Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Daniel Dahlke

Konzept und Validierung eines dezentralen LoRaWAN-Servers mit OPC-UA-Anbindung für die Gebäudeautomatisierung

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Computer Science and Engineering
Department Information and Electrical Engineering

Daniel Dahlke

Konzept und Validierung eines dezentralen LoRaWAN-Servers mit OPC-UA-Anbindung für die Gebäudeautomatisierung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Elektro- und Informationstechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Jochen Maaß
Zweitgutachter: M. Sc. Lucas-Christoph Ebel

Eingereicht am: 26. April 2023

Daniel Dahlke

Thema der Arbeit

Konzept und Validierung eines dezentralen LoRaWAN-Servers mit OPC-UA-Anbindung für die Gebäudeautomatisierung

Stichworte

LPWAN, LoRa, LoRaWAN, Gebäudeautomatisierung, Sensor, Gateway, Netzwerkserver, Applikationsserver, ChirpStack, Node-RED, MQTT, SPS, CODESYS, OPC-UA

Kurzzusammenfassung

Diese Bachelorarbeit thematisiert die Konzeptfindung und Implementierung eines lokalen und dezentralen LoRaWAN-Netzwerkservers mit einer OPC-UA-Anbindung. Dadurch wird die Bereitstellung der Messwerte von LoRaWAN-Sensoren an Variablen einer speicherprogrammierbaren Steuerung (SPS) realisiert. Es wird außerdem eine Bedienmöglichkeit für die dynamische Erstellung und Bearbeitung der Verknüpfungen zwischen den LoRaWAN-Messwerten und den Variablen der SPS erarbeitet. Die Implementierung des Konzeptes wird anhand eines Testaufbaus realisiert, mit welchem auch die Validierung der Funktion durchgeführt wird.

Daniel Dahlke

Title of Thesis

Concept and validation of a decentralised LoRaWAN server with OPC-UA connection for building automation

Keywords

LPWAN, LoRa, LoRaWAN, building automation, sensor, gateway, network server, application server, ChirpStack, Node-RED, MQTT, PLC, CODESYS, OPC-UA

Abstract

This bachelor thesis describes the conceptualisation and implementation of a local and decentralised LoRaWAN network server including an OPC-UA connection. This allows the provision of measured values from LoRaWAN sensors to variables of a programmable logic controller (PLC). In addition, an operating option for the dynamic creation and editing of the connections between the LoRaWAN measured values and the variables of the PLC is developed. The implementation of the concept is realised using a test setup, which is also used to validate the functionality.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ABP Activation-by-Personalization.

ADR adaptive Datenrate.

AES Advanced Encryption Standard.

API Application Programming Interface.

AppEUI Application Extended Unique Identifier.

AppKey Application Key.

AppSKey Application Session Key.

CSS Chirp Spread Spectrum.

DevAddr Device Address.

DevEUI Device Extended Unique Identifier.

DevNonce Device Nonce.

FB Funktionsbaustein.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IoT Internet of Things.

ISO/IEC International Organization for Standardization/International Electrotechnical Commission.

ITU International Telecommunication Union.

JSON JavaScript Object Notation.

LoRa Long Range.

LoRaWAN Long Range Wide Area Network.

LPWAN Low Power Wide Area Network.

MAC Media Access Control.

MIC Message Integrity Code.

MQTT Message Queuing Telemetry Transport.

NFC Near Field Communication.

NwkSKey Network Session Key.

OPC-UA Open Platform Communications-Unified Architecture.

OSI Open Systems Interconnection.

OTAA Over-The-Air-Activation.

PLC Programmable Logic Controller.

SF Spreading Factor.

SFD Start Frame Delimiter.

SPS Speicherprogrammierbare Steuerung.

TCP/IP Transmission Control Protocol/Internet Protocol.

UDP User Datagram Protocol.

URL Uniform Resource Locator.

1 Einleitung

Das *Internet of Things* (IoT) hat sich in den letzten Jahren sowohl im privaten, als auch im industriellen Bereich etabliert und gewinnt zunehmend an Bedeutung für die Realisierung einer nachhaltigen und effizienten Nutzung verschiedenster Ressourcen. Durch die Vernetzung von IoT-Geräten und -Systemen werden Daten gesammelt und ausgewertet, mit denen vielfältige Prozesse optimiert und Maßnahmen für einen effizienteren Betrieb der jeweiligen Anwendungen ergriffen werden können. Der Prognose in Abbildung 1.1 aus dem Jahr 2022 ist zu entnehmen, dass die Anzahl der IoT-Verbindungen voraussichtlich weiter ansteigt und sich im Zeitraum von 2019 bis 2030 von 9,8 auf 29,4 Milliarden Verbindungen verdreifachen könnte [57].

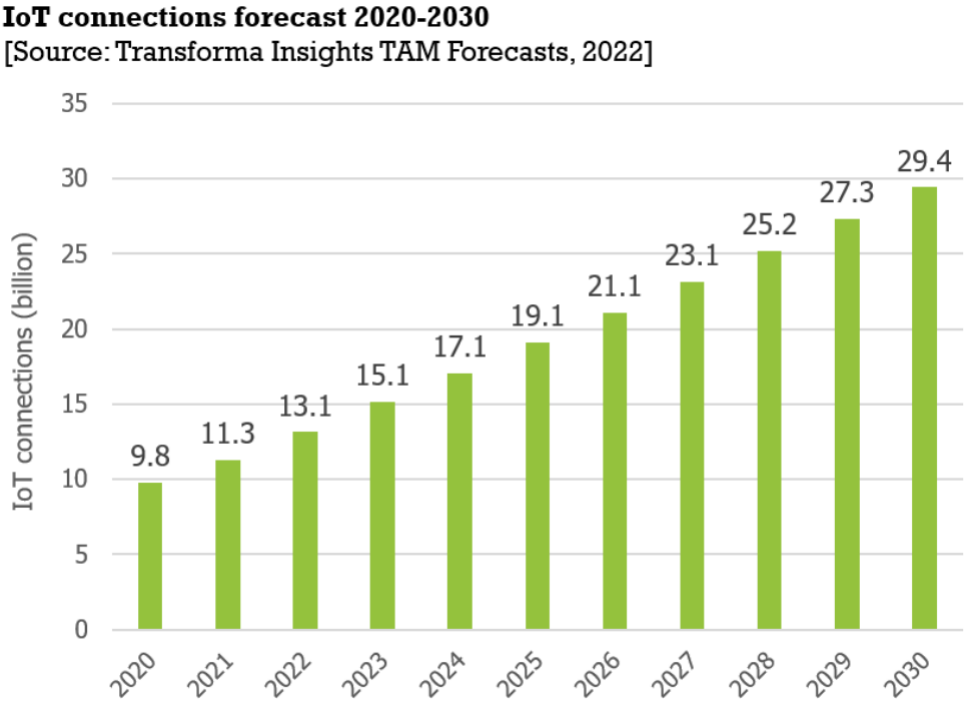

Abbildung 1.1: Entwicklung der Anzahl von IoT-Verbindungen bis zum Jahr 2022 und Prognosen bis 2030 (Quelle: Transforma Insights [57])

Eine aufstrebende und an Relevanz zunehmende Technologie für verschiedene Anwendungen aus dem IoT stellt dabei das *Low Power Wide Area Network* (LPWAN) dar, welche eine drahtlose Kommunikation für die Übertragung von Daten realisiert. Grundlegende Eigenschaften dieser Technologie sind der geringe Energieverbrauch der verwendeten Endgeräte, eine hohe Reichweite der Datenübertragungen und eine geringe Datenrate. Aufgrund dieser Merkmale eignet sich die LPWAN-Technologie besonders für die Übertragung von Sensordaten [42].

Zu den vielfältigen Einsatzgebieten von einem LPWAN im Rahmen des IoT zählen unter anderem das Transport- und Gesundsheitswesen, die Gebäudeautomatisierung und die Agrarwirtschaft [42]. Der in dieser Bachelorarbeit betrachtete Anwendungsfall bezieht sich auf die Gebäudeautomatisierung und wird in Abschnitt 1.1 näher beleuchtet.

Es gibt verschiedene Vertreter der LPWAN-Technologie, wobei NB-IoT, LoRaWAN (*Long Range Wide Area Network*) und Sigfox zu den Bekanntesten und Verbreitetsten zählen. In dieser Arbeit wird das LoRaWAN-Protokoll verwendet, welches auf der *Long Range* (LoRa)-Funktechnik der Firma *Semtech* basiert [42] (siehe Kapitel 2).

1.1 Problemstellung und Zielsetzung der Thesis

Durch die Verwendung der LoRa-Technologie sollen im Rahmen eines Gebäudeautomatisierungsprojektes Messwerte von LoRaWAN-Raumsensoren, die in verschiedenen Räumen installiert werden, als Referenzpunkte für eine Heizkreisregelung dienen. Die Regelung soll im weiteren Verlaufe des Projektes auf einer speicherprogrammierbaren Steuerung (SPS) implementiert werden, sodass die Messwerte dieser zunächst bereitgestellt werden müssen.

Diese Bachelorthesis hat das Ziel, ein Konzept zu erarbeiten, mit dem fortlaufend jeweils die aktuellen LoRaWAN-Messwerte an eine SPS per OPC-UA weitergeleitet und im Anschluss in Variablen geschrieben werden können. Konkret ist hierbei die Wahl für einen geeigneten LoRaWAN-Netzwerkserver zu treffen und eine Schnittstelle zwischen Netzwerkserver und SPS zu entwickeln. Außerdem ist eine Bedienmöglichkeit für die dynamische Erstellung und Bearbeitung der Verknüpfungen zwischen den LoRaWAN-Messwerten und den Variablen vorgesehen.

Des Weiteren ist eine Implementierung des Konzeptes zu realisieren, mit der die Funktionsweise validiert werden soll.

1.2 Aufbau der Thesis

Als Einstieg in die Bachelorthesis werden die technischen Grundlagen der behandelten Thematik erläutert. Daraufhin wird ein Konzept erarbeitet, mit welchem die in Abschnitt 1.1 beschriebene Problemstellung gelöst werden soll. Hierbei werden zunächst die für die aufzubauende Kommunikationsverbindung benötigten Komponenten bestimmt, wobei verschiedene Ansätze und Möglichkeiten für deren Realisierung erläutert werden. Daraus wird ein Gesamtkonzept erstellt, auf dessen Grundlage im Anschluss die Implementierung erfolgt. Abschließend wird die Funktionsweise der erstellten Lösung validiert und bewertet.

2 Technische Grundlagen

In diesem Kapitel werden die technischen Grundlagen der Bachelorarbeit dargelegt. Hierbei wird zunächst zwischen der LoRa-Funktechnik und dem LoRaWAN-Netzwerkprotokoll unterschieden, wobei deren grundlegenden Eigenschaften beleuchtet werden. Daraufhin erfolgt eine Betrachtung der Architektur eines LoRaWAN-Netzwerkes, in der auch die Aufgaben und Funktionen der einzelnen Komponenten beschrieben werden. Zum Abschluss des theoretischen Teiles werden die Grundlagen zu OPC-UA erläutert.

2.1 LoRa-Funktechnik

Dieser Abschnitt befasst sich mit der LoRa-Funktechnik, wobei zunächst die Modulationstechnik für die Signalübertragung erläutert wird. Im Anschluss werden die wesentlichen Parameter der Funktechnik beschrieben.

2.1.1 Was ist LoRa?

LoRa ist eine von der Firma *Semtech* entwickelte LPWAN-Technologie, die sich durch einen geringen Energieverbrauch und eine hohe Reichweite auszeichnet. Die Datenübertragung basiert auf einer hochfrequenten Modulationstechnik, mit der eine drahtlose Kommunikation der LoRaWAN-Netzwerkteilnehmer per Funk realisiert wird.

Als Modulation wird das sogenannte *Chirp Spread Spectrum* (CSS) verwendet, welches eine hohe Funkreichweite ermöglicht. Das *Spread Spectrum*, welches auch als Frequenzspreizung bezeichnet wird, beschreibt dabei ein Verfahren, bei dem die Bandbreite eines Signals gezielt vergrößert wird. Dadurch wird auch ein breiteres Frequenzspektrum bei der Übertragung nutzbar. Bei einem *Chirp* handelt es sich um ein Signal, bei dem sich die Frequenz mit der Zeit kontinuierlich verändert. Diese Signale haben eine konstante Amplitude und durchlaufen jeweils die gesamte verfügbare Bandbreite. Bei den Chirp-Impulsen

wird zwischen den *Up-Chirps*, bei denen die Frequenz zunimmt, und den *Down-Chirps*, bei denen die Frequenz abnimmt, differenziert [26].

Die Abbildung 2.1 visualisiert exemplarisch das Spektrogramm eines LoRa-Paketes, welches sich aus mehreren Chirp-Impulsen zusammensetzt. Zu Beginn werden hierbei zehn *Up-Chirps* für die Synchronisation gesendet. Daraufhin werden zwei *Down-Chirps* übertragen, die als *Start Frame Delimiter* (SFD) bezeichnet werden und den Start der Übertragung der Datensymbole signalisieren. Die verschiedenen Datensymbole werden anschließend durch modulierte *Chirps* dargestellt [1].

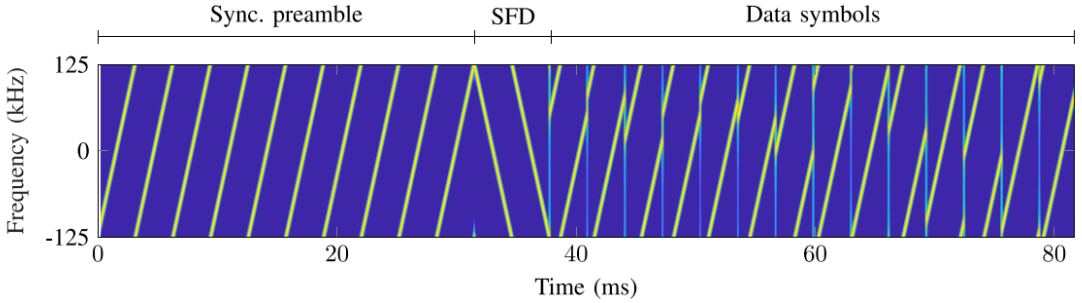

Abbildung 2.1: Spektrogramm einer LoRa-Nachricht mit der CSS-Modulation, welches die einzelnen Abschnitte eines LoRa-Paketes darstellt (Quelle: IEEE [1, S.1850])

LoRa lässt sich in das *Open Systems Interconnection* (OSI)-Referenzmodell einordnen. Das OSI-Modell basiert auf dem Standard der ISO/IEC [15] und definiert mehrere Schichten, mit denen verschiedene Netzwerkarchitekturen einheitlich gestaltet werden können (Abbildung 2.2). Die Kommunikation zwischen den einzelnen Systemen wird erreicht, indem für jede Schicht klare und eindeutige Aufgaben vorgegeben sind. Da ein LoRa-Modul die physikalische Grundlage von einem auf LoRaWAN basierenden Netzwerk bildet, wird durch die LoRa-Technologie die unterste Ebene des OSI-Referenzmodells realisiert. Diese Schicht wird Bitübertragungsschicht oder Physical Layer genannt [26].

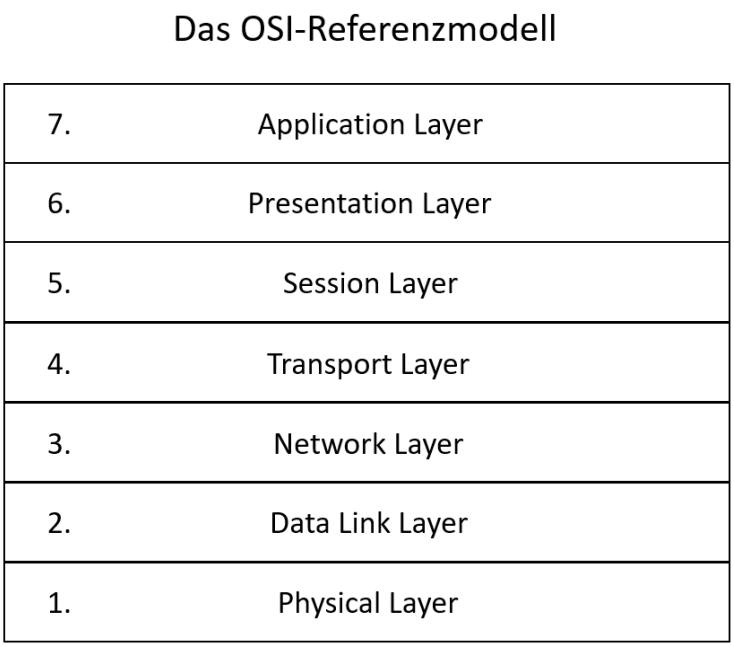

Abbildung 2.2: Darstellung der sieben Schichten des OSI-Referenzmodells (In Anlehnung an die International Telecommunication Union (ITU) [15])

2.1.2 Parameter der Funkübertragung

In diesem Abschnitt werden einige wichtige Parameter für die LoRa-Funkübertragung aufgeführt.

Der Spreading Factor

Der Spreading Factor (SF), auch Spreizfaktor genannt, bietet eine Skalierbarkeit der Datenrate und der Reichweite eines Signals. In der LoRa-Modulation werden Spreizfaktoren im Bereich von 7 bis 12 verwendet. Die Verwendung verschiedener Spreizfaktoren ermöglicht es, die Chirp-Rate und somit die Dauer eines einzelnen *Chirps* variieren zu können. Außerdem wird auch der Energieverbrauch der LoRaWAN-Endgeräte durch den Spreizfaktor beeinflusst.

Kleinere Spreizfaktoren werden für kürzere Reichweiten genutzt und verwenden eine höhere Datenrate. Außerdem ist die Sendezeit eines Signals in diesem Fall geringer als beim Einsatz eines größeren Spreizfaktors. Die kürzere Sendezeit hat zur Folge, dass die eingesetzten Batterien eine längere Lebensdauer aufweisen, da das Endgerät für einen vergleichsweise kleinen Zeitraum aktiv ist.

Höhere Spreizfaktoren hingegen sorgen für eine bessere Empfangsempfindlichkeit und eine höhere Robustheit der gesendeten Signale, wodurch größere Entfernungen bei der Übertragung ermöglicht werden. Hierfür wird die Datenrate gegenüber kleineren Spreizfaktoren verringert. Da sich bei einem größeren Spreizfaktor die Sendezeit eines Gerätes verlängert, wirkt sich dies negativ auf die Batterielebensdauer aus [21], [56].

Frequenzband

Ein wesentlicher Aspekt von LoRa ist die Verwendung des lizenzfreien Funkspektrums, sodass keine Lizenzkosten bei der Nutzung entstehen. Die zur Verfügung stehenden Frequenzbänder für LoRa-Übertragungen unterscheiden sich je nach Region. In Europa wird beispielsweise das EU863-870-MHz-Frequenzband verwendet, während in Nordamerika das US902-928-MHz-Frequenzband zum Einsatz kommt [24].

Bandbreite

Die LoRa-Funktechnik verwendet für die Datenübertragung Bandbreiten von 125 kHz, 250 kHz oder 500 kHz, welche sich je nach Region und dem jeweils genutzten Frequenzband unterscheiden [24].

Datenrate

Die Datenrate der Übertragungen bei der LoRa-Funktechnik liegt im Bereich von 0,3 kbit/s bis 50 kbit/s [23].

2.2 LoRaWAN V1.0

In diesem Abschnitt wird anfangs der Begriff LoRaWAN erläutert, woraufhin auf einige relevante Eigenschaften eines LoRaWAN-Netzwerkes eingegangen wird.

LoRaWAN steht in den beiden Hauptversionen 1.0 und 1.1 zur Verfügung, deren Spezifikation sich in einigen wesentlichen Aspekten voneinander unterscheidet. Die Version 1.1 beinhaltet gegenüber der Version 1.0 unter anderem zusätzliche Sitzungsschlüssel für die Sicherheit und einen Join Server für das Schlüsselmanagement als weitere Komponente in der LoRaWAN-Netzwerkarchitektur. Dies hat auch Auswirkungen auf den Ablauf des Netzwerkbeitrittes von Endgeräten. Außerdem ermöglicht die Version 1.1 das Roaming zwischen mehreren Netzwerken [29].

Im Rahmen dieser Arbeit wird jedoch ausschließlich die LoRaWAN-Version 1.0 betrachtet.

2.2.1 Was ist LoRaWAN?

LoRaWAN definiert die Netzwerkarchitektur und das Protokoll für die Kommunikation auf Basis der LoRa-Funktechnologie [21]. Das Kommunikationsprotokoll, sowie die Spezifikationen von LoRaWAN werden durch die LoRa-Alliance standardisiert [18].

Die LoRa-Alliance wurde im März 2015 gegründet und bildet eine offene Organisation, zu deren Mitgliedern unter anderem große Technologiekonzerne, Produkthersteller und Start-Ups zählen. Exemplarisch sind hier Unternehmen wie IBM, Cisco, Semtech, Bosch und Schneider zu nennen. Die LoRa-Alliance hat es sich zur Aufgabe gemacht, den Einsatz und Erfolg des LoRaWAN-Standards als eine der führenden LPWAN-Technologien voranzutreiben [22].

LoRaWAN bildet die *Media Access Control* (MAC)-Schicht in einem auf der LoRa-Technologie basierenden Netzwerk. Diese Schicht liefert einen Mechanismus der Medienzugriffskontrolle, welcher die Kommunikation zwischen den Endgeräten und den Gateways ermöglicht [14]. Nach Quelle [12] lässt sich das LoRaWAN-Protokoll im OSI-Referenzmodell dem Data-Link-Layer zuordnen, welches die zweite Schicht darstellt und das *Media Access Control* beinhaltet. Das OSI-Referenzmodell ist in Abbildung 2.2 zu sehen.

2.2.2 Die Geräteklassen von LoRaWAN-Endgeräten

Für die Endgeräte in einem LoRaWAN-Netzwerk sind die drei Geräteklassen A, B und C definiert. Die Geräteklassen beschreiben verschiedene Betriebsarten, welche je nach konkretem Anwendungsfall zum Einsatz kommen. Die einzelnen Geräteklassen unterscheiden sich dabei in der Art, wie die Endgeräte in dem LoRaWAN-Netzwerk kommunizieren. Die Wahl der Geräteklasse stellt einen Kompromiss zwischen der Downlink-Kommunikationslatenz des Netzwerkes und der Batterielebensdauer des Endgerätes dar [21], [49]. Ein Downlink bezeichnet hierbei eine Nachricht vom Netzwerkserver (Abschnitt 2.3) an das Endgerät, während ein Uplink die Kommunikation von einem Endgerät in Richtung des Netzwerkservers darstellt. Die Eigenschaften der einzelnen Geräteklassen werden im Folgenden beschrieben.

Die Geräteklasse A

Alle auf LoRaWAN-basierenden Endgeräte müssen die Betriebsart der Geräteklasse A unterstützen. Ein wesentliches Merkmal ist, dass die Kommunikation bei der Geräteklasse A nur durch das Endgerät eingeleitet werden kann.

Nachdem eine Uplink-Übertragung vom Endgerät durchgeführt wurde, öffnet das Endgerät mit zeitlichem Abstand die sogenannten Receive-Windows (Empfangsfenster) RX1 und RX2. In diesen beiden Fenstern kann das Gerät Downlink-Übertragungen empfangen. Nur wenn während des ersten Receive-Windows kein Downlink empfangen wird, wird das Receive-Window RX2 geöffnet. Im Anschluss geht das Endgerät in den Standby-Modus, bis ein weiterer Uplink durchgeführt werden soll. Dadurch ist die Geräteklasse A die energieeffizienteste und weist die längste Batterielaufzeit auf [46]. In Abbildung 2.3 wird die beschriebene Kommunikation für die Geräteklasse A visualisiert.

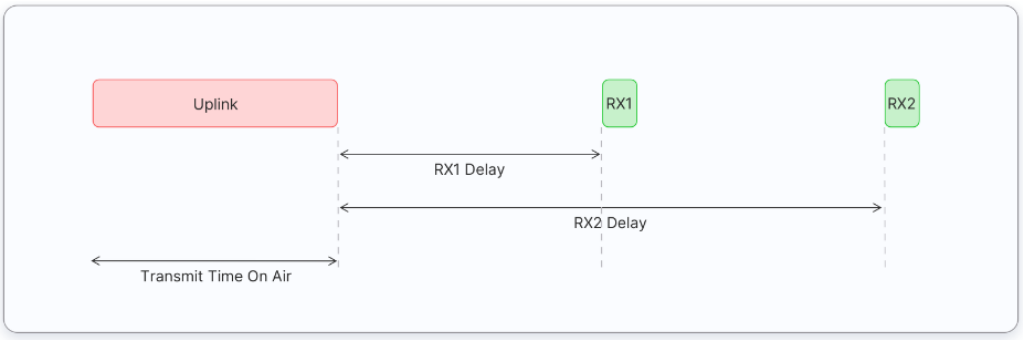

Abbildung 2.3: Kommunikation der Geräteklasse A, in der nach einem Uplink zwei feste Downlink-Fenster geöffnet werden (Quelle: The Things Network [53])

Die Geräteklasse B

Die Geräteklasse B kann als eine Erweiterung der Klasse A aufgefasst werden und ermöglicht zusätzliche, zeitlich fest definierte Downlink-Übertragungsfenster, in denen das Endgerät Daten entgegennehmen kann. Um diese Funktionalität zu ermöglichen, werden in einem LoRaWAN-Netzwerk regelmäßig sogenannte Beacons über die Gateways an die Endgeräte versendet. Anhand der Beacons synchronisiert sich das Endgerät mit dem Netzwerk und öffnet in periodischen Abständen weitere Receive-Windows, die als Ping-Slots bezeichnet werden [47] (siehe Abbildung 2.4).

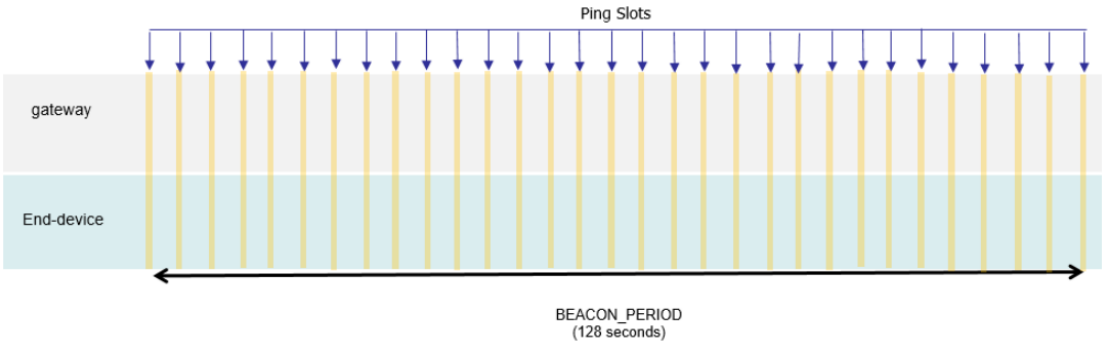

Abbildung 2.4: Darstellung der periodisch erzeugten Ping-Slots innerhalb einer Beacon-Periode, welche für eine Downlink-Nachricht genutzt werden können (Quelle: Semtech [47, S.4])

Die Abbildung 2.5 bietet eine Übersicht, wie die Kommunikation eines Gerätes mit der Klasse B aussehen kann. Dargestellt sind zum einen die Beacons und exemplarisch ein sich daraus ermittelter Ping-Slot. Außerdem wird auch die aus der Geräteklasse A bereits bekannte Funktionalität mit einer Uplink-Nachricht und den beiden darauffolgenden Receive-Windows visualisiert.

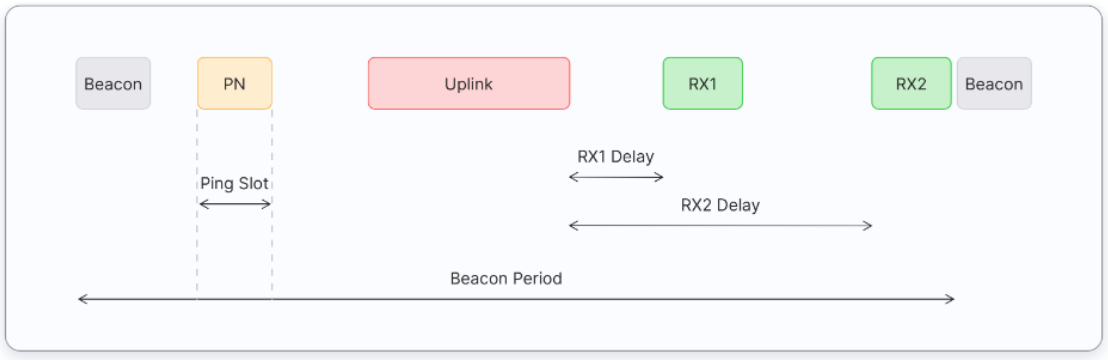

Abbildung 2.5: Kommunikation der Geräteklasse B, welche neben den zwei Receive-Windows nach einem Uplink weitere fest definierte Downlink-Fenster (Ping-Slots) öffnet (Quelle: The Things Network [53])

Die Geräteklasse C

Endgeräte der Klasse C haben die Eigenschaft, dass sie annähernd dauerhaft bereit sind, Downlink-Übertragungen entgegen zu nehmen. Dies wird ermöglicht, indem das Receive-Window RX2 so lange geöffnet bleibt, bis das Endgerät einen Uplink senden will. Nur während die Endgeräte eine Uplink-Nachricht versenden, kann kein Downlink empfangen

werden. Aufgrund dessen haben Klasse C-Endgeräte gegenüber den anderen Klassen die geringste Latenzzeit bei Downlink-Nachrichten [49]. Der beschriebene Sachverhalt ist in Abbildung 2.6 visualisiert.

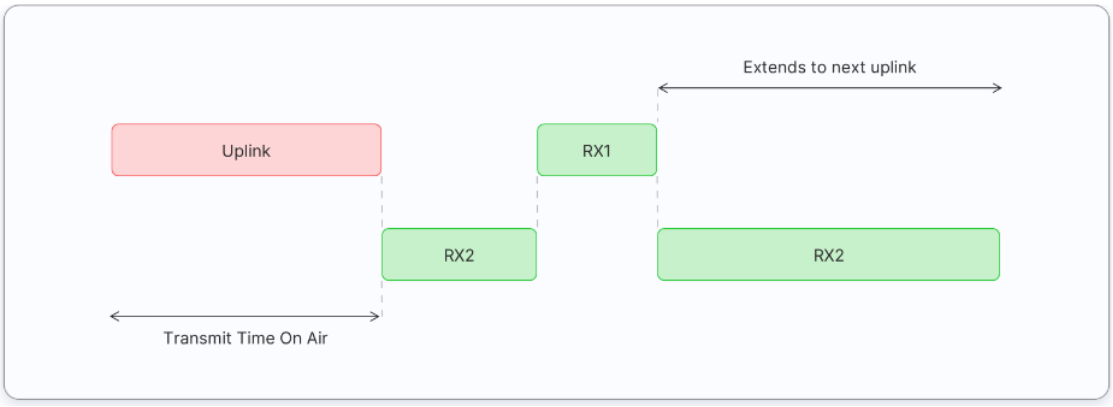

Abbildung 2.6: Kommunikation der Geräteklasse C, bei der die Receive-Windows, mit Ausnahme während eines Uplinks, dauerhaft für eine Downlink-Nachricht geöffnet sind (Quelle: The Things Network [53])

2.2.3 Die adaptive Datenrate

Die adaptive Datenrate (ADR) bietet in einem LoRaWAN-Netzwerk die Möglichkeit, die Datenrate und Sendezeit eines Endgerätes, sowie den damit einhergehenden Energieverbrauch, möglichst effizient zu gestalten. Um dies zu realisieren, wird anhand der adaptiven Datenrate Einfluss auf den Spreading Factor (siehe Unterabschnitt 2.1.2), die Bandbreite und die Übertragungsleistung genommen. Diese Funktion wird von Endgeräten eingesetzt, die statisch an einem festen Ort angebracht sind. Die Verwendung der adaptiven Datenrate wird von den Endgeräten initiiert und dem Netzwerkserver über das Setzen des ADR-Bits in einer Uplink-Nachricht mitgeteilt. Der Netzwerkserver zeichnet daraufhin beispielsweise 20 Uplink-Nachrichten des Endgerätes auf und führt eine Auswertung der Übertragungsparameter durch. Im Anschluss werden die optimierten Parameter für die folgenden Übertragungen an das Endgerät weitergeleitet [19].

2.2.4 Kennungen und Schlüssel in einem LoRaWAN-Netzwerk

In einem LoRaWAN-Netzwerk werden diverse Kennungen und Schlüssel für die Identifizierung von Netzwerkkomponenten und für die Sicherheit der Übertragungen verwendet. Diese werden im folgenden Abschnitt erläutert.

Die Device Extended Unique Identifier (DevEUI)

Die *DevEUI* eines Endgerätes ist eine weltweit eindeutige Kennung, die sich konkret dem Endgerät zuordnen lässt und dieses identifiziert. Die Kennung wird vom Hersteller während der Produktion vergeben und hat eine Länge von 64 Bits [23], [48].

Die Application Extended Unique Identifier (AppEUI)

Bei der *Application Extended Unique Identifier* handelt es sich um eine Kennung, mit der sich die Instanz, welche für die Verarbeitung des Join-Requests eines Endgerätes (Unterabschnitt 2.2.5) zuständig ist, eindeutig identifizieren lässt. Die AppEUI besteht ebenfalls aus 64 Bits [23]. Ab der LoRaWAN-Version 1.1 wird die AppEUI durch den Begriff JoinEUI ersetzt, der den Join-Server identifiziert [54].

Die Device Address (DevAddr)

Die *Device Address* bezeichnet eine 32-Bit lange Kennung, die vom Netzwerkserver jeweils für die einzelnen Endgeräte eindeutig festgelegt wird. Wird der Netzwerkbeitritt eines Endgerätes (siehe Unterabschnitt 2.2.5) über das Over-The-Air-Activation-Verfahren (OTAA) durchgeführt, erhält das Endgerät die *Device Address* im Rahmen der Join-Accept-Nachricht vom Netzwerkserver. Wird hingegen die Activation-by-Personalization-Methode (ABP) verwendet, muss die *Device Address* direkt in dem Endgerät hinterlegt werden [23].

Der Application Key (AppKey)

Der *Application Key* wird bei der Over-The-Air-Activation eines Endgerätes verwendet. Mit dem *Application Key* werden während dieser Form des Netzwerkbeitrittes die beiden eindeutigen Sitzungsschlüssel ermittelt, die als *Network Session Key* (NwkSKey) und *Application Session Key* (AppSKey) bezeichnet werden. Der *Application Key* stellt einen *Advanced Encryption Standard* (AES)-Schlüssel dar und hat eine Länge von 128 Bits [23].

Der Network Session Key (NwkSKey)

Der *Network Session Key* ist ein eindeutiger Sitzungsschlüssel, der jeweils nur einem konkreten Endgerät und dem Netzwerkserver zur Verfügung stehen darf. Diese beiden Komponenten eines LoRaWAN-Netzwerkes verwenden den *Network Session Key*, um den *Message Integrity Code* (MIC) der einzelnen Datenübertragungen während einer aktiven Sitzung zu berechnen und zu überprüfen. Anhand dessen soll die Integrität der Daten gewährleistet werden. Zusätzlich wird mit dem *Network Session Key* die Verschlüsselung und Entschlüsselung der Informationen, die nur die MAC-Ebene betreffen, realisiert [23].

Der Application Session Key (AppSKey)

Der *Application Session Key*, der einen weiteren eindeutigen Sitzungsschlüssel darstellt, ist nur dem jeweiligen Endgerät und dem dazugehörigen Applikationsserver (Abschnitt 2.3) bekannt. Mit dem *Application Session Key* wird die sogenannte *Frame Payload* einer Nachricht verschlüsselt und wieder entschlüsselt. Die *Frame Payload* beinhaltet dabei die Nutzdaten der spezifischen Applikation. Diese sind mit der Ende-zu-Ende-Verschlüsselung gesichert, die anhand des *Application Session Keys* zwischen dem Endgerät und dem Applikationsserver aufgebaut ist [23], [48].

Sowohl der *Network Session Key*, als auch der *Application Session Key* haben jeweils eine Länge von 128 Bits [49].

2.2.5 Netzwerkbeitritt von Endgeräten

Damit ein LoRaWAN-Endgerät in einem Netzwerk Daten senden und empfangen kann, muss das Endgerät zunächst dem Netzwerk beitreten. Dieser Vorgang wird auch als Aktivierung bezeichnet. Hierfür gibt es zwei verschiedene Möglichkeiten, die *Over-The-Air-Activation* (OTAA) und die *Activation-by-Personalization* (ABP). Diese beiden Varianten des Netzwerkbeitrittes werden im Folgenden beschrieben.

Zu beachten ist hierbei, dass sich die Erläuterungen auf die LoRaWAN-Version 1.0 beziehen. Der Ablauf eines Netzwerkbeitrittes von Endgeräten, welche die LoRaWAN-Version 1.1 verwenden, weist diverse Unterschiede auf und wird in dieser Arbeit nicht weiter ausgeführt.

Over-The-Air-Activation (OTAA)

Wenn ein Endgerät den Beitritt zu einem LoRaWAN-Netzwerk über die Over-The-Air-Activation-Methode unterstützt, wird das sogenannte Join-Verfahren zwischen dem Endgerät und dem Netzwerkserver durchgeführt. Diese Art des Netzwerkbeitrittes kann nur vom Endgerät selbst begonnen werden [54]. Hierbei gilt es zu beachten, dass ein Endgerät dem Netzwerk erneut beitreten muss, falls die Verbindung einer aktiven Sitzung unterbrochen wurde [23].

Damit das Join-Verfahren, welches in Abbildung 2.7 visualisiert ist, eingeleitet werden kann, muss das Endgerät zunächst über die DevEUI, die AppEUI und den AppKey verfügen. Außerdem muss der Netzwerkserver auch den jeweiligen AppKey und die DevEUI des Endgerätes besitzen. Der AppKey wird in einem LoRaWAN-Netzwerk niemals übertragen.

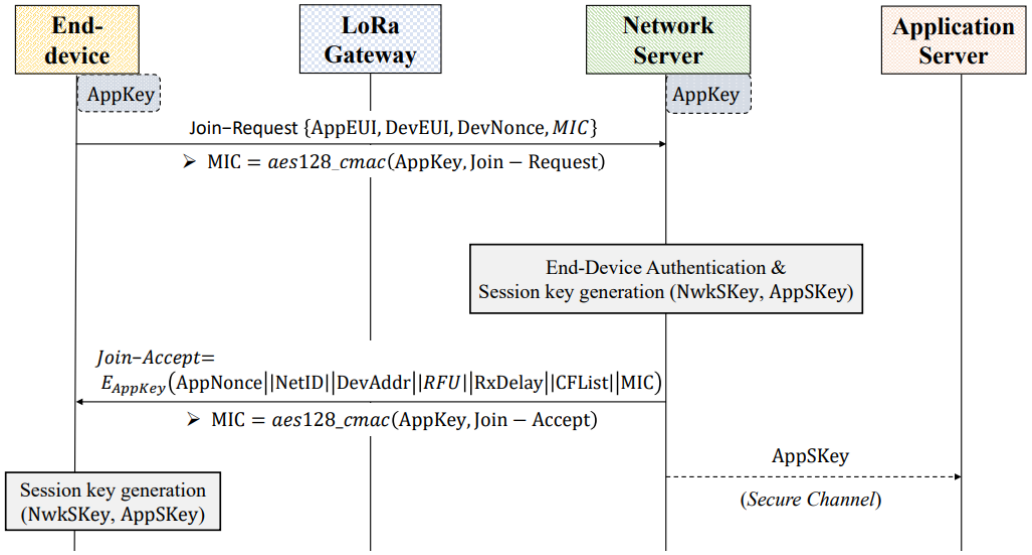

Abbildung 2.7: Darstellung des Join-Verfahrens bei der Over-The-Air-Activation eines Endgerätes, mit Join-Request- und Join-Accept-Nachricht, sowie Generierung der Sitzungsschlüssel (Quelle: MDPI Electronics Article [13, S.6])

Zu Beginn sendet das Endgerät einen Join-Request an den Netzwerkserver. Der Join-Request wird nicht verschlüsselt und beinhaltet immer die AppEUI, die DevEUI und eine *Device Nonce* (DevNonce). Bei der DevNonce handelt es sich um einen eindeutigen und einmaligen, aber zufällig festgelegten Wert des Endgerätes mit einer Länge von 16 Bits. Durch Verwendung des AppKeys wird anhand der Komponenten der Join-Request-

Nachricht ein MIC gebildet und dem Join-Request angehängt [54]. In Tabelle 2.1 sind die einzelnen Komponenten der Join-Request-Nachricht zur Verdeutlichung dargestellt.

Tabelle 2.1: Felder des Join-Requests mit Angabe der jeweiligen Größe in Bytes, in Anlehnung an [23] und erweitert durch [54]

Felder des Join-Requests	AppEUI	DevEUI	DevNonce	MIC
Größe in Bytes	8	8	2	4

Der Netzwerkserver empfängt den Join-Request und überprüft, ob das jeweilige Endgerät dazu berechtigt ist, dem Netzwerk beizutreten. Dabei kontrolliert der Netzwerkserver unter anderem, ob die gesendete DevNonce für das Endgerät einen bisher noch nicht verwendeten Wert darstellt. Wenn die DevNonce dem Netzwerkserver für das Endgerät bereits aus einem vorherigen Join-Request bekannt ist, wird der Netzwerkbeitritt untersagt. Hiermit wird verhindert, dass Angreifer die sogenannte Replay-Attacke bei einem Netzwerkbeitritt anwenden können. Ist dies jedoch nicht der Fall, so erzeugt der Netzwerkserver den AppSKey und den NwkSKey für die neu eingerichtete Sitzung.

Außerdem erstellt der Netzwerkserver die Join-Accept-Nachricht. Die einzelnen Bestandteile dieser Nachricht sind in Tabelle 2.2 aufgeführt. Auch hier wird ein MIC anhand des AppKeys und der Join-Accept-Nachricht gebildet und dem Join-Accept angehängt. Im Anschluss wird die komplette Join-Accept-Nachricht mit dem AppKey verschlüsselt und an das Endgerät gesendet.

Tabelle 2.2: Felder des Join-Accepts mit Angabe der jeweiligen Größe in Bytes, in Anlehnung an [23] und erweitert durch [54]

Felder des Join-Accepts	AppNonce	NetID	DevAddr	DLSettings
Größe in Bytes	3	3	4	1

Felder des Join-Accepts	RxDelay	CFList (Optional)	MIC
Größe in Bytes	1	16	4

Im nächsten Schritt wird der AppSKey vom Netzwerkserver an den Applikationsserver gegeben. Außerdem entschlüsselt das Endgerät die Join-Accept-Nachricht und berechnet die beiden Sitzungsschlüssel [54].

Activation-by-Personalization (ABP)

Bei der Verwendung der Activation-by-Personalization-Methode ist kein Join-Mechanismus implementiert, sodass die benötigten Kennungen und Sitzungsschlüssel direkt in dem jeweiligen Endgerät hinterlegt sein müssen. Hierzu gehören die DevAddr, der NwkSKey und der AppSKey. Auf der anderen Seite der Kommunikationsverbindung ist die DevAddr und der NwkSKey im Netzwerkserver gespeichert, während dem Applikationsserver der AppSKey bekannt sein muss [54].

Abbildung 2.8 verdeutlicht die notwendige Verteilung der Kennungen und Schlüssel an die einzelnen Komponenten eines LoRaWAN-Netzwerkes bei der ABP-Methode.

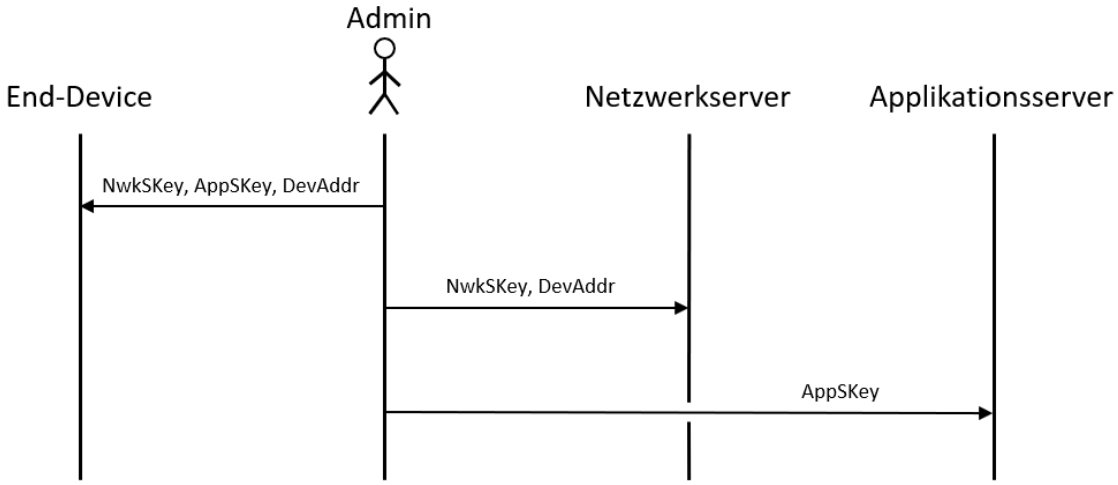

Abbildung 2.8: Verteilung der Kennungen und Schlüssel bei der Activation-by-Personalization-Methode durch einen Admin (In Anlehnung an The Things Network [54])

Ein Endgerät, welches das ABP-Verfahren verwendet, hat dauerhaft die gleichen Sitzungsschlüssel. Der AppKey, die DevEUI und die AppEUI werden für den Netzwerkbeitritt nicht benötigt [54].

2.3 Architektur eines LoRaWAN-Netzwerkes

Im folgenden Abschnitt wird zunächst die allgemeine LoRaWAN-Netzwerkarchitektur erläutert. Auf dieser Basis werden daraufhin die einzelnen Komponenten hinsichtlich deren Funktionen und Eigenschaften in einem LoRaWAN-Netzwerk beschrieben.

In der Abbildung 2.9 sind exemplarisch die wesentlichen Komponenten eines LoRaWAN-Netzwerkes und deren Zusammenhänge grafisch dargestellt.

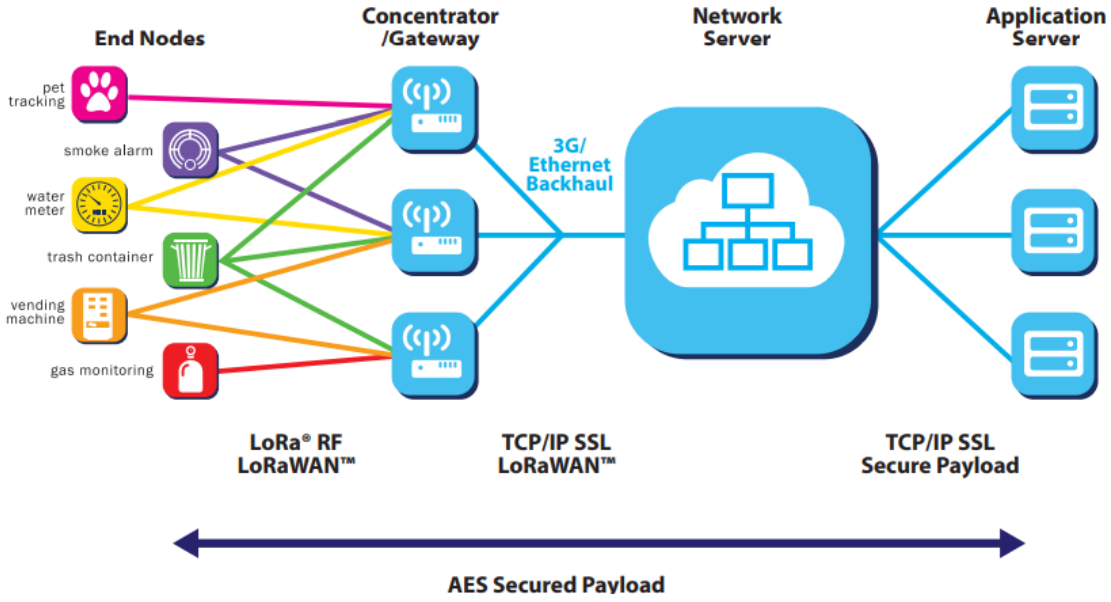

Abbildung 2.9: Architektur eines LoRaWAN-Netzwerkes, bestehend aus den Endgeräten, den Gateways, einem Netzwerkserver und den Applikationsservern (Quelle: LoRa Alliance [25, S.8])

Zu sehen ist, wie die einzelnen Endgeräte (*End Nodes*) jeweils ihre Daten mit einem oder mehreren Gateways austauschen. Jedoch erfolgt keine direkte Kommunikation zwischen den Endgeräten. Die Gateways eines Netzwerkes sind mit dem Netzwerkserver verbunden, welcher die Kommunikation mit den jeweiligen Applikationsservern ermöglicht. Die Abbildung 2.9 verdeutlicht auch, dass bei LoRaWAN die Netzwerkebene (der Netzwerkserver) von der Applikationsebene (dem Applikationsserver) getrennt ist. Zu berücksichtigen ist, dass die LoRa-Funktechnologie in einem LoRaWAN-Netzwerk nur zwischen den Endgeräten und den Gateways verwendet wird.

Das Endgerät

Ein LoRaWAN-Endgerät kann sowohl einen Sensor oder einen Aktor darstellen, als auch beide Funktionalitäten beinhalten. Das Endgerät tauscht Datenpakete mit den sich in der Reichweite befindlichen Gateways bidirektional aus, wobei die LoRa-Funktechnologie zum Einsatz kommt. Die LoRaWAN-Endgeräte werden meistens batteriebetrieben [55]. Eine weitere wesentliche Aufgabe eines Endgerätes ist die Verschlüsselung der gesendeten Datenpakete beziehungsweise die Entschlüsselung der empfangenen Nachrichten anhand der beiden Sitzungsschlüssel.

Das Gateway

Ein LoRaWAN-Gateway ist für die Weiterleitung der Datenpakete zwischen den Endgeräten und dem Netzwerkserver zuständig. Dabei ist das Gateway mit dem Netzwerkserver beispielsweise über Wi-Fi, Ethernet oder Mobilfunk verbunden. Diese Verbindung wird *Backhaul* genannt [49].

Der Netzwerkserver

Der Netzwerkserver ist für die Verwaltung des LoRaWAN-Netzwerkes zuständig. Außerdem hat der Netzwerkserver die Aufgabe, die Netzwerkparameter zu steuern. Hierfür wird unter anderem die adaptive Datenrate (siehe Unterabschnitt 2.2.3) verwendet. Da ein einzelnes Uplink-Datenpaket mehrmals von verschiedenen Gateways an den Netzwerkserver übermittelt werden kann, löscht der Netzwerkserver alle redundanten Nachrichten. Dieser Vorgang wird als Deduplizierung bezeichnet.

Der Netzwerkserver überprüft bei jedem Uplink sowohl die Authentizität eines Endgerätes, als auch die Integrität der gesendeten Nachricht. Außerdem leitet der Netzwerkserver anwendungsspezifische Nutzdaten an den vorgesehenen Applikationsserver. In Richtung der Endgeräte sendet der Netzwerkserver Datenpakete, welche die MAC-Schicht betreffen oder vom Applikationsserver initiiert wurden [49].

Der Applikationsserver

Der Applikationsserver empfängt und verarbeitet die von den Endgeräten gesendeten anwendungsspezifischen Nutzdaten. Außerdem erzeugt der Applikationsserver auch Datenpakete, welche die Anwendungsebene betreffen und an die Endgeräte gesendet werden sollen [49]. Des Weiteren bieten Applikationsserver diverse Optionen, um anhand von Integrationen eine nutzerspezifische Weiterverarbeitung der Daten zu ermöglichen.

2.4 OPC-UA

In diesem Abschnitt wird die *Open Platform Communications-Unified Architecture* (OPC-UA) grundlegend eingeführt.

OPC-UA bezeichnet einen offenen Standard für den Datenaustausch in der Industrie und dem Internet of Things (IoT). Ein wesentliches Merkmal von OPC-UA ist dabei die Realisierung eines plattform- und herstellerunabhängigen Kommunikationsstandards. Somit soll OPC-UA die Interoperabilität zwischen verschiedenen Systemen ermöglichen und verbessern. OPC-UA gehört dem IEC-Standard an (IEC 62541) [39], [40].

Die Kommunikation per OPC-UA erfolgt nach dem verbindungsorientierten Client-Server-Modell. Ein OPC-UA-Server ist dabei eine Software zur Bereitstellung von Diensten und Services für den Datenaustausch. Ein OPC-UA-Client ist das Gegenstück dieser Kommunikationsverbindung, der auf die vom OPC-UA-Server zur Verfügung gestellten Services zugreifen kann. Zu Beginn des Kommunikationsaufbaus fragt der OPC-UA-Client die Nutzung der vom Server angebotenen Dienste an. Die Identifikation des Servers findet unter anderem anhand seiner Endpoint-URL statt. Der Client authentifiziert sich gegenüber dem Server, wobei der Server daraufhin eine Rückmeldung mit wesentlichen Informationen für die erstellte Kommunikationsverbindung an den Client sendet.

Eine aktive Verbindung zwischen einem OPC-UA-Server und -Client wird als Sitzung bezeichnet. Ein OPC-UA-Client kann auf die zur Verfügung gestellten Daten des Servers je nach Anwendung lesend oder schreibend zugreifen.

Alle Daten, die ein OPC-UA-Server zur Verfügung stellt, sind in einem hierarchisch aufgebauten Adressraum strukturiert. Die einzelnen Komponenten innerhalb dieser Struktur werden als Knoten (Nodes) bezeichnet und sind durch Referenzen miteinander verbunden. Jeder Knoten in einem Adressraum besitzt diverse Attribute, welche die Eigenschaften des Knotens beschreiben. Zwei wesentliche Parameter sind hierbei das NodeId- und das NodeClass-Attribut. Das NodeId-Attribut wird für die eindeutige Identifizierung des Knotens innerhalb des OPC-UA-Servers genutzt. Anhand des NodeClass-Attributes wird erkennbar, ob der Knoten beispielsweise eine Variable, ein Objekt oder eine Methode darstellt.

3 Konzept

In diesem Kapitel wird in mehreren Schritten ein Konzept für die in Abschnitt 1.1 erläuterte Problemstellung der Thesis aufgestellt. Das Konzept soll als Grundlage für die Implementierung in Kapitel 4 dienen.

Hierbei wird zunächst das Zielsystem gewählt, auf welchem die für diese Arbeit benötigten Softwareanwendungen eingerichtet werden sollen. Außerdem werden zwei wesentliche Möglichkeiten für die Nutzung eines LoRaWAN-Netzwerkservers beschrieben, wobei auch die zur Verfügung stehenden Ausführungen verschiedener Anbieter vorgestellt werden. Im Anschluss werden die genauen Anforderungen an den Netzwerkserver im Rahmen des Projektes definiert. Auf dieser Basis erfolgt dann die Entscheidung zur Nutzung eines geeigneten Netzwerkservers. Des Weiteren werden zwei Ansätze für eine Schnittstelle erarbeitet, mit der die LoRaWAN-Messdaten vom Netzwerkserver aus über OPC-UA an eine SPS weitergeleitet werden können. Hierbei soll es einem Benutzer ermöglicht werden, die Anbindungen der Messdaten dynamisch bearbeiten und erweitern zu können. Im Anschluss wird sich für einen der Ansätze entschieden.

Abschließend wird aus den einzelnen Entscheidungen ein Gesamtkonzept erstellt und zur Übersicht bildlich dargestellt.

3.1 Auswahl eines Zielsystems

Als Zielsystem für die zu installierenden Softwareanwendungen im Laufe dieser Arbeit ist eine zentrale Rechnereinheit vorgesehen. Im Rahmen des Projektes wird sich für ein Linux-Betriebssystem entschieden, sodass sich hier die Anforderung der Kompatibilität mit Linux ergibt. Als konkrete Linux-Distribution soll Debian verwendet werden. Wesentliche Eigenschaften von Debian sind eine hohe Stabilität und Sicherheit, sowie die Verfügbarkeit von regelmäßigen Aktualisierungen [11]. Weitere Vorzüge für Debian im

Rahmen des Projektes sind der zuverlässige Paketmanager und die bereits vorhandene Erfahrung im Umgang mit diesem Betriebssystem.

Die Rechnereinheit soll im weiteren Verlaufe des Projektes in einem Schaltschrank montiert werden, sodass für diesen Anwendungsfall ein PC mit vorhandener Hutschienenbefestigung verwendet werden soll. Es wird entschieden, dass der Industrie-PC *Edge EN01J4* der Firma *Shuttle* als zentrale Rechnereinheit zum Einsatz kommt (siehe Unterabschnitt 4.1.1).

3.2 Auswahl eines LoRaWAN-Netzwerkservers

In diesem Abschnitt werden zwei verschiedene Varianten zur Nutzung eines LoRaWAN-Netzwerkservers vorgestellt. Durch das Identifizieren der konkreten Anforderungen an den Netzwerkserver wird im Anschluss eine Entscheidung getroffen, in welcher Form dieser realisiert werden soll.

3.2.1 Nutzung eines LoRaWAN-Gateways mit integriertem Netzwerkserver

Mehrere Hersteller von LoRaWAN-Gateways bieten einen im Gateway direkt integrierten, voll funktionsfähigen lokalen Netzwerkserver in Kombination mit einer LoRaWAN-Applikation an. Exemplarisch werden hier das *RAK7268 WisGate-Edge-Lite-2*-Gateway [43] und das *Milesight UG65*-Gateway [31] betrachtet. Das Bearbeiten der zur Verfügung stehenden LoRaWAN-Netzwerkeinstellungen, sowie das Management der Endgeräte erfolgt über die zentrale Web-Anwendung des Gateways. Außerdem sind je nach Gateway in der Applikation verschiedene Schnittstellen wie beispielsweise MQTT, HTTP und HTTPS integriert, mit denen die gesammelten Daten an weitere Anwendungen übertragen werden können.

3.2.2 Nutzung eines separaten LoRaWAN-Netzwerkservers

Eine weitere Möglichkeit zum Betreiben eines LoRaWAN-Netzwerkes ist die Nutzung eines eigenständigen Netzwerkservers. Dieser kann abhängig vom vorliegendem Anwendungsfall auf dem Server eines Anbieters oder lokal auf einem eigenen Server imple-

mentiert sein. Da es auf dem Markt mehrere verschiedene Anbieter von LoRaWAN-Netzwerkservern gibt, werden im Rahmen einer Recherche drei Anbieter ausgewählt. Berücksichtigt wurden bei der Recherche unter anderem:

- Bekanntheit und Etablierung des Anbieters auf dem Markt
- Vorhandene Dokumentation zur Inbetriebnahme und Nutzung des LoRaWAN-Netzwekservers
- Abdeckung möglichst verschiedener Bedürfnisse und Anwendungsfälle

Auf Grundlage dieser Aspekete werden The Things Stack, Loriot und ChirpStack als Anbieter für einen LoRaWAN-Netzwerkserver ausgewählt. Im Folgenden werden die Produkte der Anbieter vorgestellt, wobei jeweils die verschiedenen Eigenschaften und Besonderheiten erläutert werden.

The Things Stack

The Things Stack ist ein von dem Unternehmen *The Things Industries* entwickelter LoRaWAN-Netzwerkserver. Dabei wird The Things Stack in verschiedenen Ausführungen bereitgestellt, mit denen unterschiedliche Anwendungsfälle realisiert werden können. Bei der Nutzung von The Things Stack wird grundlegend zwischen einer Cloud-Anwendung und einer privaten Anwendung unterschieden. Die Dienste bei einer Cloud-Anwendung werden direkt auf den Servern von *The Things Industries* gehostet, während die private Anwendung auf einem eigenen Server betrieben wird [52].

Zu den Cloud-Anwendungen von The Things Stack gehört die Community-Edition, welche kostenlos genutzt werden kann und auf einem gemeinsamen, weltweiten Community-Gedanken basiert. Hierzu gehört unter anderem die Möglichkeit, eigene Gateways mit der Community zu teilen oder bereits vorhandene Gateways in der Umgebung zu nutzen. Die Community-Edition besitzt jedoch keine garantierte Verfügbarkeit und wird nicht für kommerzielle Zwecke empfohlen. Für einen solchen Anwendungsfall wird die The Things Stack Cloud angeboten, welche eine kostenpflichtige Alternative mit einer Verfügbarkeit von mehr als $99,9\,\%$ bereitstellt. Die Kosten für eine Nutzung von beispielhaft 1000 Endgeräten beläuft sich auf $190\,€$ pro Monat [51].

Zur Installation eines privaten Netzwerkservers stehen die Varianten The Things Stack Enterprise oder The Things Stack Open-Source zur Verfügung, die beide über Docker

installiert werden können. Bei der Nutzung dieser Ausführungen wird keine Internetverbindung benötigt. Während die Open-Source-Variante kostenlos ist, muss für The Things Stack Enterprise ein kostenpflichtiger Lizenzschlüssel erworben werden. Laut der offiziellen Webseite [51] fallen für die Enterprise-Version bei einer Nutzung von 1000 Endgeräten monatliche Kosten in Höhe von 500 € an. Jedoch sind für die Enterprise-Version keine weiteren Preise für eine abweichende Anzahl an Endgeräten einsehbar. The Things Stack Enterprise beinhaltet im Gegensatz zur Open-Source-Version unter anderem einen professionellen Support und ermöglicht das Multi-Tenancy-Prinzip. Durch das Multi-Tenancy lassen sich mehrere Anwendungen und Bereiche innerhalb eines Netzwerkservers erstellen, die getrennt voneinander betrieben werden können.

Unabhängig von der verwendeten The Things Stack-Variante steht eine Web-Anwendung zur Verfügung, in der die eigenen Geräte registriert und verwaltet werden. In der Web-Anwendung können verschiedene Integrationen eingerichtet werden, mit denen die Nutzdaten der Endgeräte je nach Anwendungsfall weitergeleitet werden können. Dazu gehören neben MQTT und HTTP auch diverse Datenbanken, Cloudanbieter, Visualisierungsprogramme und IoT-Plattformen. Außerdem stehen in The Things Stack *Application Programming Interfaces* (APIs) zur Verfügung, mit denen Erweiterungen und Schnittstellen implementiert werden können. Für die Web-Anwendung lassen sich verschiedene Benutzerprofile einrichten, mit denen unterschiedliche Zugriffsrechte und Beschränkungen in der Bedienung festgelegt werden können. Eine allgemeine Dokumentation zu The Things Stack steht unter Quelle [52] zur Verfügung.

ChirpStack

Bei ChirpStack handelt es sich um einen Open-Source LoRaWAN-Netzwerkserver. Mit einem ChirpStack-Netzwerkserver lassen sich sowohl private, als auch öffentliche LoRaWAN-Netzwerke realisieren. Eine kostenlose Installation kann entweder direkt über Docker oder durch die Verwendung einer Anleitung, die für Debian- und Ubuntu-Systeme vorgesehen ist, durchgeführt werden. ChirpStack kann somit lokal und ohne eine notwendige Internetverbindung auf einem eigenen Server genutzt werden. Die Installation enthält unter anderem Konfigurationsdateien, in denen verschiedene Netzwerkparameter vom Benutzer bei Bedarf verändert werden können. Außerdem steht eine Dokumentation für die Konfiguration und Nutzung von ChirpStack unter der Quelle [4] zur Verfügung. Für die Anzahl der verwendeten Endgeräte und Gateways ist in ChirpStack theoretisch keine Obergrenze vorgegeben. Jedoch ist dies unter anderem von der Leistungsfähigkeit des Zielsystems abhängig, auf dem ChirpStack installiert wird. Die grundlegende Archi-

tektur von ChirpStack, sowie die Kommunikationsverbindungen zwischen den einzelnen Komponenten ist im Anhang in Abbildung A.1 visualisiert.

Als zentrale Komponenten sind dort die ChirpStack Gateway Bridge, ein MQTT-Broker und ChirpStack selbst dargestellt. Diese können in einer Cloud, auf einem Rechner oder auf einer virtuellen Maschine (VM) betrieben werden. Ab der ChirpStack-Version V4 bilden der Netzwerkserver und der Applikationsserver eine gemeinsame Komponente, sodass diese in der Abbildung A.1 nicht mehr einzeln aufgeführt sind. LoRaWAN-Gateways können ihre Datenpakete über das *User Datagram Protocol* (UDP), über *Websockets* oder über MQTT mit ChirpStack austauschen, wobei je nach Verwendung verschiedene Forwarder auf dem jeweiligen Gateway eingerichtet sein müssen (siehe Abbildung A.1).

ChirpStack stellt eine Web-Anwendung bereit, in der die Registrierung und Verwaltung der Endgeräte und Gateways des LoRaWAN-Netzwerkes ermöglicht wird. In der Web-Anwendung können ebenfalls verschiedene Integrationen wie MQTT und HTTP, aber auch diverse Datenbanken, Cloudanbieter, Visualisierungsprogramme und IoT-Plattformen eingerichtet werden. Des Weiteren ermöglicht auch ChirpStack die Nutzung von APIs, mit denen sich der Netzwerkserver erweitern oder in andere Anwendungen integrieren lässt. In ChirpStack können ebenfalls mehrere Benutzerprofile erstellt werden, um den Zugriff auf bestimmte Inhalte in der Web-Anwendung einzuschränken. Außerdem unterstützt ChirpStack das Multi-Tenancy-Prinzip [4].

Loriot

Loriot ist ein von dem gleichnamigen Unternehmen implementierter LoRaWAN-Netzwerkserver, der sowohl privat als auch öffentlich genutzt werden kann. Die folgenden Informationen sind der offiziellen Produkt-Webseite von *Loriot* [27] und den darauf jeweils verlinkten Unterlagen entnommen. Loriot unterstützt neben LoRaWAN auch *mioty*, welches eine weitere LPWAN-Technologie darstellt. Der Loriot-Netzwerkserver steht in mehreren Ausführungen zur Verfügung, die sich an verschiedenen Zielgruppen und Anwendungsfällen orientieren. Unabhängig von der gewählten Variante bietet auch der Loriot-Netzwerkserver eine Weboberfläche für das Gerätemanagement und das Einrichten von Integrationen. Die weiteren Funktionen unterscheiden sich je nach verwendeter Ausführung. Eine Dokumentation zur Nutzung des Loriot-Netzwerkservers und den verschiedenen Varianten ist in Quelle [27] zugänglich.

Zu Testzwecken und Evaluierungen kleinerer LoRaWAN-Netzwerke ist der kostenlose Community Public Network Server mit den grundlegenden Funktionen vorgesehen. In

dieser Anwendung kann der Benutzer maximal 30 Endgeräte und theoretisch eine unbegrenzte Anzahl an Gateways auf einem öffentlichen Netzwerkserver registrieren.

Als Erweiterung gilt der kostenpflichtige Professional Public Server. Dieser öffentliche Netzwerkserver wird auf den Servern von Loriot betrieben und ist in vier verschiedenen Ausführungen verfügbar, welche sich in der maximalen Anzahl der Endgeräte unterscheiden. Die monatlichen Kosten liegen im Bereich von 100 € für 275 Endgeräte bis 500 € für 2750 Endgeräte [28]. Der Professional Public Server besitzt laut Anbieter eine Verfügbarkeit von 99,9 % und stellt zusätzlich Multi-Tenancy und APIs zur Verfügung. Außerdem können auch hier mehrere Benutzerprofile innerhalb einer Anwendung eingerichtet werden.

Eine weitere Möglichkeit ist die kostenpflichtige Nutzung des Loriot Private Servers, der auf einem lokalen Rechner oder in einer privaten Cloud auf den Servern von Loriot betrieben werden kann. Hierbei werden die gleichen Funktionen wie bei dem Professional Public Server angeboten. Es sind drei verschiedene Modelle vorhanden, die sich in der Art des angebotenen Supports und der Redundanz bei der Datenverarbeitung und speicherung unterscheiden. Angaben zu den Kosten sind an dieser Stelle nur auf Anfrage erhältlich. Anzumerken ist bei dieser Variante, dass die primäre Zielgruppe vor allem Telekommunikationsanbieter, städtische Einrichtungen und Unternehmen aus dem IoT-Sektor darstellen.

3.2.3 Wahl des LoRaWAN-Netzwerkservers

Im folgenden Abschnitt wird entschieden, welcher LoRaWAN-Netzwerkserver im Rahmen dieser Arbeit genutzt werden soll. Die zu berücksichtigenden Anforderungen an den Netzwerkserver in diesem Projekt lauten hierbei:

- Die Nutzung des Netzwerkservers muss zwingend ohne Internetverbindung möglich sein
- Der Anbieter darf keinen Zugriff auf den Netzwerkserver und auf die dazugehörigen Daten haben
- Bereitstellung einer Oberfläche, welche für das Gerätemanagement und auch als Diagnosetool für die Aktivitäten im Netzwerk verwendet werden kann

- Verfügbarkeit möglichst vielfältiger Schnittstellen und Integrationen zur Weiterleitung der Nutzdaten
- Möglichkeit zum Erstellen verschiedener Benutzerprofile, um den Zugriff auf bestimmte Funktionalitäten für weitere Nutzer beschränken zu können

Anhand der ersten beiden Aspekte ergibt sich zwangsläufig, dass der LoRaWAN-Netzwerkserver im Rahmen des Projektes in einer lokalen Ausführung gewählt werden muss. Aufgrund dessen wird die Möglichkeit, den Netzwerkserver auf einem externen Server eines Anbieters zu nutzen, in dieser Arbeit nicht weiter verfolgt.

Wie zuvor in Unterabschnitt 3.2.1 beschrieben, kann ein lokaler Netzwerkserver als bereits integrierte Anwendung auf einem LoRaWAN-Gateway, der diese Funktionalität unterstützt, verwendet werden. Des Weiteren stellen einige Anbieter von LoRaWAN-Netzwerkservern eine lokale Variante bereit, die auf einer eigenen Rechnereinheit betrieben werden kann (siehe Unterabschnitt 3.2.2). An dieser Stelle wird sich gegen den auf einem Gateway integrierten Netzwerkserver entschieden. Ein wesentlicher Nachteil hierbei ist die geringe Flexibilität, da das gesamte LoRaWAN-Netzwerk in diesem Fall von dem Gateway abhängig ist. Ein Wechsel des Gateways im Laufe des Projektes wäre nicht ohne Weiteres möglich, während dies bei einem separaten Netzwerkserver jederzeit durchgeführt werden kann. Außerdem wurde für die in Unterabschnitt 3.2.1 genannten Gateways mit integriertem Netzwerkserver von namhaften Herstellern festgestellt, dass deren Bedienoberfläche im Gegensatz zu den separaten Netzwerkservern nur grundlegende Funktionen bereitstellt.

Somit stehen folgende lokal verwendbare Netzwerkserver zur Auswahl:

- The Things Stack Open Source
- The Things Stack Enterprise
- ChirpStack
- Loriot Private Server

Im Rahmen einer Projektbesprechung wird entschieden, dass ChirpStack als lokaler Netzwerkserver genutzt werden soll. Ausschlaggebend ist hierbei, dass ChirpStack alle zuvor genannten Anforderungen an einen Netzwerkserver erfüllt und zusätzlich als kostenlose Open-Source-Anwendung keine Produkt-Lizenz für die Verwendung benötigt.

Außerdem wird als vorteilhaft angesehen, dass ChirpStack im Gegensatz zur ebenfalls kostenlosen The Things Stack Open-Source-Variante das Multi-Tenancy bereitstellt. Somit können im weiteren Verlaufe des Projektes bei Bedarf auf dem ChirpStack-Netzwerkserver einzelne unabhängig voneinander laufende Bereiche implementiert werden. Dies ermöglicht eine bessere Struktur des gesamten LoRaWAN-Netzwerkes und bietet eine weitere Option, um die Zugriffsrechte verschiedener Benutzer zu managen. Des Weiteren wird sich gegen die The Things Stack Enterprise-Variante entschieden, da während der Recherche im Vergleich zu ChirpStack bis auf den professionellen technischen Support keine weiteren wesentlichen Vorteile erkennbar waren. Als nachteilig werden hier jedoch die monatlich anfallenden Kosten für die Nutzung der Enterprise-Variante angesehen. Die Entscheidung gegen den Loriot Private Server wird damit begründet, dass dieser Netzwerksever grundlegend für eine andere Zielgruppe ausgelegt ist und die Konditionen für die Nutzung nur auf Anfrage einsehbar sind.

Der ChirpStack-Netzwerkserver wird in der Implementierungsphase (siehe Unterabschnitt 4.3.2) auf dem Industrie-PC installiert.

3.3 Planung der Schnittstelle für die OPC-UA-Anbindung

Als Nächstes werden zwei Möglichkeiten erarbeitet, mit denen die geforderte OPC-UA-Anbindung der LoRaWAN-Messdaten an eine SPS realisiert werden kann. Die jeweils aktuellen Messwerte der LoRaWAN-Sensoren sollen dabei fortlaufend in die dafür vorgesehenen Variablen in der SPS geschrieben werden. In dieser Phase der Konzeptfindung muss auch berücksichtigt werden, dass die bestehenden Verknüpfungen zwischen den Messwerten eines beliebigen Sensors des LoRaWAN-Netzwerkes mit den in der SPS angelegten Variablen durch einen Benutzer dynamisch veränderbar sein sollen. Zusätzlich soll es für den Benutzer möglich sein, neue Verknüpfungen erstellen zu können.

In dem Projekt wird CODESYS bereits als Programmierumgebung für die SPS verwendet, sodass auch in dieser Arbeit im Rahmen der Implementierungsphase CODESYS zum Einsatz kommt. Dadurch soll eine spätere Integration der erstellten Lösung möglichst einfach durchführbar sein. Zusätzlich wird festgelegt, dass die SPS als OPC-UA-Server genutzt werden soll. Daraus ergibt sich, dass die zu implementierende Schnittstelle einen OPC-UA-Client für die Übermittlung der Messdaten an die SPS beinhalten muss.

Des Weiteren wird entschieden, dass MQTT (*Message Queuing Telemetry Transport*) für die Verknüpfung zwischen ChirpStack und der noch zu bestimmenden Anwendung zur Erstellung der OPC-UA-Anbindung verwendet werden soll. In ChirpStack stehen hierfür alle Ereignisse im LoRaWAN-Netzwerk über eine MQTT-Integration bereit.

MQTT ist ein Nachrichtenprotokoll, welches auf dem Publish/Subscribe-Modell basiert. Mit MQTT können verschiedene Geräte und Anwendungen miteinander kommunizieren, indem sie Nachrichten über sogenannte Topics senden und empfangen. Bei MQTT werden Nachrichten über einen zentralen Broker nur an die jeweiligen Teilnehmer weitergeleitet, die das konkrete Topic der Nachricht abonniert haben. Für den Transport der Nachrichten wird das TCP/IP-Protokoll verwendet [33].

Das allgemeine MQTT-Topic aller Ereignisse im LoRaWAN-Netzwerk lautet:

application/APPLICATION_ID/device/DEV_EUI/event/EVENT

Die in Großbuchstaben geschriebenen Wörter sind Platzhalter und müssen je nach Anwendungsfall geändert werden. Durch das Festlegen der Application-ID, der DevEUI und des Events können somit die Uplink-Nachrichten eines konkreten Sensors herausgefiltert werden [5].

3.3.1 Möglichkeit der Realisierung einer Schnittstelle über Node-RED

Eine Möglichkeit für das Erstellen der Schnittstelle bietet die Verwendung von Node-RED.

Node-RED ist eine visuelle und ereignisorientierte Low-Code-Programmierumgebung, in welcher verschiedene Anwendungen durch das Verbinden von mehreren Nodes zu einem sogenannten Flow realisiert werden können. Die einzelnen Nodes führen jeweils bestimmte Funktionen aus und werden in Form von Blöcken dargestellt. Die benötigten Nodes werden über die Installation von Packages in die Programmierumgebung integriert. Außerdem lässt sich in Node-RED auch eigener Programmcode in der Programmiersprache JavaScript erstellen [36].

In Node-RED steht das *mqtt in*-Node zur Verfügung, mit dem sich ein MQTT-Topic abonnieren lässt [35]. Über diesen Weg können die Datenpakete der Sensoren von Chirp-Stack an Node-RED übermittelt werden. Hier können die Daten nun aufbereitet und die benötigten Messwerte aus der Nachricht herausgefiltert werden. Für die Realisierung der OPC-UA-Anbindung lässt sich das "node-red-contrib-opcua"-Paket verwenden. Dieses Paket beinhaltet unter anderem einen *OpcUa-Item*-Node und einen *OpcUa-Client*-Node, welche für eine Kommunikation mit dem OPC-UA-Server benötigt werden. Wichtige Parameter der beiden Nodes sind der Datentyp und die *NodeId* der jeweiligen Variable von der SPS, sowie die Adresse des OPC-UA-Servers und gegebenenfalls diverse Sicherheitseinstellungen [37].

Somit kann anhand der Implementierung eines Flows eine konkrete Verknüpfung zwischen einem Messwert eines LoRaWAN-Sensors und einer Variable der SPS erstellt werden. Durch das Verwenden mehrerer Flows lassen sich theoretisch beliebig viele Variablen unabhängig voneinander beschreiben. Ein Benutzer soll in dieser Anwendung sowohl den LoRaWAN-Sensor als auch die zu beschreibende Variable auf der SPS wählen können, indem das MQTT-Topic und die *NodeId* innerhalb eines Flows geändert wird.

3.3.2 Möglichkeit der Realisierung einer Schnittstelle über Python

Die Programmiersprache Python bietet durch die Verwendung verschiedener Bibliotheken eine weitere Möglichkeit zur Realisierung der Schnittstelle zur SPS per OPC-UA.

In Python steht mit der "paho-mqtt"-Bibliothek ein MQTT-Client zur Verfügung. Durch das Abonnieren des jeweiligen MQTT-Topics können die Datenpakete der LoRaWAN-Sensoren einem Python-Programm bereitgestellt werden. Eine Anleitung zur Verwendung der Bibliothek ist in der Quelle [20] vorhanden.

Im Anschluss kann in Python ebenfalls das Herausfiltern der Nutzdaten aus dem jeweils übermittelten Datenpaket programmiert werden. Dabei lassen sich die Nutzdaten eines Sensors beispielsweise in einer Instanz eines vordefinierten Objektes zwischenspeichern. Durch das Erzeugen weiterer Instanzen dieses Objektes können in Python die Messdaten aller Sensoren bereitgestellt werden.

Für die Weiterleitung der Nutzdaten an die SPS per OPC-UA bietet Python die "opcua-asyncio"-Bibliothek, mit der sich ein asynchroner OPC-UA-Client implementieren lässt [45]. Die notwendigen Konfigurationen für den Client und für den Verbindungsaufbau

zum Server, sowie einige weitere Beispiele für die Verwendung der Bibliothek können der Quelle [41] entnommen werden.

Durch das Auswählen der *NodeId* einer Variable von der SPS und einer konkreten Instanz des zuvor beschriebenen Objektes kann ein Benutzer eine neue Verknüpfung erstellen oder eine bereits implementierte Verknüpfung bei Bedarf verändern. Um die Bedienung zu vereinfachen, könnte hierfür eine grafische Bedienoberfläche erstellt werden.

3.3.3 Wahl der Schnittstelle für die OPC-UA-Anbindung

Nach Betrachtung der beiden Möglichkeiten wird sich für die Verwendung von Node-RED entschieden. Als ein wesentlicher Vorteil von Node-RED für die konkrete Anwendung wird die übersichtliche und nachvollziehbare Art der visuellen Programmierung der Verbindungen angesehen. Durch die Verwendung der zuvor beschriebenen Nodes in Node-RED wird die Komplexität der MQTT- und OPC-UA-Schnittstelle verborgen, wobei einem Benutzer die für ihn notwendigen Parameter gezielt zur Verfügung stehen. Außerdem eignet sich hier die grundlegend ereignisorientierte Funktionsweise von Node-RED, da die LoRaWAN-Sensoren ihre aktuellen Messdaten nur in zeitlichen Abständen versenden.

Die Node-RED-Programmierumgebung wird im Rahmen der Implementierungsphase ebenfalls auf dem Industrie-PC installiert.

3.4 Zusammenführung der Komponenten zu einem Gesamtkonzept

In diesem Abschnitt werden die zuvor getroffenen Entscheidungen zu einem Gesamtkonzept zusammengeführt. Das Gesamtkonzept soll verdeutlichen, auf welchem Wege die Messdaten der LoRaWAN-Sensoren an eine SPS weitergeleitet werden sollen. In Abbildung 3.1 wird das Gesamtkonzept visualisiert, wobei die Verbindungen der zum Einsatz kommenden Hardware- und Softwarekomponenten dargestellt sind. Hierbei ist bereits die konkrete Hardware zu sehen, die zu Beginn der Implementierungsphase in Abschnitt 4.1 vorgestellt wird.

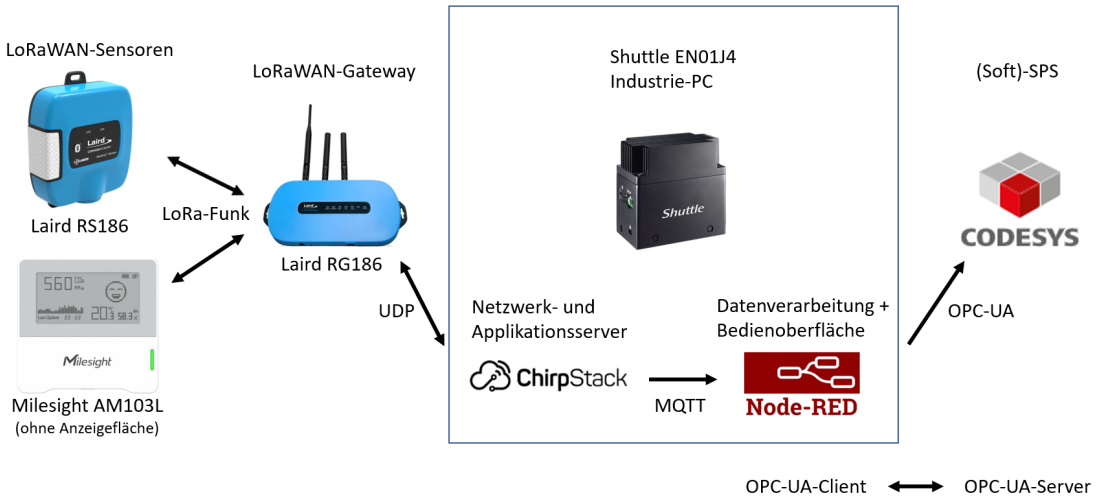

Abbildung 3.1: Darstellung des Gesamtkonzeptes zur Bereitstellung von LoRaWAN-Messwerten an eine SPS durch Verwendung einer OPC-UA-Anbindung[1]

Die Abbildung 3.1 zeigt, dass die Sensoren ihre Daten per LoRa-Funktechnologie zunächst an das Gateway senden. Das Gateway übermittelt im Anschluss die empfangenen Datenpakete mit dem UDP-Netzwerkprotokoll an den Netzwerk- und Applikationsserver von ChirpStack. Von hier aus werden die Datenpakete über MQTT an eine Node-RED-Anwendung gesendet, wo die Verarbeitung der Datenpakete erfolgt. Außerdem werden die jeweils aktuellen Messwerte per OPC-UA an eine SPS weitergeleitet und in die dafür vorgesehenen Variablen geschrieben. In Node-RED kann der Benutzer sowohl bestehende Verknüpfungen zwischen den Messwerten der LoRaWAN-Sensoren und den Variablen der SPS bearbeiten als auch neue Verknüpfungen anlegen.

[1] Bildquellen: Laird RS186 [17], Milesight AM103L [30] (Datenblatt), Laird RG186 [16], Shuttle EN01J4 [50], ChirpStack-Logo [4], Node-RED-Logo [36], CODESYS-Logo [9]

4 Implementierung

Nachdem in Kapitel 3 ein Konzept für die Lösung der Problemstellung dieser Arbeit erstellt wurde, folgt in diesem Kapitel die Implementierung des Konzeptes. Hierfür wird ein Testaufbau verwendet, welcher die in Abbildung 3.1 visualisierte Bereitstellung der Messwerte der LoRaWAN-Sensoren an eine SPS realisieren soll. Zunächst werden die einzelnen Hardwarekomponenten des Testaufbaus vorgestellt. Daraufhin folgt die Erstellung einer Gliederung der durchzuführenden Implementierung in Form von Arbeitspaketen. Dies soll eine gezielte und systematische Umsetzung der Problemlösung ermöglichen. Im Anschluss wird dann die Durchführung der einzelnen Arbeitspakete erläutert.

4.1 Verwendete Hardware für den Testaufbau

In diesem Abschnitt werden die Hardwarekomponenten vorgestellt, welche im Rahmen des Projektes für den Testaufbau zur Verfügung stehen. Die folgende Beschreibung der Hardware soll außerdem die zukünftige Bedienung der Geräte erleichtern. Die Hardware für diese Arbeit setzt sich aus einem Industrie-PC, einem LoRaWAN-Gateway und mehreren LoRaWAN-Sensoren von zwei verschiedenen Herstellern zusammen.

4.1.1 Shuttle-Industrie-PC Edge EN01J4

In der Konzeptphase in Abschnitt 3.1 wurde festgelegt, dass der Industrie-PC *Edge EN01J4* der Firma *Shuttle* als zentrale Rechnereinheit verwendet werden soll. Der Industrie-PC kann im Dauerbetrieb genutzt werden, was eine wichtige Eigenschaft für den zu implementierenden Anwendungsfall darstellt. Als Betriebssystem wird sowohl Linux als auch Windows 10 unterstützt. Die Spannungsversorgung erfolgt über einen 12-19V DC-Eingang mit einem 2-poligen Euroblock-Anschluss. Der Industrie-PC besitzt außerdem eine Halterung für eine Hutschiene, sodass der PC im Projekt in einem Schaltschrank montiert werden kann [50].

4.1.2 Laird Connectivity - Sentrius RG186 Gateway

Als LoRaWAN-Gateway wird das *Sentrius RG186* der Firma *Laird Connectivity* eingesetzt. Das *Sentrius RG186* bietet für LoRaWAN-Übertragungen acht unterschiedliche Kanäle und besitzt laut Hersteller eine Reichweite von bis zu 16 Kilometern. Als weitere Schnittstellen stehen Ethernet und Wi-Fi zur Verfügung.

Für die Konfiguration des *Sentrius RG186*-Gateways ist ein Web-Interface vorhanden, welches über zwei verschiedene Wege erreicht werden kann. Einerseits kann die URL *https://rg1xx123456.local* in einem Webbrowser eingetragen werden, wobei "123456" die letzten sechs Stellen der MAC-Adresse des Gateways darstellen. Zusätzlich kann auch eine Verbindung über die IP-Adresse des Gateways aufgebaut werden. Das Dashboard ist in Abbildung 4.1 zu sehen und bietet grundlegende Informationen zu den zur Verfügung stehenden Schnittstellen [16].

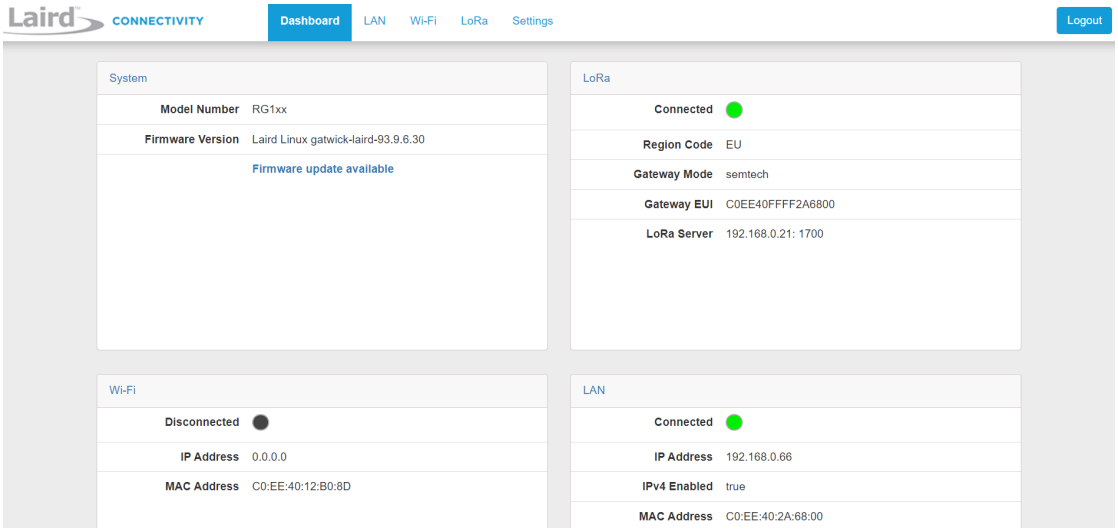

Abbildung 4.1: Dashboard des Sentrius RG186 Gateways, mit grundlegenden Informationen zu den Schnittstellen und dem System

In dem Web-Interface des Gateways muss der Packet-Forwarder zur Weiterleitung der Datenpakete an den LoRaWAN-Netzwerkserver, sowie die Adresse des Netzwerkservers und die Ports für den Uplink und Downlink eingerichtet werden. Außerdem kann hier der über das Gateway laufende Nachrichtenverkehr verfolgt werden. Des Weiteren besteht die Möglichkeit, die LAN- und Wi-Fi-Einstellungen des Gateways einzusehen und anzupassen.

4.1.3 Laird Connectivity - Sentrius RS186 Sensor

Der *Sentrius RS186* ist ein batteriebetriebener Temperatur- und Luftfeuchtigkeitssensor von *Laird Connectivity*. Zur Konfiguration des Sensors wird die "Sentrius Sensor"-App benutzt, die für Android- und iOS-Geräte zur Verfügung steht. Die Verbindung wird hierbei über Bluetooth aufgebaut, indem die Bluetooth-Taste am Sensor gedrückt wird. Die App liefert eine Übersicht zu den Kennungen des Sensors wie der DevEUI, der AppEUI und dem AppKey, die sich hier bei Bedarf verändern lassen. In der App kann auch das Sendeintervall des Sensors festgelegt werden. Die App bietet außerdem die Möglichkeit, aktuelle Parameter wie die Signalstärke, das Signal-Rausch-Verhältnis und die Datenrate des Sensors einzusehen. Der Benutzer kann sich in der App die zuletzt aufgenommenen Messwerte anzeigen lassen [17]. In Abbildung 4.2 ist die Startoberfläche der App (links) und der grafische Verlauf der Messwerte der Luftfeuchtigkeit (rechts) exemplarisch dargestellt.

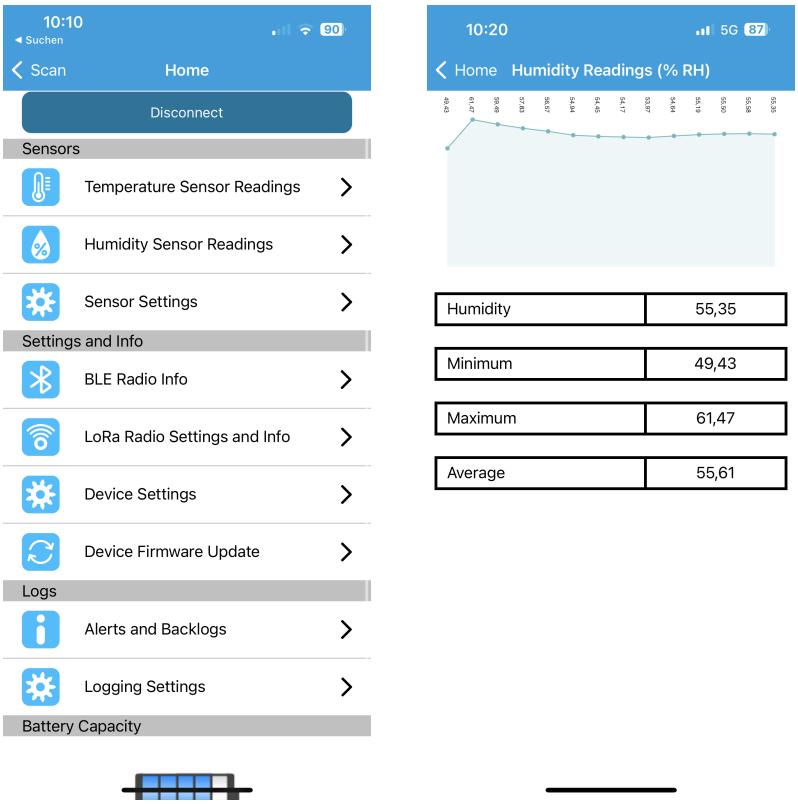

Abbildung 4.2: Screenshots aus der "Sentrius Sensor"-App; Links: Startoberfläche; Rechts: Verlauf der Luftfeuchtigkeitsmesswerte (aufgenommen mit einem iOS-Smartphone)

4.1.4 Milesight AM103L Sensor

Als weiterer Sensortyp wird der batteriebetriebene *AM103L*-Sensor der Firma *Milesight* verwendet, welcher die Temperatur, die Luftfeuchtigkeit und die CO2-Konzentration misst. Für die Konfiguration des *AM103L*-Sensors steht die "Milesight Toolbox"-App zur Verfügung, die ebenfalls von Android- und iOS-Geräten nutzbar ist. Hierbei wird die Verbindung zwischen Sensor und Smartphone-App per *Near Field Communication*[2] (NFC) realisiert. Durch das Auslesen des Sensors mit NFC werden die aktuellen Messwerte und die LoRaWAN-Einstellungen des Sensors sichtbar (siehe Abbildung 4.3). Dazu gehören unter anderem die Art des Netzwerkbeitrittes, die Geräteklasse, das Sendeintervall, die Datenrate und die Übertragungsleistung. Auch beim *AM103L* lassen sich der AppKey und die AppEUI manuell verändern, wobei jedoch die DevEUI nicht bearbeitet werden kann [30].

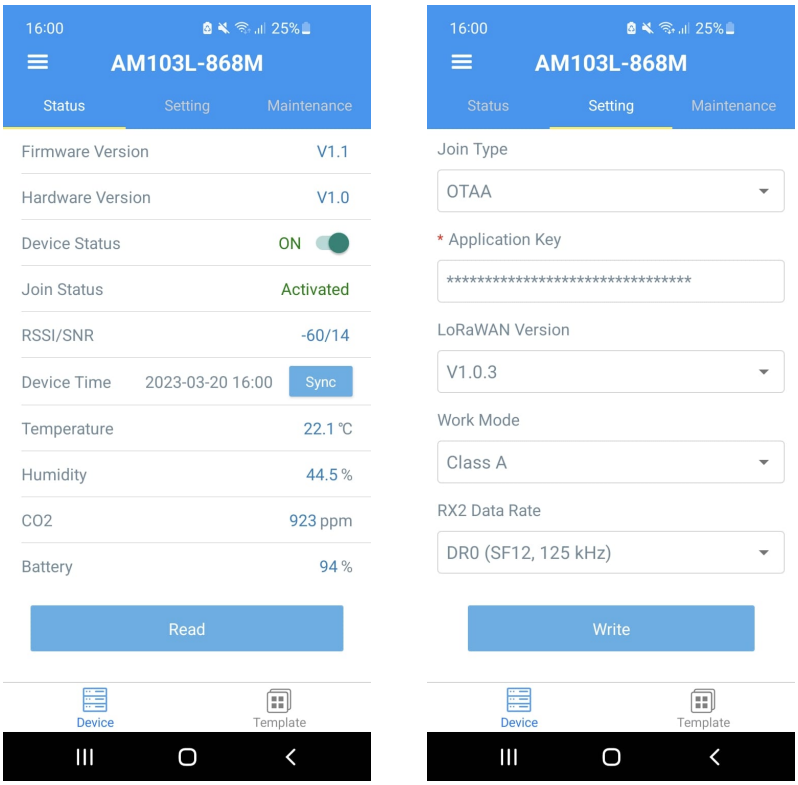

Abbildung 4.3: Screenshots aus der "Milesight Toolbox"-App; Links: Allgemeine Sensordaten; Rechts: Diverse LoRaWAN-Einstellungen (aufgenommen mit einem Android-Smartphone)

[2]NFC ist eine Technologie zur drahtlosen Datenübertragung per Funk für kurze Distanzen

4.2 Festlegung der Arbeitspakete

In diesem Abschnitt erfolgt die Definition der Arbeitspakete. Diese beschreiben die wesentlichen Schritte, welche für die Implementierung der in der Konzeptphase erarbeiteten Problemlösung durchzuführen sind. Die Arbeitspakete werden hierbei in einer aufeinander aufbauenden Reihenfolge definiert, in der auch die Umsetzung in Abschnitt 4.3 erfolgt. Es werden folgende Arbeitspakete festgelegt:

1. Aufsetzen des Debian-Betriebssystems
2. Installation des LoRaWAN-Netzwerkservers
3. Einrichtung des LoRaWAN-Netzwerkservers
4. Einbindung der LoRaWAN-Hardware
5. Erstellen eines CODESYS-Beispielprojektes
6. Einrichtung einer MQTT- und OPC-UA-Schnittstelle in Node-RED
7. Erstellen der Datenverarbeitung in Node-RED
8. Erstellen einer Bedienmöglichkeit in Node-RED

4.3 Umsetzung der Arbeitspakete

Dieser Abschnitt befasst sich mit der konkreten Implementierung des erstellten Konzeptes, indem die zuvor definierten Arbeitspakete schrittweise umgesetzt und dabei erläutert werden.

4.3.1 Aufsetzen des Debian-Betriebssystems

Das Aufsetzen von Debian Linux auf dem Shuttle-Industrie-PC erfolgt anhand eines USB-Sticks, welcher zuvor mit der Anwendung *Rufus* und einer heruntergeladenen ISO-Image-Datei (ISO-Abbild) des Betriebssystems als bootfähiges Medium eingerichtet wurde. Es wird die zum Zeitpunkt der Implementierung aktuelle Debian-Version 11.6.0 “Bullseye“ (Stand: 13.03.2023) verwendet, wobei die ISO-Image-Datei aus der Quelle [10] verwendet

wird. Beim ersten Booten des Industrie-PCs wird der USB-Stick als bootfähiges Medium ausgewählt, woraufhin die Installation des Debian-Betriebssystems durchgeführt wird.

4.3.2 Installation des LoRaWAN-Netzwerkservers

Als Nächstes wird der ChirpStack-Netzwerkserver, sowie die weiteren dazugehörigen Komponenten auf dem Industrie-PC über den "apt"-Paketmanager installiert. Hierbei wird die unter der Quelle [6] zur Verfügung gestellte Anleitung verwendet, welche die auszuführenden Befehle für die Installation der Komponenten beinhaltet. Diese Anleitung ist für die Linux-Betriebssysteme Debian und Ubuntu vorgesehen. Im Folgenden werden die einzelnen Schritte beschrieben, wobei weitere Informationen den Quellen [3] und [7] entnommen werden.

Mosquitto MQTT-Broker

Als Erstes wird der Mosquitto MQTT-Broker (auch MQTT-Server genannt) in der Version 2.0.11 installiert. ChirpStack nutzt das Nachrichtenprotokoll MQTT unter anderem für die interne Kommunikation zwischen den einzelnen Komponenten der ChirpStack-Architektur (siehe Abbildung A.1). Außerdem wird MQTT in ChirpStack als Integrationsschnittstelle für weitere Anwendungen eingesetzt, indem die Datenpakete des LoRaWAN-Netzwerkes über MQTT-Topics bereitgestellt werden (siehe Abschnitt 3.3).

Redis-Datenbank

In ChirpStack wird eine Redis-Datenbank genutzt, in der die Daten zu den jeweils aktiven Geräte-Sitzungen des LoRaWAN-Netzwerkes hinterlegt sind. Es wird die Version 6.0.16 verwendet.

Bei Redis handelt es sich um eine Open-Source-In-Memory-Datenbank, welche den Arbeitsspeicher eines Computers als Datenspeicher verwendet. Dies ermöglicht einen wesentlich schnelleren Zugriff auf die gespeicherten Daten im Vergleich zu anderen Datenbank-Typen. Außerdem bietet Redis die Möglichkeit, die Daten in definierten Intervallen auch persistent auf einem nicht-flüchtigen Speichermedium zu sichern [44].

PostgreSQL-Datenbank

Als weitere Datenbank wird PostgreSQL (Version 13.9) installiert, in welcher die persistenten Daten von ChirpStack gespeichert werden. Hierbei muss nach der Installation eine sogenannte Rolle angelegt werden, mit der sich dann eine Datenbank erstellen lässt.

ChirpStack-Repository

In diesem Schritt wird das ChirpStack-Repository, welche für Debian/Ubuntu zur Verfügung steht, der Repository-Liste des Systems hinzugefügt. Dadurch können im Anschluss die neuesten Versionen der ChirpStack-Komponenten installiert werden.

ChirpStack Gateway Bridge

Als Nächstes wird die *ChirpStack Gateway Bridge* in der Version 4.0.6 installiert. Diese Komponente wandelt das jeweils verwendete Packet-Forwarder-Format der LoRaWAN-Gateways in das von ChirpStack verwendete Datenformat und umgekehrt. In der Konfigurationsdatei der *Gateway Bridge* müssen den hier aufgeführten MQTT-Topics für die interne Kommunikation jeweils der entsprechende Regionspräfix vorangestellt werden. Bei der Verwendung der 868 MHz-Frequenz in Europa lautet der Präfix "eu868".

ChirpStack

Abschließend wird der ChirpStack-Netzwerkserver und -Applikationsserver installiert (Version 4.3.0). Wie bei der Vorstellung von ChirpStack in Unterabschnitt 3.2.2 erwähnt, bilden der Netzwerk- und Applikationsserver ab der Version V4.0.0 eine gemeinsame Komponente.

4.3.3 Einrichtung des LoRaWAN-Netzwerkservers

Nach der Installation von ChirpStack steht eine Weboberfläche zur Einrichtung des Netzwerkservers und der Applikationen zur Verfügung. Diese kann bei Nutzung des Industrie-PCs über die URL *http://localhost:8080* oder alternativ über die IP-Adresse des Industrie-PCs anstelle von "localhost" aufgerufen werden, wenn ein anderer Computer in dem Netzwerk für den Zugriff auf die Weboberfläche verwendet wird. Die folgenden Schritte werden als Administrator in ChirpStack durchgeführt.

Zunächst muss ein sogenannter Tenant erstellt werden, der einen separaten Bereich auf dem Netzwerkserver erzeugt. Diesem werden dann die LoRaWAN-Geräte hinzugefügt. Hierbei kann festgelegt werden, ob dem Tenant Gateways hinzugefügt werden können und ob diese Gateways nur für den konkreten Tenant privat nutzbar sind. Außerdem kann die maximale Anzahl der Gateways und Endgeräte für den Tenant begrenzt werden. Die Angabe "0" bedeutet an dieser Stelle, das theoretisch unendlich viele Geräte hinzugefügt werden können.

Da ChirpStack das Multi-Tenancy-Konzept unterstützt, können mehrere unabhängige Tenants innerhalb eines Netzwerkservers vom Administrator erstellt werden.

Des Weiteren lassen sich verschiedene Benutzer für den ChirpStack-Netzwerkserver und -Applikationsserver anlegen, denen bei Bedarf auch Administrator-Rechte zugeteilt werden können. Ein erstellter Benutzer kann im Anschluss einem oder auch mehreren Tenants zugeordnet werden, wobei jeweils konkrete Zugriffs- und Bearbeitungsrechte für den Benutzer festgelegt werden können. Dazu gehört, dass der Benutzer als Administrator des Tenants registriert werden kann. Außerdem wird hier eingestellt, ob der Benutzer neue Gateways und Endgeräte hinzufügen oder bereits vorhandene Geräte bearbeiten darf.

Weitere Einstellungen des ChirpStack-Netzwerkservers können über die bei der Installation erstellte Konfigurationsdatei vorgenommen werden.

4.3.4 Einbindung der LoRaWAN-Hardware

Zunächst wird in der Web-Anwendung des *Sentrius RG186*-Gateways die Adresse des Netzwerkservers eingetragen. Da ChirpStack lokal auf dem Industrie-PC betrieben wird, wird hier dessen IP-Adresse verwendet. Als Packet-Forwarder wird der *Semtech UDP Forwarder* über den Port 1700 genutzt. Daraufhin wird das Gateway anhand der Gateway-ID, die auch als Gateway-DevEUI bezeichnet wird, in ChirpStack eingefügt.

Bevor ein neuer Endgerät-Typ (zum Beispiel von einem weiteren Hersteller oder in einer anderen Ausführung) in ChirpStack registriert werden kann, muss ein Device-Profil angelegt werden. Das Device-Profil beinhaltet unter anderem Angaben zur unterstützten LoRaWAN-Version, den regionalen Parametern, der Art des Netzwerkbeitrittes, sowie zum De- und Encodieren der Datenpakete und muss für jeden Endgerät-Typ einmalig erstellt werden. Daraufhin kann das erstellte Device-Profil direkt bei der Registrierung der weiteren Endgeräte dieses Types verwendet werden. In Abbildung 4.4 ist ein Ausschnitt der einzustellenden Parameter für ein Device-Profil zu sehen.

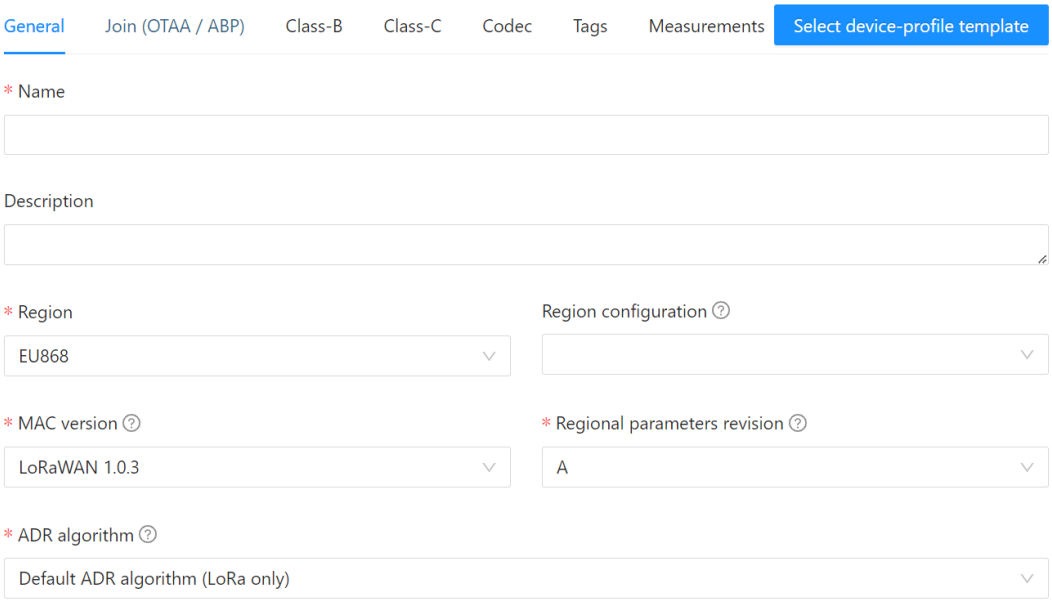

Abbildung 4.4: Ausschnitt der einzustellenden Parameter eines Device-Profiles, welches für die Registrierung der Endgeräte benötigt wird

Das De- und Encodieren der Datenpakete für die verwendeten Sensortypen kann in ChirpStack entweder über *Cayenne LPP*[3] oder über JavaScript-Funktionen realisiert werden. Für den konkreten Anwendungsfall ist zunächst nur das Decodieren der empfangenen Daten notwendig. Hierfür werden jeweils JavaScript-Funktionen verwendet, wobei für die beiden verwendeten Sensortypen Vorlagen auf Github vorhanden sind. Der Code für den *Sentrius RS186* steht in der Quelle [2] und für den *Milesight AM103L* in der Quelle [32] zur Verfügung. Die beiden Code-Vorlagen werden an den vorgegebenen Syntax von ChirpStack V4 angepasst und sind im Anhang A in Listing A.1 und Listing A.2 zu sehen.

Als Nächstes wird eine Applikation in ChirpStack angelegt, in welcher die verwendeten Sensoren registriert werden können.

Beim Hinzufügen eines Sensors wird die DevEUI und das jeweils zuvor erstellte Device-Profil benötigt. Da für den Netzwerkbeitritt die OTAA-Methode (Unterabschnitt 2.2.5) verwendet wird, muss im Anschluss der AppKey des Sensors angegeben werden. Daraufhin wird in der Applikation ersichtlich, ob der Netzwerkbeitritt erfolgreich war.

[3] *Cayenne LPP* (*Low Power Payload*) ist ein Protokoll zur Übertragung von Sensorwerten in LPWAN-Netzwerken [34]

4.3.5 Erstellen eines CODESYS-Projektes

Damit im Rahmen der Validierung die Funktionalität der OPC-UA-Schnittstelle überprüft werden kann, wird ein CODESYS-Beispielprojekt auf einem weiteren Computer innerhalb des Netzwerkes angelegt. Das CODESYS-Projekt soll mehrere Variablen und eine Verbindung zu einem OPC-UA-Server besitzen. Hierbei wird entschieden, dass zu Testzwecken die Nutzung einer Soft-SPS als OPC-UA-Server ausreichend ist. Mit einer Soft-SPS kann das Verhalten einer realen SPS auf einem Computer nachgebildet und simuliert werden. Es wird die CODESYS Version V3.5.18.40 installiert [8] und ein Standard-Projekt angelegt, wobei das *CODESYS Control Win V3* als Soft-SPS verwendet wird.

Die Messwerte der LoRaWAN-Sensoren sollen für die Heizkreisregelung voraussichtlich in verschiedenen Instanzen eines Funktionsbausteines (FB) ausgewertet werden. Um die Funktionsweise der Implementierung im Anschluss möglichst praxisnah validieren zu können, wird ein FB mit dem Namen *HeatingCircuitControl* erstellt. In dieser Arbeit werden als Messwerte die Temperatur und die Luftfeuchtigkeit betrachtet, sodass dem FB die lokalen Variablen *temperature* und *humidity* des Datentypes *REAL* hinzugefügt werden. In dem Programm *PLC_PRG* wird daraufhin die Instanz *Room1* des FBs angelegt. Es wird eine Symbolkonfiguration erstellt, in der die Berechtigung für das Beschreiben der Variablen per OPC-UA aktiviert wird.

Falls die Daten verschlüsselt und signiert per OPC-UA übertragen werden sollen, muss ein Zertifikat für den OPC-UA-Server im Security-Screen des CODESYS-Projektes erstellt werden. Das Zertifikat kann hierbei eine maximale Gültigkeit von bis zu zehn Jahren haben. In den Gerätesicherheitseinstellungen, welche in der Netzwerkansicht über den Reiter “Gerät“ erreicht werden, lassen sich die genauen Parameter für die Nutzung des OPC-UA-Servers einstellen.

Es wird entschieden, dass im Rahmen dieser Arbeit zunächst nur eine anonyme Verbindung zwischen dem OPC-UA-Server und -Client aufgebaut werden soll, sodass hier keine weiteren Sicherheitseinstellungen vorgenommen werden. Um eine anonyme Verbindung zu erlauben, muss in der Laufzeitsystemsicherheitsrichtlinie, die ebenfalls in der Netzwerkansicht in dem Reiter “Gerät“ vorzufinden ist, eine entsprechende Berechtigung erteilt werden (siehe Abbildung 4.5).

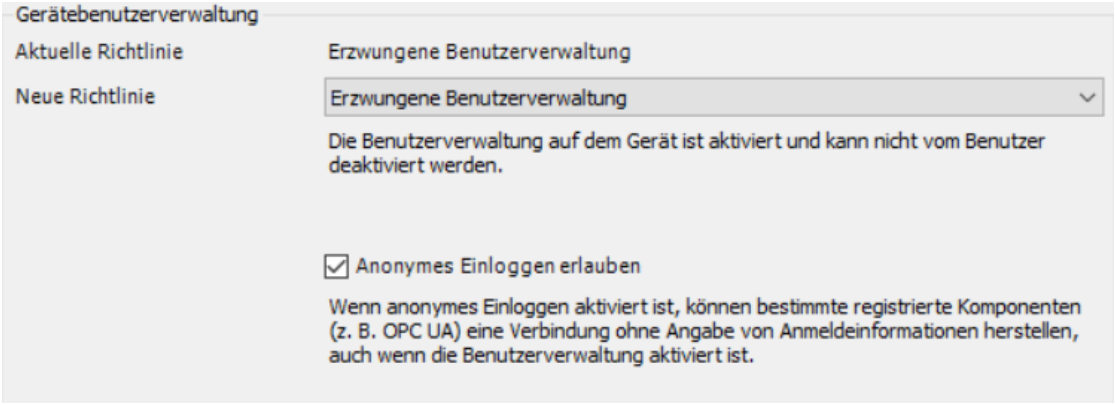

Abbildung 4.5: Darstellung des Berechtigungsfeldes, mit dem eine anonyme Verbindung zum OPC-UA-Server erlaubt wird

4.3.6 Einrichtung einer MQTT- und OPC-UA-Schnittstelle in Node-RED

Im Folgenden wird die MQTT-Schnittstelle zu ChirpStack und die OPC-UA-Anbindung zur CODESYS-Soft-SPS in Node-RED erstellt. Hierbei wird die Node-RED Version V3.0.2 auf dem Industrie-PC installiert [38]. Für die Programmierung steht eine Weboberfläche zur Verfügung. Diese lässt sich am Industrie-PC über die URL *http://localhost:1880* oder über die IP-Adresse des Industrie-PCs bei Nutzung eines anderen Computers in dem Netzwerk erreichen. Als Grundlage für dieses Arbeitspaket gelten die Erläuterungen aus Unterabschnitt 3.3.1 während der Konzeptphase.

Zuerst wird eine Verbindung zum Mosquitto-MQTT-Broker, der in Unterabschnitt 4.3.2 installiert wurde, aufgebaut. Hierfür wird das *mqtt in*-Node verwendet. Die Übertragungen per MQTT erfolgen über den Port 1883. Da Node-RED und der Mosquitto-MQTT-Broker beide auf dem Industrie-PC betrieben werden, kann die Verbindung zum Broker über *http://localhost:1883* erstellt werden. Der Aufbau der MQTT-Topics, über die ChirpStack die Ereignisse des LoRaWAN-Netzwerkes bereitstellt, ist in Abschnitt 3.3 erläutert. Die benötigte Application-ID kann in der ChirpStack-Weboberfläche eingesehen werden. Durch das Festlegen der DevEUI im Topic werden nur Datenpakete des konkreten Sensors empfangen. Der Syntax der ausgegebenen Nachricht des Nodes wird als *JavaScript Object Notation* (JSON)-Objekt gewählt. Dies ermöglicht im weiteren Verlauf den Zugriff auf die einzelnen Felder des Datenpaketes über den Punktoperator. Die Einstellungen des Nodes sind in Abbildung 4.6 dargestellt.

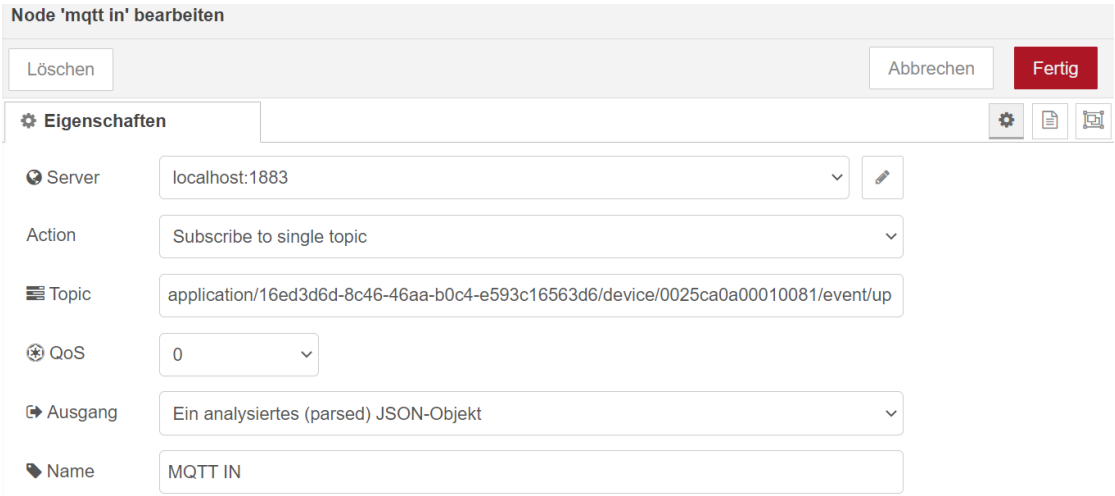

Abbildung 4.6: Einstellungen des *mqtt in*-Nodes für die Verbindung zu ChirpStack

Für die OPC-UA-Schnittstelle wird das “node-red-contrib-opcua“-Paket in der Version 0.2.296 über die Palette in Node-RED installiert. Für die Verbindung zum OPC-UA-Server wird das *OpcUa-Client*-Node verwendet. Ein Ausschnitt der Einstellungen ist in Abbildung 4.7 dargestellt. In dem Node wird die Adresse des OPC-UA-Servers im Feld “Endpoint“ eingetragen. Wie zuvor erwähnt, wird hier eine anonyme Verbindung ohne weitere Sicherheitseinstellungen eingerichtet.

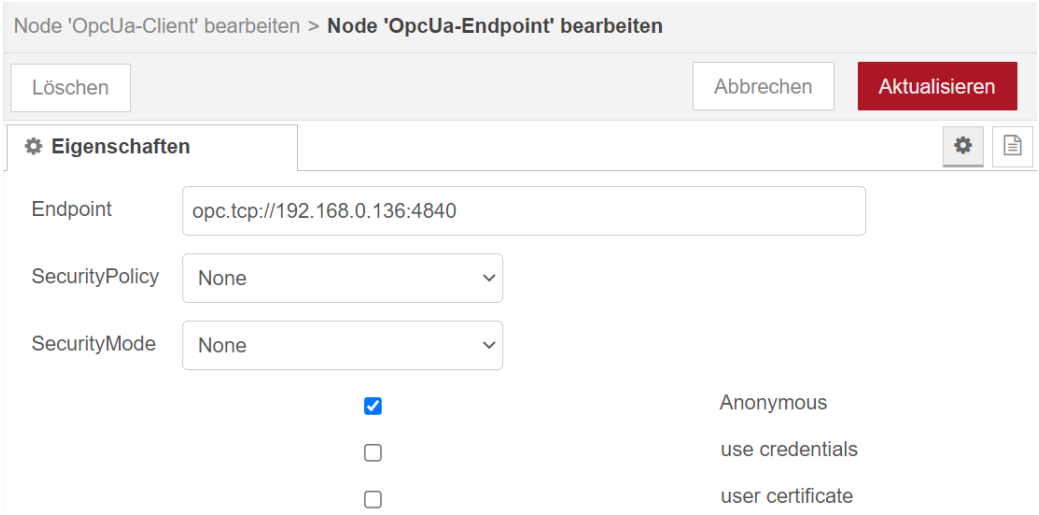

Abbildung 4.7: Ausschnitt der Einstellungen des *OpcUa-Client*-Nodes für die Verbindung zum OPC-UA-Server

Außerdem wird das *OpcUa-Item*-Node verwendet, mit dem festgelegt wird, welche konkrete OPC-UA-Variable beschrieben werden soll. Das *OpcUa-Item*-Node verwendet für das Schreiben den jeweils aktuellen Inhalt der Eigenschaft *msg.payload*, der dem Node bereitgestellt wird. An dieser Stelle muss auch die *NodeId* und der Datentyp der Variable eingetragen werden. Eine Möglichkeit zum Bestimmen der *NodeId* bietet der OPC-UA-Test-Client UaExpert der Firma *Unified Automation*, mit welchem sich der Adressraum eines OPC-UA-Servers (siehe Abschnitt 2.4) anzeigen lässt. In Abbildung 4.8 ist exemplarisch ein Ausschnitt der bereitgestellten Informationen zu der Variable *humidity* der FB-Instanz *Room1* in UaExpert zu sehen.

Attribute	Value
⌄ NodeId	ns=4;s=|var|CODESYS Control Win V3 x64.Application.PLC_PRG.Room1.humidity
NamespaceIndex	4
IdentifierType	String
Identifier	|var|CODESYS Control Win V3 x64.Application.PLC_PRG.Room1.humidity
NodeClass	Variable
BrowseName	4, "humidity"
DisplayName	"en-Us", "humidity"

Abbildung 4.8: Ausschnitt der Informationen zu einer OPC-UA-Variable, die sich mit dem Test-Client UaExpert anzeigen lassen

4.3.7 Erstellen der Datenverarbeitung in Node-RED

An dieser Stelle wird der in Unterabschnitt 3.3.1 erarbeitete Ansatz verfolgt, dass für jede zu erstellende Verbindung zwischen einem Messwert eines LoRaWAN-Sensors und einer OPC-UA-Variable jeweils ein eigener Flow angelegt werden soll. Dies bietet eine direkt zugängliche Übersicht der aktiven Verknüpfungen in Node-RED, sowie die Möglichkeit zum Hinzufügen neuer Verbindungen durch das Kopieren eines bestehenden Flows. Außerdem wird damit das Ziel verfolgt, eine kompakte Datenverarbeitung realisieren zu können.

Für die Datenverarbeitung wird ein *function*-Node verwendet, welcher gleichzeitig auch die Verbindung zwischen den Nodes der MQTT- und OPC-UA-Anbindung bildet. Der sich dadurch ergebende Flow ist in Abbildung 4.9 dargestellt.

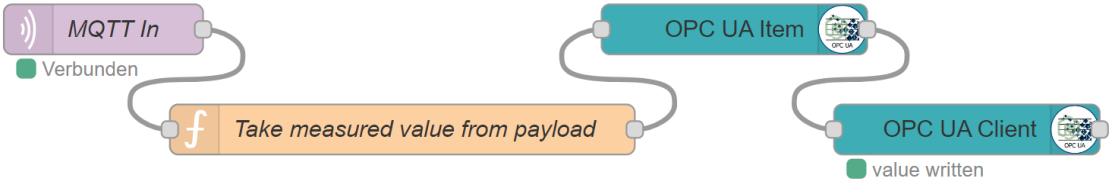

Abbildung 4.9: Implementierter Flow zur Realisierung der Messdaten-Verarbeitung und Weiterleitung per OPC-UA

In dem *function*-Node wird das Herausfiltern der aktuell gemessenen Temperatur beziehungsweise der Luftfeuchtigkeit aus dem gesendeten Datenpaket eines LoRaWAN-Sensors durchgeführt. Das vom *mqtt in*-Node empfangene und weitergeleitete Datenpaket ist über *msg.payload* erreichbar. Beim Decodieren der Nachrichten in ChirpStack werden die Messwerte der Sensoren in einem Objekt mit dem Namen *object* gespeichert. Somit lässt sich beispielsweise die Temperatur wie folgt mit dem Punktoperator erreichen:

```
1 msg.payload = msg.payload.object.temperature;
2
3 return msg;
```

Listing 4.1: Herausfiltern des benötigten Messwertes aus dem Datenpaket eines LoRaWAN-Sensors und Bereitstellung an das folgende Node

Durch die Zuweisung des Messwertes an *msg.payload* wird nur noch der konkrete Messwert an das folgende *OpcUa-Item*-Node des Flows weitergeleitet.

4.3.8 Erstellen einer Bedienmöglichkeit in Node-RED

In diesem Abschnitt wird eine Bedienmöglichkeit in Node-RED entwickelt, mit der sich die wesentlichen Parameter des in Abbildung 4.9 dargestellten Flows an einer zentralen Stelle bearbeiten lassen. Die Bedienmöglichkeit soll das Initialisieren neuer Verknüpfungen, sowie das Bearbeiten bereits bestehender Verbindungen vereinfachen.

Als Erstes wird der zuvor implementierte Flow gruppiert. Eine Gruppe wird dabei durch eine Umrandung der beinhalteten Nodes dargestellt. Hiermit kann der Flow auf der Programmieroberfläche einheitlich kopiert und verschoben werden. Außerdem lassen sich für eine Gruppe Umgebungsvariablen anlegen, mit denen die zentrale Bedienmöglichkeit für die implementierte Schnittstelle realisiert wird. Die Umgebungsvariablen sind nur für die jeweilige Gruppe sichtbar und können für jede einzelne Instanz der Gruppe unabhängig

voneinander verändert werden. Die Verwendung der Variablen in den Nodes erfolgt mit dem *${}*-Operator. In Abbildung 4.10 sind die definierten Umgebungsvariablen dargestellt.

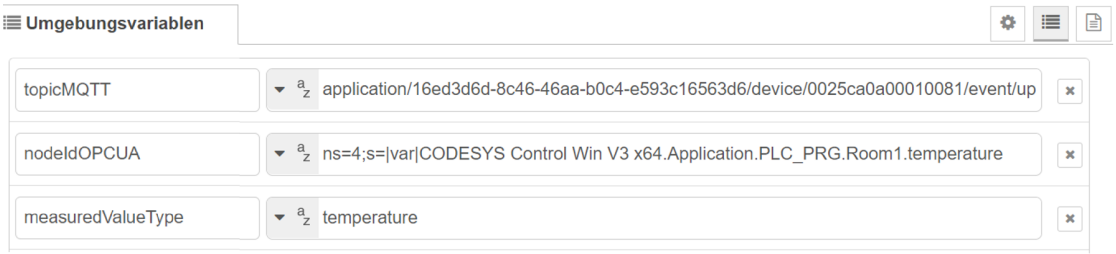

Abbildung 4.10: Umgebungsvariablen einer Gruppe für die Realisierung einer zentralen Bedienmöglichkeit eines Flows

Mit der Umgebungsvariable *topicMQTT* wird das MQTT-Topic des konkreten Sensors für das *mqtt-in*-Node festgelegt. Die *NodeId* der zu beschreibenden SPS-Variable für das *OpcUa-Item*-Node wird durch die Umgebungsvariable *nodeIdOPCUA* angegeben. Die Umgebungsvariable *measuredValueType* dient zur Bestimmung, ob der jeweilige Flow die Temperatur oder die Luftfeuchtigkeit als Messwert an die SPS weiterleiten soll. Je nach Verwendung muss der Variable entweder die Zeichenkette *temperature* oder *humidity* zugewiesen werden.

Aufgrund dieser implementierten Eigenschaft muss der Code des *function*-Nodes für das Herausfiltern des Messwertes aus Unterabschnitt 4.3.7 angepasst werden (siehe Listing 4.2). Im Programmcode wird die Umgebungsvariable durch die Funktion *env.get()* erreicht, indem die Umgebungsvariable mit Anführungszeichen als Argument übergeben wird.

```
1 if (env.get("measuredValueType") === "temperature") {
2     msg.payload = msg.payload.object.temperature;
3 } else if (env.get("measuredValueType") === "humidity") {
4     msg.payload = msg.payload.object.humidity;
5 }
6
7 return msg;
```

Listing 4.2: Bestimmung des weiterzuleitenden Messwertes durch Auslesen und Vergleichen der Umgebungsvariable *measuredValueType*

Des Weiteren wird aus dem *function*-Node und dem *OpcUa-Item*-Node ein Subflow mit dem Namen *Measurement Data Processing* erstellt. Mit einem Subflow lässt sich die visu-

elle Komplexität eines Flows verringern, indem mehrere Nodes zu einem einzelnen Node zusammengefasst werden. Außerdem besitzt ein Subflow die Möglichkeit zum Einrichten einer Status-Anzeige, welcher unterhalb des Subflow-Nodes im Gesamt-Flow angezeigt wird. Diese Eigenschaft wird genutzt, um einem Benutzer die wesentlichen Parameter der implementierten Schnittstelle direkt ersichtlich zu machen. Da die Flows in Node-RED ereignisorientiert ausgeführt werden, erfolgt das Erscheinen beziehungsweise das Aktualisieren der Status-Anzeige erst beim Empfangen eines neuen Datenpaketes. In der Status-Anzeige wird die DevEUI des LoRaWAN-Sensors, der Pfad der OPC-UA-Variable in CODESYS und der verwendete Messwerttyp dargestellt. Das Herausfiltern dieser Parameter erfolgt in einem weiteren *function*-Node. Die einzelnen Nodes des Subflows sind in Abbildung 4.11 visualisiert.

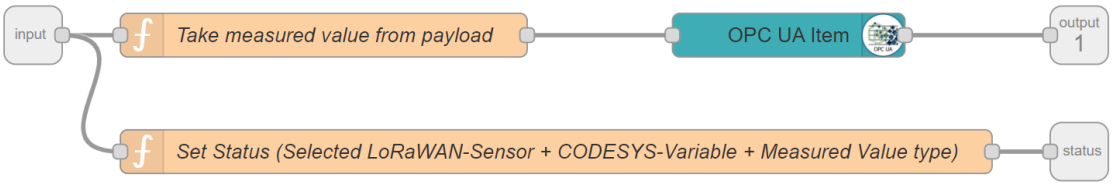

Abbildung 4.11: Darstellung der Nodes innerhalb des *Measurement Data Processing*-Subflows; zu sehen sind die Nodes der Datenverarbeitung aus Unterabschnitt 4.3.7 (oben), sowie ein *function*-Node für die Erstellung der Status-Anzeige (unten)

Der Code des *function*-Nodes für die Erstellung der Status-Anzeige ist in Listing 4.3 aufgeführt.

```
1  //Herausfiltern der DevEUI aus dem MQTT-Topic
2  var deviceEUI = env.get("topicMQTT").slice(56, 72);
3
4  //Herausfiltern des Pfades der konkreten Variable in CODESYS
5  var pathVariable = env.get("nodeIdOPCUA").slice(51);
6
7  //Erstellen der Status-Anzeige des Subflow-Nodes
8  var statusOutput = "DevEUI: " + deviceEUI + "; Variable: " + pathVariable +
                      "; Measured value type: " + env.get("measuredValueType");
9
10 msg.payload = {text : statusOutput};
11
12 return msg;
```

Listing 4.3: Erstellen der Status-Anzeige durch das Herausfiltern der wesentlichen Parameter aus den Umgebungsvariablen

In Abbildung 4.12 ist abschließend der implementierte Gesamt-Flow dargestellt, wobei auch die Status-Anzeige unterhalb des Subflow-Nodes *Measurement Data Processing* mit den Informationen zur aktuellen Verknüpfung zu sehen ist. Die finale Version der Programmierung in Node-RED ist auf der sich im Anhang befindlichen CD abgelegt und kann bei Bedarf eingesehen werden.

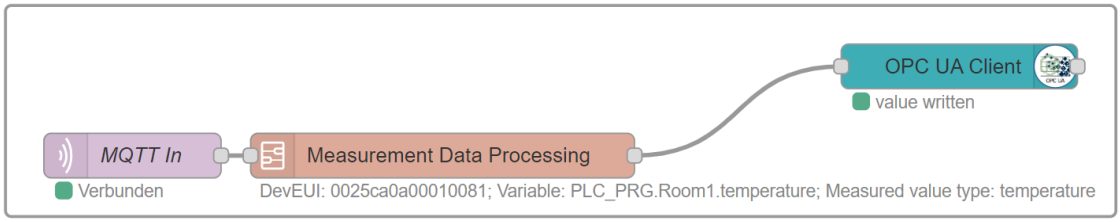

Abbildung 4.12: Implementierter Gesamt-Flow für die Schnittstelle, erweitert durch das Erstellen einer Gruppe mit zentraler Bedienmöglichkeit und eines Subflows mit Status-Anzeige

5 Validierung der Funktion

In diesem Kapitel wird nun die Funktion der zuvor implementierten Lösung für die in Abschnitt 1.1 beschriebene Problemstellung und Zielsetzung validiert. Hierbei wird konkret die korrekte Funktionsweise des LoRaWAN-Netzwerkes und der Schnittstelle in Node-RED zur OPC-UA-Anbindung überprüft. Außerdem werden jeweils verschiedene Anwendungsfälle bei der Nutzung der implementierten Lösung betrachtet.

5.1 Funktion des LoRaWAN-Netzwerkes

Zunächst wird das eingerichtete LoRaWAN-Netzwerk betrachtet. Es wird überprüft, ob die Datenpakete zuverlässig zwischen dem Netzwerk- und Applikationsserver, dem Gateway und den Sensoren übertragen werden.

Als Erstes wird die Kommunikation zwischen dem LoRaWAN-Gateway und dem ChirpStack-Netzwerkserver zu mehreren Zeitpunkten kontrolliert. Hierbei wird jeweils der Nachrichtenverkehr des Gateways über dessen Web-Interface (siehe Unterabschnitt 4.1.2) verfolgt und mit den ein- und ausgehenden Datenpaketen des Netzwerkservers abgeglichen. Die Pakete können in ChirpStack ebenfalls über dessen Web-Anwendung im Bereich der Gateway-Konfiguration eingesehen werden. Beim Abgleich wird festgestellt, dass die Uplink- und Downlink-Nachrichten zwischen den beiden Komponenten ohne Paketverluste ausgetauscht werden und somit eine zuverlässige Verbindung zwischen dem Gateway und dem Netzwerkserver besteht.

Des Weiteren wird festgestellt, dass die Datenpakete der beiden Sensor-Typen in dem jeweils konfigurierten Sendeintervall von fünf Minuten an die erstellte Chirpstack-Applikation bereitgestellt werden. Hierzu werden die registrierten Sensoren der Applikation (siehe Unterabschnitt 4.3.4) in der Web-Anwendung von ChirpStack betrachtet. An dieser Stelle wird auch erkannt, dass der *Frame Counter* wie vorgesehen kontinuierlich hochgezählt wird und somit alle Datenpakete der Sensoren an die Applikation

bereitgestellt werden. In Abbildung 5.1 ist exemplarisch ein Ausschnitt der aufgezeichneten Datenpakete des *Milesight AM103L*-Sensors in ChirpStack dargestellt.

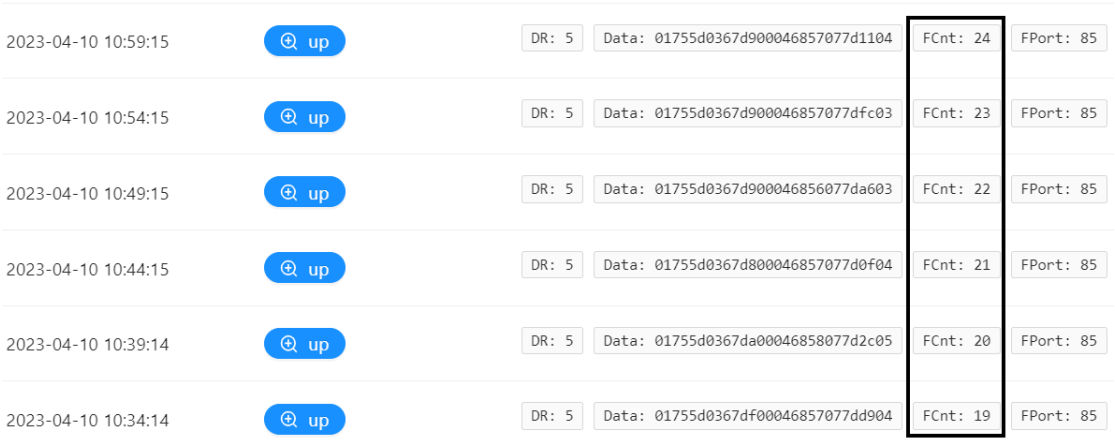

Abbildung 5.1: Ausschnitt der aufgezeichneten Datenpakete des Milesight AM103L-Sensors in der ChirpStack-Applikation, zu sehen sind die Empfangszeitpunkte der jeweiligen Datenpakete (links) und der hochzählende *Frame Counter* (schwarze Umrandung)

Außerdem bietet ChirpStack die Möglichkeit, die Aktivitäten des Netzwerk- und Appliaktionsservers in den Log-Dateien zu protokollieren, wodurch mögliche Fehler oder Probleme identifiziert werden können. Beim Betrachten der Log-Dateien wird jedoch festgestellt, dass keine Fehler bei der Nutzung von ChirpStack vorliegen.

Aus diesen Erkenntnissen wird geschlossen, dass sowohl der Netzwerk- und Applikationsserver, als auch das Gateway und die Sensoren in dem erstellten LoRaWAN-Netzwerk wie vorgesehen funktionieren.

5.1.1 Anwendungsfälle im Sensor-Management

Im Anschluss werden drei wesentliche Anwendungsfälle betrachtet, die in der Praxis bei der Verwendung der Sensoren im LoRaWAN-Netzwerk auftreten. Hierzu zählen das Hinzufügen neuer Sensoren, das Ersetzen bestehender Sensoren und das Wechseln der Batterien.

Hinzufügen eines neuen Sensors

Das Hinzufügen eines neuen Sensors zu einer ChirpStack-Applikation wurde bereits in Unterabschnitt 4.3.4 beschrieben. Falls es sich bei dem Sensor um einen neuen Sensortyp handelt, muss zusätzlich ein Device-Profil für diesen Sensortypen angelegt werden.

Ersetzen eines bestehenden Sensors

Ein weiterer Anwendungsfall ist das Ersetzen eines bestehenden Sensors durch einen neuen Sensor. Hierbei wird der neue Sensor wie zuvor erwähnt in der ChirpStack-Applikation registriert. Falls der bereits verwendete Sensor in dem Netzwerk nicht mehr genutzt werden soll, kann der Eintrag des Sensors in der Applikation zur Übersichtlichkeit gelöscht werden. Wenn der Sensor jedoch zu einem späteren Zeitpunkt zum Einsatz kommen soll, kann der Eintrag in der Applikation bestehen bleiben. Dadurch muss keine erneute Registrierung des alten Sensors in der Applikation durchgeführt werden, da die DevEUI und der AppKey bereits hinterlegt sind.

Außerdem muss die bestehende Verknüpfung in Node-RED angepasst werden, indem die DevEUI des neuen Sensors in der Umgebungsvariable des MQTT-Topics eingetragen wird.

Batteriewechsel der Sensoren

Ein Batteriewechsel der Sensoren kann im laufenden Betrieb vorgenommen werden. Im Anschluss wird automatisch der erneute Beitritt zum LoRaWAN-Netzwerk durch das Endgerät initiiert, sodass ein Benutzer keine Anpassungen in ChirpStack vornehmen muss. Bei der Verwendung der OTAA-Methode für den Netzwerkbeitritt des Endgerätes gilt es zu beachten, dass der Netzwerkserver in diesem Fall eine neue DevAddr, sowie einen neuen AppSKey und NwkSKey generiert.

5.1.2 Überprüfen der Decoder

In diesem Abschnitt soll die korrekte Funktionsweise der JavaScript-Funktionen (Listing A.1 und Listing A.2) für das Decodieren der eingehenden Datenpakete der beiden verwendeten Sensortypen gezeigt werden. Hierfür werden die aktuellen Messwerte der decodierten Datenpakete in der ChirpStack-Applikation mit den Einträgen in der jeweiligen Smartphone-App des Sensors verglichen. Dies ist in Abbildung 5.2 exemplarisch für die Messwerte des *Milesight AM103L*-Sensors zu sehen.

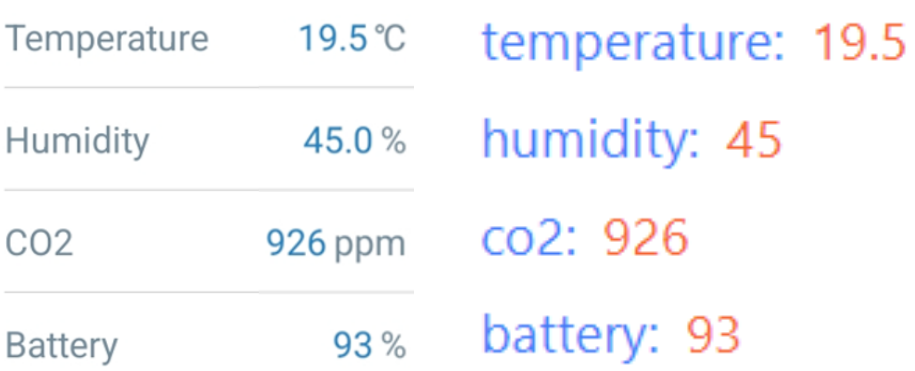

Abbildung 5.2: Vergleich der Messwerte des Milesight AM103L-Sensors in der Smartphone-App (links) und in der ChirpStack-Applikation (rechts)

Zu erkennen ist hierbei, dass die Messwerte aus der ChirpStack-Applikation mit denen der Smartphone-App für den *Milesight AM103L*-Sensor übereinstimmen. Dies kann auch für den *Sentrius RS186*-Sensor bestätigt werden.

Daraus geht hervor, dass die eingesetzten JavaScript-Funktionen die Messwerte der Datenpakete richtig decodieren.

5.2 Funktion der Schnittstelle zur OPC-UA-Anbindung

Als Nächstes wird die korrekte Funktionsweise der in Node-RED implementierten Schnittstelle für die OPC-UA-Anbindung an die CODESYS-Soft-SPS kontrolliert. Hierfür werden die in Abbildung 4.10 dargestellten Werte der Umgebungsvariablen für die in Unterabschnitt 4.3.8 entworfene Verknüpfung zwischen einem Messwert eines LoRaWAN-Sensors und einer Variable der Soft-SPS verwendet.

Nachdem das erste Datenpaket des konkreten Sensors über das dazugehörige MQTT-Topic gesendet und in Node-RED empfangen wurde, erscheinen wie programmiert die wesentlichen Parameter der Verknüpfung in der Status-Anzeige unterhalb des Subflow-Nodes (zu sehen in Abbildung 4.12). Zusätzlich ist in dieser Abbildung zu erkennen, dass das *OpcUa-Client*-Node als Status “value written“ anzeigt. Dadurch wird signalisiert, dass die Variable der Soft-SPS erfolgreich beschrieben wurde. Beim Vergleichen des Wertes der Variable *temperature* der FB-Instanz *Room1* mit dem Temperatur-Messwert des

konkreten Sensors wird festgestellt, dass die Werte übereinstimmen. Außerdem wird erkannt, dass der Variablen-Wert in der Soft-SPS fortlaufend im definierten Sendeintervall des Sensors aktualisiert wird.

Des Weiteren werden zwei Anwendungsfälle betrachtet, die bei der Bedienung der Schnittstelle in Node-RED auftreten können.

Bearbeiten einer bestehenden Verknüpfung

Das Bearbeiten einer bestehenden Verknüpfung kann notwendig sein, wenn eine Variable der Soft-SPS zukünftig durch die Messwerte eines anderen Sensors beschrieben werden soll. Hierfür muss in Node-RED in dem MQTT-Topic, welches über die Umgebungsvariable *topicMQTT* der jeweiligen Gruppe festgelegt wird, die DevEUI des neuen Sensors eingetragen werden.

Nachdem die Änderung übernommen und das erste Datenpaket des neuen Sensors empfangen wurde, erscheint die aktualisierte Status-Anzeige des Subflows mit der neuen DevEUI. Außerdem wird festgestellt, dass nun jeweils der Messwert des neuen Sensors an die gewählte Variable der Soft-SPS bereitgestellt wird.

Hinzufügen einer neuen Verknüpfung

Ein weiterer Anwendungsfall für die Bedienung in Node-RED ist das Hinzufügen einer neuen Verknüpfung zwischen einem LoRaWAN-Messwert und einer Variable der Soft-SPS. Dazu wird in CODESYS zunächst eine neue Instanz des in Unterabschnitt 4.3.5 erstellten FBs angelegt. Außerdem wird die OPC-UA-Funktionalität für die in der Instanz enthaltenen Variablen in der Symbolkonfiguration aktiviert.

In Node-RED wird eine bestehende Gruppe, welche die Verknüpfung realisiert, kopiert und in die Programmieroberfläche eingefügt. Daraufhin werden die Umgebungsvariablen der neu erstellten Gruppe entsprechend für die neu zu erstellende Verknüpfung geändert.

Auch an dieser Stelle wird die korrekte Funktionsweise der Verknüpfung nachgewiesen, indem ein Vergleich des aktuellen Wertes der zu beschreibenden Variable mit dem Messwert des konkreten Sensors durchgeführt wird.

6 Zusammenfassung und Ausblick

In dieser Bachelorarbeit wurde ein Konzept für ein lokales und dezentrales LoRaWAN-Netzwerk mit OPC-UA-Anbindung entworfen. Durch die Implementierung des Konzeptes anhand eines Testaufbaus konnte die korrekte Funktionsweise im Rahmen einer anschließenden Validierng erfolgreich festgestellt werden. Damit konnte das in Abschnitt 1.1 gesetzte Ziel erfolgreich erreicht werden, eine Bereitstellung von LoRaWAN-Messwerten an eine speicherprogrammierbare Steuerung per OPC-UA zu realisieren, wobei die jeweils aktuellen Messwerte fortlaufend in Variablen auf der SPS geschrieben werden.

Zu Beginn der Konzeptphase wurde entschieden, dass ein Industrie-PC als Zielsystem für die zu installierenden Anwendungen genutzt werden soll. Im Anschluss wurden verschiedene Anbieter und deren Ausführungen von LoRaWAN-Netzwerkservern betrachtet, wobei sich für die Verwendung des ChirpStack-Netzwerkservers entschieden wurde. Dieser Netzwerkserver erfüllt die in Unterabschnitt 3.2.3 aufgeführten Anforderungen und kann lokal ohne Internetverbindung betrieben werden. Daraufhin wurden zwei Möglichkeiten für eine Schnittstelle erarbeitet, mit der die Messdaten der LoRaWAN-Sensoren an eine SPS per OPC-UA weitergeleitet werden, wobei sich hier für die Verwendung von Node-RED entschieden wurde.

In der Implementierungsphase wurde als Erstes das Debian Linux-Betriebssystem auf dem Industrie-PC installiert. Im Anschluss wurde der Netzwerk- und Applikationsserver von ChirpStack, sowie das gesamte LoRaWAN-Netzwerk eingerichtet. Daraufhin wurde die Schnittstelle in Node-RED erstellt. Hierbei werden die Datenpakete der Sensoren zunächst per MQTT von ChirpStack aus an Node-RED gesendet, wo das Herausfiltern der Messwerte durchgeführt wird. An dieser Stelle wurde festgelegt, dass für jede Verknüpfung zwischen einem Messwert und einer Variable ein eigener Flow in Node-RED erstellt werden soll. Daraufhin werden die Messwerte per OPC-UA an die SPS gesendet. Außerdem wurde eine Bedienmöglichkeit entwickelt, mit der ein Benutzer neue Verknüpfungen

erstellen und bereits vorhandene bearbeiten kann. Jeder Flow wird dabei in einer eigenen Gruppe zusammengefasst. Die benötigten Parameter der einzelnen Verknüpfungen werden durch Umgebungsvariablen der konkreten Gruppe festgelegt.

Bei der Betrachtung der im Rahmen dieser Bachelorarbeit erstellten Problemlösung ergeben sich einige Erweiterungs- und Verbesserungsmöglichkeiten, die in der weiteren Nutzung und Anwendung innerhalb des Projektes sinnvoll sein könnten. Diese werden im Folgenden als Abschluss und Ausblick der Bachelorarbeit erläutert.

- **Implementieren einer Auswertung der momentanen Batteriewerte der LoRaWAN-Sensoren:** Durch das fortlaufende Überprüfen der Batteriewerte kann frühzeitig festgestellt werden, wann die Batterien eines eingesetzten Sensors voraussichtlich ausgetauscht werden müssen. Dementsprechend kann ein ungeplanter Ausfall eines Sensors und ein damit einhergehender Verlust von Messdaten vermieden werden. Außerdem lässt sich ein Wechsel der Batterien im Vorfeld besser planen. Die Auswertung könnte innerhalb der SPS oder in Node-RED erfolgen. Letzteres hätte den Vorteil, dass die Bereitstellung der Batteriewerte für die Auswertung weniger aufwendig ist, da sich der Wert in Node-RED direkt aus dem jeweils gesendeten Datenpaket der Sensoren entnehmen und verarbeiten lässt. Dafür könnte ein separater Flow entwickelt werden, mit dem die Batteriewerte aller Sensoren zentral gesammelt werden könnte.
- **Erstellen einer Kontrolle der Zeitstempel für die LoRaWAN-Datenpakete:** Mithilfe einer Überprüfung der Zeitstempel, die in den empfangenen Datenpaketen enthalten sind, kann die Aktualität der Uplink-Nachrichten sichergestellt werden. Dadurch kann eine unterbrochene Verbindung eines Sensors zum Netzwerk automatisch und frühzeitig erkannt werden. Somit soll sichergestellt werden, dass die der SPS zur Verfügung stehenden und für die Heizkreisregelung verwendeten Messwerte die momentanen Gegebenheiten vor Ort abbilden. Auch hier würde sich eine Umsetzung in Node-RED anbieten.
- **Erstellen einer separaten Übersicht der im Projekt existierenden LoRaWAN-Sensoren und FB-Instanzen, sowie der dazugehörigen Variablen:** Durch eine zentrale Zusammenstellung der Informationen zu den Sensoren und den FB-Instanzen kann einem Benutzer die Verwaltung der Verknüpfungen erleichtert werden. Denkbar wäre eine Erstellung dieser Übersicht direkt in der Programmieroberfläche von Node-RED oder in einer zusätzlichen Datei. An dieser

Stelle könnten auch die Sensoren und FB-Instanzen aufgenommen werden, die zeitweise nicht verwendet werden. Dies ist gerade dann sinnvoll, wenn die Anzahl der Sensoren und Variablen mit der Zeit weiter zunehmen wird.

- **Ergänzen diverser Sicherheitsvorkehrungen:** Hierzu stehen vor allem für die implementierte Schnittstelle zwischen dem ChirpStack-Netzwerkserver und der SPS verschiedene Möglichkeiten zur Verfügung. Einerseits lässt sich ein Passwort für den Mosquitto MQTT-Broker einrichten, mit welchem die Datenübertragung zwischen ChirpStack und Node-RED gesichert werden kann. Außerdem kann der Zugang zur Node-RED-Weboberfläche mit einem Passwort versehen werden, wodurch ein unberechtigter Zugriff auf die programmierten Verknüpfungen verhindert werden kann. Des Weiteren kann die OPC-UA-Kommunikation zwischen Node-RED und der SPS gesichert werden. Hierzu stehen für OPC-UA diverse Verschlüsselungsverfahren wie RSA und AES [39], sowie ein Passwortschutz zur Verfügung, die bei Bedarf anstelle der derzeitigen anonymen und ungeschützten Verbindung genutzt werden können.

Literaturverzeichnis

[1] Almeida, Ivo Bizon Franco de ; Chafii, Marwa ; Nimr, Ahmad ; Fettweis, Gerhard: Alternative Chirp Spread Spectrum Techniques for LPWANs, IEEE, 2021, S. 1846–1855

[2] Bentem, Arjan van ; Sharp, Cameron: *laird_sentrius_rs1xx_decoder*. https://github.com/SensationalSystems/laird_sentrius_rs1xx_decoder/blob/master/laird.js. – letzter Zugriff: 26.03.2023

[3] Chirpstack: *Architecture*. https://www.chirpstack.io/docs/architecture.html. – letzter Zugriff: 11.03.2023

[4] Chirpstack: *The ChirpStack project*. https://www.chirpstack.io/. – letzter Zugriff: 11.03.2023

[5] Chirpstack: *MQTT-Integration*. https://www.chirpstack.io/docs/chirpstack/integrations/mqtt.html. – letzter Zugriff: 18.03.2023

[6] Chirpstack: *Quickstart Debian / Ubuntu*. https://www.chirpstack.io/docs/getting-started/debian-ubuntu.html. – letzter Zugriff: 24.03.2023

[7] Chirpstack: *Requirements*. https://www.chirpstack.io/docs/chirpstack/requirements.html. – letzter Zugriff: 24.03.2023

[8] CODESYS: *CODESYS Development System V3*. https://store.codesys.com/de/codesys.html. – letzter Zugriff: 13.03.2023

[9] CODESYS: *CODESYS Group*. https://de.codesys.com/. – letzter Zugriff: 20.03.2023

[10] Debian: *Debian bekommen*. https://www.debian.org/distrib/. – letzter Zugriff: 13.03.2023

[11] Debian: *Gründe für den Einsatz von Debian*. https://www.debian.org/intro/why_debian. – letzter Zugriff: 16.03.2023

[12] Ertürk, Mehmet A. ; Aydin, Muhammed A. ; Büyükakkaşlar, Muhammet T. ; Evirgen, Hayrettin: A Survey on LoRaWAN Architecture, Protocol and Technologies. In: *Future Internet* (2019)

[13] Fan, Chun-I ; Zhuang, Er-Shuo ; Karati, Arijit ; Su, Chun-Hui: A Multiple End-Devices Authentication Scheme for LoRaWAN. In: *Electronics* (2022)

[14] Haxhibeqiri, Jetmir ; De Poorter, Eli ; Moerman, Ingrid ; Hoebeke, Jeroen: A survey of LoRaWAN for IoT: From Technology to Application. In: *Sensors* (2018)

[15] ITU: *Information technology - Open Systems Interconnection - Basic Reference Model: The basic model*. https://www.itu.int/rec/T-REC-X.200-199407-I. – letzter Zugriff: 04.03.2023

[16] Laird Connectivity: *Sentrius RG1xx LoRaWAN Gateway + Wi-Fi / Ethernet + Optional LTE (US Only)*. https://www.lairdconnect.com/iot-devices/lorawan-iot-devices/sentrius-rg1xx-lorawan-gateway-wi-fi-ethernet-optional-lte-us-only. – letzter Zugriff: 08.03.2023

[17] Laird Connectivity: *Sentrius RS1xx LoRa-Enabled Sensors*. https://www.lairdconnect.com/iot-devices/lorawan-iot-devices/sentrius-rs1xx-lora-enabled-sensors. – letzter Zugriff: 09.03.2023

[18] Lavric, Alexandru ; Petrariu, Adrian I.: LoRaWAN communication protocol: The new era of IoT. In: *2018 International Conference on Development and Application Systems (DAS)*, IEEE, 2018, S. 74–77

[19] Lehong, Charles ; Isong, Bassey ; Lugayizi, Francis ; Abu-Mahfouz, Adnan M.: A Survey of LoRaWAN Adaptive Data Rate Algorithms for Possible Optimization. In: *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, IEEE, 2020, S. 1–9

[20] Light, Roger: *paho-mqtt - PyPI*. https://pypi.org/project/paho-mqtt/. – letzter Zugriff: 19.03.2023

[21] Linnemann, Marcel ; Sommer, Alexander ; Leufkes, Ralf: *Einsatzpotentiale von LoRaWAN in der Energiewirtschaft*. SpringerVieweg, 2019. – ISBN 978-3-658-26917-3

[22] LoRa Alliance: *About LoRa Alliance®*. https://lora-alliance.org/about-lora-alliance/. – letzter Zugriff: 23.02.2023

[23] LoRa Alliance: *LoRaWAN® Specification v1.0.3*. https://lora-alliance.org/resource_hub/lorawan-specification-v1-0-3/. – letzter Zugriff: 27.02.2023

[24] LoRa Alliance: *RP2-1.0.3 LoRaWAN® Regional Parameters*. https://lora-alliance.org/resource_hub/rp2-1-0-3-lorawan-regional-parameters/. – letzter Zugriff: 23.02.2023

[25] LoRa Alliance: *What is Lorawan®*. https://lora-alliance.org/resource_hub/what-is-lorawan/. – letzter Zugriff: 04.03.2023

[26] lora-wan: *Was ist LoRa?* https://www.lora-wan.de/lora/. – letzter Zugriff: 21.02.2023

[27] Loriot: *Hybrid Network Management System for Massive IoT*. https://loriot.io/index.html#network-management-system. – letzter Zugriff: 15.03.2023

[28] Loriot: *LORIOT - Professional LoRaWAN Public Server*. https://www.loriot.io/professional-public-server.html. – letzter Zugriff: 15.03.2023

[29] Loukil, Slim ; Fourati, Lamia C. ; Nayyar, Anand ; Chee, K.-W.-A.: Analysis of LoRaWAN 1.0 and 1.1 Protocols Security Mechanisms. In: *Sensors* (2022)

[30] Milesight: *Ambience Monitoring Sensor*. https://www.milesight-iot.com/lorawan/sensor/am103/. – letzter Zugriff: 09.03.2023

[31] Milesight: *Semi-Industrial LoRaWAN Gateway*. https://www.milesight-iot.com/lorawan/gateway/ug65. – letzter Zugriff: 10.03.2023

[32] Milesight-IoT: *SensorDecoders*. https://github.com/Milesight-IoT/SensorDecoders/blob/main/AM_Series/AM100_Series/AM100_Chirpstack.js. – letzter Zugriff: 26.03.2023

[33] MQTT.org : *MQTT - The Standard for IoT Messaging*. https://mqtt.org/. – letzter Zugriff: 18.03.2023

[34] myDevices: *Cayenne Low Power Payload*. https://docs.mydevices.com/docs/lorawan/cayenne-lpp. – letzter Zugriff: 17.04.2023

[35] Node-Red: *Connect to an MQTT Broker*. https://cookbook.nodered.org/mqtt/subscribe-to-topic. – letzter Zugriff: 18.03.2023

[36] Node-Red: *Node-red*. https://nodered.org/. – letzter Zugriff: 19.03.2023

[37] Node-Red: *node-red-contrib-opcua*. https://flows.nodered.org/node/node-red-contrib-opcua. – letzter Zugriff: 19.03.2023

[38] Node-Red: *Running on Raspberry Pi*. https://nodered.org/docs/getting-started/raspberrypi. – letzter Zugriff: 13.03.2023

[39] OPC Foundation: *OPC UA: Interoperabilität für Industrie 4.0 und das Internet der Dinge*. https://opcfoundation.org/resources/brochures/. – letzter Zugriff: 06.03.2023

[40] OPC Foundation: *OPC UA Specification*. https://opcfoundation.org/developer-tools/documents/?type=Specification. – letzter Zugriff: 06.03.2023

[41] opcua-asyncio contributors: *FreeOpcUa/opcua-asyncio : OPC UA library for Python >= 3.7*. https://github.com/FreeOpcUa/opcua-asyncio. – letzter Zugriff: 19.03.2023

[42] Qadir, Qahhar M. ; Rashid, Tarik A. ; Al-Salihi, Nawzad K. ; Ismael, Birzo ; Kist, Alexander A. ; Zhang, Zhongwei: Low Power Wide Area Networks: A Survey of Enabling Technologies, Applications and Interoperability Needs. In: *IEEE Access* 6 (2018), S. 77454–77473

[43] RAKwireless: *RAK7268V2/RAK7268CV2 WisGate Edge Lite 2*. https://docs.rakwireless.com/Product-Categories/WisGate/RAK7268-V2/Overview/. – letzter Zugriff: 10.03.2023

[44] Redis: *Redis*. https://redis.io/. – letzter Zugriff: 24.03.2023

[45] Roulet-Dubonnet, Olivier: *asyncua - PyPI*. https://pypi.org/project/asyncua/. – letzter Zugriff: 19.03.2023

[46] Semtech: *An In-depth Look at LoRaWAN® Class A Devices*. https://lora-developers.semtech.com/uploads/documents/files/LoRaWAN_Class_A_Devices_In_Depth_Downloadable.pdf. – letzter Zugriff: 24.02.2023

[47] Semtech: *An In-depth Look at LoRaWAN® Class B Devices*. https://lora-developers.semtech.com/uploads/documents/files/LoRaWAN_

Class_B_Devices_In_Depth_Downloadable.pdf. – letzter Zugriff: 24.02.2023

[48] Semtech: *In-Depth: LoRaWAN® End Device Activation*. https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lorawan-device-activation/device-activation. – letzter Zugriff: 28.02.2023

[49] Semtech: *LoRa® and LoRaWAN®: A Technical Overview*. https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf. – letzter Zugriff: 20.02.2023

[50] Shuttle: *EN01J4 Spezifikation*. https://www.shuttle.eu/de/products/edge/en01j4. – letzter Zugriff: 07.03.2023

[51] The Things Industries: *Deployment Options*. https://www.thethingsindustries.com/deployment/. – letzter Zugriff: 12.03.2023

[52] The Things Industries: *The Things Stack for LoRaWAN*. https://www.thethingsindustries.com/docs. – letzter Zugriff: 14.03.2023

[53] The Things Network: *Device Classes*. https://www.thethingsnetwork.org/docs/lorawan/classes/. – letzter Zugriff: 24.02.2023

[54] The Things Network: *End Device Activation*. https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/. – letzter Zugriff: 02.03.2023

[55] The Things Network: *LoRaWAN Architecture*. https://www.thethingsnetwork.org/docs/lorawan/architecture/. – letzter Zugriff: 05.03.2023

[56] The Things Network: *Spreading Factors*. https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/. – letzter Zugriff: 22.02.2023

[57] Transforma Insights: *Global IoT connections to hit 29.4 billion in 2030*. https://transformainsights.com/news/global-iot-connections-294. – letzter Zugriff: 07.04.2023

A Anhang

A1 Code zum Dekodieren der Uplink-Nachrichten des Sentrius RS186-Sensors

```
/* Basic decoder for Laird RS1xx Sensors, only supporting uplink message
 * type 0x01 as defined in RS1xx LoRa Protocol v2.7.*/
function decodeUplink(input) {
  var messageType = input.bytes[0];

  // All Sensor-to-Server messages have the same options byte format.
  // The options byte is always at byte index 1.
  var options = input.bytes[1];

  var decoded = {
    raw: input.bytes,
    f_port: input.f_port,
    messageType: messageType,
    options: options,
    sensorRequestForServerTime: (options & 1<<0) > 0,
    sensorConfigurationError: (options & 1<<1) > 0,
    sensorAlarmFlag: (options & 1<<2) > 0,
    sensorResetFlag: (options & 1<<3) > 0,
    sensorFaultFlag: (options & 1<<4) > 0
  };

  switch (messageType) {
    case 0x01:
      decoded.humidity = input.bytes[3] + input.bytes[2]/100;

      // For temperature, both the integer and fractional parts are signed:
      // a positive value of 27.43 has a fractional portion of 43 and an
      // integer portion 27, and a negative value -15.87 uses -87 and -15.
      // Sign-extend a single byte to 32 bits to make JavaScript understand
      // negative values, by shifting 24 bits to the left, followed by a
```

```
      // sign-propagating right shift of the same number of bits:
      decoded.temperature = (input.bytes[5]<<24>>24) +
    (input.bytes[4]<<24>>24)/100;

      decoded.batteryIndex = input.bytes[6];
      decoded.batteryCapacity = {
        0: '0-5%',
        1: '5-20%',
        2: '20-40%',
        3: '40-60%',
        4: '60-80%',
        5: '80-100%'
      }[decoded.batteryIndex] || 'Unsupported value';

      decoded.alarmMsgCount = input.bytes[7]<<8 | input.bytes[8];
      decoded.backlogMsgCount = input.bytes[9]<<8 | input.bytes[10];
      break;

    default:
      decoded.error = 'Unsupported message type';
  }

  return {data:decoded};
}
```

Listing A.1: Code zum Dekodieren der Uplink-Nachrichten des Sentrius RS186-Sensors in Anlehnung an [2] mit eigenen Anpassungen an ChirpStack V4

A2 Code zum Dekodieren der Uplink-Nachrichten des Milesight AM103L-Sensors

```
/* Payload Decoder for Chirpstack and Milesight network server
* Copyright 2021 Milesight IoT*/
function decodeUplink(input) {
    var decoded = {};

    for (var i = 0; i < input.bytes.length;) {
        var channel_id = input.bytes[i++];
        var channel_type = input.bytes[i++];
        // BATTERY
        if (channel_id === 0x01 && channel_type === 0x75) {
```

```
11              decoded.battery = input.bytes[i];
12              i += 1;
13          }
14          // TEMPERATURE
15          else if (channel_id === 0x03 && channel_type === 0x67) {
16              // °C
17              decoded.temperature = readInt16LE(input.bytes.slice(i, i + 2)) /
                10;
18              i += 2;
19
20          // HUMIDITY
21          else if (channel_id === 0x04 && channel_type === 0x68) {
22              decoded.humidity = input.bytes[i] / 2;
23              i += 1;
24          }
25
26          // CO2
27          else if (channel_id === 0x07 && channel_type === 0x7D) {
28              decoded.co2 = readUInt16LE(input.bytes.slice(i, i + 2));
29              i += 2;
30          }
31      }
32
33      return {data:decoded};
34  }
35
36  /* ****************************************
37   * bytes to number
38   ****************************************/
39  function readUInt16LE(bytes) {
40      var value = (bytes[1] << 8) + bytes[0];
41      return value & 0xffff;
42  }
43
44  function readInt16LE(bytes) {
45      var ref = readUInt16LE(bytes);
46      return ref > 0x7fff ? ref - 0x10000 : ref;
47  }
```

Listing A.2: Code zum Dekodieren der Uplink-Nachrichten des Milesight AM103L-Sensors in Anlehnung an [32] mit eigenen Anpassungen an ChirpStack V4

A3 Darstellung der ChirpStack-V4-Architektur

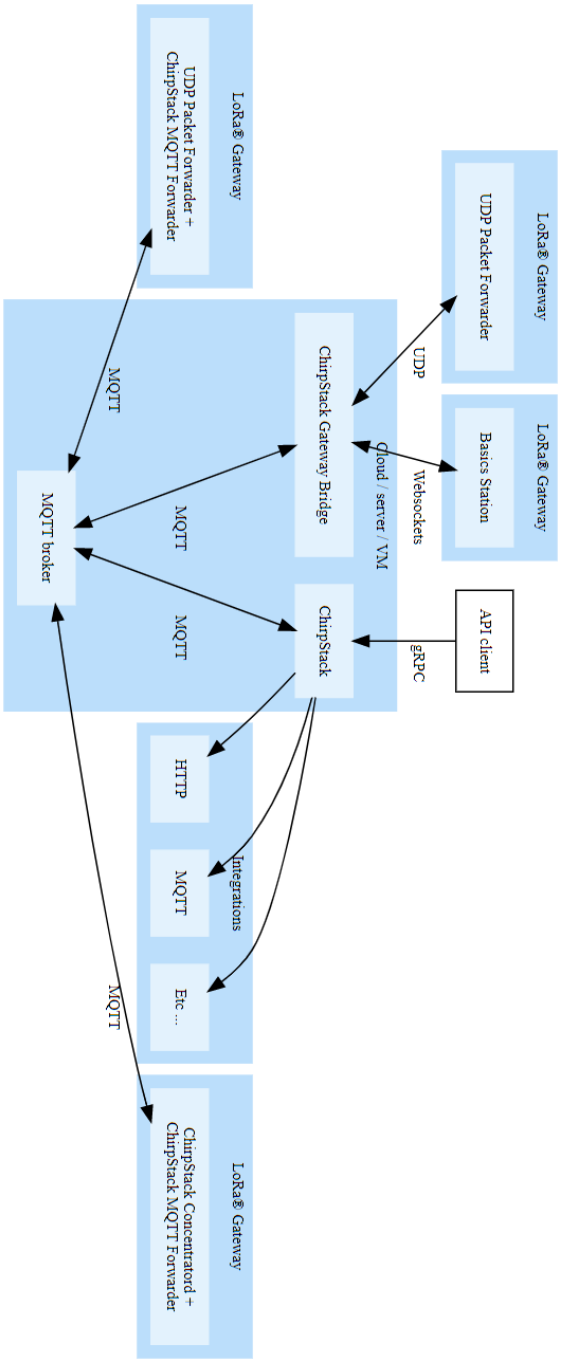

Abbildung A.1: Darstellung der ChirpStack-V4-Architektur und der Kommunikationsverbindungen zwischen den Komponenten (Quelle: ChirpStack [3])

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original