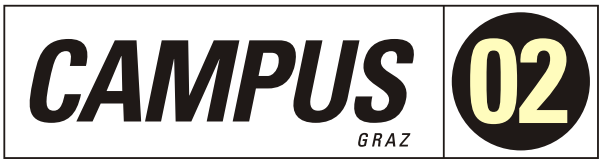
DIPLOM-ARBEIT

Internet of Things - Smart Home

ausgeführt am



Studiengang
Informationstechnologien und Wirtschaftsinformatik

Von: Adam Dolgos
Pers. Kennz. 1810320015

Graz, am 11. März 2020

.....
Adam Dolgos

Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....
Adam Dolgos

Kurzfassung

Das Ziel dieser Masterarbeit ist es, das Grundkonzept eines Smart Homes anhand eines konkreten Beispiels zu erstellen und das ganze System auf einem Raspberry Pi zu implementieren. Ein Smart Home kann basierend auf mehreren Kommunikationsprotokollen erstellt werden. Die Masterarbeit hat gezeigt, dass ZigBee, welches im Jahr 2002 entwickelt wurde, eines der geeignetsten Kommunikationsprotokolle für Smart Homes ist. Ein ZigBee-Netzwerk erfordert die Verwendung eines zentralen Koordinators zur Steuerung des gesamten Netzwerks. Eine der Einschränkungen eines ZigBee-Netzwerks besteht darin, dass jedes Gerät nur mit dem von dem Hersteller unterstützten Koordinator arbeiten kann. Mit dem CC2531 USB-Stick und der Software Zigbee2MQTT, die von Texas Instruments entwickelt wurde, kann ein eigener Koordinator erstellt werden, mit welchem jedes Gerät, welches von der Software unterstützt wird, verbunden werden kann. Die Verwaltung eines Smart Homes ist über Hausautomationsplattformen möglich. Das Projekt der Masterarbeit konzentriert sich auf die zwei Plattformen, Domoticz und Home Assistant, die von der Zigbee2MQTT-Software unterstützt werden. Im Rahmen dieser Arbeit werden der Installationsprozess beider Plattformen und der Betrieb des Systems am Beispiel von vier ausgewählten ZigBee-Geräten demonstriert.

Abstract

The aim of this master thesis is to create the basic concept of a Smart Home using a specific example and to implement the entire system on a Raspberry Pi. A Smart Home can be created based on several communication protocols. The paper showed that ZigBee, which was developed in 2002, is one of the most suitable communication protocols for Smart Homes. A ZigBee network requires the use of a central coordinator to control the entire network. One of the limitations of a ZigBee network is that each device can only work with the coordinator supported by the manufacturer. With the CC2531 USB stick, which was developed by Texas Instruments, and the software Zigbee2MQTT, you can create your own coordinator, with which any device that is supported by the software can be connected. The management of a Smart Home is possible via home automation platforms. The project of the master thesis focuses on the two platforms, Domoticz and Home Assistant, which are supported by the Zigbee2MQTT software. In this paper the installation process of both platforms and the operation of the system using four selected ZigBee devices are demonstrated.

Inhaltsverzeichnis

1 Internet of Things

In diesem Kapitel wird eine Einführung in die Welt der Smart Homes gegeben. Die am häufigsten verwendeten Verbindungs- und Kommunikationsprotokolle sollen dafür vorgestellt und verglichen werden. In vielen Fällen werden die Daten für längere Zeiträume zur Analyse gespeichert. Aus diesem Grund soll ein kurzer Überblick über die mögliche Datenverwaltung eines Smart Homes gegeben werden.

1.1 Einführung in die Internet of Things

Nach der Erfindung des Internets waren nur wenige Computer mit dem Internet verbunden. Es bot die Möglichkeit, E-Mails zu senden und Dateien über FTP hochzuladen oder herunterzuladen. Eine wichtige Weiterentwicklung des Internets war der Start des World Wide Web (WWW) mit immer mehr Webseiten und Diensten. Dann ist die Ära des mobilen Internets gekommen, in der Mobiltelefone und andere tragbare Geräte auf das Internet zugreifen konnten. Als Nächstes wurden soziale Netzwerke eingerichtet, worüber die Menschen auch mit dem Internet verbunden werden konnten. Der nächste Schritt in der Entwicklung des Internets besteht darin, dass neben Servern, Clients, mobilen Geräten und Personen auch für Dinge (Things) eine Verbindung zum Internet hergestellt werden kann. Dies nennt man Internet der Dinge (Internet of Things, IoT) (Perera, Zaslavsky, Christen & Georgakopoulos, 2013).

Über das Internet der Dinge hat Kevin Ashton, Executive Direktor des Auto ID Center, erstmals 1999 gesprochen. In seiner Rede hat er erwähnt, dass seiner Meinung nach IoT die Welt genauso verändern kann, wie das Internet diese damals verändert hat – oder sogar noch besser (Ashton, 2009). Nach Angaben der Cisco Internet Business Solutions Group (IBSG) gab es ab dem Jahr 2008 mehr Dinge, welche mit dem Internet verbunden waren, als Menschen. Dies ist das offizielle Erscheinungsjahr des Internet of Things (Postcapes, 2019a).

Es gibt mehrere Definitionen von Internet of Things:

- IoT ist ein weltweites Netzwerk von eindeutig adressierbaren, miteinander verbundenen Objekten, basierend auf Standard-Kommunikationsprotokollen (Enterprise, Micro & systems in: Co-operation with the Working Group RFID of the ETP EPOSS, 2008).

- Das Internet der Dinge ermöglicht Menschen und Dingen jederzeit, überall, mit allen und mit allem, idealerweise mit jedem Netzwerk und jedem Dienst verbunden zu sein (Vermesan et al., 2009).

- Objekte haben Identität und virtuelle Persönlichkeit in einem intelligenten Raum und verwenden intelligente Schnittstellen, um auf die Umgebung und die Benutzer um sie herum zuzugreifen und mit ihnen zu kommunizieren (Khan, Khan, Zaheer & Khan, 2012).

Grundsätzlich kann man sagen, dass IoT eigentlich nichts anderes ist, als eine Verbindung zwischen der physischen und der digitalen Welt. Die dynamische und globale Netzwerkinfrastruktur des Internets der Dinge, die bereits existiert oder sich entwickelt, ist ein wesentlicher Bestandteil des Internets der Zukunft. IoT verfügt auch über selbstkonfigurierende Funktionen, die auf standardmäßigen und interoperablen Kommunikationsprotokollen basieren. Die Dinge, die physisch oder virtuell sein können, haben eine eigene Identität, physische Attribute und eine virtuelle Persönlichkeit und verwenden intelligente Schnittstellen. Bei dem Internet of Things ist es wichtig, jedes Gerät oder Objekt eindeutig zu identifizieren, um eine ordnungsgemäße und fehlerfreie Kommunikation zu gewährleisten. In diesem Bereich unterscheidet man drei verschiedene Sichten (Atzori, Iera & Morabito, 2010, pp. 2790-2791):

1. Objektorientierte Sicht

2. Internet-orientierte Sicht

3. Semantisch orientierte Sicht

Die objektbasierte Sicht des IoT ist der Elektronische Produktcode (Electronic Product Code, EPC), welcher mit Hilfe der Radio Frequency Identification (RFID) eine Basistechnologie für IoT bildet. Die Technologie wurde speziell zur Identifizierung von Objekten entwickelt und besteht grundsätzlich aus zwei Hauptteilen: Erstens dem RFID Tag und weiters aus dem RFID Reader. Das RFID-Tag ist ein kleines Objekt, welches an das zu identifizierende Objekt angehängt oder darin eingebaut werden kann. Der RFID Reader kommuniziert über Radiowellen mit den RFID Tags, und dadurch ist es möglich, Tags global, automatisch und in Echtzeit, zu identifizieren, zu verfolgen und zu überwachen (Jia, Feng, Fan & Lei, 2012).

Web of Things ist eines der besten Beispiele für einen Internet-orientierten Ansatz. Demnach sollen eingebettete Computer in den Dingen (Things) platziert und mithilfe vorhandener Technologien mit dem Internet verbunden werden. Dazu wird es immer notwendiger, von dem Internet Protokoll Version 4 (IPv4) auf Version 6 (IPv6) umzustellen, da zu viele Dinge identifiziert werden müssen. Im Vergleich von IPv4 mit 4,3 Milliarden IP-Adressen, bietet IPv6 2^{128} IP-Adressen, die zwischen den einzelnen Geräten aufgeteilt werden können (Atzori et al., 2010, p. 2800).

Eine dritte Sicht, welche das IoT aus einer anderen Perspektive betrachtet, ist das seman
tisch-orientierte IoT, welches nur in der Literatur zur Verfügung steht. Diese Sicht beschäftigt sich mit dem Umgang und der weiteren Verarbeitung der in der IoT gespeicherten riesigen Datenmenge, wie zum Beispiel der Speicherung, Organisierung, dem Suchen und der Darstellung von Daten (Atzori et al., 2010, pp. 2790-2791).

Ein weiteres wichtiges Merkmal ist, dass die Dinge über Standardprotokolle einfach in das Informationsnetzwerk integriert werden können. Das Internet der Dinge kann durch intelligente Geräte ein aktiver Bestandteil unserer Umwelt sein. Sie können ständig das Umfeld und seine Änderung um sich herum erkennen und dann über das Internet Daten bzw. Informationen austauschen. Basierend auf den erhaltenen Informationen können sie schnell und kontinuierlich auf die Veränderungen in der Umgebung reagieren. Durch die erworbenen Informationen können sie neue Aufgaben erstellen, noch nicht laufende Prozeduren ausführen oder die Funktion anderer Geräte beeinflussen, um basierend auf den gesammelten Informationen die vorteilhaftesten und sichersten Aufgaben erfüllen zu können (Vermesan et al., 2009).

1.2 M2M vs. IoT

Die Idee, verschiedene Geräte miteinander zu verbinden, beschäftigt die Menschen seit der zweiten Hälfte des 20. Jahrhundert. Der erste große Durchbruch in diesem Bereich war die in den 1970er Jahren erfundene Maschine-to-Maschine-Kommunika
tionstechnologie (M2M), die im Wesentlichen eine Basis für die heutige moderne IoT-Kommunikation darstellt. Den beiden Technologien ist gemeinsam, dass sie entsprechende Geräte verbinden, die sensorbezogene Daten sammeln und untereinander übertragen. M2M-Geräte sind üblicherweise über eine herkömmliche SIM-Karte oder ein Kabel mit einem Kommunikationsnetzwerk verbunden. Auf der anderen Seite basiert IoT auf der Cloud-Architektur. Die IoT-Geräte können sowohl drahtgebunden als auch drahtlos eine Verbindung zum Internet herstellen. In den meisten Fällen passiert diese Verbindung drahtlos über verschiedene Kommunikationstechnologien, wie zum Beispiel über WLAN, 6LoWPAN oder auch über Mobilfunk (Cellular Network), abhängig vom Abdeckungsbereich. Eine Internetverbindung ist für den Betrieb der IoT-Systeme unerlässlich. Die Geräte bzw. Applikationen des IoT müssen sich nicht alle mit dem gleichen Internetprotokoll verbinden. Hingegen sind M2M-Netzwerke häufig geschlossen, und die Geräte können nur über das selbe Kommunikationsprotokoll kommunizieren. Aus diesem Grund bieten die IoT Plattformen erweiterte Integrationsmöglichkeiten und sind dadurch besser skalierbar, als M2M (AVSystem, 2019), (Postcapes, 2019b).

Im Gegensatz zu M2M-Systemen basieren IoT-Systeme auf Open Source-Standards, die einen Informationsaustausch über verschiedene Plattformen und Geräte ermöglichen. Ihr Hauptzweck ist die Verwendung, Analyse und Übermittlung der erhobenen

Daten. Im Gegensatz zu der bei einem M2M-System verwendeten unidirektionalen Kommunikation kann die Kommunikation bei diesen Systemen bidirektional sein. Dadurch ist es für die Geräte in einem IoT-System möglich, Daten nicht nur an andere Geräte bzw. an die Zentraleinheit des Systems zu schicken, sondern auch von anderen Geräten bzw. von der Zentraleinheit Daten zu empfangen. Durch diese Eigenschaft werden für die IoT Lösungen durch die gesammelten Daten automatisierte Reaktionen oder sogar vorausschauende Prozesse auf der Grundlage von Analysen möglich (AVSystem, 2019).

Grundsätzlich kann man IoT als eine Verbesserung und Erweiterung der M2M Kommunikationstechnologie verstehen. Durch die M2M Technologie ist es möglich, beispielsweise durch ein mit einem Sensor ausgestattetes Gerät Daten zu sammeln, und diese mit einer externen Applikation zu verteilen und dadurch eine Fernüberwachung und -verwaltung durchzuführen. Wenn man jedoch dieses Gerät mit einem anderen Gerät integriert und das erste Gerät Anweisungen an das zweite Gerät sendet bzw. bei einem Problem oder Extremfall den Benutzer/die Benutzerin warnt und die geeigneten Vorsichtsmaßnahmen ergreift, dann ist das eindeutig nur durch Verwendung des IoT möglich (vgl. (AVSystem, 2019)).

Die Tatsache, dass M2M häufig als geschlossenes Netzwerk arbeitet, führt zu einer geringeren Wichtigkeit der Cybersicherheit, da Hacker physisch auf eine der Komponenten zugreifen müssen, um das gesamte System zu infizieren. Im Fall von IoT können Geräte jedoch jederzeit über das Internet angegriffen werden. Daher ist es äußerst wichtig, die entsprechenden Sicherheitsstandards einzuhalten (Abomhara & Køien, 2015).

1.3 IoT Architecture

Im Zentrum des Internet of Things liegt die Kommunikation. Um mit Milliarden von Geräten eine Verbindung zum Internet herzustellen und über das Internet kommunizieren zu können, stehen verschiedene Protokolle zur Verfügung. In der heutigen Literatur unterscheidet man zwischen zwei verschiedenen Architekturen. Die allgemein akzeptierte Architektur des Internet of Things ist die dreischichtige Architektur, die aus dem Application Layer (Applikationsschicht), dem Network Layer (Netzwerkschicht) und dem Perception Layer (Erkenntnis- oder Sensorenschicht) besteht, wie in der Abbildung (Abb. 1.1) dargestellt ist (Wu, Lu, Ling, Sun & Du, 2010).

Die Hauptaufgabe der untersten Schicht der Iot-Architektur besteht darin, Daten in ihrer Umgebung wahrzunehmen und zu erfassen, wie zum Beispiel verschiedene Sensormessungen, WSN- und GPS-Daten. Sie hat auch die Aufgabe, die gesammelten Informationen in ein digitales Signal umzuwandeln, um die Übertragung über das Internet zu ermöglichen (Abdmeziem, Tandjaoui & Romdhani, 2015).

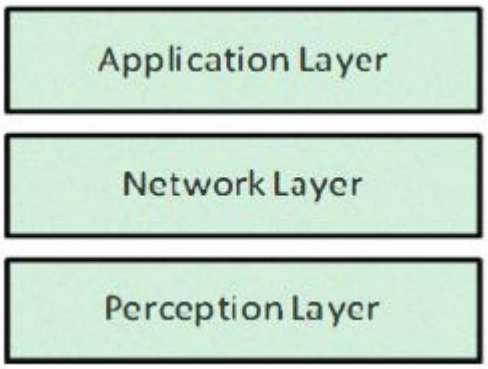


Abbildung 1.1: 3-Layer Architektur des Internet of Things.

Die zweite Schicht, der Network Layer, ist hauptsächtlich für die Verbindung zwischen Dingen, Netzwerkgeräten und Servern verantwortlich. Seine Funktion ist es, im Percetion Layer gesammelte Sensordaten zwischen dem Perception und dem Application Layer zu übertragen. Laut vieler Studien ist dieser Layer der am weitesten entwickelte Layer in der ganzen IoT Architektur. Die erfolgreiche und sichere Verbindung und Integration von Objekten in das IoT-System wird durch die einzigartige Adressierungs- und Routingfähigkeit gewährleistet. Die Übertragung kann sowohl mittels kabelgebundenen als auch mittels drahtlosen Technologien erfolgen. Zu den meistgenutzten Übertragungsmethoden bei IoT Projekten zählen ZigBee, WLAN, Bluetooth, Z-Wave, Ethernet usw. (Abdmeziem et al., 2015). Die oberste Schicht der IoT-Architektur verbindet den Benutzer/die Benutzerin mit der Welt des Internets der Dinge. Gemäß den Bedürfnissen des Benutzers/der Benutzerin bietet der Application Layer Dienste, die durch verschiedene Protokolle, wie zum Beispiel HTTP, MQTT, FTP oder SMTP verfügbar sind. In dieser Schicht befindet sich auch die Cloud, bei der es sich um einen Remote-Server handelt, der verwendet wird, um Daten in einer sicheren und geschützten Umgebung zu speichern und zu analysieren (Bilal, 2017), (Al-Fuqaha, Guizani, Mohammadi, Aledhari & Ayyash, 2015).

Ein moderner, fortgeschrittener Ansatz für die Architektur des Internets der Dinge ist die sogenannte 5-Layer-Architektur, eine weiterentwickelte Version der 3-Layer-Architektur (siehe Abb. 1.2). Grundsätzlich ähneln die beiden unteren Ebenen ihren Vorgängern. Die unterste Objektschicht entspricht im Wesentlichen dem Perception Layer. Ihre Aufgabe ist es, Daten zu sammeln, zu verarbeiten, zu digitalisieren und zu höheren Ebenen zu übertragen. Die Object Abstraction ist für die Übertragung der in der Objektschicht aufgezeichneten Daten von Sensoren verantwortlich. Die Anwendungsschicht der 3-Schicht-Architektur ist in der 5-Schicht-Architektur in 3 weitere Teile unterteilt. Das Service Management (auch als Middleware-Layer bezeichnet) spielt eine Rolle beim Speichern, Analysieren und Verarbeiten vieler Daten. Die Aufgabe des Application Layers ist hier die gleiche: Anwendungsspezifische Dienste für den Benutzer/die Benutzerin bereitzustellen. Der oberste Layer, der Business Layer, verwaltet alle Services und Aktivitäten innerhalb des IoT-Systems. Er kann Geschäfts-

modelle, Diagramme und Anwendungen auf der Grundlage der vom Application Layer bereitgestellten Daten generieren (Silva, Khan & Han, 2017).

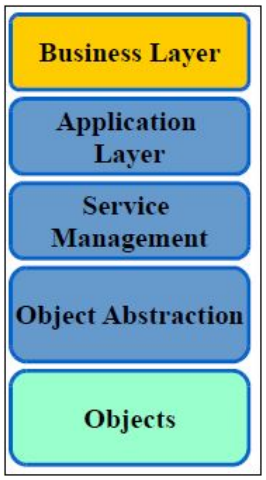


Abbildung 1.2: 5-Layer Architektur des Internet of Things.

1.4 Verbindungsprotokolle

Im Zentrum des Internet of Things liegt die Kommunikation. Um mit Milliarden von Geräten eine Verbindung zum Internet herzustellen und über das Internet kommunizieren zu können, stehen verschiedene Protokolle zur Verfügung. Eine Online-Umfrage wird jährlich unter der Führung der Eclipse IoT Working Group (Eclipse Foundation Inc., 2020) durchgeführt, um die Anforderungen, Prioritäten und Wahrnehmungen der IoT-Entwicklergemeinde besser zu verstehen. Im Jahr 2019 nahmen 1717 Personen an der Online-Umfrage teil (Eclipse Foundation Inc., 2019). Laut dieser Umfrage sind die meist verwendeten Verbindungsprotokolle TCP/IP, WLAN und Ethernet. Im nächsten Teil soll die Gruppe von Kommunikationsprotokollen TCP/IP nähergebracht werden.

TCP/IP ist eine Sammlung von Kommunikationsprotokollen in verschiedenen Schichten. Seine zwei bestimmenden Elemente sind, wovon es seinen Namen bekommen hat, das Transmission Control Protocol (TCP) und das Internet Protocol (IP). Die Grundoperation ist wie folgt: die Anwendung bereitet die zu übertragenden Daten im erforderlichen Format vor, fügt dann die Adresse des Zielcomputers und die darauf ausgeführte Hostanwendungs-ID hinzu und leitet sie an weitere Ebenen weiter. Hier werden die zu sendenden Informationen in Pakete gepackt, wobei jedes Paket eine Paketnummer sowie die Information über Adressierung und Bestätigungsanforderung enthält. Den Paketen muss dann eine geeignete Route zugewiesen werden. Schließlich werden Pakete auf einem physischen Medium an den Empfänger gesendet (Horvath, 2017, p. 3). Die Struktur des TCP/IP Modells ähnelt dem OSI-Modell,

ist jedoch mit nur vier statt sieben Schichten einfacher. Wie in Abbildung 1.3 dargestellt, besteht das TCP/IP-Modell aus vier Schichten:

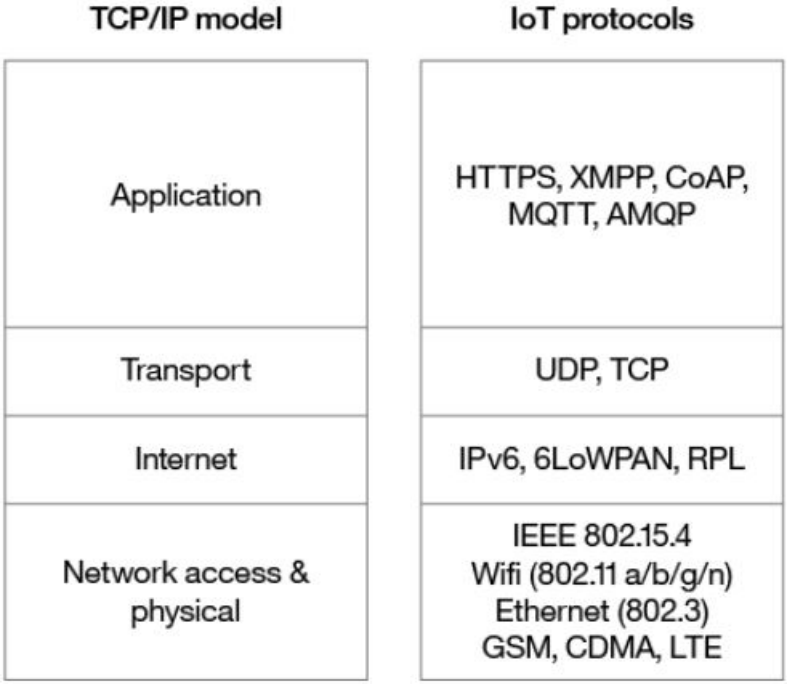


Abbildung 1.3: TCP/IP-Modell mit den zugeordneteten IoT-Netzwerkprotokollen.

Die unterste Schicht des TCP/IP Modells ist die Netzwerkschicht. Diese Schicht enthält alle physischen und logischen Komponenten, welche die physische Verbindung zwischen den einzelnen Knoten sicherstellen. Im TCP/IP Referenzmodell ist es nicht definiert, wie eine Verbindung hergestellt werden soll, sondern es ist nur festgestellt, dass der Host eine Verbindung zu einem Netzwerk herstellen muss, welches über ein Protokoll zur Übertragung von IP-Paketen verfügt. Dies beinhaltet den Standard IEEE 802.15.4, welcher die Basis für verschiedene in IoT Projekten genutzte Technologien ist, wie zum Beispiel ZigBee, das Ethernet oder WLAN (Silva et al., 2017).

An der zweiten Stelle steht die Internet- oder Netzwerkschicht (Network Layer). Die Aufgabe der Internetschicht besteht darin, von den oberen Schichten empfangene Pakete basierend auf der IP-Adresse auf einer Route ihrer Wahl zum Ziel weiterzuleiten. Die Reihenfolge der Pakete wird durch die oberen Schichten bestimmt. Eine weitere wichtige Aufgabe besteht darin, eine sogenannte Paketüberlastung zu verhindern. Die Internetschicht definiert ein offizielles Paketformat sowie ein Protokoll, welches als Internet Protocol (IP) bezeichnet wird, das über verschiedene Versionen und Formen verfügt, wie zum Beispiel die schon vorher erwähnten IPv4 und IPv6 Protokolle oder das von IoT zunehmend bevorzugte IPv6 over Low power Wireless Personal Area Network (6LoWPAN) Protokoll (Silva et al., 2017).

Über der Netzwerkschicht befindet sich die Transportschicht, die die sogenannte End-to-End-Kommunikation implementiert. Ihre Aufgabe ist es, eine Verbindung zwischen der Quelle (Sender) und dem Ziel (Empfänger) herzustellen und die Datenübertragung zwischen den beiden Punkten zu implementieren. Es wird zwischen zwei Hauptprotokollen unterschieden: Eines ist das Transmission Control Protocol (TCP), bei dem es sich um ein zuverlässiges, verbindungsorientiertes, paketsicheres Transportprotokoll handelt [IaaS mit OpenStack: Cloud Computing in der Praxis]. TCP teilt den eingehenden Bytestrom in kleinere Pakete auf und leitet diese nacheinander in der entsprechenden Reihenfolge an die Internetschicht weiter. Der TCP-Prozess am Ende sammelt eingehende Pakete und überträgt sie als einen einzigen Ausgabestream. TCP führt weiters auch eine Verkehrssteuerung durch, sodass ein schneller Sender nur so viele Nachrichten an einen langsameren Empfänger sendet, wie dieser auch empfangen kann (Silva et al., 2017). Um die oben genannten Merkmale einhalten zu können, baut TCP eine Verbindung mittels Drei-Wege-Handschlag (Three-Way-Handshake) auf, bevor die Datenübertragung zwischen Anwendungen gestartet wird.

Wie in der folgenden Abbildung (Abb. 1.4) dargestellt, initiiert der Client die Verbindung. Er sendet ein TCP-Paket, wobei das SYN-Bit (Synchronized) im Header auf „1“ gesetzt wird, somit gibt es an, dass der Client eine Verbindung zum Server herstellen möchte. Der Server empfängt die Nachricht des Clients und sendet eine Bestätigungsnachricht mit den Empfangsbestätigungen, dadurch wird das ACK-Bit (Acknowledgement) im Header-Bereich auf „1“ gesetzt. Zusätzlich zu ACK wird der SYN-Wert an den Client zurückgegeben. Der Client empfängt eine Antwort vom Server und sendet ein ACK an den Server. Nachdem die Verbindung hergestellt wurde, kann die Datenübertragung beginnen (Mašetić, Kečo, Dogru & Hajdarevic, 2017, p. 754).

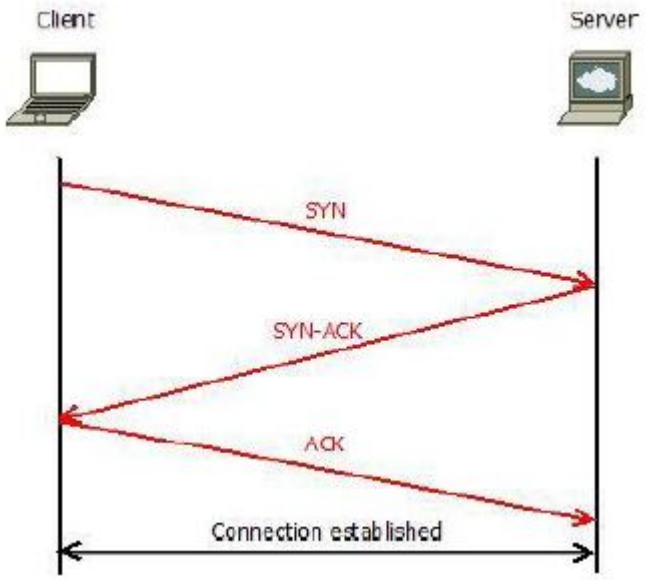


Abbildung 1.4: TCP Three-Way-Handshake.

Das andere Protokoll in dieser Schicht ist das verbindungslose UDP (User Diagram Protocol), ein Netzwerkprotokoll. Im Gegensatz zu TCP spielen hier die Reihenfolge der Pakete und die Verkehrssteuerung keine Rolle. Dieses Protokoll hat sich in Bereichen verbreitet, in denen die Paketankunft wichtiger ist als die Paketzuverlässigkeit, wie zum Beispiel bei Videostreaming (Badach & Hoffmann, 2019, p. 39). Die Anwendungsschicht bildet die oberste Schicht des TCP/IP Modells und enthält eine Reihe von vom Benutzer angeforderten Protokollen höherer Ordnung wie Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP) oder Message Queuing Telemetry Transport (MQTT). Die Aufgabe dieser Schicht besteht darin, dem Benutzer/-der Benutzerin und den Benutzerprogrammen eine Schnittstelle zur Versorgung von Netzwerkdiensten bereitzustellen (Badach & Hoffmann, 2019, pp. 42-43).

Auf TCP- oder UDP-Anwendungen kann über allgemein bekannte, eindeutig definierte, sogenannte Well-Known-Ports zugegriffen werden, die von der Internet Assigned Numbers Authority (IANA) standardisiert wurden. Eine TCP- oder UDP-Anwendung kann über sogenannte Sockets identifiziert werden. Ein Socket besteht aus einer IP-Adresse und dem Anwendungsport. Die IP-Adresse identifiziert den Rechner und der Port identifiziert die auf dem Computer ausgeführte Anwendung. Auf diese Weise kann ein Anwendungsprogramm problemlos auf TCP/IP-Protokolle zugreifen (Badach & Hoffmann, 2019, pp. 35-36).

1.5 Kommunikationsprotokolle

Im Bereich der Kommunikationsprotokolle von IoT Developer Survey (IoT Developer Survey, (Eclipse Foundation Inc., 2019)) wurden drei Kommunikationsprotokolle hervorgehoben, die bei IoT-Projekten am meisten verwendet werden. Diese sind http (üblicherweise für RESTful Web Services), MQTT und Websockets. Sowohl HTTP- als auch MQTT-Kommunikationsprotokolle werden in nahezu gleichem Umfang verwendet, und zwar in über 40% aller Projekte. Im nächsten Abschnitt wird eine kurze Einleitung in HTTP-basierte RESTful Web Services und MQTT Protokolle gegeben und der Unterschied zwischen beiden dargestellt.

1.5.1 HTTP

HyperText Transfer Protocol (HTTP) ist ein Informationsübertragungsprotokoll für verteilte, kollaborative Hypermedia-Informationssysteme. Der Well-Known-Port von HTTP ist 80. HTTP basiert auf dem Anforderungs-Antwort-Verfahren (Request-Response-Modell) zwischen Clients und Servern auf der obersten Ebene des TCP/IP-Modells. Es verwendet nur das zuverlässige TCP-Protokoll, da kein Datenverlust zulässig ist. Der ursprüngliche Zweck des Protokolls bestand darin, HTML-Seiten zu

veröffentlichen und zu empfangen. Die sichere Version von HTTP ist HTTPS mit dem Well-Known-Port 443 (Fielding & Gettys, 1999).

RESTful Web Service ist ein auf der REST-Architektur basierender Webdienst, der auf Basis des HTTP-Protokolls entwickelt wurde und für die Kommunikation und Datenübertragung zwischen Client und Server verwendet werden kann. Anfragen (Requests) werden unter Verwendung von URIs gestellt, die Ressourcen identifizieren und eine einheitliche Schnittstelle für den Client bereitstellen. Der Server antwortet auf alle Anfragen im selben Format, normalerweise JSON, er kann jedoch auch HTML und XML verwenden. Es ist wichtig, dass Anforderungen zustandslos sind, daher erfordert die Autorisierung von der Serverseite eine ständige Aufmerksamkeit. Dies bedeutet, dass der Server den Client in jeder Anforderung identifizieren muss, um zu wissen, über welche Rechte er verfügt und auf welche Ressourcen er zugreifen darf (Levin, 2016).

Die Authentifizierung erfolgt im HTTP-Anforderungsheader (HTTP-Request-Header). Es gibt verschiedene Möglichkeiten, dies zu tun. Die einfachste Variante ist die Verwendung von der sogenannten HTTP Basisauthentifizierung. In diesem Fall werden Benutzername und Passwort durch einen Doppelpunkt getrennt und mittels BASE64 codiert (Gupta, 2019).

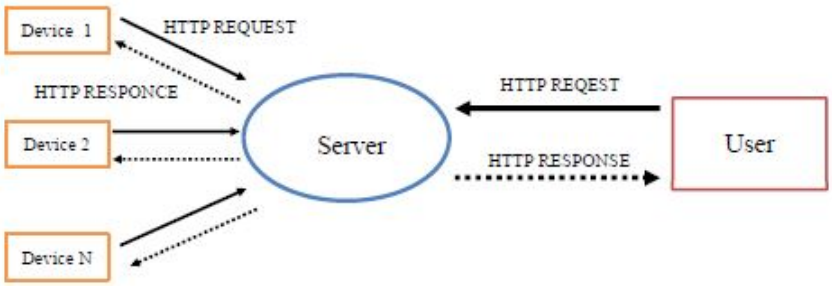


Abbildung 1.5: Systemkonfiguration über HTTP.

Die Abbildung 1.5 zeigt ein Beispiel für die HTTP-Kommunikation im Bereich des Internets der Dinge. Geräte (Device) und Benutzer (User) können eine Verbindung zum Server herstellen, indem sie die von Sensoren gemessenen Daten senden oder von anderen Klienten gesendete Daten in einer Anfrage (HTTP Request) anfordern können. Der Server verwaltet die Anforderung, bereitet die Antwort (HTTP Response) vor und schickt diese zum Klienten zurück (Dixon, 2014).

Die Kommunikation wird immer vom Client angeregt, indem eine Anforderung über eine Methode an den Server gesendet wird. Die Methoden bestimmen die für die angegebene Ressource auszuführende Operation. RESTful Web Service verwendet die folgenden fünf Methoden der acht HTTP-Protokollmethoden (Gupta, 2019):

- HTTP GET: Ruft Daten ab. Dies ist eine der sogenannten sicheren Methoden, da sie die Daten nicht verändert.

- HTTP POST: Sendet neue Daten an den Server.
- HTTP PUT: Dient zum Ändern von Daten.
- HTTP DELETE: Löscht bestehende Daten.
- HTTP PATCH: Teilaktualisierung von Daten.

Sowohl der HTTP Request als auch der HTTP Response bestehen aus drei Teilen: HTTP-Methode, Header und Body. Im unteren Aufruf wird ein POST Request an den Server geschickt (Dixon, 2014).

```
1 POST /index/sample HTTP/1.1
2 Host: www.example.com
3 Authorisation: Basic wfjdsaXCfdsdf321
4 Content-Type: application/json
5 Accept: application/json, application/xml
6 Cache-Control: no-cache
7 {
8   color:"black",
9   value:"#000"
10 }
```

Die erste Zeile enthält die Request-Method (POST), die relative Adresse des angeforderten Dokuments (`/index/sample`) und die HTTP-Protokollversion (HTTP1.1). In den Zeilen zwei bis sechs sind die Elemente des Headers definiert. Host gibt den Hostnamen des Servers an. Im HTTP Header „Authorisation“ wird der Typ der Autorisierung definiert und die entsprechenden Werte mitgegeben. In diesem Fall handelt es sich um HTTP Basic Authorisation und Benutzername und Passwort sind in BASE64 codiert. HTTP unterstützt den MIME-Standard (Multipurpose Internet Mail Extensions), und somit können die zwei folgenden Header-Felder angegeben werden. Ein Dokumenttyp des MIME-Standards besteht aus einem Typ und einem Untertyp. Der im HTTP Header „Content-Type“ definierte Dokumenttyp „application/json“ bedeutet, dass der HTTP-Body im JSON-Format übergeben wird. In der nächsten Zeile (HTTP Header „Accept“) ist spezifiziert, in welchem Format die Antwort geliefert werden kann. Bei diesem Feld ist es möglich, mehrere Inhaltstypen mit Komma getrennt anzugeben. In diesem Fall sind beispielsweise die Formate JSON und XML akzeptiert. Das letzte Feld der HTTP Header ist die Cache-Kontrolle. Cache-Control legt fest, ob der Weg zwischen Client und Server zwischengespeichert werden soll oder nicht (Gupta, 2019). Die Zeilen sieben bis zehn bilden den Body des HTTP-Requests. Hier werden die Daten in dem durch den HTTP Header „Content-Type“ angegebenen Format an den Server geschickt. In dem Beispiel werden die Werte im JSON-Format mittels Schlüsselwertpaaren (Key-Value Pairs) an den Server geschickt.

Der Server antwortet auf die Frage mit einem sogenannten HTTP-Response-Code. Im Allgemeinen bedeuten 2xx-Codes, dass die Anfrage in Ordnung ist und die Methode richtig durchgeführt wurde. 3xx, 4xx und 5xx Antwortcodes zeigen immer einen Fehler an. Unter den bekanntesten Codes sind: 200 - OK, 201 - Erstellt, 401 - Nicht autorisiert, 404 - Nicht gefunden und 500 – Interner Server Fehler. Natürlich hängt der HTTP-Antwortcode auch von der Art der HTTP-Anforderungsmethode ab. Nach erfolgreicher Ausführung des oben gezeigten Beispiels wird der Antwortcode 201 - Created zurückgeliefert. Bei anderen Methoden, zum Beispiel bei der GET-Methode, werden die angeforderten Daten im Response Body geliefert. Das Format des Response Bodies hängt vom Typ ab, der im Request-Header definiert ist (Accept: applicationjson, applicationxml).

1.5.2 MQTT

Ein weiteres häufig verwendetes Kommunikationsprotokoll ist ein datenorientiertes (data-centric), leichtes (lightweight) Messaging-Protokoll, nämlich Message Queue Telemetry Transport (MQTT). MQTT ist ein OASIS-Standard und befindet sich auf der obersten Anwendungsebene des TCP/IP-Modells. Für MQTT sind die Well-Known-Ports 1883 und 8883 reserviert. Die Verbindungen sind mit SSL/TLS gesichert. MQTT wurde speziell für die Kommunikation zwischen Geräten mit begrenzten Ressourcen und geringer Bandbreite (M2M- und IoT-Geräte) entwickelt. Für seine Ausführung sind kleine Arbeitsspeicher und leistungsschwache Prozessoren ausreichend, somit ist es energiesparend. Seine Aufgabe besteht darin, eingebettete Geräte und Netzwerke mit Verbindungen zu Anwendungen und Middleware zu versorgen. Das Funktionsprinzip des MQTT-Protokolls basiert auf der Publish-Subscribe-Infrastruktur (Pub/ Sub-Infrastruktur). Der Verbindungstyp kann one-to-one, one-to-many oder many-to-many sein (Al-Fuqaha et al., 2015), (Lampkin et al., 2012).

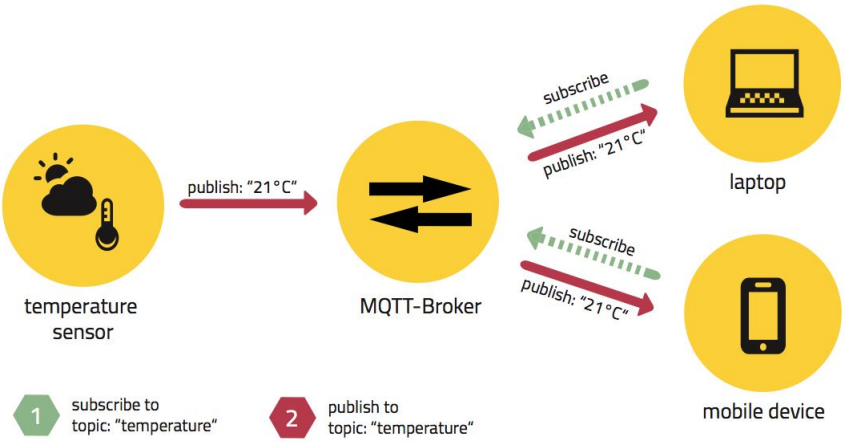


Abbildung 1.6: Beispiel für den Betrieb eines MQTT Protokolls. Das Bild zeigt, wie die Kommunikation zwischen den Clients mittels Broker durchgeführt wird.

Die Abbildung 1.6 zeigt ein einfaches Beispiel für die Kommunikation zwischen MQTT-Clients. In diesem Beispiel erfasst ein Sensor die Bürotemperatur und leitet die gemessenen Daten via Topic „temperature“ an den MQTT-Broker weiter. In diesem Fall haben zwei Geräte, ein Laptop und ein Mobiltelefon, beim MQTT-Broker das Topic „temperature“ abonniert, sodass sie die Nachricht zugestellt bekommen. Im MQTT Betrieb wird zwischen zwei Teilnehmern unterschieden: dem Klient (Client) und dem sogenannten Broker. Der Client kann die Ausführung des MQTT-Protokolls initiieren, indem er eine CONNECT-Nachricht an den Broker sendet. Der Broker antwortet darauf mit einer CONNACK-Nachricht (Connection Acknowledge). Nach der Verbindung können die einzelnen Clients ein oder mehrere sogenannte Topics abonnieren (subscribe), wo sie von anderen Clients veröffentlichte Daten erhalten und/oder sie können selbst Daten veröffentlichen (publish). Der Broker fungiert als Vermittler zwischen den Klienten. Er verwaltet die Weiterleitung von Nachrichten, empfängt die Nachrichten von sendenden Clients und leitet sie an die auf dem Topic abonnierten Clients weiter. Abonnement- und Abbestellungsanfragen bzw. der Verbindungsaufbau von Clients werden auch vom Broker verwaltet. Die Verbindung des Client wird mit der DISCONNECT Nachricht beendet (Al-Fuqaha et al., 2015), (Lampkin et al., 2012).

Da die ordnungsgemäße Funktion von Client und Kommunikationskanal nicht garantiert werden kann, wurde im MQTT-Protokoll die sogenannte Last Will and Testament-Funktion implementiert. Auf diese Weise können die verbundenen Clients im Falle einer unerwarteten Trennung benachrichtigt werden. Die Last Will-Nachricht kann in der CONNECT-Nachricht definiert werden (Götz, Obermaier, Pfefferle, Skerrett & Raschbichler, 2015a).

Das Topic dient zur Klassifizierung der Nachrichten und definiert den Inhalt der Nachricht. Die Struktur ist hierarchisch organisiert. Unterstrukturen können mit dem Zeichen “/” erstellt werden, zum Beispiel `home/office/temperature`. Sie unterstützen auch die Verwendung von Platzhaltern (Wildcards), sodass ein Client jedes Subtopic eines höheren Topics, beispielsweise `home/office/#` problemlos abonnieren kann. In diesem Fall kann der Client jedes Topic innerhalb des Büros abonnieren. Ein Client kann jedoch nur Nachrichten zu einem Thema gleichzeitig veröffentlichen, sodass in diesem Fall keine Platzhalterzeichen verwendet werden können (Banks, Briggs, Borgendale & Gupta, 2019).

Abbildung 1.7 beschreibt das MQTT-Nachrichtenformat. Die ersten beiden blau markierten Bytes sind immer der MQTT-Header. Die ersten vier Bits geben den Nachrichtentyp (Message Type), als vorzeichenloser Wert dargestellt, an, zum Beispiel CONNECT mit dem Wert Eins, PUBLISH mit dem Wert Drei, SUBSCRIBE mit dem Wert Acht usw. Das DUP-Flag gibt an, ob die Nachricht ein Duplikat ist, weil der Empfänger (Client oder Broker) die ursprüngliche Nachricht nicht bestätigt hat. Wenn es sich um ein Duplikat handelt und der Nachrichtentyp PUBLISH ist, wird der Wert Eins, ansonsten immer der Wert Null übergeben (Banks et al., 2019, p. 22).

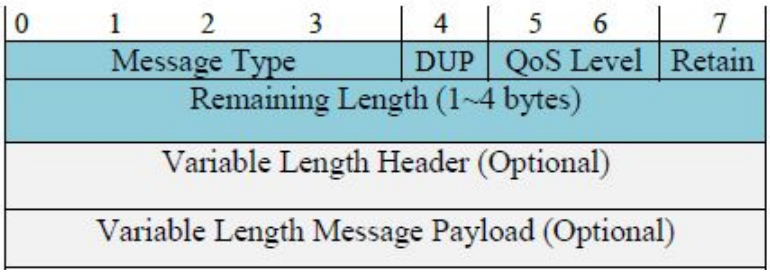


Abbildung 1.7: MQTT Nachrichtenformat.

MQTT definiert drei Servicequalitätsstufen (Quality of Service, QoS) (Götz, Obermaier, Pfefferle, Skerrett & Raschbichler, 2015b):

1. QoS 0: At most once (Höchstens einmal) - Nachrichten können verloren gehen bzw. Ankunft am Zielort ist nicht garantiert

2. QoS 1: At least once (Mindestens einmal) - Garantiert, dass Nachrichten mindestens einmal an das Ziel gesendet werden

3. QoS 2: Exactly once (Genau einmal) - Es ist wichtig, dass die Anwendung die Nachricht genau einmal empfängt

QoS 0 basiert im Wesentlichen auf dem Fire&Forget-Prinzip (schießen und es ist egal, ob Sie etwas treffen). Die Wahrscheinlichkeit eines Treffers hängt von der Transportsicherheit des zugrunde liegenden TCP-Protokolls ab. Wenn ein empfangender Client offline ist, werden die QoS 1- und 2-Nachrichten in einer Warteschlange gespeichert, von der aus der wieder verfügbare Client Daten empfangen kann. QoS 2 garantiert, dass jede Nachricht empfangen wird, und ist somit der sicherste, aber auch der langsamste Servicelevel (Götz et al., 2015b).

Im letzten Bit in diesem Byte, “Retain”, kann bei dem PUBLISH Nachrichtentyp, wie bei DUP, Eins oder Null übergeben werden. Bei den anderen Nachrichtentypen ist dieser Wert immer Null. Eins bedeutet, dass der Broker die letzte zu jedem Topic gesendete Nachricht speichert. Jeder neue Klient, der ein Topic abonniert, erhält somit die letzte zugehörige Nachricht (Götz et al., 2015b).

Die nächsten eins bis vier Bytes, beginnend ab dem zweiten Byte des Pakets, ist die sogenannte verbleibende Länge (Remaining Length). Sie ergibt die Anzahl der im aktuellen Paket verbleibenden Bytes, einschließlich der Daten im Variablenheader und in der Nutzlast (Götz et al., 2015b).

1.5.3 TLS/SSL

In der technisch orientierten Welt wird großer Wert auf die Sicherheit von Internet of Things basierenden Smart Homes gelegt. Da die beiden Protokolle TLS und SSL die gleichen Verschlüsselungsprotokolle nutzen, ist es wichtig zu zeigen, wie diese grundsätzlich funktionieren. Transport Layer Security (TLS) und dessen Vorgänger, Secure Sockets Layer (SSL), sind Verschlüsselungsprotokolle, die Schutz für die Kommunikation über das Internet bieten. Die TLS- und SSL-Protokolle verschlüsseln Netzwerkverbindungssegmente über die Transportschicht. Die beiden Protokolle unterscheiden sich wenig, daher wird meistens die Form SSL/TLS verwendet. Das SSL/TLS-Protokoll wird zur Verschlüsselung der Serverdatenübertragung sowie zur Serverauthentifizierung durch ein digitales Zertifikat verwendet. Die Absicht, SSL/TLS-Sicherheit zu verwenden, muss vom Client an den Server übermittelt werden (Grigorik, 2013, pp. 50-53).

Es gibt zwei Möglichkeiten für den Client, um dem Server bekanntzugeben, dass er SSL/TLS verwenden möchte. Entweder mit einem separaten Port wie bei HTTP (Port 80), das HTTPS (Port 443) oder der Client fordert den Server auf, SSL/TLS für die vorhandene Verbindung über einen protokollspezifischen Mechanismus zu verwenden. Der Betrieb der Zertifikate basiert auf asymmetrischer Verschlüsselung, wobei beide Seiten über zwei Schlüssel verfügen. Diese sind der private und der öffentliche Schlüssel (private and public key). Der öffentliche Schlüssel kann veröffentlicht werden, und wenn dieser öffentliche Schlüssel zum Verschlüsseln von Daten verwendet wird, ist bereits sichergestellt, dass diese Daten nur mit dem privaten Schlüssel, welcher zu dem bei der Verschlüsselung verwendeten öffentlichen Schlüssel gehört, entschlüsselt werden können (Grigorik, 2013, pp. 50-53).

Der Vorteil der Verschlüsselung mit öffentlichem Schlüssel besteht darin, dass der Sender und der Empfänger während der Kommunikation kein geheimes Kennwort oder keinen geheimen Schlüssel austauschen müssen. Stattdessen verfügt jeder Benutzer über ein Schlüsselpaar, mit dem eine sichere Kommunikation hergestellt werden kann. Die beiden Schlüssel stammen aus demselben Schlüsselerzeugungsprozess, welche untrennbar miteinander verbunden sind, aber einer kann nicht vom anderen abgeleitet werden. Der private Schlüssel muss vor allen Benutzern geheim gehalten werden, während der öffentliche Schlüssel an jedermann weitergegeben werden kann (Grigorik, 2013, pp. 50-53).

Eine Verbindung basierend auf dem Three-Way-Handshake Verfahren ist im Abschnitt 1.4 TCP/IP beschrieben. Die Abbildung (Abb. 1.8) zeigt die gleiche Verbindung nur mit TLS Verschlüsselung. Nach der Verbindung sendet der Client eine ClientHello-Nachricht an den Server, über seine größte unterstützte TLS-Version. Der Server antwortet dem Client mit einer ServerHello-Nachricht und sendet ein Zertifikat, das den öffentlichen Schlüssel des Servers enthält. Der Server verarbeitet die vom Client gesendeten Daten und sendet eine verschlüsselte Fertigmeldung an den Client zurück.

Der Client entschlüsselt die Nachricht mit dem symmetrischen Schlüssel. Danach endet der Handshake und das Anwendungsprotokoll wird aktiviert. Wie in der Abbildung 1.8 gezeigt wird, erhöht die Verschlüsselung die Dauer des Verbindungsaufbaus um zirka 200% (Grigorik, 2013, pp. 50-53).

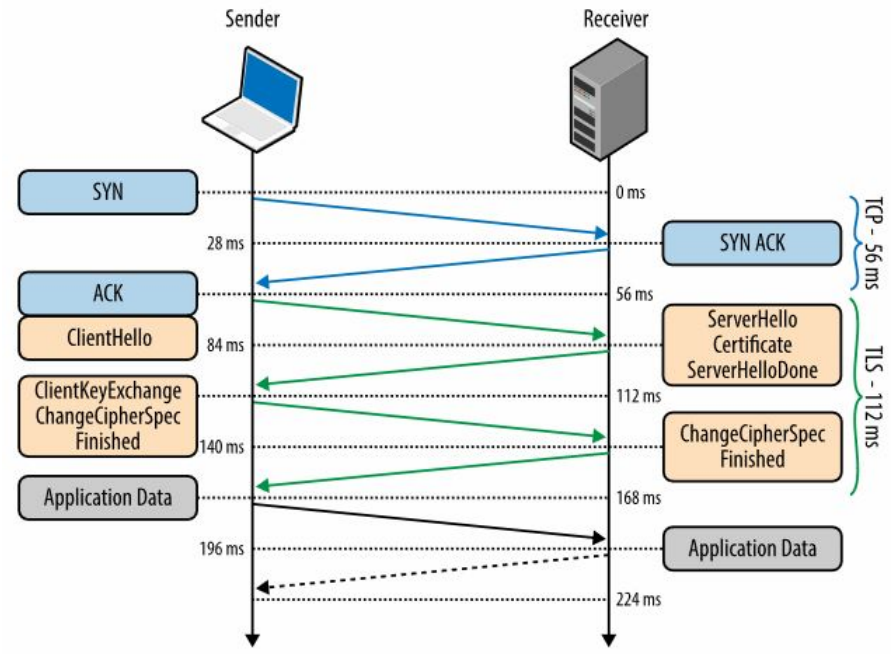


Abbildung 1.8: TCP Three-Way-Handshake mit TLS Verschlüsselung. Auf der rechten Seite sind die zu TCP und zu TLS gehörenden Ablaufzeiten angegeben.

1.5.4 HTTP vs. MQTT

Wie oben beschrieben, sind sowohl HTTP als auch MQTT grundsätzlich für die Nutzung im Internet der Dinge geeignet. Wie die IoT Statistiken (Eclipse Foundation Inc., 2019) zeigen, werden sie in verschiedenen IoT-Projekten auf ähnliche Weise verwendet. Die folgende Tabelle (Tabelle 1.1, (Lampkin et al., 2012)) zeigt die signifikanten Unterschiede zwischen den beiden Protokollen.

Das HTTP-Protokoll basiert auf einer komplexen Struktur und auf dem Request-Response Prinzip. Damit der Klient Daten an den Server senden kann, muss vor allem eine Verbindung hergestellt werden. Nach dem erfolgreichen Verbindungsaufbau kann das Senden von Daten erfolgen. Die Verbindung wird jedoch beendet, sobald die Datenübertragung abgeschlossen ist. Um Daten erneut zu senden, ist eine neue Verbindung erforderlich. Dies kann zu einem Zeitverlust führen. Im Gegensatz dazu ist das MQTT-Protokoll einfach aufgebaut und basiert auf dem Publish-Subscribe-Prinzip. Bei diesem Protokoll bleibt der Klient nach dem Verbindungsaufbau mit dem Server verbunden. Dadurch kann die Sendung von Daten schnell er-

	MQTT	HTTP
Komplexität	Einfach	Komplex
Muster	Publish-Subscribe	Request-Response
Designorientierung	Datenzentriert	Dokumentzentriert
Datenverteilung	1-1, 1-n, n-m	1-1
Nachrichtengröße	Klein, Header nur 2 bytes	Groß, Text-basierte Nachrichten
Service level	Qos 0, Qos 1, QoS 2	Alle Nachrichten haben die gleiche Stufe

Tabelle 1.1: Vergleich der wichtigsten Aspekte von HTTP- und MQTT-Protokoll.

folgen, da der Klient nicht auf die nochmalige Herstellung einer Verbindung warten muss. MQTT unterstützt weiterhin die one-to-one (1-1), one-to-many (1-n) und many-to-many (n-m) Verbindung, während HTTP nur die one-to-one (1-1) Verbindung kennt. HTTP ist dokumentzentriert (document-centric) und die damit zu verwendenden Dokumenttypen, wie zum Beispiel Content-Type, sind durch den MIME-Standard definiert. Die Nachrichten sind größer, da die Nachrichten textbasiert sind, und das Protokoll selbst ist komplexer. MQTT ist datenzentriert (data-centric), das heißt, es kümmert sich nicht den Inhalt der Nachricht, sondern überträgt sie nur in Form von Byte-Arrays. Die Komplexität ist mit nur wenigen Nachrichtentypen einfacher. Die Nachrichtengröße ist klein, die Headergröße beträgt nur zwei Bytes. Ein weiterer Vorteil von MQTT gegenüber HTTP besteht darin, dass es drei QoS-Ebenen unterstützt, wodurch die Qualität der Zustellung sichergestellt werden kann.

Expertinnen und Experten haben wissenschaftliche Publikationen über den daten- und zeitbezogenen Vergleich der beiden Protokolle erstellt. Eine Studie, durchgeführt von Yokotani und Sasaki im Jahr 2016 hat die Beziehung zwischen der Gesamtlänge von Topics in MQTT und der Anzahl der Übertragungsbytes (transmission bytes) untersucht. Da die Nutzlastgröße (payload size) der Applikation bei MQTT Null ist, ergibt sich die Anzahl der Übertragungsbytes nur durch den Protokol Overhead. Da HTTP Topics nicht unterstützt, hängt die Anzahl der Übertragungsbytes neben dem Protokol Overhead auch von der Nutzlastgröße ab (Yokotani & Sasaki, 2016).

Charlie Wang hat 2018 ein Experiment durchgeführt, wobei er Messungen über die während der einzelnen Übertragungszustände gesendeten Paketgrößen gemacht hat. Seine Ergebnisse haben folgendes gezeigt: Wenn eine Verbindung erstellt, ein Paket übertragen und die Verbindung abgebaut wird, ist das Paket des HTTP Protokolls insgesamt um zirka 13% kleiner. Von diesem Paket macht das Aufbauen und Abbauen der Verbindung mehr als 90% aus. Da bei MQTT eine Verbindung nicht gleich nach der Übertragung abgebrochen wird, und sie bei der nächsten Nachricht wiederverwendet werden kann, liegen die durchschnittliche Antwortzeit ungefähr bei 40 ms und die durchschnittliche Paketgröße bei zirka 400 Bytes (Wang, 2018).

1.6 Datenverwaltung

Daten spielen eine zentrale Rolle in IoT-Systemen. Um die von Sensoren gesammelten riesige Datenmenge verarbeiten zu können, muss die richtige Art von Datenbank zur Verfügung stehen. Einige grundlegende Funktionen einer Datenbank sind durch das CAP-Theorem von Brewer angegeben. Das CAP-Theorem besagt, dass ein verteiltes System bis zu zwei der folgenden drei grundlegenden Funktionen implementieren kann (Rautmare & Bhalerao, 2016):

1. Konsistenz (Consistency): Ein verteiltes System ist konsistent, wenn zu einem bestimmten Zeitpunkt der Wert einer Dateneinheit von einem beliebigen Knoten abgefragt wird und derselbe Wert erhalten wird.

2. Verfügbarkeit (Availability): Verfügbarkeit in einem verteilten System ist dann gewährleistet, wenn alle Anfragen vom System beantwortet werden.

3. Partitionstoleranz (Partition tolerance): Diese stellt die Zuverlässigkeit des Systems dar, das heißt, das System reagiert korrekt auf eine Anforderung wenn es partitioniert wird oder ein Knoten ausfällt, sofern nicht das gesamte Netzwerk ausgefallen ist.

Laut der IoT-Umfrage von Eclipse aus dem Jahr 2018, verwenden 93 Prozent der Befragten eine Open-Source Datenbank. Die Ergebnisse der Umfrage sind in der unteren Abbildung (Abb. 1.9) dargestellt. Die am häufigsten verwendete Datenbank ist die relationale Datenbank MySQL. Auf den nächsten zwei Plätzen sind zwei neuere, nicht relationale, sogenannte NoSQL-Datenbanken, nämlich MongoDB und InfluxDB. Die anderen Datenbanken verteilen sich ungefähr gleich auf (Eclipse Foundation Inc., 2018).

In den folgenden Abschnitten werden die wesentliche Unterschiede zwischen den beiden in IoT Projekten meist verwendeten Datenbanken, MySQL und MongoDB erläutert.

MySQL ist eines der weltweit am meisten verwendeten relationalen Datenbankverwaltungssysteme (RDBMS), das der bekannten Server-Client-Architektur folgt. Die Hauptgründe für die Verbreitung sind Benutzerfreundlichkeit, einfache Einstellbarkeit und Open Source. Die strukturierte Abfragesprache (Structured Query Language, SQL) ist eine Sprache, mit welcher Daten in einer relationalen Datenbank gespeichert, verwaltet und abgerufen werden können. Die Datenbank ist strukturiert, aus diesem Grund erfordert das Datenbankdesign einen ernsthaften theoretischen Hintergrund und es ist schwer, die am Anfang erstellten Schemata zu ändern. Die Datenbank ist nur vertikal skalierbar. Die Daten werden in Tabellen gespeichert, die für jeden Datensatz dieselben Felder und dieselbe Reihenfolge aufweisen. Jedem Feld

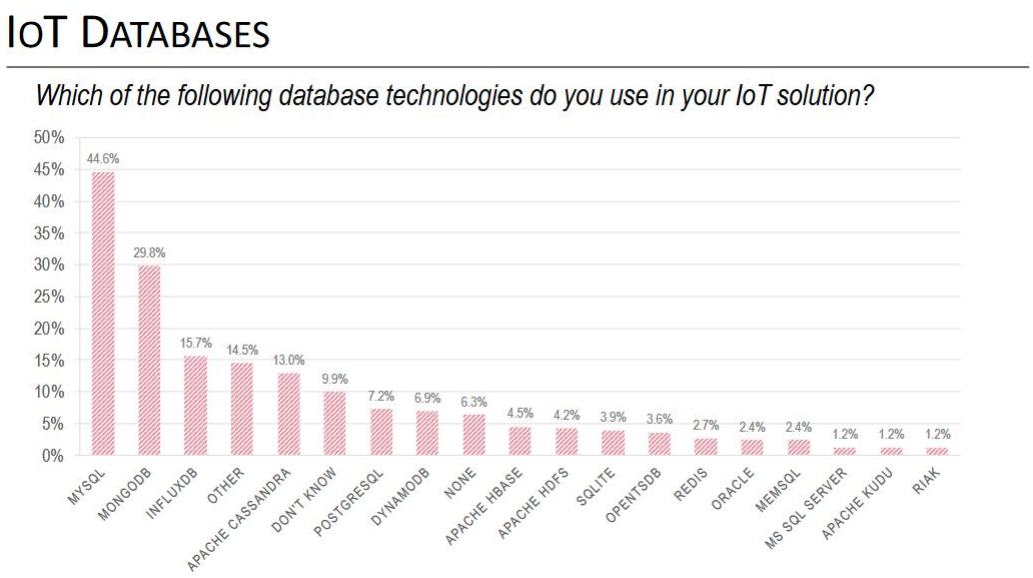


Abbildung 1.9: Verteilung der Datenbanktechnologien, die in IoT-Projekten verwendet werden.

muss ein Datentyp zugewiesen werden. Datentypen sind zum Beispiel Varchar2, Number, Timestamp oder Date. Die Beziehungen zwischen Tabellen können mit sogenannten JOINs definiert und erzeugt werden. Bei diesem Datenbanktyp kann man Backups erstellen, um die Daten zu sichern (Rautmare & Bhalerao, 2016).

MongoDB ist ein dokumentenorientiertes Open Source-Datenbanksystem. Es gehört zu den NoSQL-Datenbankservern. Im Gegensatz zu MySQL ist es nicht notwendig, die einzelnen Felder zu deklarieren. MongoDB weist horizontale Skalierbarkeit auf, was die Datenbankerweiterung erleichtert und wodurch größere Datenmengen darin gespeichert werden können. MongoDB unterstützt die Master-Slave-Replikation. In diesem Fall kann der Master Schreiboperationen ausführen, die Slave-Server kopieren die Daten und können zum Sichern verwendet werden. Slave-Datenbanken können eine neue Master-Datenbank auswählen, wenn der Master ausfällt. Im Gegensätz zu MySQL, speichert MongoDB die Daten in einer Dokumentenstruktur. Die Dokumente sind als Schlüssel-Wert-Paare (Key-Value-Pairs), in einem JSON ähnlichen Format, BSON binär abgelegt. Als Abfragesprache nutzt MongoDB eine Javascriptbasierte Abfragesprache, wobei die in der Datenbank zu speichernden Daten im JSON-Format übergeben werden (MongoDB Inc., 2019).

In einer Studie von Gyorodi (Győrödi, Gyorodi, Pecherle & Olah, 2015) wurden die Basis-Operationen, select, insert, update und delete der beiden Datenbanken verglichen. Die Ausführungszeiten wurden gemessen, wobei MongoDB signifikant kürzere Ausführungszeiten lieferte. Daraus folgt, dass MongoDB für große Datenmengen besser geeignet ist, als MySQL.

2 Hausautomatisierung

Es gibt zwei Hauptkategorien von Netzwerken: kabelgebunden und kabellos. Obwohl beide Typen ihre eigenen Vor- und Nachteile haben, die in diesem Kapitel vorgestellt werden, erleichtern drahtlose Netzwerke die Automatisierung eines vorhandenen Hauses. Die vier häufigsten drahtlosen Technologien, die in der Hausautomation verwendet werden, sollten verglichen werden.

2.1 Smart Home

Die Entwicklung des Internets der Dinge hat es möglich gemacht, andere neue Technologien und Konzepte, wie beispielsweise Hausautomatisierung entstehen zu lassen. Der Wunsch der Menschen nach Komfort und Bequemlichkeit steigt mit dem Fortschritt der Technologie. Durch das rasante Bevölkerungswachstum und den stetig steigenden Energiebedarf muss immer mehr Wert auf ressourcenschonenden Umgang gelegt werden. Bei intelligenten Häusern stehen Komfort und Energiesparen nicht im Widerspruch zueinander. Intelligenz bedeutet in diesem Fall, dass diese Häuser wissen, was und wann sie etwas tun sollten. Es gibt viele Möglichkeiten, dem Komfort einer Person zu dienen und diesen zu verbessern. Ein gutes Beispiel dafür ist unsere Morgenroutine. Morgens wird die Heizung im Badezimmer automatisch eingeschaltet, bevor der Benutzer/die Benutzerin aufwacht, um dieses vorzuwärmen. Nach dem Aufwachen werden allmählich die Rolläden hochgezogen, während die Kaffeemaschine in der Küche eingeschaltet wird. Es kann sogar durch den Kühlschrank die tägliche Einkaufsliste vorbereitet und per E-Mail versendet werden.

Wenn der Eigentümer/die Eigentümerin nicht Zuhause ist, wird das verbleibende Licht abgeschaltet und die Heizung ausgeschaltet, während der gleiche Raum belüftet wird, was ebenfalls Energie spart, oder der Benutzer/die Benutzerin wird benachrichtigt bzw. alarmiert, wenn jemand die Tür oder ein Fenster geöffnet hat, wodurch zusätzlich die Sicherheit und der Schutz des Hauses verbessert werden. Obwohl diese Funktionen unterschiedlich sind, koordiniert ein Smart-Home-System den Betrieb unabhängiger Systeme, die in einem Netzwerk integriert sind, sodass sie am selben Ort eingerichtet und verwaltet werden können. Man kann weiters Statusinformationen abrufen, um sicherzustellen, dass das Licht oder der Herd in der Wohnung ausgeschaltet ist oder alle offenen Fenster zugemacht worden sind.

Ein Ziel des Smart Homes ist es, das Komfortgefühl so zu verbessern, dass ein Zustand mit möglichst wenig Bedienvorgängen des Benutzers/der Benutzerin erreicht werden kann. Dies kann beispielsweise unter Verwendung der sogenannten Einknopfbedienung (Single-button operation) erreicht werden. Daduch können im Wesentlichen mehrere vordefinierte Aufgaben gleichzeitig per Knopfdruck gestartet werden. Wenn man zum Beispiel die Taste Älles aus"bestätigt, können alle Lampen im Haus ausgeschaltet, die Fenster zugemacht, die Heizung auf eine niedrigere Temperatur umgestellt und das Sicherheitssystem eingeschaltet werden (Kaiser, 2015). Es gibt verschiedene Möglichkeiten, um sowohl die einzelnen Teile als auch das gesamte System zu verwalten und zu steuern. Eine davon ist die stationäre Steuerung. In diesem Fall kann das Smart Home entweder konventionell oder mit an der Wand montierten LCD-Touchscreens beobachtet und gesteuert werden. Der Hauptpunkt eines Smart Homes liegt jedoch nicht darin, sondern in der gemeinsamen Nutzung der Geräte und des Internets der Dinge. Mithilfe verschiedener Anwendungen können die integrierten Geräte des Smart Homes jederzeit und überall überwacht und gesteuert werden. Die Anwendungen sind auf Smartphones, Tablets oder sogar auf PCs lauffähig.

Ein Smart Home wird technisch aus den folgenden fünf Teilen aufgebaut (Varma & Bharadwaj, 2016):

1. Geräte unter Kontrolle (devices under control, DUC)

2. Sensoren und Aktoren

3. Steuerungsnetzwerk

4. Controller

5. Fernbedienungsgeräte

In die Gruppe „Geräte unter Kontrolle“ gehören jeweils Geräte, die mit dem Hausautomatisierungssystem verbunden und durch dieses kontrolliert sind. Die einzelnen Komponenten können, wie in Kapitel 1.4 beschrieben ist, mit Hilfe verschiedener drahtloser oder drahtgebundener Technologien, beispielsweise mit ZigBee, direkt mit dem Steuerungsnetzwerk eine Verbindung aufbauen (Varma & Bharadwaj, 2016).

Die nächste Gruppe sind Sensoren bzw. Aktoren. Ein Sensor ist ein Element, welches den Zustand oder die Veränderung der Umgebung misst, aber er beeinflusst die zu messende Eigenschaft nicht. Zu diesen Eigenschaften gehören zum Beispiel die Temperatur, die Luftfeuchtigkeit, Gas, Licht und die Erkennung von Bewegung oder Lärm. Somit fungieren die mit Sensoren ausgestatteten Geräte als Kollektoren. Abhängig vom Einsatzgebiet kann man zwei Arten von Aktoren unterscheiden:

Elektrische Aktoren, wie zum Beispiel elektrische Switches oder mechanische Aktoren, wie etwa elektrische Motoren. Die mit Aktoren eingebetteten Geräte fungieren als Performer. Die Verbindung zwischen den Geräten unter Kontrolle, Sensoren und Aktoren bzw. zwischen dem Controller und den Fernbedienungsgeräten, wird durch das Steuerungsnetzwerk hergestellt. Für die Steuerungsnetzwerke im Bereich Smart Home stehen drei Technologien zur Verfügung. Diese sind das Power Line-Kommunikation (PLC) und die auch schon oben erwähnten drahtlosen bzw. drahtgebundenen Übertragungstechnologien. Grundsätzlich kann man sagen, PLC, wie zum Beispiel X10 und drahlose Technologie, wie ZigBee, WLAN oder Bluetooth werden in der Hausautomatisierung aufgrund der niedrigen Kosten der einzelnen Komponenten und der Installation am häufigsten verwendet (Varma & Bharadwaj, 2016).

Der nächste Teil ist der Controller, welcher als Gehirn des Systems bezeichnet wird. Es handelt sich dabei um ein Steuerungssystem, das ein Computer oder ein spezielles Gerät sein kann, welches die Steuerung des Systems sicherstellt. Hier werden die Informationen von den einzelnen Sensoren bzw. Systemkomponenten sowohl angenommen als auch verarbeitet und an dieser Stelle werden die auszuführenden Befehle an jedes Steuergerät übertragen. Ein Controller ist normalerweise ein ïmmer anëigenständiges Gerät oder ein eigebettetes System, meistens Linux, auf dem die Steueranwendung für das Haus ausgeführt wird. Zu den Fernbedienungsgeräte zählen Smartphones, Tablets, Laptops oder PCs, die eine Verbindung zur auf dem Controller laufenden Anwendung herstellen können (Varma & Bharadwaj, 2016).

Die Steuerung des Smart Homes kann grundsätzlich auf zwei Arten erfolgen, abhängig davon, ob die Intelligenz des Systems in einer zentralen Komponente (zentrale Steuerung), oder in allen Komponenten (dezentrale Steuerung) untergebracht ist. In einem dezentralen System ist jedes Element unabhängig vom anderen und verfügt über einen eigenen Mikrocontroller und eine eigene Intelligenz. Die Geräte sind einzeln mit dem Internet verbunden und sie kommunizieren auch miteinander. Da alle Geräte eine eigene Intelligenz haben, können alle als Stand-alone-Geräte genutzt werden. Daraus folgt, wenn ein Element ausfällt, funktioniert trotzdem das gesamte System, natürlich mit Ausnahme des defekten Elements, weiter. Diese Systeme sind im Allgemeinen günstiger, da es ausreicht, das Gerät selbst zu kaufen welches sofort einsatzbereit ist. Bei diesen Systemen ist keine separate Zentraleinheit erforderlich. Wie zuvor erwähnt, müssen alle Komponenten über die entsprechende Intelligenz verfügen, darum ist eine Erweiterung des Systems schwieriger. Neben der Erweiterung ist die Automatisierung auch schwierig, weil es nicht immer möglich ist, Geräte verschiedener Hersteller in das System zu integrieren (Kaiser, 2015, pp. 74-76).

Im Fall eines zentralisierten Systems gibt es einen zentralen, logischen Controller, der die Intelligenz beinhaltet. Der Controller verbindet sich über das Heimnetzwerk mit dem Internet, und die anderen intelligenten Geräte sind mit dem Controller verbunden. Im Gegensatz zu einem dezentralen System, bei dem jede Einheit separat mit dem Internet verbunden werden muss, belasten zentralisierte Geräte den Router nicht, da der Router nur die Zentraleinheit bedienen muss. Die Signale der Senso-

ren werden direkt an den Controller geschickt. Der Controller entscheidet danach, welche Aktionen von bestimmten Aktuatoren ausgeführt werden müssen. Ein anderer Vorteil eines zentralisierten Systems ist, dass das gesamte Gebäude mittels einer einzigen Applikation, die ausschließlich mit dem Controller kommuniziert, gesteuert und überwacht werden kann. Der größte Nachteil besteht darin, dass bei einem Ausfall der Zentraleinheit das gesamte Smart Home unbrauchbar wird. Obwohl die Erweiterung dieses Systems billiger ist, als bei einem dezentralen System, sind diese Systeme im Allgemeinen teurer, da der zentrale Controller das teuerste Teil der Installation ist (Kaiser, 2015, pp. 74-76).

2.2 Verbindung

In der Datenübertragung eines Hausautomatisierungssystems kann man grundsätzlich zwischen zwei Hauptkategorien, den drahtgebundenen und den drahtlosen Systemen unterscheiden. Drahtgebundene Systeme waren beliebt bei der Entwicklung der ersten Hausautomationssysteme. Heutzutage werden sie immer mehr in den Hintergrund gedrängt (Park, Sthapit & Pyun, 2009, p.1). Hinzu kommt meistens die Notwendigkeit einer Vorausplanung des Systems und die Komplexität des Aus- und Umbaus der installierten Geräte. Da die Geräte mit Kabel verbunden werden und die Kabel sowohl aus ästhetischen als auch aus Sicherheitsgründen in die Wand verlegt werden müssen, ist der Ausbau eines kabelgebundenen Smartsystems vor allem beim Neu- oder Umbau eines Hauses bzw. einer Wohnung zu empfehlen. Die zwei größten Vorteile eines kabelgebundenen Systems in der Hausautomatisierung sind eine stabile Verbindung und bessere Sicherheit. Solange ein Kabel nicht fehlerhaft oder beschädigt ist, ist die Kommunikation zwischen den Geräten sehr zuverlässig. Da die Geräte mit einer festen, und nicht mit einer externen Stromquelle ausgestattet sind, ist eine Wartung, wie zum Beispiel Batterieaustausch nicht notwendig. Man muss physisch mit dem System verbunden sein, um darauf zugreifen zu können. Diese Eigenschaft erschwert den unbefugten Zugriff von Hackern. Aufgrund der oben erwähnten Installationskomplexität ist es stark empfohlen, eine technische Fachkraft oder einen Spezialisten mit der Installation zu beauftragen, was natürlich zu einem höheren Preis führt (SafeWise Team, 2020). Hingegen können drahtlose Hausautomatisierungssysteme viel billiger werden, weil man diese in den meisten Fällen auch selbst, ohne die Einbeziehung eines Fachexperten/einer Fachexpertin ausbauen kann. Ein weiterer Vorteil ist der, dass keine Verkabelung notwendig ist. Das erhöht die Installationsgeschwindigkeit und schafft die Möglichkeit, die vorher installierten Geräte an einen anderen Platz zu übertragen und in das bereits laufende System neue Sensoren und Geräte zu integrieren. Da keine Kabel vorhanden sind, besteht keine Gefahr einer physischen Beschädigung der Kommunikation. Während kabelgebundene Systeme nur von zentralen, vorinstallierten Standorten gesteuert werden können, ist der Zugriff auf kabellose Systeme von überall, zum Beispiel in der Arbeit von einem Smart Phone aus möglich. Diese Eigenschaft ist ebenfalls als Nachteil zu sehen.

Diese Systeme sind einfacher, ohne physischen Zugriff zugänglich und dadurch besteht eine höhere Sicherheitsgefahr, als bei einem drahtgebundenen System. In den meisten Fällen werden die Geräte in einem drahtlosen System mit Batterien betrieben. Sie müssen regelmäßig überprüft und ausgetauscht werden, um ein fehlerfreies System zu gewährleisten. Um dies zu erleichtern, kann der Batteriestand der Geräte abgefragt werden. Darüber hinaus zählt auch die niedrigere Reichweite als Nachteil. Abhängig von der Technologie kann diese zwischen 10 und 150 Metern sein.

2.3 Drahtlose Technologien

Da der Fokus dieser Masterarbeit auf den Aufbau eines selbstgebauten Hausautomatisierungssystems gelegt werden soll, werden im Folgenden die drahtlosen Technologien im Vordergrund stehen. In den nächsten Kapiteln werden die in diesem Bereich meist verwendeten Netzwerktechnologien vorgestellt bzw. deren wesentliche Unterschiede sowie die Vor- und Nachteile hervorgehoben.

2.3.1 WLAN

WLAN (Wireless Local Area Network), auch als Wi-Fi (Wireless Fidelity) bekannt, ist die drahtlose Version der Ethernet LAN-Technologie, womit bis zu 250 Geräte in einem Netzwerk verbunden sein können. Es ist durch die IEEE 802.11 Normenfamilie definiert und hat mehrere Untergruppen, wie zum Beispiel IEEE 802.11abnac. DDer Unterschied zwischen den einzelnen Normen liegt in der verwendeten technischen Lösung und der verfügbaren Geschwindigkeit. Für die Übermittlung verwendet WLAN Radiowellen, die abhängig von der Norm sowohl das 2.4 GHz als auch das 5 GHz Frequenzband verwenden. Das 2.4-GHz-Band ist gesättigter als das 5-GHz-Band, da dieses früher von vielen anderen Geräten verwendet wurde. Die Daten werden durch die sogenannte Funknetzwerkkarte in ein Signal umgewandelt, wodurch sie versendet werden können (Horyachyy, 2017, pp. 26-27). Die Übertragungsrate bei den verschiedenen Gruppen des WLANs unterscheidet sich. Allgemein ist zu sagen, dass beim 2.4 GHz-Band (IEEE 802.11ac) die Rate zwischen 450 und 600 Mbs liegt. Bei dem 5 GHz-Band ist dieser Wert 1300 Mbs (Social WiFi, 2017). Mit einem WLAN-Router können sich theoretisch 255 Geräte verbinden. Diese Anzahl liegt in der Praxis nur bei 45 Geräten, da zu viele Geräte das Netzwerk signifikant verlangsamen. Das WLAN hat eine Reichweite von 10 bis 150 Metern (Morales, Lopez, Parado & Pasaoa, 2016, p.2). Ein WLAN besteht aus zwei Hauptteilen. Der eine Teil ist der Zugangspunkt (Access Point, AP), beispielsweise ein Router, welcher das Senden und Empfangen von Daten zwischen drahtlosen Geräten und dem Internet ermöglicht. Der andere ist die Netzwerkkarte, welche sicherstellt, dass Geräte ständig über ein kabelgebundenes Netzwerk verbunden sind und die vom AP gesendeten Daten

empfangen und umgekehrt, Daten an den AP übermittelt werden (Jobbagy, 2016, pp. 306-308).

2.3.2 Bluetooth

Die Bluetooth Technologie wurde zum ersten Mal von der Firma Ericsson im Jahr 1994 publiziert. Ziel war eine kabellose Datenübertragungstechnologie für kurze Entfernungen zu entwickeln, hauptsächlich für kleinere Geräte wie Kopfhörer oder Mikrofone, um die bis dahin verwendeten Kabelverbindungen zu ersetzen. Bluetooth basiert auf dem IEEE 802.15.4 Standard und verfügt über mehrere Versionen. Bei den Versionen ist besonders die Version 4.0 zu erwähnen, die nicht nur auf Geschwindigkeit, sondern auch auf einen viel geringeren Energieverbrauch als zuvor abgezielt hat. Diese Technologie wird als Bluetooth LE (Low Energy) oder Bluetooth Smart bezeichnet. Wie die älteren Versionen von WLAN, verwendet auch BLE das 2.4 GHz-Frequenzband. Ein Bluetooth-Netzwerk kann nur aus 7 Geräten bestehen. Die Übertragungsrate bei BLE ist niedriger als bei WLAN, sie liegt zwischen 1 und 3 Mbs. Die Entfernung liegt abhängig von den Geräten zwischen 10 und 100 Meter (Morales et al., 2016, p.3).

2.3.3 ZigBee

ZigBee ist eine drahtlose Technologie, die auf dem Standard IEEE 802.15.4 basiert und sich auf die Verbindung und das Steuern von Heimgeräten konzentriert. Diese Technologie verwendet dieselbe Frequenz wie die beiden zuvor beschriebenen Technologien, also ebenfalls 2.4 GHz. Außer diesem Frequenzband stehen in Amerika das 915 MHz und in Europa das 868 MHz-Frequenzband zur Verfügung, die aber niedrigere Datenraten ermöglichen (Kühner, 2009, pp. 205-207). Die maximale Übertragungsrate ist allgemein deutlich geringer als sowohl bei WLAN als auch bei Bluetooth, da hier dieser Wert nicht im Mittelpunkt der Entwicklung gestanden ist. Sie beträgt zirka 20 bis 250 Kbs. Bei der Reichweite bleibt ZigBee nicht hinter den anderen Technologien zurück, sie beträgt zwischen 20 und 100 Meter (Horyachyy, 2017, p. 61). Diese Technologie organisiert die am Netzwerk beteiligten Geräte in einer Stern- oder peer-to-peer Topologie. Im Zentrum steht immer genau ein Netzwerkkoordinator, auch als PAN-Koordinator (Personal Area Network-Koordinator) bezeichnet. Die zwei weiteren Typen sind funktionsreduzierte Geräte (reduced- function device, RFD) und Vollfunktionsgeräte (full-function device, FFD). Diese Topologie und die an dem Netzwerk teilnehmenden Geräte werden in dem folgenden Kapitel näher beschrieben. Der Energieverbrauch von ZigBee ähnelt der BLE-Technologie. Das Hauptziel von ZigBee ist, an Geräten mit sehr geringem Stromverbrauch zu arbeiten und deren Batterielebensdauer zu verlängern. ZigBee kann bis zu 65.000 Knoten in

einem Netzwerk unterstützen. Darüber hinaus können mehrere Netzwerkkoordinatoren verbunden werden, die zusammen ein sehr großes Netzwerk abdecken können (Horyachyy, 2017, pp. 61-63).

2.3.4 Z-Wave

Die letzte Technologie, die in dieser Masterarbeit untersucht wird, ist die sogenannte Z-Wave Technologie. Genau wie bei ZigBee wurde der Schwerpunkt der Z-Wave-Entwicklung auf Heimautomatisierung gelegt. Diese Technologie basiert auch auf dem Standard IEEE 802.15.4, und ist abhängig von der Region in verschiedenen Frequenzbereichen verfügbar, beispielsweise in Europa 868 MHz und in den USA 908 MHz. Da WLAN in den meisten Häusern zu finden ist, könnte es einen großen Vorteil bringen, da es nicht von anderen Geräten, die das 2.4 GHz-Frequenzband verwenden, gestört wird. Durch die Tatsache, dass heute die Verbreitung des 2.4-GHz-WLANs abnimmt und die Verbreitung des 5-GHz-WLANs zunimmt, wird dieser Vorteil immer mehr in den Hintergrund gedrängt. Z-Wave ermöglicht die Übertragung mit Datenraten bis zu 100 Kb/s und ähnlich wie die anderen Technologien erreicht es eine Reichweite von 10 bis 75 Meter. In einem Z-Wave Netzwerk können maximal zirka 232 Geräte verbunden sein (Kühner, 2009, p. 206). Der Betrieb eines Z-Wave-Systems ist wie folgt aufgebaut: Innerhalb des Z-Wave-Systems folgen die Hochfrequenzsignale vordefinierten Pfaden. Der Kommunikationskanal stellt sicher, dass das von der Steuereinheit gesendete Funkwellensignal das zu steuernde Gerät jederzeit erreicht. Wenn der Befehl sein Ziel erreicht, meldet das zu steuernde Gerät sich an der Steuereinheit zurück. Wenn die Signalübertragung blockiert ist und keine Rückmeldung vorliegt, versucht das Z-Wave-System das Signal unter Verwendung anderer mit dem Z-Wave-Netzwerk verbundener Komponenten erneut zu übertragen. Dadurch wird die Systemstabilität sichergestellt (SafeWise Team, 2018).

2.3.5 Vergleich

In der folgenden Tabelle 2.1 sind die wesentlichen Unterschiede zwischen den oben beschriebenen Technologien zusammengefasst.

Wie dabei ersichtlich ist, sind die Unterschiede in der Reichweite bei den vier verglichenen Technologien unbedeutend. Die Reichweite hängt eigentlich mehr von dem Gebäude ab, beziehungsweise davon, wie die Signale durch die unterschiedlichen Wandtypen übertragen werden können. Aus dieser Zusammenfassung ist klar ersichtlich, dass die Größe der Brandbreite in starkem Zusammenhang mit dem Energieverbrauch steht. Beide Größen sind beim WLAN am größten. Da bei der Hausautomatisierung in den meisten Fälle keine massiven Datenmengen, außer bei Video-Streaming oder Dateiübertragungen zustande kommen, ist WLAN wegen seines ho-

	WLAN	Bluetooth	ZigBee	Z-Wave
Normung	IEEE 802.11 Familie	IEEE 802.15.1	IEEE 802.15.4	IEEE 802.15.4
Frequenzbereich (Europa)	2.4 / 5 GHz	2.4 GHz	2.4 GHz / 868 MHz	868 MHz
Bandbreite	450-1300 Mb/s	1-3 Mb/s	20-250 Kb/s	100 Kb/s
Entfernung	10-150 m	10-100 m	20-100 m	10-75 m
Energieverbrauch	Groß	Mittelgroß	Sehr klein	Ultra klein
Batterielebensdauer	1-3 Stunden	4-8 Stunden	2-3 Jahren	viele Jahren
Anzahl der Netzwerkknoten	∼250	7	∼65000	232
Hardwarekosten	Hoch	Mittelhoch	Wenig	Wenig

Tabelle 2.1: Vergleich den in der Hausautomatisierung meist verwendeten drahtlosen Technologien.

hen Energieverbrauchs für die Hausautomatisierung wenig effizient. Laut einer Studie von Lopez (Morales et al., 2016, p. 3) ist WLAN für Internet of People besser geeignet als für IoT. Obwohl Bluetooth über eine höhere Batterielebensdauer verfügt, ist der Energieverbrauch immer noch groß, obwohl sich das seit der Entwicklung der Bluetooth Low Energie Technologie gebessert hat. Das größere Problem bei dieser Technologie ist die geringe Anzahl der Netzwerkknoten. Da sich durchschnittlich 15 zu verbindende Geräte in einem normalen Haus befinden, ist die maximale Anzahl von sieben Knoten unzureichend. Dieses Problem kann allerdings durch den Einsatz eines Dongles eliminiert werden (Morales et al., 2016, p. 4). BLE wird von Tag zu Tag vermehrt im Gesundheitswesen verwendet. Der Grund, dass es in diesem Bereich so beliebt ist, kann sein, dass die Daten dabei nicht ständig zu übertragen sind, sondern in den meisten Fällen nur nach dem Gebrauch des Geräts. Dadurch kann der Strombedarf des Geräts bzw. die Anzahl der verbundenen Geräte im Netzwerksystem niedrig gehalten werden.

Bei der Entwicklung von ZigBee bzw. Z-Wave wurde der geringe Stromverbrauch gegen eine hohe Datenübertragungsrate in den Vordergrund gestellt. Daher können Geräte, welche eine dieser Technologien nutzen, mit Batterieversorgung mehrere Jahre lang ohne Batteriewechsel betrieben werden. Da WLAN in fast jedem Haushalt bereits vorhanden ist, bedeutet es einen großen Vorteil, dass die Geräte der beiden Technologien, die auf unterschiedlichen Frequenzen kommunizieren, nicht durch die 2.4 GHz WLAN-Frequenz beeinträchtigt werden. Das ist in solchen Bereichen wichtig, in dem viele WLAN-Netzwerke gleichzeitig erreichbar sind, beispielsweise in einem Wohnblock. Die Geräte von beiden Technologien verfügen über die sogenannte „Low Latency“ Eigenschaft. Das bedeutet, dass die Anzahl der Daten, die durch einen Knoten geht, gering ist. Aus diesem Grund ist die Chance einer Datenverstopfung kleiner, als bei den anderen Technologien. Eines der einzigartigsten Attribute von Z-Wave ist die Abwärtskompatibilität, das heißt alle vorherigen Versionen von Z-Wave sind mit neueren Versionen kompatibel (Morales et al., 2016, pp. 5-6).

Da sowohl ZigBee als auch Z-Wave direkt für Hausautomatisierung entwickelt wurde, bleibt die Frage, welche Technologie in eigenen Zuhause implementiert werden soll. Bei beiden Technologien gibt es zwischen dem Energieverbrauch wesentliche Unterschiede. ZigBee ist zu einer höheren Datenübertragungsrate bis zu 250 Kbs fähig, 2.5-mal mehr als Z-Wave. Die Kosten der Geräte sind in beiden Fällen relative niedrig. Was wiederum die Auswahl beeinflusst, ist die Anzahl der verfügbaren Geräte und die Anzahl der Unternehmen, die sie herstellen. ZigBee Geräte werden von vielen bekannten Firmen, wie Xiaomi, Philips oder Ikea und auch von vielen kleineren Firmen hergestellt. Aufgrund der oben genannten Vorteile bzw. der großen Auswahl an Produkten und Herstellern bietet die ZigBee-Technologie die geeignete Möglichkeit, ein nachhaltiges, kostengünstiges und sicheres Hausautomatisierungssystem aufzubauen.

3 ZigBee-Technologie

ZigBee ist seit 2002 auf dem Markt, wurde jedoch erst in den letzten Jahren bekannter. In diesem Kapitel werden der Betrieb und der Aufbau der ZigBee-Kommunikation beschrieben, ebenso wie diese Technologie mit dem IEEE 802.15.4 Standard interagiert und in welchen drei Netzwerk Topologien die Geräte organisiert werden können. Sicherheit ist ein wesentlicher Aspekt für jedes Protokoll, daher sollten die wichtigsten Sicherheitsmerkmale von ZigBee beschrieben werden.

3.1 WPAN

Die Entwicklung der drahtlosen Netzwerktechnologien war ein großer Schritt bei der Weiterentwicklung der Datenübertragungstechnologien. Bei den drahtlosen Netzwerktechnologien werden grundsätzlich vier Kategorien unterschieden. Diese sind nach Reichweite in der Abbildung (Abb. 3.1) dargestellt. Die zwei mit der größten Reichweite sind WWAN (Wireless Wide Area Network) und WMAN (Wireless Metropolitan Area Network). Zum Ersten gehören zum Beispiel Mobilfunknetze, wie 3G, 4G und sogar 5G. Diese Netzwerke sind weltweit erreichbar und obwohl sie früher nur geringere Datenraten anbieten konnten, ist die neueste Entwicklung der fünften Generation (5G) nach Angaben der deutschen Telekom in der Lage, Daten mit Geschwindigkeiten von 100 Mbs bis zu 10 Gbs zu übertragen (Deutsche Telekom AG, 2018). Die WMAN-Technologie wird für drahtlose Verbindungen zwischen Gebäuden in Städten verwendet (K. Sharma & Dhir, 2014, pp. 2-3).

In der Hausautomatisierung kann man sich zwischen zwei drahtlosen Netzwerkklassen entscheiden. Einerseits gibt es das WLAN (Wireless Local Area Network), wozu die oben erwähnte WiFi Technologie gehört, die innerhalb eines Gebäudes die Verbindung sicherstellt. Das Ziel dieser Netzwerke ist es, die kabelgebundenen Systeme mit kabellosen zu ersetzen, wobei die Eigenschaften des weiten Übertragungsabstands und der hohen Datenübertragungsrate bleiben. Der letzte Typ ist das sogenannte WPAN (Wireless Personal Area Network). Während sich die oben genannten Netzwerkklassen auf hohe Reichweite und große Bandbreite konzentrieren, legt die WPAN-Technologie auf eine kürzere, auf den Menschen zentrierte Reichweite Wert, während die Bandbreite eine geringere Rolle spielt. Der WPAN Standard wurde von

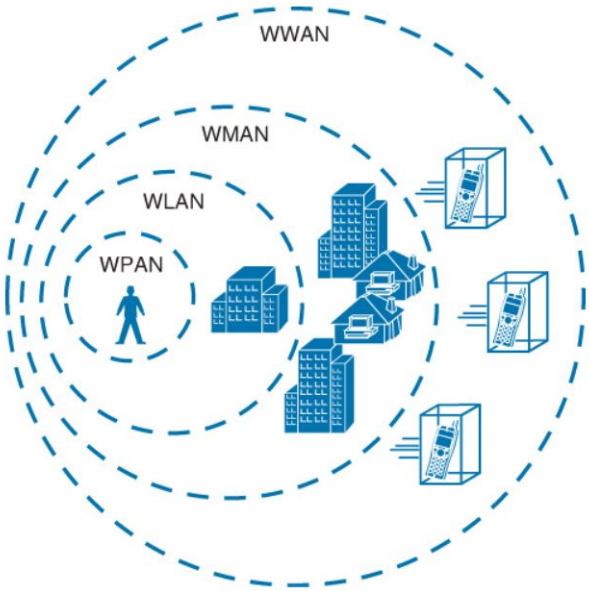


Abbildung 3.1: Drahtlose Netzwerkklassen nach Reichweite.

der IEEE 802.15-Arbeitsgruppe erstellt, die drei Klassen definiert hat. Diese Klassen unterscheiden sich in Datenrate, Batterieverbrauch und Servicequalität (QoS). Die einzelnen Technologien werden einer der folgenden Klassen zugewiesen (Ergen, 2004, p. 2):

- HR-WPAN (High Rate WPAN): Der Standard IEEE 802.15.3 gehört zu dieser Klasse mit Datenraten bis zu 55 Mbs. Dank dieser hohen Datenrate ist sogar eine Multimedia-Datenübertragung in Echtzeit möglich. Für diese Applikationen muss eine sehr hohe QoS garantiert werden.

- MR-WPAN (Medium Rate WPAN): Zu dieser Gruppe gehört auch Bluetooth, welches mit seiner Datenrate von 1-3 Mbs eine qualitativ hochwertige Sprachübertragung ermöglicht. Bei den Applikationen der MR-WPAN ist eine mittlere QoS notwendig.

- LR-WPAN (Low Rate WPAN): LR-WPANs sind einfache, kostengünstige Netzwerke, mit denen Geräte mit begrenzten Ressourcen eine drahtlose Verbindung mit geringem Durchsatz herstellen können. Bei der Entwicklung des IEEE 802.15.4-Standards für LR-WPAN-Netzwerke war es das Ziel, einen einfach zu installierenden Netzwerk-Typ mit geringer Reichweite und geringem Stromverbrauch zu schaffen. Zu dieser Klasse gehört auch die auf diesem Standard basierende ZigBee-Technologie.

3.2 Übersicht über ZigBee

Die ZigBee Technologie ist ein drahtloser Funkübertragungsstandard und wurde von der non-profit Organisation ZigBee Allianz Gruppe im Jahr 2002 entwickelt. An diesem Verein sind Hunderte von Unternehmen beteiligt, beispielsweise Motorola, Mitsubishi, Philips oder Xiaomi, um die ZigBee Technologie weiterzuentwickeln und zu verbessern. Das Ziel dieses Vereins ist, ein System zu entwickeln, welches Radiowellentechnologie für solche Geräte sicherstellt, bei denen die Batterielebensdauer wichtiger ist als die Möglichkeit einer großen Datenübertragung. Eine andere relevante Eigenschaft dieser Technologie, auf welche die Herstellerfirmen großen Wert legen, sind relativ niedrige Produktions- bzw. Wartungskosten. Die Teilnehmer der ZigBee Allianz arbeiten an Standards von der Netzwerkschicht bis zur Applikationsschicht. Die physische Schicht (Physical Layer, PHY) und die Medium Access Control (MAC) Protokolle der ZigBee Technologie wurden aus dem IEEE 802.15.4 Standard übernommen. Im Gegensatz zu den anderen Standards sind die minimalen Anforderungen bei ZigBee und IEEE 802.15.4 weniger begrenzt, wodurch sowohl deren Komplexität vereinfacht werden kann, als auch deren Implementationskosten reduziert werden können (Farahani, 2008, p. 2).

Wie im vorigen Kapitel erwähnt, ist für die Steuerung, Verwaltung und Kontrolle des Systems der sogenannte PAN-Koordinator zuständig. Zwischen den beteiligten Geräten in einem ZigBee-System kann man zwei Typen unterscheiden, nämlich das funktionsreduzierte Gerät (reduced- function device, RFD) und das Vollfunktionsgerät (full-function device, FFD). Die RFD-Geräte können nur als Netzwerkknoten mit dem Netzwerkkoordinator kommunizieren und sind billiger als der andere Typ. FFD hat eine erweiterte Funktionalität und kann als PAN-Koordinator, als Koordinator oder als Gerät fungieren (Kühner, 2009, p. 205).

Bei der Entwicklung von ZigBee war der niedrige Energieverbrauch der wichtigste Aspekt, weshalb das Protokoll dementsprechend optimiert wurde. Während des Betriebs kann ein ZigBee-Gerät in zwei Modi, im Aktiv- oder im Schlafmodus sein. Im Schlafmodus bleibt das Gerät solange, bis eine darauf laufende Applikation eine Anweisung gibt. Dank dessen kann die im aktiven Zustand verbrachte Zeit reduziert werden. Die im aktiven Modus verbrachte Zeit wird in der Literatur als Arbeitszyklen bezeichnet. Die meisten ZigBee-Geräte verbringen weniger als 1% im aktiven Modus (R. Sharma, Gupta, Suhas & Kashyap, 2014, pp. 3-4). Dies bedeutet, wenn ein Gerät beispielsweise jede Minute in einen aktiven Zustand übergeht und dort 0.6 Sekunden lang bleibt, hat dieses Gerät einen Arbeitszyklus von 0.01, sprich 1%. Als Ergebnis kann eine erhebliche Menge an Energie gespart werden, die sehr erheblich für die Sensoren und für andere Geräte mit niedrigen Datenraten ist, die lange Lebensdauer erfordern.

3.3 ZigBee Kommunikation

Die eigentliche physische Kommunikation findet auf der untersten Ebene des Modells statt. Die physikalische Schicht ist für die Übertragung von Bits über eine bestimmte Frequenz zum Kommunikationskanal verantwortlich. Auch die von der Sendeseite an den Kommunikationskanal gesendeten Bits müssen vom Empfänger richtig interpretiert werden (0 zu 0, 1 zu 1). Eine weitere Funktion der physischen Schicht besteht darin, die abgehenden Signale zu modulieren und die ankommenden Signale zu demodulieren, d.h. das ursprüngliche Signal, welches die Information trägt, wiederherzustellen (G., 2015, p. 50). Wie in dem vorherigen Kapitel beschrieben, ist ZigBee in Europa in zwei Frequenzbereichen, nämlich in 868 MHz und in 2.4 GHz verfügbar. Der erste Frequenzbereich besetzt nur einen Kanal und die Datenübertragungsrate kann nur ungefähr 20 Kbs erreichen. Obwohl bei dieser Frequenz nur wenige Daten übertragen werden können, hat diese Frequenz den Vorteil, dass sie von anderen Geräten, außer anderen ZigBee-Geräten nicht verwendet wird, wodurch dieser einzige Kanal einfacher freigehalten werden kann. Der andere Frequenzbereich hat 16 Kanäle, von Kanal Nummer 11 bis 26 zwischen 2.405 GHz und 2.480 GHz, mit einer Differenz von 0.005 GHz zwischen den einzelnen Kanälen. Bei diesen Frequenzen ist es möglich, die Daten mit 250 Kbs zu übertragen. WLAN ist im 21. Jahrhundert in den meisten Haushalten vorhanden. Die 2,4-GHz-Frequenz von WLAN ist in drei Hauptkanäle, Kanal eins, sechs und elf, die öffentlich zugänglich sind, unterteilt. Laut einer Publikation von der Firma Digi (Digi International Inc., 2020). kollidieren die ZigBee-Kanäle von Nummer 11 bis 24 mit den WLAN Hauptkanälen. Bei den restlichen zwei ZigBee Kanälen gibt es keine Überlappung mit WLAN, da sich die WLAN-Frequenzbereiche zwischen 2.402 GHz und 2.473 GHz befinden und diese beiden ZigBee Kanäle sich außerhalb dieser Frequenzbereiche befinden. Eine detaillierte Übersicht der einzelnen Frequenzen und der zugehörigen Kanäle der beiden Technologien sind in der Tabelle 3.1 dargestellt.

In den meisten Fällen sind die Netzwerkkanäle wegen der hohen Anzahl von WLAN-Netzwerken innerhalb eines Gebäudes bereits stark ausgelastet. Die zu überfüllten Kanäle können zu einer reduzierten Netzwerkleistung führen. Um sicherzustellen, dass ihre Übertragungen ohne Kollision stattfinden, verwenden ZigBee-Geräte den vorher erwähnten CSMA/CA-Algorithmus. Über der physischen Schicht befindet sich die sogenannte MAC Schicht (Medium Access Control). Diese stellt die Schnittstelle zwischen der physischen und der nächsthöheren Schicht (Netzwerkschicht) über der MAC Schicht bereit. Wie die physische Schicht, wurde auch die MAC Schicht vom Standard IEEE 802.15.4 übernommen. Die Merkmale der MAC-Unterschicht sind Beacon-Verwaltung, Kanalzugriff, GTS-Verwaltung, Rahmenvalidierung, bestätigte Rahmenübermittlung, Zuordnung und Aufhebung der Zuordnung (Ergen, 2004, p. 10).

Es werden zwei Betriebsarten für den Kanalzugriff unterschieden: der Nicht Beacon-fähige Modus und der Beacon-fähige Modus. Beacon ist als periodisches Paket zu

Frequenz	ZigBee-Kanal	WLAN-Kanal
2.405 GHz	11	1
2.410 GHz	12	1
2.415 GHz	13	1
2.420 GHz	14	1
2.425 GHz	15	6
2.430 GHz	16	6
2.435 GHz	17	6
2.440 GHz	18	6
2.445 GHz	19	6
2.450 GHz	20	11
2.455 GHz	21	11
2.460 GHz	22	11
2.465 GHz	23	11
2.470 GHz	24	11
2.475 GHz	25	-
2.480 GHz	26	-

Tabelle 3.1: Übersicht der Frequenzen der einzelnen Kanäle der ZigBee- bzw. WLAN-Technologien (Digi International Inc., 2019).

sehen. Im Beacon-fähigen Modus wird die Datenübertragung von dem Netzwerkkoordinator (PAN) synchronisiert und gesteuert. Der Netzwerkbetrieb wird durch periodisch gesendete Beacon-Nachrichten zeitgesteuert. Messaging ist in festen Steckplätzen möglich, die durch die Beacon-Nachricht definiert sind. Der Betrieb der Nachrichten wird vom PAN-Koordinator gesteuert. Die Beacon-Frames definieren die sogenannten Superframes, die die Datenübertragung in Zyklen aufteilen. Durch diesen Vorgang können den bestimmten Geräten garantierte Zeitrahmen gewährt werden und die Steuereinheit wird pausiert, wodurch Energie gespart wird. Der Superframe ist eine der wichtigsten Besonderheiten des Standards IEEE 802.15.4. Die Werte von macBeaconOrder und macSuperframeOrder beschreiben die Dauer der verschiedenen Teile des Superframes. Die Zeit, die der Koordinator benötigt, um die Beacon-Frames weiterzuleiten, wird durch den Wert von macBeaconOrder beschrieben. Die Beziehung zwischen macBeaconOrder (BO) und dem Beacon-Intervall (BI) kann in der folgenden Formel beschrieben werden ("IEEE Standard for Low-Rate Wireless Networks", 2016, pp. 57-58):

$$BI = aBaseSuperframeDuration \times 2^{BO}$$

In dieser Formel liegt der Wert von macBeaconOrder zwischen 0 und 14 (0 <= macBeaconOrder <= 14). Wenn die Variable macBeaconOrder auf 15 gesetzt ist, soll der Koordinator den Beacon-Frame nicht weiterleiten, außer wenn dies ausdrücklich angefordert wurde. Der Wert von macSuperframeOrder soll auch ignoriert werden, wenn macBeaconOrder den Wert 15 hat. Die Variable macSuperframeOrder legt fest, wie

lang der aktive Teil des Superframes sein soll. Die Beziehung zwischen macSuperframeOrder (SO) und der Superframe Dauer (superframe duration, SD) kann in der folgenden Formel beschrieben werden:

$$SD = aBaseSuperframeDuration \times 2^{SO}$$

Gleich wie bei der macBeaconOrder, liegt der Wert von macSuperframeOrder zwischen 0 und 14 (0 <= macSuperframeOrder <= 14). Wenn macSuperframeOrder den Wert 15 hat, soll der Superframe nach dem Beacon nicht mehr aktiv bleiben ("IEEE Standard for Low-Rate Wireless Networks", 2016, pp. 57-58). Die Beispielstruktur eines Superframes ist in der Abbildung x.y dargestellt. Das Beacon wird zu Beginn von Slot 0 ohne Verwendung von CSMA übertragen. Es werden zwei Teile, der aktive und der inaktive Teil unterschieden. In dem aktiven Teil eines Beacon-fähigen Netzwerks gibt es zwei Perioden in einem Beacon-Intervall, wobei die Geräte auf den Kanal zugreifen können. Während der Contention Acces Period (CAP) sendet jedes Gerät Daten an den Controller in der vom CSMA-CA-Algorithmus festgelegten Reihenfolge. Zu diesem Zeitpunkt konkurrieren die Geräte um den Kanal, da dieser für alle frei ist. Darauf folgt die Contention Free Period (CFP), welche die Position der garantierten Zeitschlitze (Guaranteed Time Slots, GTS) innerhalb des Superframes ist. Die Länge der CFP wird durch die Länge aller GTS bestimmt. Der Besitz eines GTS durch ein Gerät in der CFP-Periode hängt nicht mit der CAP-Periode zusammen, d.h. ein Gerät kann in beiden Perioden teilnehmen. Nur Geräte, die einem der verfügbaren Zeitfenster zugeordnet sind, können darauf zugreifen. Die Zeitrahmen werden vom PAN-Koordinator auf Anforderung der Geräte vergeben und können nur zur Kommunikation zwischen Koordinator und Gerät verwendet werden. Ein Gerät erhält nur dann ein GTS, wenn es noch verfügbar ist, d.h. ein bereits reservierter GTS kann nicht auf ein anderes Gerät übertragen werden. Jedes zugeordnete Gerät kann ein GTS zum Senden und Empfangen anfordern. Diese Informationen werden durch den PAN Koordinator gespeichert. Der Ort des Startslots, die Länge des GTS und die Adresse des Geräts werden ebenfalls durch den PAN gespeichert. Wie bereits erwähnt, muss jedes Gerät ein GTS belegen, um über dieses kommunizieren zu können (Eady, 2007, pp. 30-31).

Die Reservierung wird mit dem GTS Anforderungsbefehl (GTS request command) vorgenommen. Dieser Befehl bestimmt die erforderliche Länge und gibt an, ob das Gerät senden oder empfangen möchte bzw. die Anforderungen, die für die Anwendung gelten. Nach Erhalt dieses Anforderungsbefehls sendet der PAN-Koordinator einen Bestätigungsrahmen. Anschließend werden die verfügbaren Kapazitäten im Netzwerk und im aktuellen Superframe überprüft, um festzustellen, ob die erforderliche Kapazität vorhanden ist. Das Gerät bleibt nach Erhalt der vom PAN-Koordinator gesendeten Bestätigung mit dem Superframe synchron und wartet auf einen GTS-Frame. Wenn der Frame nicht im Superframe ankommt, sendet die MAC-Schicht eine Nachricht an die oberen Schichten, dass die GTS-Zuordnung fehlgeschlagen ist.

Wenn der Koordinator feststellt, dass die verfügbaren Ressourcen für das angeforderte GTS ausreichen, generiert er einen GTS-Frame. Dieser enthält die wesentlichen Informationen sowie die Adresse des anfordernden Geräts und teilt den oberen Schichten mit, dass eine neue GTS-Reservierung vorgenommen wurde. Falls erforderlich, reduziert der Koordinator die Dauer des CAP-Zeitraums, so dass das angeforderte GTS in den Rahmen passt. Allerdings kann die Dauer des CAP einen bestimmten vordefinierten Wert nicht unterschreiten. Weiters müssen die nicht besetzten GTS-Frames im CFP-Bereich vom Koordinator entfernt werden, um die CAP-Länge zu maximieren (Prasad, 2009, p. 163).

Wenn ein GTS nicht mehr benötigt wird, muss der GTS-Bereich freigegeben werden. Dies kann jederzeit durch den Koordinator oder das Gerät erfolgen, welches dies angefordert hat. Das Gerät schickt den vorher erwähnten „GTS request command" Befehl, in dem es die GTS-Daten beschreibt und deren Freigabe anfordert. Zu diesem Zeitpunkt ist der GTS-Speicherplatz freigegeben und kann nicht mehr für das Gerät verwendet werden. Dann wird ebenso eine Bestätigung vom Koordinator an das betroffene Gerät gesendet (Prasad, 2009, p. 163).

Wie in der unteren Abbildung (Abb. 3.2) ersichtlich ist, folgen die den Geräten zugewiesenen GTS am Ende des Superframes nach der CAP-Periode. Die Länge eines GTS kann sich über mehrere Superframe-Slots erstrecken. Es gibt keine Priorität, die Anforderungen werden in der Reihenfolge ihres Eingangs zugewiesen. Es ist möglich einem Gerät mehrere GTS-Slots zuzuweisen. Die Aufteilung des CFP-Gebiets ist nicht konstant, da die Länge und der Besitz der GTS variieren können.

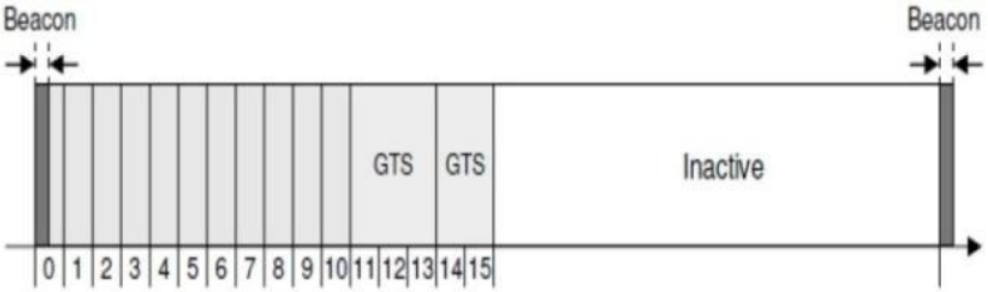


Abbildung 3.2: Beispiel für eine Superframe Struktur im Beacon-fähigen Modus.

Der Startslot jedes gebuchten GTS, seine Länge und ob gesendet oder empfangen werden soll, werden ebenfalls auf dem Gerät gespeichert. Wenn einem Gerät ein Empfangs-GTS zugewiesen ist, muss es den Empfänger nur für diese Zeit aktivieren. Dies ist eine weitere Energiesparoption im Protokoll. Darauf kann optional die inaktive Zeit folgen, die den Controllern die Möglichkeit gibt, bis zum Ende der inaktiven Zeit inaktiv zu sein. Während dieser Zeit darf der Koordinator nicht mit seinem PAN kommunizieren und kann in den Energiesparmodus wechseln. Im Nicht-Beacon-Modus findet keine Synchronisierung statt, d.h. jedes Netzwerkgerät kann jederzeit kommunizieren. Beacon-Frames werden auch hier verwendet, um die verfüg-

baren ZigBee-Netzwerke und Netzwerkgeräte einzuschalten und die Verbindung zu trennen. Im Nicht-Beacon-Modus kann jedes Gerät jederzeit senden. Dabei wird auch der CSMA-CA-Algorithmus verwendet, um Kollisionen zu vermeiden (MOHAN.K, Chandrasekara & K, 2014, p. 2155-2156).

3.4 Netzwerktopologien

ZigBee unterscheidet grundsätzlich drei verschiedene Arten der Netzwerktopologie: Star (Stern) Topologie, Peer-to-peer Topologie und Cluster Tree (Cluster-Baum). Die ersten beiden Topologien sind durch den IEEE Standard definiert. Das Cluster Tree-Layout ist eine ZigBee-Spezifikation, eine Kombination aus Netz- und Sterntopologie, die die Vorteile beider Topologien kombiniert (Craig, 2004, pp. 4-6). In allen drei Fällen gibt es eine Zentraleinheit, einen PAN-Koordinator, der ein FFD-Gerät ist, bzw. Endgeräte, die sowohl FFD als auch RFD sein können. Bei den beiden letztgenannten Topologien kommen sogenannte Router vor, die wieder nur vom FFD-Typ sein können.

3.4.1 Star Topologie

Wie in der folgenden Abbildung (Abb. 3.3) dargestellt, umfasst diese Topologie eine zentrale Steuereinheit, den PAN-Koordinator und ein oder mehrere Endgeräte, wie zum Beispiel Sensoren. Die Rolle des PAN-Koordinators wird durch ein FFD eingenommen. Die Endgeräte können nur mit und durch diesen Koordinator kommunizieren, was bedeutet, dass alle Endgeräte direkt mit dem zentralen FFD verbunden sind (Rashid, 2015, p. 116). In jedem Fall muss der PAN-Koordinator ein Gerät sein, das aufgrund der im vorigen Kapitel (Abschnitt 3.3) erwähnten Operationen über ausreichende Rechenleistung verfügt und mit kontinuierlichem Strom versorgt wird. Wenn dieses zentrale Gerät ausfällt, kann das gesamte Netzwerk nicht mehr funktionieren. Die meisten anderen Geräte haben sehr einfache Aufgaben und sind die meiste Zeit inaktiv. Das Netzwerk basiert auf der ersten Aktivierung eines FFD. Dies wird der PAN-Koordinator sein und er wählt eine eindeutige Kennzeichnung (PAN-ID) aus, die noch nicht im Einsatz ist. Aus diesem Grund können mehrere Stern Topologie-Netzwerke in der gleichen Gegend betrieben werden. Da die Endgeräte hier RFD-Typen sein können und nur ein PAN-Koordinator, sprich FFD notwendig ist, ist dieses Netzwerk einfach und sicherlich am günstigsten auszubauen.

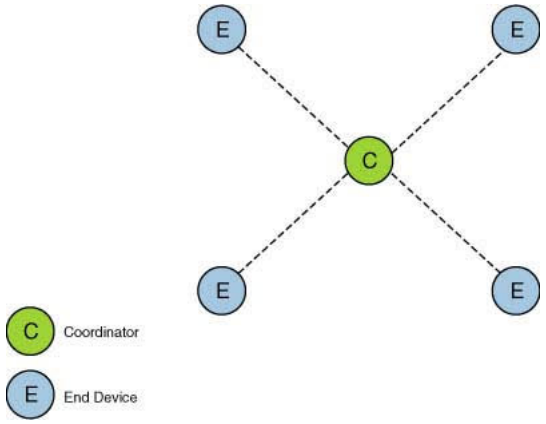


Abbildung 3.3: Stern Topologie eines ZigBee Netzwerkes.

3.4.2 Peer-to-peer Topologie

Bei der Mesh Topologie, auch Peer-to-peer Topologie genannt, spielt ein FFD die Rolle eines PAN-Koordinators, wie in Abbildung (Abb. 3.4) dargestellt ist. Im Gegensatz zu Stern Topologie, können die einzelnen Geräte miteinander kommunizieren, auch wenn sie nicht in direktem Kontakt stehen. Dies macht das Netzwerk betriebssicherer, denn wenn ein Pfad ausfällt, kann die Kommunikation über einen alternativen Pfad erfolgen. Nachrichten werden über einen oder mehrere sogenannte Router gesendet, was zu einem weiter verbreiteten Netzwerk führen kann, wodurch die Kommunikationsentfernung zwischen Geräten erhöht wird und die sogenannten toten Zonen, in denen kein Signal durchkommen kann, beseitigt werden können (Rashid, 2015, p. 117). Wie bei der Stern Topologie fungiert das erste FFD, welches im Netzwerk aktiviert wird, als PAN-Koordinator. Alle Geräte, die in diesem Netzwerk als Router bezeichnet werden, sind auch FFDs, da ausschließlich diese Geräte Nachrichten übermitteln können. RFDs können jedoch als Endgeräte vorkommen.

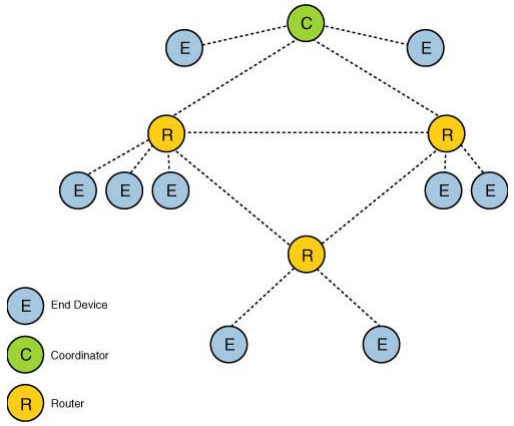


Abbildung 3.4: Peer-to-peer Topologie eines ZigBee Netzwerkes.

3.4.3 Cluster Tree Topologie

Die Cluster-Baum-Topologie (Abb. 3.5) wird von ZigBee als Peer-to-Peer-Unterart behandelt und ist eigentlich eine Kombination der beiden vorigen Topologien. Es wird von FFD-Geräten eine Baumstruktur gebildet, welche aus zwei oder mehr Ebenen bestehen kann und deren Wurzelelement der PAN-Koordinator ist. Die Endgeräte können direkt mit dem Koordinator oder wie bei der Mesh Topologie mit den einzelnen Router Elementen verknüpft sein. FFD-zentrierte Router-Knoten bilden mit Endgeräten einzelne Cluster, die mit dem Koordinator verbunden sind. Dieser Typ teilt mit der Stern-Topologie den Nachteil, dass bei einem Ausfall der Zentraleinheit das gesamte Netzwerk möglicherweise nicht mehr funktionsfähig ist. Zu seinen Vorteilen gehört, wie bei der Peer-to-Peer-Topologie, die größere Kommunikationsentfernung zwischen Geräten (Elahi & Gschwender, 2009).

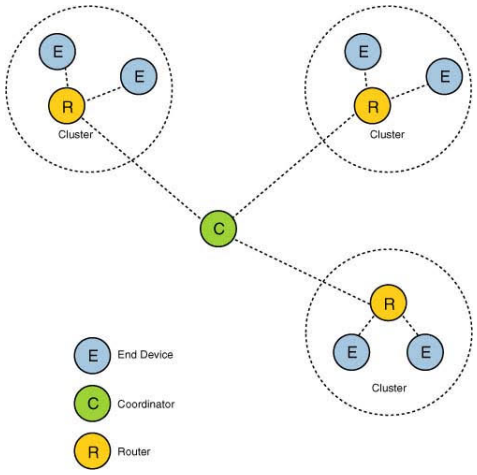


Abbildung 3.5: Cluster Topologie eines ZigBee Netzwerkes.

3.5 Sicherheit

Während der Kommunikation gesendete Daten können sensible Informationen enthalten, die vor dem Versand der Daten verschlüsselt werden sollen. Um Befehle, Beacon-Nachrichten und Bestätigungen zu verschlüsseln verwendet ZigBee auf der MAC-Ebene einen kryptografischen Algorithmus nämlich die 128-Bit-Version des Advanced Encryption Standards (AES). Bei diesem Algorithmus werden die Daten in kleinere 128-Bit-Pakete aufgeteilt und dann mit einem symmetrischen Schlüssel verschlüsselt, d.h. es wird derselbe Schlüssel zum Ver- und Entschlüsseln der Informationen verwendet. Obwohl AES drei Schlüssellängen, 128, 192 und 256-Bit anbietet, werden beim Standard IEEE 802.15.4 Schlüssellänge 128-Bit wegen des niedrigen Ressourcenbedarfs verwendet. Obwohl die Sicherheitsverarbeitung auf MAC-Ebene erfolgt, generieren die oberen Ebenen die Sicherheitsschlüssel und legen die

zu verwendenden Sicherheitsstufen fest. Der geheime Schlüssel wird nur mit dem Absender und dem Empfänger geteilt. Wenn die MAC-Ebene ein verschlüsseltes Paket sendet oder empfängt, wird die Ziel- oder die Quelladresse bzw. der zugewiesene Schlüssel überprüft und dieser Schlüssel verwendet, um das Paket zu entschlüsseln. Die Verwendung der Verschlüsselung wird durch ein Bit in den Frames angezeigt (Krauße & Konrad, 2014, pp. 152-159).

4 Systemimplementation

In diesem Kapitel soll gezeigt werden, wie man mit einem selbst eingerichteten Koordinator, ein ZigBee-Netzwerk aufbauen kann. In dem Projekt der Masterarbeit wurden zwei Hausautomationsplattformen ausgewählt. Die Installation, Konfiguration und die Verbindung von Geräten, die nach bestimmten Kriterien selektiert wurden, sollen ebenfalls beschrieben werden.

4.1 Smart Home Geräte

Die große Auswahl der Smart-Geräte bietet verschiedene Nutzungsmöglichkeiten. Viele intelligente Geräte, die im Allgemeinen mit WLAN oder Bluetooth kommunizieren, können einzeln verwendet werden. Diese Geräte verfügen über eine eigene Benutzeroberfläche, z. B. eine Anwendung für Handy, wodurch sie überwacht und gesteuert werden können. Der Vorteil dieser Anwendungen ist, dass sie im Betrieb einfach zu bedienen sind, da der User/die Userin nur eine App installieren muss und das Gerät sodann einsatzbereit ist. Diese Apps sind anwenderfreundlich, da ihre Benutzeroberfläche und Funktionen für ein bestimmtes Gerät optimiert sind. Wenn der Kundedie Kundin nur ein intelligentes Smart Home Gerät, beispielsweise eine Heizungsanlage betreiben möchte, ist diese Art und Weise für die Aufgabe gut geeignet. Wenn mehrere intelligente Geräte installiert und verwendet werden sollen, besteht der große Nachteil dieser Lösung darin, dass für jedes einzelne Gerät jeweils eine eigene Anwendung installiert werden muss. Dies behindert sowohl die Benutzerfreundlichkeit als auch die Nutzung eines Smart Homes, weil für die verschiedenen Geräten unterschiedliche Steuerungsmöglichkeiten wie zum Beispiel andere Applikationen gebraucht werden. Die unzähligen Applikationen erfordern übrigens viel Speicherplatz. Ein weiterer Nachteil ist, dass die einzelnen Geräte unabhängig voneinander arbeiten und daher nicht miteinander kommunizieren können. Dadurch können die Geräte nicht zusammenarbeiten, wodurch die Effizienz eines Smart Homes verloren geht. Einige Hersteller entwickeln zu diesem Zweck einen Hub, über den Smart-Geräte verbindet werden können. Heutzutage gibt es Geräte, die nur mit einem Hub funktionieren, weshalb man eine vom Hersteller gelieferte Zentraleinheit zusammen mit einem intelligenten Gerät kaufen muss. ZigBee Geräte gehören auch zu dieser Gruppe, da wie es in dem vorherigen Abschnitt (Abschn. 3.4) dargestellt ist, sie in allen Fällen einen Koordinator benötigen, der die einzelnen Endgeräte steuert

und verwaltet. Bei der Integration zwischen Geräten besteht jedoch ein großes Problem, da in meisten Fällen die Produkte des Herstellers nur den eigenen Koordinator unterstützen. Dies kann die große Auswahl an ZigBee-Geräten erheblich einschränken.

4.2 Endgeräte

Zur Betreibung eines ZigBee-Systems werden Endgeräte benötigt. Diese Geräte führen die eigentlichen Messaufgaben aus. Das Routing ist völlig unabhängig von ihnen, sodass der Endpunkt die meiste Zeit im Ruhemodus verbringen, in Intervallen messen und dann die Messdaten an den Router bzw. direkt an den Koordinator senden oder gegebenenfalls empfangen kann. Dadurch wird eine sehr geringe Einschaltdauer erreicht, die den Durchschnittsverbrauch des Endgeräts niedrig hält. Da die Automatisierung eines gesamten Hauses nicht Gegenstand der Masterarbeit ist, wurden drei Geräte ausgewählt, um die Funktionsweise des Systems zu veranschaulichen. Die Auswahl dieser Geräte basierte auf den folgenden Überlegungen:

- Ein allgemeines, praktisches Gerät, welches in jedem Haushalt eingesetzt werden kann
- Das Gerät kommuniziert über das ZigBee-Kommunikationsprotokoll
- Die Geräte werden durch die Software Zigbee2MQTT unterstützt
- Die Installation ist einfach durchführbar
- Beispiel für bidirektionalen Betrieb (Nachricht empfangen, Nachricht senden)

Die ausgewählten Endgeräte sind keine Vollfunktionsgeräte und dadurch keine Router, dies bedeutet sie sind nicht fähig, die von anderen Endgeräten gesendeten Nachrichten weiterzuleiten.

4.2.1 Temperatur- und Luftfeuchtigkeitssensor

Um sicherzustellen, dass die richtige Innentemperatur für das Zuhause herrscht, ist es praktisch, mindestens einen Thermometer zu Hause zu haben. Für den Zweck des Masterarbeitsprojekts wurde ein Gerät (Modell: WSDCGQ01LM) von einer Tochtergesellschaft von Xiaomi namens MIJIA ausgewählt, welches für die Langzeitüberwachung des Raumklimas entwickelt wurde. Zusätzlich zur Messung der Temperatur kann es auch die Luftfeuchtigkeit der Wohnung messen. Das Gerät hat kein

Display, sodass die gemessenen Daten nur durch eine Anwendung überprüfbar sind, welches dagegen zu einer längeren Akkulaufzeit führt. Das Temperatur- und Luftfeuchtigkeitsmessgerät ist ein 60x60x30mm großes kreisförmiges Gerät, das mit einer CR2032-Batterie betrieben wird. Der Sensor kann überall mit einem doppelseitigen Klebstoff befestigt werden und verwendet das ZigBee-Protokoll zur Kommunikation. Der Messbereich des Thermometers liegt zwischen -20 und +60 Grad Celsius mit einer Genauigkeit von plus/minus 0,3 Grad Celsius. Die Luftfeuchtigkeitsmessung kann zwischen Null und 100 liegen und mit einer Genauigkeit von drei Prozent erkannt werden.

Abbildung 4.1: Xiaomi Temperatur- und Luftfeuchtigkeitssensor.

4.2.2 Tür-, Fenstersensor

Das zweite Gerät des gleichen MIJIA Herstellers ist ein Sensor, mit dem man die Sicherheit des Zuhauses kostengünstig und einfach verbessern kann. Dieser kompakte, einfach zu installierende Tür- oder Fenstersensor (Modell: MCCGQ01LM) funktioniert mit jeder Tür oder jedem Fenster. Das Gerät besteht aus zwei kleinen Teilen, einem Sensor (41x21x11mm) und einem Magneten (26 x 10 x 9mm). Die Installation der Geräte funktioniert wie folgt: es ist empfohlen, den Sensor an einem festen Ort, d.h. am Tür- oder Fensterrahmen, und den Magneten an der Tür oder am Fenster anzubringen. Diese Sensoren sind genauso einfach zu installieren wie der Temperatur- und Feuchtigkeitssensor, da sie eine selbstklebende Rückseite haben. Der Sensor erkennt, wenn eine Tür oder ein Fenster geöffnet bzw. geschlossen wird, und kann das sofort über das ZigBee-Protokoll kommunizieren. Die Erkennung funktioniert nach einem magnetischen Prinzip. Wenn der Benutzerdie Benutzerin das von dem anderen Gerät erzeugte Magnetfeld, das gemäß der Dokumentation des Herstellers 22 Milli-

meter beträgt, nicht wahrnimmt, wird davon ausgegangen, dass die Tür bzw. das Fenster offen ist. Die Stromversorgung erfolgt hier über eine kleine CR1632-Batterie.

Abbildung 4.2: Xiaomi Tür-, Fenstersensor.

4.2.3 Rauchmelder

In ganz Österreich gibt es jährlich insgesamt mehr als 25000 Brandschäden, wobei 70 Prozent davon auf den Privatbereich entfielen. Obwohl es sich bei der Hälfte der Brandschäden um Kleinschäden handelt, beträgt die Brandschadenssumme für ganz Österreich jedoch mehr als 218 Millionen Euro. In den Brandursachen zählen zu einem relevanten Teil Blitzschlag, elektrische Geräte und Wärmegeräte. Zusätzlich zu den objektiven Werten erfordern Brände etwa 50 bis 100 Todesfälle bzw. 300 Schwerverletzungen pro Jahr. Die meisten Todesfälle passieren durch Brände und werden durch Rauchvergiftungen verursacht, aber Hunderte oder mehr Menschen werden jedes Jahr mit wenigen oder schweren Rauchvergiftungen ins Krankenhaus eingeliefert. Eine frühzeitige Erkennung kann dazu beitragen, sowohl die Verletzungen und den Todeszahl als auch die Brandfälle zu verringern. Bild- und Audiogeräte, wie Rauch- oder Gasmelder, stehen seit langem für den Brandschutz zur Verfügung. Diese Geräte können aber nur die Personen, die in der Nähe des Gefahrenbereichs sind, alarmieren. Der mit dem Reddot Award 2017 ausgezeichnete Rauchmelder von Heiman (HS1SA-N) verfügt nicht nur über eine Sirene, sondern auch über eine ZigBee-Netzwerktechnologie mit extrem geringem Stromverbrauch, um die Benutzerin/den Benutzer auf Rauch aufmerksam zu machen. Die Rückenplatte des Geräts sollte zusätzlich zu dem gewohnten Klebstoff in die Deckenwand eingeschraubt werden. Der Sensor kann danach auf der Rückplatte gefestigt werden, die einen einfachen Batteriewechsel ermöglicht. Das 60x51x43mm große Gerät ist in der Lage, eine umfassende

360-Grad-Überwachung mit hoher Empfindlichkeit durchzuführen. Es ist auch mit einer Statusanzeige ausgestattet, die gleichzeitig ein Testknopf ist. Die Alarmlautstärke in Echtzeit beträgt 85 dB. Das Gerät wird von einem CR123A-Akku gespeist, der laut Hersteller für den Betrieb von mehr als drei Jahren ausreicht. Wenn der Akkuladestand niedrig ist, sendet das Gerät eine Warnung.

Abbildung 4.3: Heiman Rauchmelder.

4.2.4 LED-Streifenlicht

Um die Steuerungsmöglichkeit der Geräte zu zeigen, soll zusätzlich ein LED-Streifenlicht von der Firma Gledopto an das ZigBee-Netzwerk angeschlossen werden. Über ZigBee soll möglich sein, eine Nachricht an diesem Gerät zu senden, in der die Lampe ein- und ausgeschaltet und die Helligkeit bzw. Farbe angepasst werden können. Das Gerät besteht aus einem kleinen Steuerelement, welches über einen USB-Anschluss mit Strom versorgt wird, und aus einem zwei Meter langen RGB LED-Streifenlicht, wobei 16 Millionen Farben eingestellt werden können. Es enthält auch drei Anschlüsse, womit das zwei Meter lange Streifenlicht in quadratischer Form zusammengefügt werden kann. In dieser Form ist dieses Stimmungslicht für Fernseher, Computer oder andere Geräte mit USB-Eingängen perfekt geeignet.

4.3 ZigBee-Koordinator

Wie bereits früher erwähnt, benötigen ZigBee-Geräte immer genau einen Koordinator, der ein FFD Gerät ist. Wenn der Benutzer/die Benutzerin weitere intelligente

Abbildung 4.4: Gledopto LED-Streifenlicht.

Geräte von verschiedenen Anbieters in seinem/ihrem Smart Home betreiben möchte, da der Hersteller in vielen Fällen nur bestimmte Produkte, zum Beispiel intelligente Glühbirnen verkauft, ist der User/die Userin gezwungen, die Zentralgeräte der Hersteller zu kaufen. Da die oben genannten Sensoren von drei verschiedenen Herstellern stammen und diese Hersteller nur die Verbindung eigener Geräte unterstützen, ist es erforderlich, eine separate Zentraleinheit von den beiden Herstellern zu erwerben, um die Geräte betreiben zu können. Die Kosten der einzelnen Zentraleinheiten sind unterschiedlich, in diesem Fall jedoch würden die Gesamtkosten der drei Zentraleinheiten ca. 170 € betragen. Selbstverständlich muss neben den Zentraleinheiten auch die von den Herstellern entwickelten Anwendungen installiert bzw. konfiguriert werden. Dies bedeutet, dass der Temperatur- und Feuchtigkeitssensor zwar mit einem gemeinsamen Xiaomi Hub und einer Telefonanwendung betrieben werden , der Hub des Rauchwarnmelders jedoch separat von der Firma Heiman und der Hub des LED- Streifenlicht von der Firma Philips, da es nur mit Philips Hub kompatibel ist, erworben bzw. die dazu gehörende Software verwendet werden müssen. Zusätzlich benötigt jeder Koordinator eine eigene Stromversorgung bzw. eine Verbindung zu dem Netzwerkrouter. Dies hat auch das Problem, dass beispielsweise während der Nutzung des Xiaomi Hubs die Benutzerdaten über den Router an die China Cloud gesendet werden, wofür in diesem Fall keine Möglichkeit besteht, die Datenübermittlung zu verhindern. Dank eines von Texas Instruments entwickelten Chips ist es nicht mehr unmöglich, ein eigenes ZigBee-Netzwerk mit einer individuellen Zentraleinheit unabhängig von jedem Hersteller und deren Cloud Systeme aufzubauen. Der sogenannte CC2531 ist eine USB-basierende echte System-on-Chip-Lösung (SoC), die für IEEE 802.15.4- und ZigBee-Anwendungen entwickelt wurde. Sie arbeitet auf einer Frequenz zwischen 2,405 und 2,448 GHz und ist auf einer Übertragungsgeschwindigkeit von 250 Kbs fähig. Das Gerät hat einen Stromverbrauch von weniger als 20mA. Die große Anzahl der Koordinatoren von den verschiedenen Herstellern können durch den geflashten CC2531 USB-Stick, und an einem Rech-

ner laufenden Software, namens Zigbee2MQTT ersetzt werden. Nach dem Aufbau des Netzwerks müssen die Benutzerinnendie Benutzer lediglich das ZigBee-Endgerät beim Hersteller kaufen, wodurch man viel Geld sparen kann. Der CC2531 USB-Stick kann als Koordinator betrieben werden, wenn er mit der richtigen bzw. idealerweise mit der neuesten Firmware geflasht ist. Texas Instruments stellt ein Gerät, nämlich den CC-Debugger bereit, um den Stick flashen zu können. Um die Ports des CC2531 mit dem CC-Debugger verbinden zu können, ist weiters ein sogenannten Downloader Kabel für CC2531 notwendig. Obwohl man einen vorgeflashten CC2531 USB kaufen kann, ist es aus mehreren Gründen empfohlen, die zum Flashen notwendige Hardware zu kaufen, um die Firmware manuell flashen zu können. Wenn eine neue Firmware auf den Markt kommt oder im Fall einer Funktionsstörung, stehen die Werkzeuge zur Verfügung, um das Problem zu beheben. Da das CC2531 USB nicht nur als Netzwerkkoordinator, sondern auch als Router verwendbar ist, ist man mit dem Besitz dieser Werkzeuge jederzeit fähig, die auch zu dem Router gehörende Firmware zu installieren. Der Preisunterschied zwischen beiden Möglichkeiten ist sehr gering. Der Preis für einen vorgeflashten CC2531 liegt ungefähr bei 14 Euro. Die vorher erwähnten Werkzeuge einschließlich dem ungeflashten USB-Sniffer haben einen Gesamtwert von 16 Euro. Bevor der CC2531 als Koordinator benutzt werden kann, muss die richtige Firmware mithilfe der vorher erwähnten Hardware darauf geflasht werden. Um den CC Debugger ins Laufen bringen zu können, muss der Debugger Driver manuell auf dem Rechner installiert werden. Der CC-Debugger-Treiber ist von der Texas Instrument-Website herunterzuladen. Die ZIP-Datei enthält eine 32- und eine 64-Bit Exe-Datei, wodurch die Installation des Treibers vereinfacht wird. Die richtige Bit-Version hängt vom Prozessortyp des installierten Windows-Betriebssystems der Benutzerindes Benutzers ab. Wenn die 32-Bit-Version installiert ist, kann nur die 32-Bit Version des CC-Debugger-Treibers ausgeführt werden. Obwohl für die 64-Bit-Version der 64-Bit-Treiber empfohlen ist, können beide Versionen installiert und ausgeführt werden. Nach der Installation über die exe-Datei muss sichergestellt werden, dass der Treiber erfolgreich installiert wurde, auch im Fall, dass die Installation ohne Fehler durchgelaufen ist. Alle installierten Treiber sind unter Windows im Geräte-Management zu finden. Unter dem Tab Cebal controlled devices soll der neue Treiber, nämlich CC Debugger auftauchen. Wenn das nicht der Fall ist, bedeutet dies, dass die Installation nicht erfolgreich war und der Treiber manuell unter dem Menüpunkt Actions – Add legacy hardware installiert werden muss. Die 32-Bit und 64-Bit Versionen der Treiber-Dateien befinden sich mit den Namen cebal2_x32.inf bzw. cebal2_x64.inf in der zuvor heruntergeladenen Zip-Datei. Nach der Auswahl der zu dem Betriebssystem passenden Version wird der Treiber installiert und er ist nun wie auf der unteren Abbildung (Abschnitt 4.3) im Geräte-Manager ersichtlich.

Der nächste Schritt besteht darin, die zum Flashen gekauften Geräte wie in der Abbildung (Abschnitt 4.3) gezeigt, miteinander zu verbinden und den CC2531-USB-Sniffer bzw. den CC-Debugger jeweils an den USB-Port des Computers anzuschließen. Der Debugger startet sofort den Geräteerkennungsprozess und sucht nach dem Gerät. Wenn kein Gerät erkannt wird, leuchtet die LED-Lampe mit roter Farbe auf und so-

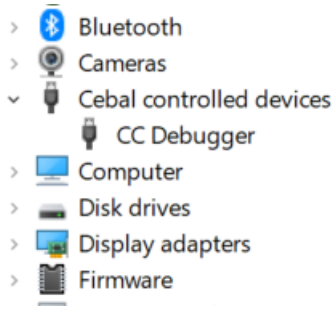


Abbildung 4.5: Überprüfen der korrekten Treiberinstallation im Geräte-Manager.

mitmuss der reset-Knopf am CC Debugger gedrückt werden. Wenn das Gerät richtig erkannt ist, leuchtet das LED-Licht grün auf. Die Firmware für den Koordinator, welches auf dem CC2531 geflasht werden muss, wurde von Koen Kanters entwickelt und die aktuelle Version vom 08.06.2019 (Stand 20.01.2020) steht im Github zur Verfügung. Dieses kann auf einem Windows Operationssystem mithilfe des SmartRF Flash-Programmierers v.1 von Texas Instruments auf dem CC2531 USB-Stick installiert werden. Die heruntergeladene Zip-Datei muss entpackt und anschließend die Hex-Datei, die sie enthält, unter Flash image des SmartRF Flash Programmer ausgewählt werden. Dann muss nur noch auf den Knopf Perform actions geklickt werden. Das File wird auf USB installiert oder wird auch überprüft. Das Firmware-image wird automatisch am CC2531 installiert und auch kontrolliert. Die Installation dauert abhängig von dem Computer ungefähr eine halbe Minute, dann ist die Rückmeldung von SmartRF ersichtlich, dass die Firmware installiert, geprüft und einsatzbereit ist.

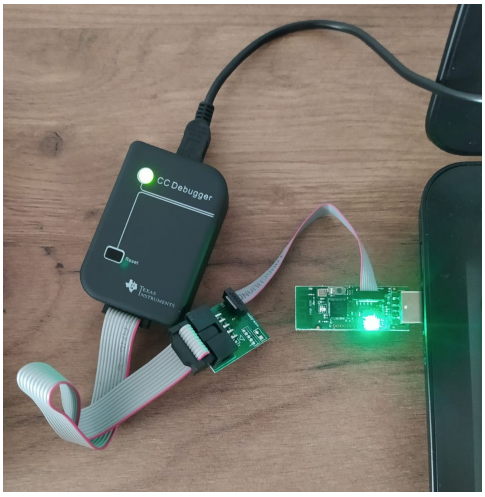


Abbildung 4.6: CC2531 und CC-Debugger verbinden.

4.4 ZigBee-Router

Mit den oben genannten Tools ist es möglich, ein ZigBee-Netzwerk in Sterntopologie aufzubauen. Ein Nachteil des Systems besteht darin, dass der CC2531 USB-Sniffer eine Beschränkung auf 20 Geräte aufweist, d.h. es können 20 Geräte direkt mit dem Koordinator verbunden sein. Eine weitere Behinderung ist, dass er nur in einer Reichweite von 30 Metern erhältlich ist. In Sterntopologie-Netzwerken wird die Reichweite des Koordinators durch ZigBee-Router erweitert, wodurch eine Mesh- oder Cluster-Topologie erzeugt werden kann. Wie bereits im vorherigen Abschnitt (Abschn. 3.4) erwähnt, ist ein Router ein solches Vollfunktionsgerät (FFD), das als Vermittler für die Weiterleitung von Steuerbefehlen und Daten der Endgeräte außerhalb der Reichweite des Koordinators über das Netzwerk dient. Außerdem sind ZigBee-Router, wie es für Mesh-Netzwerke üblich ist, dafür zuständig, neue Knoten mit dem Netzwerk zu verbinden. Um einen Router in ein ZigBee-Netzwerk zu integrieren, muss zunächst ein Netzwerk erstellt werden, in dem sich alle Knoten im Umkreis des Koordinators befinden. Sobald dieses Netzwerk eingerichtet ist, ist es möglich, das Netzwerk mit einem oder mehreren Routern zu erweitern. Die Endgeräte können sich dann direkt mit dem Router verbinden und über den Router Daten an den Koordinator senden und vom Koordinator gesendete Nachrichten empfangen.

Alle Geräte, die als Router fungieren, benötigen eine konstante Stromversorgung, da sie, wie beim Koordinator immer verfügbar sein sollten, jedoch ihre Ausfallzeiten nicht so ein großes Problem darstellen wie bei dem Fall eines Koordinators. Als Router können beispielsweise Glühlampen oder sogar ein CC2531-USB-Stick fungieren. Aus diesem Grund werden sie auch als Endgeräte eingesetzt. Wenn man den CC2531-USB-Stick als Router nutzen will, muss die Router-Firmware, die ebenfalls von Github-Benutzer Koen Kanters erstellt wurde, auf den CC2531 geflasht und an einen beliebigen USB-Port als Stromquelle angeschlossen werden. In diesem Fall können 21 zusätzliche Geräte an den als Router fungierenden CC2531 angeschlossen werden. Router sind sofort nach dem Beitritt zum Koordinator einsatzbereit. Neben der Anzahl der angeschlossenen Endgeräte wird das ZigBee-Netzwerkangebot erweitert. Für die Erweiterung der Reichweite besteht weiterhin die Möglichkeit, eine auf dem CC2531 USB-Stick montierbare Antenne mitzukaufen oder eine andere Lösung von Texas Instruments, namens CC2530 zu verwenden, welcher über eine größere Reichweite verfügt. Dies ist jedoch kein USB-Gerät und kann nur über GPIO Pins angeschlossen werden. Da während der Masterarbeit nur drei Geräte mit dem Koordinator verbunden werden und alle vier Geräte inklusive des Koordinators im gleichen Raum eingesetzt werden, ist die Nutzung eines Routers unnötig. Aus diesem Grund wird er nicht im Projekt der Masterarbeit eingebaut.

4.5 Raspberry Pi

Um den CC2531 USB-Sniffer richtig als ZigBee Koordinator verwenden zu können, ist weiterhin eine Software notwendig. Zigbee2MQTT ist eine Software, entwickelt von Koen Kanters, wodurch die Nachrichten von ZigBee-Geräten in MQTT umgewandelt werden, sodass man neue Geräte sogar von verschiedenen Herstellern mit dem CC2531 USB-Sniffer verbinden und steuern kann. Zigbee2MQTT mit dem CC2531 USB-Sniffer Zigbee2MQTT überbrückt Nachrichten von Zigbee-Geräten am MQTT, wodurch die Übertragung von Sensormessdaten, die Steuerung von Geräten wie Glühbirnen oder Schaltern und der Gerätestatus ermöglicht werden. Die ZigBee2-MQTT-Software kann sowohl unter Linux als auch Windows ausgeführt werden. Wie vorher erwähnt wurde, sind die Hauptkriterien für ein zentrales Gerät, um der Aufgabe eines PAN-Koordinators zu erfüllen, die entsprechende Performanz bzw. unterbrechungsfreie Stromversorgung, d.h. es sollte vorzugsweise nicht mit einer Batterie betrieben werden. Für diesen Zweck ist jede Version des Raspberry Pi Mini-Computers perfekt geeignet (siehe Abschnitt 4.5). Der Raspberry Pi ist ein kompakter Computer im Scheckkartenformat mit geringem Stromverbrauch, der für den Einsatz in Elektronik- und Hobbyprojekten entwickelt wurde. Aus diesem Grund wurde auch der Preis niedrig gehalten. Der Preis des 3 B+ Modells liegt ungefähr bei 35 Euro. Bei Raspberry Pi gibt es kein BIOS und keineStartreihenfolge, das Betriebssystem wird direkt von einer microSD-Karte geladen. Dafür ist eine MicroSD-Speicherkarte von mindestens 8GB zu empfehlen. Darüber hinaus ist eine unterbrechungsfreie Stromversorgung erforderlich, die über ein 2,5A starkes Micro-USB-Netzteil versorgt werden kann. Da ein Raspberry Pi nicht mit einem externen Monitor und Eingabegeräten verbunden werden muss, kann man sowohl Platz als auch Energie sparen. In diesem Projekt wird Raspberry Pi 3 Model B+ verwendet, welcher mit einer ARM Cortex-A53-CPU ausgestattet ist, bei der es sich um einen 64-Bit-1,4-GHz-Anwendungsprozessor handelt. Für die Netzwerkverbindung stehen ein zweikanaliges 802.11ac-WLAN und Bluetooth 4.2 sowie ein kabelgebundener Ethernet-Anschluss zur Verfügung. Diese Version verfügt über vier USB-Anschlüsse, sodass jedes Peripheriegerät wie eine Maus oder eine Tastatur angeschlossen werden können. Ein weiterer Eingangsanschluss sind HDMI und ein 3.5mm Audio Anschlusskabel, über die ein Monitor und ein Lautsprecher verbunden werden können. Der Raspberry Pi läuft auf einem Linux-basierten Betriebssystem, welches gegenüber anderen Betriebssystemen viele Vorteile bietet. Linux-basierte Betriebssysteme sind in meisten Fällen kostenlos, Open Source und sehr sicher, da sie sehr schwer zu infizieren sind, wodurch das Linux-basierte System weniger gefährlichen Bedrohungen ausgesetzt ist. Dies bedeutet auch, dass kein Antivirus und keine Firewall im Falle eines Linux-basierten Systems benötigt sind. Diese Systeme sind äußerst stabil und können viele Jahre problemlos ausgeführt werden, ohne dass der Computer ausgeschaltet werden muss. Dies ist eine unverzichtbare Eigenschaft für einen ZigBee-Koordinator. Sie funktionieren gut auf ressourcenarmen Computern wie auf dem Raspberry Pi, da diese Systeme deutlich weniger Ressourcen benötigen und dadurch die darauf laufenden Programme schneller geladen und ausgeführt werden können. Die als letzte

zu erwähnende Eigenschaft ist die schnelle und einfache Aktualisierung. Während der Aktualisierung unter Linux werden nicht nur das System, sondern auch alle installierten Anwendungen aktualisiert.

Abbildung 4.7: Raspberry Pi 3 B+ mit CC2531 USB-Stick.

4.6 Hausautomatisierungsplattformen

Da die von den Herstellern entwickelten Anwendungen nur mit ihrer eigenen Zentraleinheit kompatibel sind, kann die im Projekt verwendete individuelle Zentraleinheit nicht mit der Anwendung der Hersteller verwendet werden. Es werden für Hausautomatisierungssysteme viele kostenlose und kostenpflichtige Plattformen geboten. Das Ziel der Entwicklung dieser Plattform war es, eine allgemeine zentrale Steuerungsschnittstelle für die Erfassung von Daten von verschiedenen Sensortypen und die Steuerung verschiedener Geräte bereitzustellen. Aus diesem Grund kann solch eine Plattform zur Temperaturerfassung, für Türöffnungssensoren, Klingeln, Sicherheitsvorrichtungen, Wettersensoren, Lichtschalter oder zur Verarbeitung von analogen oder digitalen Eingangssignalen verwendet werden. Das Programm Zigbee2MQTT kann theoretisch auf jeder Hausautomatisierungsplattform integriert werden, weil es das Protokoll MQTT verwendet. Die offizielle Website der Software bevorzugt jedoch zwei Open Source-Plattformen, nämlich Domoticz (Domoticz, 2020) und Home Assistant (Assistant, 2020). Abhängig von der zu integrierenden Plattform ist ein geeignetes Operationssystem notwendig. In den folgenden Unterkapiteln werden die einzelnen Operationssysteme vorgestellt und deren Installationsprozesse bzw. die Installation der Zigbee2MQTT-Software beschrieben.

4.6.1 Domoticz

Domoticz ist eine mit C++ entwickelte, kostenlose Open-Source-Hausautomationsplattform, die erstmals im Jahr 2012 veröffentlicht wurde. Domoticz ist eigentlich ein Webserver, auf den mittels Browser über eine HTML5-Oberfläche zugegriffen werden kann. Domoticz kann über die sogenannte YAML-Konfigurationsdatei konfiguriert werden. Die minimale Hardwareanforderung beträgt nur 256 MB RAM und 200 MB Speicherplatz, damit ist es zum Laufen auf Raspberry Pi geeignet. Sie wurde früher mithilfe einer Image-Datei installiert, damit sie selbstständig ausgeführt werden konnte. Diese Methode wird jedoch seit Jahren nicht mehr unterstützt und kann nur durch Ausführen eines Installationsprogramms unter Linux oder Windows installiert werden.

4.6.2 Hass.io - Home Assistant

Die Home Assistant-Software wurde erstmals im Jahr 2013 veröffentlicht. Sie ist in Python geschrieben und kann auf jedem Betriebssystem verwendet werden, auf dem Python ausgeführt wird. Sie basiert, wie Domoticz auf einer Konfigurationsdatei im YAML-Format. Für die Ausführung von Home Assistant sind lt. Dokumentation 32 GB Speicherplatz und 1 GB RAM erforderlich. Dies ist deutlich mehr als beim Domoticz, ist Raspberry Pi Version 3 jedoch auch für diesen Zweck geeignet. Für Raspberry Pi ist eine Image-Datei mit dem Namen Hass.io verfügbar. Hierbei handelt es sich um eine Methode zum Installieren und Ausführen von Home Assistant. Anders als bei Domoticz muss Home Assistant aus diesem Grund nicht auf einem externen Betriebssystem installiert werden. Hass.io ist eine kostenlose Open-Source-Plattform, basiert auf ResinOS and Docker, die für direkte Raspberry-Pi optimiert ist. Der Hersteller stellt sofort einsatzbereite Add-Ons wie Mosquitto MQTT Broker, Google Assistant oder DynDNS zur Verfügung. Obwohl es möglich ist, Home Assistant auf einem eigenen Betriebssystem, wie Raspbian zu installieren, kann das System keine Erweiterungen (Add-ons) interpretieren.

4.7 Installation der Domoticz Hausautomationsplattform

Diese Sektion beschreibt, wie ein Benutzer/eine Benutzerin die Open Source Domoticz Hausautomationsplattform auf einem Raspberry Pi installiert und konfiguriert, es werden weiters die Funktionen mit den ausgewählten Sensoren angezeigt.

4.7.1 Vorbereitung des Raspbian Operationssystems

Da Domoticz in diesem Projekt auf einem Raspberry Pi laufen soll, muss zuerst das offizielle Betriebssystem, nämlich Raspbian installiert werden. Für den Server ist keine grafische Benutzeroberfläche erforderlich, da nur ein Webserver für den Zugriff auf die Domoticz-Website erforderlich ist. Daher ist die Verwendung von Raspbian Lite ausreichend, welche keine grafische Oberfläche, sondern nur Befehlszeilenschnittstelle (command line interface, CLF) anbietet. Dieses Betriebssystem ist kleiner als eine vollständige grafische Desktop-Oberfläche, benötigt weniger Speicherplatz und kann schneller heruntergeladen werden. Es gibt zwei Wege, Raspbian Lite auf einer MicroSD-Karte zu installieren. In beiden Fällen sind eine MicroSD-Karte, ein Kartenleser und ein Windows- oder Linux-Rechner, auf dem die Installation der SD-Karte ausgeführt werden kann, notwendig. Wenn ein externer Monitor bzw. Maus und Tastatur zur Verfügung steht, bietet der sogenannte NOOBS (New Out Of Box Software) eine einfachere, grafische Installation, die für alle Anfänger geeignet ist. Dazu ist die NOOBS.zip-Datei von der Download-Seite des Raspberry Pi nötig. Die Datei muss auf eine formatierte MicroSD-Karte mit 8 GB oder mehr extrahiert werden. Nach dem ersten Hochfahren von Pi kann ausgewählt werden, welche OS installiert werden soll. Auf die Installation folgt ein Neustart, nachdem das ausgewählte Betriebssystem gestartet wird. Die andere Möglichkeit ist die, bei der die Image-Datei von Raspbian Lite direkt auf der MicroSD-Karte installiert wird. Diese Methode erfordert keinen Anschluss von Eingangs- und Ausgangsperipheriegeräten. Die Image-Datei kann von der offiziellen Raspberry Pi-Website heruntergeladen werden. Nach dem Formatieren der SD-Karte kann die Bilddatei mit dem Programm balenaEtcher auf die Karte geschrieben werden. Nach dem Flashen der SD-Karte ist der Raspberry Pi einsatzbereit.

Raspbian Lite kann nur über die Befehlszeile gesteuert werden. Es gibt zwei Möglichkeiten auf diese Befehlszeile zuzugreifen. Die erste Möglichkeit besteht darin, Eingabe- und Ausgabeeinheiten wie eine Tastatur und einen Monitor anzuschließen. Die andere Option ist, eine Verbindung über SSH zu Raspberry Pi vomPC oder Laptop aus herzustellen. Nach der erfolgreichen Einrichtung ist es möglich, auf das Gerät über die IP-Adresse und den SSH Port zuzugreifen. Dazu muss eine neue leere Datei mit dem Namen SSH im Boot-Ordner der installierten SD-Karte erstellt werden, die den Zugriff über SSH aktiviert. SSH bedeutet Secure Shell und ist ein Netzwerkprotokoll, mit dem man sich über eine verschlüsselte Netzwerkverbindung an einem Remotecomputer anmelden kann. Standardmäßig läuft der SSH-Server auf TCP-Port 22. Für den Zugriff über SSH benötigt der Raspberry Pi außerdem Internetverbindung, die entweder über WLAN oder Ethernet Kabel möglich ist. Aus praktischen und platzsparenden Gründen ist die Verwaltung über SSH praktischer als der Aufbau eines eigenen Computerbereichs. Der Internetzugriff wird zuerst über Ethernet-Kabel garantiert. Im späteren Schritt wird gezeigt, wie man eine sichere WLAN-Verbindung am Raspberry Pi aufbauen kann. Nach dem Einsetzen der MicroSD-Karte und der Versorgung mit Strom wird der Raspbian Lite Operationssystem (OS) automatisch gestartet. Der Raspberry Pi Modell 3 B+ ist bereits mit einem Ethernet-Anschluss

ausgestattet, sodass es einfach mit einem Ethernet-Kabel an dem Internetrouter angeschlossen werden soll. Der Router soll bemerken, dass Pi mit ihm verbunden ist und wenn DHCP konfiguriert ist, sollte ihm automatisch eine IP-Adresse zugewiesen werden. Die von dem DHCP zugewiesene IP-Adresse kann entweder über die Seite des Routers oder mit einer Software, beispielsweise mit Advanced IP Scanner herausgefunden werden. Dieses Programm durchsucht alle verfügbaren WLANs des Netzwerkumfangs des Routers, beispielsweise von 192.168.0.1 bis 192.168.0.254 und gibt deren Status, Namen, IP-Adresse und MAC-Adresse an.

Der einfachste Weg, über SSH auf Ihren Raspberry Pi zuzugreifen, ist mit PuTTY, welches dabei helfen kann, eine Verbindung über SSH zum Raspberry Pi herzustellen. Es kann sofort nach dem Herunterladen des Programms ausgeführt werden, da er keine zusätzliche Installation erfordert. Nach dem Start des Programms muss die IP-Adresse bzw. der SSH-Port (Port 22) angegeben werden. Nach dem Einloggen erscheint eine Konsole, in der man den Benutzernamen und das Passwort eingeben muss. Für Raspbian lautet der Standardbenutzername pi und das Kennwort raspberry. Da es sich um ein allgemein bekanntes Kennwort handelt, ist es aus Sicherheitsgründen wichtig, das Kennwort so bald wie möglich zu ändern. In der Unix-Umgebung kann dies mit dem Befehl passwd erfolgen. Es ist ratsam, bei der Eingabe des neuen Passworts die allgemeinen Passwortrichtlinien zu befolgen. Dies ist zum Beispiel, dass das Passwort mindestens 8 Zeichen lang sein, Klein- und Großbuchstaben, Zahlen und mindestens ein Sonderzeichen enthalten sollte. Außerdem sollte das Passwort alle drei Monate geändert werden. Die amerikanische Normung hat das Problem falscher Passwörter ernst genommen und 2017 eine neue Richtlinie erlassen, die Standards für digitale IDs festlegt. Diese Norm ist das NIST Special Publication 800-63B - Digital Identity Guidelines (Grassi, Newton & Fenton, 2017). Dabei wurden nicht nur vergangene Fehler und Probleme, die daraus entstanden sind berücksichtigt, sondern auch welche Optimierungsmaßnahmen es gibt.

Wenn Raspbian zum ersten Mal startet, muss das Operationssystem aktualisiert werden. Hauptsächlich können aus Sicherheitsgründen alte Softwarefehler behoben werden, die seit der Erstellung der Image-Datei behoben wurden. Wenn man bereits über SSH mit dem Raspberry Pi verbunden ist, besteht der erste empfohlene Schritt darin, das Raspbian Lite-Betriebssystem und die darauf ausgeführten Programme zu aktualisieren. Wie im vorherigen Kapitel in der Kurzbeschreibung von Raspberry Pi erwähnt ist, wird für den Betrieb der Systeme mindestens eine 8-GByte-SD-Karte empfohlen. Vor den Aktualisierungen und einer weiteren Software-Installation lohnt es sich zu prüfen, ob genügend freier Speicherplatz vorhanden ist. Dies ist durch den Befehl *df-h* möglich. Dieser Befehl listet die Summe des freien Speicherplatzes in Potenzen von 1024 auf. Zuerst muss die Paketliste des Systems durch den Befehl *sudo apt update* aktualisiert werden. Am Ende des Durchlaufs wird eine kurze Information angezeigt, wie viele Pakete zu aktualisieren sind. Diese können im Detail durch den Befehl *apt list –upgradable* angezeigt werden. Alle installierten Pakete werden mit dem folgenden Befehl *sudo apt full-upgrade* auf die neueste Version gebracht. Das Installationsprogramm berechnet den genauen zusätzlichen Speicherbedarf und der Benut-

zerdie Benutzerin wird befragt, ob die Installation durchlaufen werden darf. Nachdem die Installation abgeschlossen ist, sollte das Raspberry Pi durch den Befehl *sudo reboot* neugestartet werden.

4.7.2 Einrichtung von WLAN

Nach der Aktualisierung der Raspbian Lite, lohnt es sich, eine drahtlose Internetverbindung anstelle des kabelgebundenen Ethernets einzurichten. Innerhalb der Reichweite des WLAN-Netzwerks ist es möglich, eine Verbindung zum Router für das Raspberry Pi herzustellen, wodurch die Mobilität erhöht wird und das Zuhause ohne Kabel ästhetisch ansprechend ist. Das installierte Raspbian Lite-Betriebssystem bietet nur Konsolenzugriff. Aus diesem Grund besteht nicht die Möglichkeit, eine Verbindung durch die grafische Benutzeroberfläche zum WLAN herzustellen. Durch den Befehl *sudo iwlist wlan0 scan* werden alle verfügbaren drahtlosen Netzwerke gescannt und alle WLANs aufgelistet, die sich in Reichweite des Raspberry Pi befinden. Der Wert der ESSID (Extended Service Set Identifier) und der Name des WLAN-Netzwerks müssen durch den Benutzerdie Benutzerin verwendeten WLAN vorgemerkt werden. WLAN und Authentifizierung werden vom Dienstprogramm "wpa_supplicant" gesteuert. Sie verfügt über eine Konfigurationsdatei, die unter `/etc/wpa_supplicant/wpa_supplicant.conf` angelegt ist, in dem die WLAN-Verbindung spezifiziert werden kann. In dieser Konfiguration sind es SSID und das PSK. PSK steht für Pre-Shared Key. Dies dient als Passwort und es gilt ein Minimum von acht Zeichen und ein Maximum von 63 Zeichen. Die Konfigurationsdatei hat Lese- und Schreibberechtigung für den Root-Benutzer und enthält das angegebene Passwort in der menschenlesbaren Form. Wenn dieses Konto gehackt wird, kann auf das gesamte Netzwerk zugegriffen werden. Um dies zu verhindern, soll man möglicherweise das Kennwort verschlüsselt anstelle in Form von Klartext eingeben. Es gibt unter Unix-Betriebssystemen den sogenannte wpa_passphrase <ssid><passphrase>Befehl, welcher ein 256-bit verschlüsseltes PSK aus SSID und Passwort generiert, das in dem folgenden Array in der Konsole ausgibt:

```
1 network={
2 ssid="ThePromisedLAN" # ESSID from sudo iwlist wlan0 scan
3 #psk=myPassword # unencrypted password
4 psk=3ad66107709864c39ced10ab... # encrypted password
5 }
```

Dieses Array enthält den SSID, den auskommentierten Klartext-PSK, welcher aus der Datei gelöscht werden soll und das verschlüsselte PSK. Das erzeugte Array ist in die Konfigurationsdatei von "wpa_supplicant" hinzuzufügen. Das Bearbeiten von Dateien unter Unix-Betriebssystemen ist zum Beispiel mit Hilfe eines generischen Texteditors namens nano möglich. Die Änderung der Datei kann nur durch den Superuser

geändert werden, aus diesem Grund ist der Befehl im Form von *sudo nano <Pfad >* auszuführen. Nach der Speicherung der Konfigurationsdatei sollte der Raspberry Pi automatisch eine Verbindung zum Internet herstellen. Diese kann durch den Befehl ifconfig wlan0 geprüft werden. Falls hier eine IP-Adresse auftaucht, ist die Verbindung erfolgreich geworden. Um sicher zu gehen, soll der Raspberry Pi neugestartet werden. Wenn die WLAN-Verbindung nach dem Neustart des Geräts erfolgreich hergestellt wurde, weist der Router in den meisten Fällen aus dem DHCP-Pool eine neue IP-Adresse zu. Man kann die neue IP-Adresse wieder mit der zuvor installierten Advanced IP Scanner-Software erhalten und über SSH mit dem Raspberry Pi verbinden, welches jetzt auch ohne Ethernet-Kabel einsatzbereit ist.

4.7.3 MQTT Broker

Die Software ZigBee2MQTT überbrückt Nachrichten von Zigbee-Geräten an MQTT. Wie im Abschnitt (Abschn. 1.5.2) beschrieben, gibt es zwischen den Klienten immer einen Broker, der für die Weiterleitung der Nachrichten zuständig ist. Grundsätzlich ist es die Funktion des Systems, dass die Sensoren Daten an ein bestimmtes Topic im MQTT-Broker veröffentlichen (publish). Die BenutzerinDer Benutzer ist auf diesem Topic über den Hausautomatisierungsplattform abonniert (subscribed) und kann die einzelnen Werte, wie Temperatur, Türstatus oder Batteriestand auslesen. Aus diesem Grund muss ein MQTT Broker vorhanden sein. Der Broker kann auf einem externen Computer ausgeführt werden, es gibt jedoch keinen Grund, den Broker nicht direkt auf dem Raspberry Pi ausführen zu lassen. Als MQTT Server wurde das laut Literatur meist verwendete Open Source MQTT-Broker Mosquitto von der Eclipse Foundation ausgewählt. Dieser Broker ist kostenlos und wird vom Raspbian-Betriebssystem unterstützt. Mosquitto kann durch den Befehl *sudo apt-get install mosquitto* unter Raspbian installiert werden. Durch *mosquitto -v* kann die Version der Mosquitto Broker getestet werden, außerdem auch, ob die Installation erfolgreich war. Mosquitto kann als Service mit dem folgenden Befehl gestartet werden: *sudo systemctl start mosquitto.service*. Mit dem Befehl *systemctl* können Dienste verwaltet, angezeigt oder geändert werden. Im vorherigen Befehl wird Mosquitto als Service gestartet, jedoch sind die Start- und Stoppbefehle nach dem Neustart des Systems ungültig. Wenn der Service nach dem Neustart des Computers gestartet werden soll, muss er aktiviert werden, welches durch den Befehl *sudo systemctl enable mosquitto.service* ermöglicht werden kann. Der Benutzername und das Passwort sind optional, um dem Mosquitto MQTT Broker beizutreten. Wenn sie nicht angegeben sind, kann jeder ein Topic abonnieren, da nur die IP-Adresse und der Port eingegeben werden müssen. Durch Verwendung eines Kontos kann der Zugriff auf den Broker einfach eingeschränkt werden. Bevor ein MQTT Konto erstellt wird, muss Mosquitto temporär stillgelegt werden. Wenn er als Service angelegt ist, kann man durch den Befehl *sudo service mosquitto stop* stoppen. Der erste Schritt besteht darin, eine neue Datei unter `/etc/mosquitto` zu erstellen, in der Benutzername und Passwort durch Doppelpunkt getrennt zu definieren sind. Hier besteht der gleiche Fall wie in der

Konfigurationsdatei von ”wpa_supplicant”. Die Eingabe des Benutzernamens und Kennworts ist im Klartext-Format nicht sicher. Es können gleichzeitig mehrere Konten angelegt werden. Aus diesem Grund muss die Kennwortdatei durch den Befehl *mosquitto_passwd -U <passwordfile >* konvertiert werden, welcher das Passwort verschlüsselt. Nach der Verschlüsselung kann der Inhalt der Datei wie folgt aussehen:

mosquitto_user:6A9TnH51PxPcpi6ha$dR0MvY2wvb2F5UvlTLu9REFTlicvw
1dEKC415Vmfv72H7rNHP27aS0FZtDWr9UmvU3kn8UQCm1s4ZOjmTHeR2g==

Als letzten Schritt muss der Speicherort der Passwortdatei in der Konfigurationsdatei von Mosquitto (`/etc/mosquitto/mosquitto.conf`) angegeben werden. Um zu verhindern, dass ein Benutzereine Benutzerin ohne Eingabe eines Benutzernamens und Passworts eintritt, muss die Konfigurationsdatei mit der Zeile ”allow_anonymous false” ergänzt werden. Anschließend kann der Mosquitto Broker durch den Befehl *sudo service mosquitto start* gestartet werden. Der Status des Service kann mit dem Befehl *sudo systemctl status mosquitto* abgefragt werden.

4.7.4 Zigbee2MQTT

Nachdem das Raspbian-Betriebssystem bzw. der MQTT-Broker installiert sind, steht das Raspberry Pi bereit, um den ZigBee2MQTT zu installieren und zu konfigurieren. Der vorgeflashte CC2531 USB-Sniffer muss über den USB-Port der Raspberry Pi angeschlossen werden. Der Pfad der CC2531 muss ermittelt und die Berechtigungen überprüft werden. Laut der Dokumentation von Zigbee2MQTT, befindet sich der Pfad von CC2531 in den meisten Fällen unter `/dev/ttyACM0`. Der Pfad bzw. die Berechtigung werden wie gefolgt geprüft:

```
1 $ ls -l /dev/ttyACM0
2 crw-rw---- 1 root dialout 166, 0 Jan 15 18:44 /dev/ttyACM0
```

Dies bedeutet, dass der Dateieigentümer (File owner) und der Gruppeneigentümer der Datei (group owner of the file) Berechtigungen zum Lesen (R) und Schreiben (W) hat. Da der Benutzer namens PI auch als Superuser des Systems fungiert, muss die Berechtigung nicht geändert werden. ZigBee2MQTT wurde in der Programmiersprache JavaScript geschrieben. Um die Installation dieser Software ausführen zu können, benötigt man das Run Time Environment für JavaScript. Aus diesem Grund wird Node.js Repository eingerichtet und Node.js installiert. Node.js ist eine plattformübergreifende Open Source-JavaScript-Laufzeitumgebung, die im Jahr 2009 eingeführt wurde. Für die Verwendung von Node-Paketen gibt es ein separates Paketinstallationsprogramm, nämlich npm. Npm (Node Package Manager) wird zum Installieren und Verwalten von Node-Programmen verwendet. Die aktuelle Version ist 13.8.0, aber der Zigbee2MQTT-Software unterstützt nun die Versionen 10.X bzw. 12.X. Die

neuere Version 12.5 kann durch den folgenden Befehl installiert werden:

```
$ curl -sL https://deb.nodesource.com/setup_12.x | sudo -E ↩
   bash -
$ sudo apt-get install -y nodejs
```

Npm wird damit zusammen installiert, sodass es nicht separat installiert werden muss. Laut Dokumentation von Zigbee2MQTT ist die Mindestversion von npm 6.0. Dies ist mit *npm -v* zu prüfen. Nach dieser Installation wird die ZigBee2MQTT Repository aus einem Project Hosting System, nämlich Github abgeholt. Das Projekt wird durch den folgenden Befehl unter `/opt/zigbee2mqtt` abgelegt. Der unterste Befehl gibt Genehmigungen für den pi-Benutzer auf rekursive Weise, d.h. es wird auf Unterverzeichnisse und deren Inhalt angewandt.

```
$ sudo git clone https://github.com/Koenkk/zigbee2mqtt.git ↩
   /opt/zigbee2mqtt
$ sudo chown -R pi:pi /opt/zigbee2mqtt
```

Nach dieser Genehmigung kann ZigBee2MQTT in dem Verzeichnis des abgeholten Projektes mithilfe von npm installiert werden.

```
$ cd /opt/zigbee2mqtt
$ npm install
```

Obwohl die Installation manche Warnings geliefert hat, können diese laut Dokumentation ignoriert werden. Die weiteren Konfigurationen können in der configuration.yaml Datei vorgenommen werden, die sich unter `/opt/zigbee2mqtt/data` befindet. Da Home Assistant in diesem Beispiel nicht verwendet wird, sollte der Wert von *homeassistant* auf den vorgegebenen Wert "false" gelassen werden. Die Möglichkeit, Zigbee-Geräte mit dem Netzwerk zu verbinden, kann über die Variable "*permit_join*" in der Konfigurationsdatei aktiviert bzw. deaktiviert werden. Um die Netzwerksicherheit zu gewährleisten und eine erneute Verbindung eines Gerätes zu vermeiden, ist es wichtig, dass die Variable auf den Wert "false" gesetzt wird, sobald die gewünschten Geräte verbunden sind. Um auf dem MQTT Broker verbinden zu können, ist die IP-Adresse des Gerätes neben dem Standardport 1883 erforderlich, auf dem der Broker läuft. Da der MQTT Broker Mosquitto direkt auf dem Raspberry Pi installiert wurde, hat dieser die gleiche IP-Adresse wie der Raspberry. Aus diesem Grund kann dieser als Localhost mit dem Standardport von MQTT 1883 angegeben werden. Unter serial port wird bestimmt, wo der CC2531 USB-Stick angeschlossen ist und der Zugriff gewährt wird. Nach der Verbindung der einzelnen ZigBee-Geräte werden die wesentlichen Daten der Geräte, wie unfreundlicherfreundlicher

Name (unfriendlyfriendly name) angelegt bzw. wenn nötig die Gruppen in demselben Verzeichnis definiert. Ein ZigBee-Netzwerk verfügt über einen voreingestellten Netzwerkschlüssel. Es handelt sich um einen 128-Bit-Schlüssel, der von allen Geräten im Netzwerk gemeinsam genutzt wird. Dieser Schlüssel ist in die Zigbee2MQTT-Software integriert, und jederjede, derdie diese Software verwendet, besitzt denselben Schlüssel. Dies kann zu Sicherheitslücken führen. Die Konfigurationsdatei kann um zusätzliche Einstellungen erweitert werden, in denen ein neuer 128-Bit-Schlüssel im Dezimal- oder Hexadezimalformat angegeben werden kann. Der Schlüssel besteht aus 16 Zahlen, die durch Kommas getrennt sind. Dezimalzahlen können zwischen Null und 255, Hexadezimalzahlen zwischen 0x00 und 0xFF liegen. Nachdem alle Einstellungen vorgenommen worden sind, kann die Software Zigbee2MQTT nun im Verzeichnis `/opt/zigbee2mqtt` mittels Aufrufes von Node Package Manager *npm start* zum Laufen bringen. Es besteht auch die Möglichkeit ZigBee2MQTT als Service zu verwenden, wodurch es im Hintergrund läuft und bei dem Neustart der Raspberry Pi automatisch durchgelaufen wird und der Benutzer/die Benutzerin nicht manuell zum Beispiel über SSH starten muss. Dazu wird die Systemvorbereitungs- und Systemverwaltungssoftware Systemd unter Linux verwenden, die ebenfalls für die Ausführung bzw. Verwaltung der Dienste verantwortlich ist. Um die Zigbee2MQTT-Software als Service auszuführen, muss man eine zigbee2mqttt.service-Datei im Verzeichnis `/etc/systemd/system` erstellen und den folgenden Inhalt einfügen:

```
1  [Unit]
2  Description=zigbee2mqtt
3  After=network.target
4
5  [Service]
6  ExecStart=/usr/bin/npm start
7  WorkingDirectory=/opt/zigbee2mqtt
8  StandardOutput=inherit
9  StandardError=inherit
10 Restart=always
11 User=pi
12
13 [Install]
14 WantedBy=multi-user.target
```

IIn dem Unit Block wird mittels After=network.target sichergestellt, dass das Service nachdem das Netzwerk aktiv ist, gestartet wird. In der Service-Sektion wird definiert, was unter welchem Laufzeitverzeichnis ausgeführt werden soll. Falls vorhanden werden Fehlermeldungen bzw. die Ausgabe ebenfalls hier angegeben. In dieser Datei sind beide auf inherit gestellt, was bedeutet, dass die Standardeingabe (standard input) bzw. Standardausgabe (standard output) vom Betriebssystem übernommen wird. Die letzten zwei Zeilen der Service-Sektion gibt an, dass der Service nach dem Neustart des Betriebssystems in jedem Fall bzw. unter dem pi-Benutzer gest-

artet werden muss. In der Install-Sektion steht fest, dass dieser Service im Rahmen des Systemstarts mit gleichwertiger Priorität gestartet werden soll. Nach der Erstellung der service-Datei kann der Zigbee2MQTT-Service, wie beim Mosquitto-Broker mit dem Befehl *sudo systemctl enable zigbee2mqtt* aktiviert und mit dem Befehl *sudo systemctl start zigbee2mqtt* gestartet werden.

4.7.5 Einrichtung von Domoticz als Service

Mit den obigen Installationen steht nun alles bereit, Daten von den ZigBee-Geräten zu senden bzw. zu empfangen. Bevor man die Geräte mit dem Koordinator verbindet, kann das Domoticz Hausautomatisierungssystem installiert werden. Für die Installation muss nur der Befehl *curl -L https://install.domoticz.com | bash*x am Raspbian ausgeführt werden. Dieser Befehl öffnet einen grafischen Software-Installer, wodurch manche Einstellungen vorgenommen werden müssen. Für die Installation wird um eine statische IP-Adresse gebeten, welche ganz am Anfang automatisch durch das Installationsprogramm erledigt wird. Der Benutzer/die Benutzerin wird nachher gefragt, welche Services, HTTP und/oder HTTPS ermöglicht werden sollen bzw. über welchen Port darauf zugegriffen werden kann und zum Schluss nach dem Installationsordner. Diese Werte können jedoch auf Default gelassen werden. Wenn die Installation fertig ist, kommt eine Nachricht mit den HTTP bzw. HTTPS Adressen, worüber der Domoticz zum Beispiel über einen Browser erreichbar ist. Das Zigbee2MQTT ist als Python-Plugin für Domoticz verfügbar, der vom Github-Benutzer Stanislav Demydiuk erstellt wurde. Das Zigbee2MQTT Python Plugin wird mit den folgenden Befehlen im Unterverzeichnis des vorher installierten Domoticz heruntergeladen.

```
1 $ cd domoticz/plugins/
2 $ git clone https://github.com/stas-demydiuk/domoticz-↩
    zigbee2mqtt-plugin.git zigbee2mqtt
```

Sobald die Installation abgeschlossen ist, kann Domoticz als Service auf dem Raspberry Pi ausgeführt werden. Der Service muss zuerst mit den gewohnten systemctl-Befehlen neugestartet und danach aktiviert werden. Falls in Zukunft ein Fehler auftritt oder wenn man prüfen möchte, ob der Domoticz-Service ordnungsgemäß funktioniert, ist es unerlässlich, alle Protokolldaten (log data) an einem Ort zu sammeln. Mit den folgenden Befehlen wird der Speicherort des Domoticz-Protokolls, der in `etc/init.d/domoticz.sh` definiert ist, festgelegt, bei dem es sich um das Basisverzeichnis für die Linux-Protokollierung handelt. Dieser Shell-Skript ist mit dem folgenden Parameter zu erweitern:

```
1 DAEMON_ARGS=$DAEMON_ARGS -log /var/log/domoticz.log
```

Da die Quellkonfigurationsdatei von Domoticz-Service geändert wurde, muss der Systemd Manager-Konfiguration neugeladen und der Domoticz-Service durch den Befehl unten neugestartet werden. Nach dem Neustart ist der Protokollierung unter `/var/log/domoticz.log` zu erreichen.

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart domoticz.service
```

4.7.6 Externer Zugang

Auf Domoticz kann von einer statischen IP-Adresse aus zugegriffen werden, aber nur von einem Gerät aus, welches mit dem gleichen Router wie der Raspberry Pi verbunden ist. In den meisten Fällen verfügt ein Heimrouter über eine öffentliche IP-Adresse, auf die von außen zugegriffen werden kann. Falls sie nicht vorhanden ist, muss sie vom Netzwerkanbieter beauftragt werden. Um diese öffentliche IP-Adresse aufrufen zu können, muss eine Portfreigabe bzw. Portweiterleitung eingestellt werden. Die Portweiterleitung wird von dem Router durchgeführt, der eine Verbindung zu der öffentlichen IP-Adresse an einem bestimmten Port herstellt und diese dann mit einer internen IP-Adresse an einem bestimmten Port verknüpft. Da die Portweiterleitung pro Router unterschiedlich konfiguriert werden muss, wird dies in dieser Masterarbeit nicht behandelt. Domoticz wurde bei der Installation auf den Ports 8080 (HTTP) und 443 (HTTPS) eingerichtet. Bei der Konfiguration der Portfreigabe muss man diese obigen Ports angegeben werden. Der Port für die externe IP-Adresse kann beliebig sein. Da im folgenden Unterkapitel der externe Zugriff über HTTPS konfiguriert wird, ist es notwendig, dass der Port 8080 der internen IP-Adresse mit dem Port 80 der externen IP-Adresse verknüpft wird. Ein fester Domänenname ist erforderlich, um von einem externen Netzwerk auf die Automatisierungsplattform nicht über die IP-Adresse, sondern über ein Domain Name zugreifen zu können. Dies kann durch einen DDNS (Dynamic Domain Name System) ermöglicht werden. Ein DNS-Server verknüpft einen textuellen Namen mit einer öffentlichen IP-Adresse. Im Gegensatz zum DNS, verwendet DDNS eine Methode, die einen DNS-Eintrag automatisch in Echtzeit aktualisiert, dadurch kann ein DDNS mit einer nicht-statischen IP-Adresse auch genutzt werden. Es gibt verschiedene kostenlose und kostenpflichtige DDNS-Anbieter, bei denen man nach der Registrierung einen Domain-Namen für ihre/seine öffentliche IP-Adresse erstellen kann.

4.7.7 Umstellung auf HTTPS

Wie bereits in den Kapiteln 1.5.1 und 1.5.3 beschrieben, ist es aus Datenschutzgründen wichtig, sichere Protokolle zu verwenden. Wenn der Datenaustausch mit TLSSSL

verschlüsselt ist, handelt es sich um die sichere Version von HTTP, nämlich um HTTPS (HTTP secured). Die Domoticz-Webseite ist jedoch in dieser Phase mit einem Browser ausschließlich über HTTP erreichbar. Obwohl während der Installation das sichere HTTPS-Protokoll zwar aktiviert wurde, ist es erst verfügbar, wenn Domoticz ein SSL Zertifikat hinzufügt. Für diesen Zweck kann sich beispielsweise ein kostenloser SSL-Dienst, namens Let's Encrypt (Let's Encrypt - Internet Security Research Group, 2020) eingesetzt werden. Let's Encrypt-Zertifikate sind 90 Tage gültig. Dies ist hauptsächlich aus Sicherheitsgründen am wichtigsten. Da ein Schlüssel kürzer gültig ist, können gestohlene Schlüssel und schlecht ausgestellte Zertifikate früher außer Betrieb genommen werden. Obwohl das Zertifikat 90 Tage gültig ist, schlägt Let's Encrypt eine Verlängerung um 60 Tage vor. Mit dem Certbot-Client kann das Zertifikat aktualisiert werden, welches alle 60 Tage durch einen in einer Unix-Umgebung verwendetes Automatisierungssystem namens Cron-Job gestartet wird.

Der erste Schritt besteht darin, das Letsencrypt-Repository von Github durch den Befehl *sudo git clone https://github.com/letsencrypt/letsencrypt* in das Verzeichnis etc zu klonen und im resultierenden letsencrypt-Ordner durch *letsencrypt-auto* zu installieren. Bei der Installation ist ein Problem mit folgender Fehlermeldung aufgetaucht: "Certbot has problem setting up the virtual environmen". Nach ein wenig Recherche stellte sich heraus, dass die Datei pop.conf einen Fehler aufwies, der für die Installation und Verwaltung von Python-Paketen verantwortlich ist. Die Lösung besteht darin, die Konfigurationsdatei zu löschen. Nach der erneuten Ausführung des Installationsbefehls ist die Installation erfolgreich durchgelaufen. Es besteht nun die Möglichkeit, ein Zertifikat mit dem folgenden Befehl zu erstellen:

```
1 $ sudo /etc/letsencrypt/letsencrypt-auto certonly --webroot↩
    --email xygmail.com -d mydns.ddns.com -w
   domoticz/www/
```

Dieser Befehl kann nur in dem Fall funktionieren, wenn man vorher die Portweiterleitung auf dem Router von dem externen Port 80 auf dem internen Port von Domoticz, meistens auf 8080 eingestellt hat. Wenn das Zertifikat erfolgreich erstellt wurde, erscheint eine Nachricht in der Konsole, die auch das Ablaufdatum des Zertifikats enthält. Das erstellte Zertifikat muss zu Domoticz hinzugefügt bzw. im letzten Schritt muss Domoticz neugestartet werden, wie folgt:

```
1 $ sudo cat /etc/letsencrypt/live/mydns.ddns.com/privkey.pem↩
    >> ~/domoticz/server_cert.pem
2 $ sudo cat /etc/letsencrypt/live/mydns.ddns.com/fullchain.↩
   pem >> ~/domoticz/server_cert.pem
3 $ sudo cp ~/domoticz/server_cert.pem ~/domoticz/↩
   letsencrypt_server_cert.pem
4 $ sudo /etc/init.d/domoticz.sh restart
```

Domoticz ist nun über HTTPS unter *mydns.ddns.com:443* erreichbar. Es wird empfohlen, den Port 80 im Domoticz auszuschalten, damit die Webseite über das unverschlüsselte Protokoll nicht abgerufen werden kann. Wie bereits erwähnt wurde, muss das Zertifikat mindestens alle 90 Tage erneuert werden. Für diesen Zweck ist ein Cron-Job einzurichten, der automatisch in bestimmten Zeitintervallen die Erneuerung des Zertifikates ausführt. Unter Linux wird dies in der Crontab verwaltet. Hier soll folgende Zeile eingefügt werden, die den Befehl am ersten Tag jedes Monats um zwei Uhr ausführt und dadurch das Zertifikat monatlich erneuert.

```
1 0 2 1 * * sudo certbot-auto renew --webroot -w ~/domoticz/↩
    www/ --deploy-hook ~/domoticz/scripts/deploy-cert.sh >/↩
    dev/null
```

4.7.8 Geräte verbinden

Um die Domoticz Smart-Home-Plattform nutzen zu können, müssen noch grundlegende Einstellungen im Webinterface vorgenommen werden. Dazu muss man Domoticz in einem Browser öffnen, wobei die URL aus der IP-Adresse des Raspberry Pis und einem bei der Installation angegebenen Port, zum Beispiel 8080, besteht. Der erste Schritt besteht darin, Domoticz mitzuteilen, dass Zigbee2MQTT verfügbar ist. Dies kann aus der Liste im Feld Typ unter dem Menüpunkt Einstellungen/Hardware ausgewählt werden. Einen Anzeigenamen muss man ebenfalls angeben. Obwohl das Domoticz Zigbee2MQTT-Plugin erfolgreich installiert wurde, wurde es nicht in der Auswahlliste angezeigt. Da es sich hier um einen Python Plugin handelt, muss Python auf dem Raspbian zur Verfügung stehen. Obwohl die Python-Versionen 2 und 3 auf dem Raspbian-Betriebssystem Standard sind, stellte sich nach einigen Recherchen im Internet heraus, dass die für Domoticz erforderliche Entwickler-Python-Bibliothek (Python-dev) in der aktuellen Version von Raspbian nicht standardmäßig installiert ist. Diese Installation kann dies mit dem folgenden Befehl nachgeholt werden mit einem anschließenden Neustart des Domoticz Service:

```
1 $ sudo apt installiere dev-python3
2 $ sudo systemctl restart domoticz.service
```

Es kann nun Zigbee2MQTT aus der Liste ausgewählt werden. Außerdem müssen die IP-Adresse, der Port bzw. das Topic des MQTT-Brokers und, falls konfiguriert, der Benutzername und das Kennwort angegeben werden. Mit diesen Einstellungen kann Domoticz das Haupttopic des MQTT-Brokers abonnieren. In diesem Menüpunkt kann auch die Geräteverbindungsberechtigung aktiviert werden. Alternativ kann diese Berechtigung direkt in der YAML-Konfigurationsdatei angegeben werden. Jedes Gerät muss gemäß den Angaben des Herstellers an den ZigBee-Koordinator

angeschlossen werden. Bei allen drei Geräten muss man den Reset-Knopf 5 Sekunden lang gedrückt halten, damit sich das Gerät automatisch mit dem Koordinator verbindet. Den LED-Streifen muss man viermal aus dem USB-Port herausziehen und dann wieder einstecken. Dadurch beginnt das Streifenlicht grün zu blinken, was bedeutet, dass der Streifen in den Pairing-Modus gewechselt ist. Wenn die Verbindung hergestellt ist, hört er auf zu blinken und leuchtet dauerhaft grün. Verbundene Geräte werden in der Zigbee2MQTT-Konfigurationsdatei `/opt/zigbee2mqtt/data/configuration.yaml` angezeigt. Hier besteht die Möglichkeit, einen sprechenden Namen anzugeben, welcher dann als Anzeigename statt der Geräte-ID dient. Wie im ersten Kapitel (Abschn. 1.5.2) beschrieben, werden drei Servicequalitätsstufen (QoS) von MQTT definiert. Der Wert von 0 bis 2 kann in dieser Konfigurationsdatei pro Gerät definiert werden, abhängig von der Wichtigkeit der Ankunft der Pakete.

Logdaten werden vom Journald-Service-System unter Linux mithilfe von systemd erfasst, gespeichert und verarbeitet. Journald kann durch den Befehl *journalctl -u zigbee2mqtt.service* aufgerufen werden. Nach der Verbindung der Geräte mit dem Koordinator wurde im Log vom zigbee2mqtt.service die Meldung angezeigt, dass das fire_alarm-Gerät (alias Heiman's Smoke Detektor) nicht unterstützt ist. Wie sich herausstellte, wird diese neue Version HS1SA-N von Heiman noch nicht von Zigbee2MQTT unterstützt. Laut der Dokumentation von Zigbee2MQTT ist es möglich, Geräte, welche noch nicht unterstützt werden, manuell zu erfassen und dadurch unterstützen zu lassen. Dazu muss man eine Datei von `zigbee2mqtt/node_modules/zigbee-herdsman-converters/devices.js` ergänzen. Basierend auf der alten Version des Heiman-Rauchmelders wurde das folgende Objekt geschrieben und in die als Devices bezeichnete JavaScript-Datei eingefügt:

```
1  {
2    zigbeeModel: ['SmokeSensor-N-3.0'],
3    model: 'HS1SA-N',
4    vendor: 'HEIMAN',
5    description: 'Smoke detector',
6    supports: 'smoke',
7    fromZigbee: [fz.heiman_smoke, fz.battery_200],
8    toZigbee: [],
9    meta: {configureKey: 1},
10   configure: async (device, coordinatorEndpoint) => {
11         const endpoint = device.getEndpoint(1);
12         await bind(endpoint, coordinatorEndpoint, ['↩
             genPowerCfg']);
13         await configureReporting.batteryPercentageRemaining↩
             (endpoint);
14 },
```

Nach dem Neustart des Zigbee2MQTT-Services und dem Betrachten der Log-Datei stellte sich heraus, dass das Gerät nun erfolgreich unterstützt wurde. Nach dem Bestätigen des Test-Knopfes auf dem Rauchmelder wurde die folgende ZigBee-Nachricht an den Koordinator gesendet bzw. die folgende Nachricht mittels des MQTT-Brokers veröffentlicht:

```
1 Configuring 'fire_alarm'
2 Received Zigbee message from 'fire_alarm', type '↩
    commandStatusChangeNotification', cluster 'ssIasZone', ↩
    data '{"zonestatus":32,"extendedstatus":0}' from ↩
    endpoint 1 with groupID 0
3 MQTT publish: topic 'zigbee2mqtt/fire_alarm', payload '{"↩
    smoke":false,"battery_low":false,"linkquality":15,"↩
    battery":100}'
4 Succesfully configured 'fire_alarm'
```

Wie in der Protokolldatei gezeigt wird, war die Datenübertragung erfolgreich und das Gerät ist mit Zigbee2MQTT betriebsbereit. Im nächsten Schritt werden die Geräte zu der von Domoticz konfigurierten Website hinzugefügt, damit sie überwacht werden können. Die verbundenen Geräte werden unter dem Menüpunkt "Einstellungen/Geräte" aufgelistet. Einzelne Daten können mit einem Knopfdruck zur Seite hinzugefügt bzw. wider entfernt werden. Es besteht darüber hinaus noch die Möglichkeit, als besonders wichtig erachtete Daten direkt an der Hauptseite der Domoticz-Webseite anzuheften. Die Liste enthält viele wählbare Daten wie Temperatur, Luftfeuchtigkeit, Batteriestand der Temperatursensoren usw. sowie auch Kombinationen einzelner Daten. Obwohl das Xiaomi-Thermometer und der Türsensor in der Liste aufgeführt waren, wurde der Heiman-Rauchmelder auch hier nicht in der Liste aufgeführt. Der Domoticz-Service verfügt des Weiteren über ein Log, das ebenfalls mit journalctl eingesehen werden kann. Hier war eine Meldung ersichtlich, dass der Plugin für Heiman von Domoticz nicht das Modell ´´HS1SA-N" unterstützt. Es besteht jedoch die Möglichkeit, einen neuen Issue auf Github zu erstellen und das Plugin unterstützen zu lassen. Aufgrund der Informationen im Domoticz-Forum ist es jedoch einfach möglich, zuvor nicht unterstützte Geräte manuell dem Domoticz-Service hinzuzufügen. Es wurden folgende Zeilen zu der genannten Python-Datei hinzugefügt:

```
1 $ sudo nano /home/pi/domoticz/plugins/zigbee2mqtt/adapters/↩
    __init__.py
2   # HS1SA-N - Heimann Smoke Sensor
3   'HS1SA-N': SmokeSensorAdapter
```

Die Datei __init__.py ist für die ordnungsgemäße Funktion von Python-Paketen unerlässlich. Wenn etwas aus dem Paket importiert wird, wird der Inhalt dieser Python-Datei automatisch ausgeführt. In dieser Datei wurde der unter Zigbee2MQTT de-

finierte Modellname hinzugefügt und ein Adapter, wie zum Beispiel TemperatureSensorAdapter oder ContactAdapter zugewiesen. Da dieses Gerät ein Rauchmelder ist, wurde es sinnvollerweise mit dem SmokeSensorAdapter verknüpft. Nach dem Neustart des Domoticz-Services war die untenstehende Meldung in der Log-Datei ersichtlich und die Daten des Rauchmelders wurden nun auf der Domoticz-Webseite angezeigt.

```
1 Device HS1SA-N 0x00158d0002758c2d (fire_alarm)
2 Creating domoticz device to handle "linkquality" key for ↩
     device with ieeeAddr 0x00158d0002758c2d
3 Creating domoticz device to handle "battery_voltage" key ↩
     for device with ieeeAddr 0x00158d0002758c2d
4 Creating domoticz device to handle "smoke" key for device ↩
     with ieeeAddr 0x00158d0002758c2d
```

Dadurch wurden alle drei zum Test ausgewählten Geräte erfolgreich mit dem zentralen CC2531-Koordinator bzw. mit der darauf laufenden Zigbee2MQTT-Software verbunden und in Domoticz eingerichtet. Wenn die Verbindung aller Geräte hergestellt ist, ist es aus Sicherheitsgründen erforderlich, die Geräteverbindungsberechtigung wieder zu deaktivieren. Geräte senden über das ZigBee-Kommunikationsprotokoll eine Nachricht an den Koordinator, der diese Nachricht als MQTT-Nachricht durch die Zigbee2MQTT-Software an ein bestimmtes Topic des MQTT-Brokers sendet. MQTT-Nachrichten verwenden das JSON-Format. In dem JSON-Format sind die Daten als Schlüssel-Wert-Paare definiert und es werden zwei Typen unterschieden. JSON-Objekte sind in geschweiften Klammern als Schlüssel: Wert anzugeben. Falls zu einem Schlüssel mehrere Werte gehören, können diese als JSON-Array angegeben werden. JSON-Arrays sind durch Komma separierte Elemente, welche zwischen eckigen Klammern gestellt sind. Diese Meldungen sind beispielsweise für das Temperatur- und Feuchtigkeitsmessgerät wie folgt aufgebaut:

```
1 MQTT publish: topic 'zigbee2mqtt/temperature', payload '{"↩
     battery":86,"voltage":2975,"temperature":22.99,"humidity↩
     ":54.92,"linkquality":123}
```

Wein ein Client das Topic abonniert, erhält er die gewünschten Informationen und kann diese beispielsweise dem Benutzer/der Benutzerin graphisch präsentieren, wie die Domoticz Web-Oberfläche. Der Benutzerdie Benutzerin kann vom LED-Streifen Informationen wie Status oder derzeitige Lichtstärke erhalten. Wichtiger ist aber natürlich die Möglichkeit Befehle senden zu können. Der LED-Streifen kann in Farbe, Status bzw. Helligkeit geändert werden. Wenn man über die Domoticz Web-Oberfläche die Farbe ändert, wird eine Nachricht an das MQTT-Topic `zigbee2mqtt/led_strip` mit folgenden Parametern gesendet:

```
Publishing 'set' 'color' to 'led_strip'
MQTT publish: topic 'zigbee2mqtt/led_strip', payload '{"state":"ON","brightness":127,"color":{"x":0.162,"y":0.2335},"color_temp":60}'
```

Wenn der Status oder die Lichtstärke der LED geändert wird, wird auch eine ähnliche Nachricht auf das gleiche Topic gesendet. Im Gegensatz zu der zuvor beschriebenen Nachricht wird hier anstelle der Farbe (color) der Status (state) oder die Lichtstärke (brightness) gesetzt. In der Log-Datei der Software Zigbee2MQTT ist nach der Ausschaltung des LED-Streifens die untere Nachricht ersichtlich:

```
Received MQTT message on 'zigbee2mqtt/led_strip/set' with data '{"state": "Off"}'
```

4.8 Installation der Home Automation Hausautomationsplattform

In diesem Abschnitt wird beschrieben, wie der Open Source Home Assistant Hass.io auf einem Raspberry Pi installiert und mit den zuvor verwendeten Smart Tools konfiguriert und verwendet wird.

4.8.1 Installation des Hass.io Operationssystems

Im Gegensatz zur Domoticz-Hausautomationsplattform verfügt die Home Assistant-Plattform über ein eigenes Betriebssystem, nämlich Hass.io, welches für den Raspberry Pi optimiert wurde. Imagedateien für die verschiedenen Raspberry Pi Versionen sind auf der Homepage des Home Assistenten verfügbar.Beim Raspberry Pi 3 B+ kann man zwischen einem 32-Bit- und einem 64-Bit-Betriebssystem wählen. Da Home Assistent das 32-Bit-Betriebssystem für diese Raspberry Pi Version empfehlt, wird dieses installiert. Für die Installation ist eine microSD-Karte mit einer Speichergröße von mindestens 32 GB erforderlich. Hier ist es wie bei Domoticz wieder möglich, die Software balenaEtcher zu verwenden, um die Hass.io-Imagedatei auf der SD-Karte zu installieren. Nach dem Installationsprozess wird ein Überprüfungsprozess von balenaEtcher durchgeführt. Sobald dies erledigt ist, ist die SD-Karte einsatzbereit. An den Raspberry Pi muss die microSD-Karte, das Netzteil und ein Ethernet-Kabel angeschlossen werden.

Da Hass.io kein normales Betriebssystem ist, ist die Konfiguration eingeschränkter als beispielsweise bei Raspbian. Hass.io benutzt NetworkManager, um das Host-Netzwerk zu steuern. Wenn ein drahtloser Internetzugang erforderlich ist, kann dies nach der Installation des Hass.io auf einer SD-Karte oder einem USB-Stick manuell erfolgen, da dies in der aktuellen Entwicklungsphase noch nicht über die Web-Oberfläche verfügbar ist. Auf der SD-Karte wird eine zweite Partition namens hassos-boot angezeigt. Dieser Ordner sollte mit der folgenden Verzeichnisstruktur erweitert werden: `hassos-boot/CONFIG/network/my-network`, wobei es sich bei "my-network" um die Konfigurations-Textdatei handelt. Auf einem leeren USB-Stick soll ebenso dieselbe Verzeichnisstruktur erstellt werden. In der Hass.io Dokumentation kann der entsprechende Inhalt dieser Konfigurationsdatei entnommen werden, die mit den eigenen Internetdaten, wie SSID und Passwort (PSK) ergänzt werden muss (Vizeli, 2019).Da diese Einstellungen bereits im vorherigen Abschnitt (Abschn. 4.7.2) beschrieben wurden, wird dies hier nicht wiederholt. Während des Startvorgangs des Raspberry Pi wird diese Konfigurationsdatei von der SD-Karte oder vom USB-Stick automatisch eingelesen und das WLAN entsprechend konfiguriert.

Laut Hass.ios Dokumentation kann der erste Start abhängig von der Geschwindigkeit des Internets bis zu 20 Minuten dauern, da hier die neuesten Updates automatisch heruntergeladen werden. Sobald Hass.io fertig ist, ist es unter http://hassio.local:8123 oder der vom Router zugewiesenen IP-Adresse verfügbar. Wie hier ersichtlich ist, ist der Port der Netzwerkadresse nicht wählbar, sondern die Standardeinstellung ist 8123. Man kann die IP-Adresse entweder auf der Routerseite oder mit der Software Advanced IP Scanner abfragen. Sobald das Update abgeschlossen ist, wird der Raspberry Pi neu gestartet. Beim ersten Zugriff auf das Webinterface muss ein neuer Benutzer erstellt und wesentliche Informationen, wie der aktuelle Ort, angegeben werden. Im Gegensatz zu Domoticz ist hier kein SSH-Zugriff erforderlich, da alle Einstellungen, einschließlich der Konfigurationsdatei, über die Weboberfläche zugänglich sind. Dazu muss man das Konfigurator Add-On (Configurator) installieren, um die Konfigurationsdatei des Home Assistant direkt über die Weboberfläche des Home Assistant zu verwalten. Die Add-Ons sind auf der rechten Seite der Webseite unter dem Menüpunkt Namen Supervisor zu verwalten. Nachdem die Installation fertig ist, können hier die Grundeinstellungen, wie zum Beispiel Add-Ons, Zertifikate oder Sicherheitseinstellungen konfiguriert werden. Neben der in Domoticz verwendeten YAML-Datei wird in den Konfigurationsdateien auch JSON (JavaScript Object Notation) verwendet. Im Konfigurator-Add-On kann weiters eingestellt werden, dass Hass.io beim Hochfahren des Raspberry Pi gestartet und automatisch Updates installiert werden. Das Konfigurator-Add-On muss gestartet werden und es kann in der Log-Datei, die direkt im Add-On verfügbar ist, geprüft werden, ob die Einrichtung erfolgreich war. Die Home Assistant-Dateien, einschließlich der Hauptkonfigurationsdatei namens configuration.yaml sind ebenfalls über diese Oberfläche erreichbar.

Zusätzlich sollte auch SSH aktiviert sein, um Hass.io effizient steuern zu können. Hierfür gibt es auch ein Addon, den SSH-Server. Für die Verwendung von SSH ist eine Authentifizierung erforderlich. Es gibt zwei Möglichkeiten, dies zu tun. Die erste

besteht darin, im Konfigurationsbereich des SSH-Servers ein Kennwort einzugeben. Mit diesem Kennwort können Sie sich dann über ein Terminal, zum Beispiel über Putty anmelden. Der Benutzername ist standardmäßig root. Die andere Zugriffsmethode ist die Verwendung eines Schlüssels anstelle eines Kennworts. Dies ist viel sicherer, da der Zugriff auf Hass.io nur dann möglich ist, wenn man den richtigen Schlüssel hat. Dies erfordert zum Beispiel die Verwendung der Putty-basierten Software, Putty Gen. Nach dem Öffnen des Programms wird einfach auf die Schaltfläche Generieren geklickt, wodurch automatisch ein öffentlicher und ein privater SSH-RSA-Schlüssel generiert werden. Der private Schlüssel muss auf dem Gerät gespeichert werden, von dem aus über SSH auf den Raspberry Pi zugegriffen werden kann. Es lohnt sich, ein Backup des privaten Schlüssels zu erstellen, da man sich nicht mehr am Raspberry Pi anmelden kann, wenn dieser verschwindet. Der private Schlüssel muss von Putty beim Verbindungsaufbau über SSH mit dem Raspberry Pi verwendet werden. Diese Putty-Einstellung kann unter dem Menüpunkt ”SSH - Auth.” gemacht werden. Der öffentliche Schlüssel ist als Text zu finden. Dieser Text muss zwischen Anführungszeichen in das Array des Felds ”authorisation_keys” in der Konfigurationsdatei des SSH Server Addon kopiert werden. Sobald dies erledigt ist, kann über Putty via SSH eine Verbindung zum Raspberry Pi hergestellt werden, wo nur mehr ein Benutzername eingegeben werden muss.

Ein anderer Weg, von einem Windows-Gerät aus auf Hassio-Systemdateien zuzugreifen, ist über Samba File Sharing. In Hass.io ist dies ebenso als Add-On verfügbar. Nach der Installation steht eine Konfigurationsdatei zur Verfügung, auf die Zugriff durch Eingabe eines Benutzernamens und eines Kennworts gewährt werden kann. Es muss lediglich mit einem Dateimanager wie Windows Explorer eine Verbindung zur IP-Adresse des Raspberry Pi hergestellt werden. Im Dateimanager werden alle Hassio-Verzeichnisdateien angezeigt, sodass auf Konfigurationsdateien ohne Verwendung von SSH zugegriffen werden kann. Diese Funktion erleichtert Windows-Benutzern und Benutzerinnen die zukünftige Konfiguration des Home Assistant, indem über die grafische Benutzeroberfläche von Windows auf sie zugegriffen werden kann.

4.8.2 MQTT Broker Add-On

Wie bereits bei der Installation von Domoticz (Abschn. 4.7.3) erwähnt ist, ist ein MQTT-Broker für die Verwendung der Zigbee2MQTT-Software unerlässlich. Der bekannte Mosquitto Broker ist auf Hass.io als Add-On erhältlich. Im Abschnitt Anmeldung zum Mosquitto-Add-On können ein MQTT-Benutzer und ein Kennwort definiert werden, wodurch die Sicherheit des Systems verbessert wird. Diese müssen als JSON-Objekt im JSON-Array namens login angegeben werden. Erwähnenswert ist hier die Verwendung von Geheimnissen, die in einer Datei namens secrets.yaml gespeichert werden können. Diese Datei ist ein Schlüsselwörterbuch, in dem Kennwörter oder andere wichtige Werte als Schlüssel-Wert-Paar gespeichert werden können.

Die Verwendung dieser Option bietet keine zusätzliche Sicherheit. Es ist jedoch praktisch, da es ausreicht, wenn ein Kennwort in mehreren Konfigurationen verwendet wird, dieses zentral in der Datei secrets.yaml zu ändern, anstatt dieses für jede Konfiguration separat ändern zu müssen. Auf die Werte der gespeicherten Geheimnisse kann durch Eingabe von secret <secret_key> zugegriffen werden.

In der Konfigurationsdatei des Mosquitto Brokers ist weiterhin wichtig, dass der zum Feld anonymous gehörende Wert auf "false" gesetzt ist, da dies verhindert, dass Klienten ohne Benutzername bzw. Passwort auf den Broker zugreifen können. Alle anderen Werte in der Konfiguration können standardmäßig beibehalten werden. Grundsätzlich verfügt das Add-On Mosquitto zusätzlich zum Standardport 1883 noch über die Ports 8883 und 8884. Diese sind für eine verschlüsselte MQTT-Verbindung erforderlich. Da im integrierten Smart Home der Masterarbeit keine datenschutzrelevanten Daten gesendet werden, wird keine Verschlüsselung durchgeführt. Wenn dies der Fall ist, können diese Ports mittel Web-Oberfläche deaktiviert werden. Nach dem Speichern der Einstellungen kann das Add-On gestartet und danach der Broker zum Hass.io hinzugefügt werden, was auf zwei Arten möglich ist. Eine davon geschieht über die Konfigurationsdatei. Dazu müssen folgende Werte in der Hassio-Konfigurationsdatei, nämlich configuration.yaml hinzugefügt werden:

```
1 mqtt:
2   broker: core-mosquitto
3   username: '!secret mqtt_user'
4   password: '!secret mqtt_pw'
```

Wie ersichtlich, können die in der Datei secrets.yaml gespeicherten Passwörter auch in dieser Konfigurationsdatei verwendet werden. Der Broker-Wert ist ein Alias für die IP-Adresse des MQTT-Brokers, die bei der Installation des Add-Ons automatisch eingestellt wurde. Die andere, einfachere Variante ist über die Web-Oberfläche. Nach der Installation des Mosquitto Add-Ons taucht ein Block namens MQTT: Mosquitto Broker unter dem Menüpunkt Konfiguration/Integrationen auf, welcher durch einen Klick automatisch den Broker in Hass.io integriert.

4.8.3 Zigbee2MQTT Add-On

Nach dem Flashen der Firmware auf dem CC2531 USB-Stick (siehe Abschn. 4.7.4) kann die Zigbee2MQTT-Software ins Home Assistant integriert werden. Die Zigbee2-MQTT-Software ist nicht Teil der Standard Hassio-Bibliothek. Aus diesem Grund muss das Zigbee2MQTT-Repository zuerst hinzufügt werden, um sie als Add-On zu installieren. Die Integration von Zigbee2MQTT Home Assistent, welche von Daniel Welch entwickelt wurde, ist auf Github verfügbar. Das Repository kann im Bereich Repositories des Add-On-Stores hinzugefügt werden, wohin einfach der folgende

Link kopiert werden muss: https://github.com/danielwelch/hassio-zigbee2mqtt. Im Anschluss daran werden zwei Add-Ons mit den Namen zigbee2mqtt bzw. zigbee2mqtt-edge angezeigt, die auf Knopfdruck installiert werden können. Ersteres enthält die veröffentlichte Zigbee2MQTT-Version, während letzteres die Entwicklerversion darstellt. Die Konfiguration des Add-Ons ähnelt der bei Domoticz verwendeten Datei, außer dass für die Geräte bzw. Gruppen separate Dateien vorhanden sind. Die Verfügbarkeit dieser Dateien kann in der Variablen "data_path" definiert werden. Da in diesem Fall Zigbee2MQTT mit Home Assistant verwendet wird, muss der Wert für den Heimassistenten auf "true" gesetzt werden. Die Adresse des MQTT-Brokers, das Basistopic, der Benutzername und das zugehörige Kennwort müssen ebenfalls in der Konfigurationsdatei angegeben werden. Wie üblich kann anstelle des freien Textes die in der Datei secrets.yaml gespeicherten Schlüssel verwendet werden. Nach dem Ändern der Einstellungen kann die Zigbee2MQTT-Software sofort gestartet werden. Es gibt auch eine Umschalttaste, mit der dieses Add-On automatisch gestartet werden kann, wenn der Raspberry Pi gestartet wird. Die Logdaten für die Software sind an zwei Speicherorten verfügbar. Eine davon befindet sich direkt im Abschnitt Log des Add-Ons, welches ausschließlich die aktuellen Protokolldaten enthält. Das andere befindet sich in dem während der Installation definierten Verzeichnis, nämlich `share/zigbee2mqtt/log`. Dieser Ordner enthält Unterordner mit Zeitstempeln, die die Zeiten sind, zu der Zigbee2MQTT gestartet wurde. Im Gegensatz zur vorherigen sind hier alle historisierten Logdaten verfügbar.

4.8.4 Geräte im Hass.io verbinden

Wie bei Domoticz erwähnt, muss der Wert für "allow_join" in der Konfigurationsdatei Zigbee2MQTT auf "true" eingestellt sein, damit Geräte mit dem Koordinator verbunden werden können. Hass.io bietet eine Option zum Aktivieren der Gerätepaarung direkt auf der Website. In diesem Fall müssen sogenannte Automatisierungen manuell erstellt werden, da sie nicht Teil der grundlegenden Zigbee2MQTT-Bibliothek sind. Unter Hass.io stehen sogenannten Entitätskarten zur Verfügung, die die einzelnen Elemente gruppieren. Diese Karten sind die Hauptkomponenten der Website, über die jede Einheit, wie zum Beispiel die gemessene Temperatur und Luftfeuchtigkeit, Batteriestatus eines Sensors oder der Koordinatenstatus angezeigt und die steuerbaren Geräte, wie zum Beispiel LED-Streifen eingestellt werden können. Mithilfe des folgenden Codes ist es möglich, eine Umschalttaste in einer Entitätskarte zu erstellen. Dieser muss in die Datei configuration.yaml einfügt werden.

```
1 input_boolean:
2   zigbee_permit_join:
3     name: Allow devices to join
4     initial: off
5     icon: mdi:cellphone-wireless
```

Bei der Umschaltung des Knopfes wird eine MQTT-Nachricht gesendet, die den Wert "allow_join" der Konfigurationsdatei ändert. Damit diese Schaltfläche eine MQTT-Nachricht senden kann, muss eine automatische Funktion erstellt werden. Dies kann entweder direkt in der Konfigurationsdatei oder in der Datei namens automations.yaml, welcher von configuration.yaml importiert wird, gemacht werden.

Der folgende Code ist ein Beispiel für das Erstellen solcher Funktionen. In diesem Code wird eine neue Entität mit dem Namen "enable_zigbee_join" erstellt. Der Aliaswert kann verwendet werden, um einen Anzeigenamen anzugeben, der auf der Entitätskarte angezeigt wird. Unterhalb der Trigger Variablen wird angegeben, dass nach dem Umschalten des vorher erstellten Umschaltknopfes auf Ein, der Befehl in der Aktionsvariablen ausgeführt wird. Unter Aktion wird ein MQTT Publish Service ausgeführt, welches den Wert der Variablen "permit_join" über das MQTT-Topic auf "true" setzt. Somit wird die Verbindung der Geräte mit dem Koordinator erlaubt.

```
- id: enable_zigbee_join
  alias: Enable Zigbee joining
  hide_entity: true
  trigger:
    platform: state
    entity_id: input_boolean.zigbee_permit_join
    to: 'on'
  action:
  - service: mqtt.publish
    data:
      topic: zigbee2mqtt/bridge/config/permit_join
      payload: 'true'
```

Um die Verbindungsmöglichkeit zu deaktivieren, wenn die Umschalttaste auf Aus gestellt wird, muss noch eine Entität ähnlich der vorherigen in die Datei hinzugefügt werden. Es muss lediglich ein neuer Name unter ID angegeben werden, der Wert der Variable "to" unterhalb des Triggers in "off" bzw. der Payload-Wert unterhalb der Action in "false" geändert werden. Nach dem Neustart von Hass.io kann in der Log Sektion des ZigBee2MQTT-Add-Ons überprüft werden, ob die Umschalttaste einwandfrei funktioniert.

Die Geräte werden auf die gleiche Weise wie für Domoticz beschrieben (siehe Abschnitt 4.7.8) mit dem Koordinator verbunden. Der Temperatur- und Luftfeuchtigkeitssensor, der Tür- und Fenstersensor sowie der LED-Streifen müssen gemäß den Angaben des Herstellers in den Verbindungsmodus gestellt werden und werden automatisch mit dem Koordinator verbunden. Bei Domoticz wurde bereits herausgefunden, dass das HS1SA-N Modell des Heiman-Rauchmelders von der aktuellen Version von Zigbee2MQTT nicht unterstützt wird. Nach dem Versuch der Verbindung des Geräts an den Home Assistant wurde die folgende Nachricht in der ZigBee2MQTT-

Protokolli angezeigt:

```
Device '0x00158d0002758c2d' joined
MQTT publish: topic 'zigbee2mqtt/bridge/log', payload '{"type":"device_connected","message":{"friendly_name":"0x00158d0002758c2d"}}'
Starting interview of '0x00158d0002758c2d'
.. "sw_version":"Zigbee2mqtt 1.10.0","model":"Smoke detector (HS3SA)","manufacturer":"HEIMAN"},"availability_topic":"zigbee2mqtt/bridge/state"..
```

Dies bedeutet, dass das System über Home Automation das Gerät automatisch als älteres Modell (HS3SA) erkannt und mit dem Koordinator verbunden hat. Dies ist im Wesentlichen eine Problemumgehung, da es sich nicht um die richtige Version handelt, der Rauchmelder jedoch weiterhin funktioniert. Wie bei Domoticz ist es möglich, noch nicht unterstützte Modelle manuell zu unterstützen. Basierend auf der Dokumentation sieht dies für Home Assistant jedoch etwas anders aus. Wenn Stapeländerungen an dem Home Assistant-System vorgenommen werden sollen, müssen die Änderungen wieder für das Entwicklungsprojekt von Home Assistant bereitgestellt werden, damit das Entwicklerteam sie bis zur nächsten Freisetzung implementieren und Out-of-the-box zur Verfügung stellen kann. Dies ist aufgrund des Supportdienstes von Home Assistant am wichtigsten. Wenn eine Änderung nicht von den Entwicklern implementiert wird, verschwinden die von dem Benutzer/der Benutzerin vorgenommenen Änderungen im Falle eines Neustarts des Systems über die Web-Oberfläche oder einer neuen Systemfreigabe von Home Assistant.

4.8.5 Externer Zugriff

Es ist von großer Bedeutung, das sichere HTTPS-Protokoll zu verwenden, um auf den Home Assistant außerhalb des Heimnetzwerks zuzugreifen. Wie bei Domoticz, sollten hier auch die Grundeinstellungen wie Portfreigabe von dem WLAN-Router vorgenommen werden. Der grundlegende Zugangsport für Home Assistant ist 8123, welche als externes Port 443 (HTTPS) freigegeben werden soll. Home Assistant bietet DDNS-Add-Ons von verschiedenen Anbietern wie No-IP oder DuckDNS an, die einfach über die Web-Oberfläche installiert werden können. Die Anzahl der verwendbaren DDNS ist im Gegensatz zu Domoticz geringer, da bei Hass.io nur die Out-of-the-box Add-Ons verwendet werden können. Für die Masterarbeit wurde das DuckDNS-Add-On ausgewählt, da dies ein kostenloses DDNS anbietet. Nach der Installation muss in der Konfigurationssektion des Add-Ons ein Token bzw. der Domainname, der vom DDNS-Anbieter bereitgestellt wird, angegeben werden. Eine weitere Einstellung ist in der Konfigurationsdatei des Home Assistant vorzunehmen. Unter einer neuen Variablen namens http soll die Variable ”base_url” und der erstellte

Domainname als Wert angegeben werden. Weiters sollen unter diesem Punkt der Pfad des SSL-Zertifikates (`/ssl/fullchain.pem`) bzw. des SSL-Schlüssels (`/ssl/privkey.pem`) hinzugefügt werden. Das SSL-Zertifikat bzw. der Schlüssel können mit dem auch bei Domoticz verwendeten Add-On namens Let's Encrypt erstellt werden. Zur Erstellung der Dateien werden eine E-Mail-Adresse und ein Domainname gebraucht. Hass.io muss neugestartet werden und somit steht Home Assistant unter dem angegebenen Domainname außerhalb des Heimnetzwerks zur Verfügung.

5 Diskussion

Um eine bessere Übersicht über die beiden zuvor installierten Hausautomatisierungsplattformen zu erhalten, sollen ihre wesentlichen Eigenschaften bzw. die notwendigen Maßnahmen verglichen werden (Khusnutdinov, Usachev, Mazzara, Khan & Panchenko, 2018, pp. 46-49). Die Gegenüberstellung soll in einer Tabelle zusammengefasst werden, wobei die Unterschiede zwischen den beiden Plattformen bewertet werden sollen. Aufgrund der Demonstration des System- und Plattformbetriebs wurde in der Masterarbeit ein Smart Home nur teilweise aufgebaut. Zukünftige Verbesserungen des integrierten ZigBee-Netzwerks und der Hausautomationsplattform sollen ebenso vorgestellt werden.

5.1 Installationsverfahren

Um ein ZigBee-Netzwerk als Smart Home zu betreiben, ist eine geeignete Plattform notwendig, worüber die Benutzerin/der Benutzer jedes Gerät im Netzwerk überwachen bzw. steuern kann. Der CC2531 USB-Stick und die zugehörige Zigbee2MQTT-Software können auf jedem Computer oder Betriebssystem ausgeführt werden und haben geringe Hardwareanforderungen. Zum Ausführen des Systems ist ein MQTT-Broker erforderlich, der jedoch von einem separaten Computer aus ausgeführt werden kann. Ziel des Masterarbeitsprojekts war jedoch, das Smart Home System zu einem Master-Computer zu machen, auf dem jede einzelne Software ausgeführt wird, wodurch die Hardware- und Energiekosten reduziert werden. Die beiden getesteten Hausautomatisierungsplattformen haben unterschiedliche Systemanforderungen. Die Hausautomationsplattform von Domoticz stellt nur geringe Systemanforderungen, somit reicht für die Ausführung bereits die schwächste Version des Raspberry Pi, der Raspberry Pi Zero aus. Der Speicherbedarf beträgt laut Dokumentation 8 GB. Im Gegensatz dazu benötigt Home Assistant mindestens 32 GB Speicherplatz und obwohl Home Assistant ab Raspberry Pi Version 1 unterstützt wird, wird der Version 3 oder höher empfohlen. Obwohl laut Dokumentation so viel Speicherplatz benötigt wird, wurden während des zweiwöchigen Testlaufs nur 1.5 GB Festplattenspeicher verwendet.

Das Installationsverfahren ist für die beiden Plattformen völlig unterschiedlich. Für die Installation von Domoticz ist ein vorhandenes Betriebssystem erforderlich, für

den Raspberry Pi ist zurzeit das Linux basierte Raspbian OS notwendig. Der Benutzer/die Benutzerin muss über Linux-Kenntnisse verfügen, um diese Plattform in Betrieb nehmen zu können. Da es sich um ein Betriebssystem mit vollem Zugriff handelt, hat die Benutzerin/der Benutzer vollen Zugriff auf das Stammverzeichnis des Systems, somit ist die Installation und Konfiguration der Komponenten im Laufe des Betriebs uneingeschränkt möglich. Im Gegensatz zu Hass.io ist der drahtlose Internetzugang sogar im Laufe des Betriebs konfigurierbar. Die Domoticz Hausautomatisierungsplattform und die weitere Software wie MQTT Broker und Zigbee2MQTT können als Service auf dem Raspbian installiert werden, damit sie beim Hochfahren des Raspberry Pis automatisch gestartet werden. Jede Software arbeitet unabhängig von Domoticz und kann über SSH oder Peripheriegeräte konfiguriert bzw. überwacht werden. Sowohl Programm- als auch Betriebssystemupdates müssen separat und manuell behandelt werden, da dieses über die Web-Oberfläche nicht gemacht werden können. Eine weitere Option für den Zugriff auf Konfigurationsdateien wäre, ein Datenfreigabesystem namens Samba einzurichten, wodurch alle Dateien von einem Windows-Rechner erreicht werden können. Obwohl Samba auch unter Raspbian ausgeführt werden kann, ist die Einrichtung jedoch viel komplizierter, als bei Hass.io, wo diese über ein Add-On gemacht wird. Eine offizielle Wiki-Seite und Community-Foren unterstützen die Benutzer/Benutzerinnen bei Domoticz, jedoch ist die Wiki-Seite etwas veraltet, da sie zuletzt am 6. März 2018 aktualisiert wurde (Stand 15.02.2020).

Home Assistant verfügt über ein eigenes Betriebssystem, Hass.io, das für eingebettete Geräte wie den Raspberry Pi optimiert ist. Seine Installation ist einfach, die dem Gerät entsprechende Image-Datei muss auf eine SD-Karte geflasht werden. Falls ein Internetzugang vorhanden ist, werden die neuesten Updates beim ersten Einschalten automatisch heruntergeladen und Hass.io ist dann an Port 8123 des Raspberry Pis verfügbar, der beispielsweise in einem Browser geöffnet werden kann. Da Hass.io kein Standard-Linux-Betriebssystem ist, sondern eine Methode zum Installieren und Ausführen von Home Assistant, können die grundlegenden Linux-Befehle nicht ausgeführt werden. Dadurch wird das Stammverzeichnis des Systems für den Benutzer/die Benutzerin eingeschränkt. Eine Folge davon ist, dass das drahtlose Netzwerk nur entweder während der Installation oder während des Betriebs durch einen externen USB-Stick eingerichtet werden kann. Die Möglichkeiten eines Smart Homes, das auf dem Hass.io-Betriebssystem basiert, hängen von den verfügbaren Add-Ons ab. Jede verfügbare Software und alle Systeme werden vom Hersteller als Add-Ons bereitgestellt, darüber hinaus können keine zusätzlichen Komponenten direkt auf dem Operationssystem installiert werden. Dadurch wird sichergestellt, dass nur vom Hersteller als vertrauenswürdig eingestufte Erweiterungen installiert und verwendet werden können, was die Sicherheit erhöht, indem die Wahrscheinlichkeit verringert wird, dass schädliche Programme zufällig installiert werden. Alle Add-Ons werden im Bereich Supervisor verwaltet, d.h. sie können einfach über die Web-Oberfläche aktualisiert werden. Falls ein neues Update von Hass.io zur Verfügung steht, wird der Benutzer/die Benutzerin benachrichtigt. Die zugehörigen Konfigurationsdateien, einschließlich jener zum Protokoll, befinden sich auf einer Registerkarte des ent-

sprechenden Add-Ons. Die Dokumentation für Hass.io ist aktueller als jene für Domoticz, das letzte Update war am 11. Februar 2020 (Stand 15.02.2020). Im Falle eines neuen Updates wird der Benutzer/die Benutzerin auf die Versionshinweise sowohl auf der Home Assistant-Web-Oberfläche als auch auf der offiziellen Hass.io-Website informiert. Weiters gibt es ein weit verbreitetes, offizielles Community-Forum, in dem Fragen und Kommentare zu verschiedenen Themen gepostet werden können. Gemessen an der Anzahl der Beiträge in diesem Forum verfügt Home Assistant über eine viel aktivere Gemeinschaft als Domoticz. Aus den vorher erwähnten Gründen waren die bei dem Projekt der Masterarbeit installierten Komponenten, der MQTT Broker, und die Zigbee2MQTT-Software, unter Hass.io schneller und einfacher zu installieren, zu konfigurieren bzw. zu aktualisieren als unter Raspbian und ist daher eher für die/den durchschnittlichen Benutzerin/Benutzer geeignet. Bei Domoticz ist der Benutzer/die Benutzerin uneingeschränkt, benötigt jedoch Linux-Kenntnisse, da alles unter dem Raspbian-Betriebssystem über Linux-Befehle eingerichtet werden muss.

5.2 Web-Oberfläche

Das Wesentliche der beiden Plattformen ist die Webbenutzeroberfläche, worüber jedes der angeschlossenen Geräte gesteuert werden kann. Auf das Interface kann über einen Browser zugegriffen werden und es ist auch mit mobilen Geräten kompatibel. Alle Domoticz-Einstellungen sind ausnahmslos im Setup Menüpunkt verfügbar. Unter dem Menüpunkt „Devices“ werden alle verfügbaren Daten zu den Geräten aufgelistet, die dem ZigBee-Koordinator hinzugefügt wurden, zum Beispiel Temperatur, Fenstersensorstatus, Batteriestand oder LED-Schalter, die separat mithilfe eines Knopfes zu den Dashboards hinzugefügt werden können. Nach dem Hinzufügen werden die Geräte in Hauptmenüelemente wie Schalter oder Sensoren eingeteilt, und jedes hat seine eigene Karte. Hier besteht weiterhin die Möglichkeit, die bevorzugten Geräte mit dem Sternsymbol zu markieren und dadurch direkt zur Hauptseite hinzuzufügen (siehe Abb. 5.1).

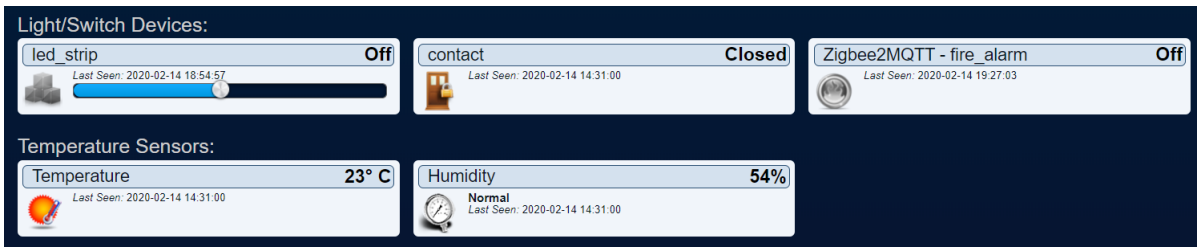


Abbildung 5.1: Dashboard der Domoticz Hausautomatisierungsplattform mit den wichtigsten Daten der verbundenen Geräte.

Wie auf dem Bild ersichtlich, ist das Standardthema von Domoticz dunkel und leicht veraltet, aber es sind zusätzliche Themen verfügbar, mit denen die Oberfläche verschönert werden können. Die Geräteanzeigekarte ist für jedes Gerät gleich und kann nicht personalisiert werden. In dieser Konfiguration ist der LED-Streifen (led_strip) das einzige Gerät, an welches ein Befehl gesendet werden kann. Auf der Lampenkarte können nur der Zustand und die Leuchtdichte geändert werden. Die Farbe des Streifens kann in einem separaten Popup-Fenster mit einem RGB-Farbwähler geändert werden. Die anderen Geräte senden die Daten automatisch an den ZigBee-Koordinator, wodurch diese auf der Web-Oberfläche angezeigt werden, jedoch mit einer Zeitverzögerung von ungefähr 3 Sekunden.

Die Einstellungen in Home Assistant sind nicht so zentralisiert wie in Domoticz. Sicherheits- und Benutzereinstellungen sind in einem separaten Menüelement verfügbar. Die meisten Optionen sind unter Konfiguration verfügbar, jedoch gibt es einige, die in der Benutzeroberfläche nicht zur Verfügung stehen. Diese müssen zur Datei configuration.yaml hinzugefügt werden, wofür, obwohl alles gut dokumentiert ist, YAML-Kenntnisse erforderlich sind. Da die YAML-Sprache nicht zu den einfachen Markup Sprachen gehört, ist dies ein großer Nachteil für die Benutzer/Benutzerinnen, die über geringere Informatik-Kenntnisse verfügen. Mit dem Koordinator verbundene Geräte sind in einer Tabelle unter dem `Configuration\Devices` Menüpunkt ersichtlich. Durch Auswahl eines Geräts wird ein neues Fenster geöffnet, in dem mehrere Optionen ausgewählt werden können. Einige Gerätefunktionen wie Batteriestand, Rauchmelderstatus oder Temperatur sind auswählbar. Für diese Entitäten kann ein Anzeigename vorgegeben und dem Dashboard der Benutzeroberfläche als Karte hinzugefügt werden. Im Gegensatz zu Domoticz kategorisiert Home Assistant nicht, sondern alle Geräte und Entitäten werden gleichbehandelt. Das Dashboard kann später personalisiert werden, um beispielsweise separate Registerkarten für separate Räume zu erstellen.

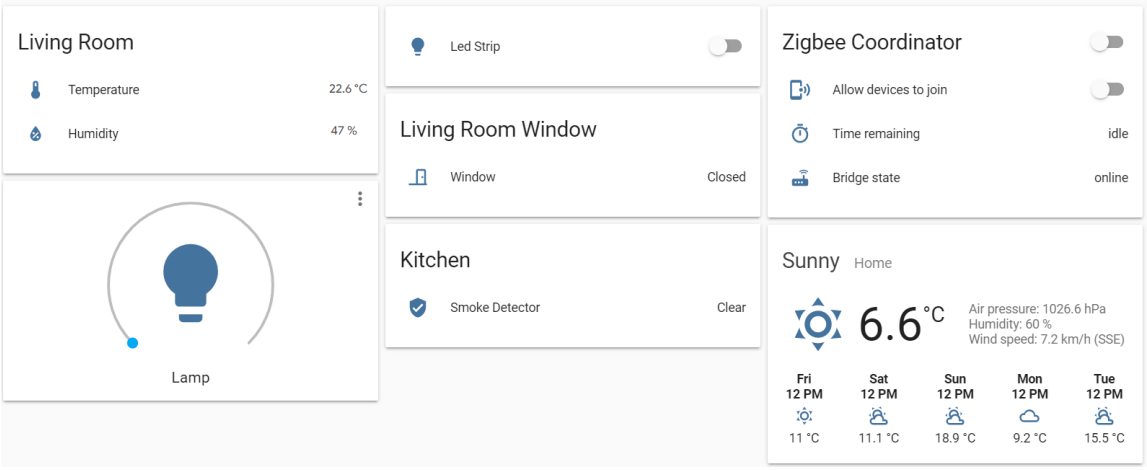


Abbildung 5.2: Dashboard der Home Assistant Hausautomatisierungsplattform mit den wichtigsten Daten der verbundenen Geräte.

Das obere Bild (Abb. 5.2) zeigt das für das Masterarbeitsprojekt erstellte Home Assistant-Dashboard. Das Standardthema namens Lovelance sieht modern aus, verwendet helle Farben und einzigartige Symbole. Es gibt in Community-Foren unzählige Open-Source-Themen, womit das Standardthema und die einzelnen Komponenten durch entsprechende CSS-Dateien (Cascading Style Sheets) geändert werden können. Einzelne Karten können auch angepasst werden, wie beispielsweise auf dem obigen Dashboard die zwei Arten von Karten für LED-Streifen, nämlich Lamp und LED Strip. Wenn der Standort bestimmt wird, wird automatisch eine Wetterkarte für das aktuelle und zukünftige Wetter generiert. Wie auf dem Bild ersichtlich, ist die Farbauswahl des LED-Streifens im Grunde nicht Teil der Karte. Dies erfolgt ebenso in einem Popup-Fenster mit einem RGB-Farbwähler. Die Home Assistant Reaktionszeit ist sehr schnell, es gibt fast keine Zeitverzögerung. Bei einem Smart Home ist es meistens der Fall, dass es von mehr als einer Person verwendet wird. Aus diesem Grund ist es wichtig, mehrere Benutzer/Benutzerinnen mit unterschiedlichen Berechtigungen anzulegen. Auf beiden Plattformen ist dies einfach über die Web-Oberfläche zu machen. Der Benutzer/die Benutzerin kann sich dann mit seinem/ihrem eigenen Benutzernamen und Passwort über einen Webbrowser anmelden. Dieses System ist etwas weiter fortgeschritten beim Home Assistant, da die Benutzeroberfläche und sogar die Anzeigesprache angepasst werden können. Beide Plattformen unterstützen auch die Grundrissansicht, wodurch die Steuerung und Überwachung von Geräten in Mehrraumhäusern erleichtert werden. Dies zu bauen ist jedoch nicht Gegenstand der Masterarbeit, da nur wenige Geräte und alle davon in dem gleichen Raum installiert wurden. Wenn das System aus irgendeinem Grund fehlerhaft funktioniert oder die Dateien beschädigt sind, kann ein Backup helfen, einen früheren, noch funktionsfähigen Zustand wiederherzustellen. Während Domoticz über eine integrierte automatische Sicherung verfügt, stellt Home Assistant diese Funktionalität über ein Add-On namens Auto Backup zur Verfügung, mit dem der Prozess automatisiert werden kann. Backups werden grundsätzlich auf der Raspberry Pi SD-Karte gespeichert, daher ist es wichtig, sie auf einen externen Speicher zu kopieren oder auf einen Datei-Hosting-Service wie Google Drive hochzuladen. Home Assistant verfügt außerdem über eine Snapshot-Funktion, die manuell über das Dashboard erstellt werden kann. Wenn etwas schief geht, kann das Hass.io neu installiert und durch den Snapshot wiederhergestellt werden.

5.3 Automatisierungsmöglichkeiten

In der Welt der Smart Homes und des Internets der Dinge ist es neben von Menschen gesteuerten Geräten unerlässlich, Smart Devices zu automatisieren, da dadurch das Zuhause komfortabler, sicherer und energieeffizienter gemacht wird. Ein effizientes Smart Home kann Dinge nicht nur nach von einem Menschen gegebenen Befehlen, sondern auch durch Daten, die von anderen Geräten gesendet wurden ausführen. Einer der wichtigsten Teile von Hausautomationsplattformen ist die intelligente Auto-

matisierung, die automatisch auf der Grundlage einer vordefinierten Logik selbständig ausgeführt wird. Bei Domoticz können zwei Arten von Automatisierungsmethoden unterschieden werden. Eine davon funktioniert über eine blockbasierte visuelle Programmierung namens Blocky, wodurch einfachere Automatisierungsprozesse direkt über die Web-Oberfläche ermöglicht werden. Auf diese Weise können mehrstufige, bedingungsbasierte Befehle ausgeführt werden. Blocky ist einfach zu bedienen und erfordert keine Programmierkenntnisse. Die folgende Abbildung (Abb. 5.3) zeigt eine mit Blocky erstellte Automatisierung, die eine E-Mail sendet, wenn das Fenster um 21 Uhr noch geöffnet ist. Um eine E-Mail zu senden, müssen die erforderlichen Informationen wie die E-Mail-Adresse des Absenders, der E-Mail-Server und der zugehörige Port sowie der Benutzername und das Kennwort, die der E-Mail des Absenders zugeordnet sind, voreingestellt werden. Diese Einstellungen des E-Mails können unter dem Menüpunkt „Benachrichtigung“ vorgenommen werden.

Abbildung 5.3: Visuelle Programmiersprache Blocky. Erstellung einer automatischen E-Mail-Nachricht, wenn das Fenster im Wohnzimmer um 21 Uhr geöffnet ist.

Obwohl Blocky viel automatisieren kann, muss eine Skriptsprache verwendet werden, wenn komplexere Automatisierungen mittels Schleifen und mehreren separaten Einheiten erstellt werden sollen. Domoticz bietet zwei Arten von integrierten Funktionen: LUA und die erweiterte Version DzVentz, die mehr integrierte Methoden und Funktionen bieten. Darüber hinaus können andere Sprachen wie Python, Bash oder PHP verwendet werden (Domoticz, 2017).

Home Assistant verfügt nicht über eine visuelle Programmiersprache, die Automatisierung erfolgt stattdessen über eine YAML-Datei. Eine Automation besteht aus drei Teilen: Trigger, Bedingung und Aktion, wobei Bedingung optional ist. Der Trigger ist das Ereignis, welches den Prozess auslöst. Ausgehend vom vorherigen Beispiel ist dies der Fall, wenn das Fenster geöffnet ist. Die zweite Variable ist die Bedingung, ohne die der Prozess nicht ausgeführt werden kann, welches im Beispiel die Zeit ist. Der letzte Teil ist die Aktion, im Beispiel E-Mail versenden, welche erst dann ausgeführt wird, wenn der Trigger ausgelöst wird und alle Bedingungen erfüllt sind. Abschnitt 4.8.4 zeigt bereits die Details einer Art von Automatisierung, die in eine YAML-Datei geschrieben wurde und die Geräteverbindungsberechtigung über eine Karte bzw. eine Umschalttaste verwaltet. Home Assistant verfügt außerdem über einen integrierten Automatisierungseditor, auf den über die Web-Oberfläche zugegriffen werden kann. Dies ist ein dreiteiliger Editor, in dem die Trigger-, Bedingungs- und Aktionswerte angegeben werden müssen. Der Editor ist jedoch eher begrenzt. Das obige Beispiel für das Senden von E-Mails kann damit nicht gelöst werden. In diesem Editor können einfachere Automatisierungen erstellt werden, die in der Da-

tei namens automations.yaml gespeichert werden. Diese Datei kann beliebig bearbeitet werden und dadurch kann eine im Editor generierte einfache Automatisierung auf eine komplexere mit mehreren Schritten und Funktionen erweitert werden. Wie bei Domoticz ist es auch möglich, Skripte wie zum Beispiel Python zu erstellen, die als Service ausgeführt werden können (Domoticz, 2017). Außerdem stehen Add-Ons zur Verfügung, wie Visual Studio Code, wodurch die Funktionen von Visual Studio auch verwendet werden können, was die Arbeit mit den YAML-Files erleichtert. Eine weitere visuelle Programmierumgebung ist NODE-RED, mit der größere komplexe Automatisierungsprozesse erstellt werden können. Node-RED besteht aus Prozessen und Knoten. Knoten sind Prozesse, die bereits mit einer der vordefinierten Funktionen von Node-RED erstellt wurden. Knotenfunktionen sind in die folgenden Grundkategorien eingeteilt: Eingabe, Ausgabe, Funktion, Sozial, Speicher, Zeit und Analyse. Wenn es als Add-On im Home Assistant installiert ist, gibt es zwölf weitere Knoten, die auf die Funktionen von Home Assistant spezialisiert sind. Für Domoticz muss Node-Red auf dem Raspbian-Betriebssystem installiert sein, wodurch es dann automatisch als Dienst ausgeführt werden kann. Das Automatisierungsverfahren ist für beide Plattformen völlig unterschiedlich. Das in Domoticz integrierte Blocky ist benutzerfreundlich und leicht zu erlernen. Die einfachen Automatisierungen, die eineeinen durchschnittlichedurchschnittlichen BenutzerinBenutzer benötigt, können leicht damit durchgeführt werden. Es gibt kein einfaches Automatisierungsverfahren für Home Assistant, entweder muss eine YAML-Datei bearbeitet werden, oder eine externe Skriptsprache wie Python oder Node-RED verwendet werden.

5.4 Architektur

Die Struktur des aufgebauten Smart Home ist in Abbildung 5.4 dargestellt. Jedes Gerät sendet Daten über das ZigBee-Kommunikationsprotokoll, welches von der Zigbee-2MQTT-Software im JSON-Format veröffentlicht wird, über das MQTT Kommunikationsprotokoll an ein Topic des Mosquitto MQTT Brokers. Der Client der Hausautomationsplattform hat dasselbe Topic abonniert und empfängt die Daten auch über MQTT. Die Zigbee2MQTT-Software, der MQTT Broker und die Hausautomationsplattform laufen alle auf dem Rasberry Pi. Der Benutzer/die Benutzerin greift auf die Plattform beispielsweise über einen Webbrowser zu, worüber die Daten der Sensoren angezeigt werden. Wenn die Benutzerin/der Benutzer an einem Endgerät einen Befehl senden will, zum Beispiel die Farbe des LED-Streifens ändern, läuft der Vorgang in die entgegengesetzte Richtung.

Es wurden verschiedene Maßnahmen ergriffen, um das System sicher zu halten. Für das ZigBee-Protokoll ist es wichtig, die Geräteverbindungsberechtigung falsch zu halten, da sonst jedes Gerät jederzeit eine Verbindung zum gesamten ZigBee-Netzwerk herstellen kann. Eine weitere Änderung besteht darin, den standardmäßigen ZigBee-Netzwerkschlüssel zu ändern. Dies ist ein 16-stelliger Wert zwischen 0 und 255. Beide

Einstellungen können in der Datei configuration.yaml von Zigbee2MQTT geändert werden. Das MQTT-Netzwerk ist nicht vollständig sicher, da es nicht mit TLS / SSL gesichert ist. Es wird jedoch eine grundlegende Sicherheit verwendet, da auf den MQTT-Broker nur mit einem Benutzernamen und einem Kennwort zugegriffen werden kann. Der letzte Schritt zur Sicherung besteht darin, den Zugriff außerhalb des Heimnetzwerks ausschließlich über HTTPS zur Verfügung zu stellen. Der standardmäßige externe HTTP-Zugriff ist deaktiviert.

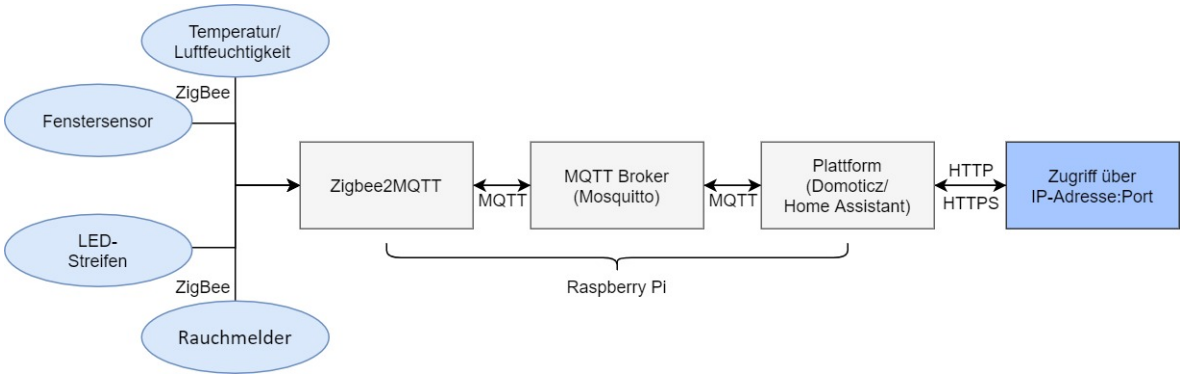


Abbildung 5.4: Architektur des aufgebauten Smart Homes. Das Bild zeigt auch das Kommunikationsprotokoll zwischen den verschiedenen Geräten. Wie ersichtlich ist, laufen die drei mit Grau markierten Teile, Zigbee2MQTT-Software, MQTT Broker und Hausautomationssoftware auf dem Raspberry Pi.

5.5 Mobile Applikationen

In der heutigen, auf Smartphones ausgerichteten Welt ist es wichtig, ein Smart Home mobil überwachen zu können und die zugehörigen Geräte von einem Smartphone aus steuern zu können. Obwohl beide Hausautomationsplattformen eine mobile Ansicht über einen Webbrowser bieten, verfügen sie auch über eine Android- bzw. iOS-Anwendung. In beiden Fällen ist die Einrichtung der Applikation einfach. Nach der Eingabe der IP-Adresse, des Ports, des Benutzernamen und des Passworts sind beide Applikationen einsatzbereit. Domoticz bietet zwei Anwendungen, eine kostenlose Lite- und eine kostenpflichtige Premium-Version an. Die Lite-Version bietet eine erheblich eingeschränkte Unterstützung für Sensordaten und die Kontrolle über intelligente Geräte. Das Layout der Domoticz Applikation (siehe Abb. 5.5), welches modern und mit hellem Thema gestaltet ist, spiegelt auf keine Weise die altmodische, dunkle Web-Oberfläche wider. Die Premium-Version unterstützt eine Vielzahl weiterer integrierter Funktionen, einschließlich Android Wear für den Smart Home-Zugriff. Sie ist außerdem mit einer eindeutigen Fingerabdruckidentifikation, Widgets, Benachrichtigungen bzw. mit administrativen Funktionen ausgestattet. Der einmalige Preis beträgt sieben Euro. Home Assistant hat nur eine kostenlose Anwendung mit allen verfügbaren Funktionen, jedes installierte Add-On und alle Konfigurationsdateien

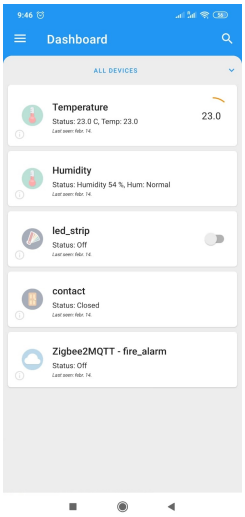


Abbildung 5.5: Startseite der Domoticz Lite mobile Applikation.

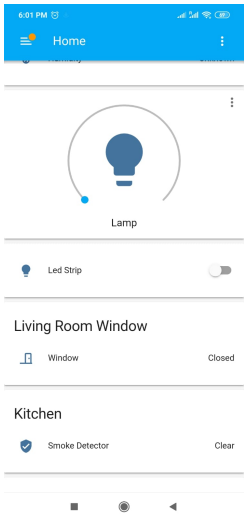


Abbildung 5.6: Startseite der Home Assistant mobile Applikation.

können über das Telefon aufgerufen werden. Zusätzlich zu den auf der Weboberfläche verfügbaren Funktionen kann der Tracking-Modus aktiviert werden. Dieser Modus kann in Verbindung mit der Automatisierung nützlich sein. Ein Beispiel dafür: Wenn kein/keine Benutzer/Benutzerin zu Hause ist, schalten sich alle Lampen aus. Home Assistant verfügt derzeit nicht über eine Android Wear-Integration. Das Thema der Anwendung, wie im Bild unten (Abb. 5.6) gezeigt, ähnelt der Web-Oberfläche und bietet ein modernes, klares Erscheinungsbild. Zusammenfassend bietet die kostenpflichtige Version von Domoticz die meisten Funktionen und bietet mehr als eine reibungslose Weboberfläche. Im Gegensatz dazu ist Home Assistant bis auf manche zusätzlichen Funktionen fast identisch mit seiner Weboberfläche. Die kostenlose Anwendung von Domoticz bietet wenig, kann aber für die/den durchschnittliche/n Benutzerin/Benutzer mit wenigen Smart Geräten ausreichen.

5.6 Ergebnisse

Zur Vereinfachung des Vergleiches ist jeder Hauptpunkt in der Tabelle 5.1 aufgeführt. Da die in der Tabelle aufgeführten Punkte auf beiden Plattformen durchgeführt werden können, werden die Punkte auf einer Skala nach Likert von eins bis drei vergeben, wobei ”1” für ”Sehr Gut”, ”2” für ”Mittelmäßig” und ”3”für ”Sehr Schlecht” steht. Basierend auf dem Verfahren zum Einrichten beider Hausautomationssysteme und zu deren Verwendung für jeweils zwei Wochen, wurden folgende Punkte vergeben. Wie es dem Ergebnis entnommen werden kann, gibt es insgesamt kaum einen Unterschied zwischen den beiden Hausautomationsplattformen, da jede über Vor- und Nachteile verfügt. Die größten Vorteile von Domoticz sind die einfache Automatisierung, und da es auf einem separaten Betriebssystem ausgeführt

wird, die Anpassbarkeit. Hinter diesen Eigenschaften steckt Home Assistant, da eine Automatisierung ohne Kenntnis der YAML-Sprache fast unmöglich ist. Im Gegensatz dazu, erleichtert Home Assistant die Installation zusätzlicher Software, die über das Add-On verfügbar ist. Es ist weiterhin wichtig zu erwähnen, dass das Hass.io-Betriebssystem während der Masterarbeit verwendet wurde. Home Assistant kann genau wie Domoticz auf einem separaten Betriebssystem installiert werden, wodurch die durch Hass.io verursachten Einschränkungen behoben werden können. Wenn eine Benutzerinein Benutzer wenig IT- und Linux-Kenntnisse hat, ist es aus der Sicht der Ergebnisse empfohlen, Home Assistant über das Hass.io Operationssystem zu verwenden.

	Domoticz	Home Assistant
Es ist einfach, das System zu installieren.	2	1
Die notwendigen Anwendungen sind einfach zu installieren.	3	1
Unbegrenzte Einstellungen des Operationssystems.	1	3
Die Konfigurationen sind einfach durchführbar.	2	2
Die Verbindung der Geräte erfolgt über die Web-Oberfläche.	1	1
Die Web-Oberfläche ist gut erreichbar.	1	1
Gutes Design der Web-Oberfläche.	3	1
Alle Einstellung können leicht über die Web-Oberfläche verwaltet werden.	1	2
Die Änderungen werden gleich in der Oberfläche angezeigt.	2	1
Die Plattform verfügt über geeignete Userverwaltung.	1	1
Systembackup ist leicht zu verwalten.	1	1
Der Automatisierungsprozess ist einfach und effektiv.	1	3
Eine mobile Anwendung, die für den/die durchschnittlichen User/in geeignet ist.	1	1
Gesamt	20	19

Tabelle 5.1: Vergleich der Hauptaspekte der beiden Hausautomationsplattformen.

5.7 Zukünftige Möglichkeiten

Das Projekt der Masterarbeit umfasste die Entwicklung eines eigenen ZigBee-Netzwerks mit einer zentralen Einheit, welche auf dem Raspberry Pi läuft und vier zu Demonstrationszwecken verwendeten intelligenten Geräten. Die Topologie des erstellten Netzwerks ist in der Abbildung 5.7 dargestellt. Die Darstellung wurde aus Home Assistant mithilfe eines von einem Benutzer erstellten Services erledigt (Gruebel,

2019). Dieser Service zeichnet den Netzwerk-Plan basierend auf Zigbee2MQTT-Software auf. Wie in dieser Abbildung ersichtlich ist, sind die Geräte in dem ZigBee-Netzwerk in Stern-Topologie angeordnet. Auf den einzelnen Karten sind die grundlegenden Informationen bzw. der letzte Zeitpunkt, wann Daten gesendet oder empfangen worden sind, dargestellt.

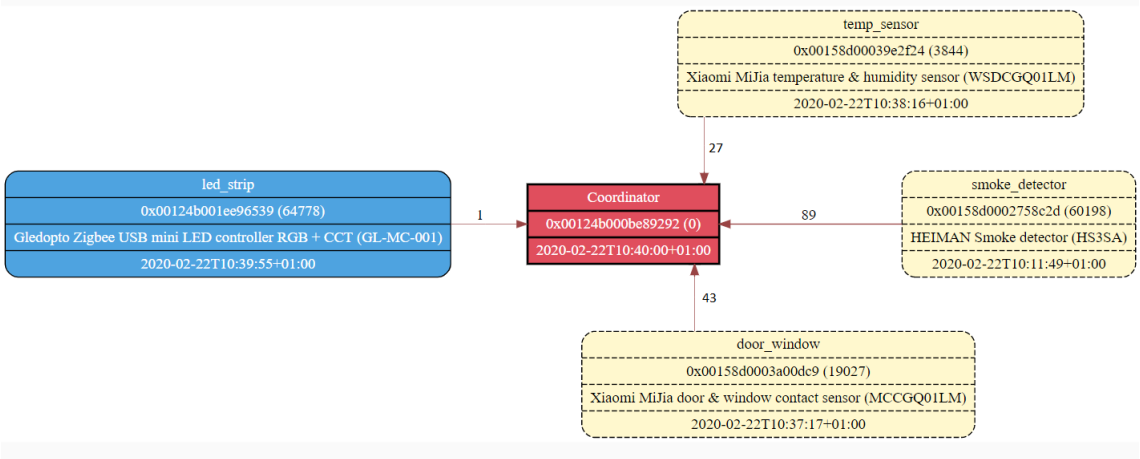


Abbildung 5.7: Netzwerk-Topologie der Geräte in dem ZigBee-Netzwerk.

Es besteht natürlich die Möglichkeit, weitere Geräte ins Smart Home zu integrieren. Das erstellte ZigBee-Netzwerk ist in einer Sterntopologie organisiert, d. h. das Zentrum ist der Koordinator, und alle anderen Geräte sind direkt mit dem Koordinator verbunden. Wie in Abschnitt 4.4 beschrieben, können an den CC2531-Koordinator nur 20 Geräte angeschlossen werden und er hat eine begrenzte Reichweite von bis zu 30 Metern. In Kapitel 3.4 wurden die in ZigBee-Netzwerken verfügbaren Topologien vorgestellt. Da die Clusterbaum-Topologie in industriellen Anwendungen häufiger verwendet wird, ist es ratsam, ein Mesh-Topologie-Netzwerk für ein Smart Home aufzubauen. Dafür kann das aktuelle Netzwerk mit einem oder mehreren Routern erweitert werden. Dies kann entweder ein CC2531-USB-Stick sein, auf dem die Router-Firmware installiert ist, oder andere Geräte mit ZigBee-Technologie, die die Rolle eines Routers spielen können. In den meisten Fällen handelt es sich um Geräte mit einer stabilen Stromquelle, wie zum Beispiel Glühbirnen oder Steckdosen. Wenn intelligente Geräte in mehr als einem Raum verwendet werden, kann es nützlich sein, Gruppen zu definieren, zu denen Geräte hinzugefügt werden können. Das Wesentliche an Gruppen ist es, dass die zugehörigen Geräte gleichzeitig gesteuert werden, so dass beispielsweise alle Glühbirnen im Wohnzimmer zur selben Zeit ausgeschaltet werden können. Eine weitere Funktion zur Überwachung mehrerer Räume ist der Gebäudeplan, wodurch der Status von Geräten und Räumen leicht überwacht werden kann. Wie bereits erwähnt, war bei ZigBee ein geringer Stromverbrauch wichtiger als die Datenübertragungsgeschwindigkeit, welche nur bis zu 250 Kb/s beträgt. Aus diesem Grund können nicht alle Gerätetypen für ZigBee produziert werden. Beide Hausautomationssysteme sind jedoch unabhängig vom ZigBee-Netzwerk, sodass für Videostreams verwendete WLAN-Geräte oder im Gesundheitswesen häufig

verwendete Bluetooth-Geräte verbunden und gesteuert werden können. Wenn die Daten für spätere Analysen benötigt werden, können sie in die Datenbank gespeichert oder sogar in die Cloud hochgeladen werden. Home Assistant speichert die Daten standardmäßig in einer internen SQLite Datenbank, die alle Daten lokal auf der SD-Karte des Raspberry Pi speichert. Die Daten werden für 10 Tage gespeichert, danach werden sie automatisch gelöscht. Home Assistant verfügt weiters out-of-the-box über eine externe Influx- oder MariaDB-Datenbankintegration, mit der Daten einfach gespeichert werden können. Domoticz verfügt ebenso über eine interne SQLite-Datenbankintegration. Hier werden die Daten jedoch nicht automatisch gelöscht. Aus diesem Grund ist es bei Domoticz notwendig, die Größe der Datenbank zu überwachen, die von Tag zu Tag wächst.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Abdmeziem, M., Tandjaoui, D. & Romdhani, I. (2015). Architecting the internet of things: State of the art. In (S. 55-75).

Abomhara, M. & Køien, G. (2015). Cyber security and the internet of things: Vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security*, *4*, 65-88. doi: 10.13052/jcsm2245-1439.414

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols and applications. *IEEE Communications Surveys and Tutorials*, *17*, Fourthquarter 2015. doi: 10.1109/COMST.2015.2444095

Ashton, K. (2009). That 'internet of things' thing. *RFID Journal LLC*, *22*, 97-114.

Assistant, H. (2020). *Home assistant.* Zugriff am 10. Jan. 2020 auf `https://www.home-assistant.io/`

Atzori, L., Iera, A. & Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, *54* (15), 2787 - 2805.

AVSystem. (2019). *Iot vs m2m - what is the difference?* Zugriff am 17. Sep. 2019 auf `https://www.avsystem.com/blog/iot-and-m2m-what-is-the-difference/`

Badach, A. & Hoffmann, E. (2019, 11.). In (S. 1-48). Carl Hanser Verlag GmbH & Co. KG. (0) doi: 10.3139/9783446455115.001

Banks, A., Briggs, E., Borgendale, K. & Gupta, R. (2019). Mqtt version 5.0 - oasis standard. Zugriff auf `http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf`

Bilal, M. (2017). A review of internet of things architecture, technologies and analysis smartphone-based attacks against 3d printers. In (S. 1-21).

Craig, W. (2004). Zigbee : " wireless control that simply works "..

Deutsche Telekom AG. (2018). *5g speed is data transmission in real time.* Zugriff am 20. Feb. 2020 auf `https://www.telekom.com/en/company/details/5g-speed-is-data-transmission-in-real-time-544498`

Digi International Inc. (2019). *Channels, zigbee.* Zugriff am 21. Dec. 2019 auf `https://www.digi.com/resources/documentation/digidocs/90001537/references/r_channels_zigbee.htm`

Digi International Inc. (2020). *Digi.* Zugriff am 21. Dec. 2019 auf `https://www.digi.com/`

Dixon, J. (2014). *Http headers and ords rest web services.* Zugriff am 17. Sep. 2019 auf https://smartdogservices.com/http-headers-and-ords-rest-web-services/

Domoticz. (2017). *Automation - domoticz.* Zugriff am 14. Jan. 2020 auf https://www.domoticz.com/wiki/Automation

Domoticz. (2020). *Domoticz.* Zugriff am 10. Jan. 2020 auf https://www.domoticz.com/

Eady, F. (2007). *Hands-on zigbee: Implementing 802.15.4 with microcontrollers (embedded technology)*. USA: Newnes.

Eclipse Foundation Inc. (2018). *Iot developer survey - 2018.* Zugriff am 17. Sep. 2019 auf https://iot.eclipse.org/resources/iot-developer-survey/iot-developer-survey-2018.pdf

Eclipse Foundation Inc. (2019). *Iot developer survey - 2019.* Zugriff am 17. Sep. 2019 auf https://iot.eclipse.org/resources/iot-developer-survey/iot-developer-survey-2019.pdf

Eclipse Foundation Inc. (2020). *Eclipse.* Zugriff am 17. Sep. 2019 auf https://www.eclipse.org/

Elahi, A. & Gschwender, A. (2009). *Introduction to the zigbee wireless sensor and control network* (1st Aufl.). Prentice Hall.

Enterprise, I. D. N., Micro, R. I. G. & systems in: Co-operation with the Working Group RFID of the ETP EPOSS, N. (2008). Internet of things in 2020: Roadmap for the future. In (Bd. 1.1). Zugriff auf https://docbox.etsi.org/erm/Open/CERP%2020080609-10/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v1-1.pdf

Ergen, S. C. (2004). Zigbee/ieee 802.15.4 summary. In (S. 2).

Farahani, S. (2008). *Zigbee wireless networks and transceivers*. USA: Newnes.

Fielding, R. & Gettys, J. (1999). *Hypertext transfer protocol - http/1.1.* Zugriff am 17. Sep. 2019 auf https://tools.ietf.org/pdf/rfc2616.pdf

G., O. (2015). A survey of zigbee wireless sensor network technology: Topology, applications and challenges. *International Journal of Computer Applications*, *130*, 47-55. doi: 10.5120/ijca2015907130

Grassi, P. A., Newton, E. M. & Fenton, J. L. (2017). Nist special publication 800-63b - digital identity guidelines. Zugriff auf https://pages.nist.gov/800-63-3/sp800-63b.html

Grigorik, I. (2013). *High performance browser networking: What every web developer should know about networking and web performance*. O'Reilly Media.

Gruebel, R. (2019). *Zigbee2mqtt networkmap.* Zugriff am 14. Jan. 2020 auf https://github.com/rgruebel/ha_zigbee2mqtt_networkmap

Götz, C., Obermaier, D., Pfefferle, D., Skerrett, I. & Raschbichler, F. (2015a). *Last will and testament.* Zugriff am 17. Sep. 2019 auf https://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament/

Götz, C., Obermaier, D., Pfefferle, D., Skerrett, I. & Raschbichler, F. (2015b). *Quality of service 0,1 and 2.* Zugriff am 17. Sep. 2019 auf https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

Gupta, L. (2019). *Http methods.* Zugriff am 17. Sep. 2019 auf https://restfulapi.net/http-methods/

Győrödi, C., Gyorodi, R., Pecherle, G. & Olah, A. (2015). A comparative study: Mongodb vs. mysql.. doi: 10.13140/RG.2.1.1226.7685

Horvath, I. (2017). *Számítógép hálózatok kiépítése - kommunikációs protokollok* (Bd. 3).

Horyachyy, O. (2017). *Comparison of wireless communication technologies used in a smart home: Analysis of wireless sensor node based on arduino in home automation scenario* (Unveröffentlichte Dissertation). Blekinge Institute of Technology.

Ieee standard for low-rate wireless networks. (2016). *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, 1-709. doi: 10.1109/IEEESTD.2016.7460875

Jia, X., Feng, Q., Fan, T. & Lei, Q. (2012). Rfid technology and its applications in internet of things (iot). *2012 2nd International Conference on Consumer Electronics, Communications and Networks, CECNet 2012 - Proceedings*. doi: 10.1109/CECNet.2012.6201508

Jobbagy, S. (2016). Wlan hálózatról röviden. In (S. 306-308).

Kaiser, G. (2015). *ecourse on integrated home systems*. Leonardo ENERGY.

Khan, R., Khan, S., Zaheer, R. & Khan, S. (2012). Future internet: The internet of things architecture, possible applications and key challenges. In (S. 257-260). doi: 10.1109/FIT.2012.53

Kühner, J. (2009). *Expert .net micro framework*. Apress.

Khusnutdinov, A., Usachev, D., Mazzara, M., Khan, A. & Panchenko, I. (2018). Open source platform digital personal assistant. In (S. 46-49).

Krauße, M. & Konrad, R. (2014). *Drahtlose zigbee-netzwerke: Ein kompendium*. Springer Fachmedien Wiesbaden.

Lampkin, V., Leong, W., Olivera, L., Rawat, S., Subrahmanyam, N., Xiang, R., ... others (2012). *Building smarter planet solutions with mqtt and ibm websphere mq telemetry*. IBM Redbooks.

Let's Encrypt - Internet Security Research Group. (2020). *Let's encrypt.* Zugriff am 10. Jan. 2020 auf https://www.letsencrypt.org/

Levin, R. G. (2016). *Restful api authentication basics.* Zugriff am 17. Sep. 2019 auf https://blog.restcase.com/restful-api-authentication-basics/

Mašetić, Z., Kečo, D., Dogru, N. & Hajdarevic, K. (2017). Syn flood attack detection in cloud computing using support vector machine. *TEM Journal*, *6*, 752-759. doi: 10.18421/TEM64-15

MOHAN.K, M., Chandrasekara, B. & K, S. (2014). A power efficient mac protocol for quality of service evaluation in wireless sensor networks. *IJIREEICE*, 2154-2159. doi: 10.17148/IJIREEICE.2014.21107

MongoDB Inc. (2019). *Mongodb and mysql compared.* Zugriff am 17. Sep. 2019 auf https://www.mongodb.com/compare/mongodb-mysql

Morales, J., Lopez, N. A., Parado, J. & Pasaoa, J. (2016). A comparative study of thread against zigbee, z-wave, bluetooth, and wi-fi as a home-automation networking protocol. In (S. 3-4).

Park, Y., Sthapit, P. & Pyun, J.-Y. (2009). Smart digital door lock for the home automation. In (S. 1). doi: 10.1109/TENCON.2009.5396038

Perera, C., Zaslavsky, A. B., Christen, P. & Georgakopoulos, D. (2013). Context aware computing for the internet of things: A survey. *CoRR, abs/1305.0982*. Zugriff auf http://arxiv.org/abs/1305.0982

Postcapes. (2019a). *Internet of things (iot) history.* Zugriff am 17. Sep. 2019 auf https://www.postscapes.com/internet-of-things-history/

Postcapes. (2019b). *Iot standards and protocols.* Zugriff am 17. Sep. 2019 auf https://www.postscapes.com/internet-of-things-technologies/

Prasad, R. (2009). *My personal adaptive global net (magnet)* (1st Aufl.). Springer Publishing Company, Incorporated.

Rashid, M. (2015). Zigbee: Simulation and investigation of star and mesh topology by using different transmission bands. *AIUB Journal of Science and Engineering (AJSE), 14*.

Rautmare, S. & Bhalerao, D. (2016). Mysql and nosql database comparison for iot application. In (S. 235-238). doi: 10.1109/ICACA.2016.7887957

SafeWise Team. (2018). *Mi az a z-wave?* Zugriff am 20. Feb. 2020 auf https://bravosmart.hu/mi-az-a-z-wave/

SafeWise Team. (2020). *Are wired or wireless home security systems better?* Zugriff am 20. Feb. 2020 auf https://www.safewise.com/home-security-faq/wired-vs-wireless-security/

Sharma, K. & Dhir, N. (2014). A study of wireless networks : Wlans , wpans , wmans , and wwans with comparison..

Sharma, R., Gupta, S., Suhas, K. & Kashyap, G. (2014). Performance analysis of zigbee based wireless sensor network for remote patient monitoring. In (S. 58-62). doi: 10.1109/CSNT.2014.21

Silva, B., Khan, M. & Han, K. (2017). Internet of things: A comprehensive review of enabling technologies, architecture, and challenges. *IETE Technical Review*, 1-16. doi: 10.1080/02564602.2016.1276416

Social WiFi. (2017). *What's the difference between 2.4 and 5 ghz wifi?* Zugriff am 07. Jan. 2020 auf https://socialwifi.com/knowledge-base/wifi-technology/difference-between-24-and-5-ghz-wifi/

Varma, U. R. & Bharadwaj, A. S. (2016). Home automation using smart watch..

Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., ... Doody, P. (2009). Internet of things strategic research roadmap..

Vizeli, P. (2019). *Home assistant - network.* Zugriff am 07. Jan. 2020 auf https://github.com/home-assistant/operating-system/blob/dev/Documentation/network.md

Wang, C. (2018). *Http vs. mqtt: A tale of two iot protocols.* Zugriff am 17. Sep. 2019 auf https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols

Wu, M., Lu, T.-J., Ling, F.-Y., Sun, J. & Du, H. (2010). Research on the architecture of internet of things. In (Bd. 5, S. V5-484). doi: 10.1109/ICACTE.2010.5579493

Yokotani, T. & Sasaki, Y. (2016). Comparison with http and mqtt on required network resources for iot. In *2016 international conference on control, electronics, renewable energy and communications (iccerec)* (S. 1-6).