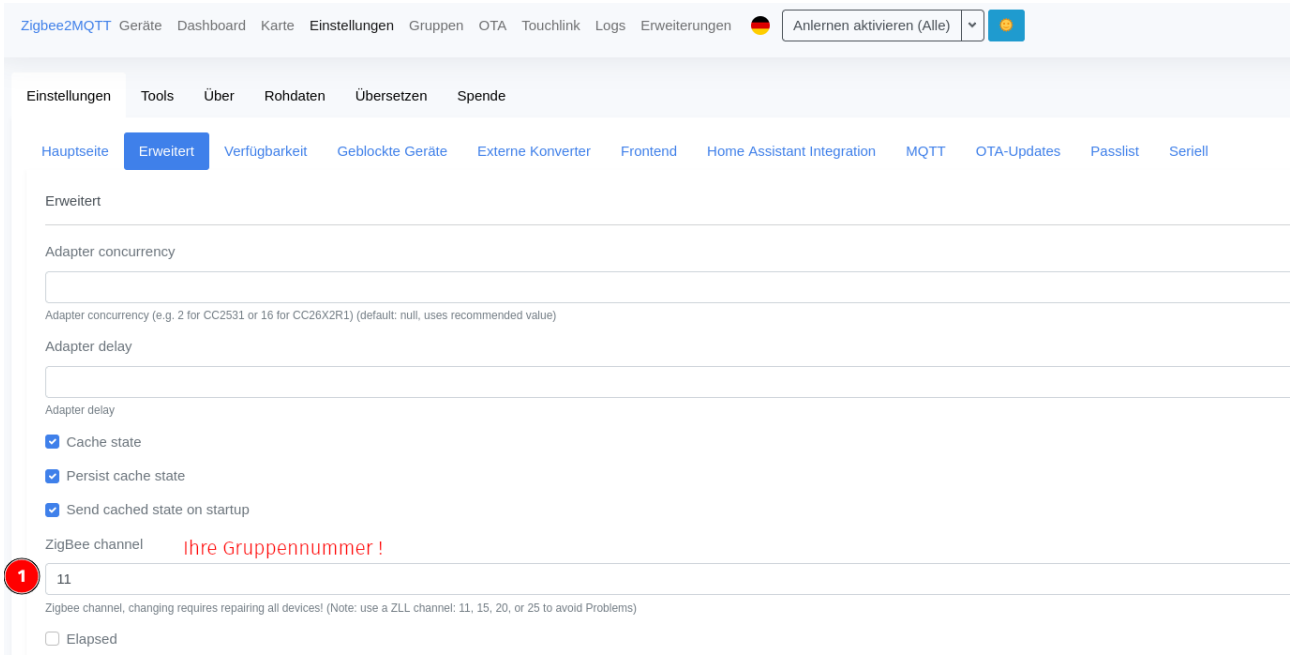


Abbildung 5.1: Zigbee Kanal Einstellung

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:

```
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

Wireshark Einrichtung

Mit folgendem Befehl können sie ein Wireshak Capture auf entsprechenden Kanal starten:

```
> zbwireshark -c <Kanal>
```

Starten sie ein Konsolenfester und testen sie, ob der Befehl erfolgreich ausgeführt wird. Wireshark sollte starten, und Pakete sollten ersichtlich sein. Für “<Kanal> “ setzen sie den von Ihnen gewählten Kanal ein.

Beenden sie den automatisch 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.

```
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 Mitschnits so kurz wie möglich zu halten, und in dieser Zeit nur die in der Aufgabenstellung beschrieben Aktionen durchzuführen. Anderenfalls wird Ihr Mitschnitt sehr unübersichtlich.

5.2.2 Aufgabe 2 - Joining einer Phillips Hue Lampe

Durchführung

Schalten sie eine der beiden Zigbee Lampen ein. Die Lampe sollte leuchten. Dies ist das Standard verhalten, wenn die Lampen in keinem ZigBee Netz integriert sind. Starten sie nun ein Wireshark Capture 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 ebenfalls einen erfolgreiches Interview.

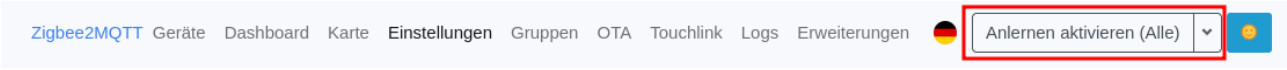


Abbildung 5.2: Zigbee Anlernen aktivieren

Aufgabe

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

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!

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

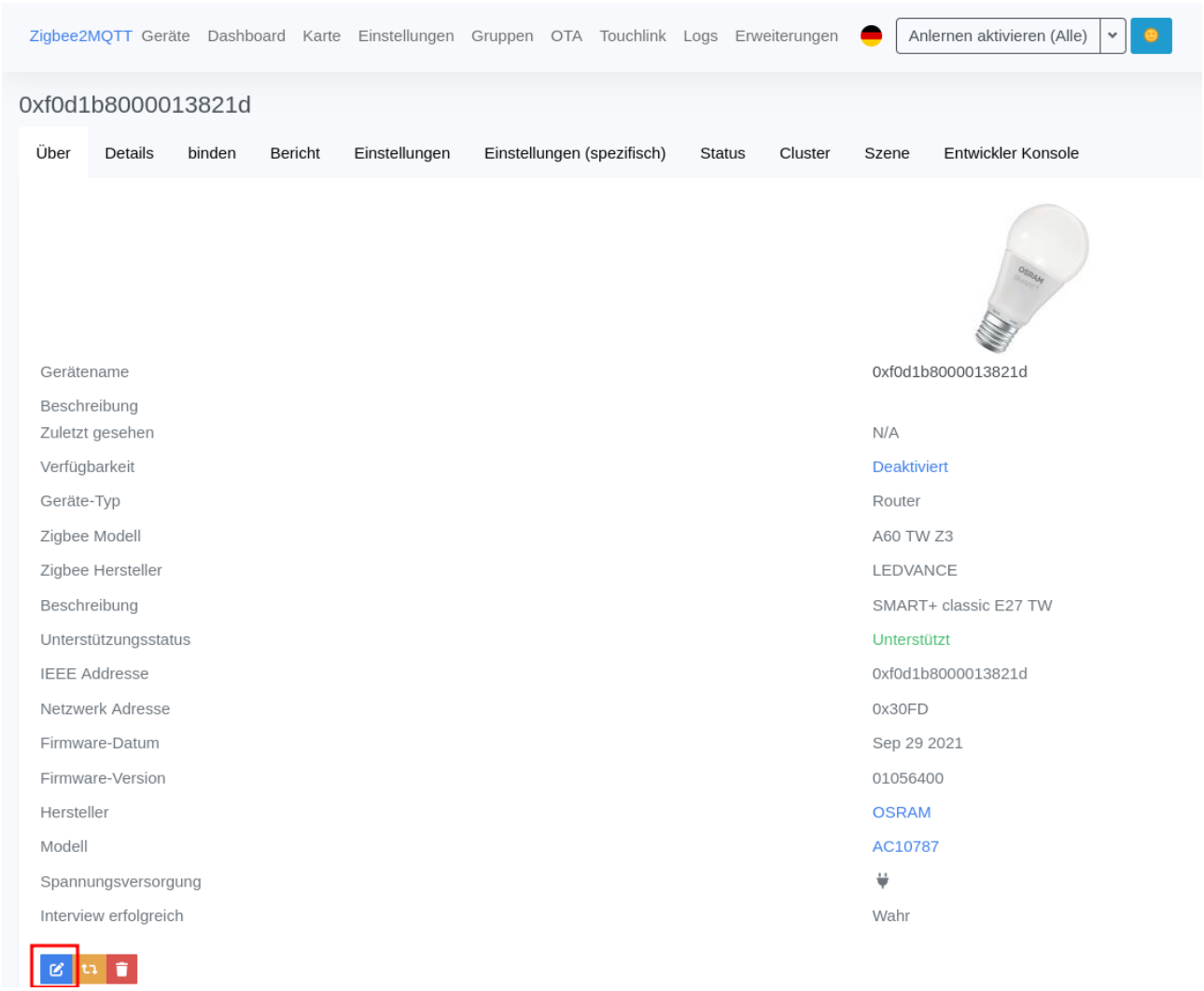


Abbildung 5.3: Zigbee Device Übersicht

Vergeben sie in der Übersichsseite der Lampe einen nutzerfreundlichen Namen. Die geschieht über den blauen Button im unteren Teil der Übersicht. Dimmen und schalten sie die Lampe über die Weboberfläche. Die ist unter dem Reiter “Details“ möglich. Starten sie einen weiteren Capture Vorgang und scheiden in diesem einen Schaltvorgang mit.

Aufgabe

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

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

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

5.2.4 Durchführung

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. Drücken sie den Setup Button auf der Fernbedienung, bis die LED dauerhaft grün leuchtet. Ein erfolgreiches anlernen wird auch hier in der Weboberfläche und durch ein blinken der grünen LED signalisiert. Schneiden sie diesen Vorgang wieder mit Wireshark mit und speichern den Capture unter "<Gruppe> - ZigbeeLab - Aufgabe3".

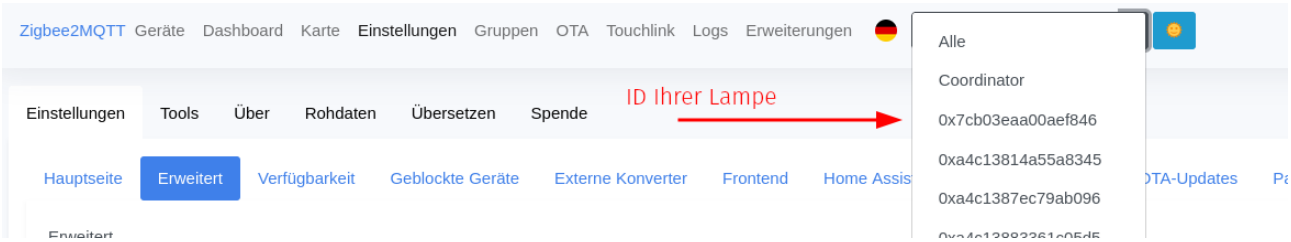


Abbildung 5.4: Zigbee Anlernen aktivieren - nur Lampe

Aufgabe

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

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 kryp-

tographisch 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?

Hinweis

Eventuell muss Ihre Fernbedienung zuerst zurückgesetzt werden, bevor sie wieder einem neuen ZigBee Netzwerk beitreten kann. Der Reset-Knopf der Fernbedienung ist empfindlich, und bei Ihnen mit hoher Warscheinlichkeit bereits defekt. Alternativ lässt sich der Phillips Hue Dimmer Switch durch drücken aller 4 Taster für ca. 5 Sekunden zurücksetzen. Ein erfolgreicher Reset wird durch eine abwechselns Grün/Rot blinkende LED signalisiert. Dies stellt ebenfalls die Standardmethode für das Modell V2 dar, die keinen Reset-Knopf mehr besitzt.

5.2.5 Aufgabe 4 - Binding der Fernbedienung

5.2.6 Durchführung

Navigieren sie in der Weboberfläche zu der Übersicht Ihrer Lampe. Dort finden sie einen Reiter “binden“. Hier binden sie den Endpunkt X Ihrer Lampe mit dem Endpunkt X Ihrer Fernbedienung. 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 nachfolgenden Fragen in Ihrem Versuchsbericht.

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.

5.2.7 Aufgabe 5 - Gruppenbildung

Navigieren sie in der Weboberfläche zu dem Reiter “Groups “. Legen sie eine Gruppe mit dem Namen “Hue-Lights-<Gruppe> “ an. Editieren sie nun die Gruppe. Fügen sie die Endpunkte der beiden Lampen, die zum Steuern verwendet werden, der Gruppe hinzu.
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.

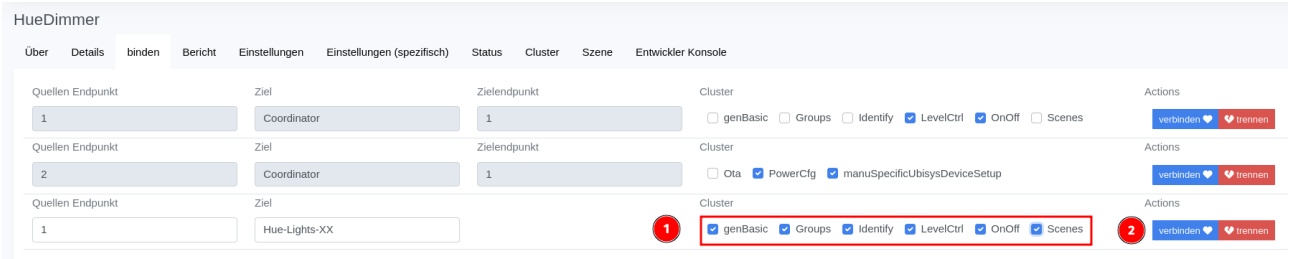


Abbildung 5.5: Zigbee Anlernen aktivieren - nur Lampe

Achten sie darauf, alle hier genannten Cluster anzuwählen.

Aufgabe

Speichern sie den Wireshark Capture ab als **“<Gruppe> - ZigbeeLab - Aufgabe 5“**.
Beantworten sie die nachfolgenden Fragen in Ihrem Versuchsbericht.

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?

5.2.8 Weiterführende Fragen

1. Welchem Zweck dienen die „Link Status" Nachrichten? Über welche Anzahl von „Hops" wird diese Nachricht übertragen? Analysieren Sie exemplarisch einige Link-StatusNachrichten und Interpretieren Sie diese!

2. Untersuchen Sie die „Link Quality Request"- bzw. „Link Quality Response"-Nachrichten. Gehen Sie auf den LQI-Wert in der „Link Quality Response" Nachricht und was bedeutet dieser?

3. Interpretieren Sie „Route Request" Nachrichten und zugehörige „Route-Response"- Nachrichten.

Kapitel 6

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)

6.0.1 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>
```

6.0.2 Zurücksetzen des Versuchs

6.0.3 Update der eingesetzten Software

Abbildungsverzeichnis

Literatur

[Ama12] Amazon. *Understand Smarthome Zigbee Support*. [Online; Stand 03. Oktober 2022]. 2012. URL: https://developer.amazon.com/en-US/docs/alexa/smarthome/zigbee-support.html.

[Ins12] Texas Instruments. *Z-Stack Monitor and Test API*. [Online; Stand 05. Oktober 2017]. 2012. URL: https://github.com/koenkk/zigbee-herdsman/raw/master/docs/Z-Stack%20Monitor%20and%20Test%20API.pdf.

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