HOCHSCHULE MITTWEIDA
University of Applied Sciences

BACHELORARBEIT

Frau
Carolina da Rocha Nobre

Implementierung eines LoRaWAN-basierten Netzwerks für das Internet der Dinge

Herausforderungen und Perspektiven

2022

HOCHSCHULE
MITTWEIDA
University of Applied Sciences

BACHELOR THESIS

Ms.
Carolina da Rocha Nobre

Implementation of a LoRaWAN-based network for the Internet of Things

Challenges and Perspectives

2022

Fakultät **INW**

BACHELORARBEIT

Implementierung eines LoRaWAN-basierten Netzwerks für das Internet der Dinge

Herausforderungen und Perspektiven

Autorin:
Carolina da Rocha Nobre

Studiengang:
Elektro- und Informationstechnik

Seminargruppe:
EI18WI-B

Erstprüfer:
Prof. Dr.-Ing. Thomas Beierlein

Zweitprüfer:
M.Sc. Andreas Weger

Mittweida, 05 2022

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

µU ..... Mikrokontroller
ABP ..... Activation By Personalisation
ADR ..... Adaptative Data Rate
AES ..... Advanced Encryption Standard
API ..... Application Programming Interface
AS ..... Application Server
BW ..... Bandwidth
CR ..... Coding Rate
CRC ..... cyclic redundancy check
CSS ..... Chirp Spread Spektrum
DSP ..... Digital Signal Processor
ERP ..... Effective radiated power
EUI ..... Extended Unique Identifier
GPIO ..... General-purpose input/output
GPS ..... Global Positioning System
HAL ..... Hardware Abstraction Layer
IoT ..... Internet of Things
IP ..... Internet Protocol
JS ..... Join Server
LAN ..... Local Area Network
LoRa ..... Low Range
LoRaWAN ..... Long Range Wide Area Network
LPWAN ..... Low Power Wide Area Network
LSB ..... Least Significant Bit
LTE ..... Long Term Evolution
M2M ..... Machine-to-Machine
MAC ..... Media Access Control Layer
MHDR ..... Mac-Header
MIC ..... Message Integrity Code
MQTT ..... Message Queuing Telemetry Transport
MSB ..... Most Significant Bit

NS ..... Network Server
OTAA ..... Over The Air Activation
PHDR ..... Physiche Header
PHY ..... Physical Layer
POE ..... Power Over Ethernet
SF ..... Spreading Factor
SPI ..... Serial Peripheral Interface
SRD ..... Short Ranged Devices
SSH ..... Secure Shell
TCP ..... Transmission Control Protocol
UDP ..... User Datagram Protocol
WAN ..... Wide Area Network
WLAN ..... Wireless Local Area Network

1 Einleitung

Das Internet der Dinge (IoT) revolutioniert den IT-Sektor. Schon jetzt haben IoT-Geräte einen erheblichen Einfluss auf unser Leben. Ein spezieller Teilbereich des IoT, das Low Power Wide Area Network (LPWAN), oder auf „Deutsch Weitverkehrsnetz mit geringem Stromverbrauch“, gewinnt stetig an Bedeutung.

Es gibt verschiedene Kommunikationsmethoden, die sich für IoT Geräte eignen. Eine Möglichkeit diese zu kategorisieren ist die Reichweite. Auf Grund ihres geringen Energieverbrauchs werden für kleinere Reichweiten Kommunikationstechnologie wie Bluetooth, ZigBee und Z-Wave von ressourcenbeschränkten IoT-Netzwerken genutzt. Wenn die gewünschte Reichweite in die Kilometerskala geht, sind diese Technologien durch ihre geringe Signalreichweite nicht mehr zu nutzen. Zellulare IoT-Systeme wurden eingesetzt, um dieses Problem zu lösen. Jedoch benötigen diese Systeme eine leistungsfähige Stromversorgung und höhere Betriebskosten.

All dies führte zu einer Lücke in der IoT-Kommunikation, die eine Technologie für eine stromsparende, kostengünstige Kommunikation mit großer Reichweite benötigte. Um diese Lücke zu schließen, wurde LPWAN entwickelt ([4],[26]).

	LAN	LPWAN	WAN
+	• Gute etablierte Standards	• geringer Stromverbrauch • niedrige Kosten • große Reichweite	• bestehende Abdeckung • große Datenrate
-	• kurze Batterielebensdauer • geringe Reichweite	• niedrige Datenrate • nicht etablierte Standards	• kurze Batterielebensdauer • teuer
Bsp.	• Bluetooth • WLAN	• LoRa • SigFox	• GSM • 4G

Tabelle 1.1: LAN, WAN, LPWAN-Vergleich

Dank LPWAN können ressourcenbeschränkte Sensoren- und Aktorensignale bis zu 10 Kilometer weit übertragen werden und bis auf 8–10 Jahre ohne externe Energiequelle in Dauerbetrieb laufen.

Mehrere LPWAN-Technologien sind bereits auf dem Markt. Zum Beispiel SigFox (SigFox Inc. [46]), NB-IoT (3GPP [18]), Weightless (Weightless Special Interest Group [59]), WAVIoT (Waviot Inc. [58]), Nwave (Nwave Technologies Inc.[33]), UNB (Telensa Inc. [50]), RPMA (Ingenu Inc. [22]) und LoRaWAN [2].

LoRaWAN bietet im Gegensatz zu anderen Anbietern die Möglichkeit des Einsatzes privater Netzwerke und der einfachen Integration weltweit verbreiteter Netzwerkplattformen (zum Beispiel TTN [51] und LORIOT [28]). Dazu ist der Aufbau eines LoRaWAN-Netzwerkes günstiger im Vergleich zu anderen LPWAN Technologien. Infolgedessen

und aufgrund seiner offenen Zugangsspezifikationen hat LoRaWAN die Aufmerksamkeit der Forschungsgemeinschaft sowie Bastlern gewonnen [19].

Diese Arbeit soll eine Zusammenfassung verschiedener Aspekte des LoRaWAN-Protokolls darstellen.

1.1 Anwendungen

Es wird erwartet, dass die Anzahl der Anwendungsbereiche von LPWAN mit der Zeit stetig zunehmen werden. Die folgenden Beispiele sind einige mögliche Anwendungen von LoRaWAN:

- Waldbranderkennung [24]
- intelligenten Mülltonnen [21]
- Öl- und Gasbetriebe
- intelligente Straßenlaternen
- Überwachung des Viehbestands
- Überwachung der Landwirtschaft
- drahtlose Sensornetzwerke
- Gesundheit und Wohlbefinden
- Verkehrsüberwachung
- Überwachung des Luftverschmutzungsgrads
- Smart-City-Anwendungen [25]
- intelligenten Stromnetzen und Fernmessungen
- im Falle einer Katastrophe, wie einer Überschwemmung oder einem Flächenbrand, können Drohnen mit LoRa Gateways zur Unterstützung der Retter eingesetzt werden, um Opfer zu lokalisieren und ein Backhaul-Netz aufzubauen, wenn die Kommunikationseinrichtungen unterbrochen sind [48].

1.2 Unterschied zwischen LoRa und LoRaWAN

LoRa (Low Range) und LoRaWAN (Low Range Wide Area Network) sind zwei Begriffe, die oft verwechselt werden. Auch wenn diese Begriffe eng zusammenhängen, sind sie klar voneinander abzugrenzen. LoRa ist die physische Schicht, d.h. die Technik und Modulation, die von SEMTECH entwickelt wurde, um Kommunikationsverbindungen über Langstrecken zu erreichen.

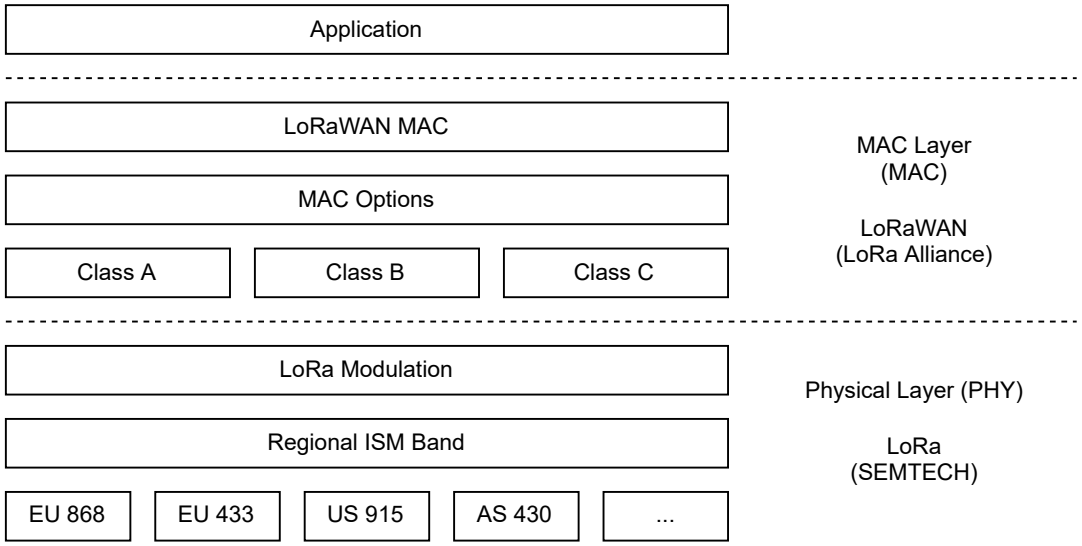

Abbildung 1.1: LoRaWAN Schichten

LoRaWAN definiert das Kommunikationsprotokoll und die Systemarchitektur für das Netzwerk, welche die LoRa Modulation in der physischen Schicht verwendet. Das Protokoll und die Netzwerkarchitektur haben den größten Einfluss bei der Bestimmung der Batterielebensdauer eines Endgerätes, der Netzwerkkapazität, der Servicequalität, der Sicherheit und der Vielfalt der vom Netzwerk bedienten Anwendungen.

2 Einführung in die LoRa Modulation

Dieses Kapitel soll einen detaillierten Überblick über die physikalische Schicht LoRa geben. Darauf aufbauend befasst sich Kapitel 3 dann mit dem LoRaWAN Protokoll.

2.1 Was ist LoRa?

LoRa ist die physikalische Schicht bzw. die Modulation, die verwendet wird, um weitreichende Kommunikationsverbindungen zu schaffen.

Viele drahtlose Systeme verwenden Frequenzumtastung (FSK) als physikalische Schicht. FSK ist eine sehr effiziente Modulation, um einen geringen Energiebedarf zu erreichen. LoRa hat den gleichen niedrigen Energieverbrauch, aber erhöht über dies hinaus die Kommunikationsreichweite deutlich.

Die LoRa Modulation basiert auf der Chirp-Spread-Spektrum (CSS) Modulation. Dabei wird die Datenrate gegen Reichweite oder Leistung innerhalb einer festen Kanalbandbreite eingetauscht. Diese Modulation wird seit Jahrzehnten in der Militär- und Weltraumkommunikation eingesetzt. LoRa ist die erste kostengünstige CSS Implementierung für den kommerziellen Gebrauch [37].

Unter Verwendung von LoRa kann ein einzelnes Gateway, je nach Umfeld und Hindernissen des jeweiligen Standortes, ganze Städte und hunderte von Quadratkilometer abdecken. Dadurch können ganze Länder mit einer minimalen Infrastruktur leicht erfasst werden.

2.2 Chirp

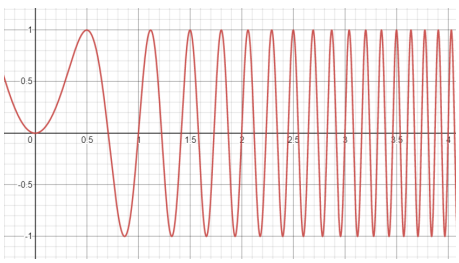

Abbildung 2.1: Chirp

Ein Chirp ist ein sinusförmiges Signal, dessen Frequenz mit der Zeit zunimmt oder abnimmt (Bild 2.1).

Einfache Chirps sind eine Rampe von f_{min} auf f_{max} (Up-Chirp) oder von f_{max} auf f_{min} (Down-Chirp). Datentragende bzw. modulierte Chirps sind zyklisch verschoben (Bild 2.2) [17].

Neben der hohen Übertragungsreichweite ist ein weiterer Vorteil der Verwendung von Chirps, dass die Signale schwer abzufangen (Eavesdropping)

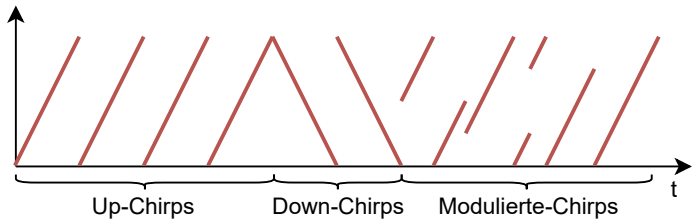

Abbildung 2.2: up-, down- und modulierte-Chirps

oder zu stören (Jamming) sind, wenn die Codesequenz nicht bekannt ist [37]. Unter der Codesequenz versteht man die Bitfolge, die einen modulierten Chirp darstellt.

2.2.1 Spreizfaktor

In Abhängigkeit zum Spreizfaktor (SF) wird die Information möglicherweise nicht vollständig in einem Schritt moduliert, sondern vereinzelt. Der zu modulierende Teil der Information wird im Folgenden als Symbol bezeichnet [15].

Der Spreizfaktor definiert, wie viel Bits ein Symbol repräsentieren. In Europa werden sechs unterschiedliche Spreizfaktoren verwendet, beginnend mit SF7 bis auf SF12. SF7 ist in der Übertragung am schnellsten, hat aber eine kürzere Reichweite. Mit wachsendem Spreizfaktor wird die Bitrate niedriger, dafür ist die Reichweite größer. Die Bandbreite (BW) ist bei CSS konstant (Abschnitt 2.3) und beträgt in Deutschland meistens 125 kHz (Abschnitt 2.4). Die Bitrate R_b lässt sich nach der folgenden Formel berechnen:

$$R_b = SF \cdot \frac{BW}{2^{SF}} \tag{2.1}$$

Somit definiert der Spreizfaktor zwei Werte:

- Die Anzahl von enthaltenen Chips in jedem Symbol = 2^{SF}
- Die Anzahl von Bits, die durch dieses Symbol kodiert ist = SF.

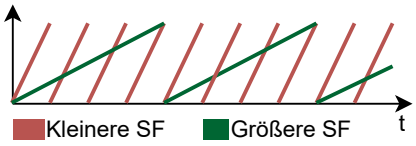

Abbildung 2.3: orthogonale Chirps

Unterschiedliche Spreizfaktoren ermöglichen, mehrere Spreizsignale gleichzeitig und eine Übertragung auf demselben Kanal. Modulierte Signale mit unterschiedlichen Spreizfaktoren erscheinen dem Zielempfänger als Rauschen und können so behandelt werden (Bild 2.3).

2.3 Spreizspektrum

Mit Spreizspektrum ist gemeint, dass die gesamte zugewiesene Bandbreite für die Übertragung eines Signals genutzt wird. In diesem Abschnitt werden die Vorteile vom Spreizspektrum dargestellt und die Theorie dahinter erläutert.

Zunächst ist es wichtig zu verstehen, welchen Einfluss die Bandbreite bei der Übermittlung von Informationen über die Luft hat. In der Informationstheorie gibt das Shannon-Hartley-Theorem die maximalen Datenraten an, mit der Informationen über einen Kommunikationskanal mit einer bestimmten Bandbreite bei Vorhandensein von Rauschen übertragen werden können [37].

$$C = BW \cdot log_2(1 + S/N) \tag{2.2}$$

$C \ldots$ Kanalkapazität $BW \ldots$ Kanalbandbreite $S/N \ldots$ Signal-Rausch-Verhältnis

Hieraus kann abgeleitet werden, dass zur fehlerfreien Übertragung von Informationen in einem Kanal mit festem Rausch-Signal-Verhältnis nur die Bandbreite des übertragenen Signals erhöht werden muss.

Somit ist vorausgesetzt, dass die Erhöhung der Bandbreite zu einer besseren Signalqualität führt. Die Bandbreite ist jedoch in den ISM-Bändern auf 125 kHz begrenzt (Abschnitt 2.4). Um die Kommunikation zu optimieren, wird immer über die gesamte Bandbreite übertragen, um die beste mögliche Signalqualität zu erreichen.

Um das Datensignal zu spreizen, wird es mit einer Codesequenz multipliziert, auch als Chipsequenz bezeichnet. Die Chipsequenz hat eine deutlich höhere Rate als das Datensignal und spreizt so das ursprüngliche Signal über die gesamte zugewiesene Bandbreite. Im Empfänger wird das gespreizte Signal mit einer lokalen Kopie der Chipsequenz multipliziert, um das ursprüngliche Signal wieder herzustellen (Bild 2.4)[1].

Der Vorteil von Spreizspektrum gegenüber Signalen mit fester Frequenz ist, dass die Signale sehr widerstandsfähig gegen Kanalrauschen sind. Dazu sind die Signale schwer abzufangen (Eavesdropping) oder zu stören (Jamming), wenn die Codesequenz nicht bekannt ist.

2.4 Regionale Parameter (EU)

Ein LoRa Radio wird als SRD (Short Range Devices, dt.: Funkanwendungen mit geringer Reichweite) eingestuft und kann in den ISM-Bändern senden. Als ISM werden Fre-

[1] Das Bild wurde aus dem *„AN1200.22 LoRa Modulation Basics“* entnommen [37].

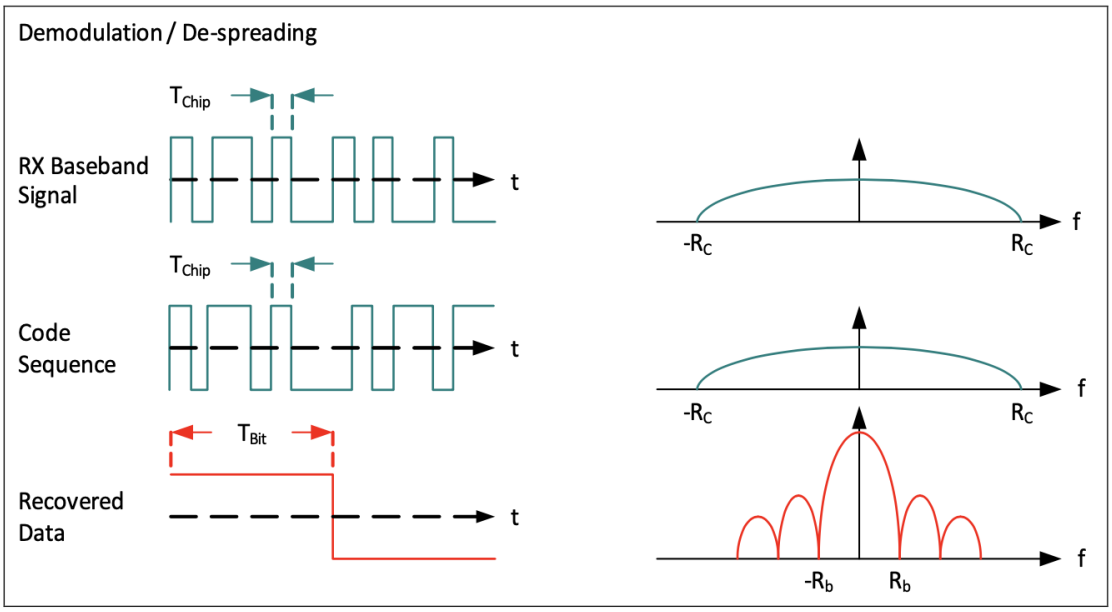

Abbildung 2.4: Demodulation / De-spreading Process

quenzbereiche bezeichnet, die durch Hochfrequenzgeräte in Industrie, Wissenschaft, Medizin, in häuslichen und ähnlichen Bereichen lizenzfrei und meist genehmigungsfrei genutzt werden können.

In Deutschland wurden die Beschränkungen für solche Geräte von der Bundesnetzagentur vorgegeben und sind im Dokument [3] erläutert. Im Allgemeinen hat man je nach Frequenzband eine maximal zugelassene ERP (Effektive Strahlungsleistung) und Arbeitszyklus definiert. ERP ist eine Rechengröße, welche im Bereich der Antennentechnik die in eine Sendeantenne eingespeiste Leistung mit dem Antennengewinn multipliziert ausdrückt [61]. Der Arbeitszyklus ist im Dokument wie folgt definiert:

Der „Arbeitszyklus ist das in Prozent ausgedrückte Verhältnis von $\Sigma(Ton)/(Tobs)$, wobei Ton die ‚Ein-Zeit' eines einzelnen Sendegeräts und $Tobs$ der Beobachtungszeitraum ist. Der Ton wird in einem Beobachtungsfrequenzband (Fobs) gemessen. Sofern in dieser Allgemeinzuteilung nicht anders bestimmt, ist $Tobs$ ein fortlaufender Zeitraum von einer Stunde und Fobs das zutreffende Frequenzband in dieser Allgemeinzuteilung."

Die LoRa Alliance hat diese Begrenzungen je nach Land in Betracht genommen, und neun Hauptfrequenzpläne für das LoRaWAN Protokoll definiert. Das Dokument *„Regional Parameter"* beschreibt, wie die LoRa Alliance diese Frequenzbänder benutzt und fasst die jeweiligen regionalen Beschränkungen zusammen. Interessant für Deutschland sind die EU433 und EU868 Frequenzpläne. Diese Frequenz-

Subband (MHz)	Arbeitszyklus
863.0 - 868.6	1%
868.7 - 869.2	0.1%
869.4 - 869.65	10%
869.7 - 870.0	1%

Tabelle 2.1: Arbeitszyklus für die EU868 Subbänder

bänder sind hier als freie Lizenzen zugewiesen [60].
In der Tabelle 2.4 ist der Arbeitszyklus für die EU868
Subbänder dargestellt. In dieses Frequenzband ist die zugelassene Bandbreite 125 kHz und die maximale ERP +14 dBm (25 mW).

Spreizfaktor	Bytes
SF7 - SF8	222
SF9	115
SF10 - SF12	51

Tabelle 2.2: Bytes Begrenzung je nach SF

Das EU433 Band hat keine Beschränkung im Arbeitszyklus. Jedoch wird es auch für den Betrieb von verschiedenen anderen Geräten und Diensten genutzt, wie z. B. medizinischen Geräten, schnurlosen Thermometern, schnurlosen Kopfhörern und mobilen Geschwindigkeitsmessgeräten, usw. Durch diese gemeinschaftliche Nutzung kann sich die wirksame Reichweite der Geräte nochmals verringern, da hier vor störenden Einflüsse kein Schutz garantiert wird.

Je nach Spreizfaktor hat die LoRa Alliance auch eine Grenze in der Anzahl von gesendeten Datenbytes definiert, damit das Band nicht für längere Zeiten belegt wird (Tabelle 2.2) [14].

2.5 Kodierrate

Zudem fügt die LoRa-Modulation bei jeder Datenübertragung eine Vorwärtsfehlerkorrektur hinzu. Diese Implementierung erfolgt durch die Kodierung von 4-Bit Daten mit 5- bis 8-Bit Redundanz. Durch die Verwendung dieser Redundanz kann das LoRa Signal kürzere Störungen anpassen. Der Wert der Kodierrate (CR) muss entsprechend den Bedingungen des für die Datenübertragung verwendeten Kanals angepasst werden. Sollten zu viele Störungen im Kanal auftreten, ist es empfehlenswert, den CR-Wert zu erhöhen. Durch die Erhöhung der Kodierrate verlängert sich jedoch auch die Dauer der Übertragung.

$$R_b = SF \cdot \frac{\left[\frac{4}{4+CR}\right]}{\left[\frac{2^{SF}}{BW}\right]} \tag{2.3}$$

2.6 Physische LoRa Paket Format

Ein LoRa Paket hat eine Präambel von 8 unmodulierten Up-Chirps. Die Präambel wird bei dem Empfänger zur Erkennung von LoRaWAN Paketen, Einstellung des richtigen Spreizfaktor und Synchronisierung verwendet (Bild 2.5).

Daraufhin kommt das physischer Header (PHDR). Das PHDR enthält die Länge der Nachricht in Bytes, die Kodierrate und das Vorhandensein einer optionalen zyklischen

Redundanzprüfung (CRC). Folgend kommen die Nutzdaten (Payload) und die CRC-Prüfsumme. Das entspricht dem *Explicit Header Mode*.

Falls die Daten eine feste Länge haben, kann man den *Implicit Header Mode* benutzen. Der *Implicit Header Mode* enthält kein PHDR und kann demzufolge eine höhere Anzahl an Nutzdaten übertragen. Die Nachrichtenlänge muss vom Empfänger bekannt sein. Er wird bei Endgeräten der Klasse B verwendet. Näheres zu Geräteklassen wird im Abschnitt 3.6 erläutert.

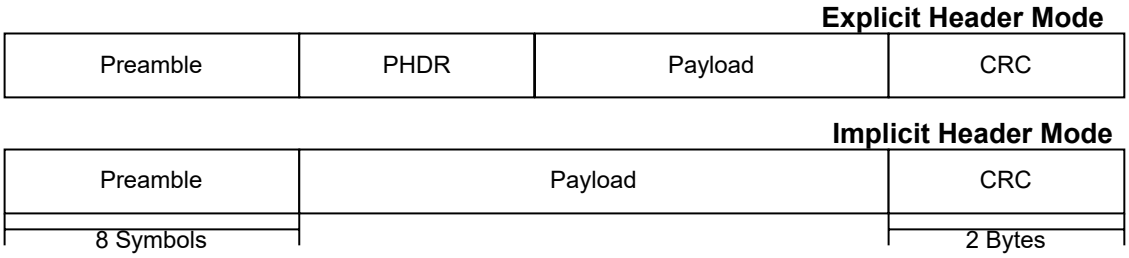

Abbildung 2.5: LoRa physischer Header

2.7 Übertragungsentfernung

LoRa kann große Entfernungen erreichen, der aktuelle Rekord liegt bei 832 km [49]. Es gibt einige Satelliten mit installierten LoRa-Funkgeräten, die von der Erdoberfläche aus erreicht werden können [53]. Trotzdem ist es nicht realistisch, in einer Stadt eine höhere Reichweite als 2 km zu erreichen, da der Pfadverlust in urbaner Umgebungen sehr hoch ist. Mit dem maximalen Spreizfaktor sollten die Geräte optimalerweise zwischen 500 m und 1,5 km vom Gateway entfernt bleiben, um auf diese Weise erhebliche Nachrichtenverluste zu vermeiden.

3 LoRaWAN MAC Schicht

Beim LoRaWAN handelt es sich um ein Low Power Wide Area Netzwerkprotokoll (LPWAN), das für die drahtlose Anbindung batteriebetriebener Geräte an das Internet entwickelt wurde. Es verwendet LoRa, um größere Reichweiten zu erreichen.

Das Protokoll zielt auf die Anforderungen des Internets der Dinge (IoT) ab, wie z. B. bidirektionale Kommunikation, Ende-zu-Ende-Sicherheit, Mobilität, Lokalisierungsdienste und Stromverbrauchminimierung.

LoRaWAN wurde von der LoRa Alliance entwickelt. Sie versuchen, die Standardisierung des Protokolls für LoRa Geräte zu erreichen.

3.1 LoRaWAN Standard Dokumente

Die LoRaWAN-Standardisierungsdokumente werden von der LoRa Alliance entwickelt und gepflegt. Um die LoRaWAN-Standards vollständig zu definieren, werden folgende Dokumente genutz:

„Link Layer Standard“ definiert die MAC Schicht, einschließlich Rahmenformat, Geräteklassen und Kommando von LoRaWAN Nachrichten. Die neueste Version ist die 1.0.4 und diese wird auch empfohlen [13].

„Backend Interfaces Standard“ beschreibt die Schnittstellen zwischen dem Netzwerkserver (NS), dem Join-Server (JS) und dem Anwendungsserver (AS), wie sie für das Roaming und Aktivierung von Geräten erforderlich ist. Das Dokument stellt die Protokolle bereit, die Server miteinander verbinden, wie z. B. die Steuerung der MAC-Schicht, die Endpunkt-Authentifizierung oder die Anwendungsschicht [12].

Das Dokument *„Regional Parameters“* enthält die Frequenzkanalpläne für verschiedene globale Regionen und folgt den in diesen Regionen geltenden regulatorischen Beschränkungen [14]. Dazu gehören: Kanalfrequenzen, Datenraten, Ausgangsleistung usw.

Das *„Certification Program“* stellt sicher, dass die Endgeräte den nationalen Vorschriften in jeder Region entsprechen und die LoRaWAN-Funktionen erfüllen, die zur Gewährleistung der Interoperabilität und Konformität erforderlich sind[1].

3.2 Begriffe

Zunächst ist es wichtig, die Aufgaben der einzelnen Teile des LoRaWAN-Netzes zu definieren.

- **Endgerät:** Das Endgerät ist ein Sensor oder ein Aktor und ist drahtlos mit einem LoRaWAN Netzwerk über ein Gateway verbunden. Die Applikation-Schicht des Endgerätes ist mit einem bestimmten Applikation-Server in der Cloud verbunden. Alle Nutzdaten der Anwendungsschicht dieses Endgeräts werden an den entsprechenden Applikation Server weitergeleitet
- **Gateway:** Das Gateway leitet alle empfangenen LoRaWAN-Funkpakete an den NS weiter, der mit dem Internet verbunden ist. Das Gateway arbeitet vollständig auf der physikalischen Schicht. Seine Aufgabe besteht lediglich darin, Uplink Funkpakete aus der Luft zu decodieren und sie unverarbeitet an den NS weiterzuleiten, ohne die Nutzdaten zu interpretieren.
- **Netzwerkserver:** Der NS terminiert die LoRaWAN-MAC-Schicht für die mit dem Netzwerk verbundenen Endgeräte. Er ist das Zentrum der Sterntopologie. Jeder NS wird durch eine eindeutige NSID identifiziert und kann mit einer oder mehreren NedIDs konfiguriert werden [12].
- **Join-Server:** Der JS verwaltet die OTAA (Over-the-Air-Activation, Abschnitt 4.1.2).
- **Application-Server:** Ein Anwendungsserver ist ein Server in einem Netzwerk, der spezielle Dienste zur Verfügung stellt, wie z.B. Authentifizierung oder den Zugriff auf Webservices und Datenbanken über definierte Schnittstellen.

3.3 Netzwerk-Architektur

Viele bestehenden Netzwerke nutzen eine Mesh-Netzwerk-Architektur. In einem Mesh-Netzwerk leiten Endgeräte die Information anderer Knoten weiter, um die Reichweite zu erhöhen. Eine LoRaWAN Netzwerk nutzt dagegen ein Sternnetzwerk mit asynchronen Knoten.

Die Knoten können nicht in den Tiefschlafmodus wechseln, solange sie Informationen von anderen Knoten übertragen. Dadurch verliert das Gerät Batterielebensdauer. Außerdem kostet die Weiterleitung Sendezeit, die in den ISM-Bändern schon sehr begrenzt ist, was zur Verringerung von Netzwerkkapazität führen würde.

Ein Stern-Netzwerk ist sinnvoller für die Erhaltung der Batterielebensdauer und um Sendezeit zu sparen, wenn weitreichende Konnektivität aus der physischen Schicht LoRa schon erreicht werden kann. Daher bilden Endknoten und das Gateway ein Netz mit der Sterntopologie, d. h. die Endgeräte kommunizieren nur mit dem Gateway, aber nie miteinander.

In einem synchronen Netzwerk, müssen die Knoten häufig aufwachen, um sich mit dem Netzwerk zu synchronisieren. Nur danach können sie Nachrichten senden. Diese Synchronisation verbraucht Batterielebensdauer und ist der Hauptgrund, weshalb die Knoten in LoRaWAN asynchron sind. Sie kommunizieren sobald Daten bereit sind. Das Gateway muss immer auf Empfangsbereitschaft eingestellt sein und verbraucht dadurch sehr viel Energie. Deswegen muss das Gateway idealerweise an eine zuverlässige Stromversorgung angeschlossen sein.

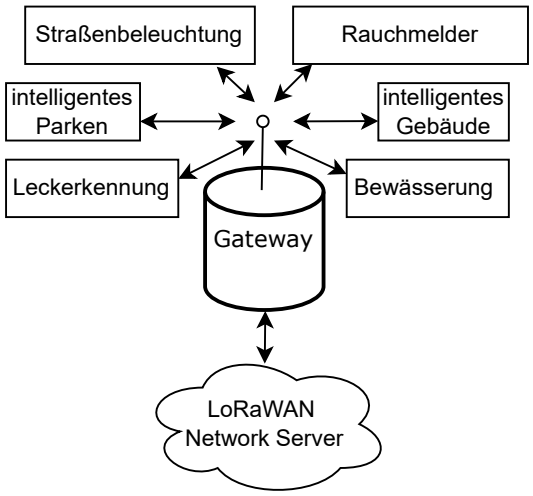

Abbildung 3.1: Stern LoRaWAN Netzwerk

In einem LoRaWAN sind Netzwerkknoten nicht einem bestimmten Gateway zugeordnet. Stattdessen werden Daten, die von einem Knoten übertragen werden, typischerweise von jedem Gateway in der Reichweite empfangen. Jedes Gateway leitet das empfangene Paket auf dem Endknoten in den Netzwerk-Server weiter, ohne es zu verarbeiten.

Die Komplexität des LoRaWAN Protokolls liegt beim NS. Zunächst werden doppelte Pakete, die von einem Endgerät stammen und von mehreren Gateways weitergeleitet wurden, verworfen. Danach nimmt der NS die Sicherheitsüberprüfungen vor, plant eine adaptive Datenrate für das Endgerät (Abschnitt 3.8) und sendet die Antwort über das Gateway an das Endgerät zurück.

Das LoRaWAN Protokoll unterstützt Nachrichten in beide Richtungen. Nachrichten vom Endgerät zum Gateway werden als Uplink bezeichnet. Downlink sind Nachrichten, die vom Gateway an Endknoten geschickt werden.

3.4 Netzwerkkapazität

Um mit einem Langstrecken-Sternnetz hohe Netzwerkkapazität zu erreichen, muss das Gateway Nachrichten von sehr vielen Endgeräte gleichzeitig empfangen können. LoRaWAN verwendet verschiedene Strategien, um dies zu erreichen. Einerseits wird das durch die Verwendung einer adaptiven Datenrate (ADR) erreicht (Abschnitt 3.8). Andererseits verwendet LoRaWAN einen mehrkanaligen Multi-Modem-Transceiver, sodass gleichzeitige Nachrichten auf mehreren Kanälen empfangen werden.

Außerdem verwendet LoRaWAN Spreizspektrum-Modulation. Wie bereits im Abschnitt 2.2.1 erwähnt, sind solche Signale praktisch orthogonal zueinander, wenn unterschied-

liche Spreizfaktoren verwendet werden. Das Gateway nimmt diese Eigenschaft zum Vorteil und kann dadurch mehrere Nachrichten mit unterschiedlichen Datenraten auf dem gleichen Kanal zur gleichen Zeit empfangen.

3.5 Roaming

Das Roaming von IoT-Knoten ist ein Architekturmerkmal, das in letzter Zeit an Bedeutung gewonnen hat. Um ein riesiges Netz in einem landesweiten Einsatz zu betreiben, müsste ein nationaler Versorger Gateways über ein ganzes Land verteilen. Die Gateways würden dann alle mit einem einzigen NS verbunden werden, der das gesamte LoRaWAN Netz verwaltet. Ein Projekt in solchem Ausmaß ist jedoch schwierig. Bei der eigentlichen Implementierung müssen Aspekte wie Fehlertoleranz, Skalierbarkeit und die Fähigkeit, eine große Anzahl von Nachrichten zu verarbeiten, berücksichtigt werden. Diese Herausforderungen sind bei der Einrichtung eines nationalen Netzes besonders wichtig. Infolgedessen wurde die Möglichkeit, dass ein IoT Gerät von einem Netz an ein anderes wechselt, sehr lange einfach nicht in Betracht gezogen.

Mit dem Aufkommen der LPWANs wird das Ziel verfolgt, eine große Anzahl von IoT Geräten drahtlos zu verbinden. Sie können sowohl zu einem privaten Netzwerk in einem Gebäude gehören, bis hin zu einem landesweiten IoT-Netzwerk.

Ein Einsatzbeispiel der Roaming Funktion wäre, dass einen LoRaWAN End-Knoten Teil eines präventiven Wartungsprogramms ist. Damit der Gerätehersteller seine Produkte in vielen Ländern und Märkten auf der ganzen Welt effektiv verkaufen kann, benötigt er einen weltweiten Verbindungsplan von lokalen LoRaWAN Netzbetreiber. Die von Geräteherstellern geforderte Funktion ist technisch als Roaming Plan bekannt und konnte vor der *„Backend Interfaces Standard“* v1.1 nicht bereitgestellt werden [55].

In der *„Backend Interfaces Standard“* v1.1 hat die LoRa Alliance den NS in drei Teile geteilt und einen zusätzlichen JS hinzugefügt, um das Roaming zu ermöglichen. Der JS speichert die Identifikatoren aller Endgeräte (Tabelle 4.1).

- **Home-NS:** Es ist der NS des Netzbetreibers, dem das Endgerät angehört.
- **Serving-NS:** Es ist der NS eines Betreibers, dessen Netz sich um Endknoten kümmert, die sich außerhalb der Reichweite ihres eigenen Home-NS befinden. Er führt alle Funktionen des Home-NS aus, soweit es das MAC-Protokoll betrifft. Er verwaltet das sogenannte aktive Roaming. Wenn der Endknoten unter der Kontrolle des Home-NS steht, übernimmt der Home-NS die Rolle des Serving-NS.
- **Forwarding-NS:** Es handelt sich um den NS eines Betreibers, dessen Netz sich beispielsweise mit dem Home-Server einiger Endgeräte überschneidet. Er ist transparent und leitet den Verkehr der Endgeräte über die an ihn angeschlossenen Gateways weiter. Es verwaltet das sogenannte Passive Roaming [12].

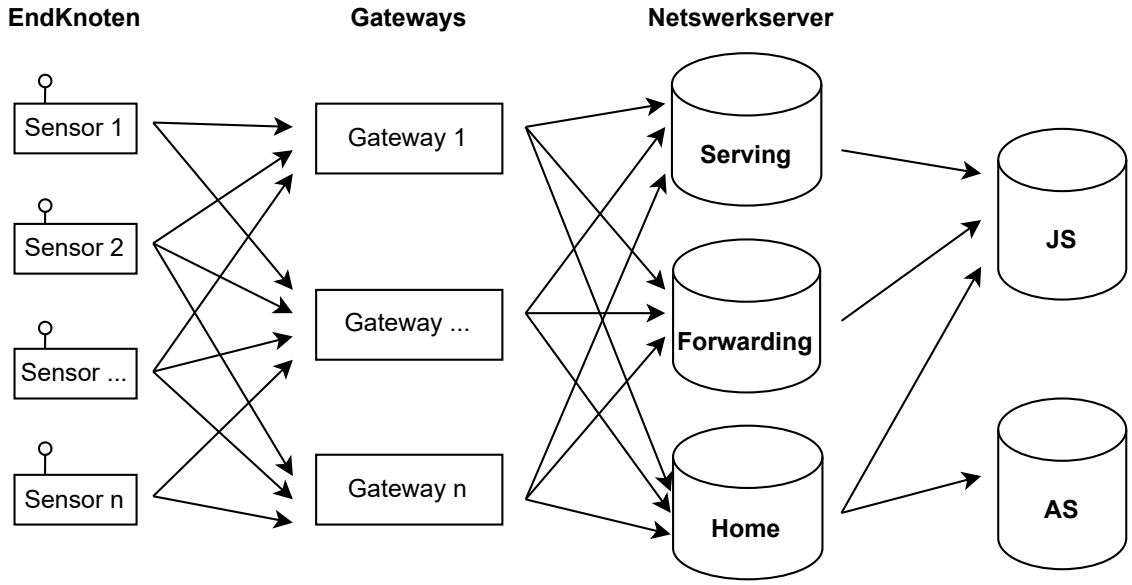

Abbildung 3.2: Aufbau der LoRaWAN Netzarchitektur mit Roaming.

3.6 Endgeräte Klassen

Das Protokoll unterstützt drei Klassen von Endgeräten. Diese Klassen sind je nach Anwendungsfall auszuwählen. Hauptsächlich ist zu beachten, ob das Gerät mit Batterie betrieben wird.

- **Klasse A**
  Das Gerät öffnet nach jeder gesendeten Nachricht ein Empfangsfenster (RX1) nach einer bestimmten Verzögerung, der auch bei dem Gateway bekannt ist. Dieses Empfangsfenster wird zum gleichen Spreizfaktor und Frequenz wie die Uplink-Nachricht eingestellt. Wenn das Gateway zu diesem Empfangsfenster nicht antwortet, wird ein zweites Empfangsfenster (RX2) mit vorbestimmter Frequenz und SF geöffnet. Diese Frequenz ist im Frequenzplan vorgegeben.
  Wird in beiden Fenster keine Nachricht empfangen, geht das Gerät in Stromsparmodus und kann nur dann etwas empfangen, wenn die nächste Nachricht gesendet wird.
  Diese Klasse hat den geringsten Energieverbrauch von allen und wird meistens für batteriebetriebene Geräte verwendet.
- **Klasse B**
  Geräte der Klasse B erweitern die Klasse A, indem sie periodisch ein Empfangsfenster (RX) öffnen. Das Gerät muss dem Gateway diese Periode bekannt geben.
- **Klasse C**
  Bei Geräten der Klasse C ist das Empfangsfenster immer geöffnet, solange das Gerät nicht am Senden ist. Sie sind oft netzbetriebene Geräte (Bild 3.3).

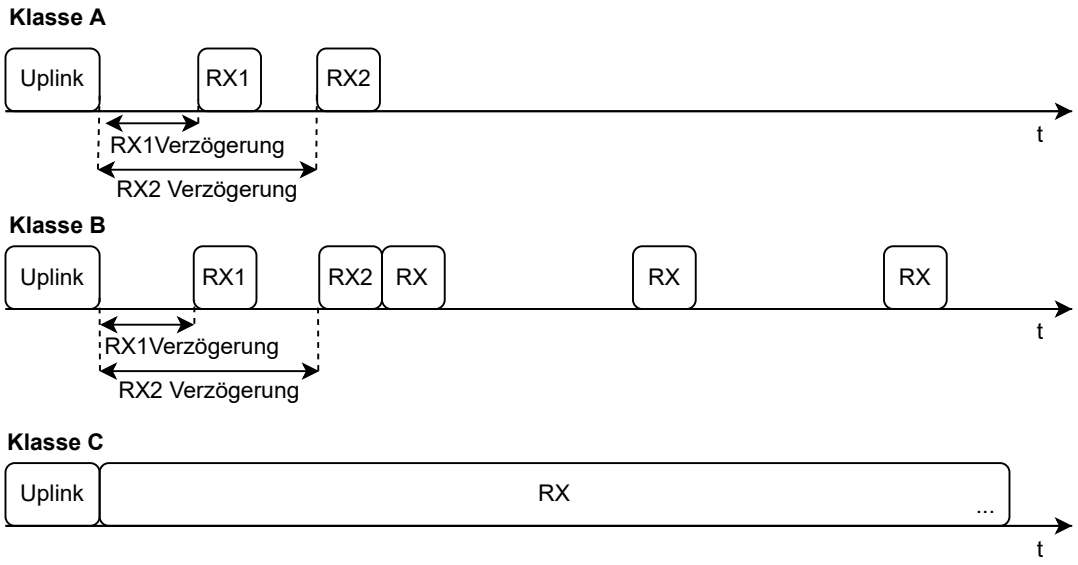

Abbildung 3.3: LoRaWAN Klassen

3.7 Paketaufbau

In LoRaWAN sind zwei Hauptpakettypen definiert: Uplink sind Pakete, die vom Endgerät an das Gateway gesendet werden. Downlink sind Packete, die vom Gateway an die Endgeräte gesendet werden.

Sowohl Uplink als auch Downlink LoRaWAN Pakete beginnen mit einem MAC-Header (MHDR), gefolgt von Nutzdaten und enden mit einem MIC (Message Integrity Check). Der MIC wird über alle Felder des Rahmens berechnet.

Der MHDR gibt an, was für einen Rahmen die Nutzdaten enthalten. Dieses kann entweder ein *Data-, Join-Request- oder Join-Accept*-Rahmen sein. Der *Join-Request*- und *Join-Accept*-Rahmen werden vom OTAA verwendet (Abschnitt 4.1.2).

Die Nutzdaten (Payload) von Datenrahmen enthaltet ein *Frame Header* (FHDR), der *Frame Port* und die Nutzdaten (Payload). Das FHDR enthaltet Steuerbytes (Frame Control), das Gerätadresse (DevAddr), ein Paketzähler (Frame Counter) und zusätliche Rahmeneinstellungen (Frame Opts). Die Struktur von Datenrahmen ist auf dem Bild 3.4 dargestellt.

Die DevAddr sollte für jedes Gerät eindeutig sein (Tabelle 4.1). Der Paketzähler wird mit jeder gesendeten Nachricht um eins erhöht. Weitere Einzelheiten über die Steuerbytes und die Rahmeneinstellungen befinden sich in der LoRaWAN *„Link Layer Spezifikation“* [13].

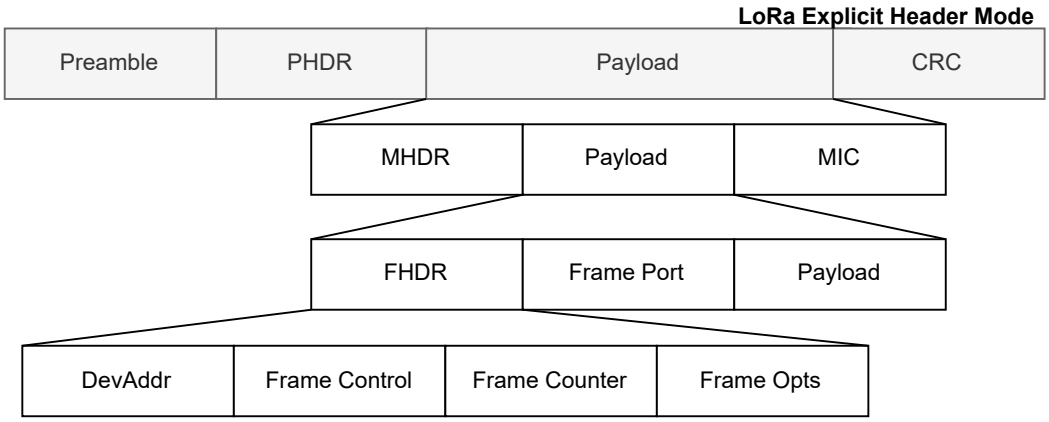

Abbildung 3.4: LoRaWAN Datenrahmen

3.8 Adaptive Datenrate (ADR)

ADR steuert die Übertragungsparameter von Endknoten, nämlich Bandbreite (BW), Spreizfaktor (SF), Sendeleistung (TP), und Kodierrate (CR). Das Ziel dabei ist, den Energieverbrauch zu minimieren und die Netzwerkkapazität zu maximieren, indem die Datenrate in Abhängigkeit von der Leistungsübertragungsbilanz für jeden Endknoten angepasst wird.

Die ADR-Funktionalität muss am Endgerät aktiviert werden, damit sie genutzt werden kann. Das Endgerät tritt dem Netzwerk immer zu einem vorbestimmten SF bei. Nach ein paar Paketen, durch Analyse von RSSI, Kanalbelegung und anderen Parametern, berechnet das Netzwerk den idealen ADR-Wert für das Endgerät und passt es bei Bedarf an.

Die größte Herausforderung beim ADR besteht darin, dass die LoRaWAN Spezifikation nicht festlegt, wie der NS die Endknoten bezüglich der Ratenanpassung anweisen muss. Viele unterschiedliche ADR Methoden wurden vorgeschlagen und sind im Einsatz und viele Anbieter halten ihre eigene Implementierung geheim. Das führt zu unterschiedlichen Servicequalitäten, was eine Herausforderung für die Zuverlässigkeit eines LoRaWAN Netzwerks darstellt.

In [27] wird eine umfassende Analyse der bereits auf dem Markt befindlichen Lösungen durchgeführt, die zur Optimierung von ADR Algorithmen entwickelt wurden. Die Algorithmen befassen sich mit gezielten Aspekten von LoRaWAN, sowie Skalierbarkeit, Datendurchsatz und Energieeffizienz.

4 Sicherheit

Obwohl LoRaWAN ein sehr weitverbreitetes Protokoll ist, zeigen viele Literaturwerke Schwachstellen und Risiken in Bezug auf die Sicherheit von LoRaWAN.

Die Verbreitung von IoT Geräten und damit des LoRaWAN Standards hängt von der öffentlichen Akzeptanz dieser Geräte in der Öffentlichkeit als Teil eines vertrauenswürdigen Systems ab. Daher ist die Verbesserung des Sicherheitsniveaus von LoRaWAN Geräten notwendig, um öffentliche Unterstützung und Akzeptanz zu erlangen.

Die LoRa Alliance hat viele solcher Probleme in der LoRaWAN v1.1 gelöst. Das ist gut, denn es bedeutet, sie hören der Gemeinschaft zu und arbeiten noch aktiv am Protokoll. Jedoch benutzen immer noch viele Netzwerke das alte LoRaWAN v1.0.

Dieses Kapitel konzentriert sich auf die Version 1.1 des Protokolls. Dabei wird eine umfassende Sicherheitsanalyse des Protokolls geboten. Zunächst im Abschnitt 4.1 wird das Join-Verfahren und dessen Sicherheitskontext erklärt, sowie die Rolle aller Netzwerkschlüssel. Danach wird im Abschnitt 4.2 die Hauptschwachstellen eines LoRaWAN-Netzwerks gelistet und im Abschnitt 4.5 werden verschiedene Abhilfemaßnahmen zur Beseitigung der beschriebenen Sicherheitsrisiken erörtert.

4.1 Join-Verfahren und Sicherheitskontext

4.1.1 Schlüsselübersicht

LoRaWAN hat verschiedene Schlüssel, die an verschiedenen Stellen verwendet werden. Um das Protokoll am sichersten zu implementieren, ist es wichtig, die Rolle von jedem Schlüssel zu verstehen und wie sie verwendet werden.

Jedes Endgerät verfügt beim LoRaWAN Protokoll über zwei eigene 128-Bit-AES-Keys, der Applikationsschlüssel (AppKey) und der Netzwerkschlüssel (NwkKey). Beide Schlüssel sollen geheim gehalten werden und sollen niemals drahtlos übertragen werden. Außerdem hat jedes Endgerät auch eine DevAddr und eine eindeutige Kennung (JoinEUI).

Bei erfolgreicher Authentifizierung werden dann für jede Sitzung einige Schlüssel abgeleitet: Der Applikationssitzungsschlüssel (AppSKey) sichert die Ende-zu-Ende Verschlüsselung der Nutzdaten von Datenpaketen. Die drei Netzwerksitzungsschlüssel (NwkSKeys) prüfen die Integrität aller *Up*- und *Downlink* Pakete, und sind verantwortlich für die Integrität und Verschlüsselung von MAC-Kommandos. Die Verwendung beider Schlüs-

sel stellt auch die Integrität der Datenübertragung sicher. Die Sitzungsschlüssel sind nur für eine einzige Kommunikationssitzung zwischen einem Endgerät und Server gültig.

Es ist wünschenswert, dass AppSKey und die NwkSKeys vor der Überlauf der Paketzähler (Abschnitt 3.7) durch einen Rejoin-Request aktualisiert werden, um einen möglichen Replay-Angriff zu vermeiden [19]. Wie ein Replay-Angriff in diesen Zusammenhang funktioniert, wird im Abschnitt 4.3 erläutert. Der MIC verhindert die Manipulation der Daten im Verlauf der Datenübertragung und gewährleistet zudem, dass nur ein authentifiziertes Endgerät einen gültigen Frame erzeugen kann.

Alle erforderlichen Schlüssel und Identifikatoren, egal ob sie generiert oder im Voraus gespeichert wurden, sind in der Tabelle 4.1 aufgeführt.

4.1.2 Join-Verfahren

Um an einem LoRaWAN-Netzwerk teilzunehmen, muss jedes Endgerät personalisiert und aktiviert werden. Die Aktivierung eines Endgeräts kann auf zwei Arten erfolgen:

ABP

Dies ist der unsicherste Weg, ein Endgerät zu aktivieren und sollte nur verwendet werden, wenn das Gerät nur Uplinks sendet und infolgedessen nicht das Join-Verfahren vollständig durchführen kann.

Alle Sitzungsschlüssel und die DevAddr müssen bereits vom Endgerät, NS und AS bekannt sein, bevor es das erste Mal sendet. Das Endgerät kann Nachrichten senden, sobald es eingeschalten wird.

OTAA

Alle Endgeräteschlüssel und Identifikatoren müssen im Voraus zum Join-Server gegeben werden. Nur dann kann das Gerät durch das Join-Verfahren dem Netzwerk beitreten. Der Join-Server und das Endgerät sind die einzigen, die den AppKey und NwkKey kennen dürfen.

(1) Das Endgerät beginnt immer das Join-Verfahren mit dem Senden einer Join-Request (Bild 4.1). Die Join-Request wird nicht verschlüsselt und enthält das JoinEUI, DevEUI und DevNonce.

(2) Der NS schaut nach den JoinEUI und leitet das Paket an den entsprechenden Join-Server weiter.

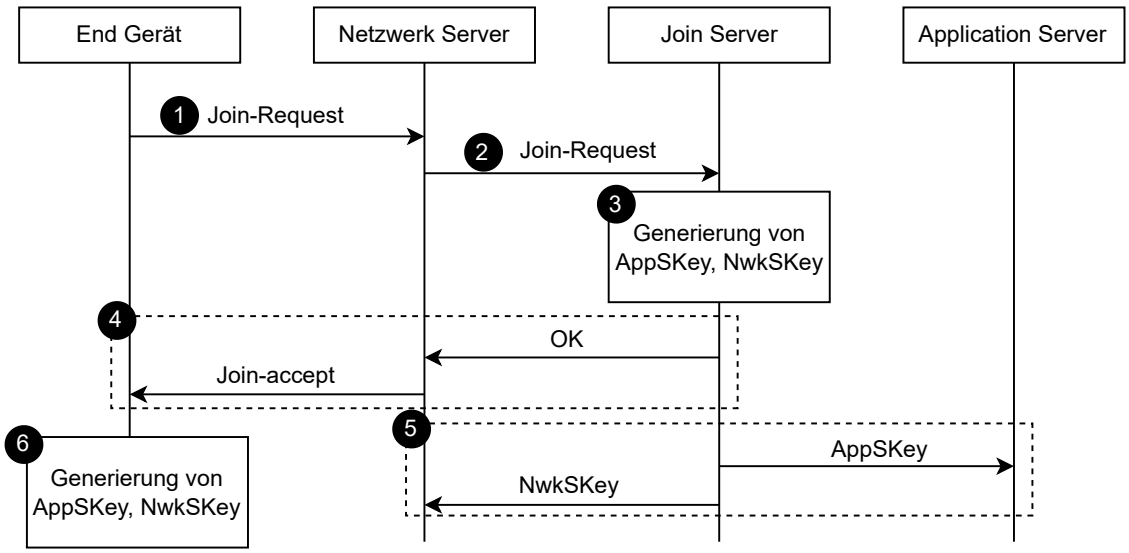

Abbildung 4.1: OTAA Paketverlauf in LoRaWAN v1.1

(3) Falls der Join-Server das DevEUI kennt, generiert er mithilfe der AppKey und NwkKey den Sitzungsschlüsseln, der JoinNonce und das DevAddr des Endgerätes.

(4) Danach schickt der Join-Server eine Join-Accept Nachricht, mit der Teilnahmebestätigung und einem JoinNonce. Das Netzwerk fügt noch seine eigene NetID hinzu und leitet das Paket an das Endgerät weiter.

(5) Die NwkSKeys werden vom Join-Server an die Netzwerkserver gegeben, und die AppSKey wird an die App Server gegeben.[2] Die Sitzungsschlüssel sind für jedes Gerät in jedem Netz und für jeden Anwendungsserver eindeutig. Selbst wenn das Gerät gestohlen werden würde, wäre nur das Gerät selbst betroffen, ohne das gesamte Netz zu gefährden.

(6) Das Endgerät kann die Sitzungsschlüssel unter Verwendung der JoinNonce aus der Join-Accept und des AppKey und NwkKey, die schon bekannt waren, erstellen. Beachten Sie, dass der NwkKey und der AppKey niemals drahtlos gesendet werden sollten.

DevNonce und JoinNonce sind Zähler, die bei jeder Join-Request inkrementiert werden. Ein Nonce Wert sollte niemals für einen bestimmten JoinEUI Wert wiederverwendet werden.

Roaming

[2] Alternativ kann der AppSKey zurück an das Endgerät über NS gesendet werden. Damit der NS nicht den AppSKey lesen kann, muss er mit einem ASJSKey verschlüsselt werden. Der ASJSKey ist nur von den AS und JS bekannt. Der AS muss den AppKey mit dem ASJSKey entschlüsseln und dann die Nutzdaten mit dem AppSKey entschlüsseln. Obwohl dieses Verfahren in der LoRaWAN Backend Interfaces [12] definiert ist, wird es nicht so oft verwendet.

Die Aktivierung für ein Gerät im Roaming ist etwas komplizierter und verwendet alle drei im Abschnitt 3.5 genannte Netzwerkserver. Da die Aktivierung mehrere Antworten vom Netzwerk benötigt, ist das Roaming mit ABP ausgeschlossen.

4.2 Schwachstellen

Hier werden einige der Schwachstellen bezogen auf das LoRaWAN Protokoll aufgelistet.

- **Gateway:**
  - Die Gateways stellen die schwächste Stelle von LoRaWAN dar. In den meisten Implementierungsszenarien werden sie in geringer Zahl eingesetzt. Jede Art von Angriff einschließlich Diebstahl würde die Kommunikation zwischen den Endknoten und dem Rest des Netzes zerstören.
- **Server:**
  - Die drei Netzwerkserver, zusammen mit dem Application-Server und Join-Server, erschweren die Netzverwaltung.
  - ABP unterstützt keine Wiederzuweisung von Keys.
  - Es ist unklar, wie lange eine Sitzung dauert.
- **Schlüsselverteilung:**
  - ABP: Wenn ein Problem bei der Speicherung des Paketzählers oder Rauschen bei der Wiederaufnahme der letzten Zahl nach einem Reset-Ereignis auftritt, ist das Gerät nicht mehr mit dem Netzwerk synchronisiert und somit unbrauchbar.
  - OTAA: Für das Aktivierungsverfahren werden die Root-Schlüssel für die Erzeugung der Session Keys benötigt (NwkKey und AppKey). Diese Schlüssel werden vom Hersteller gegeben und müssen sicher aufbewahrt werden. Stattdessen könnte PKI (Public Key Infrastruktur) verwendet werden, dies würde mehr Flexibilität und Sicherheit für das LoRaWAN bringen.
- **Implementierung:**
  - Es gibt keine explizite und dedizierte LoRaWAN-Signalisierung für die Außerbetriebnahme von Endgeräten nach Ablauf ihrer Lizenz oder falls sie als Gefahr gelten und vom Netz getrennt werden sollten.
  - Alle Netzwerk-Server enthalten eine Liste der DevEUI, die während der gesamten Lebensdauer des Netzes verwendet werden. Das Verfahren zum Widerruf oder zur Erneuerung von DevEUI's von End-Geräte ist nicht im Spezifikation-Dokument.

  - Da das JoinEUI auf den Endgeräten gespeichert sein muss, bevor das OTAA gestartet wird, würde jede mögliche Änderung des Join-Servers, die eine JoinEUI Änderung erfordern könnte, sich auf alle Endgeräte betreffen. Alle diese Geräte wären dann unbrauchbar. Daher sollte eine JoinEUI Erneuerungsverfahren definiert werden.
  - In den Spezifikationen wird erwähnt, dass neben AppKey und NwkKey, auch alle Sitzungsschlüssel sicher gespeichert werden müssen, sodass die Wiederverwendung dieser Schlüssel durch Angreifer verhindert wird. Es ist möglich, den nichtflüchtigen Speicher der Endgeräte zu sichern, aber die Sicherheit des flüchtigen Speichers ist nicht hardwareunterstützt und sehr anspruchsvoll aus Softwaresicht.

- **Roaming:**
  - Das Roaming kann zu einem Fall-Back führen, wenn der Netzwerkserver, der das Roaming Endgerät bedient, die ältere Version von LoRaWAN verwendet (v1.1).
  - Das Handover Roaming bietet mehr Möglichkeiten für MITM-Angriffe. [4], [16]

4.3 Häufige Bedrohungen

Hier werden die am häufigsten auftretenden Bedrohungen gegen LoRaWAN-Netzwerke beschrieben.

- **Netzwerküberflutungsangriff:**
  Die Endgeräte können gestohlen werden und für Angriffe auf das restliche Netz verwendet werden. So ist es möglich, dass das gestohlene Endgerät das Netz mit Paketen überflutet. Eine mögliche Lösung wäre, die Sendezeit von Endgeräten einzuschränken.
- **Analyse des Netzwerkverkehrs:**
  Bei dieser Art von Angriff, kann ein Angreifer ein Gateway einrichten, um Pakete zu empfangen und daraus Kenntnisse über die übertragenen Daten abzuleiten. Es ist zu beachten, dass ein Angreifer ohne die notwendigen Schlüssel nicht in der Lage ist, den Inhalt der empfangenen Pakete zu entschlüsseln.
- **Self-Replay Angriffe**
  Ein Angreifer kann die Signale, die für die OTAA-Sitzung verwendet werden, blockieren. Bei dieser Art von Angriff wird die Übertragung einer Join-Request von einem Endgerät erfolgreich beobachtet. Die entsprechende Join-Accept vom NS an das Endgerät wird durch selektives Stören blockiert. Nach einer Zeitüberschreitung für den Empfang der Join-Accept Nachricht versucht das Endgerät erneut dem Netz beizutreten und sendet erneut die gleiche Join-Request mit dem glei-

chen DevNonce, wie in der Spezifikation gefordert ist. Da das Join-Verfahren nicht erfüllt ist und alles gemäß der Spezifikation in Ordnung ist, akzeptiert der NS die Join-Request und sendet wieder einen Join-Accept zurück. Dieser Angriff wird fortgesetzt, bis der tägliche Arbeitszyklus des Endgeräts erreicht ist. Um diese Art von Angriff zu vermeiden, sollte der Endknoten mehr als ein Gateway in Reichweite haben. Bei kleineren LoRaWAN-Netzwerken ist dies jedoch eher die Ausnahme.

4.4 Fazit

Durch Analyse von Schwachstellen des LoRaWAN Protokolls ist klar, dass die Daten ausgezeichnet geschützt sind. Ein Angriff zur Entschlüsselung des Inhalts von Paketen ist sehr unwahrscheinlich. Jedoch ist es sehr einfach, ein LoRaWAN-Netzwerk zu stören oder sogar zu zerstören. Um Diebstahl zu vermeiden sollen alle Geräte gesichert sein.

4.5 Abhilfemaßnahmen und Schutzmechanismen

4.5.1 Schlechtes Vorgehen

Die Sicherheit hängt natürlich auch noch von anderen Faktoren als der Verschlüsselung ab. Im Kernbereich des Netzes sollte die Kommunikation zwischen Join-Server, Netzwerk-Server und Applikationsserver zusätzlich durch HTTPS oder VPN abgesichert werden. Hier gelten dann die normalen Regeln der Netzwerksicherheit.

Weiter unten wird auf einige Vorfahren hingewiesen, die bei LoRaWAN Implementierungen häufig anzutreffen sind und das Netzwerk in Gefahr bringen können.

- Verwendung von ABP anstatt von OTAA. Der einzige Fall, dass ABP über OTAA ausgewählt wird, ist, wenn das Gerät nur Uplinks sendet und infolgedessen nicht das Join-Verfahren vollständig durchführen kann.
- Private Schlüssel über E-Mail Anhänge, QR-Codes, oder Rechnungen versenden.
- Wiederverwendung von Nonces.
- Das Extrahieren von privaten Schlüsseln von Endgeräten auf irgendeine Weise erleichtern (z. B. über Programmcode, Sticker auf dem Mikrocontroller, usw.)

Um die Hardware zu Schützen wird folgende empfohlen:

- Hardware Schutz gegen Manipulationen

- Hardware-Sicherheitsmodul an der Serverseite (HSM)
- sicheres Element auf dem Endgerät (SE)
- Diebstahl erschweren

Schlüssel	Beschreibung	Erzeugung oder im Voraus gespeichert
	Vor der Aktivierung benötigte Schlüssel	
NwKKey	Wird verwendet, um dem MIC von Join-Request Pakete zu berechnen, zur Verschlüsselung von Join-Accept Paketen, und zur Ableitung aller NwK Sitzungsschlüssel.	im Voraus gespeichert
AppKey	Wird verwendet, um AppSKey zu erzeugen	im Voraus gespeichert
JSIntKey	Wird für MIC von Rejoin-Request und Join-Accept Paketen	Generiert aus NwkKey und DevEUI
JSEncKey	Wird verwendet, um Join-Accept zu verschlüsseln. Ausgelöst durch rejoin-request	Generiert aus NwkKey und DevEUI
	Nach der Aktivierung benötigte Schlüssel	
FNwkSIntKey	Wird für die Berechnung des MIC eines Teils aller Uplink-Datenpakete	Erzeugt aus NwkKey und Join-Accept Nachricht
SNwkSIntKey	Wird zur Überprüfung der MIC aller Downlink-Datenpaketen und zur Berechnung Teil der MIC von Uplink-Paketen	Erzeugt aus NwkKey und Join-Accept Nachricht
NwkSEncKey	Wird zur Verschlüsselung aller Downlink- und Uplink-MAC-Pakete	Erzeugt aus NwkKey und Join-Accept Nachricht
AppSKey	Wird zur Verschlüsselung/Entschlüsselung der Nutzdaten von Datenpaketen verwendet	Erzeugt aus AppKey und Join-Accept Nachricht
	IDs	
JoinEUI	64-Bit global eindeutige Anwendungs-ID die den Join-Server identifiziert	im Voraus gespeichert
DevEUI	64-Bit global eindeutige Geräte-ID durch den Netzwerkserver	im Voraus gespeichert
DevAddr	32-Bit eindeutige Geräteadresse im aktuellen Netzwerk	Received by join-accept message

Tabelle 4.1: LoRaWAN v1.1 Sicherheitsschlüssel

5 Skalierbarkeit

5.1 Problemstellung

Die größte Herausforderung von LoRaWAN im Sinne der Skalierbarkeit ergibt sich aus der hohen Anzahl von ACK Anfragen der Endgeräte und der Arbeitszyklusbeschränkung, die das Gateway einhalten muss.

Ein LoRaWAN Endgerät überprüft nicht, ob der Kanal belegt ist, bevor eine Nachricht gesendet wird. Falls das Gateway eine Nachricht empfängt, sendet das Gateway einen ACK an das Endgerät zurück. Falls das Gateway keine Nachricht empfängt und demzufolge das Endgerät keine ACK bekommt, versucht das Endgerät die Nachricht später wieder zusenden.

Wie lange ein Endgerät wartet, bis es sendet und die Wahrscheinlichkeit, dass eine Kollision auftritt, hängen zusammen und beeinflussen, wie effizient das Gateway genutzt werden kann. Dies bedeutet, dass die Qualität des ACKs einen erheblichen Einfluss auf die Effizienz des Protokolls und die endgültige Kanalkapazität hat.

In anderen Worten: Im Fall, dass eine Nachricht vom Gateway empfangen wird, aber das ACK aufgrund frühere Kollisionen nicht am Endgerät ankommt, wird das Endgerät den Spreizfaktor erhöhen und unnötig Nachrichten erneut senden. Das führt zur Überflutung des Netzwerks. Daher stecken Geräte, die weit entfernt vom Gateway sind, in einem Zyklus fest: Während sie versuchen, einen zuverlässigeren Kanal zu bekommen, erhöhen sie nur die Wahrscheinlichkeit zusätzlicher Kollisionen und stoßen schnell an ihre Arbeitszyklusbeschränkung.

Die Skalierbarkeitanalyse kann in zwei Teile getrennt werden. Der erste Teil analysiert die physische Schicht und inwieweit Interferenzen die Skalierbarkeit von einem LoRaWAN Netzwerk beeinflussen. Der andere Teil analysiert, welche Änderungen in der MAC Schicht gemacht werden können.

5.1.1 Mögliche Lösungen

Der effektivste Weg in der physischen Schicht, ein LoRaWAN-Netz skalierbar zu machen, besteht darin, mehr Gateways einzurichten (Abschnitt 5.2.1).

Ein weiterer Vorschlag ist die Verwendung von Richtantennen bei den Endgeräten. Die Richtantennen erhöhen die Signalstärke an den Empfängern und vermindern Kollisionen, ohne die Energiekosten für die Übertragung zu erhöhen. Richtantennen funktio-

nieren aber nicht für bewegende Endgeräte. Studien haben aber gezeigt, dass obwohl diese Antennen die Interferenz reduzieren, der Unterschied sehr gering ist.

Es ist trotzdem gut, diesen Ansatz zu beachten, wenn ein Netzwerk von Grund auf aufgebaut wird. Falls das Netzwerk schon aufgebaut ist, wird es viel aufwendiger alle Endgeräte mit einer Richtantenne auszustatten. In diesem Fall ist es besser neue Gateways zum Netzwerk hinzufügen und sie optimal zu positionieren. Die Verwendung mehrerer Gateways übertrifft die Verwendung von Richtantennen deutlich [56].

Ein paar Vorschläge wurden auch in der MAC-Schicht gemacht. Forscher sind sehr aktiv bei der Suche nach dem optimalen ADR-Algorithmus. Das Ziel ist Kollisionen zu minimieren und eine automatische Änderung der Datenrate zu ermöglichen, wenn eine Kollision erkannt wird. Damit würde der Energieverbrauch von LoRa-Knoten optimiert werden.

Die Arbeit [30] schlägt Verbesserungen bezüglich des LoRaWAN Protokolles vor. Dabei wurde die Möglichkeit erforscht, dass das Gateway das Hören von Paketen über das Senden priorisiert.

Ein weiterer Vorschlag ist, den ACK Mechanismus zu eliminieren oder zu reduzieren, was die Anzahl der Kollisionen verringern könnte und somit das Gateway nicht so schnell seinen Arbeitszyklus erreicht [2].

5.2 Simulationen

Viele Simulationstools wurden für die Analyse verschiedene Probleme entwickelt. In den folgenden Abschnitten sind die bekanntesten Simulationsprogramme, ihre Eigenschaften und Anwendungsbeispiele kurz beschrieben.

5.2.1 LoRaSim

LoRaSim ist ein auf SimPy basierender Ereignisdiskreter Simulator zur Simulation von Kollisionen in LoRa-Netzwerken und zur Analyse der Skalierbarkeit [52]. LoRaSim erlaubt uns N LoRa-Knoten und M LoRa-Basisstationen in einem 2-dimensionalen Raum zu platzieren. Eine Nachricht gilt als empfangen, wenn mindestens eine LoRa Basisstation des entsprechenden Netzes sie empfängt. Die Übertragung ist erfolgreich empfangen, wenn die empfangene Signalleistung über der Empfindlichkeitsschwelle des Gateways liegt. Wie stark die empfangene Signalleistung ist, hängt von der Sendeleistung und allen Verstärkungen und Verlusten entlang des Übertragungsweges ab.

LoRaSim verwendet Python-Skripte für die Simulation. Damit ist es möglich, die Skalierbarkeit und den Leistungsverbrauch entweder mit einem oder mehreren Gateways zu

simulieren. Außerdem kann man entscheiden, ob die Endgeräte direktionale oder omnidirektionale Antenne haben. Mit LoRaSim ist es auch möglich, konkurrierende Netzwerke zu simulieren. Leider unterstützt LoRaSim keine Downlink Nachrichten und ADR Verhalten. Dadurch eignet sich LoRaSim eher für die Analyse der physischen Schicht, und weniger für die MAC Schicht.

LoRaSim ist relativ einfach zu benutzen. Das Programm hat keine grafische Oberfläche und der Ausgang ist eine .dat Datei, die mit anderen Plot-Programmen abgebildet werden kann.

Mit LoRaSim kann man zum Beispiel, den Einfluss der Anwendung mehrere Gateways in Vergleich zur Verwendung eines einzigen Gateways simulieren. Auf dem Bild 5.1 wurde die Simulation mit einer aufsteigenden Anzahl von Knoten erst mit einem Gateway, und dann mit zwei Gateways durchgeführt. Auf der vertikalen Achsen ist das Verhältnis von erfolgreich empfangene Nachrichten zu gesendete Nachrichten dargestellt. Dabei sieht man, dass das Hinzufügen eines Gateways die Qualität des Netzes erheblich erhöht [57].

In der Arbeit [56] wurde dieser Simulator intensiv genutzt um die Verwendung von Richtantennen sowohl durch die Gateways als auch durch die Endgeräte zu veranschaulichen. Überdies wurde auch die Interferenz zwischen mehreren sich überlappenden Netzwerken untersucht.

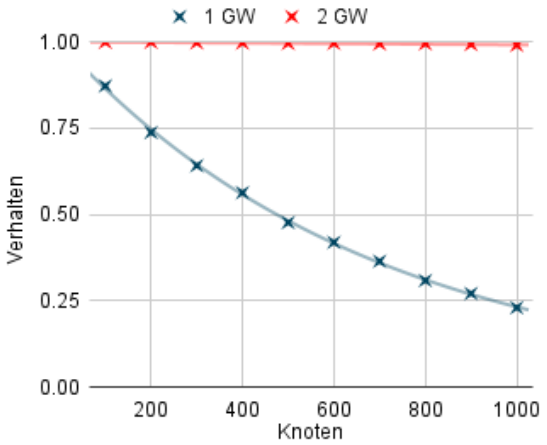

Abbildung 5.1: Gateway Skalierbarkeit

5.2.2 NS-3

NS-3 ist einen Netzwerksimulator auf C++ und Python basierend, der nicht nur LoRa, sondern auch andere Protokolle simuliert, wie Wi-Fi, LTE, SigFox, usw. Bei andere Simulatoren werden oft vereinfachte Versionen des Protokolls verwendet. Dagegen wurden für NS-3 verschiedene LoRaWAN Module entwickelt, um ein reales LoRaWAN Netzwerk zu simulieren. Dadurch kann eine umfassende Analyse sowohl der physischen Schicht als auch der MAC Schicht in LoRa Netzwerken gemacht werden.

Dazu enthält NS-3 einen Paket-Tracer, der das Verhalten eines Netzwerks überwachen und analysieren kann. Die LoRaWAN Module unterstützen nur Klasse A Geräte, das heißt, dass Downlink-Nachrichten ebenfalls unterstützt sind.

Mit NS-3 wird zum Beispiel, die Verwendung von Carrier Sense Multiple Access (CSMA) in einem LoRaWAN Netzwerk untersucht. Bei CSMA sollen alle Teilnehmer den Status der Kanäle beobachten und ihre Nachrichten nur senden, wenn kein anderer Teilnehmer sendet und der Kanal frei ist. Die Arbeit [54] schlägt die Verwendung einer CSMA Variante vor. Dabei sollen Endgeräte für ein kleines Zeitintervall den Kanal abhören, bevor eine Übertragung versucht wird.

Die Ergebnisse sind, dass der Leistungsverbrauch der Endgeräte leicht steigt, und somit steigt auch der Akkuverbrauch. Die Kollisionen werden aber so weit reduziert und infolgedessen die sich wiederholende Übertragungen verringert. Der Batterieverbrauch wird dadurch reduziert. Mit weniger Übertragungen insgesamt lässt sich das Netzwerk auch weiter skalieren.

5.2.3 FLoRa

FloRa wurde entwickelt, um den ADR Mechanismus zu analysieren. Es stützt sich auf den OMNeT++ Netzwerk Simulator. Eine der größten schwachen von LoRaWAN ist, dass die LoRa Allianz keinen offiziellen Algorithmus für die adaptive Datenrate (ADR) zur Verfügung stellt und dies den Netzbetreibern überlässt. Demzufolge spiegeln viele kleinere Netzwerke den ADR-Algorithmus von größeren Netzwerken wie TNN (The Things Network). TNN ist derzeit das größte öffentliche LoRaWAN-Netzwerk auf dem Markt.

Forscher arbeiten intensiv daran, den bestmöglichen ADR-Algorithmus zu entwickeln. Unter allen Simulatoren ist FloRa der beste, um die Wirksamkeit verschiedener ADR-Algorithmen in Netzen unterschiedlicher Größe zu simulieren, sowohl in der Stadt als auch auf dem Land.

In der Arbeit [47] haben die Autoren den ADR-Algorithmen von TNN mit einem von ihnen selbst entwickelten Algorithmus verglichen und FloRa benutzt, um die Effizienz ihrer Algorithmen zu beweisen. Das Ergebnis ist, dass mit einem besser implementierten Algorithmus die Skalierbarkeit von LoRaWAN sowohl in der Stadt als auch auf dem Land erheblich gesteigert werden kann.

5.3 Zusammenfassung

Unabhängig von Programmiersprachen, die in den Implementierungen der Simulatoren verwendet werden, sind alle Simulatoren auf bekannten Programmierumgebungen entwickelt worden. Dies ist essenziell, da ein Spezialist die Fähigkeiten des Simulators schnell erweitern und neue Module für die bestehenden Simulatoren implementieren kann [2].

6 Implementierung und Integration eines LoRaWAN Netzwerk

6.1 Aufgabestellung

In den folgenden Abschnitten soll eine komplettes LoRaWAN Netzwerk implementiert werden, um ein tieferes Verständnis über die Arbeitsweise des Protokolls zu schaffen. Dabei soll eine Ende-zu-Ende Kommunikation erfolgreich ablaufen, d. h. die Sensormessungen am Endgerät sollen in der Applikationschicht dargestellt werden. Die notwendige LoRaWAN Schichten müssen implementiert werden. Das Gateway sollte einsatzbereit sein und auf dem Dach eines hohen Gebäudes platziert werden können. Deshalb sollte das Gateway auch wetterfest sein. Mit der Implementierung wird auch erzielt, dass die Installation für die Verbraucher sehr einfach wird. Am besten sollte das Gateway einsatzbereit sein, sobald es in die Stromversorgung eingesteckt wird.

6.2 Hardware Anforderungen

Folgendes sind die Mindestvoraussetzungen für den Entwurf eines LoRaWAN Netzwerk:

- 1x LoRa Endgerät
- 1x Gateway

Bei der Auswahl der Hardware ist zu berücksichtigen, dass es sich um einen Prototyp handelt. Es sollte auf fertige Entwicklungsboards zurückgegriffen werden, um entsprechend Zeit und Kosten zu sparen. Außerdem werden weitverbreitete Chips mit hoher öffentlicher Akzeptanz bevorzugt, da diese bessere Verfügbarkeit von nutzbaren Bibliotheken und Anwendungsprogramme bieten.

Das verwendete Frequenzband sollte vor der Hardwareauswahl schon bekannt sein. In Deutschland dürfen LoRa Geräte im 433 MHz und im 868 MHz Frequenzband betrieben werden. Wenn man beide Frequenzbänder in einem Spectrum Analyser betrachtet, kann man sehen, dass das 868 MHz Band nicht so stark belegt ist wie das 433 MHz Band (Bilder 6.1 und 6.2). Die Spektren wurden mit dem Gerät SDRplay Model RSP 1 und mit das Programm SDRuno aufgenommen [35]. Auf den Bildern sieht man die Aktivität in dem 433 MHz band und wie der Grundrausch im Vergleich zu dem EU868 band etwas höher ist. Außerdem wird das 868 MHz-Band von der LoRaWAN Community und von den Dienstleistern häufiger genutzt. Für dieses Band existieren auch bereits

Frequenzpläne. Es müssen demzufolge kein neuer Frequenzplan erstellt werden und die Implementierung wird dadurch erleichtert [39].

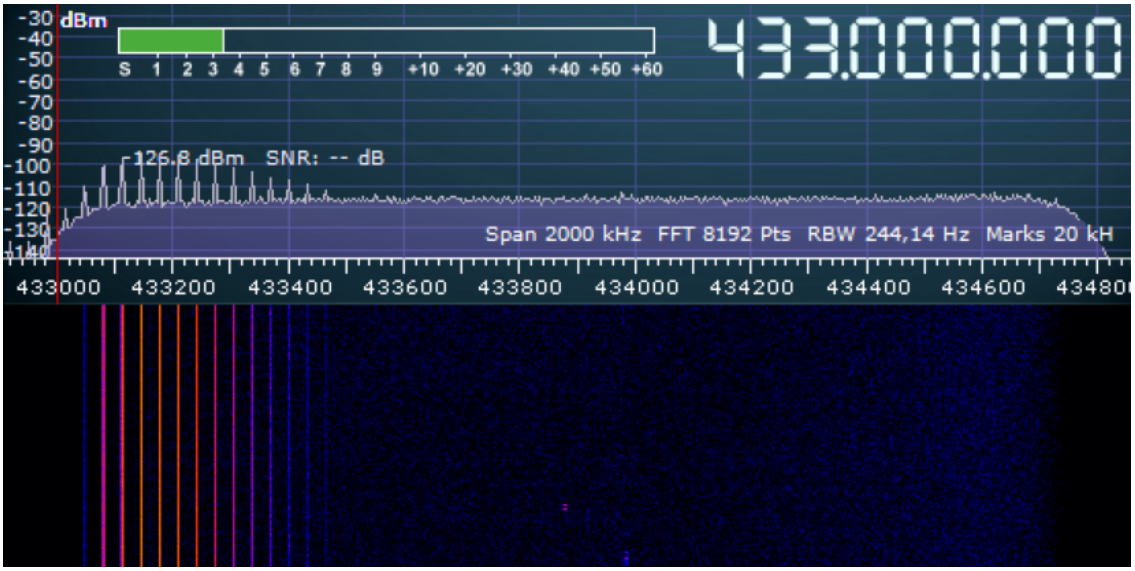

Abbildung 6.1: SDRUno EU433

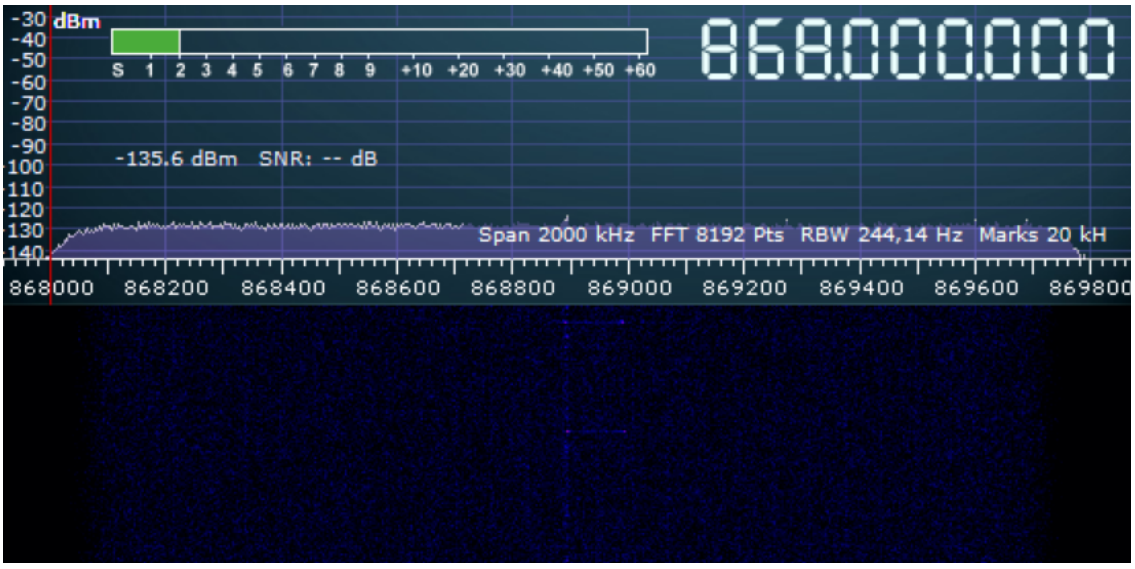

Abbildung 6.2: SDRUno EU868

6.2.1 LoRa Module von SEMTECH

- **Module für LoRa Gateways**
  Ein Gateway besteht aus vier Teilen: einem Hostrechner, einem digitalen Signalverarbeitungsprozessor (DSP) und zwei Sendeanlagen (Bild 6.4). Wahlweise kann auch ein GPS-Empfänger gebaut werden. Der Hostrechner kann ein beliebiger Rechner sein und wird nicht von SEMTECH vermarktet. Eine Übersicht von allen SEMTECH Produkten befindet sich in SEMTECH *„LoRa Products“* Guide [42] und auf der SEMTECH Website [44].

SEMTECH bietet zurzeit vier DSP. Die Produktnummerierung von solchen Chips beginnt mit SX130. Der SX1308 ist für den Innenbereich und kleinere Netzwerke gedacht. Der SX1301 ist der beliebteste Chip. Er wurde speziell entwickelt, um bahnbrechende Gateway Funktionen in den ISM-Bändern weltweit anzubieten. Die SX1302 und 1303 sind die jüngeren Versionen des SX1301. Sie bieten einen geringeren Stromverbrauch und ein höheres Verkehrsaufkommen im Vergleich zu den frühen Geschwistern. Die SX1303 bietet zusätzlich eine neue Funktion für höher auflösbare Zeitstempel. SEMTECH bietet für alle DSP die HAL auf ihrem Github an [43].
Die Sendeanlagen werden von SEMTECH *„Analog Front End ICs“* genannt. Diese sind je nach gewünschtem Frequenzband und DSP auszuwählen. Das Datenblatt der Prozessoren zeigt, welche Sendeanlage sie unterstützt. Zurzeit gibt es vier Varianten solcher Anlagen. Sie sind mit SX125 gekennzeichnet. In einem DSP können zwei Sendeanlagen gekoppelt werden. Es ist möglich zwei Sendeanlagen des selben Models zu verbauen, um mehr Pakete in einem Band empfangen zu können oder eine Kombination aus verschiedenen Sendeanlagen, um Pakete in unterschiedlichen Bändern zu empfangen.

- **Module für LoRa Endgeräte**
  SEMTECH bietet auch verschiedene Modelle für Endknoten an. Sie sind hauptsächlich je nach Frequenzband auszuwählen. Die Produktnamen fangen mit SX127 für die älteren Versionen und mit SX126 für die neueren Versionen an.
- **Sonstige Produkte**
  Die anderen Produkte im LoRa Katalog sind verschiedene Entwicklungskits.

6.2.2 Grundaufbau

Endgerät

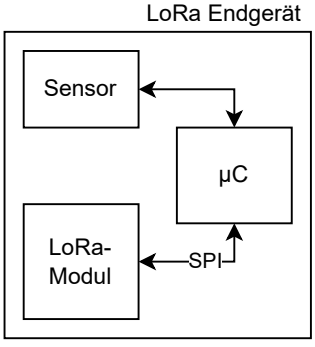

Abbildung 6.3: Endgerät Grundaufbau

Für das Endgerät benötigt man mindestens einen LoRa Transceiver und einen Mikrocontroller. Damit die Implementierung einem realen Anwendungsfall entspricht, wird ebenfalls ein Sensor zum Einsatz gebracht.

Zunächst ist es wichtig zu beachten, dass der LoRaWAN Chip mit 3,3V betrieben wird. Um den Entwurf zu vereinfachen, sollten unterschiedliche Spannungswerte in der Schaltung vermieden werden. Deshalb sollten der Mikrokontroller (µU) und der Sensor auch in 3,3V funktionsfähig sein.

Bei der Auswahl eines LoRa Transceiver ist zu beachten, dass das Gerät in dem ausgewählten Frequenzbereich funktionsfähig ist. Die Module SX1276 und SX1272 sind in Europa

am weitesten verbreitet. Das SX1276 unterstützt sowohl das 433 MHz Band als auch das 868 MHz Band, während das SX1272 nur das 868 MHz Frequenzband unterstützt.

Es wird ebenfalls eine Antenne benötigt. Für den Prototypen reicht eine Kabelantenne aus, aber zur Produktfertigung sollte auf etwas Stabileres zurück gegriffen werden. Wie lang eine Kabelantenne für praktische Zwecke sein soll, entspricht dabei ungefähr der Hälfte einer Wellenlänge. Die Wellenlänge einer Frequenz wird als v/f berechnet, wobei v die Übertragungsgeschwindigkeit und f die (durchschnittliche) Übertragungsfrequenz ist. In der Luft ist v gleich c, der Lichtgeschwindigkeit, die $299.792.458 m/s$ beträgt. Die Wellenlänge für das 868-MHz-Band beträgt also $299.792.458/868.000.000 = 34,54 cm$. Die Hälfte davon ist $17,27 cm$.

Die empfohlenen Anforderungen für den Mikrocontroller können im Dokument *„MCU Requirements for LoRaWAN“* von SEMTECH[45] nachgeschaut werden. Der Arm M0+ ist für IoT Anwendungen,wegen des geringen Energieverbrauchs, sehr beliebt.

Der Sensor kann ein beliebiger Sensor sein. Dabei ist nur zu beachten, dass eine SPI Schnittstelle bereits vom LoRa Modul belegt ist.

Das Arduino Framework ist weitverbreitet, Open-Source, hat eine sehr große Community und eine Menge Beispielprojekte. Alternativ dazu bietet sich das mbed Framework an, welches eine reife LoRaWAN Bibliothek der Herstellerseite bietet. Die Entwicklungsplatine sollte deshalb am besten eine der beiden Frameworks unterstützen.

Gateway

Für ein Gateway ist ein Hostrechner und ein Gatewaymodul mindestens vorausgesetzt. Außerdem empfiehlt SEMTECH die Installation eines GPS-Empfängers am Gateway, um Standortdienste implementieren zu können. Das Gatewaymodul besteht aus einem DSP und zwei Funkgeräten.

Im Prinzip kann jeder Rechner als Host ausgewählt werden, der eine SPI Schnittstelle zu Verfügung stellt. Der Raspberry Pi ist für Prototypen sehr beliebt, weil er kostengünstig ist und ein GPIO Interface bietet. Außerdem läuft darauf standardmäßig ein Linux-basierte Distribution, was die Installation des SEMTECH HAL und anderen benötigten Paketen stark vereinfacht.

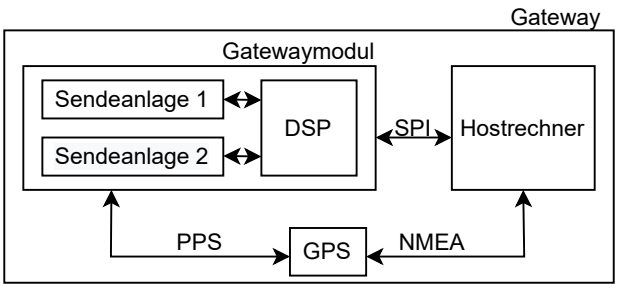

Abbildung 6.4: Gateway Grundaufbau

Die Anzahl von Endgeräten, die sich mit einem Gateway verbinden können, ist bei diesem Entwurf eher irrelevant, da zunächst das Gateway mit nur einem Endgerät verbunden wird. Da höhere Übertragungsentfernungen erwünscht sind, ist die Implementierung mit dem SX1308 ausgeschlossen.

Bei dem Gatewaymodul ist zu beachten, dass die zwei eingebetteten Sendeanlagen in der richtigen Sendefrequenz funken können. Die SPI Pins des Gatewaymoduls sollten über eine Steckerleiste herausgeführt sein, um die Verbindung mit dem Raspberry Pi einfach herstellen zu können.

6.2.3 Produktauswahl

Basiert auf dem oben beschriebenen Anforderungen, bietet sich das LoRa/LoRaWAN Gateway Kit von Seeeduino für den Prototyp an:

- RHF0M301 LoRa Gateway
- Seeeduino LoRaWAN mit einem SX1276 LoRa Transceiver
- Raspberry Pi 3b+

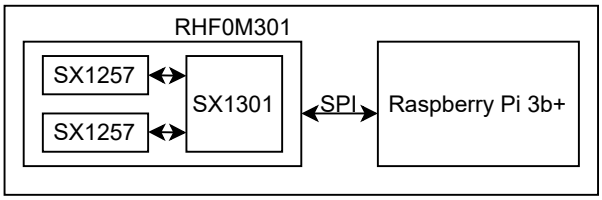

Abbildung 6.5: Mein Gateway

RISINGHF bietet Entwicklungsplatinen mit einem SX1301, einem Global Positioning System (GPS)-Empfänger und den bereits zwei verbauten Sendeanlagen an. Das SX1301 erreicht sehr große Distanzen und kann sich mit ausreichend vielen Endknoten verbinden. Dazu bietet RISINGHF noch einen Adapterplatine für das Raspberry PI. Mit dem Adapterplatine werden Verkabelung vermieden, damit ist das Gerät zum Schluss stabiler. Man kann das Modul je nach Frequenzband auswählen. Im RISINGHF Wiki kann man alle Module sehen, die zur Verfügung stehen [20]. Ausgewählt wurde ein RHF0M301 868 MHz mit zwei SX1257 Sendeanlagen. Mit diesen Sendeanlagen ist es nur möglich in das EU868 Frequenzband zu übertragen. Das Senden im 433MHz Band ist somit ausgeschlossen. Im Entwicklungskit ist auch eine Antenne für das EU868 Band enthalten.

Die Seeeduino LoRaWAN Entwicklungsplatine ist mit einem SX1276 und einem ARM Cortex M0+ ausgestattet. Außerdem kommt das Modul mit einem GPS-Empfänger, einem Batteriestecker und einer eingelöteten Kabelantenne. Seeeduino bietet verschiedene Bibliotheken für das Arduino Framework auf ihrer Website an. Dort befinden sich auch einige Beispiele dafür, wie das Board in den Energiesparmodus versetzt wird, wie LoRaWAN Pakete gesendet und empfangen werden und weitere Beispiele.

Die Seeeduino LoRaWAN hat unglücklicherweise keinen eingebetteten Sensor, weshalb noch ein zusätzlicher Sensor benötigt wurde. Ausgewählt wurde einen DHT11 Temperatursensor. Dieser ist problemlos zu bedienen und benötigen nur einen Data Pin für die Kommunikation mit dem Board. Der Sensor wurde mit der Seeeduino LoRaWAN über den Pin 4 verbunden.

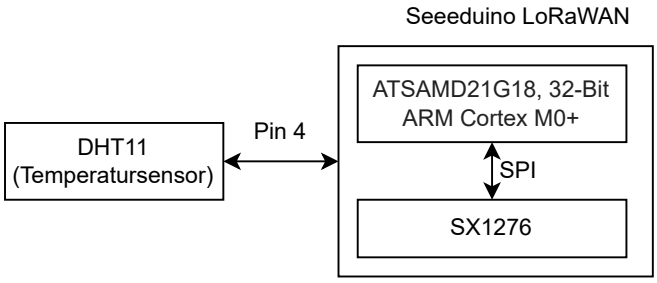

Abbildung 6.6: Mein Endgerät

6.2.4 Gateway Aufbau

Damit das Gateway größere Übertragungsreichweiten erreichen kann, muss der Pfadverlust minimiert werden. Der beste Weg, um in städtischen Umgebungen einen geringen Pfadverlust zu erreichen, besteht darin, die Antenne im Freien so hoch wie möglich zu platzieren.

Das Gateway wird dadurch den Witterungsbedingungen ausgesetzt sein. Deshalb muss dieser in einem wasserdichten Gehäuse verbaut sein. Für den Einbau des Gateways in einem wasserfesten Gehäuse wurde folgendes verwendet:

- POE Adapter für Raspberry Pi
- POE Wandadapter
- wasserdichte Kabeldurchführungen
- Abstandshalter und Schrauben
- Antennenverlängerung (10 cm)
- wasserdichtes Gehäuse
- Ethernet Kabel

Anders als bei den Endknoten, ist der Leistungsverbrauch eines Gateways sehr hoch. Das Gateway sollte deshalb über eine ständige Stromversorgung verfügen. Damit die Anzahl der Löcher, welche in das Gehäuse eingearbeitet werden müssen, gering bleibt und die Installation so einfach wie möglich läuft, wird ein Power Over Ethernet (POE) Adapter verwendet. Ein POE ist in der Lage, Strom und Daten gleichzeitig über Twisted-Pair-Ethernet-Verkabelung bereitzustellen. Dadurch wird lediglich ein Kabel für die Stromversorgung und die Ethernetverbindung benötigt.

Das Ethernetkabel muss durch ein gebohrtes Loch in das Gehäuse geführt werden.

Das Regenwasser stellt hier eine Gefahr für die Schaltung im Inneren des Gehäuses dar. Deshalb sollte das Eindringen von Wasser in das Gehäuse, unter allen Umständen verhindert werden. Eine Kabeldurchführung wird verwendet, um das Loch zu isolieren und das Problem somit zu beseitigen. Der Stecker des Ethernetkabels durfte erst gecrimpt werden, nachdem die Kabeldurchführung ordnungsgemäß im Loch befestigt wurde und ein Ende des Kabels im Gehäuse war.

Die Gateway Schaltung sollte im Gehäuse fest verbaut sein und nicht wackeln. Hier werden die Befestigungslöcher vom Raspberry Pi mit Schrauben am Boden des Gehäuses befestigt. Die RHF0M301 und die Ansteckerplatine werden mit den Abstandshaltern auf dem Raspberry gestapelt und gefestigt.

Die Antenne soll außerhalb des Gehäuses angebracht werden. Eine Antenneverlängerung wird verwendet, um den Antennenausgang des Gateways zu einer der Gehäuseseite zu bringen. Die Kabelbuchse des SMA-Kabels musste durch ein gebohrtes Loch zur Außenseite des Gehäuses geführt und mit einer Mutter gefestigt werden.

Das Gehäuse muss nicht nur für das Gateway ausreichend sein, sondern auch für alle Kabel, Erweiterungen und Adapter, die auch angeschlossen werden müssen. Somit ist das Gehäuse für das Gateway fertig. Das Gehäuse des Endgerätes kann in einem ähnlichen Verfahren gefertigt werden.

6.3 Software Implementierung

6.3.1 Netzwerkanbieter

Um ein eigenes LoRaWAN aufzubauen, könnte man einen kommerziellen Anbieter oder eine frei verfügbare Lösung nutzen. Sie umfassen die Rolle des NS und AS im LoRaWAN Protokoll und abstrahieren den Protokollstack, um ein LoRaWAN einfsvh einzurichten. Die drei folgenden Netzwerke sind in Deutschland verfügbar und bieten kostenlose Optionen zur Gestaltung eines LoRaWAN-Netzwerkes.

- **Loriot:** Loriot bietet eine wachsende verteilte LoRaWAN Infrastruktur und eine benutzerfreundliche Softwareplattform, die es ermöglicht, ein sicheres IoT-Netzwerk aufzubauen, zu betreiben und zu skalieren. Die kostenlose Version kann unendliche Gateways aufnehmen, aber diese werden mit dem gesamten Loriot-Netz geteilt. Man darf bis auf 30 Endgeräte in das Netzwerk eintragen.
- **TTN:** TTN funktioniert ähnlich wie Loriot. Es bietet eine kostenlose und entwicklerfreundliche Möglichkeit, um sich mit der LoRaWAN Technologie vertraut zu machen. Der Vorteil bei TTN ist die Community. Sie hat mehr als 100.000 Mitglieder aus mehr als 100 Ländern weltweit und ist somit der meistverbreitete Netzwerkserveranbieter. Das Gateway wird auch geteilt. Die kostenlose Version

von TTN begrenzt die Anzahl an Geräten nicht, es hat aber eine Begrenzung an gesendeten Nachrichten pro Tag. Durchschnittlich 30 Sekunden Uplink-Zeit pro 24 Stunden pro Gerät und höchstens 10 Downlink Nachrichten pro 24 Stunden, einschließlich die ACKs für bestätigte Uplinks, sind erlaubt. Diese 30 Sekunden Uplink-Zeit ist für viele Anwendungen ausreichend, sie liegt jedoch weit unter dem täglich erlaubten Arbeitszyklus in Europa.
Sowohl bei Loriot als auch bei TNN wird kein Gateway selbst benötigt, solange ein zum Netzwerk gehörendes Gateway noch erreichbar ist. Weil kein Gateway auf beiden Netzen in der Reichweite gefunden wurden, musste ein Gateway aufgebaut werden.

- **Chirpstack:** Chirpstack ist eine Open-Source Implementierung von LoRaWAN, und anders als Loriot und TNN, hat es keine physisches Netzwerkinfrastruktur. Chirpstack ist die bessere Lösung für den Aufbau eines privaten Netzwerks.
  Die Gateways werden nicht mit anderen geteilt und das Netzwerk hat keine Begrenzung. Jedoch ist die Installation nicht so einfach wie bei den bisher genannten. Außerdem hängt die Sicherheit des Netzwerks sehr von der Implementierung ab. Wenn der Protokollstack erforscht wird, eignet sich Chirpstack am besten. Da der Code Open-Source ist, kann man das Programm auf Wunsch anpassen.

Während die zwei andere Netzwerke sehr einfach zu implementieren sind, wurde eine private Umsetzung bevorzugt. Deshalb wurde Chirpstack ausgewählt. Mit Chirpstack ist es möglich ein privates Netzwerk aufzubauen, darüber hinaus ist es offener für zukünftige Anpassungen. Da in dieser Arbeit hauptsächlich den Protokollstack untersucht wird, eignet sich Chirpstack dafür am besten.

6.3.2 Raspberry Pi Einstellung

Für den Raspberry Pi benötigt man eine ausreichend große SD-Karte. Eine SD-Karte mit 16 GB Speicherplatz wird empfohlen. Als Betriebssystem reicht das Raspberry Pi OS Lite Betriebssystem aus. Die komplette Version ist mit einem Graphical User Interface (GUI) ausgestattet, während die Lite Version nur das Terminal beinhaltet. Da auf den Raspberry Pi nur ohne Monitor durch Secure Shell (SSH) zugegriffen wird, kann man auf das UI verzichten und spart dadurch eine Menge Speicherplatz. Empfohlen wird die Installation mit dem Raspberry Pi Imager [34].

Da das Gateway sehr hoch platziert wird, ist es nicht mehr erreichbar, sobald es auf dem Dach steht. Deshalb ist es wichtig, den Fernzugriff zu ermöglichen. Mit dem Raspberry Pi Imager kann man das SSH bereits bei der Installation konfigurieren.

SSH, auch bekannt als Secure Shell oder Secure Socket Shell, ist ein Netzwerkprotokoll, das Benutzern, insbesondere Systemadministratoren, eine sichere Möglichkeit bietet, über ein Netzwerk auf einen Computer zuzugreifen.

Ersteinrichtung

Zunächst muss das Raspberry Pi Betriebssystem in der SD-Karte installiert werden. Anweisungen für die Installation befinden sich auf der Raspberry Pi Website [34].

Im „CHOOSE OS“ Menü ist das Raspberry Pi OS Lite (32-Bit) und im „CHOOSE STORAGE“ Menü ist die jeweilige SD-Karte auszuwählen. Durch einen Klick auf das Zahnrad-Symbol können weitere Einstellungen vorgenommen werden. Dort muss der SSH-Zugriff konfiguriert werden. Der ausgewählte Benutzername und das Passwort müssen sicher gespeichert werden, da es in der Zukunft oft benötigt wird. Zusätzlich können auch die lokale Zeitzone und Tastaturlayout eingerichtet werden. Zum Schluss kann man auf „WRITE“ klicken, damit das Betriebssystem auf die SD-Karte übertragen wird. Danach kann die SD-Karte im Raspberry Pi eingesteckt werden.

Der Raspberry Pi soll jetzt mit Strom versorgt werden. Für den SSH-Zugriff kann ein beliebiger SSH Client auf einem beliebigen Rechner verwendet werden. Für diesen Arbeit wurde die Remote-SSH Erweiterung für Visual Studio Code benutzt ([31], [32]). Für den Zugriff muss die Raspberry PI IP und das in der Installation ausgewählte Passwort bekannt sein.

Nach dem ersten Start müssen einige Einstellungen vorgenommen werden. Das Raspberry-Einstellungsmenü kann mit folgendem Befehl initialisiert werden:

```
$ sudo raspi-config
```

Im geöffneten Fenster muss folgendes aktiviert werden:

- Expand File System (Dateisystem erweitern): damit der ganze verfügbare Speicher in der SD-Karte genutzt werden kann.
- Enable SPI
- Enable SSH

Danach kann das Fenster geschlossen werden. Nun soll das SSH konfiguriert werden, damit es nach jedem Hochfahren automatisch gestartet wird. Dafür müssen in die Kommandozeile folgende Befehle eingetippt werden:

```
$ sudo apt-et install ssh
$ sudo /etc/init.d/ssh start
$ sudo update-rc.d ssh defaults
```

Damit ist gesichert, dass SSH beim Hochfahren aktiviert ist. Um die übergenommenen Änderungen zu überprüfen, kann das Raspberry mit folgendem Befehl neu gestartet werden.

```
$ sudo reboot
```

Nach ein paar Sekunden kann man versuchen der SSH Client mit dem Raspberry Pi wieder zu verbinden. Wenn die Verbindung erfolgt, ist alles in Ordnung.

6.3.3 Ende-zu-Ende Paketverwaltung

Endgerät

In der Realisierung wurde nur ein aktives Endgerät als Prototyp verwendet. Das Endgerät verschickt jede zehn Minuten eine Nachricht. Diese Nachricht ist 10 Byte lang. Für das Endgerät wurden das Arduino Framework und die von Seeeduino bereitgestellten Bibliotheken verwendet [36]. Um das Programm in die Entwicklungsplatine hochzuladen wurde die Arduino IDE verwendet. Auf der Seeeduino Website befinden sich auch Informationen, wie man die Arduino IDE für die Seeeduino LoRaWAN einstellt. Folgender Code wurde für den Prototyp verwendet:

```
#include <LoRaWan.h>
#include <dht11.h>
#define DHT11PIN 4

char buffer[256];
dht11 DHT11;

void setup(void)
{
    lora.init();
    while(!lora.setOTAAJoin(JOIN));
}

void loop(void)
{
    DHT11.read(DHT11PIN);
    int a = DHT11.temperature;
    char buf [2];
    sprintf (buf, "%02i", a);
```

```
    lora.transferPacket(buf, 10);
    delay(10*60*1000);
}
```

Beim Einschalten versucht das Endgerät sich an einem LoRaWAN Netzwerk zu verbinden. Das wird wiederholt, bis das Gerät erfolgreich mit einem Netzwerk verbunden ist. Das Join-Verfahren muss bei jedem Einschalten durchgeführt werden, da die generierten Sitzungsschlüssel in dem flüchtigen Speicher des Gerätes gespeichert sind. Wenn die Verbindung erfolgreich ist, tritt das Gerät in die Hauptschleife ein. In der Hauptschleife liest das Gerät einmal den Sensor, sendet ein Paket mit der Temperatur, wartet 10 Minuten und wiederholt den Vorgang.

Gateway-HAL

SEMTECH stellt den SX1301 HAL in seinem GitHub zur Verfügung [40]. In diesem Repository befinden sich alle Programme, die für die Hardwaresteuerung benötigt werden. Außerdem befinden sich auch dort ein paar nützliche Programme zum Testen der Funktionen des Gateways.

Vor der Verwendung des Gateways wird empfohlen, den `util_spi_stress"'` Test durchzuführen. Diese Software wird verwendet, um die Zuverlässigkeit der SPI Verbindung zwischen dem Hostrechner und dem Gatewaymodul zu testen. Auf diese Weise kann sichergestellt werden, dass es keine Probleme mit der Pinbelegung oder Verkabelung gibt.

Bevor der Packet-Forwarder gestartet wird, muss das Programm `reset_lgw.sh` ausgeführt werden, um das Gatewaymodul über GPIO zurückzusetzen.

Ersteinrichtung

Man kann den SEMTECH HAL mit der Kommandozeile wie folgt herunterladen:

```
$ sudo apt install git
$ git clone https://github.com/Lora-net/lora_gateway.git
```

Die erste Zeile installiert das Git Versionskontrollsystem und die Zweite erstellt eine Kopie des Repositorys lokal auf dem Rechner. Das Ausführen des `util_spi_stress` Programms hat folgende Fehlermeldungen zurückgegeben:

```
ERROR: Failed to load fw 1
ERROR: Version of calibration firmware not expected, actual:0
expected:2
ERROR: [main] failed to start the concentrator
```

Das lag an einer Aktualisierung des Kernels des Raspberry Pi, die die Einstellungen für den SPI geändert hat. Eine Lösung wäre, dass das Gateway an anderen SPI Pins verbunden wird. Dadurch wird der RisingHF Hat unbrauchbar. Eine bessere Lösung, die es ermöglicht die Adapterplatine weiterzuverwenden, wird über das Hinzufügen folgender Zeile in der `/boot/config.txt` erreicht [29]:

```
dtoverlay=spi0-cs,cs1_pin=25
```

Am besten soll das `util_spi_stress` Programm noch einmal ausgeführt werden, um zu testen, ob alles in Ordnung ist. Jetzt kann das Gateway mit

```
$ ./reset_lgw.sh start
```

gestartet werden. Der nächste Schritt ist, den Packet-Forwarder einzustellen.

Packet-Forwarder

Der Packet-Forwarder ist ein Programm, das auf dem Hostrechner eines LoRa-Gateways läuft und die vom Gatewaymodul empfangenen Pakete an einen Server weiterleitet und vom Server gesendete Pakete aussendet ([41],[38]).

Vor der Ausführung des Hauptprogramms ist es notwendig den Frequenzplan und den Ausgangsport zu konfigurieren. Außerdem kann ausgewählt werden, ob das Gateway alle Pakete weiterleitet, oder nur Pakete mit gültiger CRC-Prüfsumme. Diese Konfigurationen werden in den Dateien `global_conf.json` und `local_conf.json` vorgenommen. SEMTECH bietet auch mehrere Beispiele für diese Dateien mit vorgefertigten Konfigurationen für verschiedene Frequenzpläne an.

Die Art und Weise, wie das Programm Konfigurationsdateien berücksichtigt, ist die folgende:

1. Wenn es eine `debug_conf.json` gibt, wird diese gelesen, andere werden ignoriert
2. Wenn es eine `global_conf.json` gibt, wird diese gelesen und nach der nächsten Datei gesucht
3. Wenn es eine `local_conf.json` gibt, lese sie. Wenn einige Parameter sowohl in globalen als auch in lokalen Konfigurationsdateien definiert sind, überschreibt die lokale Definition die globale Definition.

Das Skript `update_gwid.sh` ermöglicht die automatische Aktualisierung der `Gateway_ID` mit einer eindeutigen MAC-Adresse in der JSON-Konfigurationsdatei.

Das Program kann dann mit dem `./lora_pkt_fwd` gestartet werden.

Ersteinrichtung

Das Packet-Forwarder Programm kann man ebenfalls aus GitHub klonen [41]:

```
$ git clone https://github.com/Lora-net/packet_forwarder.git
```

Der Packet-Forwarder wird mit den Konfigurationsdateien `global_conf.json` und `local_conf.json` eingestellt. Die Verwendung ist wie folgt:

1. Als Ausgangspunkt wählen Sie die JSON-Konfigurationsdateien aus dem Verzeichnis `cfg/` die zu ihrem Gateway, Region und den benötigten Funktionen passt. Für dieser Arbeit eignet sich die `global\_conf.json.PCB\_E286.EU868.basic` am besten. Kopieren Sie den Inhalt dieser Datei in die `global_conf.json` Konfigurationsdatei. Passen Sie die Konfigurationsdateien wie gewünscht an. Hier können einige Einstellungen geändert werden, wie das Modell der Frontend-Radios, den gewünschten Frequenzplan, die Standard Ports usw. Mit dem `local_conf.json` können gatewayspezifische Einstellungen, wie die Gateway-ID, vorgenommen werden. Die hier verwendeteten `global_conf.json` und `local_conf.json` Dateien befinden sich im Anhang.
2. Führen Sie `./update_gwid.sh local_conf.json` aus, um die Gateway-ID zu aktualisieren.
3. Führen Sie das `./lora_pkt_fwd` Programm aus, um das Packet-Forwarder zu starten.

Ab diesem Punkt können alle LoRaWAN Pakete, die vom Gateway empfangen werden, in der Kommandozeile gesehen werden. Um diese Funktionalität nachzuweisen, kann mit dem Endgerät ein LoRaWAN Paket in der Gateway-Reichweite gesendet werden. Nur Pakete mit dem LoRaWAN Header werden erkannt. Der Ausgang jedes empfangenen Paket ist im JSON Format. Das Root-Objekt kann ein Array namens “rxpk” enthalten:

```
json {
"rxpk":[ {...}, ...]
}
```

Dieses Array enthält mindestens ein JSON-Objekt, jedes Objekt enthält ein RF-Paket und zugehörige Metadaten. Diese JSON Upstream wird in den ausgewählten Port geleitet. Die Pakete sind verschlüsselt. Zudem ist das Packet-Forwarder nicht in der Lage, sie zu entschlüsseln. Dafür muss das LoRaWAN Netzwerk implementiert werden.

Wenn das Programm mit Ctrl+c verlassen wird, wird das Programm abgebrochen und somit kein Paket mehr empfangen. Deshalb sollte das Programm parallel zu den anderen Gateway-Programmen laufen. Dafür wurde ein `systemd` Service verwendet, um das Programm im Hintergrund laufen zu lassen. Man kann den Service auch so einstellen, dass der Packet-Forwarder beim Hochfahren ausgeführt wird.

Systemd ist für die Verwaltung aller auf dem System laufenden Dienste über die gesamte Betriebszeit des Rechners, vom Startvorgang bis zum Herunterfahren, zuständig. Damit kann man Programme im Boot starten und das Program mit dem `systemctl` Kommando steuern. Mit dem `systemctl` Kommando können Befehle an `systemd` Services gesendet werden, z.B. zum Starten, Stoppen, zum Abfragen des Status, usw. Chirpstack verwendet auch `systemd`, um seine Server zu verwalten. Im Anhang befindet sich die hier verwendete `.service` Datei.

Netzwerk

Chirpstack ist dafür verantwortlich, den Packet-Forwarder abzuhören, das LoRaWAN Netzwerk zu verwalten und die Nachricht an eine Endanwendung weiterzuleiten. Folgende Komponenten werden bereitgestellt:

- **Chirpstack Gateway Bridge:** Die Chirpstack Gateway Bridge befindet sich zwischen dem Packet-Forwarder und dem MQTT-Broker. Sie wandelt das Packet-Forwarder-Format in ein Datenformat um, das von dem Chirpstack-Netzwerk-Server verwendet wird.
  MQTT (Message Queuing Telemetry Transport) ist ein Protokoll für begrenzte Netzwerke mit geringer Bandbreite und IoT-Geräte mit extrem hoher Latenz. Da MQTT auf Umgebungen mit geringer Bandbreite und hoher Latenz spezialisiert ist, ist es ein ideales Protokoll für die Machine-to-Machine (M2M) Kommunikation. MQTT arbeitet nach dem Publisher/Subscriber-Prinzip und wird über einen zentralen Broker betrieben. Das bedeutet, dass Sender und Empfänger keine direkte Verbindung haben. Die Datenquellen melden ihre Daten über einen Publish und alle Empfänger mit Interesse an bestimmten Nachrichten bekommen die Daten zugestellt, da sie sich als Abonnenten angemeldet haben.
- **Chirpstack Network-Server:** Der Chirpstack Network Server ist eine LoRaWAN Network Server Implementierung. Er ist für die Verwaltung des Netzwerkstatus verantwortlich, hat Kenntnis von Geräteaktivierungen im Netzwerk und ist in der Lage Join-Anfragen zu bearbeiten, wenn Geräte dem Netzwerk beitreten wollen.

Die Chirpstack Implementierung hat nur einen Netzwerkserver (Home-Server), Roaming ist somit ausgeschlossen.

- **Chirpstack Application-Server:** Der Chirpstack Application-Server ist eine LoRaWAN Application Server-Implementierung. Er hört den Netzwerkserver ab und dient als Brücke zwischen dem Netzwerk-Server und andere Integrationsmöglichkeiten. Der Applikation Server bietet auch einen API-Client und einen Webansicht, womit der Nutzer das Netzwerk steuern kann. Dort kann man neue Gateways und Endgeräte in das Netzwerk hinzufügen, Anwenderzugriff verwalten, usw. Der Chirpstack Application-Server dient auch als Join-Server.

Ersteinrichtung

- **Chirpstack Gateway Bridge Installation:** Damit die Chirpstack Gateway Bridge korrekt funktioniert wird einem MQTT-Broker Vorausgesetzt. Mosquitto ist ein beliebter Open-Source MQTT-Broker, der das MQTT-Protokoll implementiert, Um Mosquitto zu installieren, führen Sie den folgenden Befehl aus [9]:

  ```
  $ sudo apt install mosquitto
  ```

  Da alle Pakete mit einem PGP-Schlüssel signiert werden, muss dieser Schlüssel zunächst importiert werden:

  ```
  $ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00
  ```

  So wird das Chirpstack-Repository zu Ihrem System hinzugefügt:

  ```
  $ sudo echo "deb https://artifacts.chirpstack.io/packages/3.x/deb stable main" | sudo tee /etc/apt/sources.list.d/chirpstack.list
  $ sudo apt-get update
  ```

  Um die Chirpstack Gateway Bridge zu installieren, führen Sie den folgenden Befehl aus:

  ```
  $ sudo apt install chirpstack-gateway-bridge
  ```

  Um dem Chirpstack Gateway Bridge zu starten:

  ```
  $ sudo systemctl start chirpstack-gateway-bridge
  ```

- **Chirpstack Netzwerkserver Installation**
  Damit der Chirpstack Netzwerkserver korrekt funktioniert, braucht er eine eigene Datenbank. Um eine Datenbank zu erstellen, installieren Sie PostgreSQL mit folgendem Befehl:

  ```
  $ sudo apt install postgresql
  ```

  Um eine neue Datenbank zu erstellen, starten Sie die PostgreSQL-Kommandozeile als Benutzer postgres:

  ```
  $ sudo -u postgres psql
  ```

  Geben Sie in der PostgreSQL Kommandozeile folgende befehle:

```
# create role chirpstack_ns with login password 'dbpassword';
# create database chirpstack_ns with owner chirpstack_ns;
# \q
```

In der erste Zeile wird einem „chirpstack_ns“ User mit dem Passwort „dbpassword“ erzeugt. In der zweite Zeile wird die „chirpstack_ns“ Datenbank erstellt. Mit der dritten Zeile verlässt man die PostgreSQL Kommandozeile. Um zu überprüfen, ob der Benutzer und die Datenbank korrekt eingerichtet wurden, versuchen Sie, sich mit der Datenbank zu verbinden:

```
$ psql -h localhost -U chirpstack_ns -W chirpstack_ns
```

Außerdem wird Redis und Mosquitto vorausgesetzt [10]. Mosquitto wurde bereits vor der Installation der Gateway Bridge installiert. Um Redis zu installieren, führen Sie den folgenden Befehl aus:

```
$ sudo apt install redis-server
```

Um den Chirpstack Netzwerkserver zu installieren, führen Sie den folgenden Befehl aus (vorausgesetzt, dass das Chirpstack-Repository bereits installiert ist) [8].

```
$ sudo apt install chirpstack-network-server
```

Bearbeiten Sie die Konfigurationsdatei mit den PostgreSQL-Zugangsdaten:

```
$ sudo nano /etc/chirpstack-network-server/chirpstack-
network-server.toml
```

Entfernen Sie die folgende Zeile

```
dsn="postgresql://localhost/chirpstack_ns_ns?sslmode=disable"
```

und fügen Sie folgende Zeile hinzu:

```
dsn = 'user=chirpstack_ns dbname=chirpstack_ns password=
dbpassword sslmode=disable'
```

Starten Sie den Netzwerkserver mit:

```
$ sudo systemctl start chirpstack-network-server
```

- **Chirpstack Applikationsserver Installation**
  Eine eigene Datenbank muss auch für die Applikationsserver erzeugt werden:

```
$ sudo -u postgres psql
# create role chirpstack_as with login password 'dbpassword';
# create database chirpstack_as with owner chirpstack_as;
```

  Die Erweiterungen "‘pg_trgm"’ (trigram) und "´hstore"’ mussen aktiviert werden:

```
# \c chirpstack_as
# create extension hstore;
# \q
```

  Um den Chirpstack Applikationsserver zu installieren, führen Sie den folgenden Befehl aus (vorausgesetzt, dass das Chirpstack-Repository bereits installiert ist):

```
$ sudo apt-get install chirpstack-application-server
```

  Nach der Installation ändern Sie die Konfigurationsdatei, die sich unter `/etc/chirpstack-applicati` befindet. Sie müssen die Konfigurationsvariablen `postgres.dwn` wir folgt ändern:

```
postgres://chirpstack_as:dbpassword@localhost/chirpstack_as
?sslmode=disable
```

Eine weitere erforderliche Einstellung, die Sie ändern müssen, ist:

```
jwt_secret="[Passwort]"
```

Sie könnten ein zufälliges Passwort durch Ausführung vom folgenden Befehl erzeugen:

```
openssl rand -base64 32
```

Der Ausgang wird dann in die `chirpstack-application-server.toml` Datei kopiert.

- **Konfigurationsdateien**
  Die Konfigurationsdateien heißen `chirpstack-gateway-bridge.toml` [5], `chirpstack-networ` [6] und `chirpstack-application-server.toml` [7]. Sie können mit einem beliebigen Texteditor bearbeitet werden (z.B. nano). Standardmäßig suchen alle Chirpstack Programme in der folgenden Reihenfolge nach einer Konfigurationsdatei:

  1. aktuelles Arbeitsverzeichnis
  2. `$HOME/.config/<chirpstack-verzeichnis>/`
  3. `/etc/<chirpstack-verzeichnis>/`

  Der Wert in `<chirpstack-verzeichnis>` kann entweder `chirpstack-gatew ay-bridge`, `chirpstack-network-server` oder `chirpstack-application-s erver` sein.
  Wichtig ist, dass bei jeder Kommunikationsschnittstelle der Server auf den richtige Port hört und an den richtigen Port weiterleitet. Ein Port erlaubt die Kommunikation zwischen zwei Computern sowie mit dem Internet.
  Standardmäßig sind die Chirpstack Komponenten richtig konfiguriert. Hauptsächlich ist zu beachten, dass die Gateway-Bridge auf den in dem Paket Forwarder `global_conf.json` konfigurierten Port hört. Dieser Port wird in der `chirpstack-gateway-brid` Datei wie folgt gesetzt:

  ```
  [backend.semtech_udp]
      udp_bind = "0.0.0.0:1680"
  ```

  Der ganze Umfang von Konfigurationen, die in den .toml Dateien gemacht werden können, liegt außerhalb des Rahmens dieses Dokuments. Im Anhang sind die hier verwendeten Konfigurationsdateien.

6.4 Webansicht Anwendung

Um neue Geräte im Netzwerk hinzufügen, wird die Chirpstack Webansicht verwendet. Die Webansicht befindet sich unter http://141.55.229.15:8080/. Der Benutzername lau-

tet admin und das Passwort auch. In der linken Spalte kann man verschiedene Optionen einstellen (Bild 6.7):

1. „Dashboard“ ist die Startseite.
2. Unter „Network-Server“ können Netzwerkserver zum API zugeordnet werden.
3. Unter „Gateway_profiles“ können Gatewayprofile erstellt werden. Das ist z. B.dann nützlich, wenn einem Netzwerk mehreren Gateways angehören und sie in verschiedenen Frequenzbändern arbeiten. Man kann die Gateways gruppieren, anstatt sie einzeln zu konfigurieren.
4. Ein Netzwerk kann mehreren Firmen bedienen. Unter „Organizations“ können neue Organisationen hinzugefügt werden.
5. Unter „All Users“ können die API Benutzer verwaltet werden.
6. für einige Endpunkte muss einen API-Schlüssel eingegeben werden. Der API-Schlüssel kann unter „API-Keys“ generiert werden.
7. In die Klappliste kann einer der Organisationen ausgewählt werden. Alles unter der Klappliste sind Konfigurationen ausschließlich für die Organisation einzustellen.
8. „Org. Dashboard“ zeigt allgemeine Informationen über die Organisation.
9. Zu einer Organisation können auch private Benutzer (a) und API-Schlüsseln (b) zugeordnet werden.
10. Unter „Service-Profile“ kann ein neues Dienstprofil erstellt werden. Das Dienstprofil beschreibt die Funktionen, die für den Benutzer aktiviert sind und die Rate der Nachrichten, die über das Netz gesendet werden können [11].
11. Unter „Device-Profiles“ kann man ähnlich wie bei den Gateways auch Geräte-Profile hinzufügen. Einem Gerätprofil enthaltet gerätspezifische Informationen wie die LoRaWAN Version, der ADR-Algorithmus, das Join-Verfahren usw. Mehrere Geräte können ein Gerätprofil übernehmen.
12. Unter „Gateways“ kann eine Übersicht über alle zu der Organisation zugehörigen Gateways erhalten werden. Man kann auf einem Gateway klicken und eine Zusammenfassung aller gesendeten und empfangenen Paketen sehen. Dort kann man auch Gateway spezifische Konfigurationen vornehmen, wie ein eindeutiger Name und der Gateway-Standort.
13. Unter „Applications“ kann man unterschiedlichen Applikationen erstellen. Ein Gateway gehört zum Netzwerk und wird von allen Applikationen geteilt. Ein Endgerät gehört zu der Applikation und muss mit seinem EUI eindeutig spezifiziert werden.

    a) Um ein Endgerät zu einer Applikation zuzuordnen gehen sie wie folgt vor: 1. in „Applications“ > MeinAppName gehen (Bild 6.8). 2. Stellen Sie sicher, dass Sie sich in dem „Devices“-Menü befinden. Klicken Sie auf den Button

„CREATE“ (Bild 6.9. Dort wird nach einem Namen, einem Geräteprofil und nach der Gerät EUI gefragt. Der Name kann ein beliebiger Namen sein. Der Gerät EUI soll vom Hersteller gegeben werden.

b) In der Applikation kann man auch einen Endpunkt unter dem Tab „Integrations“ setzen. Chirpstack bietet unterschiedliche Integrationsmöglichkeiten an. Unter anderen: MQTT, AWS SNS, Azure Service-Bus, GCP Pub/-Sub, HTTP, InfluxDB, myDevices, PilotThings, SEMTECH LoRa Cloud, und ThingsBoard.io.

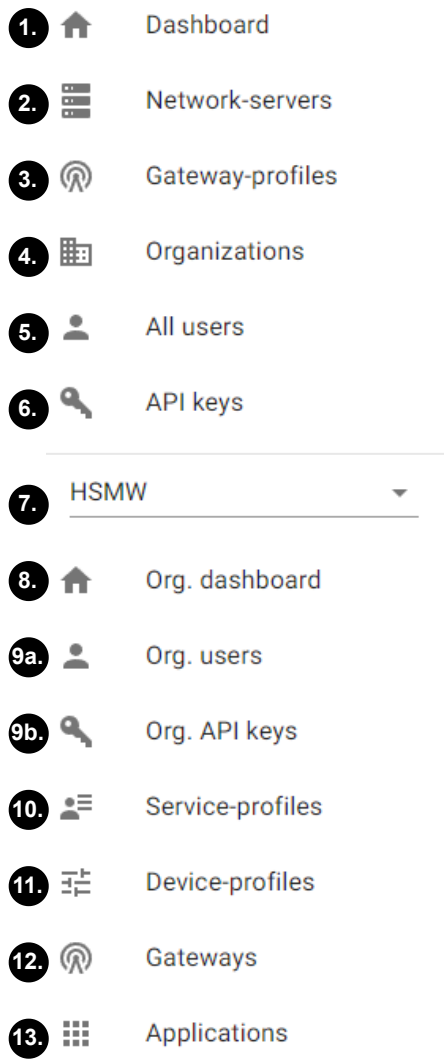

Abbildung 6.7: Chirpstack Linke Balken

Endpunkt

Hier wurde die HTTP Intrgration verwendet. Obwohl es teilweise aufwendiger ist, bietet die HTTP Integration dennoch deutlich mehr Flexibilität und Raum für zukünftige Erwei-

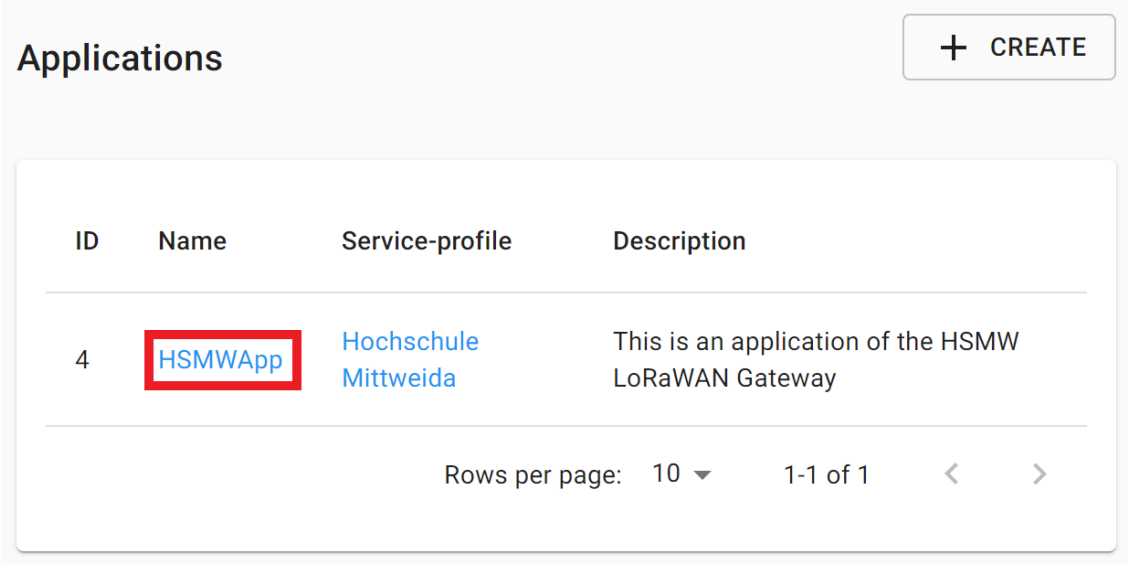

Abbildung 6.8: Application

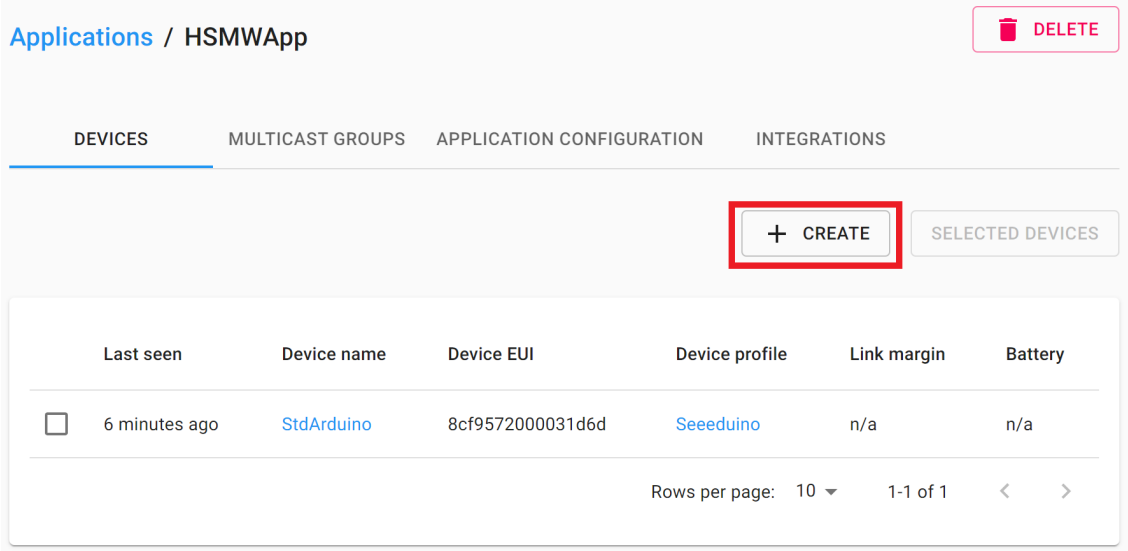

Abbildung 6.9: Devices

terung. Mit der HTTP Integration sendet Chirpstack zu einem ausgewählten Endpunkt eine POST Request mit einem JSON im HTTP Request Body für jedes empfangene Paket.

Dafür wurde einen Webserver mit Python auf dem Localhost des Raspberry Pi eingerichtet. Für die Requests-Verwaltung wurde die Bibliothek Flask verwendet. Der Webserver hört auf die POST Requests der Chirpstack API. Die Sensordaten werden in einer Datenbank gespeichert. Für die Datenbankverwaltung wurde die die Python Bibliothek SQLAlchemy verwendet. Zur Datendarstellung wurde eine HTML Tabelle und ein Diagramm mit der Matplotlib Bibliothek erstellt. Bei jedem POST und GET Request wird eine Zeile zu der Tabelle und eine neue Datenzeile in der Datenbank hinzugefügt. Zusätzlich wird das Diagramm mit dem neuen Datenpunkt neu geladen.

Das sollte nicht als komplettes Endprodukt gesehen werden, sondern als Beispiel, wie man die Integration der Chirpstack API verwenden könnte. Dieses Beispielprogramm befindet sich im Verzeichnis `/home/rxhf/AppLayer/flask-sensors`. Das Programm hat auch einer zugehörigen `systemd .service` Datei unter `/etc/systemd/system/app.service`.

Der Aufbau eines WebServers hängt nicht mit dem LoRaWAN Protokoll zusammen und ist deswegen außerhalb des Umfangs dieses Dokuments

6.4.1 Zusammenfassung

In diesem Kapitel wurde ausführlich erklärt, wie alle Gateway-Programme auf dem Raspberry Pi installiert wurden. Die hier verwendete Implementierung ist nicht mit dem Internet verbunden und deshalb nur im Hochschulnetz erreichbar. In der Tabelle 6.4.1 sind die wichtigsten Gateway-Informationen zusammengefasst.

Gateway Informationen	
Hostname	caro.local
Username	rxhf
Password	risinghf
Raspberry IP	141.55.229.15
Chirpstack API Endpunkt	141.55.229.15:8080
Chirpstack API Username	admin
Chirpstack API Password	admin
Temperatursensor Endpunkt	141.55.229.15:3333

Tabelle 6.1: wichtige Gateway-Informationen

7 Auswertung und zukünftige Arbeit

In dieser Arbeit wurde die Grundlage für den Aufbau eines privaten LoRaWAN-Netzwerks vorgestellt. Dabei wurden alle Produkte von SEMTECH beschrieben, und alle Schichten des LoRaWAN Protokolls, wie von der Lora-Alliance vorgeschlagen, erläutert. Es gibt noch einige Verbesserungen, die vorgenommen werden können.

Was die Sicherheit betrifft, könnte die Kommunikation zwischen den Servern mit weiteren Einstellungen in den .toml Dateien angepasst werden. Die Sicherheit der SSH-Kommunikation mit dem Gateway könnte durch die Verwendung von SSH-Schlüsseln verbessert werden.

Idealerweise sollte nur der Packet-Forwarder auf dem Hostrechner des Gateways laufen. Alle anderen Server können in einem anderen Rechner installiert werden. Zur Vereinfachung wurden alle Server im Raspberry Pi installiert. Wenn das Netzwerk zunehmend nach oben skaliert, wäre es von Vorteil, das Netzwerk und den Anwendungsserver auf einen leistungsfähigeren Computer zu übertragen.

Für die Interaktion mit den Daten, die vom Anwendungsserver kommen, könnte eine schnellere und schönere Benutzerschnittstelle gebaut werden. Der Webserver könnte mit einer Web-Domain an das Internet angeschlossen werden, sodass sie auch außerhalb des Hochschulnetzes zugänglich ist.

Man könnte auch andere Gateways in das bereits existierende Netzwerk hinzufügen. Der GPS-Empfänger der RHF0M301 könnte in Einsatz gebracht werden. Mit einem zweiten Gateway können neue Funktionen getestet werden, wie beispielweise die Ermittlung des Standorts von Endknoten durch Triangulationsverfahren [23].

In der Chirpstack Implementierung hat der Anwendungsserver die Rolle des Join-Servers. Eine Möglichkeit wäre es, einen Join-Server zu implementieren, und somit alle von LoRaWAN definierten Schichten voneinander zu trennen. Die Implementierung könnte auch erweitert werden, um Roaming zu ermöglichen.

Der Endknoten verfügt über kein Batteriemanagementsystem. Ein Energiesparmodus könnte im Code implementiert werden, um die Batterie über Jahre hinweg zu betreiben. Die Nutzung nichtflüchtiger Speicher könnte überprüft werden, damit das Endgerät nicht bei jedem Hochfahren erneut das Join-Verfahren durchführen muss.

Anhang A: Anlage

A.1 Konfigurationsdateien

A.1.1 Packet-Forwarder

global_conf.json

/home/rxhf/risinghf/pktfwd/global_conf.json

```
{
"SX1301_conf": {
    "lorawan_public": true,
    "clksrc": 1, /* radio_1 provides clock to concentrator */
    "antenna_gain": 0, /* antenna gain, in dBi */
    "radio_0": {"enable": true,"type": "SX1257","freq": 867500000,"
   rssi_offset": -166.0,"tx_enable": true,"tx_freq_min": 863000000,"
   tx_freq_max": 870000000},
    "radio_1": {"enable": true,"type": "SX1257","freq": 868500000,"
   rssi_offset": -166.0,"tx_enable": false},
    "chan_multiSF_0": {"enable": true,"radio": 1,"if": -400000},
    "chan_multiSF_1": {"enable": true,"radio": 1,"if": -200000},
    "chan_multiSF_2": {"enable": true,"radio": 1,"if": 0},
    "chan_multiSF_3": {"enable": true,"radio": 0,"if": -400000},
    "chan_multiSF_4": {"enable": true,"radio": 0,"if": -200000},
    "chan_multiSF_5": {"enable": true,"radio": 0,"if": 0},
    "chan_multiSF_6": {"enable": true,"radio": 0,"if": 200000},
    "chan_multiSF_7": {"enable": true,"radio": 0,"if": 400000},
    "chan_Lora_std": {"enable": true,"radio": 1,"if": -200000,"
   bandwidth": 250000,"spread_factor": 7},
    "chan_FSK": {"enable": true,"radio": 1,"if": 300000,"bandwidth":
    125000,"datarate": 50000},
    "tx_lut_0": {"pa_gain": 0,"mix_gain": 8,"rf_power": -6,"dig_gain
   ": 0},
    "tx_lut_1": {"pa_gain": 0,"mix_gain": 10,"rf_power": -3,"
   dig_gain": 0},
    "tx_lut_2": {"pa_gain": 0,"mix_gain": 12,"rf_power": 0,"dig_gain
   ": 0},
    "tx_lut_3": {"pa_gain": 1,"mix_gain": 8,"rf_power": 3,"dig_gain
   ": 0},
```

```
    "tx_lut_4": {"pa_gain": 1,"mix_gain": 10,"rf_power": 6,"dig_gain
   ": 0},
    "tx_lut_5": {"pa_gain": 1,"mix_gain": 12,"rf_power": 10,"
   dig_gain": 0},
    "tx_lut_6": {"pa_gain": 1,"mix_gain": 13,"rf_power": 11,"
   dig_gain": 0},
    "tx_lut_7": { "pa_gain": 2,"mix_gain": 9,"rf_power": 12,"
   dig_gain": 0},
    "tx_lut_8": {"pa_gain": 1,"mix_gain": 15,"rf_power": 13,"
   dig_gain": 0},
    "tx_lut_9": {"pa_gain": 2,"mix_gain": 10,"rf_power": 14,"
   dig_gain": 0},
    "tx_lut_10": {"pa_gain": 2,"mix_gain": 11,"rf_power": 16,"
   dig_gain": 0},
    "tx_lut_11": {"pa_gain": 3,"mix_gain": 9,"rf_power": 20,"
   dig_gain": 0},
    "tx_lut_12": {"pa_gain": 3,"mix_gain": 10,"rf_power": 23,"
   dig_gain": 0},
    "tx_lut_13": {"pa_gain": 3,"mix_gain": 11,"rf_power": 25,"
   dig_gain": 0},
    "tx_lut_14": {"pa_gain": 3,"mix_gain": 12,"rf_power": 26, "
   dig_gain": 0},
    "tx_lut_15": {"pa_gain": 3,"mix_gain": 14,"rf_power": 27,"
   dig_gain": 0}
},
"gateway_conf": {
    /* change with default server address/ports, or overwrite in
   local_conf.json */
    "gateway_ID": "AA555A0000000000",
    "server_address": "localhost",
    "serv_port_up": 1680,
    "serv_port_down": 1680,
    /* adjust the following parameters for your network */
    "keepalive_interval": 10,
    "stat_interval": 30,
    "push_timeout_ms": 100,
    /* forward only valid packets */
    "forward_crc_valid": true,
    "forward_crc_error": true,
    "forward_crc_disabled": true
    }
}
```

local_conf.json

/home/rxhf/risinghf/pktfwd/local_conf.json

```
{
    "gateway_conf": {
        "gateway_ID": "205518e710504889",
        "autoquit_threshold": 5,
        "server_address": "localhost",
        "serv_port_up": 1680,
        "serv_port_down": 1680
    }
}
```

pktfwd.service

/home/rxhf/risinghf/pktfwd/pktfwd.service

```
[Unit]
Description=packet forwarder
Wants=network-online.target
After=network-online.target

[Service]
Restart=always
RestartSec=5
WorkingDirectory=/home/rxhf/risinghf/pktfwd/
ExecStartPre=/usr/local/sbin/gwrst
ExecStartPre=/home/rxhf/risinghf/pktfwd/update_gwid.sh /home/rxhf/
   risinghf/pktfwd/local_conf.json
ExecStart=/home/rxhf/risinghf/pktfwd/pktfwd
ExecStopPost=/usr/local/sbin/gwrst

[Install]
WantedBy=multi-user.target
```

A.1.2 Chirpstack

chirpstack-application-server.toml

/etc/chirpstack-application-server/chirpstack-application-server.toml

```
[postgresql]
dsn="postgres://chirpstack_as:dbpassword@localhost/chirpstack_as?
   sslmode=disable"
[redis]
url="redis://localhost:6379"
[application_server]
  [application_server.integration]
  marshaler="json_v3"
  enabled=["mqtt"]
    [application_server.integration.mqtt]
    event_topic_template="application/{{ .ApplicationID }}/device/{{
    .DevEUI }}/event/{{ .EventType }}"
    command_topic_template="application/{{ .ApplicationID }}/device
   /{{ .DevEUI }}/command/{{ .CommandType }}"
    server="tcp://localhost:1883"
    username=""
    password=""
    [application_server.api]
    bind="0.0.0.0:8001"
    public_host="localhost:8001".
    [application_server.external_api]
    bind="0.0.0.0:8080"
    tls_cert=""
    tls_key=""
    jwt_secret="fq+Mi3y3TW0by9SoRS4Jkg8w20SqLjSEb3i6+C0bJl0="
[join_server]
bind="0.0.0.0:8003"
```

chirpstack-gateway-bridge.toml

/etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml

```
[backend]
type="semtech_udp"
  [backend.semtech_udp]
  udp_bind = "0.0.0.0:1680"
[integration]
marshaler="protobuf"
  [integration.mqtt]
  event_topic_template="gateway/{{ .GatewayID }}/event/{{ .EventType
    }}"
  command_topic_template="gateway/{{ .GatewayID }}/command/#"
  [integration.mqtt.auth]
  type="generic"
    [integration.mqtt.auth.generic]
    server="tcp://127.0.0.1:1883"
    username=""
    password=""
```

chirpstack-gateway-bridge.toml

/etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml

```
[postgresql]
dsn="postgres://chirpstack_ns:dbpassword@localhost/chirpstack_ns?
   sslmode=disable"
[redis]
url="redis://localhost:6379"
[network_server]
net_id="000000"
  [network_server.band]
  name="EU868"
  [network_server.network_settings]
    [[network_server.network_settings.extra_channels]]
    frequency=867100000
    min_dr=0
    max_dr=5
    [[network_server.network_settings.extra_channels]]
    frequency=867300000
    min_dr=0
    max_dr=5
    [[network_server.network_settings.extra_channels]]
    frequency=867500000
    min_dr=0
    max_dr=5
    [[network_server.network_settings.extra_channels]]
    frequency=867700000
    min_dr=0
    max_dr=5
    [[network_server.network_settings.extra_channels]]
    frequency=867900000
    min_dr=0
    max_dr=5
    [network_server.network_settings.class_b]
    ping_slot_dr=0
    ping_slot_frequency=0
  [network_server.api]
  bind="0.0.0.0:8000"
  [network_server.gateway.backend]
  type="mqtt"
    [network_server.gateway.backend.mqtt]
    event_topic="gateway/+/event/+"
```

```
    command_topic_template="gateway/{{ .GatewayID }}/command/{{ .
   CommandType }}"
    server="tcp://localhost:1883"
    username=""
    password=""
[metrics]
timezone="Local"
[join_server]
  [join_server.default]
  server="http://localhost:8003"
```

Literatur

[1] LoRa Alliance. *About the LoRaWAN® Standards*. [Online; Stand 14. Marz 2022]. URL: https://lora-alliance.org/lorawan-for-developers/.

[2] Mukarram AM Almuhaya u. a. „A Survey on LoRaWAN Technology: Recent Trends, Opportunities, Simulation Tools and Future Directions“. In: *Electronics* 11.1 (2022), S. 164.

[3] Bundesnetzagentur. *Allgemeinzuteilung von Frequenzen zur Nutzung durch Funkanwendungen geringer Reichweite (SRD)*. [Online; Stand 13. April 2022]. Dez. 2020. URL: https://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Frequenzen/Allgemeinzuteilungen/FunkanlagenGeringerReichweite/2018_05_SRD_pdf.pdf?__blob=publicationFile&v=7.

[4] Ismail Butun, Nuno Pereira und Mikael Gidlund. „Security risk analysis of LoRaWAN and future directions“. In: *Future Internet* 11.1 (2019), S. 3.

[5] Chirpstack. *Configuration*. https://www.chirpstack.io/gateway-bridge/install/config/. [Online; Stand 10. Mai 2022].

[6] Chirpstack. *Configuration*. https://www.chirpstack.io/network-server/install/config/. [Online; Stand 10. Mai 2022].

[7] Chirpstack. *Configuration*. https://www.chirpstack.io/application-server/install/config/. [Online; Stand 10. Mai 2022].

[8] Chirpstack. *Debian / Ubuntu installation*. https://www.chirpstack.io/gateway-bridge/install/debian/. [Online; Stand 10. Mai 2022].

[9] Chirpstack. *Requirements*. https://www.chirpstack.io/gateway-bridge/install/requirements/. [Online; Stand 10. Mai 2022].

[10] Chirpstack. *Requirements*. https://www.chirpstack.io/network-server/install/requirements/. [Online; Stand 10. Mai 2022].

[11] Chirpstack. *Service profiles*. https://www.chirpstack.io/application-server/use/service-profiles/. [Online; Stand 16. Mai 2022].

[12] LoRa Alliance Technical Committee. „LoRaWAN® Backend Interfaces Technical Specification (TS002-1.1.0)“. In: (Okt. 2020).

[13] LoRa Alliance Technical Committee. „LoRaWAN® L2 1.0.4 Specification (TS001-1.0.4)“. In: (Okt. 2020).

[14] LoRa Alliance Technical Committee. *RP002-1.0.3 LoRaWAN® Regional Parameters*. 2021. URL: https://lora-alliance.org/resource_hub/rp2-1-0-3-lorawan-regional-parameters/.

[15] Ronald Creutz. „Entwurf und Implementierung eines IoT Systems zur Informationsgewinnung in der Agrarwirtschaft (hier: Imkerei) unter Nutzung von LoRaWan“. Diss. 2019.

[16] Tahsin CM Dönmez und Ethiopia Nigussie. „Security of lorawan v1. 1 in backward compatibility scenarios“. In: *Procedia computer science* 134 (2018), S. 51–58.

[17] Eric B. *AN1200.22 LoRa Modulation Basics*. https://lora.readthedocs.io/en/latest/\#id27. [Online; Stand 11. Mai 2022].

[18] Kevin Flynn. *Standardisation of NB-IoT Completed*. Abgerufen am 18.02.2022. URL: https://www.3gpp.org/news-events/3gpp-news/1785-nb_iot_complete.

[19] Jetmir Haxhibeqiri u. a. „A survey of LoRaWAN for IoT: From technology to application“. In: *Sensors* 18.11 (2018), S. 3995.

[20] RISING HF. *Product Brief - RHF0M301*. [Online; Stand 24. April 2022]. URL: https://wiki.risinghf.com/en/01/01/03/01/#ordering-information.

[21] Melanie Hofheinz. *Karlsruhe wird zur Smart City: Mülleimer sollen künftig selbst denken*. [Online; Stand 15. Marz 2022]. URL: https://www.ka-news.de/wirtschaft/karlsruhe-innovativ/Karlsruhe-wird-zur-Smart-City-Muelleimer-sollen-kuenftig-selbst-denken;art516981,2177874.

[22] *Ingenu Inc.* Abgerufen am 18.02.2022. URL: https://www.ingenu.com/technology/rpma/.

[23] Johannes Dultz. *LoRaWAN-Tracking ohne GPS*. https://www.lora-wan.de/anwendungen/lorawan-tracking-gps/. [Online; Stand 11. Mai 2022].

[24] Léon Klick. „Entwurf und Implementierung eines LoRaWAN-basierten Sensornetzes für die quantitative Bewertung einer Waldbrandgefahr“. In: (2021).

[25] Robert Koning. „LoRaWAN als Treiber der digitalen Stadt“. In: *Smart City – Made in Germany: Die Smart-City-Bewegung als Treiber einer gesellschaftlichen Transformation*. Hrsg. von Chirine Etezadzadeh. Wiesbaden: Springer Fachmedien Wiesbaden, 2020, S. 659–669. ISBN: 978-3-658-27232-6. DOI: 10.1007/978-3-658-27232-6_68. URL: https://doi.org/10.1007/978-3-658-27232-6_68.

[26] Robert Koning, Timo Stricker und Peter Schneider. „Was ist LoRa und LoRaWAN“. In: *Smart City Solutions GmbH Kahrlsruhe* (2017).

[27] Rachel Kufakunesu, Gerhard P Hancke und Adnan M Abu-Mahfouz. „A survey on adaptive data rate optimization in lorawan: Recent solutions and major challenges“. In: *Sensors* 20.18 (2020), S. 5044.

[28] *LORIOT*. Abgerufen am 18.02.2022. URL: https://loriot.io/.

[29] ludionisio. *Upgrade to kernel 5.4.51-v7+ breaks Raspberry Pi Hats that use SPI and expect GPIO 7 to be free*. https://github.com/Lora-net/lora_gateway/issues/162. [Online; Stand 10. Mai 2022].

[30] Davide Magrin, Martina Capuzzo und Andrea Zanella. „A Thorough Study of LoRaWAN Performance Under Different Parameter Settings“. In: *IEEE Internet of Things Journal* 7.1 (2020), S. 116–127. DOI: 10.1109/JIOT.2019.2946487.

[31] Microsoft. *Remote development over SSH*. https://code.visualstudio.com/docs/remote/ssh-tutorial. [Online; Stand 10. Mai 2022].

[32] Microsoft. *Visual Studio Code*. https://code.visualstudio.com/. [Online; Stand 10. Mai 2022].

[33] *Nwave Technologies Inc.* Abgerufen am 18.02.2022. URL: https://www.nwave.io/.

[34] Raspberry Pi. *Raspberry Pi OS*. [Online; Stand 24. April 2022]. URL: https://www.raspberrypi.com/software/.

[35] SDRPlay. *SDRuno*. https://www.sdrplay.com/sdruno/. [Online; Stand 16. Mai 2022].

[36] Seeeduino. *Seeeduino LoRaWAN*. [Online; Stand 24. April 2022]. URL: https://wiki.seeedstudio.com/Seeeduino_LoRAWAN/.

[37] SEMTECH. *AN1200.22 LoRa™ Modulation Basics*. 2015. URL: %5Curl%7Bhttps://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001OJk/yDEcfAkD9qEz6oG3PJryoHKas3UMsMDa3TFqz1UQOkM%7D (besucht am 18. 10. 2021).

[38] SEMTECH. *Basic communication protocol between Lora gateway and server*. [Online; Stand 3. Marz 2022]. URL: https://github.com/Lora-net/packet_forwarder/blob/master/PROTOCOL.TXT.

[39] SEMTECH. *global_conf.json Frequency Plans*. https://github.com/Lora-net/packet_forwarder/tree/master/lora_pkt_fwd/cfg. [Online; Stand 16. Mai 2022].

[40] SEMTECH. *LoRa Gateway project*. [Online; Stand 7. Marz 2022]. URL: https://github.com/Lora-net/lora_gateway.

[41] SEMTECH. *LoRa network packet forwarder project*. [Online; Stand 7. Marz 2022]. URL: https://github.com/Lora-net/packet_forwarder.

[42] SEMTECH. *LoRa Products - Long Range, Low Power Consumption Secure Devices to Cloud Solutions*. [Online; Stand 17. April 2022]. 2021. URL: https://www.semtech.com/uploads/selector-guides/SEMTECH_LORA_PG_web.pdf.

[43] SEMTECH. *LoRa-Net GitHub*. [Online; Stand 16. Marz 2022]. URL: https://github.com/Lora-net.

[44] SEMTECH. *LoRa® Products, Software, Cloud Services & Servers*. [Online; Stand 17. April 2022]. 2021. URL: https://www.semtech.com/lora/lora-products.

[45] SEMTECH. *MCU Requirements for LoRaWAN*. [Online; Stand 7. Marz 2022]. URL: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000HSRD/hyKkYJa7E.71emIBZ2LpgwFomMaTHRwCcb.dCogwdUg.

[46] *Sigfox Inc.* Abgerufen am 18.02.2022. URL: https://sigfox.de/.

[47] Mariusz Slabicki, Gopika Premsankar und Mario Di Francesco. „Adaptive configuration of LoRa networks for dense IoT deployments“. English. In: IEEE/IFIP Network Operations and Management Symposium; 2018, S. 1–9. ISBN: 9781538634165. DOI: 10.1109/NOMS.2018.8406255. URL: http://urn.fi/URN:NBN:fi:aalto-201812105973.

[48] Marco Stellin, Sérgio Sabino und António Grilo. „LoRaWAN networking in mobile scenarios using a WiFi mesh of UAV gateways“. In: *Electronics* 9.4 (2020), S. 630.

[49] The Things Network Global Team. *LoRa World Record Broken: 832km/517mi using 25mW*. [Online; Stand 28. Marz 2022]. URL: https://www.thethingsnetwork.org/article/lorawan-world-record-broken-twice-in-single-experiment-1.

[50] *Telensa Inc.* Abgerufen am 18.02.2022. URL: https://www.telensa.com/.

[51] *The things network*. Abgerufen am 18.02.2022. URL: https://www.thethingsnetwork.org/.

[52] Martin Bor Thiemo Voigt. *LoRaSim*. [Online; Stand 2. Februar 2022]. 2017. URL: https://www.lancaster.ac.uk/scc/sites/lora/lorasim.html.

[53] TinyGS. *Satellites*. [Online; Stand 28. Marz 2022]. URL: https://tinygs.com/satellites.

[54] Thanh-Hai To und Andrzej Duda. „Simulation of lora in ns-3: Improving lora performance with csma“. In: *2018 IEEE International Conference on Communications (ICC)*. IEEE. 2018, S. 1–7.

[55] Lorenzo Vangelista und Marco Centenaro. „Worldwide connectivity for the internet of things through LoRaWAN“. In: *Future Internet* 11.3 (2019), S. 57.

[56] Thiemo Voigt u. a. „Mitigating Inter-Network Interference in LoRa Networks“. English. In: *EWSN '17 Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*. ACM Press, Feb. 2017, S. 323–328. ISBN: 9780994988614.

[57] Thiemo Voigt u. a. „Mitigating inter-network interference in LoRa networks“. In: *arXiv preprint arXiv:1611.00688* (2016).

[58] *Waviot Inc.* Abgerufen am 18.02.2022. URL: https://waviot.com/.

[59] *Weightless SIG*. Abgerufen am 18.02.2022. URL: https://www.openweightless.org/.

[60] Wikipedia. *Short Range Device — Wikipedia, die freie Enzyklopädie*. [Online; Stand 11. Oktober 2021]. 2020. URL: https://de.wikipedia.org/w/index.php?title=Short_Range_Device&oldid=198670506.

[61] Wikipedia contributors. *Effective radiated power — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Effective_radiated_power&oldid=1085650938. [Online; Stand 2. Mai 2022]. 2022.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 24. 05 2022