Universität Bremen

Entwurf und Implementierung eines LoRaWAN-basierten Sensornetzes für die quantitative Bewertung einer Waldbrandgefahr

verfasst von

Léon Klick

Thesis zur Erlangung des akademischen Grades
Master of Science (M. Sc.)

1. Gutachter/in: Dr.-Ing. Olaf Bergmann
2. Gutachter/in: Prof. Dr.-Ing. Ute Bormann

Universität Bremen, Deutschland
Fachbereich für Mathematik und Informatik

3. August 2021

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit mit dem Titel „*Entwurf und Implementierung eines LoRaWAN-basierten Sensornetzes für die quantitative Bewertung einer Waldbrandgefahr*“ selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der in den beigefügten Verzeichnissen angegebenen Hilfsmittel nicht bedient habe. Alle Textstellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Die eingereichte elektronische Fassung der Arbeit entspricht der eingereichten schriftlichen Fassung exakt. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Krummesse, 2021-08-03
Ort, Datum

Léon Klick

Danksagung

Mein besonderer Dank gilt Herrn Dr.-Ing. Olaf Bergmann für die Unterstützung während der Bearbeitungszeit. Durch zahlreiche Anregungen und diverse detaillierte Anmerkungen konnte die Qualität der Arbeit zusätzlich verbessert werden.

Des Weiteren möchte ich mich bei Frau Prof. Dr.-Ing. Ute Bormann dafür bedanken, dass sie sich dazu bereiterklärt hat, die Aufgabe der Zweitgutachterin zu übernehmen.

Zusätzlich möchte ich mich bei Frau Marie Rogalla für das sorgfältige Korrekturlesen und die stilistischen Anmerkungen bedanken.

Inhaltsverzeichnis

1 Einleitung

Geräte des Internet of Things (IoT) sind häufig in ihren Ressourcen beschränkt[1]. Diese eingeschränkten Geräte werden auch als Constrained Devices (RFC 7228 [1]) bezeichnet. Constrained Devices werden beispielsweise für den Aufbau von Sensornetzen eingesetzt. In solchen Netzen muss die Möglichkeit bestehen, drahtlos über hunderte Meter, im besten Fall sogar mehrere Kilometer, kommunizieren zu können. Des Weiteren muss die Energieeffizienz im Fokus stehen. Ein Protokoll, das diese Eigenschaften erfüllt, ist dem Oberbegriff Low Power Wide Area Network (LPWAN) zuzuordnen. Innerhalb der Klasse der LPWANs existieren verschiedene Protokolle, die immer mehr an Bedeutung gewinnen und in der Praxis eingesetzt werden. [2]

Ein rapide an Bedeutung gewinnendes LPWAN für große, drahtlose Sensornetze ist das Protokoll Long Range Wide Area Network (LoRaWAN) [3]. LoRaWAN ist im OSI-Modell der Sicherungs- und Vermittlungsschicht zuzuordnen. In der Bitübertragungsschicht wird das Protokoll Long Range (LoRa) [4] eingesetzt. In einem LoRaWAN werden Nachrichten von Knoten versendet, von einem Gateway empfangen und demoduliert. Die demodulierten Daten werden dann durch das Gateway an Backend-Server im Internet weitergeleitet, sodass Clients die Daten abrufen können. Die Sensorknoten werden demnach in Form einer Sterntopologie mit dem Gateway verbunden. Der beschriebene Aufbau ist in Abbildung 1.1 zu sehen.

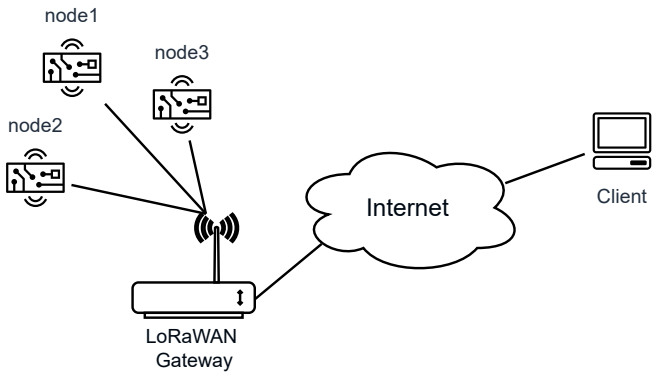

Abbildung 1.1: Vereinfachter Aufbau eines LoRaWANs

[1] Aus Sicht der Hardware sind Constrained Devices vor allem in ihrer Rechenleistung sowie in RAM und ROM/Flash beschränkt und nutzen als Spannungsquelle häufig eine Batterie oder einen Akku mit geringer Kapazität.

1.1 Motivation

LPWANs haben Grenzen, die eine Skalierung eines hochdichten Sensornetzes[2] einschränken können. Die Einschränkung der Skalierbarkeit der Netze ist vor allem auf zwei Ebenen festzustellen:

- **Hardwareebene:** Zwei grundlegende Faktoren auf der Hardwareebene sind die Kosten der Infrastruktur, insbesondere der Sensorknoten per se, sowie der Energieverbrauch der Sensorknoten. Ein Netz, in dem beispielsweise 2500 Sensorknoten auf 1 km^2 Fläche verteilt sind[3], birgt einen hohen finanziellen und organisatorischen Aufwand. Angenommen ein Sensorknoten kostet 100 €, so wäre der finanzielle Aufwand lediglich für die Sensorknoten bereits 250 000 €. Des Weiteren ist ein Wechsel der Batterien oder das Laden der Akkus der Sensorknoten nur in großen Zeitabständen, beispielsweise alle acht Jahre, organisatorisch sinnvoll.
- **Protokollebene:** In einem solchen Netz müssen auch die eingesetzten Kommunikationsprotokolle skalieren. LPWAN-Protokolle haben Grenzen, die berücksichtigt werden müssen. Diese Grenzen sind vor allem in der maximalen Anzahl der beteiligten Knoten, der maximalen Datenrate sowie in rechtlichen Einschränkungen[4] zu finden.

In Bezug auf LoRaWAN sind die Einschränkungen der Skalierbarkeit ebenso festzustellen. In der MAC-Schicht von LoRaWAN wird für den Medienzugriff der Knoten ein ALOHA-ähnliches Verfahren genutzt. Da ALOHA [5] ein konkurrierendes Zugriffsverfahren ist, stellt sich die Frage, ob diese Art des Zugriffs auch für hochdichte Sensornetze skaliert, in denen in geringen Zeitabständen gesendet wird.

Eine weitere Einschränkung ist durch den Einsatz der LoRaWAN-Gateways erkennbar, schließlich ist die erforderliche Verbindung zum Internet nicht immer möglich. Diese Einschränkung ist vor allem im ländlichen Raum festzustellen, da dort eine Mobilfunkabdeckung nicht immer garantiert ist. Dadurch kann LoRaWAN wiederum nicht im gewünschten Umfang eingesetzt werden. Deshalb sollte in der Planung eines hochdichten LoRaWANs in Gegenden mit schlechter Internetanbindung berücksichtigt werden, dass

[2] In einem hochdichten Sensornetz können hunderte bis tausende Sensorknoten auf wenigen Quadratkilometern Fläche verteilt sein.

[3] Bei einer gleichmäßigen Verteilung der 2500 Sensorknoten auf 1 km^2 Fläche wäre ein Abstand von etwa 20 m möglich, was zeigt, dass ein solches Netz in der Praxis nicht unwahrscheinlich ist (beispielsweise für Smart-Metering-Anwendungen).

[4] Besonders im Rahmen lizenzfreier Frequenzbereiche.

möglichst viele Knoten über möglichst wenig Gateways erreichbar sind. Wünschenswert wäre es, in der Planung lediglich ein Gateway einsetzen zu müssen, was wiederum die mögliche Anzahl der Knoten im Netz einschränkt. Es müssen demnach Überlegungen angestellt werden, inwieweit die Single-Hop-Architektur von LoRaWAN erweitert werden kann, um die Anzahl der Gateways zu reduzieren.

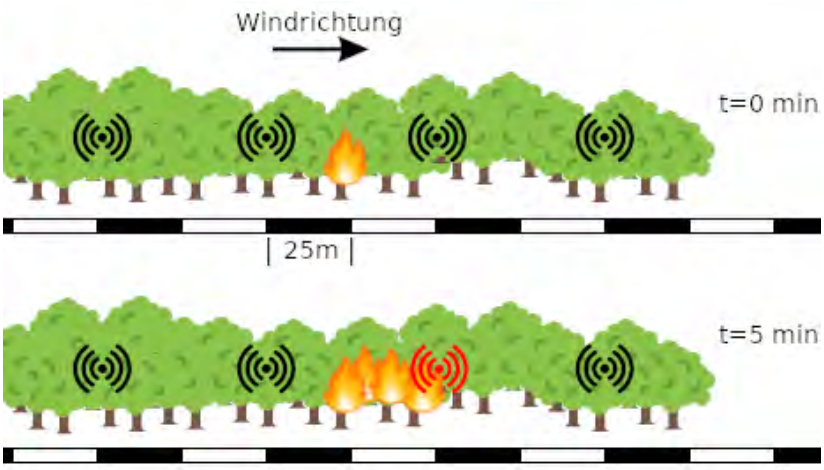

Abbildung 1.2: Verlauf eines Waldbrandes nach Anderson [6]

Ein Anwendungsszenario, in dem hochdichte Sensornetze benötigt werden, ist die Waldbranderkennung. Die Ausbreitung von Waldbränden ist schwer vorherzusagen und verläuft unter Umständen äußerst schnell, weshalb viele Sensoren in geringer Distanz zueinander benötigt werden. Der Abstand zwischen den Sensoren lässt sich approximieren, indem als Grundlage nach Anderson [6] günstige Bedingungen für eine schnelle Ausbreitung des Brandes angenommen werden. Ein Waldbrand kann sich bei einer geringen Feuchtigkeit im Holz, hoher Windgeschwindigkeit und mittlerer Holz-Härteklasse[5] mit etwa 5 m/min ausbreiten[6]. Werden die Sensorknoten mit einem Abstand von beispielsweise 50 m platziert, dann müssten alle 10 min Sensorwerte übermittelt werden, um den Waldbrand unter den genannten Bedingungen mit genügend großer Auflösung verfolgen zu können. In Abbildung 1.2 ist der zuvor beschriebene Abstand von 50 m zwischen den Sensorknoten verdeutlicht. Nach der Entstehung eines Waldbrandes, der zwischen zwei Sensorknoten begonnen hat, würde der Waldbrand unter den zuvor genannten Parametern der Ausbreitung nach 5 min an einem Sensorknoten erkannt werden. Um Flexibilität bei der Bewertung der Waldbrandgefahr zu

[5] Die mittlere Holz-Härte ist ein Kompromiss, um typische Mischwälder abbilden zu können.
[6] Der ursprüngliche Wert ist 16,70 Fuß/min, was umgerechnet 5,09 m/min entspricht.

ermöglichen, sollten die ermittelten Sensorwerte nicht auf den Sensorknoten, sondern im Backend evaluiert werden. Beispielsweise werden für eine Kombination aus einem Temperatursensor, einem Luftfeuchtigkeitssensor sowie einem Gassensor mindestens 4 Byte in den Nutzdaten beansprucht. Die 4 Byte können auf 1 Byte für die Temperatur (inkl. 1 Bit Vorzeichen), 1 Byte für die Luftfeuchtigkeit und 2 Byte für den Messwert eines MQ-2 Gassensors aufgeteilt werden. Dementsprechend müssen Pakete mit einer Größe von mindestens 16 Byte (mindestens 8 Byte LoRaWAN-Header, 4 Byte Prüfsumme [3] und 4 Byte Nutzdaten) in einem zeitlichen Abstand von 10 min pro Sensorknoten versendet werden.

Mit dem zuvor beschriebenen Anwendungsszenario werden die Grenzen von LoRaWAN deutlich. Selten ist in Wäldern flächendeckend Mobilfunk oder ein kabelgebundener Zugang in das Internet verfügbar, sodass nur eine geringe Anzahl Gateways eingesetzt werden kann. Verwendet man für die Aufteilung der Sensorknoten ein quadratisches Raster, platziert die Sensorknoten in der Mitte einer jeden Zelle und legt die Abstände auf Basis einer Von-Neumann-Nachbarschaft fest[7], so lässt sich berechnen, dass für eine Fläche von 25 km^2 bei einem Abstand der Sensorknoten von 50 m insgesamt 10000 Sensorknoten benötigt werden. Bisherige Arbeiten zu diesem Thema verdeutlichen, dass Netze einer solchen Größenordnung die Grenzen von LoRaWAN aufzeigen.

1.2 Verwandte Arbeiten

Mit der Skalierung von LoRaWAN haben sich in der Vergangenheit bereits verschiedene Forschungsarbeiten beschäftigt [7–10].

Varsier und Schwoerer [7] haben mit Hilfe einer Matlab-Simulation untersucht, ab welcher Anzahl von LoRaWAN-Knoten das Netz überlastet ist. Jeder Sensorknoten hat ein Datenpaket mit einer Gesamtgröße von 11 Byte pro Stunde versendet. Des Weiteren wurde die in der LoRaWAN-Spezifikation eingeführte Adaptive Data Rate (ADR) [11] eingesetzt, wodurch der Spreading Factor (SF) abhängig von der Distanz zwischen Sensorknoten und Gateway automatisch angepasst wird. Der SF gibt an, wie stark das Signal über einen Frequenzbereich gespreizt werden soll (siehe Unterabschnitt 2.1.1). Ein hoher SF erhöht zwar die Reichweite, resultiert aber in einer geringeren Datenübertragungsrate und längerer Dauer der Übertragung. Der SF kann zwischen 7 und 12 (SF7–SF12) liegen. ADR verhindert demnach einen zu hohen SF für Sensorknoten, die eine kurze Distanz zu

[7] In der Von-Neumann-Nachbarschaft gelten alle Zellen als Nachbar, deren Kanten direkt an der betrachteten Zelle anliegen.

einem Gateway haben. Mit Hilfe der Simulation wurde ermittelt, dass im urbanen Umfeld bei 1000 Sensorknoten pro Gateway 92,25 % der versendeten Nachrichten empfangen werden konnten. In einer weiteren Simulation wurden 2000 Sensorknoten pro Gateway eingesetzt, wodurch der Anteil erfolgreich empfangener Nachrichten auf 84,10 % gesunken ist. Die Ergebnisse zeigen, dass die Zuverlässigkeit in einem LoRaWAN unter urbanen Bedingungen bereits bei 1000–2000 Sensorknoten pro Gateway beeinträchtigt wird.

Van den Abeele et al. [8] haben ein LoRaWAN-Modul für den Simulator ns3[8] entwickelt, um die Skalierbarkeit unter Betrachtung verschiedener Parameter analysieren zu können. In den Simulationsdurchläufen wurden Pakete mit einer Größe von 21 Byte in einem zeitlichen Abstand von 10 min versendet. Die Anzahl beteiligter Knoten wurde in jedem Simulationsdurchlauf erhöht. Bei einem fest definierten SF von 12 wurden bei 1000 Sensorknoten lediglich etwa 25 % der Pakete erfolgreich empfangen. Den geringen Anteil erfolgreich empfangener Pakete haben die Autoren auf Kollisionen zurückgeführt. Eine variable Anpassung des SFs auf Basis der gemessenen Fehlerrate konnte die Anzahl der erfolgreich empfangenen Pakete bei 1000 Sensorknoten auf etwa 75 % erhöhen. Im Vergleich zu Varsier et al. zeigen die Ergebnisse, dass sowohl der konfigurierte SF als auch die Paketgröße einen Einfluss auf die Zuverlässigkeit des Netzes in Relation zu der Anzahl der Sensorknoten haben. Es ist jedoch zu berücksichtigen, dass sich die Simulationsparameter (z.B. Antennengewinn, Algorithmus zur Anpassung des SFs etc.) unterscheiden.

In einer theoretischen Betrachtung der Grenzen von LoRaWAN haben Adelantado et al. [9] die Performanz auf Basis des Datendurchsatzes für verschieden große LoRaWAN-Zellen[9] berechnet. Die Autoren nutzten als Berechnungsgrundlage, dass die Kodierungen der SFs orthogonal zueinander sind. Dadurch können Nachrichten mit unterschiedlichem SF gleichzeitig empfangen werden. Des Weiteren wurde ein 3-Kanal-Gateway als Grundlage verwendet. Deshalb wurde die maximale Größe des LoRaWANs als die Anzahl der sich überlagernden Netze, bestehend aus den drei verfügbaren Kanälen und den sechs SFs pro Kanal, bestimmt. In den Ergebnissen wird deutlich, dass der Datendurchsatz aufgrund von Kollisionen mit steigender Anzahl der Sensorknoten zunimmt. Adelantado et al. empfehlen eine Optimierung der MAC-Schicht, sodass die Zuverlässigkeit auch in großen LoRaWANs gewährleistet ist.

[8] Website des Simulators ns3: https://www.nsnam.org/ (Abgerufen am 2020-11-18)

[9] Eine LoRaWAN-Zelle besteht aus einem Gateway und einer bestimmten Anzahl von Sensorknoten.

Einige Forscher haben sich in der Vergangenheit bereits mit Sensornetzen in Wäldern auf Basis von LoRaWAN zur Ermittlung einer Waldbrandgefahr beschäftigt [12, 13, 15].

Vega-Rodríguez et al. [12] haben eine LoRaWAN-Architektur vorgestellt, in der Sensorknoten mit Temperatur-, Feuchtigkeits-, CO_2- und Windrichtungssensoren einen Indikator für eine Waldbrandgefahr übermittelten. Im Testaufbau wurden die Sensorwerte von 15 Sensorknoten, aufgeteilt in einem Umkreis um das Gateway von etwa 1,1 km, alle 28 min versendet. Des Weiteren haben die Autoren eine Auflistung der Kosten vorgestellt. Für die Hardware und das Gehäuse wurden pro Sensorknoten 101,01 € ausgegeben[10]. Über die Skalierbarkeit des Netzes wurde keine Aussage getroffen.

Mit Hilfe einer Mesh-Struktur haben Adnan et al. [13] eine LoRaWAN-Architektur zur Waldbranderkennung entwickelt. Durch den Mesh-Ansatz nutzten die Autoren die Möglichkeit, Sensorknoten auch außerhalb der Reichweite des Gateways zu platzieren. Zur Erkennung eines Brandes wurden Luftfeuchtigkeits-, Temperatur- und CO_2-Sensoren eingesetzt. Um Kollisionen innerhalb des Mesh-Netzes zu vermeiden, werden Wartezeiten von etwa 20 s bei vier Sensorknoten empfohlen. Weitere Aspekte zur Skalierung des Netzes und zum Energieverbrauch der Sensorknoten wurden nicht beschrieben.

Rizanov et al. [15] haben in ihrer LoRaWAN-Architektur vor allem den Energieverbrauch der Sensorknoten fokussiert. Deshalb verwendeten die Autoren einen Low-Power Mikrocontroller STM32L082[11]. Als Spannungsquelle diente jedem Sensorknoten ein Akku, der mit Hilfe von Solartechnik geladen wurde. Zur Positionsbestimmung und Zeit-Synchronisation verbauten die Autoren ein GPS-Modul. Sensorknoten wurden in der Architektur in Zonen eingeteilt, wobei jeder Zone ein Gateway zugeordnet wurde. Eine Zusammenfassung der Kosten und eine Betrachtung der Skalierung von LoRaWAN bei einer großen Anzahl von Sensorknoten pro Zone wurde in der Arbeit nicht vorgestellt.

[10] Ein Mengenrabatt ist in der Auflistung nicht einberechnet.

[11] Datenblatt des STM32L082: https://www.st.com/resource/en/datasheet/stm32l082cz.pdf (Abgerufen am 2020-11-18)

1.3 Ziele und Methodik

Das Ziel der Arbeit ist es, die Skalierung von LoRaWAN exemplarisch anhand einer Architektur zur Waldbranderkennung zu evaluieren sowie Parameter und mögliche Erweiterungen zur besseren Skalierung zu entwickeln. Das Ziel teilt sich auf drei Aspekte auf:

1. Ermitteln des Ist-Zustands durch Evaluieren der Skalierbarkeit von LoRaWAN für hochdichte Sensornetze.
2. Entwickeln geeigneter LoRaWAN-Parameter und möglicher Erweiterungen in Bezug auf den Medienzugriff, wobei eine reduzierte Anzahl verfügbarer Gateways berücksichtigt werden soll.
3. Evaluieren geeigneter Hardware für einen prototypischen Aufbau der Zielarchitektur.

Die Schwierigkeit dieser Arbeit liegt vor allem im Nachweis der Skalierbarkeit des Netzes, da ein Aufbau eines hochdichten Sensornetzes mit tausenden Sensorknoten im Rahmen dieser Arbeit nicht möglich ist. Um die Skalierbarkeit der entwickelten Architektur betrachten zu können, sollen deshalb zwei Ansätze verfolgt werden:

1. Theoretische Betrachtung der entwickelten LoRaWAN-Architektur
2. Praktische Umsetzung der LoRaWAN-Architektur in Form eines Versuchaufbaus mit einer reduzierten Anzahl prototypischer Sensorknoten

Mit der theoretischen Betrachtung soll zunächst gezeigt werden, inwieweit die Architektur auf Basis der vorgestellten Parameter und möglichen Erweiterungen für hochdichte Sensornetze skaliert. Die theoretische Betrachtung soll im Vergleich zum zuvor ermittelten Ist-Zustand durchgeführt werden und kann beispielsweise mathematisch oder in Form einer Computersimulation stattfinden. Die praktische Umsetzung soll wiederum zeigen, dass die vorgestellte Architektur in realen Sensornetzen umgesetzt werden kann. Des Weiteren soll mit der praktischen Umsetzung nachgewiesen werden, dass die evaluierte Hardware sowohl finanziell als auch in Hinblick auf den Energieverbrauch skaliert.

Für die Ermittlung des Ist-Zustands und der Betrachtung der Skalierbarkeit der entwickelten Architektur eignet sich, wie zuvor erwähnt, eine Computersimulation. Für diesen Zweck kann das von Van den Abeele et al. [8] vorgestellte LoRaWAN-Modul für den Simulator ns3 verwendet werden. Um das Modul verwenden zu können, muss zunächst überprüft werden, inwieweit realistische Abbildungen hochdichter Sensornetze darin möglich sind. Deshalb ist das Bewerten des ns3-Moduls in Hinblick auf hochdichte Sensornetze ein weiterer Bestandteil dieser Arbeit. Auf Basis der Ergebnisse dieser Bewertung

soll festgelegt werden, ob eine Computersimulation oder ein mathematischer Ansatz für einen abschließenden Vergleich zwischen dem Ist-Zustand und der entwickelten Architektur in Bezug auf die Skalierbarkeit stattfinden soll.

2 Grundlagen

In diesem Kapitel sollen die notwendigen Grundlagen für diese Arbeit gelegt werden. Zunächst werden in den ersten zwei Abschnitten die technischen Details der im Rahmen dieser Arbeit eingesetzten Protokolle LoRa und LoRaWAN vorgestellt. Im Anschluss wird der Simulator ns3 genauer betrachtet. Der Simulator ns3 soll im Zusammenhang mit dem ns3-LoRaWAN-Modul von Van den Abeele et al. [8] für das Überprüfen der Skalierbarkeit von LoRaWAN in Wäldern evaluiert werden. Daher ist es notwendig, einige Details zum Simulator ns3 zu vermitteln. Zum Ende des Kapitels wird erläutert, wie ein Waldbrand mit Hilfe von Sensoren festgestellt und die begünstigte Ausbreitung eines Waldbrandes ermittelt werden kann.

2.1 Protokoll LoRa

In der Bitübertragungsschicht des LoRaWAN-Protokollstapels befindet sich das proprietäre Protokoll Long Range (LoRa) [4]. LoRa wurde von Semtech entwickelt und nutzt als Basis das Modulationsverfahren Chirp Spread Spectrum (CSS). Details zur Umsetzung von LoRa sind von Semtech bisher nicht veröffentlicht worden. Aus diesem Grund wurde in der Vergangenheit Reverse Engineering von LoRa betrieben (siehe beispielsweise [16, 17]). Zwar existieren Dokumente von Semtech über Grundlagen und Implementierungen von LoRa, jedoch sind diese nicht detailliert genug (siehe beispielsweise [4, 18]). Aus diesem Grund wird insbesondere die Arbeit von Knight et al. [16], in der Ergebnisse des Reverse Engineerings von LoRa vorgestellt werden, als Grundlage für diesen Abschnitt verwendet.

2.1.1 Modulationsverfahren in LoRa

CSS ist unter anderem aus dem Standard IEEE 802.15.4a [19] für WPANs mit einer geringen Datenrate bekannt. WPANs werden für kurze Übertragungsstrecken eingesetzt. Der Fokus von LoRa liegt jedoch in der Übertragung über Distanzen von mehreren hundert Metern, bis hin zu mehreren Kilometern. Um dieses Ziel zu erreichen, muss als Kompromiss unter anderem die Datenrate weiter eingeschränkt werden.

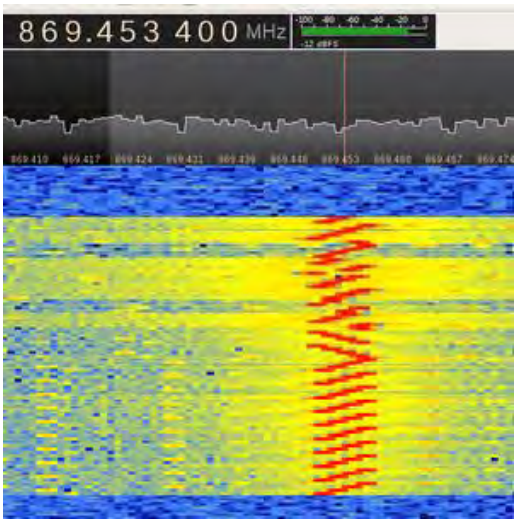

Abbildung 2.1: Chirps einer LoRa-Nachricht[1]

LoRa moduliert Daten, indem kurze Impulse in einem definierten Frequenzbereich aufeinanderfolgend versendet werden. Diese Impulse werden Chirps genannt und sind in Abbildung 2.1 als die diagonal verlaufenden Linien zu erkennen. Ein Chirp ist ein Signal, das die Frequenz in Abhängigkeit zur Zeit entweder erhöht (Up-Chirp) oder verringert (Down-Chirp). Im Kontext von LoRa ändert sich die Frequenz linear in Abhängigkeit zur Zeit. Das Erhöhen bzw. Verringern der Frequenz ist so lange möglich, bis die maximale oder minimale Frequenz erreicht wird. Die maximale oder minimale Frequenz steht im direkten Zusammenhang mit der definierbaren Bandbreite, welche entweder 125 kHz, 250 kHz oder 500 kHz beträgt. Ein LoRa-Signal beginnt zunächst mit einer gewissen Anzahl vollständiger Up-Chirps, gefolgt von zwei vollständigen Down-Chirps, die der Präambel und einem Synchronisationswert entsprechen (siehe die untersten zwölf Chirps in Abbildung 2.1). Nach der Präambel folgen zwei kurze Trennsymbole, woraufhin die Daten-Chirps beginnen. Je nachdem, welcher Wert übermittelt werden soll, startet ein Daten-Chirp bei einer anderen Frequenz. [16] Chirps haben immer die gleiche Länge, sodass ein Chirp beispielsweise beim Erreichen der maximalen Frequenz bei der minimalen Frequenz fortgesetzt wird. Die zur Verfügung stehende Bandbreite wird in Chips aufgeteilt, wobei ein Chip einem Hertz der Bandbreite entspricht. Somit ist eine Bandbreite von 125 kHz in 125000 Chips aufgeteilt. Ein Symbol entspricht in LoRa einem Chirp, wobei die Länge eines Symbols 2^{SF} Chips entspricht. Demnach kann die Länge eines Symbols durch den Spreading Factor (SF) beeinflusst werden. Es können sechs verschie-

[1] Quelle: Bertrik Sikken, https://revspace.nl/File:DecodingLora_Project.jpg (Abgerufen am 2021-04-13)

dene SFs konfiguriert werden, wobei SF7 dem niedrigsten und SF12 dem höchsten SF entspricht. Der SF definiert des Weiteren die Anzahl der Bits, die pro Symbol kodiert werden können. Dementsprechend besteht je nach konfiguriertem SF die Möglichkeit, 7–12 Bit pro Symbol zu kodieren. Es wird deutlich, dass beim Nutzen eines höheren SFs zwar mehr Bit pro Symbol kodiert werden können, jedoch die Länge eines Symbols mit jeder Erhöhung des SFs verdoppelt wird. Aufgrund dessen, dass mehr Zeit pro Symbol in Anspruch genommen wird, ist die Sensitivität des LoRa-Empfängers bei einem höheren SF größer, wobei die Time on Air (TOA) zunimmt. [20] Deshalb ist es mit einem höheren SF möglich, über eine größere Distanz zu senden. Ebenso wird deutlich, dass ein Erhöhen der Bandbreite die Sensitivität des Empfängers verschlechtert, schließlich werden in kürzerer Zeit mehr Daten moduliert. Die Kombination aus dem konfigurierten SF und der verwendeten Bandbreite wird im Kontext von LoRa als Datenrate bezeichnet. Es wird deutlich, dass die verwendete Datenrate die Reichweite von LoRa-Signalen beeinflusst. Es existieren weitere LoRa-Parameter, die einen Einfluss auf die Reichweite haben:

- **Sendeleistung:** Die Sendeleistung beeinflusst, so wie auch der SF und die Bandbreite, die maximale Reichweite der LoRa-Nachricht. Auf den für den europäischen Raum vorgesehenen Frequenzbändern ist eine maximale Sendeleistung von 25 mW (14 dBm) zulässig (siehe EN 300 220-1 [21]). Eine höhere Sendeleistung erhöht zwar die Reichweite, jedoch erhöht sich dadurch auch der Energieverbrauch während der Übertragung. Somit ist der höchste Energieverbrauch mit SF12 und den zuvor beschriebenen 25 mW Sendeleistung festzustellen.
- **Code Rate (CR):** Die CR ermöglicht eine Forward Error Correction (FEC) in LoRa. Die CR basiert auf dem Hamming Code und wird im Format $4/x$ angegeben, wobei für 4 Bit $x-4$ Paritätsbit bereitgestellt werden. Durch die CR können fehlerhafte Nachrichten korrigiert werden, welche mit größerer Distanz zwischen Sender und Empfänger wahrscheinlicher werden. [20] Durch mehr Paritätsbits wird zwar eine bessere Korrektur ermöglicht, jedoch erhöht sich so auch die TOA.

Die unterschiedlichen SFs sind annähernd orthogonal zueinander, sodass parallel auf der gleichen Frequenz gesendet bzw. empfangen werden kann. Einige Arbeiten haben in der Vergangenheit jedoch gezeigt, dass die Orthogonalität nicht perfekt ist (siehe beispielsweise [22]). Eine weitere Möglichkeit, parallel LoRa-Nachrichten senden und empfangen zu können, ist durch das Nutzen verschiedener Frequenzen möglich. Die Parallelität beim Nutzen unterschiedlicher Frequenzen kann mit den im Rahmen dieser Arbeit verwendeten SX1276-LoRa-ICs[2] nur durch Verwenden mehrerer LoRa-ICs ermöglicht werden.

2.1.2 Aufbau eines LoRa-Paketes

In LoRa wird zwischen Uplink- und Downlink-Paketen unterschieden. Uplink-Pakete sind die Pakete, die vom Sender zum Empfänger versendet werden. Downlink-Pakete sind die Pakete, die vom Empfänger insbesondere als Bestätigung versendet werden (siehe Abschnitt 2.2).

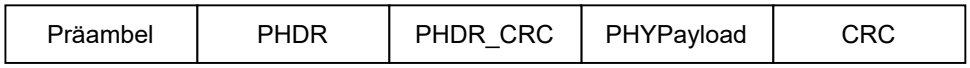

Abbildung 2.2: Aufbau eines LoRa-Uplink-Paketes

In Abbbildung 2.2 ist der Aufbau eines LoRa-Uplink-Paketes zu sehen. Das Paket beinhaltet zunächst die Präambel, welche bereits in Abbildung 2.1 in Form von Chirps zu erkennen war. Darauf folgt der Header des LoRa-Paketes, welcher entweder explizit oder implizit sein kann. Wird ein impliziter Header verwendet, so muss sowohl der PHYPayload immer konstant groß als auch die CR vorab bekannt sein. Daraus folgt, dass beim Einsatz eines impliziten Headers sowohl auf das Feld PHDR als auch auf das Feld PHDR_CRC verzichtet werden kann. Im expliziten Header sind beide Felder vorhanden, wobei im PHDR die Länge des Paketes und die CR eingefügt werden. Das Feld PHYPayload enthält die Nutzdaten des Paketes. Beim Einsatz von LoRaWAN befinden sich im Feld PHYPayload demnach die LoRaWAN-Header und die LoRaWAN-Nutzdaten (siehe Abschnitt 2.2). Abschließend wird im CRC-Feld ein CRC-Wert hinzugefügt. Ein LoRa-Downlink-Paket ist ähnlich zu einem LoRa-Uplink-Paket, jedoch wird auf das CRC-Feld am Ende des Paketes verzichtet. [3]

[2] Datenblatt des SX1276-LoRa-ICs: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE (Abgerufen am 2021-04-14)

2.2 Protokoll LoRaWAN

Ein LoRaWAN ist ein drahtloses Weitverkehrsnetz, das insbesondere für Constrained Devices im Kontext des Internet of Things (IoT) geeignet ist. Zum Zeitpunkt der Arbeit ist LoRaWAN von der LoRa Alliance durch den Standard 1.1 [3] definiert. LoRaWAN-Teilnehmer werden in einer Sterntopologie um ein Gateway angeordnet, was bereits in Abbildung 1.1 deutlich wurde. In einem LoRaWAN muss das Gateway eine Internetanbindung besitzen. Sobald das Gateway ein Paket eines LoRaWAN-Teilnehmers empfangen hat, wird das Paket an Backend-Server im Internet weitergeleitet. Im Backend werden zwei Server-Arten unterschieden: Network Server und Application Server. Häufig werden beide Server-Arten in einem Server integriert. Der Network Server ist für die Weiterleitung der empfangenen Daten an die jeweilige Anwendung, die auf dem Application Server läuft, zuständig. Die Anwendung implementiert die vom Benutzer vorgesehene Verarbeitung der empfangenen Daten. [3]

2.2.1 Regionale Parameter

Das Protokoll LoRaWAN sieht unterschiedliche Sendeparameter für verschiedene Regionen vor. Im Folgenden wird LoRaWAN lediglich für den europäischen Raum und nur in Kombination mit LoRa betrachtet[3]. Für den europäischen Raum werden die lizenzfreien Bänder für sogenannte Short Range Devices (SRD) in einem Frequenzbereich von 863–870 MHz verwendet.

SRD-Band	max. Sendeleistung	max. Duty Cycle
863–865 MHz	25 mW	0,1 %
865–868 MHz	25 mW	1 %
868,0–868,6 MHz	25 mW	1 %
868,7–869,2 MHz	25 mW	0,1 %
869,7–870,0 MHz	25 mW	1 %

Tabelle 2.1: Maximale Parameter für LoRaWAN nach Verfügung 133/2019 der Bundesnetzagentur

In der europäischen Norm EN 300 220-1 [21] vom Europäischen Institut für Telekommunikationsnormen (ETSI) bzw. der in Deutschland nationalen Umsetzung von der Bundesnetzagentur (Verfügung 133/2019 [23]) wird der Frequenzbereich von 863–870 MHz für

[3]Theoretisch können in der Bitübertragungsschicht auch andere Protokolle verwendet werden, die wiederum anderen Parametern zugrunde liegen.

Funkgeräte der SRD-Klasse aufgeteilt. In Tabelle 2.1 ist die Aufteilung nach Verfügung 133/2019 [23] zu sehen. In Tabelle 2.1 wird deutlich, dass zwar die maximale Sendeleistung für alle SRD-Bänder gleich ist, der maximale Duty Cycle jedoch variiert. Der maximale Duty Cycle begrenzt die TOA pro Funknetz-Teilnehmer und wird meist durch eine minimale Wartezeit zwischen zwei Nachrichten eines LoRaWAN-Teilnehmers in einem Band umgesetzt.

Eine Alternative zum maximalen Duty Cycle ist das Umsetzen von Listen Before Talk (LBT) und Adaptive Frequency Agility (AFA) für den Medienzugriff der LoRaWAN-Teilnehmer. Zwar wäre ein Netz, in dem die Teilnehmer nach LBT in Kombination mit AFA senden, weniger eingeschränkt als durch den maximalen Duty Cycle, jedoch ist das Einhalten des Duty Cycles besonders in großen Netzen sinnvoll. Andernfalls besteht die Möglichkeit, dass die betrachteten Frequenzen zu sehr belegt werden, sodass andere Funkgeräte den Übertragungskanal im schlechtesten Fall nicht mehr nutzen könnten. Aus ähnlichen Gründen wird beispielsweise im The Things Network (TTN), einer Umsetzung von LoRaWAN, nach einer sogenannten Fair Use Policy gesendet[4]. Die Fair Use Policy setzt nicht nur den maximalen Duty Cycle voraus, sondern schränkt die maximale Time on Air (TOA) jedes LoRaWAN-Endgerätes auf maximal 30 s pro Tag ein. Im Rahmen dieser Arbeit liegt der Einsatz von LBT in Kombination mit AFA deshalb nicht im Fokus.

2.2.2 Endgeräte in LoRaWAN

In LoRaWAN wird ein ALOHA-ähnliches Zugriffsverfahren verwendet. Es existieren jedoch einige Unterschiede zum ursprünglichen ALOHA [5]. Beispielsweise kann in LoRaWAN auf Empfangsbestätigungen durch Setzen bestimmter Flags im LoRaWAN-Paket verzichtet werden, was im ursprünglichen ALOHA nicht vorgesehen ist. Des Weiteren wird in ALOHA davon ausgegangen, dass Pakete immer gleich lang sind. Dies ist in LoRaWAN nicht der Fall. Trotz dessen ist das Grundkonzept von ALOHA, nämlich dem Senden zu beliebigen Zeitpunkten, in LoRaWAN adaptiert. Dass durch das ALOHA-Verfahren Kollisionen auftreten können, wird demzufolge akzeptiert.

In LoRaWAN werden drei Endgeräte-Klassen unterschieden (vgl. [3], S. 10):

- **Klasse A:** Endgeräte der Klasse A müssen eine bidirektionale Kommunikation ermöglichen und, sofern in den Paketen eine Bestätigung gefordert, zwei Emp-

[4] Siehe Fair Use Policy des TTNs: https://www.thethingsnetwork.org/docs/lorawan/duty-cycle/index.html#fair-use-policy (Abgerufen am 2021-04-03)

fangsfenster nach dem Versenden öffnen. Der Grund für zwei Empfangsfenster ist, dass eine höhere Flexibilität für die Bestätigung ermöglicht wird. Eine Bestätigung wird nur in einem der beiden Empfangsfenster erwartet.

- **Klasse B:** Endgeräte der Klasse B öffnen zusätzlich Empfangsfenster zu definierten Zeitpunkten, um Beacons vom Gateway zu empfangen.
- **Klasse C:** Endgeräte der Klasse C haben nahezu dauerhaft geöffnete Empfangsfenster und schließen diese lediglich, wenn gesendet wird.

Aus der vorherigen Beschreibung wird deutlich, dass Endgeräte der Klasse A den geringsten Energieverbrauch aufweisen. Soll der Energieverbrauch minimal gehalten werden, sollte demnach auf Empfangsfenster bei den Endgeräten und somit auf bestätigte Nachrichten verzichtet werden.

2.2.3 Verschlüsselung, Datenintegrität und Authentifizierung

In einem LoRaWAN wird eine Ende-zu-Ende-Verschlüsselung zwischen den LoRaWAN-Teilnehmern und sowohl dem Network Server als auch der Anwendung aufgebaut. Die Ende-zu-Ende-Verschlüsselung wird auf Basis von AES mit Hilfe des in IEEE 802.15.4-2006 Annex B [24] beschriebenen Verfahrens CCM* durchgeführt, wobei die Länge der Schlüssel 128 Bit beträgt ([3], S. 25). Neben der Verschlüsselung der LoRaWAN-Nutzdaten wird zum Sichern der Integrität der Nachrichten ein Message Integrity Code (MIC) hinzugefügt. Der MIC wird aus zwei Teil-MICs zusammengesetzt, wobei zwei Blöcke des Paketes mit unterschiedlichen Sitzungsschlüsseln nach RFC 4493 [25] berechnet werden. Der Grund für die Aufteilung des MICs liegt in der Möglichkeit, dass beim Empfang eines Paketes aus einem anderen LoRaWAN ein Network Server noch vor der Weiterleitung in das vorgesehene LoRaWAN die Hälfte des MICs verifizieren kann. Für diesen Zweck ist es notwendig, dass die unterschiedlichen LoRaWANs eine Vereinbarung untereinander treffen. Dieses Prinzip wird in LoRaWAN Roaming genannt ([3], S. 50).

Insgesamt werden vier Sitzungsschlüssel pro LoRaWAN-Teilnehmer benötigt. Zunächst wird zwischen dem Teilnehmer und dem Network Server eine Ende-zu-Ende-Verschlüsselung mit Hilfe des *NwkSEncKey* (Network Session Encryption Key) etabliert. Wie zuvor beschrieben, wird zum Gewährleisten der Integrität des Paketes der *FNwkSIntKey* (Forward Network Session Integrity Key) und der *SNwkSIntKey* (Serving Network Session Integrity Key) verwendet. Die Ende-zu-Ende-Verschlüsselung zur Anwendung wird durch den *AppSKey* (Application Session Key) ermöglicht.

Die zuvor erwähnten Sitzungsschlüssel werden abhängig von der Methode, mit der ein LoRaWAN-Teilnehmer dem Netz beitritt, generiert. Der LoRaWAN-Standard [3] sieht zwei Aktivierungsverfahren vor, mit denen ein LoRaWAN-Teilnehmer dem Netz beitreten kann:

- **Over The Air Activation (OTAA):** In OTAA existieren zwei Basis-Schlüssel (*AppKey*, *NwkKey*), mit Hilfe derer die zuvor beschriebenen Sitzungsschlüssel (*NwkSEncKey*, *FNwkSIntKey*, *SNwkSIntKey*, *AppSKey*) bei jeder Aktivierung neu berechnet werden. Der Ablauf des Beitritts wird als Join-Procedure bezeichnet. Eine Join-Procedure findet statt, wenn keine Sitzungsinformationen vorliegen, wie beispielsweise nach einem Neustart eines LoRaWAN-Gerätes.
- **Activation By Personalization (ABP):** In ABP werden die zuvor beschriebenen Sitzungsschlüssel im Speicher des LoRaWAN-Teilnehmers hinterlegt und sind über die gesamte Laufzeit konstant. Aus diesem Grund wird empfohlen, OTAA als Aktivierungsverfahren zu bevorzugen ([3], S. 64).

2.2.4 Adaptive Data Rate (ADR)

Die Konfiguration eines festen SFs bringt einige Nachteile mit sich. Wie in Unterabschnitt 2.1.1 beschrieben, resultiert ein hoher SF in einer hohen TOA und einem höheren Energieverbrauch. Nicht zu vernachlässigen ist ebenso die längere Nutzung des Frequenzbereichs pro Übertragung. Da in lizenzfreien Frequenzbändern gesendet wird, sollte die Nutzung der jeweiligen Frequenzen für alle Teilnehmer möglichst fair sein. Dementsprechend sollte im Rahmen von LoRaWAN der geringstmögliche SF konfiguriert werden. Jedoch ist es nicht trivial, den bestmöglichen SF bereits bei der Planung des Netzes zu konfigurieren. Deshalb gibt es in LoRaWAN die Möglichkeit, die Datenrate, also die Kombination aus dem SF und der Bandbreite, während der Laufzeit dynamisch anzupassen. Dieses Verfahren wird Adaptive Data Rate (ADR) genannt.

ADR wird vom LoRaWAN-Endgerät durch Setzen eines Bits in einem Feld des LoRaWAN-Headers (siehe Unterabschnitt 2.2.5) initiiert. Erhält der Network Server das Paket, überprüft er, ob das ADR-Bit gesetzt ist. Auf Basis einiger Parameter, wie beispielsweise der Empfangsleistung, entscheidet sich der Network Server für einen geeigneten SF. Der SF wird auch unter Berücksichtigung eines gewissen Puffers gewählt, falls sich die Bedingungen zum Senden verschlechtern sollten, wie beispielsweise bei ungünstigen Witterungsverhältnissen. Danach sendet der Network Server dem LoRaWAN-Endgerät über das Gateway ein Downlink-Paket, in dem eine geeignete Datenrate vorgeschlagen wird. Es wird deutlich, dass für den Einsatz von ADR auf die Empfangsfenster des LoRaWAN-Endgerätes nicht verzichtet werden kann. [11]

2.2.5 Aufbau eines LoRaWAN-Paketes

Ebenso wie ein LoRa-Paket, werden auch LoRaWAN-Pakete in Uplink- und Downlink-Pakete aufgeteilt. Neben Paketen zur Übermittlung von Daten gibt es zusätzliche Pakete für das Aktivierungsverfahren OTAA, das in Unterabschnitt 2.2.3 beschrieben wurde. Die folgende Beschreibung des Aufbaus eines LoRaWAN-Paketes kann mit Hilfe von Abbildung 2.3 nachvollzogen werden.

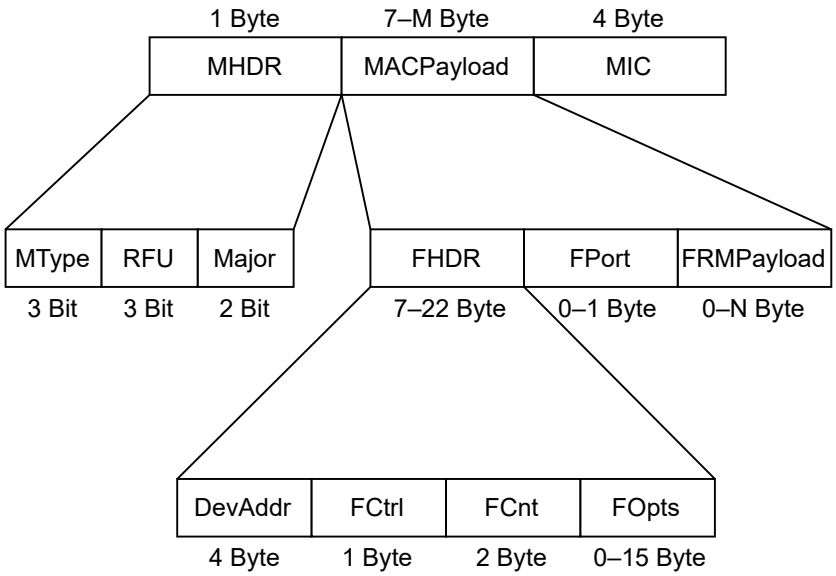

Abbildung 2.3: Aufbau eines LoRaWAN-Uplink-Paketes (vgl. [3], S. 16 ff.)

Zunächst besitzt jedes LoRaWAN-Paket ein MHDR-Feld, in welchem der Typ (MType) des Paketes (z.B. bestätigt, unbestätigt, Join-Anfrage) und die Paket-Version (Major) definiert ist. Drei Bit des Headers sind für zukünftige Änderungen reserviert (RFU). Nach dem MHDR-Feld folgt je nach Paket entweder ein Feld mit den LoRaWAN-Nutzdaten oder OTAA spezifische Felder. Im Folgenden werden die LoRaWAN-Nutzdaten genauer betrachtet. Die LoRaWAN-Nutzdaten teilen sich auf drei Felder auf. Zu Beginn wird ein weiterer Header, der sogenannte Frame Header (FHDR), gefordert. Der FHDR beinhaltet zunächst die Adresse des Endgerätes (DevAddr), die eine Länge von 4 Byte aufweisen muss und im Netz eindeutig sein sollte. Das FCtrl-Feld dient zur Steuerung von LoRaWAN-spezifischen Optionen, wie beispielsweise ADR. Das FCnt-Feld muss eine 2 Byte große Ganzzahl enthalten und entspricht der Anzahl der Pakete, die ein Endgerät bereits versendet hat. Das Endgerät ist für das Inkrementieren des Zählers verantwortlich. Das letzte Feld des Headers ist das FOpts-Feld, welches optional ist und zusätzliche Befehle beinhalten kann, wie beispielsweise das Überprüfen der Verbindung in das Netz (siehe [3], S. 29 ff.). Nach dem Header FHDR folgt das FPort-Feld, durch das in der Anwendung die Art der Nutzdaten unterschieden werden kann. Die möglichen Werte und die Bedeutung der Werte im FPort-Feld sind, bis auf wenige Ausnahmen, anwendungsspezifisch (vgl. [3], S. 24 f.). Der MACPayload wird mit den tatsächlichen Nutzdaten abgeschlossen, die durch den Sitzungsschlüssel *AppSKey* verschlüsselt sind. Das gesamte LoRaWAN-Paket wird mit dem in Unterabschnitt 2.2.3 beschriebenen MIC abgeschlossen.

2.3 Simulator ns3

In Computersimulationen wird angestrebt, einen Prozess oder ein System möglichst realitätsnah für eine gewisse Zeit nachzubilden, um neue Erkenntnisse daraus schließen zu können. Der in C++ entwickelte Simulator ns3 ist speziell für die Simulation von Rechnernetzen ausgelegt. Solche Netz-Simulatoren haben den Vorteil, dass ein Netz mit wenig Aufwand und geringen Kosten frühzeitig evaluiert werden kann. ([26], S. 1) Deshalb werden Netz-Simulatoren auch häufig für das Überprüfen der Skalierbarkeit eines Netzes eingesetzt, was bereits in Abschnitt 1.2 deutlich wurde.

2.3.1 Funktionsweise

Der Simulator ns3 ist ein ereignisorientierter Simulator. Das bedeutet, dass alle zukünftigen und bereits bekannten Ereignisse im Simulator-Kern in einer Liste gespeichert werden. Jedem Ereignis ist ein Zeitstempel zugeordnet, nach welchem die Liste aufsteigend sortiert wird. Dementsprechend ist das erste Ereignis der Liste immer das Ereignis, welches als nächstes berechnet wird. ([26], S. 26)

Listing 2.1 Exemplarischer Quellcode zum Erstellen eines Ereignisses in ns3

```
1  void LoRaWANEndDeviceApplication::ScheduleNextTx ()
2  {
3      if (m_maxBytes == 0 || m_totBytes < m_maxBytes)
4      {
5          Time nextTime (Seconds (m_interval));
6          m_txEvent = Simulator::Schedule (nextTime,
   ↪  &LoRaWANEndDeviceApplication::SendPacket, this);
7      }
8      else
9      {
10         StopApplication ();
11     }
12 }
```

Um ein Ereignis in einer ns3-Simulation erstellen zu können, werden statische Funktionen bereitgestellt. In Listing 2.1 ist ein Quellcode-Ausschnitt aus dem ns3-LoRaWAN-Modul von Van den Abeele et al. [8] zu sehen. In Zeile 6 wird eine der statischen Funktionen des Simulators ns3 aufgerufen, um das Versenden eines Paketes im Simulator-Kern anzumelden. Wie zuvor beschrieben, benötigt jedes Ereignis im Rahmen einer ereignisorientierten Simulation einen Zeitstempel. Aus diesem Grund wird in Zeile 5 ein Zeitpunkt definiert und in Zeile 6 der statischen Funktion als erstes Argument übergeben. Der Zeitpunkt ist im Kontext von Listing 2.1 relativ, was bedeutet, dass die in `m_interval` definierte Zeit auf die bei der Ereignisanmeldung aktuelle Zeit addiert wird.

2.3.2 Aufbau

Der Simulator ns3 beinhaltet verschiedene Grundelemente, die in einem Rechnernetz typischerweise identifiziert werden können. Im Folgenden werden die wichtigsten Grundelemente des Simulators ns3 vorgestellt (vgl. [26], S. 17 f.):

- **Node:** Alle physischen Teilnehmer eines Netzes sind im Kontext von ns3 sogenannte Nodes. In einem LoRaWAN sind also Sensorknoten, Gateways und auch die Backend-Server vom Typ Node.
- **NetDevice:** Ein NetDevice entspricht der Schnittstelle zwischen dem Node und dem Netz. Demnach hat jeder Node auch ein NetDevice, um am Netz teilnehmen zu können. Im Kontext von LoRaWAN wäre beispielsweise der LoRa-IC das NetDevice, welcher die Kommunikation mit anderen LoRaWAN-Geräten ermöglicht.
- **Kanal:** Ein Kanal ist in ns3 das Medium, über das gesendet wird. Dementsprechend sind alle Nodes über einen Kanal miteinander verbunden. In einem ns3-Kanal sind auch die Eigenschaften des Mediums definiert.
- **Pakete:** Um Informationen auszutauschen, werden in Rechnernetzen Pakete versendet. Pakete besitzen Header und meist auch Nutzdaten.

Die zuvor erwähnten Grundelemente bilden eine Schnittstelle für eigene Implementierungen. Soll beispielsweise ein LoRaWAN-Paket in ns3 versendet werden, so muss die Schnittstelle des ns3-Paketes implementiert und die in Unterabschnitt 2.2.5 beschriebenen Felder eines LoRaWAN-Paketes hinzugefügt werden. Der beschriebene Aufbau ermöglicht es, beliebige Rechnernetze mit Hilfe des Simulators ns3 zu simulieren.

2.4 Verfahren zur Waldbranderkennung

Verfahren zur Waldbranderkennung teilen sich auf die tatsächliche Erkennung eines Waldbrandes und die Wahrscheinlichkeit der Entstehung bzw. der begünstigten Ausbreitung des Waldbrandes auf. Um die Wahrscheinlichkeit für die Entstehung bzw. die begünstige Ausbreitung ermitteln zu können, ist es zunächst notwendig, die Initiatoren eines Waldbrandes genauer zu betrachten (vgl. [27]):

- **Naturereignisse:** Waldbrände können beispielsweise durch einen Blitzeinschlag oder durch vulkanische Aktivitäten entstehen.

- **Menschliche Ursachen:** Der Großteil der Waldbrände ist auf menschliches Fehlverhalten oder einer Fehlfunktion von technischen Geräten in der Nähe von Wäldern zurückzuführen[5]. Beispielsweise kann Funkenflug durch Lagerfeuer, eine glimmende Zigarette auf trockenem und leicht brennbarem Untergrund oder eine defekte Stromtrasse einen Waldbrand auslösen.

Die Ausbreitung eines Waldbrandes wird, neben der Dichte des möglichen Brandguts, durch bestimmte Witterungsverhältnisse beeinflusst. Vor allem kann starke Trockenheit die schnelle Ausbreitung eines Waldbrandes begünstigen. Da die meisten Waldbrände durch menschliche Ursachen entstehen, die schwer vorhersagbar sind, wird in vielen Arbeiten versucht, Parameter für eine begünstigte Ausbreitung zu erkennen. [27] Aus diesem Grund wird in einigen Arbeiten eine Kombination aus einer Temperaturmessung der Umgebungsluft und der Feuchtigkeitsmessung des Bodens oder der Umgebungsluft als Indikator für eine schnelle Ausbreitung eines Waldbrandes verwendet. Einen solchen Ansatz verfolgen beispielsweise Chaparro et al. [29]. Wie in Abschnitt 1.2 beschrieben, wurde in der Vergangenheit das Thema der Waldbranderkennung auch mit Hilfe von LoRaWAN-Sensorknoten in verschiedenen Arbeiten betrachtet (siehe [12, 13, 15]). In allen der zuvor erwähnten Arbeiten wurden Temperatur- und Luftfeuchtigkeitssensoren in den Sensorknoten integriert.

Die Betrachtung der Temperatur und der Feuchtigkeit der Umgebungsluft wird überwiegend für das Ermitteln einer Wahrscheinlichkeit eines Waldbrandes bzw. der begünstigten Ausbreitung verwendet [29]. Für das Feststellen des Vorhandenseins eines Waldbrandes werden zusätzliche Informationen benötigt. Die Betrachtung der Temperatur der Umgebungsluft reicht nicht als Indikator für das Vorhandensein eines Waldbrandes, da davon auszugehen ist, dass eine Änderung der Temperatur der Umgebungsluft insbesondere am Anfang eines Brandes in einigen Metern Entfernung kaum messbar ist. Jedoch ist das frühzeitige Erkennen eines Waldbrandes essentiell, um eine schnelle Ausbreitung verhindern zu können, was bereits in Abschnitt 1.1 deutlich wurde. Um das Vorhandensein eines Waldbrandes besser einschätzen zu können, wurden in den Sensorknoten in [12, 13, 15] Gassensoren integriert. Durch die Gassensoren hatten die Autoren die Möglichkeit, auf Rauch in der Umgebungsluft zu schließen. Demnach ermöglichen die Sensorwerte der Sensorknoten sowohl eine Möglichkeit zur Berechnung einer Wahrscheinlichkeit für die begünstige Ausbreitung eines Waldbrandes als auch der tatsächlichen Waldbranderkennung.

[5] Jazebi et al. [27] beziehen sich in ihren Aussagen auf die USA. Im europäischen Raum ist der prozentuale Anteil von durch Menschen verursachten Waldbränden ebenfalls als sehr groß (etwa 95 %) einzustufen [28].

3 LoRaWAN-Simulation

Da die Skalierbarkeit von LoRaWAN in Wäldern betrachtet werden soll, wird als Grundlage das ns3-LoRaWAN-Modul von Van den Abeele et al. [8] verwendet. Für diesen Zweck ist es notwendig, die Eigenschaften der Funkwellenausbreitung und somit des Pfadverlusts von LoRa-Signalen in Wäldern näher zu betrachten. In diesem Kapitel werden deshalb der Aufbau des zuvor erwähnten ns3-LoRaWAN-Moduls beschrieben, ein geeignetes Pfadverlustmodell ermittelt und notwendige Anpassungen im Simulator vorgestellt.

3.1 Funkwellenausbreitung in Wäldern

Um die Funkwellenausbreitung in Wäldern im Rahmen einer Simulation abschätzen zu können, wird ein geeignetes Modell benötigt. Typischerweise wird für Pfadverlustmodelle eine Gleichung bereitgestellt, die die Empfangsleistung in Abhängigkeit zur Distanz zwischen Sender und Empfänger sowie ggf. weiterer Parameter berechnet. Im Anschluss wird verglichen, ob die berechnete Empfangsleistung groß genug ist, sodass der Empfänger die Funksignale demodulieren kann. Solche Modelle können in einfache und komplexe Modelle aufgeteilt werden [30]. Ein einfaches Modell ist beispielsweise die Dämpfung des Signals in der Freiraumausbreitung. Im Modell der Freiraumausbreitung wird lediglich der Pfadverlust, der durch die Distanz zwischen Sender und Empfänger vorliegt, berechnet. Die Betrachtung von solchen einfachen Modellen genügt jedoch nicht für reale Umgebungen. Soll beispielsweise ein urbaner Raum modelliert werden, so kann es bei der Ausbreitung der Funkwellen, neben dem eigentlichen Pfadverlust, zu weiteren Effekten kommen (vgl. [30], S. 290 ff.):

- **Abschattung:** Abschattungen sind vor allem aufgrund großer Gebäude festzustellen. Der Effekt zeigt sich darin, dass sich die Empfangsleistung verändert. Die Änderung der Empfangsleistung kann sich durch Abschattung sowohl verbessern als auch verschlechtern.
- **Mehrwegausbreitung:** Breitet sich die Funkwelle auf unterschiedlichen Wegen aus, so kommt es beim Empfänger zu Interferenzen, die die Empfangsleistung verschlechtern können. Die unterschiedlichen Wege können durch jegliche Hindernisse, wie beispielsweise Gebäude oder Bäume, entstehen.

Aufgrund der zuvor beschriebenen Effekte werden typischerweise komplexe Modelle verwendet. Komplexe Modelle können einer der folgenden Gruppen zugeordnet werden (vgl. [30], S. 302 f.):

- **Empirisch:** Basieren auf Messreihen, die in der betrachteten Umgebung durchgeführt wurden.
- **Semi-Empirisch:** Verfeinerung des rein empirischen Ansatzes durch zusätzliches Verwenden von physikalischen Zusammenhängen.
- **Physikalisch:** Komplexe physikalische Berechnung der betrachteten Umgebung.

Es wird deutlich, dass im Rahmen dieser Arbeit kein physikalisches Modell eingesetzt werden kann, da es den Umfang der Arbeit übersteigt. Des Weiteren ist eine zu starke Abhängigkeit zum Einsatzort festzustellen, sodass bestehende physikalische Modelle zur Ausbreitung von Funkwellen in Wäldern nicht ohne weiteres übertragen werden können.

Eine umfangreiche empirische Studie zum Pfadverlust von Funkwellen in Wäldern haben Azevedo und Santos [31] vorgestellt. In deren Arbeit wird der Einfluss auf die Funkwellenausbreitung durch die Diversität von Wäldern respektive die Abhängigkeit zur eingesetzten Frequenz deutlich. In umfangreichen Tabellen fassen die Autoren die Parameter der Vegetation allgemein, aber auch Parameter zu den verschiedenen Bäumen und Gräsern zusammen, für die Messungen durchgeführt wurden. Darin sind starke Abhängigkeiten zur Dichte, dem Durchmesser aber auch der Art des Baumes festzustellen. Aus den Ergebnissen lässt sich ableiten, dass starke Abhängigkeiten zum gewünschten Einsatzort vorliegen und Messreihen für die betrachtete Umgebung notwendig sind.

Eine empirische Studie haben auch Palaios et al. [32] für einen Wald in Aachen (Nordrhein-Westfalen, Deutschland) durchgeführt. Die Autoren haben auf Basis eigener Messungen bei einer Frequenz von 485 MHz ein geeignetes Pfadverlustmodell für typische europäische Mischwälder gesucht. Für diesen Zweck wurden unterschiedliche Modelle, die vor allem für Wälder oder suburbane Gegenden entwickelt wurden, mit den eigenen Messergebnissen verglichen. Die Autoren kommen zum Ergebnis, dass das Tewari-Modell von Tewari et al. [33] nach Anpassung der variablen Parameter auf Basis der eigenen Messungen den Pfadverlust in europäischen Mischwäldern am besten widerspiegelt.

$$L_B = -27.57 + 20 \cdot log(f) - 20 \cdot log\left(\frac{A \cdot e^{-\alpha d}}{d} + \frac{B}{d^2}\right) \tag{3.1}$$

Tewari et al. [33] haben ihre Studie in indischen Regenwäldern durchgeführt. Es wurden vier Frequenzen (50 MHz, 200 MHz, 500 MHz und 800 MHz) für 15 verschiedene Distanzen

zwischen Sender und Empfänger (zwischen 40 m und 4000 m) getestet. Aus den Ergebnissen haben die Autoren eine Gleichung (siehe Gleichung 3.1) hergeleitet, mit der die Empfangsleistung (L_B) in Abhängigkeit von der Frequenz (f), der Distanz zwischen Sender und Empfänger (d) sowie drei Variablen, die aus den Messwerten abgeleitet werden (A, B, α), berechnet wird. Die Aufteilung der Summanden, in denen A und B vorkommen, hat den Grund, dass dadurch sowohl geringere als auch größere Distanzen in einer Gleichung betrachtet werden. Der Summand, in dem A vorkommt, soll insbesondere Reichweiten unter 400 m abdecken. Der Summand, in dem B vorkommt, soll wiederum den Einfluss größerer Distanzen einbeziehen. [33]

Palaios et al. [32] haben eine Anpassung der variablen Parameter A, B und α des Tewari-Modells auf Basis ihrer eigenen Messungen durchgeführt. Diese Anpassung kann jedoch nicht auf den gegebenen Anwendungsfall übertragen werden. Zunächst wurde eine andere Frequenz und eine andere Sendeleistung eingesetzt, als es in LoRaWAN vorgesehen ist. Des Weiteren ist LoRa aufgrund der Verwendung von CSS als recht robust gegen mögliche Störquellen einzustufen [9]. Aus diesen Gründen ist es notwendig, Messreihen der Empfangsleistung für LoRaWAN in Wäldern zu verwenden. Da zum Zeitpunkt der Thesis keine Studien ermittelt werden konnten, in denen für LoRaWAN genaue Messwerte für europäische Mischwälder in der gewünschten Frequenz von 868 MHz vorliegen, wurden eigene Messungen durchgeführt und die Parameter des Tewari-Modells entsprechend angepasst.

3.1.1 Messdurchführung

Der Ablauf der Messungen ähnelt denen von Tewari et al. [33] und Palaios et al. [32]. Jedoch wurde der Umfang geringer gehalten und keine weiteren Messinstrumente eingesetzt. Für die Messungen wurde ein Printed Circuit Board (PCB) entworfen, auf dem ein ESP32 (siehe Abbildung 3.1, Markierung 1), ein SX1276-LoRa-Modul (siehe Abbildung 3.1, Markierung 2), ein GPS-Modul (siehe Abbildung 3.1, Markierung 3.1) und eine GPS-Antenne (siehe Abbildung 3.1, Markierung 3.2) sowie ein SD-Karten-Modul (siehe Abbildung 3.1, Markierung 4) verlötet werden können. Eine für 868 MHz angepasste Antenne wird an einem SMA-Anschluss befestigt (siehe Abbildung 3.1, Markierung 5). Das PCB hat den Vorteil, dass die Elemente fest verlötet sind und die Leiterbahnen nicht, wie beispielsweise bei einem Breadboard, offenliegen. Somit ist das PCB resistenter gegen Erschütterungen. Da nach jeder Messung der Standort gewechselt wurde, war das PCB ein guter Ansatz für eine zuverlässige Messdurchführung. Der Schaltplan ist in Anhang A.1 zu sehen.

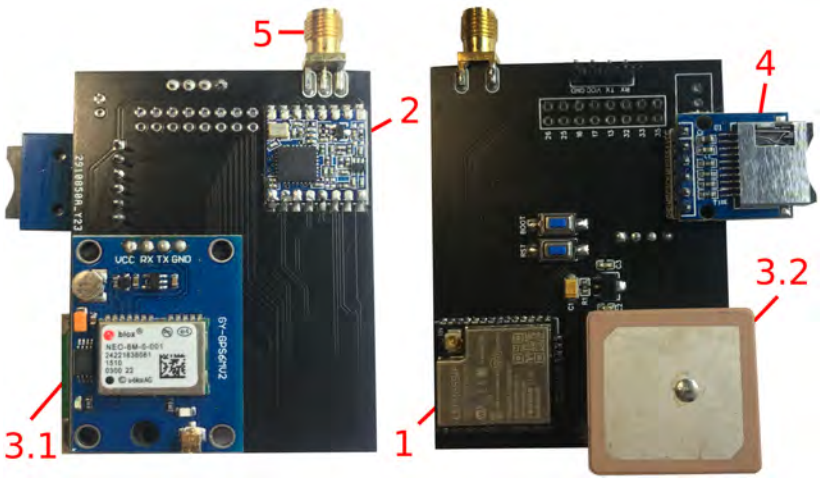

Abbildung 3.1: Vorder- und Rückseite des PCBs zur Reichweitenmessung

Um festzustellen, ob die vorgesehene Antenne tatsächlich für eine Frequenz von 868 MHz einsetzbar ist, wurde die Antenne mit Hilfe eines NanoVNAs überprüft. In Abbildung 3.2 ist das Ergebnis der Messung zu sehen. Auf der rechten Seite ist auf der Ordinate die Rückflussdämpfung und auf der Abszisse die Frequenz in MHz zu sehen. Der markierte Punkt auf der Kurve ist der Punkt mit der besten Rückflussdämpfung, welche bei etwa 855 MHz auftrifft (vgl. „Frequency“ auf der linken Seite in Abbildung 3.2). Das Stehwellenverhältnis lag im Test bei 1,14 (vgl. „VSWR“ auf der linken Seite in Abbildung 3.2). Eine perfekte Antenne hätte ein Stehwellenverhältnis von 1 ([30], S. 87 f.). Auf Basis der ermittelten Werte schien die Antenne geeignet zu sein, um sie für die LoRa-Messungen zu verwenden.

Abbildung 3.2: Screenshot der Software „NanoVNA Saver“[1] nach einer Messung der für die Sender verwendeten Antenne

Die Verwendung des SD-Karten-Moduls ist lediglich für den Empfänger notwendig, da dieser die Empfangsleistung und die ermittelten Koordinaten speichern soll. Für die Messung wurden zwei des in Abbildung 3.1 gezeigten PCBs eingesetzt, wobei ein PCB als Sender und das andere als Empfänger diente. Sowohl der Sender als auch der Empfänger wurden mit Hilfe eines Stativs in einer Höhe von 2 m befestigt. Zunächst wurde für den Sender die GPS-Position ermittelt, da dieser während der Tests immer an der gleichen Position blieb. Der Empfänger hingegen wurde mit jedem Messdurchlauf um eine gewisse Distanz vom Sender entfernt. Für jeden Messpunkt wurden dann vom Sender nacheinander fünf Pakete pro SF (SF7–SF12) versendet. Der Empfänger hat daraufhin alle empfangenen Pakete sowie die ermittelte GPS-Position auf der SD-Karte gespeichert. In der Auswertung wurde dann für jeden aufgezeichneten Messdurchlauf die jeweilige Distanz zum Sender ermittelt. Für die Kommunikation, um beispielsweise die jeweiligen Parameter am Sender bzw. am Empfänger einstellen zu können, wurden Sprechfunkgeräte eingesetzt[2].

3.1.2 Software zur Messdurchführung

Listing 3.1 Quellcode-Ausschnitt des LoRa-Senders

```
while (1) {
    if (gpio_read(btn_sf7) == 0) {
        lora_init_sf(&sf7);
        break;
    }
    [...]
}
size_t i;
for (i = 0; i < 5; i++) {
    send(payload, strlen(payload));
    xtimer_usleep(1000000);
}
```

Sowohl der Sender als auch der Empfänger basieren softwareseitig auf dem Betriebssystem RIOT, da für die zuvor beschriebenen Komponenten bereits Treiber in RIOT existie-

[1] GitHub-Repository von „NanoVNA Saver“: https://github.com/NanoVNA-Saver/nanovna-saver (Abgerufen am 2021-02-01)

[2] Die verwendeten Sprechfunkgeräte haben auf einer Frequenz von 446 MHz gesendet.

ren. Der Ablauf des Senders ist so definiert, dass zunächst auf das Betätigen eines Tasters durch den Benutzer gewartet wird. Je nach betätigtem Taster wird anschließend ein bestimmter SF konfiguriert, wie in Listing 3.1 in den Zeilen 2–5 für SF7 angedeutet. Die Funktion `lora_init_sf`, die in Listing 3.1 in Zeile 3 zu sehen ist, konfiguriert neben dem SF auch eine Bandbreite von 125 kHz sowie eine Code Rate (CR) von 4/5. Der Grund für die Bandbreite und die CR liegt darin, dass die Sensitivität des Empfängers für eine Bandbreite von 125 kHz am höchsten ist und die meisten Arbeiten zu LoRaWAN diese Kombination nutzen (siehe beispielsweise [7–9, 34]). Ebenso wurde die maximal erlaubte Sendeleistung von 14 dBm konfiguriert, um die höchstmögliche Reichweite erzielen zu können. Nach der Konfiguration des SX1276-ICs werden fünf Pakete mit einem Abstand von einer Sekunde versendet, was in Listing 3.1 in den Zeilen 9–12 zu sehen ist.

Listing 3.2 Quellcode-Ausschnitt des LoRa-Empfängers

```
1 if (strncmp(argv[1], "7", 1) == 0) {
2     lora_setup_sf(&sf7);
3 }
4 [...]
5 gps_to_sd();
6 receive();
```

Auf der Empfängerseite findet die eine Konfiguration mit den gleichen Parametern wie bei dem Sender statt, wie in Listing 3.2 in Zeile 1–3 stellvertretend für SF7 zu sehen. Jedoch wird der Empfänger über eine serielle Schnittstelle konfiguriert und nicht, wie zuvor in Listing 3.1 gezeigt, auf Basis von Tastern. Der Grund liegt darin, dass so während der Messungen geprüft werden kann, mit welchem SF welche Reichweite möglich ist. Nach der Konfiguration ermittelt der Empfänger die aktuelle GPS-Position und speichert die ermittelten Koordinaten auf einer SD-Karte (siehe Listing 3.2, Zeile 5). Im Anschluss wartet der Empfänger auf Pakete (siehe Listing 3.2, Zeile 6). Wird ein Paket empfangen, so wird der Paketinhalt sowie der ermittelte Received Signal Strength Indication (RSSI)-Wert ebenfalls auf der SD-Karte gespeichert. Der zuvor genannte Ablauf wird für jeden SF an jedem Messpunkt wiederholt.

3.1.3 Messumgebung

Es sei anzumerken, dass die in Unterabschnitt 3.1.1 beschriebene Messdurchführung Abhängigkeiten zu einigen Umgebungsparametern aufweist. Beispielsweise wurden Höhenunterschiede über die Distanz zwischen Sender und Empfänger nicht weiter betrachtet. Zwar wurde versucht, den Höhenunterschied möglichst gering zu halten, jedoch konnte dies auf einer größeren Distanz nicht vollständig gewährleistet werden.

Des Weiteren wurden die Messungen im Winter durchgeführt. Verschiedene Jahreszeiten können jedoch einen Einfluss auf die Ausbreitung der Funkwellen nehmen, was vor allem durch die Witterungsparameter und dem Laubwerk bedingt ist (vgl. [30], S. 293 f.). Während der durchgeführten Tests wurde eine Temperatur von -2 °C sowie eine relative Luftfeuchtigkeit von 26 % gemessen.

Ein Parameter, der im Rahmen der Messdurchführung Ungenauigkeiten aufweisen kann, ist die GPS-Position. Das GPS-Modul hat im Rahmen eigener Tests bis zu 15 m Abweichungen aufgewiesen, was auch an der verwendeten Antenne liegen kann. Da im Rahmen dieser Arbeit insbesondere die maximale Reichweite von Interesse ist und die Abweichung somit prozentual gering ausfällt, kann die ermittelte Distanz trotz dessen verwendet werden.

Abbildung 3.3: Sender (links) und Empfänger (rechts) befestigt auf einem Stativ in einem Wald in der Nähe von Steinhorst (Schleswig-Holstein, Deutschland)

In Abbildung 3.3 sind der Sender (links) und der Empfänger (rechts) in einem Wald in der Nähe von Steinhorst (Schleswig-Holstein, Deutschland) zu sehen. Dieser Wald wurde exemplarisch ausgewählt, da dort eine dicht bewaldete Umgebung vorzufinden ist, wie im Hintergrund von Abbildung 3.3 zu sehen. Die Bäume des Waldes sind vor allem durch dünnere Äste in einer Höhe von 1–4 m geprägt, wobei es sich um einen Mischwald handelt. Andere Wälder in der näheren Umgebung haben keine ähnlich dichte Bewaldung, weshalb dieser Wald ausgewählt wurde. Wie bereits zuvor beschrieben, wurden die einzelnen SFs am Gehäuse des Senders jeweils mit einem dedizierten Taster ausgewählt, wohingegen am Empfänger eine Verbindung zu einer seriellen Schnittstelle zur Konfiguration bereitgestellt wurde.

3.1.4 Auswertung

Wie bereits in Unterabschnitt 3.1.1 beschrieben, wurde der Sender nicht bewegt und hatte deshalb über die gesamte Messdurchführung eine konstante Position, wobei für den Sender die GPS-Koordinaten `53.736972 N 10.503182 E` ermittelt wurden[3]. Am Empfänger wurden pro Messpunkt für jeden SF-Durchlauf wiederum GPS-Koordinaten ermittelt. Ein stellvertretender Log der ersten Messung für SF7 ist in Listing 3.3 zu sehen. In Zeile 1 von Listing 3.3 sind die ermittelten GPS-Koordinaten aufgelistet. Alle Logs der Messungen sind auf der beigefügten CD hinterlegt.

Listing 3.3 Log des ersten Messpunktes für SF7

```
1 Latitude/Longitude: 53.736912 N 10.502392 E
2 {Payload: "00000" (5 bytes), RSSI: -80}
3 {Payload: "00000" (5 bytes), RSSI: -81}
4 {Payload: "00000" (5 bytes), RSSI: -79}
5 {Payload: "00000" (5 bytes), RSSI: -79}
6 {Payload: "00000" (5 bytes), RSSI: -82}
```

Die Berechnung der Distanz vom Sender zum Empfänger auf Basis der Koordinaten wurde mit Hilfe des Python-Moduls *geopy* durchgeführt[4]. Für die Distanz wurde in *geopy* der *geodesic*-Algorithmus konfiguriert. Der *geodesic*-Algorithmus liefert den kürzesten Weg zwischen zwei Koordinaten, wobei die Erde als perfekter Ellipsoid betrachtet wird [35]. Um mögliche Ungenauigkeiten des GPS-Empfängers bestmöglich auszugleichen, wurde von den ermittelten Distanzen der SF-Durchläufe pro Messpunkt der Median berechnet[5].

Messung-Nr.	1	2	3	4	5	6	7	8	9	10	11	12
Distanz (m)	53	71	103	134	156	281	343	410	461	512	605	696

Tabelle 3.1: Ermittelte Distanzen für alle Messpunkte

In Tabelle 3.1 sind die berechneten Distanzen zwischen Sender und Empfänger für alle Messpunkte aufgelistet. Zwischen Messpunkt 5 und Messpunkt 6 ist im Vergleich zu den anderen Messpunkten ein größerer Abstand festzustellen. Dieser Abstand ist dadurch

[3] Die Koordinaten des Senders wurden zu Beginn der Messungen ermittelt und visuell auf einer Landkarte auf zu starke Abweichungen überprüft.

[4] GitHub-Repository von geopy: https://github.com/geopy/geopy (Abgerufen am 2021-02-01)

[5] Es wurde der Median gewählt, da dieser weniger empfindlich gegen Ausreißer ist.

entstanden, dass das GPS-Modul am Empfänger keine GPS-Koordinaten während der Messung zwischen beiden Punkten ermitteln konnte. Aus diesem Grund wurde dieser Messpunkt aus den Daten entfernt. Da vor allem die maximale Reichweite pro SF von Interesse ist und bei Messpunkt 6 für alle SF Pakete empfangen werden konnten, beeinträchtigt der fehlende Messpunkt die Auswertung nur geringfügig. Die in Tabelle 3.1 aufgelisteten Distanzen können in Abbildung 3.4 nachvollzogen werden. Wie in Abbildung 3.4 zu erkennen ist, wurde der Empfänger während der Tests entlang eines schmalen Pfades bewegt. Am Rand des Pfades beginnt bereits die dichte Bewaldung, wie zuvor in Abbildung 3.3 deutlich wurde.

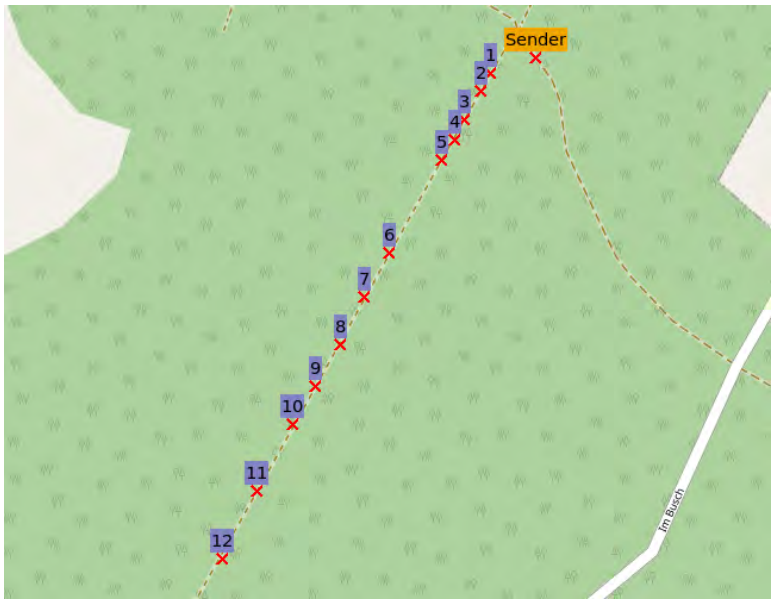

Abbildung 3.4: Messpunkte im Vergleich zur Position des Senders[6]

Mit Hilfe der ermittelten Distanzen in Tabelle 3.1 und den jeweiligen RSSI-Werten war es möglich, die Parameter A, B und α des Tewari-Modells für jeden SF anzupassen. Es sei anzumerken, dass die Messungen besonders auf die Umgebung angepasst sind, in der die Messungen durchgeführt wurden. Eine allgemeine Aussage für den betrachteten Wald oder gar andere Wälder ist somit kaum möglich. Um eine allgemeinere Aussage treffen zu können, müssten wesentlich mehr Messungen an verschiedenen Orten bei unterschiedlichen Witterungsverhältnissen durchgeführt werden. Jedoch ist es aufgrund der zeitlichen Einschränkung dieser Arbeit nicht möglich, umfangreiche Messungen, wie beispielsweise zu unterschiedlichen Jahreszeiten, durchzuführen. Deshalb ist davon auszu-

[6]Der Ausschnitt der Landkarte stammt von OpenStreetMap: https://www.openstreetmap.de/ (Abgerufen am 2021-02-01)

gehen, dass eine Parameteranpassung für die getestete Umgebung überangepasst ist. Diese Messungen sind deshalb als exemplarischer Ablauf zu verstehen.

Für die Parameteranpassung wurde ein Python-Skript entwickelt, welches das LMFIT-Python-Modul verwendet[7]. LMFIT ist ein Python-Modul, das eine Parameteranpassung für selbst definierte Funktionen durchführt. Die Parameteranpassung findet auf Basis der Methode der kleinsten Quadrate (MKQ) statt, sodass die kleinste Summe der Abweichungen der Datenpunkte von der Funktion unter Anbetracht der anzupassenden Parameter ermittelt wird. In Listing 3.4 ist ein Auszug des Ablaufs der Parameteranpassung zu sehen, wobei stellvertretend die Messergebnisse für SF7 verwendet wurden.

Listing 3.4 Exemplarischer Ablauf des LMFIT-Python-Moduls für gegebene Messwerte und ermittelte Distanzen für SF7

```
1 d = [53, 71, 103, 134, 156, 281, 343]
2 rssi = [-80, -95, -95, -96, -103, -115, -120]
3 fmodel = Model(tewari)
4 params = fmodel.make_params(A=0.01, B=0.01, alpha=0.001)
5 result = fmodel.fit(rssi, params, distance=d)
6 print(result.fit_report())
```

In Listing 3.4 werden in den Zeilen 1–2 zunächst die ermittelten RSSI-Werte und die ermittelten Distanzen in zwei Listen gespeichert. In Zeile 1 ist zu sehen, dass mit SF7 bis zu einer Distanz von 343 m Pakete empfangen werden konnten. In Zeile 3 wird ein LMFIT-Modell erstellt, das als Übergabeparameter die in Gleichung 3.1 gezeigte Tewari-Gleichung erhält. In Zeile 4 werden die Parameter initialisiert, die das LMFIT-Modell anpassen soll. Die dort angegebenen Initialwerte wurden unter Betrachtung der von Tewari et al. berechneten Parameter für deren Messreihen ermittelt (vgl. [33], Tabelle II). In Zeile 5 werden dann die Parameter für die Funktion angepasst, sodass in Zeile 6 die Ergebnisse ausgegeben werden können.

[7] Dokumentation von LMFIT: https://lmfit.github.io/lmfit-py (Abgerufen am 2021-02-02)

SF	A	B	α
7	0.1032	2.8290	0.0090
8	0.0869	2.8033	0.0061
9	0.0912	4.8068	0.0106
10	0.0761	2.5582	0.0048
11	0.0915	4.6881	0.0094
12	0.0160	2.7025	0.0026

Tabelle 3.2: Ergebnisse der Parameteranpassung für das Tewari-Modell auf Basis der durchgeführten Messungen

In Tabelle 3.2 sind die Ergebnisse der Parameteranpassung für alle SFs aufgelistet. Diese Parameter können für die Umsetzung des Tewari-Modells in der LoRaWAN-Simulation verwendet werden, um den Pfadverlust für den betrachteten Wald beschreiben zu können. Um einen Überblick über die einzelnen Tewari-Modelle zu erhalten, wurden diese auf Basis der in Tabelle 3.2 aufgelisteten Parameter grafisch umgesetzt. Stellvertretend ist in Abbildung 3.5 ein Auszug für das Tewari-Modell für SF7 gezeigt. Alle weiteren Tewari-Modelle der Parameteranpassungen pro SF sind in Anhang B zu finden.

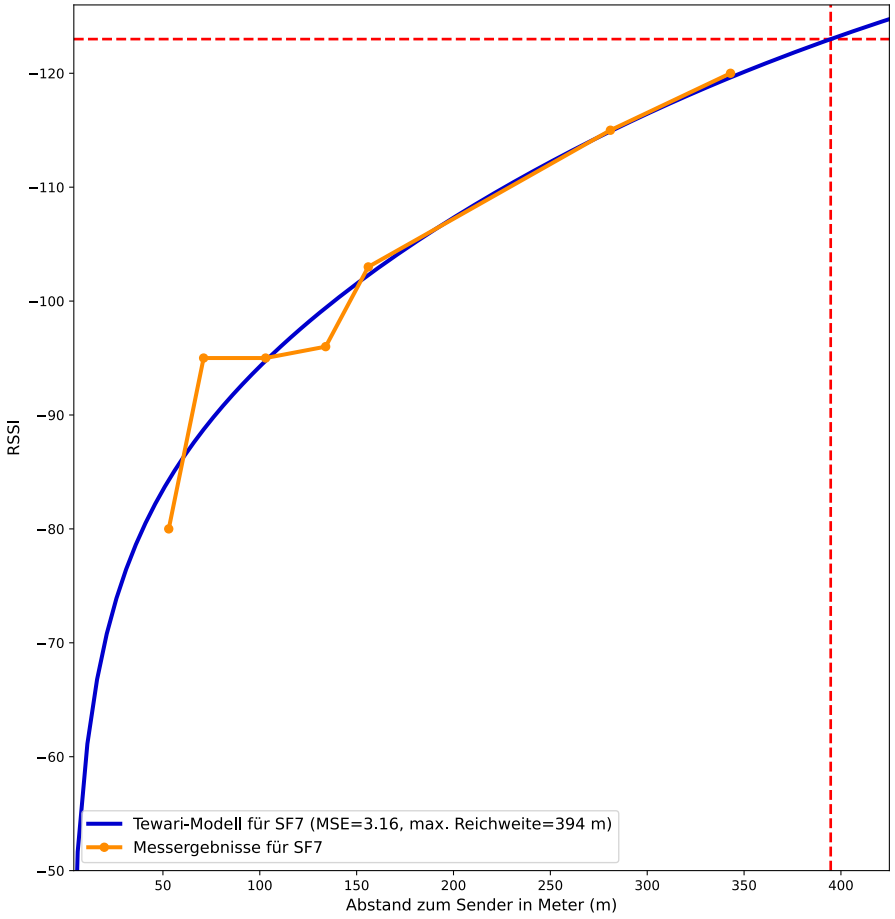

Abbildung 3.5: Auszug der grafischen Repräsentation des Tewari-Modells für SF7 mit angepassten Parametern aus Tabelle 3.2

Die maximale Reichweite wird in Abbildung 3.5 auf Basis des minimalen RSSI-Wertes für SF7 des SX1276-ICs ermittelt und entspricht der gestrichelten horizontalen Linie. Der Schnittpunkt der horizontalen Linie mit der Tewari-Funktion entspricht der maximalen Reichweite für den SF, welcher für SF7 bei 394 m liegt. Eine Auflistung der maximalen Reichweite pro SF ist in Tabelle 3.3 zu sehen.

SF	7	8	9	10	11	12
max. Reichweite (m)	394	512	615	695	763	857

Tabelle 3.3: Maximale Reichweite pro SF auf Basis der durchgeführten Messungen

3.1.5 Zusammenfassung der Messungen

In den vorherigen Abschnitten wurde beschrieben, dass für eine Simulation eines LoRaWANs ein geeignetes Modell für den Pfadverlust von LoRa-Signalen benötigt wird. In der Vergangenheit haben Palaios et al. [32] solche Modelle für Wälder evaluiert. Die Autoren sind zum Ergebnis gekommen, dass das Tewari-Modell [33] mit einer Parameteranpassung für europäische Mischwälder geeignet ist. Da Palaios et al. [32] in ihren Messungen nicht mit LoRa gearbeitet haben, wurden eigene Messungen für die verschiedenen SFs in einem exemplarischen Wald durchgeführt. Da der Umfang der Messungen im Vergleich zu Palaios et al. [32] gering ausfiel, ist eine Überanpassung an die betrachtete Umgebung wahrscheinlich. Wie eingangs beschrieben, soll der Ablauf vor allem einen Ansatz für tatsächliche Umsetzungen in Wäldern liefern. Der Fokus dieser Arbeit liegt insbesondere, wie in Abschnitt 1.3 beschrieben, auf einer prototypischen Umsetzung des Sensornetzes und ist daher nicht für den Produktiveinsatz vorgesehen. Des Weiteren ist davon auszugehen, dass für unterschiedliche Wälder verschiedene Parameter für das Tewari-Modell verwendet werden müssen, schließlich weisen Wälder in anderen Regionen andere Eigenschaften auf.

3.2 LoRaWAN-Simulation in ns3

Wie in Abschnitt 1.3 beschrieben, soll zunächst auf Basis einer theoretischen Betrachtung gezeigt werden, inwieweit eine LoRaWAN-Architektur für den Anwendungsfall einer Waldbranderkennung skaliert. Diese Betrachtung soll mit Hilfe des LoRaWAN-Moduls für den Simulator ns3 von Van den Abeele et al. [8] durchgeführt werden. Deshalb wird im Folgenden der Aufbau und die Funktionsweise des Moduls genauer betrachtet sowie die Implementierung des in Abschnitt 3.1 beschriebenen Pfadverlustmodells vorgestellt.

3.2.1 Aufbau des ns3-Moduls

Im ns3-Modul von Van den Abeele et al. [8] wird zwischen EndDevices und Gateways unterschieden. Ein EndDevice entspricht in der bisherigen Terminologie einem Sensorknoten. Die Bezeichnungen EndDevice und Sensorknoten sind demzufolge im Rahmen dieser Arbeit synonym zueinander[8]. Die EndDevices können im Quellcode zwar für alle Geräteklassen (A, B und C) konfiguriert werden, jedoch ist zum Zeitpunkt der Thesis keine Logik für Klasse B und Klasse C implementiert. Da Geräte der Klasse A wesentlich energieeffizienter sind, wird auf die Betrachtung von Klasse B und Klasse C verzichtet.

Ein LoRaWAN besteht aus mindestens einem EndDevice sowie mindestens einem Gateway. Im LoRaWAN-Modul ist eine Platzierungsstrategie für maximal vier Gateways implementiert. Sowohl ein Gateway als auch ein EndDevice ist aus vier Schichten aufgebaut, was in Abbildung 3.6 zu sehen ist.

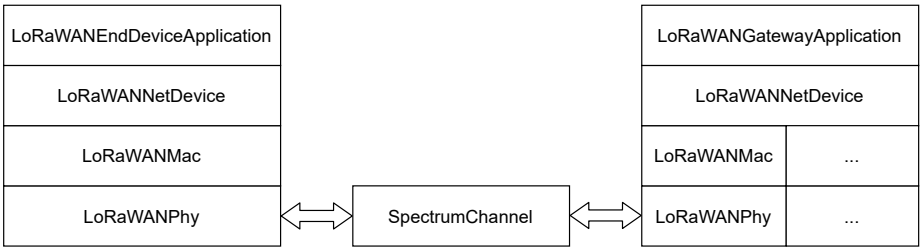

Abbildung 3.6: Aufbau der Komponenten eines LoRaWANs in ns3 (vgl. [8])

Ein EndDevice, das in Abbildung 3.6 auf der linken Seite gezeigt ist, ist mit einem Gateway, das in Abbildung 3.6 auf der rechten Seite gezeigt ist, über einen Kanal *SpectrumChannel* verbunden. Dieser Kanal entspricht somit der Umgebung, in der sich die LoRaWAN-Teilnehmer befinden. Die einzelnen Schichten der LoRaWAN-Teilnehmer erfüllen unterschiedliche Aufgaben, die im Folgenden kurz beschrieben werden:

- **PHY-Schicht:** In der PHY-Schicht werden insbesondere der Empfang und die Demodulation von LoRa-Signalen vorgenommen. Des Weiteren haben Van den Abeele et al. ein Fehlermodell auf Basis von Matlab-Simulationen und der Signal to Noise Ratio (SNR) entwickelt und in der PHY-Schicht eingebettet. Ebenso wird in dieser Schicht ein endlicher Automat implementiert, der die Zustände *RX_ON* (Empfangsmodus aktiviert), *TX_ON* (Sendemodus aktiviert), *BUSY_RX*

[8] Da im Quellcode meist die Bezeichnung EndDevice, im Rahmen des Anwendungsfalls jedoch die Bezeichnung Sensorknoten zutreffender ist, werden im Folgenden beide Begriffe verwendet.

(empfängt Nachricht), *BUSY_TX* (sendet Nachricht) und *TRX_OFF* (Abbruch beim Empfangen oder Senden einer Nachricht) annehmen kann. Ein Gateway kann mehrere PHY-Instanzen enthalten, wie in Abbildung 3.6 auf der rechten Seite angedeutet. Der Grund hierfür liegt in dem Bereitstellen paralleler Empfangspfade, wie es auch in der Praxis üblich ist. Die parallelen Empfangspfade können aus den verschiedenen SF, die als annähernd orthogonal anzusehen sind, sowie verschiedene Frequenzen aufgebaut sein. Demnach ergibt sich die Anzahl der PHY-Instanzen aus der Kombination der zur Verfügung stehenden Frequenzen und konfigurierter SF pro Frequenz.

- **MAC-Schicht:** Die MAC-Schicht ist für das strukturierte Weiterreichen eines zu versendenen Paketes an die PHY-Schicht, das Öffnen von Empfangsfenstern bei bestätigten Paketen, der Umsetzung des ALOHA-ähnlichen Medienzugriffs (siehe Unterabschnitt 2.2.2) sowie die Einhaltung des Duty Cycles verantwortlich. Für die Einhaltung des Duty Cycles wird überprüft, ob das Gerät die Wartezeit im Subband seit dem letzten Senden eingehalten hat. Ein Gateway kann mehrere MAC-Instanzen enthalten, wobei jeder PHY-Instanz auch eine MAC-Instanz zugeordnet ist.

- **NetDevice-Schicht:** Die NetDevice-Schicht entspricht der Schnittstelle zwischen der Anwendungs- und der MAC-Schicht, dient aber auch der Zuordnung der Kombinationen aus PHY- und MAC-Instanzen. Möchte eine Anwendung beispielsweise ein Paket versenden, so interagiert die Anwendung mit der NetDevice-Schicht, welche für die nächsten Schritte verantwortlich ist.

- **Anwendungs-Schicht:** In der Anwendungs-Schicht befindet sich die Logik, die das Senden und Empfangen von Paketen steuert. In der Anwendung des Gateways findet beispielsweise eine Kommunikation mit einem NetworkServer statt, der die durch das Gateway weitergeleiteten Pakete erhält. In der Anwendung der EndDevices wird hingegen das Sendeverhalten definiert.

Für die PHY-Instanzen wird, wie zuvor beschrieben, ein Fehlermodell auf Basis von zuvor durchgeführten Matlab-Simulationen eingesetzt. Dieses Fehlermodell beschreibt sowohl den Pfadverlust als auch die LoRa-interne Interferenz aufgrund nicht perfekt orthogonaler SFs [8]. Jedoch ist der Pfadverlust in Wäldern anders zu beurteilen, als in herkömmlichen urbanen oder suburbanen Gegenden, schließlich wurde für diesen Zweck das Tewari-Modell entwickelt. Deshalb konnte das vordefinierte Fehlermodell nicht übernommen werden und es wurde stattdessen das Tewari-Modell zum Ermitteln

des Pfadverlusts eingebaut. Dieser Schritt bedeutet auch, dass die Empfangsleistung als einziger Indikator dient. Demnach ändert sich auch die Definition einer Kollision von Paketen von Van den Abeele et al. [8] insoweit, dass eine Kollision nur noch dann auftritt, wenn auf der gleichen Frequenz mit dem gleichen SF während der Übertragung einer anderen Nachricht gesendet wird. Diese Definition einer Kollision ist zwar schwächer als die Definition auf Basis des Fehlemodells von Van den Abeele et al. [8], jedoch wird sie häufig eingesetzt, wie beispielsweise im mathematischen LoRaWAN-Modell von Bankov et al. [36]. Die Grenzwerte der Empfangsleistung für die verschiedenen SFs sind dem Datenblatt der SX127X-Reihe von Semtech [37] entnommen.

Es wird deutlich, dass der Aufbau des ns3-Moduls von Van den Abeele et al. [8] alle notwendigen Aspekte abdeckt, um ein LoRaWAN mit den in Abschnitt 1.1 vorgestellten Parametern simulieren zu können. Das LoRaWAN-Modul musste lediglich um das in Abschnitt 3.1 vorgestellte Tewari-Modell erweitert werden, um so die Ausbreitung von LoRa-Signalen in Wäldern in der Simulation berücksichtigen zu können.

3.2.2 Implementierung des Tewari-Modells

Listing 3.5 Funktion zum Berechnen des Pfadverlusts in ns3 auf Basis des Tewari-Modells

```
1 double
2 TewariPropagationLoss::DoCalcRxPower (double txPowerDbm,
  ↪  Ptr<MobilityModel> a, Ptr<MobilityModel> b) const
3 {
4   double distance = a->GetDistanceFrom (b);
5   double pathLossDb = ...
6   return pathLossDb;
7 }
```

In Listing 3.5 ist ein Auszug der Implementierung des Tewari-Modells in ns3 zu sehen. Der Simulator ns3 bietet für diesen Zweck eine Schnittstelle an, sodass alle Pfadverlustmodelle die gleichen Funktionsdefinitionen aufweisen. Der Funktion in Listing 3.5 wird die Sendeleistung `txPowerDbm` sowie die Position der betrachteten LoRaWAN-Teilnehmer übergeben. In Zeile 4 wird dann die Entfernung zwischen beiden Teilnehmern ermittelt, die im Anschluss für die Berechnung des Pfadverlusts auf Basis von Gleichung 3.1 benötigt wird. Zur besseren Übersicht wurde die Gleichung in Listing 3.5 nicht zusätzlich

aufgeführt. Der daraus resultierende Wert wird dann in Zeile 6 dem Aufrufer zurückgegeben. Das Pfadverlustmodell wird dann dem zuvor erwähnten Kanal während der Initialisierungsphase der Simulation zugeordnet, schließlich entspricht der Kanal der Umgebung, in der die LoRaWAN-Teilnehmer senden. Somit liegt einer jeden PHY-Instanz die Möglichkeit zugrunde, den Pfadverlust beim Empfang einer Nachricht zu berechnen. Da jeder SF durch ein eigenes Tewari-Modell repräsentiert wird, werden die Parameter aus Tabelle 3.2 in Abhängigkeit zum SF im Quellcode-Ausschnitt in Listing 3.5 verwendet.

Listing 3.6 Überprüfen der Empfangsleistung im PHY auf Basis der Sensitivität von LoRaWAN-Modulen der SX127X-Reihe

```
1 if ((rxPower < -123. && transmissionDataRateIndex == 5) ||
2     (rxPower < -126. && transmissionDataRateIndex == 4) ||
3     (rxPower < -129. && transmissionDataRateIndex == 3) ||
4     (rxPower < -132. && transmissionDataRateIndex == 2) ||
5     (rxPower < -133. && transmissionDataRateIndex == 1) ||
6     (rxPower < -136. && transmissionDataRateIndex == 0))
7 {
8     drop = true;
9 }
```

In Listing 3.6 ist die Überprüfung, ob die Empfangsleistung für die konfigurierte Datenrate ausreichend groß ist, zu sehen. Die Datenrate entspricht der Kombination aus SF und Bandbreite. Da die Bandbreite mit 125 kHz in der Simulation vordefiniert ist, müssen nur die sechs Fälle SF7–SF12 überprüft werden, wobei SF7 dem `transmissionDataRateIndex` 5 zugeordnet ist. Der in Listing 3.6 gezeigte Ausschnitt ist in der `StartRx`-Funktion der PHY-Klasse vorzufinden. Grundsätzlich wird beim Beginn des Sendens eines Paketes bei allen Teilnehmern des LoRaWANs, die sich im PHY-Zustand *RX_ON* befinden, die `StartRx`-Funktion aufgerufen. Ist die Empfangsleistung zu gering, so wird das Paket im späteren Verlauf verworfen, was in Listing 3.6 in Zeile 8 eingeleitet wird.

3.3 Zusammenfassung

In diesem Kapitel wurde ein geeignetes Pfadverlustmodell für die Funkwellenausbreitung in Wäldern ermittelt und auf Basis von eigenen Messreihen mit LoRa angepasst. Die je nach SF entstandenen Modelle wurden anschließend in den Simulator ns3 implementiert, um so ein LoRaWAN im beschriebenen Wald betrachten zu können. Da im Rahmen dieser Arbeit umfangreiche Messungen aufgrund zeitlicher Einschränkungen nicht möglich waren, konnte nur ein kleiner Abschnitt des Waldes in der Messung berücksichtigt werden. Deshalb ist der beschriebene Ablauf vor allem als exemplarischer Ablauf für eine Umsetzung in der Praxis zu verstehen. Ebenso wurde in diesem Kapitel der Aufbau des LoRaWAN-Moduls von Van den Abeele et al. [8] beschrieben. Dieses Kapitel bildet demnach die Grundlage für das Ermitteln der Skalierbarkeit von LoRaWAN in einem Wald.

4 Skalierbarkeit von LoRaWAN

Wie in Abschnitt 1.2 beschrieben, gab es in der Vergangenheit bereits einige Studien zur Skalierbarkeit von LoRaWAN. Diese Studien haben vor allem urbane Umgebungen untersucht, wie beispielsweise die Studien von Varsier und Schwoerer [7] und Farhad et al. [10]. Zwar gab es in der Vergangenheit bereits einige Arbeiten zu LoRaWAN-basierten Netzen zur Waldbranderkennung, jedoch wurden darin keine Aussagen zur Skalierbarkeit getroffen (vgl. [12, 13, 15]). Da im Rahmen dieser Arbeit die Anzahl der Sensorknoten im drei- bis vierstelligen Bereich liegen soll, ist die Skalierbarkeit ein wichtiger Faktor. Deshalb ist es notwendig, zunächst den Ist-Zustand der Skalierbarkeit eines LoRaWANs in Wäldern mit einer großen Anzahl Sensorknoten zu ermitteln. Für diesen Zweck wurden in Kapitel 3 bereits Grundlagen, wie beispielsweise ein für Wälder geeignetes Pfadverlustmodell, gelegt. Zusammenfassend sollen in diesem Kapitel Simulationsdurchläufe mit Hilfe des ns3-Moduls von Van den Abeele et al. [8] zeigen, inwieweit LoRaWAN für den Einsatz in Wäldern mit einer großen Anzahl Sensorknoten geeignet ist.

4.1 Simulationsparameter

Um die Simulation durchführen zu können, müssen zunächst geeignete Parameter für die Simulation festgelegt werden. Diese Parameter beziehen sich beispielsweise auf das Nutzen verschiedener Frequenzen, um so Interferenzen beim gleichzeitigen Senden verhindern zu können. In diesem Abschnitt werden deshalb die verwendeten Simulationsparameter vorgestellt.

4.1.1 Zeitverhalten

In einem LoRaWAN dürfen Teilnehmer im Prinzip zu beliebigen Zeitpunkten senden. Eine Beschränkung beim Senden ist bezüglich des Zeitverhaltens lediglich im maximalen Duty Cycle zu finden, was bereits in Abschnitt 2.1 beschrieben wurde. Van den Abeele et al. [8] haben in ihrem LoRaWAN-Modul das Sendeverhalten so definiert, dass das erste Paket zu einem zufälligen Zeitpunkt versendet wird. Einzig die Zeit zwischen zwei Paketen eines Teilnehmers kann im Modul definiert werden. Dieses Verhalten ist vergleichbar mit dem initialen Aktivieren der Sensorknoten in einem Wald. Die Sensorknoten starten also zu beliebigen Zeitpunkten, senden ein Paket und warten anschließend die definierte Zeit. Da eine Änderung des beschriebenen Ablaufs bereits einer Erweiterung von LoRaWAN entsprechen würde, wurde das von Van den Abeele et al. [8] definierte Zeitverhalten übernommen. Der Abstand zwischen zwei Paketen wurde auf 10 min festgelegt, wie bereits in Abschnitt 1.1 beschrieben.

4.1.2 Sendeparameter

Wie in Abschnitt 2.1 beschrieben, haben die LoRa-Parameter Spreading Factor (SF), Code Rate (CR), Bandbreite und Sendeleistung einen Einfluss auf die maximale Reichweite und die Time on Air (TOA). Da im Rahmen der Messungen in Abschnitt 3.1 und der anschließenden Parameteranpassung des Tewari-Modells eine Bandbreite von 125 kHz und eine CR von 4/5 gewählt wurde, wurden diese auch im Rahmen der Simulation eingesetzt. Ebenso war die Sendeleistung mit 14 dBm aus den Messreihen und den daraus entwickelten Tewari-Modellen fest definiert. Für die Konfiguration des SFs existieren verschiedene Möglichkeiten. Der einfachste Ansatz besteht darin, allen Sensorknoten den gleichen SF zuzuordnen. In Anbetracht der in Tabelle 3.1 ermittelten Distanzen für SF7–SF12 wird deutlich, dass die in Abschnitt 1.1 beschriebene Fläche von 25 km^2 mit lediglich einem Gateway nicht umsetzbar ist. Deshalb soll der Fokus im Rahmen der Simulation auf die höchstmögliche Reichweite gelegt werden, welche mit SF12 erreicht wird. Ein anderer, im LoRaWAN-Standard [3] festgelegter Ansatz zur Konfiguration des SFs, ist die ADR-Funktion, was bereits in Unterabschnitt 2.2.4 erläutert wurde. Einen ähnlichen Ansatz ermöglichen Van den Abeele et al. [8] im LoRaWAN-Modul. Dort wird jedoch, statt den Ablauf von ADR während der Simulation durchzuführen, der SF initial vor Beginn der Simulation festgelegt. Es erschien sinnvoll, sowohl den Ansatz eines fest konfigurierten SFs für alle Sensorknoten als auch den ADR-ähnlichen Ansatz zu simulieren. Des Weiteren wurde eine Paketgröße von 16 Byte angenommen, wie bereits in Abschnitt 1.1 beschrieben.

4.1.3 Abzudeckende Fläche und Anzahl eingesetzter Gateways

Im LoRaWAN-Modul von Van den Abeele et al. wird eine Kreisfläche mit einem definierbaren Radius für das Netz aufgebaut. Dieses Prinzip war auch für die angestrebte Simulation einsetzbar. Die Fläche ist jedoch abhängig von der Anzahl eingesetzter Gateways und der maximalen Reichweite für den höchsten SF. In Abschnitt 1.1 wurde erläutert, dass es vorteilhaft wäre, lediglich ein Gateway aufgrund der schlechten Internetanbindung in Wäldern einzusetzen. Der Ansatz, lediglich ein Gateway einzusetzen, wurde deshalb in der Simulation fokussiert, wie bereits in Unterabschnitt 4.1.2 angedeutet. Die maximale Reichweite auf Basis der in Abschnitt 3.1 durchgeführten Messungen und dem dadurch angepassten Tewari-Modell für SF12 beträgt 857 m (siehe Abbildung B.7). Beim Einsatz von lediglich einem Gateway folgt aus der maximalen Reichweite für SF12, dass maximal 2,3 km^2 Fläche abgedeckt werden können. In der Praxis ist davon auszugehen, dass zusätzlicher Puffer in der Reichweite pro SF einberechnet werden muss, da die Dichte

des Waldes, die topografischen Gegebenheiten (insbesondere Höhenunterschiede) sowie Witterungsverhältnisse die Reichweite negativ beeinflussen können. Da eine Fläche mit mehr als 2,3 km² außerhalb der Reichweite liegt und somit keine sinnvollen Ergebnisse geliefert hätte, wurde die Fläche von 2,3 km² in der Simulation zugrunde gelegt. Bereits an diesem Aspekt fällt auf, dass das exemplarisch vorgestellte Netz in Abschnitt 1.1 mit einer Fläche von 25 km² in der beschriebenen Konfiguration nicht umsetzbar ist.

4.1.4 Nutzen verschiedener Frequenzen

Im angestrebten Netz soll, wie zuvor beschrieben, eine große Anzahl Sensorknoten eingesetzt werden. Deshalb ist es nicht unüblich, verschiedene Frequenzen für die Sensorknoten zu konfigurieren. Van den Abeele et al. [8] nutzen für diesen Zweck eine zufällige Zuordnung von Frequenzen zu den Sensorknoten. Dieser Ansatz erscheint praxistauglich, weshalb die zufällige Zuordnung auch im Rahmen der Simulation eingesetzt wurde. Jedoch ist die Anzahl der Frequenzen durch die maximale Anzahl paralleler Empfangspfade des Gateways eingeschränkt, was bereits in Unterabschnitt 2.1.1 deutlich wurde. Aus diesem Grund wurde angenommen, dass maximal vier parallele Empfangspfade möglich sind und deshalb maximal vier Frequenzen zufällig auf die Sensorknoten verteilt werden können.

4.1.5 Anzahl eingesetzter Sensorknoten

Wie in Unterabschnitt 4.1.4 beschrieben, soll die maximale Reichweite mit Hilfe von SF12 erreicht werden. Mit SF12 können etwa 2,3 km² um das Gateway abgedeckt werden. Die Anzahl der Sensorknoten wurde aus der gegebenen Fläche und der in Abschnitt 1.1 beschriebenen Fläche von 25 km² und den 10000 Sensorknoten abgeleitet. Die mit SF12 abdeckbare Fläche von 2,3 km² entspricht 10,87 % der 25 km², sodass der gleiche prozentuale Anteil für die Sensorknoten eingesetzt wurde, was 1087 Sensorknoten entspricht[1].

[1] Da in der Simulation eine Kreisfläche zugrunde gelegt wurde, ist der Abstand zwischen den Sensorknoten nicht mehr gleich dem in Abschnitt 1.1 beschriebenen Abstand. Die Betrachtung des prozentualen Anteils ist für die Simulation aber ausreichend genau und wäre in der Praxis ebenfalls nicht exakt einhaltbar.

4.2 Messen der Skalierbarkeit

Um die Simulationsdurchläufe untereinander vergleichen und die Skalierbarkeit bezüglich der Größe des Netzes einordnen zu können, wurde eine Kennzahl benötigt. Des Weiteren besteht mit Hilfe der Kennzahl die Möglichkeit, eventuelle Verbesserungen mit den in diesem Kapitel vorgestellten Simulationen vergleichen zu können. Die Kennzahl muss widerspiegeln, wie gut das LoRaWAN bezüglich der Fläche und der Anzahl der Sensorknoten skaliert. Für diesen Zweck wurde die Packet Delivery Ratio (PDR) genutzt. Die PDR gibt an, wie viele der von den Sensorknoten versendeten Pakete erfolgreich am Gateway empfangen wurden. Einfluss auf die PDR haben Interferenzen, die durch das Nutzen des gleichen SFs auf der gleichen Frequenz zu einem gleichen Zeitpunkt während des Sendens von mindestens zwei Paketen entstehen. Ebenso wird die PDR negativ beeinflusst, wenn sich Sensorknoten außerhalb der Reichweite befinden. Dieses Problem trat jedoch in der folgenden Simulation nicht auf, da in Unterabschnitt 4.1.2 eine Reichweite größer der maximalen Reichweite von SF12 ausgeschlossen wurde.

4.3 Simulationsdurchläufe

Im ersten Schritt wurde simuliert, welche PDR erreicht werden kann, wenn SF12 für alle Sensorknoten fest konfiguriert wird. In Listing 4.1 ist das Ergebnis für den genannten Durchlauf zu sehen. In der Simulation wurde mit Hilfe von Callbacks aufgezeichnet, wie viele Pakete erfolgreich in der MAC-Schicht des Gateways empfangen werden konnten (siehe Listing 4.1, Zeile 2). Im Vergleich wurde ebenso aufgezeichnet, wie viele Pakete von den Sensorknoten versendet wurden (siehe Listing 4.1, Zeile 3). Aus den zuvor genannten Zahlen lässt sich schließlich die PDR ermitteln, die für diesen Durchlauf 0,22 betragen hat. Daraus folgt, dass lediglich 22 % der versendeten Pakete auch am Gateway empfangen werden konnten. Dies lässt sich zusätzlich überprüfen, da die Anzahl der verworfenen Pakete in der PHY-Schicht aufgezeichnet wurde (siehe Listing 4.1, Zeile 7). Der Grund für die Aufzeichnung der Anzahl verworfener Pakete in der PHY-Schicht liegt darin, dass nur in der PHY-Schicht Interferenzen bemerkt werden können.

Listing 4.1 Ergebnis der Simulation für einen fest konfigurierten Spreading Factor (SF) von 12

```
PDR STATS:
nPackets_macRx=1428
nPackets_macTx=6522
PDR=0.218951

ADDITIONAL STATS:
nPackets_phyRxDrop=5094
```

Die geringe PDR in Listing 4.1 ist vermutlich durch die hohe TOA für SF12 zu erklären. Dadurch wäre die Wahrscheinlichkeit für Interferenzen deutlich höher. Um dies nachweisen zu können, wurden weitere Simulationen mit einem festen, jedoch geringeren SF durchgeführt. In Listing 4.2 ist das Ergebnis für die Simulation für SF11 zusehen, wobei die maximale Reichweite auf 605 m (vgl. Tabelle 3.1) angepasst wurde. Die restlichen Parameter sind in dem SF11-Durchlauf gleich geblieben. In Listing 4.2 wird deutlich, dass die Vermutung der größeren Anzahl von Interferenzen bei einer höheren TOA einen großen Einfluss hat, schließlich werden für SF11 37 % aller versendeten Pakete am Gateway empfangen.

Listing 4.2 Ergebnis der Simulation für einen fest konfigurierten Spreading Factor (SF) von 11

```
PDR STATS:
nPackets_macRx=2440
nPackets_macTx=6522
PDR=0.374118

ADDITIONAL STATS:
nPackets_phyRxDrop=4082
```

Die Durchläufe wurden zum Nachweis der Vermutung für alle SFs durchgeführt und sind zusammengefasst in Tabelle 4.1 zu sehen. In Anhang C sind die Ergebnisse der Simulationsdurchläufe zusätzlich aufgelistet. Es wird deutlich, dass die TOA eines fest konfigurierten SFs einen großen Einfluss auf die PDR hat. Eine genauere Betrachtung der TOA findet in Unterabschnitt 5.1.1 statt.

Aus einem geringeren SF folgt zwar eine bessere PDR, jedoch wird die abdeckbare Fläche mit einem geringeren SF kleiner. Es ist insbesondere von Interesse, die Fläche beim Ein-

SF	7	8	9	10	11	12
PDR (%)	85,86	79,38	69,53	50,43	37,41	21,90

Tabelle 4.1: Ermittelte Packet Delivery Ratio (PDR) für alle SFs

satz von einem Gateway zu maximieren, weshalb SF12 für die initiale Simulation gewählt wurde. Deshalb wurde im nächsten Schritt der ADR-Ansatz simuliert, sodass die Parallelität beim Senden zu einem Zeitpunkt nicht mehr nur durch verschiedene Frequenzen, sondern auch durch die orthogonalen SFs möglich war[2]. In Abbildung 4.1 sind 1087 Sensorknoten um das in der Mitte platzierte Gateway zu sehen. Die sich ändernde Farbgebung der Sensorknoten ist bedingt durch die unterschiedlichen SFs, wobei die äußersten Sensorknoten mit SF12 konfiguriert wurden. SF7 waren die meisten Sensorknoten zugeordnet, was in Abbildung 4.1 deutlich wird. Daraus profitierte das Netz bezüglich der PDR insofern, dass Interferenzen aufgrund der geringen TOA SF-intern unwahrscheinlicher wurden.

[2]Wie zuvor beschrieben, wurde in verschiedenen Arbeiten bereits gezeigt, dass die Orthogonalität in LoRa nicht perfekt ist und trotz dessen leichte Interferenzen auftreten können. Solche Interferenzen wurden aufgrund des geänderten Fehlermodells in dieser Arbeit nicht weiter betrachtet.

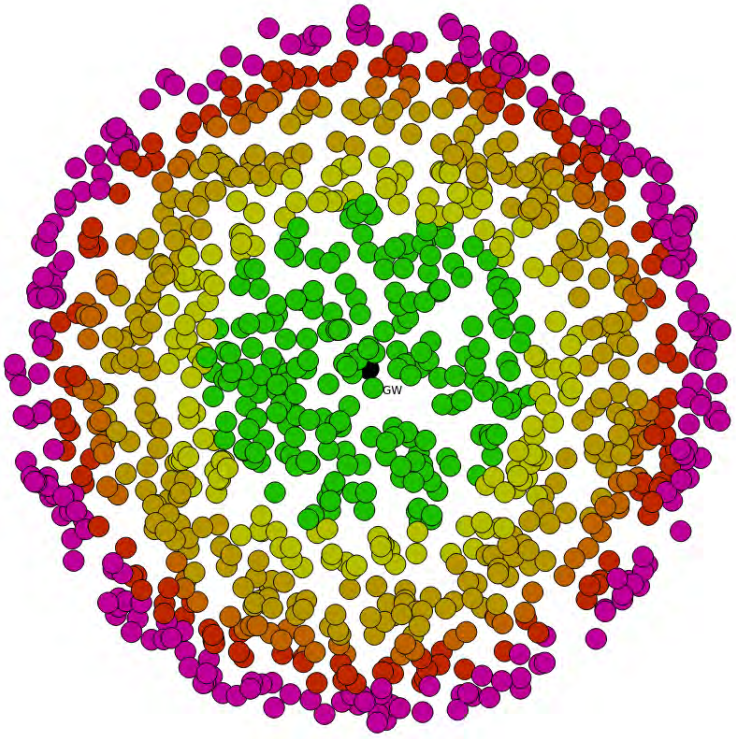

Abbildung 4.1: Ein exemplarisches LoRaWAN aus einem ns3-Simulationsdurchlauf mit konfigurierter Adaptive Data Rate (ADR) (Dargestellt mit Hilfe des Visualisierungstools NetAnim)

Die Verwendung der verschiedenen SFs und der Aspekt, dass SF7 den meisten Sensorknoten zugeordnet wurde und diese Sensorknoten somit die geringste TOA aufweisen, zeigt sich in den Ergebnissen der Simulation. In Listing 4.3 ist in Zeile 4 zu sehen, dass 85,97 % der von den Sensorknoten versendeten Pakete vom Gateway empfangen wurden. Eine ähnliche PDR konnte zwar auch mit SF7 erreicht werden (vgl. Tabelle 4.1), jedoch deckt das LoRaWAN mit dem ADR-ähnlichen Ansatz eine wesentlich größere Fläche ab.

Listing 4.3 Ergebnis der Simulation für einen ADR-ähnlichen Ansatz mit SF7–SF12

```
1 PDR STATS:
2     nPackets_macRx=5607
3     nPackets_macTx=6522
4     PDR=0.859706
5
6 ADDITIONAL STATS:
7     nPackets_phyRxDrop=915
```

4.4 Auswertung des Ist-Zustands

Die Ergebnisse der Simulationsdurchläufe zeigen, dass das angestrebte Netz nicht in dem Maße umsetzbar ist, wie es in Abschnitt 1.1 angedacht wurde. Folgende Probleme lassen sich aus den Simulationsdurchläufen ermitteln:

- **Interferenzen:** Damit keine Kollisionen entstehen können, muss ein strukturierter Sendeplan für jeden einzelnen Sensorknoten vorliegen. In der Simulation wurde der Zeitpunkt des initialen Sendens eines jeden Sensorknotens zufällig bestimmt. Lediglich der Abstand zwischen den Paketen von 10 min wurde fest definiert. Es wird deutlich, dass dieser Ansatz in einem Netz der in Abschnitt 4.3 beschriebenen Größe nicht skaliert.
- **Reichweite:** Durch die Simulation wird deutlich, dass das Ziel einer hohen Reichweite beim Einsatz von lediglich einem Gateway nur mit Hilfe von hohen SFs für die Sensorknoten möglich ist. Ein hoher SF resultiert jedoch in einem höheren Energieverbrauch, was in Unterabschnitt 5.1.1 genauer betrachtet wird. Eine Multi-Gateway-Struktur ist jedoch in vielen ländlichen Regionen und insbesondere in Wäldern aufgrund einer fehlenden Internetanbindung kaum möglich.

Aufgrund der identifizierten Probleme sollten geeignete Erweiterungen für LoRaWAN entwickelt werden, die sowohl in der Reichweite, der PDR als auch dem Energieverbrauch der Sensorknoten skalieren.

5 Protokollerweiterungen

In Kapitel 4 wurde deutlich, dass LoRaWAN für den betrachteten Anwendungsfall in der bestehenden Form nicht ausreichend skaliert. Das Protokoll LoRaWAN bringt jedoch viele Vorteile mit sich, auf die im Kontext des zu entwickelnden Sensornetzes nicht verzichtet werden kann:

- **Kosten:** Die Hardware, um ein LoRaWAN aufzubauen, ist im Vergleich zu anderen LPWAN-Technologien vergleichsweise günstig (siehe beispielsweise [38]).
- **Geringer Energieverbrauch:** LoRaWAN weist in Relation zur maximalen Reichweite einen geringen Energieverbrauch auf, sodass Sensorknoten mehrere Jahre mit lediglich einer Batterie oder einem Akku eingesetzt werden können.
- **Reichweite:** Wie in Unterabschnitt 3.1.1 gezeigt, können LoRaWAN-Geräte auch in Wäldern über hunderte Meter senden, was für den gewünschten Anwendungsfall erforderlich ist.

Aus den zuvor genannten Gründen sollen in diesem Kapitel Erweiterungen für LoRaWAN vorgestellt werden, die den Einsatz des Protokolls für den gewünschten Anwendungsfall der Waldbranderkennung ermöglichen. Die Erweiterungen sollen die Reichweite des Sensornetzes erhöhen und ein strukturiertes Sendeverhalten ermöglichen, wobei die in Abschnitt 1.3 geforderte Skalierbarkeit des Netzes berücksichtigt werden soll. Die Erweiterungen werden zum Ende des Kapitels mit Hilfe des LoRaWAN-Simulators von Van den Abeele et al. [8] analysiert. Restriktionen, wie beispielsweise der maximale Duty Cycle für die in LoRaWAN verwendeten SRD-Subbänder, werden zunächst mathematisch betrachtet, um gewisse Parameter und Einschränkungen schon vorab in der Entwicklung der LoRaWAN-Erweiterungen berücksichtigen zu können. Es sei darauf hingewiesen, dass der maximale Duty Cycle im Rahmen dieser Arbeit aus der europäischen Norm EN 300 220-1 [21] vom Europäischen Institut für Telekommunikationsnormen (ETSI) bzw. der in Deutschland nationalen Umsetzung von der Bundesnetzagentur (Verfügung 133/2019 [23]) betrachtet wird. Außerdem findet die Betrachtung von LoRaWAN nach den Vorgaben des LoRaWAN-Standards [3] und keiner tatsächlichen Implementierung dessen, wie beispielsweise dem The Things Network (TTN), statt. Der Grund liegt darin, dass in den tatsächlichen Implementierungen häufig striktere Restriktionen herrschen, da eine Vielzahl unterschiedlicher Benutzer angenommen wird. Im Rahmen dieser Arbeit wird also von einer eigenen Implementierung des LoRaWAN-Standards ausgegangen.

5.1 Erweiterung zum Erhöhen der Reichweite

Ein erstes Problem, das in Kapitel 4 deutlich wurde, ist die reduzierte Anzahl verfügbarer Gateways. Im Folgenden wird davon ausgegangen, dass lediglich ein Gateway für das gesamte LoRaWAN zur Verfügung steht. Aus diesem Grund kann die Reichweite des Sensornetzes in einem Wald stark eingeschränkt sein. Mögliche Ansätze zum Erhöhen der Reichweite können die folgenden Optionen sein:

- Konfigurieren des SFs der Sensorknoten mit Hilfe von ADR
- Sensorknoten-basiertes Multi-Hop-Verfahren
- Dedizierte Multi-Hop-Knoten

Im Folgenden werden die drei zuvor beschriebenen Optionen in Hinblick auf eine höhere Reichweite des LoRaWANs evaluiert, wobei die in Abschnitt 1.3 geforderte Skalierbarkeit im Fokus steht.

5.1.1 Erhöhen der Reichtweite durch die Adaptive Data Rate (ADR)

Das Protokoll LoRa sieht, wie in Abschnitt 2.1 beschrieben, verschiedene Parametrisierungen der Modulation vor. Ein Parameter, der die maximale Reichweite des LoRa-Signals beeinflusst, ist der Spreading Factor (SF). In Abschnitt 3.1 wurde bereits festgestellt, dass das Pfadverlustmodell von Tewari gute Prognosen für die maximale Reichweite von LoRa-Signalen in Wäldern liefert. Es muss daher zunächst überprüft werden, ob die Konfiguration von hohen SFs für das in Abschnitt 1.1 angestrebte Netz und das in Abschnitt 3.1 ermittelte Pfadverlustmodell skalieren.

Die Skalierung des Netzes bezieht sich, neben der Reichweite, auf den Energieverbrauch pro Sensorknoten. Ein hoher SF hat den Nachteil, dass die Dauer der Übertragung zunimmt, was wiederum den Energieverbrauch erhöht. Deshalb stellt sich die Frage, inwieweit der Energieverbrauch beim Einsatz eines hohen SFs zunimmt. Dafür muss bekannt sein, wie lange ein Sensorknoten zum Versenden eines Paketes benötigt. Um die Dauer der Übertragung zu berechnen, können die vom Hersteller Semtech bereitgestellten Gleichungen verwendet werden ([37], S. 31 ff.). Die Berechnung kann aufgeteilt werden, indem zunächst die Dauer der Übertragung der Präambel und im Anschluss die Dauer der Übertragung der Nutzdaten inklusive der Header berechnet wird. Grundlage für die Berechnung ist die Symbolrate, die mit Gleichung 5.1 ermittelt wird, wobei BW der

Bandbreite in kHz und SF dem SF entspricht. Im Rahmen dieser Arbeit wird davon ausgegangen, dass eine Bandbreite von 125 kHz genutzt wird, wie bereits in Unterabschnitt 3.1.2 beschrieben.

$$T_{sym} = \frac{2^{SF}}{BW} \tag{5.1}$$

Mit Hilfe der Symbolrate aus Gleichung 5.1 kann die Dauer der Übertragung der Präambel mit Gleichung 5.2 berechnet werden. n_p entspricht der Länge der Präambel, welche typischerweise acht ist [9].

$$T_{praeambel} = (n_p + 4.25) \cdot T_{sym} \tag{5.2}$$

Die Dauer der Übertragung der Nutzdaten und der Header lässt sich mit Gleichung 5.3 bestimmen. Die Gleichung ist im Vergleich zur originalen Gleichung in [37] aus Gründen der Übersichtlichkeit bereits insoweit reduziert, dass sowohl ein expliziter Header als auch eine Prüfsumme angenommen werden und die Low Data Rate Optimization deaktiviert ist. Der Grund für die fehlende Verwendung der Low Data Rate Optimization liegt darin, dass der Hersteller Semtech im Datenblatt zur Reihe der SX127X-ICs [37] keine Angaben zur Umsetzung vorstellt und die Effekte somit nicht in Gänze abgeschätzt werden können. Der explizite Header ist notwendig, da bei einem impliziten Header die Länge aller Bestandteile des Pakets konstant sein müssen. Dieser Aspekt kann nicht immer gewährleistet werden, weshalb ein expliziter Header betrachtet wird. Ebenso wird die Prüfsumme vorausgesetzt, um fehlerhafte Pakete frühestmöglich verwerfen zu können.

$$T_{nutzdaten} = (8 + \left\lceil \frac{8 \cdot PL - 4 \cdot SF + 28 + 16}{4 \cdot SF} \right\rceil \cdot CR + 4) \cdot T_{sym} \tag{5.3}$$

In Gleichung 5.3 entspricht PL der Länge der Nutzdaten in Byte, SF dem SF (SF7–SF12), CR der Code Rate (CR) und T_{sym} der in Gleichung 5.1 berechneten Symbolrate. Wird die Übertragungsdauer für die in Abschnitt 1.1 vorgeschlagene Paketgröße von insgesamt 16 Byte berechnet sowie eine CR von 4/5 angenommen, so ergeben sich die in Tabelle 5.1 aufgelisteten Übertragungszeiten. In Tabelle 5.1 entspricht T_{paket} der Summe aus der Übertragungsdauer der Nutzdaten inklusive der Header ($T_{nutzdaten}$ aus Gleichung 5.3) und der Übertragungsdauer der Präambel ($T_{praeambel}$ aus Gleichung 5.2). Mit Hilfe dieser Auflistung kann überprüft werden, inwieweit die Dauer der Übertragung die Akkulaufzeit eines Sensorknotens beeinflusst.

SF	$T_{nutzdaten}$	$T_{praeambel}$	T_{paket}
7	38,912 ms	12,544 ms	51,456 ms
8	67,584 ms	25,088 ms	92,672 ms
9	114,688 ms	50,176 ms	164,864 ms
10	229,376 ms	100,352 ms	329,728 ms
11	376,832 ms	200,704 ms	577,536 ms
12	753,664 ms	401,408 ms	1155,072 ms

Tabelle 5.1: Übertragungsdauer für verschiedene SFs bei 16 Byte Nutzdaten und Header

Im nächsten Schritt muss ermittelt werden, welcher SF geeignet ist, um das Ziel der Akkulaufzeit eines Sensorknotens von mindestens acht Jahren, wie in Abschnitt 1.1 beschrieben, erfüllen zu können. Dafür muss berechnet werden, wie lange ein Sensorknoten maximal in einer Wach-Phase sein darf[1]. Da in diesem Abschnitt noch keine genaue Hardware zu Grunde gelegt wird und das Vorgehen lediglich exemplarisch verdeutlicht werden soll, werden stellvertretend exemplarische Werte angenommen. Im Rahmen der Hardwareentwicklung in Kapitel 6 werden genaue Messungen zugrunde gelegt.

Gleichung 5.4 ermittelt zunächst, wie hoch der Strombedarf im Durchschnitt pro Stunde für einen Sensorknoten ist, wobei I_w dem durchschnittlichen Strombedarf in mA während der Wach-Phase, t_w der Dauer der Wach-Phase in ms pro Stunde, I_s dem Strombedarf in mA während der Schlaf-Phase und t_s der Dauer der Schlaf-Phase in ms pro Stunde entspricht. Die Multiplikation der Summanden im Dividend mit fünf ergibt sich durch die in Abschnitt 1.1 vorgestellte Bedingung, dass alle 10 min gesendet werden soll. Demnach ist es innerhalb einer Stunde möglich, fünf Pakete zu versenden. Für I_w wird der Durchschnitt angenommen, da dieser im Gegensatz zum Median sensibler gegenüber Spitzenlasten ist und Spitzenlasten den Wert somit stärker beeinflussen, was für eine realistische Betrachtung besser geeignet ist. In der Praxis würde das Integral des Strombedarfs pro Wach-Phase betrachtet werden.

$$I_h = \frac{5 \cdot I_w \cdot t_w + I_s \cdot t_s}{3600000\,ms} \tag{5.4}$$

Um die Akku-Laufzeit mit Hilfe von Gleichung 5.5 berechnen zu können, wird ein bestimmter Akkutyp mit einer Akkukapazität vorausgesetzt. Im Folgenden wird deshalb ein Lithium-Ionen-Akku (Li-Ion-Akku) mit 2850 mAh Kapazität angenommen. Dieser Akku-

[1] In der Laufzeit der Sensorknoten existieren zwei Phasen: Die Wach-Phase, in der der Mikrocontroller aktiv ist, Sensorwerte ermittelt und versendet werden sowie die Schlaf-Phase, in der die meisten Hardware-Komponenten deaktiviert sind, um so den Energieverbrauch möglichst gering zu halten.

Typ ist in Relation zur Kapazität kostengünstig[2]. Zwar weisen Li-Ion-Akkus eine gewisse Robustheit gegenüber Temperaturschwankungen auf [40], jedoch sollte die Kapazität nicht in Gänze in die Rechnung aufgenommen werden. Da die Sensorknoten in Wäldern platziert werden und somit jeglicher Witterung ausgesetzt sind und eine gewisse Selbstentladung des Akkus über die Laufzeit eintritt, sollte die Kapazität des Akkus für die Überschlagsrechnung nicht vollständig eingesetzt werden. Deline et al. [41] haben die Kapazität von Li-Ion-Akkus bei verschiedenen Umgebungstemperaturen untersucht und kamen zum Ergebnis, dass die Kapazität über 10 Jahre 20–35 % abnehmen kann[3]. Besonders Temperaturen unter 0 °C scheinen einen hohen Einfluss auf die Kapazität zu haben. Im Folgenden wird davon ausgegangen, dass das Sensornetz in einer Region eingesetzt wird, in der Temperaturen von unter 0 °C nur in einem kurzen Zeitraum in den Wintermonaten üblich ist. In Anbetracht der Annahme ist es ausreichend, in Gleichung 5.5 für C 80 % der eigentlichen Kapazität einzusetzen[4]. Um die Akkulaufzeit in Jahren ermitteln zu können, wird die Kapazität durch den durchschnittlichen Strombedarf pro Stunde aus Gleichung 5.4 geteilt und auf Jahre hochgerechnet, wie in Gleichung 5.5 zu sehen.

$$t_y = \left(\frac{C \cdot 0,8}{I_h}\right) /24 \cdot 365 \tag{5.5}$$

Um die maximale Dauer der Wach-Phase zum Erreichen der acht Jahre berechnen zu können, wird Gleichung 5.4 in Gleichung 5.5 eingesetzt, t_w als unbekannt angenommen und $t_y = 8$ gesetzt. Um abschätzen zu können, welchen Strombedarf ein SX1276-LoRa-Modul beim Senden hat, wurde zuvor eine exemplarische Messung mit Hilfe eines µCurrent Gold Multimeter-Adapters (siehe Abschnitt 7.1), eines Oszilloskops und dem in Unterabschnitt 3.1.1 vorgestellten Sender durchgeführt. Der Quellcode aus Unterabschnitt 3.1.2 wurde insoweit angepasst, dass lediglich ein Paket versendet wurde. Für die Sendeleistung wurden die in Unterabschnitt 3.1.1 verwendeten 14 dBm beibehalten. In Abbildung 5.1 ist im markierten Feld im unteren Bereich zu erkennen, dass im Maximum eine Spannung von 102 mV gemessen wurde[5]. Die 102 mV können ohne weitere Umrechnung als 102 mA abgelesen werden (siehe Abschnitt 7.1 für eine genauere Erklärung der

[2] Meist kosten Akkus des beschriebenen Typs weniger als 5 €.

[3] Die untersuchten Akkus von Deline et al. [41] weisen im Vergleich zu den hier betrachteten Akkus eine höhere Kapazität auf.

[4] Der Ansatz, eine geringere Kapazität des Akkus vorauszusetzen, dient lediglich einer Überschlagsrechnung und entspricht somit keiner exakten Modellierung.

[5] In Abbildung 5.1 wurde SF12 verwendet. Für andere SFs wurden in etwa gleiche Maximalwerte gemessen (98–105 mA, wobei die unterschiedlichen Maximalwerte auch durch das erkennbare Rauschen verursacht worden sein können).

Messungen mit einem µCurrent Gold). Demnach hat ein SX1276-LoRa-Modul einen maximalen Strombedarf von 102 mA beim Senden mit einer Sendeleistung von 14 dBm. Der Einfachheit halber wird der Maximalwert im Folgenden als Durchschnittswert für das Senden angenommen, schließlich ist der Strombedarf in Abbildung 5.1 recht konstant. Des Weiteren wird der Boot-Vorgang und das Ermitteln der Sensorwerte nicht mit in die Rechnung aufgenommen, da das Senden prozentual vermutlich die längste Zeit in einer Wach-Phase in Anspruch nehmen würde. Würde der Strombedarf der gesamten Wach-Phase betrachtet werden, wäre der durchschnittliche Strombedarf vermutlich geringer.

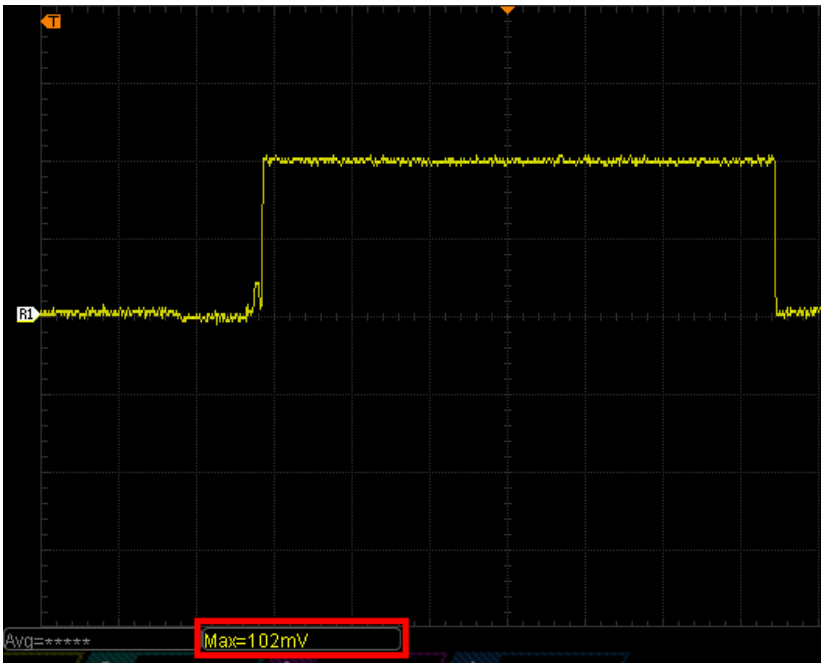

Abbildung 5.1: Messung des Strombedarfs eines SX1276-LoRa-Moduls mit SF12 bei 14 dBm Sendeleistung

Für die exemplarische Berechnung soll $I_w = 102\,\text{mA}$ und $I_s = 0.001\,\text{mA}$ eingesetzt werden. Der Ruhestrom I_s entspricht mit 1 µA dem Ruhestrom von LDOs für sehr energieeffiziente Hardware[6]. t_s entspricht der Zeit, die nach Abzug der fünf Wach-Phasen von einer Stunde übrig bleibt, also $t_s = 3600000 - (t_w \cdot 5)$. Das Ergebnis zeigt, dass eine Wach-Phase unter den genannten Bedingungen eine maximale Dauer von 222 ms aufweisen darf. Die Wach-Phase teilt sich auf den Boot-Vorgang des Mikrocontrollers, dem Ermitteln der Sensorwerte und das Versenden des Paketes auf. Im Vergleich zu Tabelle

[6]Siehe beispielsweise Datenblatt eines Texas Instruments TPS7A05 LDOs: https://www.ti.com/lit/ds/symlink/tps7a05.pdf (Abgerufen am 2021-02-18)

5.1 wird also deutlich, dass lediglich SF7–SF9 in Frage kommen würden. Somit ist das Erhöhen des SFs in ADR theoretisch nur bis SF9 möglich, was die Reichweite wiederum reduziert. Ein einfaches Erweitern des Netzes mit Hilfe von hohen SFs erscheint im Kontext des Anwendungsfalls demnach nicht sinnvoll. Des Weiteren ist festzustellen, dass eine möglichst geringe Wach-Phase die Akkulaufzeit immens erhöhen kann. Es ist also nicht wünschenswert, die maximale Wach-Phase, wie die exemplarisch berechneten 222 ms, auszureizen.

5.1.2 Sensorknoten-basiertes Multi-Hop-Verfahren

Wie im vorherigen Ansatz beschrieben, ist es bei der Betrachtung des Energieverbrauchs nicht empfehlenswert, einen hohen SF für die Sensorknoten zu konfigurieren. Ein anderer Ansatz wäre, die Reichweite pro Sensorknoten durch Nutzen eines geringen SFs zu minimieren und ein Routing der Pakete zwischen den Sensorknoten zum Gateway aufzubauen. Das Problem eines solchen Ansatzes besteht darin, dass die Sensorknoten nicht nur senden, sondern auch empfangen und weiterleiten müssen, was den Energieverbrauch erhöht. Zusätzlich kommen Restriktionen des Duty Cycles für die in LoRaWAN verwendeten SRD-Subbänder hinzu, sodass bei einer Vielzahl von Sensorknoten ein komplexes Routing resultieren und die Sensorknoten, die als Constrained Devices einzustufen sind, nicht ausreichend skalieren würden. Das bedeutet, dass das Problem des Energieverbrauchs des vorherigen Ansatzes verschärft werden würde und zusätzlich der maximale Duty Cycle für jeden Sensorknoten pro Subband eingehalten werden müsste. Zusammenfassend lässt sich feststellen, dass die Sensorknoten als Constrained Devices im Rahmen dieser Arbeit nicht für Multi-Hop-Verfahren geeignet sind.

5.1.3 Dedizierte Multi-Hop-Knoten

Der vorherige Ansatz hat gezeigt, dass dedizierte Multi-Hop-Knoten im Netz notwendig sind, um so die Reichweite zu erhöhen. Diese Knoten müssen vor allem bezüglich des Energieverbrauchs flexibel sein, was beispielsweise durch Akkus mit größerer Kapazität und Energy Harvesting möglich ist. Des Weiteren ist davon auszugehen, dass eine im Vergleich zu den Sensorknoten wesentlich geringere Anzahl dedizierter Multi-Hop-Knoten eingesetzt werden muss, sodass der finanzielle Mehraufwand gering ausfallen würde. Um dedizierte Multi-Hop-Knoten einsetzen zu können, müssen verschiedene Teil-Probleme gelöst werden:

- Anordnung der Multi-Hop-Knoten auf der betrachteten Fläche
- Routing-Strategie
- Berücksichtigen des Duty Cycles für die Multi-Hop-Knoten

Wie in Abschnitt 1.1 beschrieben, wird von einer annähernden Gleichverteilung der Sensorknoten im Netz ausgegangen. Eine weitere Bedingung ist, dass lediglich Pakete von den Sensorknoten versendet werden, die nicht vom Gateway bestätigt werden müssen. In einem hochdichten Netz, in welchem lediglich Sensorwerte übermittelt werden, erscheint der Aufwand für bestätigte Nachrichten auch in Anbetracht des maximalen Duty Cycles der Multi-Hop-Knoten nicht sinnvoll. Außerdem wird im Folgenden davon ausgegangen, dass ein Multi-Hop-Knoten empfangene Pakete als direkte Kopie weiterleitet und keine eigenen Informationen hinzufügt. Demnach bleibt auch die Paketgröße im Verlauf der Weiterleitung konstant.

Die Anordnung der Multi-Hop-Knoten im betrachteten Netz kann zunächst auf die Anordnung von Mobilfunk-Zellen zurückgeführt werden. Häufig wird zur Planung von Mobilfunk-Zellen eine Sechseck-Anordnung mit einem Mobilfunk-Mast in der Mitte des Sechsecks verwendet (vgl. [42], S. 92). Diese Anordnung hat den Vorteil, dass die einzelnen Bereiche nahtlos ineinander übergehen und somit übersichtlich dargestellt werden können. Auf das LoRaWAN übertragen, wäre in der Mitte der Sechseck-Anordnung das Gateway, um ein effizientes Routing zu ermöglichen. Das Routing wird im späteren Verlauf des Unterabschnitts genauer beschrieben. Im Folgenden wird ein Sechseck, in welchem sich ein Multi-Hop-Knoten oder das Gateway befindet, als Region und ein in einer Region platzierter Multi-Hop-Knoten als Relay Node (RN) bezeichnet. Eine Ausnahme bildet die Gateway-Region, schließlich ist in dieser Region kein RN platziert. Eine beispielhafte Anordnung für 19 Regionen ist in Abbildung 5.2 zu sehen.

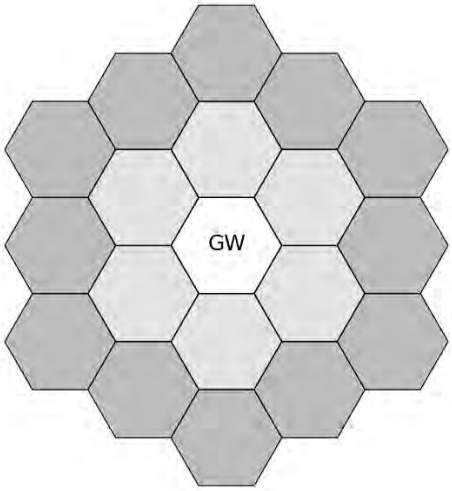

Abbildung 5.2: Sechseck-Anordnung von 19 Regionen eines Multi-Hop-LoRaWANs

Die einzelnen Regionen werden im Folgenden in Ringe aufgeteilt. Die Regionen, die direkt um das Gateway platziert sind (vgl. hellgraue Regionen in Abbildung 5.2), werden dem ersten Ring zugeordnet. Mit jedem neuen Ring (vgl. graue Regionen in Abbildung 5.2) wird der Index des Rings um eins erhöht. Demnach sind in Abbildung 5.2 zwei Ringe um das Gateway aufgebaut. Im Folgenden wird davon ausgegangen, dass innerhalb der inneren Ringe eine gleichmäßige Verteilung der Regionen, wie in Abbildung 5.2 gezeigt, existiert. Anzumerken ist, dass der äußerte Ring nicht immer in Gänze mit Regionen versehen werden kann. Werden beispielsweise 21 Regionen gefordert, würde ein dritter Ring in Abbildung 5.2 entstehen, in dem zwei Region platziert werden müssten. Dieses Problem wird im späteren Verlauf genauer behandelt.

Um die Größe einer Region zu bestimmen, wird der Radius benötigt, der mit einem RN für einen bestimmten SF abgedeckt wird. Da die RNs in der Mitte einer Region platziert werden, könnten die RNs untereinander keine Pakete empfangen, sofern der Radius aus Tabelle 3.3 verwendet wird. Das liegt daran, weil die Entfernung zwischen zwei RNs dann dem Durchmesser entsprechen würde. Deshalb wird der Radius für die Regionen im Folgenden halbiert[7]. In Unterabschnitt 3.1.4 wurde die maximale Reichweite bereits für einen exemplarischen Wald ermittelt.

[7] In der Praxis würde voraussichtlich ein größerer Puffer eingeplant werden, sodass sichergestellt ist, dass sich die RNs innerhalb der Reichweite befinden.

$$n_{RN} = \left\lceil \frac{A}{\frac{3 \cdot \sqrt{3} \cdot s^2}{2}} \right\rceil - 1 \tag{5.6}$$

Mit Hilfe von Gleichung 5.6 kann die Anzahl der benötigten Regionen berechnet werden, wobei A der abzudeckenden Fläche in Quadratmetern und s der halbierten maximalen Reichweite in Metern entspricht[8]. Von der Anzahl der ermittelten Regionen muss im Anschluss noch eine Region subtrahiert werden, da die Gateway-Region eine spezielle Region ist, welche die Besonderheit aufweist, weniger eingeschränkt bezüglich des maximalen Duty Cycles zu sein. Im weiteren Verlauf kann diese Region deshalb zunächst unbeachtet bleiben.

SF	s	Regionen
7	197	99
8	256	58
9	307	40
10	347	31
11	381	26
12	428	20

Tabelle 5.2: Anzahl benötigter Regionen pro SF bei 10 km² Fläche (vgl. Anhang B)

In Tabelle 5.2 ist auf Basis der in Kapitel 3 durchgeführten Messungen aufgeführt, wie viele Regionen für SF7–SF12 bei einer exemplarischen Fläche von 10 km² benötigt werden. Aus Tabelle 5.2 lässt sich bereits ablesen, dass für die in Abschnitt 1.1 beschriebenen 25 km² Fläche der organisatorische Aufwand nur für die RNs zu groß wäre. Aus diesem Grund sollen die Erweiterungen im Folgenden mit 10 km² Fläche exemplarisch vorgestellt werden.

Im nächsten Schritt muss ein Routing zwischen den RNs zum Gateway integriert werden. Für diesen Zweck stehen diverse Routing-Algorithmen zur Auswahl. Im Kontext von IoT-Netzen wird häufig das Routing-Protokoll RPL (RFC 6550 [44]) verwendet. Da RPL für IPv6-basierte Netze ausgelegt ist, kann RPL jedoch nicht ohne Anpassungen in LoRaWAN eingesetzt werden. Deshalb haben Sartori et al. [45] LoRaWAN für den Einsatz von RPL erweitert. Weitere Routing-Algorithmen, die für den Einsatz in einem LoRaWAN evaluiert und angepasst wurden, sind in der Arbeit von Osorio et al. [46] zusammengefasst

[8] Es ist anzumerken, dass die Summe der Flächen der Regionen grundsätzlich größer oder gleich der abzudeckenden Fläche ist. Dieser Effekt ist bedingt durch das Aufrunden auf die nächst größere Ganzzahl in Gleichung 5.6.

beschrieben. Das Routing sollte im Rahmen dieser Arbeit darauf ausgelegt sein, die kürzeste Strecke von einem Sensorknoten zum Gateway zu finden. Nur dann, wenn der kürzeste Pfad verwendet wird, kann der maximale Duty Cycle der RNs ausgereizt und die Dauer der Übertragung für ein Paket vom Sensorknoten zum Gateway minimiert werden. Des Weiteren ist es notwendig, die Anzahl einsetzbarer Sensorknoten im resultierenden Netz noch vor dem Einsatz abschätzen zu können. Die Anzahl einsetzbarer Sensorknoten steht im direkten Zusammenhang mit der abdeckbaren Fläche, was bereits in Abschnitt 1.1 beschrieben wurde. RPL ist im Rahmen dieser Arbeit deshalb kein geeigneter Routing-Algorithmus, da der Aufbau des Routings während der Laufzeit dynamisch erfolgt. Der Routing-Aufbau zur Laufzeit hätte zur Folge, dass die Last möglicherweise nicht über alle RNs gleichverteilt wird. Daraus folgt, dass der Duty Cycle eines RNs möglicherweise kaum, der Duty Cycle eines anderen RNs jedoch in Gänze ausgereizt wäre. Dadurch könnte es vorkommen, dass Sensorwerte aus Regionen gepuffert oder, was in Anbetracht der Größe des angestrebten Netzes wahrscheinlicher ist, verworfen werden. Da in RPL dedizierte Pakete mit Routing-Informationen versendet werden, wäre der maximale Duty Cycle pro RN zusätzlich eingeschränkt. In allen weiteren Routing-Algorithmen, die in der Arbeit von Osorio et al. [46] vorgestellt wurden, liegt der Fokus ebenfalls nicht auf dem Ausreizen des maximalen Duty Cycles pro Multi-Hop-Knoten. Beispielsweise nutzen Dias und Grilo [47] einen Routing-Ansatz in LoRaWAN mit Hilfe des Protokolls DSDV. Jedoch werden die Routing-Tabellen in DSDV in gewissen Abständen durch Kommunikation zwischen den Teilnehmern aktualisiert, was den Duty Cycle pro RN beeinträchtigen würde. Des Weiteren verwenden Dias und Grilo [47] einen eigenen Header für das Routing, welcher die TOA und somit den Duty Cycle zusätzlich negativ beeinflussen würde. Im Folgenden wird deshalb ein eigener, baumbasierter Routing-Ansatz vorgestellt, der ein statisches Routing auf Basis des zuvor beschriebenen Aufbaus der Regionen durchführt und versucht, den maximalen Duty Cycle pro RN auszureizen.

Um das Routing möglichst effizient durchführen zu können, bieten sich möglichst gleichmäßig verteilte Bäume an. Jeder Knoten im ersten Ring entspricht der Wurzel eines RN-Baumes, wobei direkt angrenzende Regionen aus einem größeren Ring an eine Wurzel-Region als dessen Kinder definiert werden können. Es wird deutlich, dass die Regionen gleichmäßig auf die Bäume verteilt werden müssen. Es kann in dem baumbasierten Routing vorkommen, dass eine Region zu zwei Bäumen zugeordnet werden könnte. Dieser Fall tritt insbesondere bei einer Vielzahl von Ringen auf. Welche RN-Kinder zu welchem Baum hinzugefügt werden, wird zur besseren Flexibilität nicht weiter definiert und soll im Rahmen einer Umsetzung festgelegt werden. Es gilt lediglich die Bedingung, dass RN-Bäume nur RNs enthalten, die direkt an eines der RN-Kinder des Baumes grenzen. Die im Rahmen dieser Arbeit umgesetzte Strategie zur Elter-Kind-Beziehung der RNs wird in

Abschnitt 5.3 vorgestellt. Des Weiteren ist anzumerken, dass jedes RN ein einzelner Ausfallpunkt ist und ein Ausfall eines RN für einen Gesamtausfall eines Teilbaums sorgen kann.

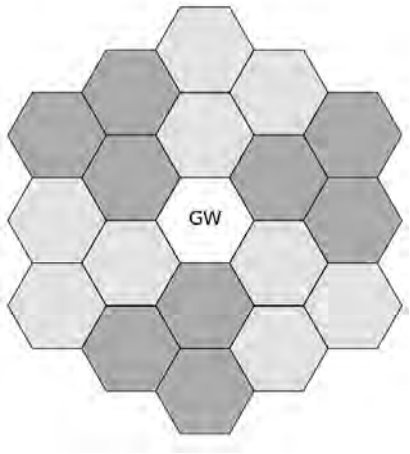

Abbildung 5.3: Sechseck-Anordnung von sechs Routing-Bäumen in einem um Regionen in einem Multi-Hop-LoRaWAN

Wie bereits erwähnt, kann es vorkommen, dass der äußerste Ring nicht in Gänze mit Regionen gefüllt werden kann. Ist beispielsweise nur noch eine Region übrig, so kann diese schließlich einer der sechs verfügbaren Bäume zugeordnet werden. In Abbildung 5.3 sind die sechs Bäume durch abwechselnde Graustufen gekennzeichnet. Demnach könnte jedem dieser Bäume die letzte Region zugeordnet werden. Die Entscheidung, welchem Baum die letzte Region zugeordnet wird, wird in der Protokoll-Erweiterung ebenso offen gelassen. Demnach kann die Verteilung der verbleibenden Regionen so ablaufen, wie es für einen Anwendungsfall geeignet ist.

$$n_{tree} = \begin{cases} \left\lceil \frac{n_{RN}}{6} \right\rceil & \text{falls } n_{RN} > 6 \\ 1 & \text{falls } n_{RN} \leq 6 \end{cases} \tag{5.7}$$

Um die Restriktionen des maximalen Duty Cycles in einem RN-Baum berücksichtigen zu können, muss die Anzahl der Regionen pro Baum ermittelt werden, was mit Hilfe von Gleichung 5.7 berechnet werden kann. n_{RN} entspricht der Anzahl benötigter Regionen, die in Gleichung 5.6 ermittelt wurde. Sofern $n_{RN} > 6$, werden die Regionen gleichmäßig auf die sechs Wurzel-RNs verteilt. Ist $n_{RN} \leq 6$, so muss kein Routing-Baum aufgebaut werden. In Tabelle 5.3 ist die Erweiterung von Tabelle 5.2 um die maximale Größe

der Bäume pro SF der RNs zu sehen. In Tabelle 5.3 wird deutlich, dass beispielsweise für SF7 99 Regionen für eine Fläche von 10 km² benötigt werden, wobei zur Gleichverteilung ein Routing-Baum maximal 17 RNs beinhalten darf.

SF	s	Regionen	n_{tree}
7	197	99	17
8	256	58	10
9	307	40	7
10	347	31	5
11	381	26	5
12	428	20	4

Tabelle 5.3: Anzahl benötigter Regionen und maximale Baumtiefe bei 10 km² Fläche pro SF

Um den maximalen Duty Cycle betrachten und somit die maximale Größe einer Region bezüglich der einsetzbaren Sensorknoten bestimmen zu können, kann zunächst betrachtet werden, wie viel Zeit ein RN benötigt, um für eine bestimmte Anzahl Sensorknoten eine Weiterleitung durchzuführen. In Gleichung 5.8 wird zunächst im Dividend ermittelt, wie viel Zeit die Weiterleitung des Paketes in Anspruch nimmt und wie lange für das gegebene Subband gewartet werden muss, sodass erneut gesendet werden darf. In Gleichung 5.8 entspricht n_{ed} der Anzahl der Sensorknoten, n_p der Anzahl der Pakete, die in einer betrachteten Zeiteinheit (hier: eine Stunde) versendet werden sollen, t_{tx} der Dauer der Übertragung in Millisekunden, d dem Multiplikator, der die Wartezeit in Relation zur Dauer der Übertragung t_{tx} repräsentiert[9] und n_f der Anzahl zur Verfügung stehender Subbänder.

$$t = \frac{(n_{ed} \cdot n_p \cdot t_{tx} \cdot d) + (n_{ed} \cdot n_p \cdot t_{tx})}{n_f} \tag{5.8}$$

Gleichung 5.8 kann so umgestellt werden, dass die maximale Anzahl möglicher Sensorknoten für lediglich ein RN ohne Kinder im Routing-Baum berechnet wird, wie in Gleichung 5.9 zu sehen. Dieser Schritt ist notwendig, um im weiteren Verlauf die Anzahl der Sensorknoten auf den Baum aufzuteilen und den maximalen Duty Cycle im Wurzel-RN einzuhalten.

$$n_{ed} = \frac{n_f \cdot t}{(n_p \cdot t_{tx} \cdot d) + (n_p \cdot t_{tx})} \tag{5.9}$$

[9] Beispielsweise ist der Multiplikator für ein Subband mit 1 % Duty Cycle 99, da 99 mal die Übertragungsdauer des Paketes gewartet werden muss.

Für diesen Zweck wird für t die betrachtete Zeiteinheit von einer Stunde ($t = 3600000\,ms$) eingesetzt. Wie in Abschnitt 1.1 beschrieben, soll alle 10 min innerhalb von einer Stunde von einem Sensorknoten gesendet werden, weshalb $n_p = 5$ eingesetzt wird. Des Weiteren werden zwei Subbänder angenommen ($n_f = 2$) und ein 1 % Duty Cycle ($d = 99$) pro Subband. Die Dauer der Übertragung t_{tx} wird für den jeweiligen SF aus Tabelle 5.1 entnommen. In Tabelle 5.4 sind die Ergebnisse zusammenfassend aufgelistet. Tabelle 5.4 zeigt, wie viele Sensorknoten eine Region umfassen kann, damit ein RN ohne RN-Kinder innerhalb der betrachteten Zeiteinheit alle empfangenen Pakete weiterleiten kann und gleichzeitig den maximalen Duty Cycle einhält. Da die Dauer der Übertragung für SF7 am geringsten ist, kann ein SF7-konfiguriertes RN die meisten Sensorknoten beinhalten.

SF	n_{ed}
7	279
8	155
9	87
10	43
11	24
12	12

Tabelle 5.4: Maximale Anzahl Sensorknoten für ein RN bei zwei Subbändern mit 1 % Duty Cycle

Tabelle 5.4 bezieht sich, wie zuvor erwähnt, lediglich auf ein RN und nicht auf weitergeleitete Pakete von RN-Kindern. Um die Anzahl der Sensorknoten für das vorgestellte Baum-basierte Routing berechnen zu können, wird die maximale Anzahl Sensorknoten für das Einhalten des Duty Cycles in Tabelle 5.4 für die Wurzel des Baumes vorausgesetzt. Demnach kann die Wurzel des RN-Baumes für SF7 maximal 279 Pakete von den eigenen Sensorknoten und den Sensorknoten der RN-Kinder pro Stunde versenden.

$$ed_{rn} = \left\lfloor \frac{n_{ed}}{n_{tree}} \right\rfloor \tag{5.10}$$

Durch Gleichung 5.10 kann die maximale Anzahl der Sensorknoten pro Region in einem Routing-Baum berechnet werden. Da die maximale Anzahl Sensorknoten zum Einhalten des maximalen Duty Cycles berechnet wird, muss in Gleichung 5.10 auf die nächst kleinere Ganzzahl abgerundet werden. Tabelle 5.5 listet die maximale Anzahl Sensorknoten pro Region und die nach Gleichung 5.11 resultierende Gesamtanzahl Sensorknoten im Netz auf. In Tabelle 5.5 fällt auf, dass die Gesamtanzahl Sensorknoten im LoRaWAN bei einem für die RN konfigurierten SF größer sieben bereits unter den in Kapitel 4 simulierten

SF	s	Regionen	n_{tree}	ed_{rn}	ed_{ges}
7	197	99	17	16	1600
8	256	58	10	15	885
9	307	40	7	12	492
10	347	31	5	8	256
11	381	26	5	4	108
12	428	20	4	3	63

Tabelle 5.5: Auflistung der maximal möglichen Parameter für ein LoRaWAN mit einer Multi-Hop-Erweiterung und einer maximalen Fläche von 10 km²

1087 Sensorknoten liegt. Es sei jedoch anzumerken, dass die abgedeckte Fläche durch den Multi-Hop-Ansatz größer als in Kapitel 4 ist. Ebenso fällt in Tabelle 5.5 auf, dass der organisatorische Aufwand für SF7 mit 99 RNs sehr umfangreich erscheint und SF12 mit lediglich 63 Sensorknoten auf 10 km² Fläche die Anforderung nicht erfüllt. Im Folgenden wird deshalb auf die Betrachtung von SF7 und SF12 für die RNs verzichtet.

$$ed_{ges} = ed_{rn} \cdot n_{tree} \tag{5.11}$$

Die Wahl des SFs für die RNs beeinflusst also, wie in Tabelle 5.5 zu sehen, die Anzahl benötigter RNs sowie die maximale Anzahl einsetzbarer Sensorknoten. Dass die Wahl des SFs die TOA und somit auch die Betrachtung des Duty Cycles beeinflusst, wurde bereits in Abschnitt 4.3 gezeigt und bestätigt sich erneut in Tabelle 5.5. Die vorherigen Ausführungen beziehen sich auf 10 km² abzudeckender Fläche. Sollen also mehr Sensorknoten eingesetzt werden, so ist es notwendig, die Fläche zu reduzieren. Durch die reduzierte Fläche müssen weniger RNs eingesetzt werden, was wiederum die Größe eines Routing-Baumes reduziert. Eine weitere Möglichkeit, die Anzahl der Sensorknoten zu erhöhen, ist, den zeitlichen Abstand zwischen dem Senden pro Sensorknoten zu erhöhen. In den vorherigen Beispielen wurde angenommen, dass alle 10 min ein Paket pro Sensorknoten versendet wird. Wird der Abstand zwischen zwei zu versendenen Paketen pro Sensorknoten erhöht, so wird jedoch die Auflösung in Bezug auf die Anzahl von Sensorwerten pro Zeiteinheit reduziert. Demnach ist es abhängig vom Anwendungsfall, welche Parameter im resultierenden Netz aufgrund besonderer Wichtigkeit angepasst werden.

Ein Aspekt, der abschließend betrachtet werden muss, ist der Ablauf des Routings im Baum. Ein erstes Problem beim Routing ist darin festzustellen, dass ein RN nur die Pakete weiterleiten darf, die von den eigenen Kindern versendet wurden. Leitet ein RN auch Pakete anderer Routing-Bäume weiter, so hätten die vorherigen Berechnungen bezüglich der maximalen Anzahl Sensorknoten pro Routing-Baum keine Anwendung mehr.

Wird davon ausgegangen, dass alle RNs dauerhaft im Empfangsmodus sind, so kommt es zwangsläufig vor, dass ein RN ein Paket eines anderen RNs oder ein Paket eines Kindes eines anderen RNs empfängt, die nicht dem eigenen Routing-Baum zugeordnet sind. Da ein RN, wie eingangs beschrieben, lediglich eine Kopie des Sensorknoten-Paketes weiterleitet und keine eigenen Informationen hinzufügt, muss ein RN die Sensorknoten seiner Kinder kennen. Zunächst stellt sich die Frage, wie ein RN die empfangene Nachricht zuordnen soll, schließlich setzt LoRaWAN auf eine Ende-zu-Ende-Verschlüsselung.

Größe (Byte)	4	1	2	0..15
FHDR	DevAddr	FCtrl	FCnt	FOpts

Abbildung 5.4: Aufbau des Frame Header (FHDR) (vgl. [3], S. 18)

Im LoRaWAN-Standard [3] wird beschrieben, dass jedes Paket einen Frame Header (FHDR) enthalten muss, was bereits in Unterabschnitt 2.2.5 erwähnt wurde. Der Aufbau des FHDR ist in Abbildung 5.4 zu sehen. Der FHDR wird, im Gegensatz zu den Nutzdaten, nicht verschlüsselt versendet (vgl. [3], S. 18 f.). Im FHDR ist die Adresse DevAddr vorzufinden, die einen Knoten in einem LoRaWAN identifiziert ([3], S. 49). Empfängt ein RN ein Paket eines Sensorknotens, so kann das RN anhand der DevAddr erkennen, ob das empfangene Paket von einem Sensorknoten aus seiner Region oder von einem Sensorknoten seiner RN-Kinder stammt. Mit diesem Ansatz wird verhindert, dass ein RN Pakete weiterleitet, die nicht von den RN-Kindern oder von Sensorknoten der RN-Kinder stammen.

Ein weiteres Routing-Problem ist das Verhindern von Routing-Schleifen. Mit dem zuvor beschriebenen Ablauf ist es möglich, dass ein RN-Kind dem RN-Elter ein Paket weiterleitet, das RN-Elter dieses Paket ebenso weiterleitet und das RN-Kind das Paket erneut empfängt. Dieses Fehlverhalten muss unterbunden werden, schließlich endet eine solche Weiterleitung womöglich in endlosen Schleifen. Eine Idee zum Unterbinden könnte zeitbasiert ablaufen. Sobald ein RN ein Paket von einem RN-Kind oder einem Sensorknoten empfangen hat, wartet es eine gewisse Zeit, bis es ein Paket der gespeicherten Adresse wieder annimmt und weiterleitet. Bei diesem Ansatz ist jedoch festzustellen, dass es nicht allgemeingültig verwendet werden kann. Ist der zeitliche Abstand zwischen zwei Paketen eines Sensorknotens recht gering (beispielsweise 30 s) und verzögert sich die Weiterleitung beim RN-Elter um gleich oder mehr als 30 s, so könnte das Paket beim RN-Kind doppelt empfangen werden. Zwar sind endlose Schleifen in diesem Fall unwahrscheinlich, jedoch sollte auf eine stabilere Lösung gesetzt werden. Im zuvor beschriebenen FHDR wird eine weitere Information bereitgestellt, mit Hilfe derer solche Schleifen

allgemeingültig unterbunden werden können[10]. Das FCnt-Feld, das in Abbildung 5.4 zu sehen ist, stellt eine Information darüber bereit, wie viele Pakete ein Sensorknoten bereits versendet hat. Der Sensorknoten inkrementiert dieses Feld eigenständig mit Hilfe eines internen Zählers. Somit entspricht FCnt einem Unique Identifier (UID) pro Paket eines Sensorknotens. Da das RN die Pakete nicht verändert, kann ein RN eine Zuordnung über die DevAddr und das FCnt-Feld vom letzten Paket speichern. Sobald ein neuer FCnt-Wert für die im FHDR hinterlegte DevAddr erkannt wird, kann das Paket weitergeleitet werden.

Auf Basis der in diesem Unterabschnitt beschriebenen Ansätze ist es möglich, ein LoRaWAN mit Multi-Hop-Knoten aufzubauen, sodass die Reichweite des Netzes erhöht wird. In Abschnitt 4.4 wurde, neben der zu geringen Reichweite des LoRaWANs, die Beeinträchtigung der PDR aufgrund von Interferenzen beschrieben. Interferenzen werden mit dem zuvor beschriebenen Ansatz unter Umständen sogar verschärft. Deshalb ist ein strukturierter Sendeplan unerlässlich, was im folgenden Abschnitt beschrieben wird.

5.2 Zeitschlitz-Erweiterung

In Anbetracht der Anzahl einsetzbarer Sensorknoten sowie der Größe eines möglichen Routing-Baumes, wie in Tabelle 5.5 exemplarisch gezeigt, wird deutlich, dass eine zeitliche Abstimmung für das Senden der Sensorknoten benötigt wird. Ein bekanntes Zugriffsverfahren, das eine Abstimmung in Form von Zeitschlitzen und unterschiedlichen Frequenzen vorsieht, ist das Verfahren Time-Slotted Channel Hopping (TSCH) im Standard IEEE 802.15.4 [48]. TSCH kombiniert die Zugriffsverfahren Time Division Multiple Access (TDMA) und Frequency Division Multiple Access (FDMA), wobei letzteres bereits in Unterabschnitt 5.1.3 für eine bessere Ausnutzung des Duty Cycles im Multi-Hop-Ansatz adaptiert wurde. Einige Arbeiten haben sich in der Vergangenheit bereits mit einem Zeitschlitz-Verfahren für LoRaWAN beschäftigt, wie beispielsweise die Arbeiten von Zorbas et al. [49, 50]. In beiden Arbeiten wird kritisiert, dass das Zugriffsverfahren TSCH zu umfangreich für Constrained Devices im Rahmen von LPWANs sei. Deshalb wird im Folgenden die Idee von TSCH weiter verfolgt, die Umsetzung jedoch insoweit vereinfacht, dass es für den in Unterabschnitt 5.1.3 beschriebenen Multi-Hop-Ansatz eingesetzt werden kann.

[10] Gemäß dem Fall, dass der LoRaWAN-Standard [3] wie gefordert implementiert wurde.

5.2.1 Paralleles Senden zu einem Zeitpunkt

Ein Vorteil beim Einsatz unterschiedlicher Frequenzen in einem LoRaWAN ist, dass parallel ohne Interferenzen gesendet bzw. empfangen werden kann. Daraus folgt wiederum, dass nicht nur innerhalb einer Region zu gleichen Zeitpunkten gesendet, sondern auch, dass aus zwei unterschiedlichen RN-Bäumen zu gleichen Zeitpunkten an das Gateway gesendet werden kann. Dieser Ansatz erhöht den Durchsatz der Pakete in einem Zeitraum. Es muss jedoch beachtet werden, dass der Grad der Parallelität begrenzt ist. Diese Begrenzung ist vor allem in der Anzahl Frequenzen festzustellen, auf denen ein RN bzw. ein Gateway parallel empfangen kann. Je mehr Parallelität vorzufinden ist, desto höher sind die Kosten für die eingesetzte Hardware. Wie bereits in Unterabschnitt 4.1.4 beschrieben, wird in der folgenden Betrachtung von vier parallelen Empfangspfaden an einem Gateway ausgegangen. Um die Kosten für ein RN möglichst gering zu halten, wird angenommen, dass ein RN maximal zwei parallele Empfangspfade besitzt. Aus dieser Entscheidung folgt, dass lediglich vier Sensorknoten zu einem Zeitpunkt aus zwei verschiedenen RN-Bäumen senden können, ohne dass Interferenzen am Gateway entstehen.

Die Aufteilung der Zeitschlitze der Sensorknoten kann auf verschiedene Arten erfolgen. So wäre es beispielsweise möglich, zu einem Zeitpunkt vier Sensorknoten aus vier Regionen senden zu lassen oder zwei Sensorknoten aus zwei Regionen. Abhängig ist diese Entscheidung von der Anzahl eingesetzter Subbänder. So wäre es beispielsweise nicht sinnvoll, wenn vier Sensorknoten aus einer Region gleichzeitig senden dürften, jedoch nur zwei Subbänder zur Verfügung stehen. Das würde bedeuten, dass ein RN die Pakete zunächst puffern müsste und erst nach der vorgeschriebenen Wartezeit wieder senden dürfte. Ein besserer Ansatz bei sechs RN-Bäumen und zwei zur Verfügung stehenden Subbändern ist, zwei Sensorknoten aus jeweils zwei RN-Bäumen zu einem Zeitpunkt senden zu lassen.

5.2.2 Frequenzzuteilung im Zeitschlitz

Die Zuteilung der Frequenzen an die Sensorknoten kann unter Berücksichtigung der Subbänder pro RN-Baum beliebig erfolgen. Des Weiteren kann die Frequenz des Sensorknotens unverändert über die Laufzeit bleiben, sofern die Zeit zwischen zwei Paketen pro Sensorknoten größer ist, als die durch den maximalen Duty Cycle bedingte Wartzeit. Ob eine Frequenz unverändert bleiben kann, kann mit Hilfe von Bedingung 5.12 überprüft werden, wobei t_{next} dem Abstand zwischen zwei zu versendenen Paketen pro Sensorknoten und t_w der Wartezeit zum Einhalten des Duty Cycles entspricht. Da im vorliegenden Anwendungsfall $t_{next} = 600\,s$ und $t_w \leq 1,155\,s \cdot 99$ gilt[11], ist Bedingung 5.12 erfüllt. Die Frequenz kann demnach pro Sensorknoten über die Laufzeit in dem hier behandelten Anwendungsfall unverändert bleiben.

$$t_{next} > t_w \tag{5.12}$$

5.2.3 Toleranzzeit pro Zeitschlitz

Um den Sensorknoten einen Zeitschlitz zuzuteilen, muss berücksichtigt werden, dass eine gewisse Toleranzzeit einkalkuliert werden muss. Die Toleranzzeit wird durch zwei Aspekte bestimmt:

- **Längster Pfad im Baum:** Dadurch, dass ein Paket durch mehrere RNs weitergeleitet werden kann und somit unterschiedliche Zeit für die Weiterleitung benötigt wird, kann es am Gateway zu Interferenzen kommen. Die Interferenzen sind dann wahrscheinlich, wenn die benötigte Zeit für das Routing bis zum Gateway größer ist, als die Zeit, bis die gleiche Frequenz in einem Zeitschlitz von einem anderen Sensorknoten erneut verwendet wird.
- **Ungenauigkeit der Zeit:** Ein Problem bei Betrachtung des Anwendungsfalls und der zeitsensitiven LoRaWAN-Erweiterung ist, dass die Sensorknoten jeglichen Witterungsverhältnissen ausgesetzt sind. Dadurch schwankt die Genauigkeit der Uhr der Sensorknoten.

[11] t_w ist bezogen auf Subbänder mit 1 % Duty Cycle und der für Tabelle 5.1 verwendeten Parameter, wobei die längste Wartezeit 1,155 s mit SF12 beträgt.

Um die Dauer der Übertragung für den längsten Pfad im Baum zu bestimmen, kann Gleichung 5.13 verwendet werden, wobei len_{path} der längste Pfad der RN-Bäume, t_{txRn} der Übertragungsdauer eines RNs und t_{txEd} der Übertragungsdauer eines Sensorknotens entspricht.

$$t_{max} = (len_{path} \cdot t_{txRn}) + t_{txEd} \tag{5.13}$$

In Anbetracht der berechneten Übertragungsdauer für verschiedene SFs in Tabelle 5.1 lässt sich berechnen, wie viel Zeit mindestens benötigt wird, bis eine Frequenz wieder genutzt werden kann. Wird beispielsweise SF7 für die Sensorknoten und SF9 für die RNs verwendet, so liegt bei einer Anzahl von sieben RNs in einem Routing-Baum eine maximale Pfadlänge von drei vor. Daraus lässt sich mit Hilfe von Gleichung 5.13 berechnen, dass 546,048 ms für den längsten Pfad benötigt werden, wobei $len_{path} = 3, t_{txRn} = 164,864\,ms$ und $t_{txEd} = 51,456\,ms$ (vgl. Tabelle 5.1). Demnach müssen mindestens 546,048 ms vergehen, bis die gleiche Frequenz wieder verwendet werden kann.

Die vorherige Betrachtung zeigt, dass die Toleranzzeit für das Verwenden einer gleichen Frequenz für die meisten Anwendungsfälle recht gering ausfällt. Ein größeres Problem liegt in der Ungenauigkeit des eingesetzten Schwingquarzes der Real Time Clock (RTC), aus dem die Uhrzeit abgeleitet wird. Typischerweise werden für zeitsensitive Anwendungen Schwingquarze mit einer Frequenz von 32768 kHz verwendet, da sich ein Takt von einer Sekunde beim Teilen der Frequenz durch 2^{15} ergibt ([51], S. 195).

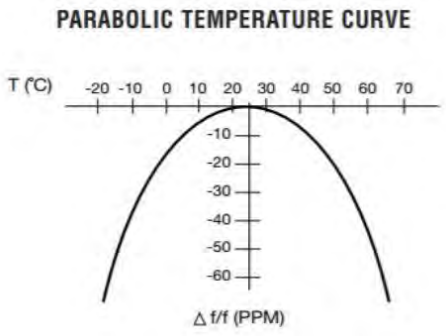

Abbildung 5.5: Parabelförmige Frequenzabweichung bei sich ändernder Temperatur für einen ECS-.327 Schwingquarz (vgl. [52])

Frequenzänderungen des Schwingquarzes durch Temperaturunterschiede entsprechen meist einer Parabel, wobei die Referenztemperatur mit der geringsten Abweichung meist bei 25 °C liegt ([53], S. 270). In Abbildung 5.5 ist die Frequenzabweichung bei-

spielhaft für einen ECS-.327 32768 kHz Schwingquarz zu sehen. Auf der Ordinate ist die Abweichung in parts per million (ppm) und auf der Abszisse die Temperatur in Grad Celsius zu sehen. Bei circa -20 °C bzw. circa 62 °C liegt die Abweichung für diesen Schwingquarz bei -60 ppm. Mit Hilfe von Gleichung 5.14 kann berechnet werden, dass dieser Schwingquarz bei den zuvor genannten Temperaturen etwa 5 s Abweichung pro Tag aufweisen kann, wobei $f_{deviation}$ der Abweichung in ppm entspricht.

$$t_{deviation} = f_{deviation} \cdot 10^{-6} \cdot (60\,s \cdot 60\,min \cdot 24\,h) \tag{5.14}$$

Summiert sich diese Abweichung über mehrere Tage und für einige Sensorknoten insbesondere aus verschiedenen Regionen unterschiedlich, so wäre das Zeitschlitzverfahren nicht mehr anwendbar. Des Weiteren degenerieren Quarze über eine längere Laufzeit, sodass sich die Genauigkeit weiter verschlechtert. Durch die entstehenden Abweichungen könnten im schlechtesten Fall die gesetzlichen Vorgaben bezüglich des maximalen Duty Cycles insbesondere bei den RNs verletzt werden, was nicht toleriert werden kann. Zwar ist eine Temperatur von 62 °C unter normalen Umständen in einem Wald unwahrscheinlich, dennoch sind besonders tiefe Temperaturen durchaus möglich. Dieses Problem muss demnach genauer betrachtet werden.

Es gibt unterschiedliche hardware- und softwareseitige Ansätze, um die Frequenz des Schwingquarzes in verschiedenen Temperaturbereichen stabiler zu halten. So ist es beispielsweise mit Hilfe eines Thermostats möglich, die Temperatur des Schwingquarzes für eine bessere Stabilisierung der Frequenz annähernd konstant zu halten ([53], S. 518). Schwingquarze mit einem Thermostat werden Temperaturstabilisierte Quarzoszillatoren (OCXO) genannt. Ein Nachteil der OCXO ist der hohe Strombedarf von teilweise 100 mA, was für die angestrebten Sensorknoten nicht akzeptabel ist. Andere Ansätze verfolgen eine Kompensation mit Hilfe von Temperaturmessungen und davon abhängiger analoger oder digitaler Frequenzanpassung. Solche Schwingquarze werden als Temperaturkompensierte Quarzoszillatoren (TCXO) bezeichnet. Ob ein OCXO, ein TCXO oder ein normaler Schwingquarz eingesetzt werden sollte, ist abhängig vom Anwendungsfall, den zeitlichen Restriktionen und dem Einsatzort.

Ein anderer Ansatz ist die Synchronisation der Uhren der Sensorknoten. Eine häufig eingesetzte Möglichkeit zur Zeitsynchronisation bietet sich durch Zeitzeichensender an. Zeitzeichensender senden meist auf niedriger Frequenz mit hoher Leistung in regelmäßigen Abständen Informationen zur Uhrzeit. In Deutschland wird mit dem Zeitzeichensender DCF77 auf 77,5 kHz mit 50 kW einmal pro Minute gesendet. Die Reichweite des Senders, der in Mainflingen stationiert ist, beträgt tagsüber etwa 1900 km

und in der Nacht etwa 2100 km. Es existieren weltweit Zeitzeichensender, die ähnliche Parameter aufweisen. [54] Da Funkuhren Zeitzeichensender zur Synchronisation nutzen und handelsübliche Funkuhren als Massenprodukt einzustufen sind, ist Hardware zum Empfangen und Demodulieren der Funksignale recht günstig[12]. Des Weiteren ist die benötigte Zeit zum Öffnen eines Empfangsfensters recht gering, schließlich wird, zumindest vom DCF77, jede Minute eine Synchronisation ermöglicht. Der Strombedarf liegt im aktivierten Zustand bei handelsüblichen DCF77-Modulen meist unter 100 µA[13].

Ein weiterer Ansatz zur Zeitsynchronisation ist mit Hilfe von GPS-Empfängern möglich. GPS-Satelliten besitzen Atomuhren, da Zeit die Grundlage für eine Positionsbestimmung mit Hilfe von GPS ist. Wird für den gewünschten Anwendungsfall die Position eines Sensorknotens via GPS ermittelt, so kann die Synchronisation der Zeit direkt aus den empfangenen NMEA-Datensätzen stattfinden.

Ebenso wäre es denkbar, eine Synchronisation der Uhren protokollseitig zu bestimmten Zeitpunkten durch das Gateway zu initiieren und über die RNs weiterzutragen. Eine erste Schwierigkeit fällt beim Betrachten der Dauer der Übertragung und der vom Gateway ermittelten Uhrzeit auf. Im Prinzip müsste jeder Sensorknoten wissen, wie viele RNs das Paket zur Synchronisation weitergeleitet haben, um so die Uhrzeit um die Dauer der Übertragung pro Weiterleitung anzupassen. Dies könnte recht einfach gelöst werden, indem das RN einen Zähler im nicht verschlüsselten Teil des Paketes inkrementiert. Diesen Zähler könnte der Sensorknoten auslesen und mit der gemessenen Dauer der Übertragung verrechnen. Das Resultat wäre vermutlich genau genug, sofern in regelmäßigen Abständen synchronisiert wird. Das größere Problem ist jedoch, dass das Öffnen von Empfangsfenstern bei den Sensorknoten den Energieverbrauch erhöht. Durch die potentiell abgewichene Uhrzeit muss das Empfangsfenster zusätzlich mit einer gewissen Toleranzzeit geöffnet werden, sodass beim Warten auf ein Paket viel Energie[14] verbraucht wird. Die Kombination aus dem erhöhten Energieverbrauch durch grobe Synchronisationsfenster und dem Erhöhen der Komplexität der Protokoll-Erweiterung scheinen keine gute Lösung zu bieten.

Die vorherigen Erklärungen zeigen, dass insbesondere hardwareseitig Möglichkeiten existieren, ausreichend genaue Uhrzeiten an den Sensorknoten vorauszusetzen. Wie zuvor erläutert, soll protokollseitig keine Synchronisation der Uhren vorgenommen

[12] Fertige Module für Empfang und Demodulation einer DCF77-Nachricht kosten zum Zeitpunkt der Thesis weniger als 5 €. Es ist davon auszugehen, dass die einzelnen Hardware-Komponenten weit weniger kosten.

[13] Beispielhaftes Datenblatt eines DCF77-Moduls: https://cdn-reichelt.de/documents/datenblatt/C300/SPEZIFIKATIONF.NR.9500511.pdf (Abgerufen am 2021-02-18)

[14] Beispielsweise benötigt ein Semtech SX1276 LoRaWAN-Modul etwa 13 mA im RX-Zustand [37].

werden, da die Nachteile überwiegen und Flexibilität bezüglich des Anwendungsfalls genommen werden würde. In Bezug zum Anwendungsfall der Waldbranderkennung sorgen die Temperaturschwankungen pro Tag im Wald für einen gewissen Ausgleich in der Genauigkeit des Schwingquarzes (vgl. [55], S. 5). Des Weiteren können zeitliche Synchronisationen, je nach Einsatzort, mit Hilfe eines Zeitzeichensenders vorgenommen werden. Zusätzlich sollte beim Einsatz der LoRaWAN-Erweiterung evaluiert werden, inwieweit der gegebene Anwendungsfall die Stabilisierung der Frequenz des Schwingquarzes erfordert. Möglicherweise genügt ein herkömmlicher Schwingquarz für bestimmte Anwendungsfälle. Diese Aspekte müssen im Rahmen der tatsächlichen Umsetzung evaluiert werden.

5.2.4 Zuteilung der Zeitschlitze

Zusammenfassend lässt sich feststellen, dass vor allem eine genügend große Toleranzzeit die Zuteilung der Zeitschlitze beeinflusst. Im Folgenden soll ein Beispiel verdeutlichen, wie eine Zuteilung der Zeitschlitze auf Basis der zuvor erläuterten Ansätze erfolgen kann.

Dieses Beispiel basiert auf Tabelle 5.1, wobei für die RNs SF9 und für die Sensorknoten SF7 festgelegt wurde. Insgesamt soll mit vier Frequenzen auf zwei Subbändern gearbeitet werden. Des Weiteren steht ein TCXO-Quarz mit ±5 ppm Abweichung nach Korrektur, also einer theoretischen Abweichung von ±432 ms pro Tag, zur Verfügung. Die Toleranzzeit kann nun wie folgt betrachtet werden: In einem Zeitschlitz senden zwei Sensorknoten aus zwei RN-Bäumen mit zwei Frequenzen, die pro RN-Baum unterschiedlichen Subbändern zuzuordnen sind. Es stehen demnach vier Frequenzen in zwei Subbändern zur Verfügung, wobei die Frequenzen im nächsten Zeitschlitz wiederverwendet werden müssen. Die maximale Wartezeit zwischen beiden Zeitschlitzen entspricht dann der Summe aus der Dauer der Übertragung auf dem längsten Pfad im Baum und der Toleranzzeit für die Ungenauigkeit des Quarzes. Soll beispielsweise jeden Tag eine Zeitsynchronisation mit einem Zeitzeichensender vorgenommen werden, so wären ±432 ms Abweichung möglich. Da die Uhr eines Sensorknotens im schlechtesten Fall 432 ms vor der globalen Zeit des LoRaWANs und die Uhr eines anderen Sensorknotens 432 ms nach der globalen Zeit des LoRaWANs laufen kann, muss die Abweichung verdoppelt werden, was eine Toleranzzeit von 864 ms ergibt. Da für die RNs SF9 und für die Sensorknoten SF7 angenommen wird, ergibt sich aus Gleichung 5.13 und Tabelle 5.1 eine Übertragungsdauer auf dem längsten Pfad, welcher eine Länge von drei RN hat, von 546,048 ms. Somit müssen 1410,048 ms zwischen zwei Zeitschlitzen gewartet werden, wobei der Einfachheit halber

auf 1500 ms aufgerundet werden kann. Mit der vorgestellten Berechnung der Toleranzzeit zwischen zwei Zeitschlitzen ist es somit unwahrscheinlich, dass es zu Interferenzen am Gateway aufgrund der Verwendung gleicher Frequenzen bei zwei aufeinanderfolgenden Zeitschlitzen kommt. Da nach dem Senden in einem RN-Baum der maximale Duty Cycle eingehalten werden muss, darf aus einem RN-Baum erst nach der Wartezeit erneut gesendet werden. Die Wartezeit für die RNs mit SF9 entspricht nach Tabelle 5.1 16,33 s, was auf 16,5 s aufgerundet werden kann. Somit darf pro RN-Baum nur alle 16,5 s gesendet werden, da andernfalls mindestens das RN im ersten Ring gegen die Duty Cycle-Restriktionen verstößt. In Tabelle 5.6 sind die ersten vier Iterationen der Zeitschlitze zu sehen. Nach dem Senden muss jedes RN-Baum-Paar die zuvor beschriebenen 16,5 s warten, um den maximalen Duty Cycle einzuhalten. In jeder Zeile ist zwischen den Zeitschlitzen (also zwischen den Spalten) die zuvor beschriebene Toleranzzeit von 1500 ms addiert. Es muss beachtet werden, dass die Zeit zwischen dem Senden des letzten RN-Baum-Paares und des ersten RN-Baum-Paares in einer neuen Iteration mindestens der Toleranzzeit entspricht.

Iteration	RN-Baum 1 & 2	RN-Baum 3 & 4	RN-Baum 5 & 6
1	0 ms	1500 ms	3000 ms
2	16500 ms	18000 ms	19500 ms
3	33000 ms	34500 ms	36000 ms
4	49500 ms	51000 ms	52500 ms

Tabelle 5.6: Die ersten vier Iterationen für Zeitschlitze bei sechs RN-Bäumen mit vier Frequenzen aufgeteilt auf zwei Subbändern

Das Beispiel zeigt, dass die Aufteilung der Zeitschlitze vom Anwendungsfall und den verschiedenen Parametern abhängig ist. Der Ablauf lässt sich wie folgt zusammenfassen:

1. Berechnen der Wartezeit zum Einhalten des Duty Cycles pro Region, was abhängig von Einsatzort und Subband ist.
2. Berechnen der Übertragungsdauer für ein Paket auf dem längsten Pfad im Routing-Baum.
3. Berechnen der Zeit, die im schlechtesten Fall durch Aufsummieren der Abweichung des Quarzes über einen bestimmten Zeitraum entsteht.
4. Toleranzzeit zwischen zwei Zeitschlitzen berechnen, indem Ergebnisse aus 2. und 3. addiert werden.
5. Innerhalb einer Iteration muss die in 4. berechnete Zeit zwischen zwei Zeitschlitzen gewartet werden.

6. Nach einer Iteration muss die in 1. ermittelte Zeit gewartet werden, um den maximalen Duty Cycle für die RNs einzuhalten.

Sensorknoten, die der Gateway-Region zugeordnet sind, sind von den Restriktionen des Duty Cycles der RNs befreit und passen somit nur noch teilweise in den vorher beschriebenen Ablauf. In der Gateway-Region sind lediglich die Restriktionen des Duty Cycles der Sensorknoten sowie die Toleranzzeit zu berücksichtigen. Die Auswahl eines passenden Zeitschlitzes für die Gateway-Region ist demnach flexibel möglich. Im zuvor beschriebenen Beispiel wäre es empfehlenswert, den Sensorknoten der Gateway-Region einen Zeitschlitz zwischen den Iterationen und somit zwischen der durch den Duty Cycle bedingten Wartzeit zuzuteilen. Vor dem Senden der Sensorknoten der Gateway-Region muss jedoch die Toleranzzeit gewartet werden, da die Frequenzen erneut verwendet werden sollen. Danach können pro Zeitschlitz vier Sensorknoten gleichzeitig senden. Dieser Ablauf kann so lange durchgeführt werden, bis die neue Iteration durch die RN-Bäume beginnt, wobei die Toleranzzeit auch am Ende der Wartezeit zwischen den Iterationen berücksichtigt werden muss. Es ist zu beachten, dass die Dauer der Übertragung des Sensorknotens und die Toleranzzeit nach dem letzten RN-Baum-Paar und vor dem ersten RN-Baum-Paar der neuen Iteration eingehalten wird. Dies ist in Tabelle 5.6 der Fall.

Es wird deutlich, dass eine Abhängigkeit zum Anwendungsfall und der restlichen Parameter gegeben ist und somit keine allgemeingültige Zuteilung der Zeitschlitze der Gateway-Sensorknoten stattfinden kann. Eine weitere Bedingung bei der Zuteilung der Zeitschlitze ist in Bedingung 5.15 formuliert. Bedingung 5.15 besagt, dass die Summe aus dem betrachteten relativen Zeitschlitz, der Übertragungsdauer und der Toleranzzeit für keinen Sensorknoten größer als die Zeit zwischen zwei Paketen sein darf. Dieses Problem tritt auf, wenn zu viele Sensorknoten einen Zeitschlitz erhalten sollen.

$$(ts + t_{tx} + t_{toleranz}) \leq tx_{intervall} \tag{5.15}$$

Sofern es nicht möglich ist, die Zeitschlitze innerhalb des gewünschten Intervalls zu platzieren oder den Gateway-Sensorknoten einen Zeitschlitz in der gegebenen Konfiguration zuzuteilen, so kann beispielsweise der zeitliche Abstand zwischen zwei Paketen eines Sensorknoten erhöht, die Toleranzzeit durch geeignete Maßnahmen verkürzt oder die Anzahl der Sensorknoten insgesamt verkleinert werden.

Zur besseren Einordnung wird die vorgestellte Protokoll-Erweiterung im Folgenden als Time-slotted Region-based Extension for LoRaWAN (TiReX) bezeichnet.

5.3 Multi-Hop-Erweiterung in ns3

In dem LoRaWAN-Modul von Van den Abeele et al. [8] wird, wie in Abschnitt 3.2 beschrieben, zwischen EndDevices und Gateways unterschieden. EndDevices entsprechen in der bisherigen Terminologie den Sensorknoten und haben eine PHY- und eine MAC-Instanz. Ein Gateway hat mehrere Kombinationen aus PHY- und MAC-Instanzen, um parallel auf verschiedenen Frequenzen LoRaWAN-Nachrichten empfangen zu können. Des Weiteren besitzen Gateways eine Schnittstelle zu einem NetworkServer. Da ein RN ebenso parallel mehrere Nachrichten empfangen soll, erscheint es sinnvoll, die Gateway-Implementierung als Grundlage für die RN-Implementierung zu verwenden. Lediglich auf die NetworkServer-Schnittstelle kann bei einem RN verzichtet werden. Im Folgenden werden die wichtigsten Änderungen im ns3-Modul von Van den Abeele et al. vorgestellt.

5.3.1 Anwendung der Relay Nodes

Als Grundlage der RNs wurde, wie zuvor beschrieben, die Gateway-Anwendung verwendet. Um eine Unabhängigkeit zwischen den RNs und dem Gateway zu schaffen, musste ein neuer Geräte-Typ in den Simulator integriert werden. Deshalb wurde das ns3-Modul von Van den Abeele et al. [8] um einen Geräte-Typ für RNs erweitert, was in Listing 5.1 in Zeile 7 zu sehen ist. Durch den neuen Gerätetyp war es möglich, Aktionen, die von der Anwendung in einer unterliegenden Schicht ausgelöst werden, von einem Gateway zu unterscheiden.

Listing 5.1 Auflistung aller verfügbaren Gerätetypen im erweiterten LoRaWAN-Modul von Van den Abeele et al. [8]

```
1 typedef enum
2 {
3     LORAWAN_DT_GATEWAY = 0,
4     LORAWAN_DT_END_DEVICE_CLASS_A,
5     LORAWAN_DT_END_DEVICE_CLASS_B,
6     LORAWAN_DT_END_DEVICE_CLASS_C,
7     LORAWAN_DT_RELAY_NODE,
8     LORAWAN_DT_UNDEF,
9 } LoRaWANDeviceType;
```

Der Ablauf einer RN-Anwendung ähnelt zunächst dem einer Gateway-Anwendung. Das RN ist mit allen PHY im Zustand *RX_ON*, wartet also auf Nachrichten. Ein Unterschied ist beim Überprüfen eines Pakets in der RN-Anwendung festzustellen, da der Ablauf von TiReX aus Unterabschnitt 5.1.3 implementiert werden soll. Sobald LoRa-Signale vom PHY empfangen, demoduliert und das Paket durch die MAC-Schicht und das NetDevice an die RN-Anwendung weitergereicht wurde, wird überprüft, ob die Kombination aus FCnt und DevAddr bereits bekannt ist. Diese Überprüfung ist in Listing 5.2 zu sehen. In Listing 5.2 wird in Zeile 2 zunächst ermittelt, ob Pakete von dieser Adresse bereits empfangen wurden. Ist dies der Fall, wird in Zeile 4 überprüft, ob der FCnt-Wert für die Adresse bereits bekannt ist. Ist dies der Fall, wird das Paket im weiteren Verlauf verworfen. Andernfalls wird der neue FCnt-Wert für die gegebene Adresse gespeichert.

Listing 5.2 Ausschnitt der RN-Anwendung zum Verhindern einer Routing-Schleife

```
1  bool isKnownPacket = false;
2  if (addressFcntMapping.find(address) != addressFcntMapping.end())
3  {
4      if (addressFcntMapping[address] == fhdr.getFrameCounter())
5          isKnownPacket = true;
6      else
7          addressFcntMapping[address] = fhdr.getFrameCounter();
8  }
9  else
10     addressFcntMapping[address] = fhdr.getFrameCounter();
```

Im zweiten Schritt muss ermittelt werden, ob das empfangene Paket von einem dem RN zugeordneten Sensorknoten oder einem RN-Kind stammt. Dieser Ablauf ist in Listing 5.3 zu sehen[15]. Für diesen Zweck wird über eine Liste der dem RN bekannten Sensorknoten iteriert. In dieser Liste sind alle Adressen der dem RN zugeordneten Sensorknoten sowie die Adressen der Sensorknoten der RN-Kinder enthalten. Der Algorithmus zur Zuordnung der RN-Kinder wird im späteren Verlauf in Unterabschnitt 5.3.2 beschrieben. Wird die gesuchte Adresse gefunden, so handelt es sich um einen Sensorknoten des RNs oder um eine Weiterleitung durch ein RN-Kind. Entsprechend wird in Zeile 12 von Listing 5.3 geprüft, ob die Absenderadresse des empfangenen Paketes vom RN akzeptiert wird

[15]Da die Simulation im Umfang recht gering ist, kann auf Optimierungen hinsichtlich der Performanz weitestgehend verzichtet werden. Die vorgestellten Implementierungen lassen sich aber durchaus optimieren.

und ob das Paket dem RN bisher unbekannt ist. Ist dies der Fall, wird eine Weiterleitung des Paketes veranlasst. Andernfalls wird das Paket verworfen.

Listing 5.3 Ausschnitt der RN-Anwendung zum Zuordnen der empfangenen LoRaWAN-Nachricht

```
bool isChild = false;
for (const auto& ed: m_edChilds)
{
    auto a = ed->GetDevice(0)->GetAddress();
    if (a == address)
    {
        isChild = true;
        break;
    }
}

if (isChild && !isKnownPacket)
{
    relayPacket = packet->Copy();
    m_txEvent = Simulator::ScheduleNow (&SendPacket, this);
}
```

Wie in Unterabschnitt 5.1.3 beschrieben, müssen RNs die eigenen Sensorknoten und die Sensorknoten der RN-Kinder kennen. Aus diesem Grund stellt die RN-Anwendung zwei zusätzliche Funktionen bereit, die in Listing 5.4 zu sehen sind. Mit der Funktion `AddEdChilds(Ptr<Node> n)` kann ein Sensorknoten dem RN zugeordnet werden. Die Funktion `GetEdChilds()` liefert wiederum die Liste über die zugeordneten Sensorknoten. So können die in Unterabschnitt 5.1.3 beschriebenen Elter-Kind-Beziehungen der RNs beim Empfang eines Paketes überprüft werden.

Listing 5.4 Funktionen der RN-Anwendung, die eine Schnittstelle zu den zugeordneten Sensorknoten der RN-Kinder liefern

```
1 void LoRaWANRelayNodeApplication::AddEdChild(Ptr<Node> n)
2 {
3     m_edChilds.push_back(n);
4 }
5
6 std::vector<Ptr<Node>> LoRaWANRelayNodeApplication::GetEdChilds()
7 {
8     return m_edChilds;
9 }
```

Aus Unterabschnitt 5.1.3 wird deutlich, dass die RNs im äußerten Ring lediglich ihre eigenen Sensorknoten in der in Listing 5.4 gezeigten Liste `m_edChilds` haben können, schließlich besitzen RNs im äußersten Ring keine RN-Kinder. In der Liste `m_edChilds` eines RNs im ersten Ring sind wiederum alle Sensorknoten des Baumes vorzufinden, da die RNs im ersten Ring jeweils der Wurzel eines RN-Baumes entsprechen.

5.3.2 Anordnung der Relay Nodes

Da eine RN-Anwendung zur Verfügung steht, kann im nächsten Schritt die Anordnung der RNs umgesetzt werden. Zunächst müssen die RN-Knoten gleichmäßig auf die einzelnen Bäume, ausgehend von den Knoten des ersten Rings, verteilt werden. Hierfür müssen die Koordinaten, ausgehend vom Gateway, bestimmt werden. Da im Rahmen von Unterabschnitt 5.1.3 Sechsecke als Berechnungsgrundlage verwendet wurden und ein Sechseck die Fläche der in etwa kreisförmigen Propagation eines LoRa-Signals zu einem großen Teil abdeckt[16], wird eine solche Anordnung auch im Kontext der Simulation verwendet. Um Nachbar-Koordinaten von einem RN zu finden, können sechs Fälle unterschieden werden. Diese Fälle sind in Listing 5.5 zu sehen. In den Übergabeparametern der Funktion `GetNeighbourCoordinates` entsprechen `x_ref` und `y_ref` den Koordinaten, von denen die Nachbarn ermittelt werden sollen, `r` dem größten Radius im Sechseck[17] und `t` der Position des Nachbarn (vgl. Abbildung 5.6).

[16] Ein Sechseck ist bei gleichem Radius flächenmäßig kleiner als ein Kreis. Des Weiteren wird vorausgesetzt, dass eine Rundstrahlantenne eingesetzt wird.

[17] Der größte Radius entspricht der halbierten maximalen Reichweite eines betrachteten SF aus Tabelle 3.1.

Listing 5.5 Funktion zum Ermitteln der Koordinaten des Zentrums eines Nachbar-RNs

```
Vector LoRaWANRelayNodeApplication::GetNeighbourCoordinates(float
↪  x_ref, float y_ref, uint32_t r, NeighbourType t)
{
    float x, y;
    if(t == TOP || t == BOTTOM)
    {
        x = x_ref;
        if (t == TOP)
            y = y_ref - 2 * GetDistance(r);
        else
            y = y_ref + 2 * GetDistance(r);
    }
    else if (t == TOP_RIGHT || t == BOTTOM_RIGHT)
    {
        x = x_ref + (r + r/2);
        if (t == TOP_RIGHT)
            y = y_ref - GetDistance(r);
        else
            y = y_ref + GetDistance(r);
    }
    else
    {
        x = x_ref - (r + r/2);
        if (t == BOTTOM_LEFT)
            y = y_ref + GetDistance(r);
        else
            y = y_ref - GetDistance(r);
    }
    return Vector(x, y);
}
```

Die in Listing 5.5 gezeigten Fälle berechnen die jeweiligen Koordinaten des benachbarten RNs. Die möglichen Nachbarn eines RNs, welche in Listing 5.5 als `NeighbourType` bezeichnet werden, sind in Abbildung 5.6 für die Referenz-Region in der Mitte zu sehen.

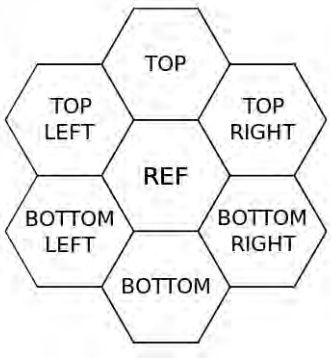

Abbildung 5.6: Mögliche Nachbarschaftstypen eines RNs

Da für das spätere Routing die Elter-Kind-Beziehungen zwischen den RNs bekannt sein müssen, kann bereits während der Berechnung der Koordinaten die Elter-Kind-Beziehung zwischen den RNs festgehalten werden. Für diesen Zweck werden demnach für jedes RN die Kinder mit Hilfe des in Listing 5.5 gezeigten Verfahrens berechnet.

Für die Berechnung der Koordinaten der RNs wird eine Warteschlange, bestehend aus RN-Koordinaten und einer Zuordnung zum jeweiligen Routing-Baum, eingesetzt. Da das Gateway kein RN ist und der Ablauf für die Berechnung der RNs im ersten Ring ein Sonderfall ist, enthält die Warteschlange initial die RNs des ersten Rings. Der Ablauf ist so definiert, dass das vorderste Element aus der Warteschlange entfernt und dessen Kinder berechnet werden. Für das aktuelle RN werden dann im Uhrzeigersinn, gestartet bei dem `NeighbourType` TOP, die RN-Kinder bestimmt. Die Suche der RN-Kinder für das betrachtete RN ist dann beendet, wenn zwei neue RNs gefunden wurden. Die jeweils neu berechneten RNs werden dann der Warteschlange hinzugefügt. Eine Ausnahme der RN-Kind-Berechnung tritt dann ein, wenn die Anzahl gefundener RNs im Routing-Baum gleich der durch Gleichung 5.10 ermittelten maximalen Anzahl von RNs pro Baum ist. Trifft die beschriebene Ausnahme zu, wird dem Routing-Baum kein weiteres RN-Kind zugeordnet. In Unterabschnitt 5.1.3 wurde beschrieben, dass eine Aufteilung der RNs bei einem unvollständigen letzten Ring nach Bedarf umgesetzt werden kann. In der ns3-Simulation wird in diesem Fall im Uhrzeigersinn über die Routing-Bäume iteriert, bis ein Baum gefunden wird, der eine weitere Region aufnehmen kann. Der Algorithmus terminiert, wenn alle durch Gleichung 5.6 berechneten Regionen gefunden wurden.

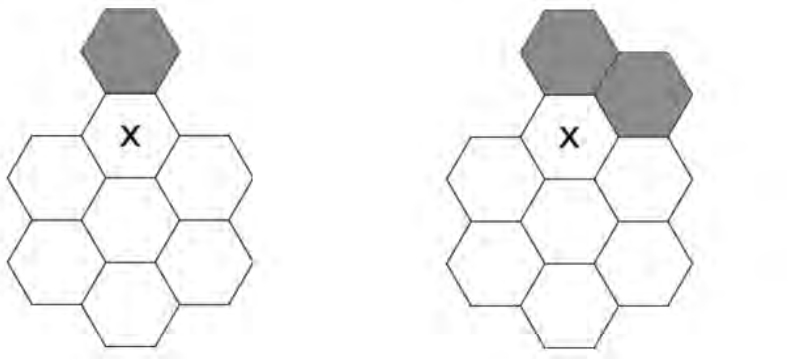

Abbildung 5.7: Exemplarisches Berechnen von benachbarten RNs

In Abbildung 5.7 ist der Ablauf des Algorithmus für ein RN des ersten Rings exemplarisch gezeigt. Es wird davon ausgegangen, dass die RNs im ersten Ring bereits berechnet wurden. Deshalb werden im nächsten Schritt die Nachbarn des mit dem Kreuz markierten RNs ermittelt. Zunächst wird die oberste Region (*TOP*) gefunden, wobei im Anschluss das RN oben rechts (*TOP RIGHT*) ermittelt wird. Da zwei RNs gefunden wurden, wird das nächste RN des ersten Rings im Uhrzeigersinn ausgewählt und dessen Nachbarn berechnet.

5.3.3 Zuordnung von Sensorknoten zu Relay Nodes

Nachdem die Koordinaten der RNs ermittelt und die RN-Bäume erstellt wurden, müssen den RNs die Sensorknoten zugeordnet werden. Für das Überprüfen der entwickelten Erweiterungen im Simulator ns3 ist es zunächst irrelevant, wie die Sensorknoten in einer Region angeordnet werden. Die einzige Bedingung ist, dass das RN die Nachrichten der Sensorknoten empfangen kann. Somit muss bei der Konfiguration des Simulators ein geeigneter SF für die Sensorknoten verwendet werden. Für die Positionierung wird ein in ns3 vordefinierter Algorithmus, der sogenannte *UniformDiscPositionAllocator*, verwendet. Dieser Algorithmus ordnet die Sensorknoten innerhalb der Region zufällig auf Basis eines definierbaren Radius an.

Da in Unterabschnitt 5.3.2 die Elter-Kind-Beziehungen der RNs bereits festgehalten wurden, ist das Zuordnen der Sensorknoten im Baum trivial. Für die Zuordnung muss über alle aufgebauten Bäume rückwärts traversiert werden und dem RN-Elter mit Hilfe der in Listing 5.4 gezeigten Funktionen die Sensorknoten des RN-Kinds zugeordnet werden. Durch diesen Ablauf kennt das Wurzel-RN am Ende die Adressen aller Sensorknoten seines Baumes.

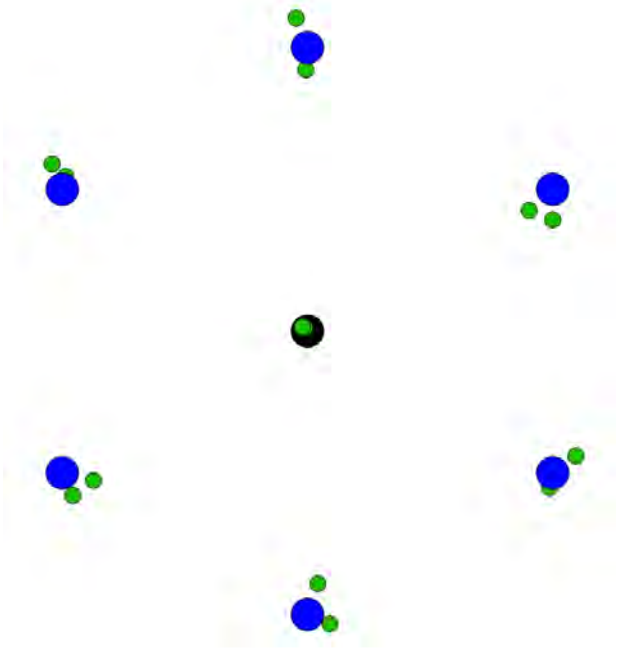

Abbildung 5.8: Anordnung in einem Simulationsdurchlauf mit sechs RNs, einem Gateway und 14 Sensorknoten

In Abbildung 5.8 ist eine beispielhafte Anordnung von 14 Sensorknoten auf sieben Regionen zu sehen, wobei das Gateway in der Mitte platziert ist. Die RNs sind in Abbildung 5.8 die größeren, um das Gateway positionierten Punkte und wurden mit Hilfe des in Unterabschnitt 5.3.2 beschriebenen Algorithmus angeordnet. Die Sensorknoten wiederum sind die kleineren Punkte und wurden durch den zuvor erläuterten *UniformDiscPositionAllocator*-Algorithmus positioniert. Abbildung 5.8 wurde im Rahmen eines Simulationsdurchlaufs erstellt und mit Hilfe des ns3-Visualisierungstools NetAnim[18] erzeugt.

[18]Siehe NetAnim-Dokumentation: https://www.nsnam.org/wiki/NetAnim_3.108 (Abgerufen am 2021-02-26)

5.4 Implementieren der Zeitschlitz-Erweiterung in ns3

Um die Zeitschlitze und die Frequenzen für jeden Sensorknoten setzen zu können, muss so lange über alle RN-Bäume bzw. Wurzel-RNs im ersten Ring iteriert werden, bis alle Sensorknoten eingeordnet wurden. Es ist ausreichend, die RNs des ersten Rings zu betrachten, da in Unterabschnitt 5.3.3 die Elter-Kind-Beziehung zwischen den RNs festgelegt wurde und alle Sensorknoten der RN-Kinder formal auch die Sensorknoten der RN-Elter sind. Daraus folgt, dass das RN im ersten Ring alle Sensorknoten seiner RN-Kinder kennt. Für die Simulation kann angenommen werden, dass das Gateway vier und die RNs zwei parallele Empfangspfade besitzen. Diese Aufteilung wurde bereits in den vorherigen Abschnitten verwendet. Demnach können, wie in Unterabschnitt 5.3.3 erläutert, vier Sensorknoten aus zwei Regionen gleichzeitig auf vier verschiedenen Frequenzen senden. Die vier Frequenzen müssen entsprechend auf mindestens zwei Subbänder je Region aufgeteilt sein, sodass der Duty Cycle eingehalten wird.

5.4.1 Algorithmus zum Setzen der Zeitschlitze und Frequenzen

Listing 5.6 Algorithmus zum Setzen der Zeitschlitze und Frequenzen für die Sensorknoten der RN-Bäume in vereinfachter Form

```
1  uint32_t childsDone = SetTimeSlotsGateway();
2  uint32_t treeCounter = 0;
3  while (childsDone < m_nEndDevices) {
4      for (size_t j = 0; j < 2; j++) {
5          localChildIndex = globalChildIndex;
6          auto edApps = GetChilds(treeCounter, j);
7           for (size_t i = 0; i < 2; i++) {
8              if (localChildIndex < edApps.size()) {
9                  auto ed = edApps.at(localChildIndex);
10                 ed->SetChannelIndex(GetChannel(j, i));
11                 ed->SetTimeSlot(GetTimeSlot(treeCounter));
12                 ++childsDone;
13             }
14             ++localChildIndex;
15          }
16     }
17     if (++treeCounter > 2) {
18         treeCounter = 0;
19         globalChildIndex += 2;
20     }
21 }
```

In Listing 5.6 ist die in Abschnitt 5.2 vorgestellte Aufteilung der Zeitschlitze und Frequenzen für zwei Subbänder gezeigt. Zur besseren Übersicht wurden einige Abfragen entfernt, wie beispielsweise das Prüfen, ob der erste Ring in Gänze mit RNs versehen ist. In Zeile 1 werden die Zeitschlitze der Gateway-Sensorknoten zunächst gesetzt, um im Anschluss mit dem ersten RN-Baum, der mit `treeCounter` indiziert wird, zu starten. Der Algorithmus läuft so lange, bis alle Sensorknoten einen Zeitschlitz und eine Frequenz erhalten haben, wie in Zeile 3 zu sehen. Innerhalb dieser Schleife gibt es wiederum zwei verschachtelte Schleifen. Die äußere Schleife betrachtet die RN-Bäume, in denen die Sensorknoten den gleichen Zeitschlitz erhalten sollen. Die innere Schleife betrachtet die Sensorknoten, die innerhalb eines der RN-Bäume den gleichen Zeitschlitz erhalten sollen. Da zwei Subbänder pro Zeitschlitz und pro RN-Baum zur Verfügung stehen, wird die innere Schleife zwei mal durchlaufen. Aufgrund dessen, dass in zwei RN-Bäumen immer

zwei Sensorknoten pro Zeitschlitz betrachtet werden, wird zwischen einem lokalen und einem globalen Index unterschieden. Der lokale Index wird nach der Betrachtung eines RN-Baumes wieder auf den globalen Index zurückgesetzt, was in Zeile 5 zu sehen ist. So ist sichergestellt, dass durch die Indizierung alle Sensorknoten eingeordnet werden. In Zeile 6 werden die Kinder des RN-Baumes ermittelt. Damit unterschieden werden kann, welcher RN-Baum betrachtet werden soll, wird neben dem `treeCounter` auch der Index `j` übergeben. Nachdem die Sensorknoten des RN-Baumes in Zeile 6 ermittelt wurden, wird die innere Schleife, beginnend in Zeile 7, durchlaufen. In Zeile 8 wird überprüft, ob die Sensorknoten indizierbar sind. Es kann vorkommen, dass ein Baum mehr Sensorknoten besitzt als ein anderer, sofern eine gleichmäßige Verteilung der Regionen nicht möglich war. Ist eine Indizierung möglich, so werden die Frequenz und der Zeitschlitz entsprechend des in Unterabschnitt 5.2.4 vorgestellten Ansatzes gesetzt und die Anzahl erfolgreich betrachteter Sensorknoten in Zeile 12 inkrementiert. Da in der nächsten Iteration der inneren Schleife der nächste Sensorknoten ausgewählt werden soll, wird der lokale Index in Zeile 14 ebenso inkrementiert. Nachdem die äußere Schleife durchlaufen wurde, muss in Zeile 17 geprüft werden, ob die Variable `treeCounter` bereits alle drei Kombinationen der RN-Baum-Paare betrachtet hat (vgl. Tabelle 5.6). Sofern alle drei Kombinationen betrachtet wurden, wird der Variablen `treeCounter` in Zeile 18 für die neue Iteration der Wert 0 zugewiesen. Des Weiteren wurden in jedem RN-Baum zwei Sensorknoten betrachtet, weshalb der globale Index in Zeile 19 um den Wert 2 erhöht wird. Das angepasste ns3-Projekt inklusive dem angepassten LoRaWAN-Modul ist auf der beigefügten CD hinterlegt.

5.5 Evaluation der LoRaWAN-Erweiterung in ns3

In Kapitel 4 wurde der Ist-Zustand der Skalierbarkeit von LoRaWAN bezüglich des gewünschten Anwendungsfalls der Waldbranderkennung evaluiert. Es wurde festgestellt, dass bei der Verwendung von lediglich einem Gateway und dem Einsatz von ADR eine Abdeckung von 2,3 km^2 erreicht werden kann. Es hat sich gezeigt, dass nicht nur die Abdeckung gering ist, sondern auch, dass es zu Interferenzen beim Senden kommt. Aufgrund der Interferenzen wurde eine PDR von 85,97 % festgestellt. Soll lediglich ein fester SF konfiguriert werden, um sowohl den organisatorischen Aufwand beim Wechseln der Batterie oder Laden der Akkus gering zu halten, so muss entweder die abzudeckende Fläche verkleinert oder eine größere Anzahl Gateways eingesetzt werden. Wie in Abschnitt 1.1 beschrieben, ist es im gewünschten Anwendungsfall aufgrund der schlechten Internetanbindung nicht möglich, eine größere Anzahl Gateways zu installieren. Ein hoher SF

hat jedoch den Nachteil des hohen Energieverbrauchs, was in Unterabschnitt 5.1.1 bereits festgestellt wurde. Des Weiteren ist die PDR beim Einsatz eines hohen SFs und einer großen Anzahl Sensorknoten nicht akzeptabel (vgl. Tabelle 4.1). Aus den zuvor genannten Gründen wurde in diesem Kapitel die LoRaWAN-Erweiterung TiReX vorgestellt.

Mit der Implementierung von TiReX in das ns3-Modul von Van den Abeele et al. [8] war es möglich, einen Vergleich zu den in Kapitel 4 vorgestellten Ergebnissen zur Skalierbarkeit von LoRaWAN anzustellen. Durch TiReX wird zwar die Anzahl der benötigten Gateways reduziert und der SF der Sensorknoten möglichst niedrig gehalten, jedoch sinkt auch die Anzahl einsetzbarer Sensorknoten bei einer größeren Fläche. Die geringere Anzahl Sensorknoten resultiert vor allem aus Restriktionen des Duty Cycles der RNs (vgl. Tabelle 5.5). Deshalb soll im Rahmen der Simulation ermittelt werden, inwieweit TiReX skaliert und welche Parameter sinnvoll erscheinen, bevor eine prototypische Umsetzung erfolgt.

Zunächst ist von Interesse, ob das beschriebe Routing in Bezug auf das Zeitverhalten und die Restriktionen des Duty Cycles eine bessere PDR liefert. Da in den Gleichungen, die für TiReX aufgestellt wurden, grundsätzlich Maximalwerte angenommen wurden, sollten keine Interferenzen auftreten, sodass eine PDR von 100 % zu erwarten wäre.

SF (RN)	Anzahl Sensorknoten	Längster Pfad	Toleranzzeit	Wartezeit
8	885	5	1300 ms	9200 ms
9	492	4	1500 ms	16500 ms
10	234	3	2000 ms	32700 ms
11	108	3	3200 ms	57200 ms

Tabelle 5.7: Parameter der RNs der TiReX-Simulation

Wie zuvor erwähnt, wird im Kontext der Simulation angenommen, dass ein Gateway vier parallele Empfangspfade und ein RN zwei parallele Sende- und Empfangspfade besitzt. Des Weiteren soll die Anzahl einsetzbarer Sensorknoten pro Region maximiert werden, wobei die Sensorknoten gleichzeitig einen möglichst geringen Energieverbrauch aufweisen sollen. Da SF7 die maximale Reichweite für die betrachteten SFs in Tabelle 5.7 pro Region abdecken kann, werden im Rahmen der Simulation von TiReX die Sensorknoten mit SF7 konfiguriert. Alle weiteren Parameter der Simulation sind in Tabelle 5.7 aufgelistet. Die Toleranzzeit wurde auf Basis des Beispiels aus Unterabschnitt 5.2.4 berechnet. SF7 und SF12 wurden, wie bereits in Unterabschnitt 5.1.3 beschrieben, aufgrund der beeinträchtigten Skalierbarkeit nicht weiter betrachtet. Im Vergleich zu Tabelle 5.5 fällt auf, dass für SF10 aufgrund der hohen Toleranzzeit nicht alle Sensorknoten eingesetzt werden konnten. Dieses Problem wurde bereits in Abschnitt 5.2 erwähnt und wurde im

Rahmen dieser Simulation gelöst, indem die Anzahl der Sensorknoten soweit reduziert wurde, bis eine Zuteilung der Zeitschlitze wieder möglich war. Für SF10 konnten deshalb nur 91 % der Sensorknoten eingesetzt werden. Alle weiteren Parameter wurden aus den TiReX-Beispielen in Unterabschnitt 5.1.3 und Abschnitt 5.2 übernommen und sind in Tabelle 5.8 aufgelistet.

Paketgröße	Sendeintervall	Anz. Subbänder	Duty Cycle	Anz. Freq./Subband
16 Byte	10 min	2	1 %	2

Tabelle 5.8: Auflistung der vom Spreading Factor (SF) der RNs unabhängigen Parameter

In Listing 5.7 sind alle Ergebnisse der Simulationsdurchläufe aufgelistet. Es wird deutlich, dass für jeden Simulationsdurchlauf alle von den Sensorknoten versendeten Pakete auch am Gateway empfangen werden konnten. Die PDR bestätigt, dass die Ansätze in TiReX umgesetzt werden können und dass die Erwartungen auf Basis der mathematischen Betrachtung erfüllt wurden.

Listing 5.7 Auflistung der Ergebnisse der TiReX-Simulationen mit SF8–SF11 für die RNs und SF7 für die Sensorknoten

```
SF8 PDR STATS:
    nPackets_macRx=4425
    nPackets_macTx=4425
    PDR=1

SF9 PDR STATS:
    nPackets_macRx=2460
    nPackets_macTx=2460
    PDR=1

SF10 PDR STATS:
    nPackets_macRx=1160
    nPackets_macTx=1160
    PDR=1

SF11 PDR STATS:
    nPackets_macRx=540
    nPackets_macTx=540
    PDR=1
```

5.6 Zusammenfassung und Ausblick

In Kapitel 4 wurde gezeigt, dass LoRaWAN für den Anwendungsfall einer Waldbranderkennung nicht im erforderlichen Umfang einsetzbar ist. Insbesondere hat die Kombination aus maximaler Reichweite der Sensorknoten und dem erhöhten Energieverbrauch bei einem höheren SF gezeigt, dass das Netz nicht skaliert. Trotz dessen überwiegen die Vorteile von LoRaWAN. Deshalb wurde die LoRaWAN-Erweiterung TiReX entwickelt, die die Reichweite des Netzes mit Hilfe eines Multi-Hop-Ansatzes sowie eines Zeitschlitz-Verfahrens erhöht. Dadurch können die Sensorknoten mit einem geringen SF konfiguriert werden, sodass der geringste Energieverbrauch resultiert, die versendeten Pakete aber trotzdem am Gateway empfangen werden können. Der Multi-Hop-Ansatz in TiReX ähnelt der Zonen-Aufteilung in der Arbeit von Rizanov et al. [15], die in Abschnitt 1.2 vorgestellt wurde. Jedoch wird in TiReX statt einer Gateway-Infrastruktur ein Multi-Hop-Verfahren mit Hilfe von RNs aufgebaut, wobei die Restriktionen durch den maximalen Duty Cycle berücksichtigt werden.

Wie zuvor erwähnt, basiert TiReX zusätzlich auf einem Zeitschlitz-Verfahren. Die Schwierigkeit beim Zeitschlitz-Ansatz liegt vor allem im Festlegen der Toleranzzeit, die benötigt wird, um Ungenauigkeiten der Uhren der Sensorknoten ausgleichen zu können. Demnach müssen Hardwarekomponenten bei der Umsetzung von TiReX evaluiert, LoRaWAN-spezifische Parameter angepasst und die Toleranzzeit auf Basis dessen festgelegt werden. Für diesen Zweck eignet sich die Verwendung des LoRaWAN-Moduls von Van den Abeele et al. [8], in das TiReX implementiert wurde. Dadurch kann bereits vorab festgestellt werden, wie viele Sensorknoten im Netz eingesetzt werden können und inwieweit Parameteranpassungen das Netz beeinflussen. Die Simulation kann durch verschiedene Aspekte erweitert werden. Interessant wäre, ein besseres Fehlermodell zu Grunde zu legen, wie es in [8] durchgeführt wurde. Darin wäre besonders der Einfluss von verschiedenen Witterungsverhältnissen von Interesse. Des Weiteren wäre auch eine Implementierung von topografischen Eigenschaften von Wäldern durch Anbinden von Geländeplänen wünschenswert, sodass die Anordnung der RNs auf den gegebenen Einsatzort angepasst werden kann. So könnte beispielsweise die Befestigungshöhe der Sensorknoten oder der RNs schon vorab festgestellt werden.

TiReX bringt jedoch auch Nachteile mit sich. Der maximale Duty Cycle der RNs schränkt die Anzahl einsetzbarer Sensorknoten pro Region ein. Ein weiteres Problem liegt vor, wenn innerhalb eines Routing-Baumes ein RN ausfällt. In diesem Fall könnten alle Sensorwerte der RN-Kinder nicht mehr am Gateway empfangen werden. Im schlechtesten Fall könnte ein ganzer Baum ausfallen, sofern ein RN aus dem ersten Ring ausfällt. Mög-

licherweise können das Gateway und die RNs in solchen Fällen untereinander kommunizieren, sodass das temporäre Ändern des Routing-Baumes möglich wäre.

Ein Parameter, der bisher nicht näher betrachtet wurde, ist die Sendeleistung der Sensorknoten. Bisher wurde davon ausgegangen, dass alle Sensorknoten mit 14 dBm senden. 14 dBm entspricht der höchstmöglichen Sendeleistung auf den betrachteten Frequenzen in Deutschland, benötigt beim Senden aber auch mehr Strom. Die Sendeleistung von 14 dBm musste in diesem Kapitel verwendet werden, da die Messungen in Kapitel 3 ebenfalls darauf beruhen. Da das Tewari-Modell die Sendeleistung im Pfadverlustmodell nicht mit aufnimmt, müssten die Messungen für verschiedene Sendeleistungen erneut durchgeführt werden. Solche umfangreichen Messungen übersteigen jedoch den zeitlichen Rahmen dieser Arbeit.

Ebenso wurde der Aktivierungsprozess eines Sensorknotens bei einem Gateway bisher nicht weiter betrachtet. In Unterabschnitt 2.2.3 wurden die Aktivierungsprozesse ABP und OTAA vorgestellt. Die aktuelle Version von TiReX ist lediglich mit ABP möglich, da für ABP keine Pakete vom Gateway empfangen werden müssen. Jedoch gilt OTAA aufgrund der bei jedem Aktivierungsprozess neu generierten Session-Keys als sicherer. OTAA kann derzeit nur dann verwendet werden, wenn die Anmeldung in der Nähe eines Gateways erfolgt. Das Einbinden von OTAA in TiReX ist ein Ansatz für zukünftige Arbeiten.

6 Prototypenentwicklung

In diesem Kapitel sollen die verwendeten Hardware-Komponenten insbesondere der Sensorknoten näher beschrieben und der Grund für die Auswahl erläutert werden. Neben den Hardware-Komponenten wird auch auf Aspekte der Software der Sensorknoten eingegangen. Ob die Hardware-Komponenten in Bezug auf den Energieverbrauch skalieren und eine Umsetzung eines solchen Netzes auch aus finanzieller Sicht realistisch ist, wird im Rahmen der Evaluation in Kapitel 7 näher betrachtet. Zum Abschluss dieses Kapitels wird der Aufbau der prototypischen RNs vorgestellt.

6.1 Sensorknoten

In Abschnitt 1.1 wurde beschrieben, dass der Fokus im Rahmen dieser Arbeit insbesondere auf den Sensorknoten liegen soll. Deshalb werden in diesem Abschnitt die verwendeten Hardware-Komponenten und der Ablauf der entwickelten Software genauer erläutert.

6.1.1 Mikrocontroller

Der Mikrocontroller dient in den Sensorknoten als zentrale Steuerungseinheit. Da als Betriebssystem RIOT verwendet werden soll (siehe Unterabschnitt 6.1.7), muss der Mikrocontroller aufgrund von zeitlichen Einschränkungen dieser Arbeit bereits durch RIOT unterstützt werden. Des Weiteren existieren einige Anforderungen, die der Mikrocontroller erfüllen muss:

- **Größe des Flash-Speichers:** In RIOT wird LoRaWAN auf Basis einer Implementierung von Semtech unterstützt[1]. Da diese Implementierung umfangreich ist und zusätzlich weitere Treiber für die Sensorknoten benötigt werden, sollte der Flash-Speicher des Mikrocontrollers eine Größe von mindestens 64 KiB aufweisen.
- **Real Time Clock (RTC):** Der Mikrocontroller sollte bereits eine RTC beinhalten, um so auf zusätzliche Hardware-Komponenten verzichten zu können.
- **Strombedarf während der Schlaf-Phase:** Damit die Sensorknoten über mehrere Jahre eingesetzt werden können, muss der Mikrocontroller während der Schlaf-Phase bei aktivierter RTC einen sehr geringen Strombedarf aufweisen. Angestrebt werden Mikrocontroller mit einem Strombedarf in der Schlaf-Phase von weniger als 1 µA bei aktivierter RTC.

[1] Siehe: https://github.com/RIOT-OS/RIOT/tree/master/pkg/semtech-loramac (Abgerufen am 2021-04-17)

- **Kosten:** Damit die Sensorknoten auch in finanzieller Hinsicht skalieren, müssen die Kosten pro Mikrocontroller möglichst gering sein. Als Grenze wird ein Preis von 2 € pro Mikrocontroller verwendet. Da Mengenrabatte den Preis pro Mikrocontroller beeinflussen, wird der Preis bei einer Abnahme von 1000 Stück betrachtet.

In RIOT werden eine Vielzahl Mikrocontroller unterstützt[2]. Um den Mikrocontroller zu finden, der die Anforderungen am besten erfüllt, wurde eine Tabelle mit den in RIOT unterstützten Mikrocontrollern aufgestellt und die zuvor genannten Anforderungen anhand der jeweiligen Datenblätter überprüft. Um die Kosten im Rahmen dieser Arbeit möglichst gering zu halten, wurden alle Bauteile in einer Bestellung bei einem Händler aufgegeben. Als Händler wurde Digi-Key[3] ausgewählt. Deshalb musste zusätzlich überprüft werden, ob der jeweilige Mikrocontroller bei Digi-Key zum Zeitpunkt der Bestellung verfügbar war. Sofern der Mikrocontroller nicht verfügbar war, wurde dieser nicht weiter betrachtet. Werden in RIOT ganze Mikrocontroller-Familien unterstützt (beispielweise STM32F, STM32L etc.), so wurde die Mikrocontroller-Reihe in die Tabelle aufgenommen, die die Anforderungen am besten erfüllt (beispielsweise die STM32L-Familie aufgrund des geringen Strombedarfs während der Schlaf-Phase). Die Tabelle der Mikrocontroller ist in Anhang D beigefügt.

Die beiden einzigen Mikrocontroller-Reihen, die die Anforderungen in Gänze erfüllen, sind die STM32L-Reihe von STMicroelectronics und die SAML1x-Reihe von Microchip Technology. Beide Mikrocontroller-Reihen weisen ähnliche Werte auf, wobei die SAML1x-Reihe die Anforderungen in den Kategorien Ruhestrom und Kosten geringfügig besser erfüllt. Da die Unterschiede jedoch gering sind und der Erfahrungswert mit der STM32L-Reihe zum Zeitpunkt der Arbeit größer war, wurde die STM32L-Reihe ausgewählt. Um die Kosten gering zu halten, wurde aus der STM32L-Reihe der Mikrocontroller gewählt, der den kleinsten Flash-Speicher größer gleich den geforderten 64 KiB aufweist. Deshalb wurde der STM32L062K8T6 als Mikrocontroller für die Sensorknoten ausgewählt, da dieser einen Flash-Speicher mit einer Größe von 64 KiB besitzt.

[2] Siehe: https://doc.riot-os.org/group__cpu.html (Abgerufen am 2021-04-17)
[3] Website von DigiKey: https://www.digikey.de (Abgerufen am 2021-04-17)

6.1.2 Temperatur- und Luftfeuchtigkeitssensor

Wie in Abschnitt 2.4 beschrieben, kann auf Basis der Temperatur und der relativen Feuchtigkeit der Umgebungsluft die Wahrscheinlichkeit für eine schnelle Ausbreitung eines Waldbrandes ermittelt werden. Deshalb sollen die Sensorknoten im Rahmen dieser Arbeit mit einem Sensor ausgestattet werden, der die Temperatur und die relative Feuchtigkeit der Umgebungsluft ermitteln kann. Es existieren viele Sensoren, die sowohl einen Temperatur- als auch einen Luftfeuchtigkeitssensor beinhalten. Eine Bedingung für den Einsatz eines solchen Sensors ist, dass innerhalb kürzester Zeit die benötigten Werte ermittelt und an den Mikrocontroller übertragen werden. Wie in Unterabschnitt 5.1.1 exemplarisch gezeigt, muss die Wach-Phase des Sensorknotens so gering wie möglich sein. Beispielsweise werden für das Auslesen der Werte aus einem Temperatur- und Luftfeuchtigkeitssensor DHT22 bereits mehrere hundert Millisekunden benötigt, was unter anderem durch das eingesetzte Protokoll hervorgerufen wird (siehe Datenblatt[4] des DHT22). Aus diesem Grund sind Temperatur- und Luftfeuchtigkeitssensoren dieses Typs für die Sensorknoten nicht akzeptabel. Eine Alternative zum DHT22 ist beispielsweise der Temperatur- und Luftfeuchtigkeitssensor BME280[5] von Bosch. Ein BME280-Modul ist in Abbildung 6.1 zu sehen. Der BME280-IC ist in Abbildung 6.1 mit Markierung 1 gekennzeichnet.

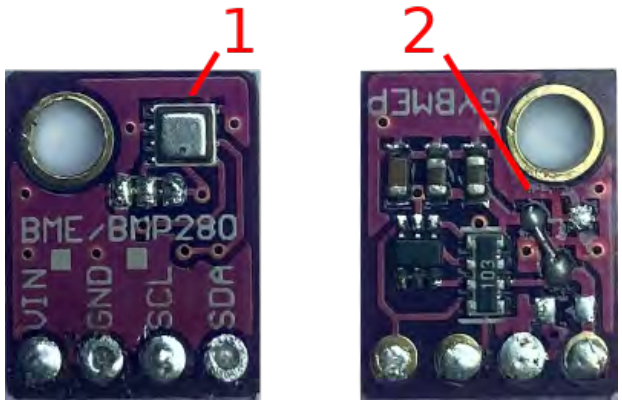

Abbildung 6.1: Vorder- und Rückseite eines BME280-Moduls

[4]Datenblatt des DHT22: https://cdn-shop.adafruit.com/datasheets/DHT22.pdf (Abgerufen am 2021-04-20)

[5]Datenblatt des BME280: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf (Abgerufen am 2021-04-20)

In Abbildung 6.2 ist eine Messung mit Hilfe eines Oszilloskops zu sehen, in der die Zeit ermittelt wurde, die für das Auslesen und Übertragen der Sensorwerte (Temperatur und relative Luftfeuchtigkeit) aus einem BME280 benötigt wird. Auf der Abszisse ist die Zeit in Millisekunden und auf der Ordinate die Spannung zu sehen. Um die benötigte Zeit für das Auslesen und Übertragen der Sensorwerte ermitteln zu können, wurde vor der Initialisierung des BME280 ein GPIO-Pin aktiviert und nach dem Auslesen wieder deaktiviert. Dementsprechend wurde die Prüfspitze des Oszilloskops mit dem zuvor erwähnten GPIO-Pin verbunden. Im oberen linken Bereich in Abbildung 6.2 ist zu erkennen, dass der GPIO-Pin 51,6 ms aktiviert war und der BME280 somit in etwa diese Zeit zum Ermitteln und Übertragen der Sensorwerte benötigt. Da innerhalb der 51,6 ms zwei Parameter ermittelt werden, scheint die benötigte Zeit für die Sensorknoten ausreichend gering zu sein.

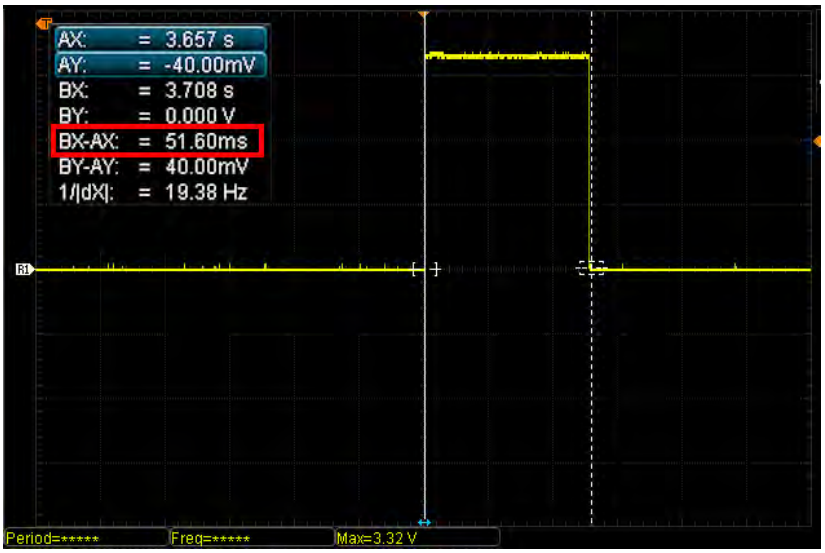

Abbildung 6.2: Benötigte Zeit für das Auslesen von der Temperatur und der relativen Luftfeuchtigkeit mit Hilfe eines BME280

Ein weiterer Vorteil des BME280 liegt darin, dass dieser automatisch nach dem Ermitteln und Übertragen der angefragten Sensorwerte in einen Schlaf-Zustand wechselt und während des Schlaf-Zustands einen Strombedarf von lediglich 100 nA aufweist. Daraus folgt, dass der BME280 nicht mit zusätzlicher Hardware von der Spannungsversorgung getrennt werden muss. Um den niedrigen Ruhestrom zu erreichen, musste das BME280-Modul aus Abbildung 6.1 angepasst werden. In Abbildung 6.1 ist an Markierung 2 eine Lötbrücke zu sehen. An den Lötstellen war zuvor ein Spannungsregler verlötet, sodass das Modul auch mit einer Spannung von über 3,3 V versorgt werden konnte. Der Ruhestrom

des Spannungsreglers liegt jedoch bei etwa 6 µA. Da auf dem PCB der Sensorknoten eine Spannung von 3,3 V genutzt wird (siehe Unterabschnitt 6.1.6), konnte der Spannungsregler des BME280-Moduls durch die beschriebene Lötbrücke ersetzt werden. Dadurch werden statt der 6 µA in etwa 100 nA während des Schlaf-Zustands des BME280-Moduls benötigt.

Die vorherigen Erklärungen zeigen, dass der BME280 ein geeigneter Sensor für das Ermitteln von der Temperatur und der relativen Luftfeuchtigkeit ist und deshalb für die Sensorknoten verwendet werden kann.

6.1.3 Sensoren zur Raucherkennung

In Abschnitt 2.4 wurde beschrieben, dass zur Erkennung eines Waldbrandes ein Sensor eingesetzt werden muss, mit Hilfe dessen auf Rauch in der Umgebungsluft geschlossen werden kann. Es existieren unterschiedliche Möglichkeiten, um Rauch mit Hilfe von Sensoren erkennen zu können:

- **Gassensoren:** Sensoren, die Rauchgase oder Gase ermitteln, die ein Indikator für Rauch in der Luft sind, wurden in der Vergangenheit in verschiedenen Arbeiten für Sensornetze zur Waldbranderkennung eingesetzt (siehe beispielsweise [12, 13, 15]). Typischerweise werden zwei Arten solcher Gassensoren unterschieden. Zum einen existieren infrarotbasierte Gassensoren, zum anderen elektrochemische Gassensoren. Die infrarotbasierten Gassensoren werden nicht weiter betrachtet, da mit einem ähnlichen Aufbau Rauchpartikel direkt ermittelt werden können, ohne den Rauch aus einem Gas herleiten zu müssen. Dieser Ansatz wird im weiteren Verlauf genauer betrachtet (siehe „Photoelektrische Rauchsensoren“).

Abbildung 6.3: Strombedarf eines MQ-135 Gassensors (in Milliampere)

Elektrochemische Gassensoren, die in den zuvor erwähnten Arbeiten eingesetzt wurden, haben einige Nachteile. Aufgrund der Abhängigkeit zur Umgebungstemperatur, besitzen elektrochemische Gassensoren meist zusätzliche Heizmodule. Das Erhitzen resultiert in einem hohen Energieverbrauch, wie in Abbildung 6.3 für einen exemplarischen MQ-135 elektrochemischen Gassensor gezeigt. In Abbildung 6.3 ist zu sehen, dass etwa 103 mA zum Betrieb des Gassensors benötigt werden. Der Strombedarf ist über den gesamten Testlauf mit einer Dauer von mehreren Minuten annähernd konstant geblieben. Des Weiteren werden nach dem Aktivieren des Sensors häufig mehrere Minuten benötigt, bis verlässliche Sensorwerte ermittelt werden können. Aufgrund der Kombination aus dem hohem Strombedarf und der Zeit, die für das Ermitteln verlässlicher Sensorwerte benötigt wird, sind solche Gassensoren für die Sensorknoten nicht geeignet.

- **Ionisationsrauchsensoren:** In Ionisationsrauchsensoren wird die Luft in einer Kammer mit radioaktiven Partikeln versehen. Durch die Verwendung der radioaktiven Partikel fließt ein geringer Strom, sofern keine Rauchpartikel vorhanden sind. Sind Rauchpartikel in der Kammer vorhanden, so nimmt der Strom ab. Dieses Prinzip sorgt für eine geringe Latenz, da bereits eine geringe Menge Rauchpartikel ausreicht, um einen Alarm auszulösen. Aufgrund der radioaktiven Partikel sorgen solche Sensoren nach einem Brand jedoch für eine Einstufung des Brandguts als Sondermüll. Des Weiteren ist der Einsatz solcher Sensoren in vielen Ländern mittlerweile untersagt. [56] Aufgrund der beiden zuvor genannten Aspekte eignen sich solche Sensoren nicht für die Sensorknoten.

- **Photoelektrische Rauchsensoren:** Photoelektrische Rauchsensoren nutzen häufig das Prinzip der Lichtstreuung[6]. Meist wird in einer vom Umgebungslicht abgeschirmten Kammer eine Lichtquelle (z.B. eine Infrarot-LED) von einer Fotodiode getrennt. Ist Rauch in der Kammer vorhanden, so wird das Licht der Lichtquelle gestreut und trifft auch auf die Fotodiode. Ein solcher photoelektrischer Sensor ist in Abbildung 6.4 gezeigt, wobei keine Kammer um den Sensor platziert ist.

[6] Es existieren auch photoelektrische Rauchsensoren, die das Prinzip der Lichtabschattung nutzen.

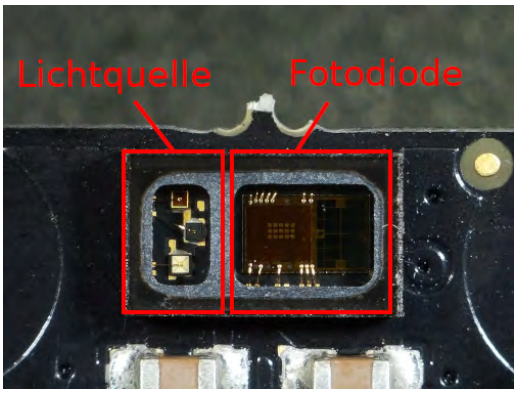

Abbildung 6.4: Nahaufnahme eines MAX30105

Photoelektrische Rauchsensoren sind aufgrund der benötigten elektronischen Bauelemente günstig. Aufgrund der geringen Kosten und dem im Vergleich zu elektrochemischen Gassensoren geringen Energieverbrauch, ist diese Art von Rauchsensor für die Sensorknoten am besten geeignet. Es sei jedoch anzumerken, dass sowohl eine hohe Luftfeuchtigkeit als auch Staub in der Umgebungsluft die Messwerte verfälschen können. Da die Luftfeuchtigkeit gemessen wird, ist diesbezüglich eine softwareseitige Kompensation denkbar. Staub ist ein größeres Problem für die photoelektrischen Rauchsensoren [57]. Es existieren Arbeiten, die bestimmte Filter vor den Kammern einsetzen, um Staubpartikel aus der Luft zu entfernen (siehe beispielsweise [58]). Ein ähnliches Filter-Prinzip kann auch für die Sensorknoten verwendet werden.

6.1.4 Hardware-Komponente zur Zeitsynchronisation

TiReX nutzt ein Zeitschlitz-Verfahren, um strukturiertes Senden im Netz zu ermöglichen und so Kollisionen zu verhindern. Wie in Abschnitt 5.2 beschrieben, wird eine Synchronisation der Zeit vorausgesetzt, da die Ganggenauigkeit der Schwingquarze starke Abweichungen bei besonders niedrigen oder besonderes hohen Temperaturen aufweisen kann.

In Unterabschnitt 5.2.3 wurde die Möglichkeit vorgestellt, eine Synchronisation der Uhr mit Hilfe des Zeitzeichensenders DCF77 vorzunehmen. Die Vorteile sind der geringe Energieverbrauch während des einminütigen Empfangs der Nachricht vom DCF77 und die Möglichkeit, eine Uhr im Prinzip jede Minute synchronisieren zu können. Der Nachteil liegt in den zusätzlichen Kosten pro Sensorknoten.

Abbildung 6.5: DCF77-Modul mit 77,5 kHz Ferritstabantenne

Anzumerken ist, dass der auf dem DCF77-Modul in Abbildung 6.5 verbaute 77,5 kHz Schwingquarz keine Kompensation der Temperatur durchführt. Auf einem PCB im Produktiveinsatz müssten entsprechende Maßnahmen zum Erhöhen der Ganggenauigkeit vorgenommen werden. Da es sich im Rahmen der Umsetzung der Sensorknoten zunächst um Prototypen handelt, sind die verbauten Hardware-Komponenten auf dem Modul akzeptabel.

6.1.5 Spannungsversorgung

Typischerweise werden in drahtlosen Sensornetzen Batterien eingesetzt [59], was durch die lange Laufzeit von teilweise 10–15 Jahren zu erklären ist. Soll der Akku der Sensorknoten nach 10–15 Jahren aufgeladen werden, so ist die Wahrscheinlichkeit hoch, dass die angestrebte Laufzeit aufgrund eines Ausfalls nicht erneut erreicht wird. Deshalb werden bei einer langen Laufzeit Batterien angestrebt. Es existieren Lithium-Thionylchlorid-Batterien, die für eine lange Laufzeit (20–30 Jahre) beispielsweise in Sensornetzen ausgelegt sind. Um eine lange Laufzeit zu erreichen, muss die Selbstentladung möglichst niedrig sein, schließlich liegt der Strombedarf der Hardware-Komponenten die meiste Zeit typischerweise im Bereich von wenigen Mikroampere. Die Anforderung der niedrigen Selbstentladung erfüllen Lithium-Thionylchlorid-Batterien. Des Weiteren sind Lithium-Thionylchlorid-Batterien in einem Temperaturbereich von -40 °C bis 85 °C stabil, was somit auch den Einsatz in Wäldern ermöglicht. [60]

Abbildung 6.6: Lithium-Thionylchlorid-Batterie für die Sensorknoten

In Abbildung 6.6 ist eine Lithium-Thionylchlorid-Batterie zu sehen, die für die Sensorknoten eingesetzt wurde. Die Batterie besitzt eine Nennspannung von 3,6 V, entspricht den Maßen der Mignon-Baugröße und hat eine Kapazität von 2200 mAh[7].

6.1.6 Platine

Um alle zuvor beschriebenen Hardware-Komponenten in einem Prototyp zu vereinen, wurde ein PCB für die Sensorknoten entworfen. Der zugehörige Schaltplan ist in Anhang E beigefügt. Die Vorderseite des PCBs ist in Abbildung 6.7 zu sehen.

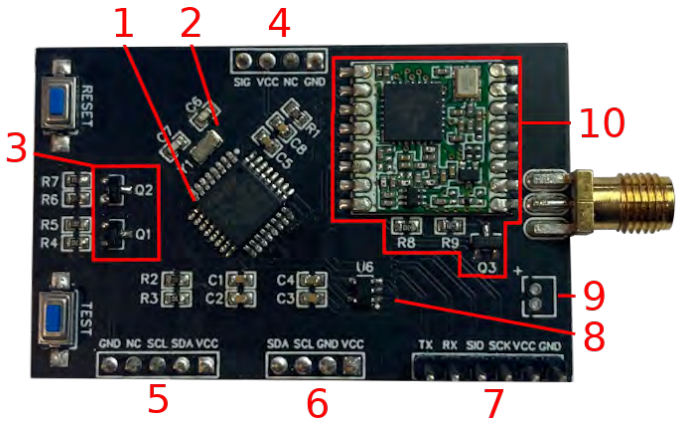

Abbildung 6.7: Vorderseite eines Sensorknoten-PCBs

Der in Unterabschnitt 6.1.1 beschriebene Mikrocontroller (STM32L062K8T6) ist in Abbildung 6.7 an Markierung 1 zu sehen. Für eine möglichst genaue Uhrzeit, welche für den Einsatz von TiReX notwendig ist, wurde ein 32768 kHz Schwingquarz auf der Platine verbaut (siehe Abbildung 6.7, Markierung 2). Im eigentlichen Designentwurf sollte ein TCXO (siehe Unterabschnitt 5.2.3) verbaut werden. Jedoch war zum Zeitpunkt der Bestellung bei Digi-Key kein geeigneter TCXO verfügbar. Deshalb wurde für den Prototyp zunächst ein ABS07 von Abracon LLC mit einer Ganggenauigkeit von 10 ppm verbaut[8]. Dementsprechend muss in der Software des Prototypen häufiger eine Synchronisation der Uhr vorgenommen oder die Toleranzzeit im Zeitschlitzverfahren von TiReX entsprechend erhöht werden. Um sowohl die Spannungsversorgung des Sensors zur Raucherkennung (MAX30105) und des DCF77-Moduls trennen zu können, wurden

[7] Siehe Datenblatt der Batterie: https://tadiranbatteries.de/pdf/lithium-thionyl-chloride-batteries/SL-760.pdf (Abgerufen am 2021-04-21)

[8] Datenblatt des ABS07: https://abracon.com/Resonators/ABS07.pdf (Abgerufen am 2021-04-22)

zwei Transistoren verwendet (siehe Abbildung 6.7, Markierung 3). Da der eingesetzte Temperatur- und Luftfeuchtigkeitssensor (BME280) einen sehr geringen Ruhestrom aufweist, wurde kein dritter Transistor benötigt (siehe Unterabschnitt 6.1.2). In Abbildung 6.7 ist die Schnittstelle für das DCF77-Modul mit Markierung 4, die Schnittstelle für das MAX30105-Modul mit Markierung 5 und die Schnittstelle für das BME280-Modul mit Markierung 6 gekennzeichnet. Die drei Sensoren werden auf der Rückseite platziert, was Vorteile für ein späteres Gehäuse-Design ermöglicht. Um das Programmieren und Debuggen des Prototyps zu ermöglichen, existiert sowohl eine SWD- als auch eine UART-Schnittstelle (siehe Abbildung 6.7, Markierung 7). In Unterabschnitt 6.1.5 wurde beschrieben, dass eine Lithium-Thionylchlorid-Batterie für die Sensorknoten eingesetzt werden soll. Jedoch muss die Spannung der Batterie von 3,6 V auf eine Spannung reduziert werden, mit der alle Hardware-Komponenten betrieben werden können. Deshalb wurde ein Spannungsregler verbaut, der die Spannung auf 3,3 V reduziert. Der verwendete Spannungsregler (AP7354-33W5-7) von Diodes Incorporated ist für einen Strom bis zu 150 mA ausgelegt (siehe Abbildung 6.7, Markierung 8)[9]. Dass ein maximaler Strom von 150 mA für das SX1276-LoRa-Modul ausreicht, wurde bereits in Unterabschnitt 5.1.1 deutlich. Der Ruhestrom des AP7354-33W5-7 beträgt 600 nA. Die Spannungsversorgung wird mit Hilfe einer Schnittstelle für 1,5 mm JST-Steckverbinder ermöglicht (siehe Abbildung 6.7, Markierung 9). Das SX1276-LoRa-Modul ist in Abbildung 6.7 mit Markierung 10 gekennzeichnet. Unterhalb des SX1276-LoRa-Moduls befindet sich ein SI2333DDS-T1-GE3 MOSFET[10]. Der MOSFET wird als Logik-Pegel-Schalter verwendet, sodass das LoRa-Modul von der Spannungsversorgung getrennt werden kann[11].

[9] Datenblatt des AP7354-33W5-7: https://www.diodes.com/assets/Datasheets/AP7354.pdf (Abgerufen am 2021-04-22)

[10] Datenblatt des SI2333DDS-T1-GE3: https://www.vishay.com/docs/63861/si2333dds.pdf (Abgerufen am 2021-04-22)

[11] Je nach verwendetem SX1276-LoRa-Modul kann der Strombedarf im Schlaf-Zustand des SX1276-ICs aufgrund zusätzlicher Hardware (z.B. Pegelwandler) mehrere Mikroampere betragen. Deshalb wurde ein zusätzlicher MOSFET verbaut.

6.1.7 Software der Sensorknoten

Für die Sensorknoten wurde das IoT-Betriebssystem RIOT verwendet, da in RIOT für die meisten der zuvor beschriebenen Hardware-Komponenten bereits Treiber integriert sind. Lediglich für den Rauchsensor MAX30105 musste ein Treiber entwickelt werden. Der Treiber für den MAX30105 ist auf Grundlage einer Arduino-Implementierung des Unternehmens Sparkfun entwickelt worden[12], wobei der in RIOT integrierte Treiber lediglich eine minimale Untermenge der tatsächlichen Funktionen des MAX30105 bereitstellt. Die Untermenge besteht aus dem initialen Konfigurieren des MAX30105 sowie dem Auslesen des Analog-zu-Digital-Wandlers der Photodiode sowohl für Licht aus dem rötlichen Spektrum als auch für Licht aus dem Infrarot-Spektrum.

Listing 6.1 Ausschnitt der Header-Datei des MAX30105-Treibers

```
1 int max30105_init(void);
2 int max30105_shutdown(void);
3 uint32_t max30105_get_ir(void);
4 uint32_t max30105_get_red(void);
```

Für die Verwendung des MAX30105 im betrachteten Anwendungsfall wird lediglich ein Zugriff auf vier Funktionen des Treibers benötigt, um die gewünschten Sensorwerte ermitteln zu können. Die vier Funktionen sind im Ausschnitt der Header-Datei des Treibers in Listing 6.1 zu sehen. In Zeile 1 in Listing 6.1 ist die Funktion zur Initialisierung des MAX30105 zu sehen. Die Initialisierung beinhaltet neben dem Konfigurieren einiger Register des Sensors auch das Aktivieren der Spannungsversorgung zum Sensor-Modul. Die Funktion `max30105_shutdown` trennt das Sensor-Modul wiederum von der Spannungsversorgung, sodass während der Schlaf-Phase des Sensorknotens kein Strom für das Modul benötigt wird. In den Zeilen 3–4 in Listing 6.1 sind die Funktionen zu sehen, mit denen der ermittelte Anteil aus dem jeweiligen Licht-Spektrum abgefragt werden kann. Der RIOT-Treiber für den MAX30105 ist auf der CD hinterlegt.

[12] Arduino-Treiber für den MAX30105 von Sparkfun: https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library (Abgerufen am 2021-04-22)

Abbildung 6.8: Testlauf des MAX30105 auf dem Sensorknoten-PCB

In Abbildung 6.8 ist ein Testlauf des MAX30105 mit dem minimalen RIOT-Treiber zu sehen. Im Testlauf wurde ein qualmendes Stück Holz in die Nähe des Sensors gehalten und die Werte für sowohl die rote LED als auch die Infrarot-LED mit Hilfe von Log-Ausgaben über die serielle Schnittstelle des Mikrocontrollers überprüft. In Abbildung 6.8 ist innerhalb des gekennzeichneten Bereichs an Markierung 1 zu sehen, wie sich die Werte bei Rauch in der Umgebungsluft erhöhen. Alle Logs oberhalb des gekennzeichneten Bereichs wurden ausgegeben, bevor das qualmende Stück Holz unter dem Sensor platziert wurde. In den Log-Ausgaben innerhalb des gekennzeichneten Bereichs ist eine deutliche Erhöhung der Werte des Analog-zu-Digital-Konverters der Photodiode festzustellen. Da im Testlauf kein Gehäuse um den MAX30105 platziert wurde, der bestmöglich vom Umgebungslicht abschottet, muss der Rauch direkt auf den Sensor treffen, sodass sich die Werte deutlich erhöhen. Mit einem geeigneten Gehäuse kann die Sensitivität weiter erhöht werden.

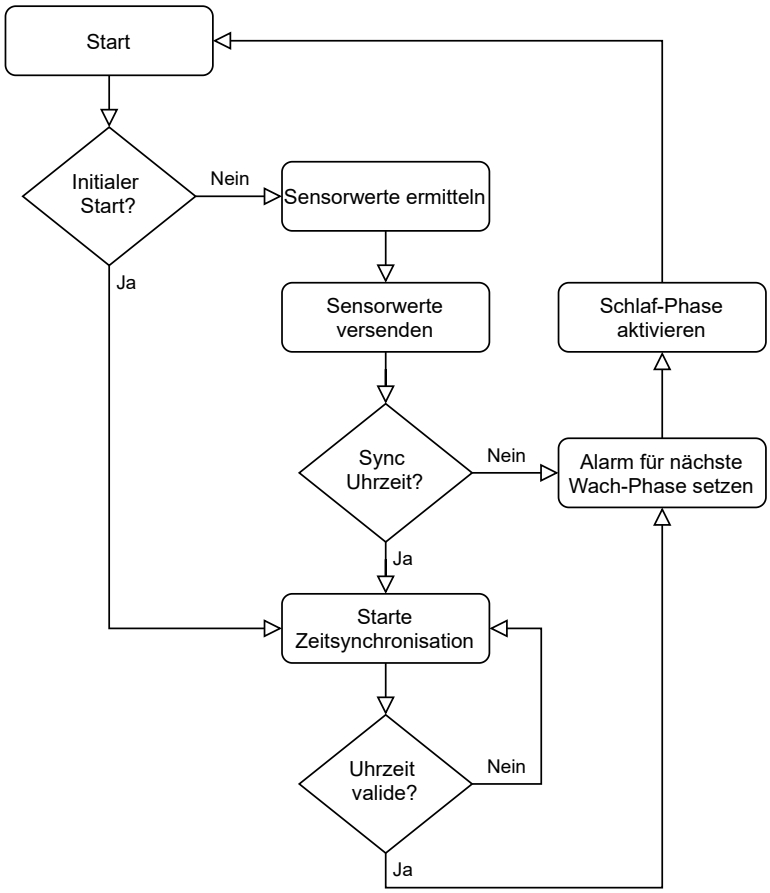

Abbildung 6.9: Flussdiagramm der Software der Sensorknoten

Im letzten Schritt der Softwareentwicklung mussten die einzelnen Funktionen logisch miteinander verknüpft werden. In Abbildung 6.9 ist der Ablauf der Software in Form eines Flussdiagramms zu sehen. Der Ablauf beginnt mit einer Abfrage, ob der Sensorknoten zum ersten mal gestartet wurde. Diese Abfrage ist notwendig, damit die Uhrzeit initial mit Hilfe des DCF77-Moduls synchronisiert und eine Startzeit für das gesamte Netz definiert werden kann. Sofern der initiale Start bereits durchgeführt wurde, werden die Sensorwerte des BME280 sowie des MAX30105 ermittelt. Die ermittelten Sensorwerte werden im Anschluss in einem LoRaWAN-Paket versendet. Daraufhin wird geprüft, ob eine Synchronisation der Uhrzeit notwendig ist. In der Version der Software, die auf der CD hinterlegt ist, wird jeden Tag auf Basis eines Wach-Phasen-Zählers die Uhrzeit synchronisiert. Bei der Synchronisation kann es vorkommen, dass die empfangene Kombination aus Datum und Uhrzeit fehlerhaft vom DCF77-Modul interpretiert wird. Um Fehler im Zeitschlitz-Verfahren in TiReX bestmöglich verhindern zu können, wird deshalb

mit Hilfe von RIOT-internen Funktionen überprüft, ob die Kombination aus Datum und Uhrzeit valide ist. Zwar können minimale Fehler, wie beispielsweise eine minimale Verschiebung der Uhrzeit, mit den Funktionen nicht erkannt werden, jedoch äußert sich die Fehlinterpretation der DCF77-Signale meist in zufälligen Zahlenkombinationen. Die Wahrscheinlichkeit, dass bei einer zufälligen Anordnung der Zahlen eine valide Kombination aus Datum und Uhrzeit entsteht, ist vermutlich gering. Die Synchronisation der Uhrzeit wird maximal drei mal versucht, wobei nach dem dritten Versuch abgebrochen wird. Die Synchronisation wird dann bei der nächsten Wach-Phase erneut durchgeführt. Die Wach-Phase wird mit dem Setzen des nächsten Alarms und dem Aktivieren der Schlaf-Phase beendet. Sobald die RTC des Mikrocontrollers den Alarm auslöst, wird ein Neustart durchgeführt und der Ablauf aus Abbildung 6.9 wird erneut durchlaufen. Die nach Abbildung 6.9 entwickelte Software für den Betrieb der Sensorknoten ist auf der beigefügten CD hinterlegt.

6.2 Relay Nodes

Wie in Unterabschnitt 5.1.3 beschrieben, sind RNs bezüglich der verwendeten Hardware weniger eingeschränkt. In Abschnitt 5.3 wurden die RNs im Simulator ns3 auf Basis der Gateway-Implementierung entwickelt, da gewisse Parallelen zwischen beiden LoRaWAN-Teilnehmern existieren. In der Praxis ist festzustellen, dass die Gateways verschiedenster LoRaWAN-Projekte häufig einen Einplatinencomputer mit einer Linux-Distribution als Grundlage verwenden[13]. Ein solcher Ansatz wurde ebenso im Rahmen dieser Arbeit für die RNs verfolgt.

[13] Siehe beispielsweise eine GitHub-Übersicht quelloffener LoRaWAN-Gateway-Implementierungen: https://github.com/topics/lorawan-gateway?l=c (Abgerufen am 2021-04-22)

6.2.1 Prototypische Hardware

Für die RNs wurde als Hardware-Grundlage ein Raspberry Pi Zero eingesetzt, da der Raspberry Pi Zero mit einer Linux-Distribution verwendet werden kann, die Kosten im Vergleich zu anderen Einplatinencomputern gering sind[14] und eine ausreichende Anzahl GPIO-Pins zur Verfügung stehen, um mehrere LoRa-Module gleichzeitig verwenden zu können. Für einen prototypischen Aufbau der RNs wurden zwei SX1276-LoRa-Module verwendet, sodass ein RN gleichzeitig auf zwei Frequenzen LoRa-Nachrichten senden und empfangen kann. In Abbildung 6.10 ist der Aufbau des prototypischen RNs zu sehen.

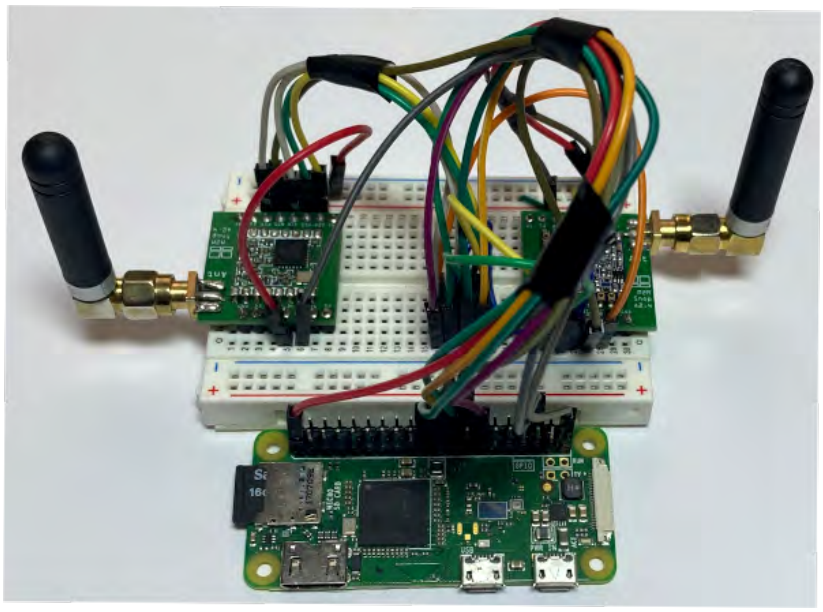

Abbildung 6.10: Prototypischer Aufbau eines RNs mit einem Raspberry Pi Zero und zwei SX1276-LoRa-Modulen

[14] Zum Zeitpunkt der Arbeit kostet ein Raspberry Pi Zero etwa 6 €.

6.2.2 Software der Relay Nodes

Wie bereits zuvor erwähnt, wurde eine quelloffene LoRaWAN-Gateway-Implementierung als Grundlage für das prototypische RN verwendet[15]. Die Gateway-Implementierung wurde insoweit angepasst, dass Funktionen zur Kommunikation mit den Backend-Servern des TTNs entfernt und die in Unterabschnitt 5.1.3 beschriebenen Routing-Abläufe implementiert wurden.

In Listing 6.2 ist ein Ausschnitt der Funktion zu sehen, die beim Empfang einer LoRaWAN-Nachricht aufgerufen wird. In Zeile 1 ist die Überprüfung zu sehen, ob die Adresse des Paketes zum Routing-Baum des RNs zugeordnet und ob die Kombination aus Adresse und FCnt-Feld der LoRaWAN-Nachricht neu ist (vgl. Unterabschnitt 5.1.3). Falls die Adresse zum Routing-Baum zuzuordnen ist und die Kombination aus Adresse und FCnt-Feld bisher unbekannt sind, wird das Paket ohne Wartezeit direkt weitergeleitet. Die Funktion `txLoraModem` wird in Zeile 3 mit sechs Parametern aufgerufen. Der erste Parameter entspricht der Nachricht, die weitergeleitet werden soll. Der zweite Parameter gibt die Länge der Nachricht an. Im dritten Parameter wird der SF übergeben. In Listing 6.2 wird SF9 verwendet, da sich SF9 bereits in Abschnitt 5.5 als guter Kompromiss zwischen maximaler Anzahl RNs und maximal abdeckbarer Fläche herausgestellt hat. Der dritte und vierte Parameter sind die Sendeleistung in dBm und die Frequenz in MHz. Der letzte Parameter entspricht einem Index und definiert, welches der beiden SX1276-LoRa-Module die Nachricht versenden soll. Im Quellcode ist hinterlegt, dass immer das SX1276-LoRa-Modul die Nachricht weiterleiten soll, das die Nachricht empfangen hat.

Listing 6.2 Ausschnitt der Funktion `ReceivePacket` der Software des RN-Prototyps

```
1 if (isChild(address) && isNewPacket(address, fcnt)) {
2     log("RX from child node with DevAddr=%x\n", addr);
3     txLoraModem(message, len, SF9, power, freq, dio_port);
4     return true;
5 }
6 return false;
```

Es wird deutlich, dass nur geringfügige Änderungen der Gateway-Implementierung notwendig waren, um die Funktion eines RNs zu ermöglichen.

[15] GitHub-Repository der Gateway-Implementierung: https://github.com/bokse001/dual_chan_pkt_fwd/tree/dual_chan_pkt_fwd_up_down (Abgerufen am 2021-04-22)

6.3 Zusammenfassung

Im Rahmen dieses Kapitels wurde sowohl die Hardware als auch die Software für die Sensorknoten und die RNs vorgestellt. Bei der Entwicklung der Sensorknoten wurde beachtet, dass der Ruhestrom der Hardware-Komponenten in der Schlaf-Phase besonders niedrig ist. Deshalb werden einige Module während der Schlaf-Phase von der Spannungsversorgung getrennt. Im nächsten Schritt muss gemessen werden, wie hoch der Strombedarf der Sensorknoten sowohl in der Schlaf- als auch in der Wach-Phase ist. Zusätzlich ist von Interesse, wie hoch die Kosten pro Sensorknoten mit den ausgewählten Hardware-Komponenten ist. Neben den Sensorknoten wurde im Rahmen dieses Kapitels auch ein prototypischer RN vorgestellt. Die Basis des RNs ist ein Raspberry Pi Zero mit zwei SX1276-LoRa-Modulen und einer angepassten Gateway-Software. Um die Funktionalität des RNs zeigen zu können, wird ein Versuchsaufbau benötigt. Die Evaluierung der zuvor erwähnten Aspekte wird im folgenden Kapitel durchgeführt.

7 Evaluation der Prototypen

Auf Basis der Entwicklung der prototypischen Sensorknoten und des RNs war es nun möglich, die Komponenten und die Umsetzbarkeit von TiReX zu evaluieren. Wie in Abschnitt 1.1 beschrieben, soll insbesondere die Skalierbarkeit der Sensorknoten evaluiert werden. Dass das Netz bezüglich des Aufbaus, der Anzahl der Sensorknoten und der maximal abdeckbaren Fläche skaliert, wurde bereits im Rahmen der ns3-Simulation in Abschnitt 5.5 ermittelt. Deshalb soll der Fokus im Rahmen dieses Kapitels auf der Evaluation hardwarebezogener Aspekte liegen. Zunächst wird in Abschnitt 7.1 ermittelt, inwieweit die Sensorknoten in Hinblick des Energieverbrauchs skalieren. In Abschnitt 7.2 werden anschließend die Kosten pro Prototyp der Sensorknoten vorgestellt, wodurch die Umsetzbarkeit des Netzes aus finanzieller Sicht betrachtet wird[1]. Zum Abschluss des Kapitels wird in Abschnitt 7.3 ein Testlauf eines minimalen TiReX-Netzes vorgestellt.

7.1 Strombedarf der Sensorknoten

Um den Strombedarf der Sensorknoten mit den in Kapitel 6 vorgestellten Hardware-Komponenten messen zu können, wurde ein µCurrent Gold Multimeter-Adapter eingesetzt[2]. Der gemessene Strom wird auf dem Display des Multimeters in Millivolt angezeigt und kann je nach konfigurierter Einheit am µCurrent Gold (Nanoampere, Mikroampere oder Milliampere) ohne weitere Umrechnung vom Multimeter abgelesen werden. Der Vorteil des µCurrent Gold besteht darin, dass ein sehr niedriger Spannungsabfall beim Messen des Stroms sichergestellt wird. Dadurch kann der Strombedarf auch im Bereich weniger Nanoampere exakt ermittelt werden.

[1] Wie in Abschnitt 6.2 beschrieben, wurden die Hardware-Komponenten der RNs im Rahmen dieser Arbeit nicht weiter optimiert. Aus diesem Grund wird auf eine Kostenaufstellung der Hardware-Komponenten der RNs verzichtet.

[2] Zusammenfassung der technischen Daten des µCurrent Gold: https://www.eevblog.com/projects/ucurrent/ (Abgerufen am 2021-05-06)

7.1.1 Strombedarf in der Schlaf-Phase

Zunächst war von Interesse, den Strombedarf der Sensorknoten während der Schlaf-Phase zu ermitteln. Während der Schlaf-Phase sind das DCF77-, das MAX30105- und das SX1276-Modul von der Spannungsversorgung getrennt und das BME280-Modul im Schlaf-Zustand. Lediglich die RTC des Mikrocontrollers ist insoweit aktiviert, dass der interne Zähler der RTC auf Basis des 32768 kHz Schwingquarzes inkrementiert und auf den Zeitpunkt der nächsten Wach-Phase gewartet wird. Ist der Zeitpunkt der nächsten Wach-Phase erreicht, löst die RTC einen Alarm aus, durch den der Mikrocontroller aktiviert und die restlichen Hardware-Komponenten bei Bedarf zugeschaltet werden. Es ist von Interesse, dass die Sensorknoten während der Schlaf-Phase einen möglichst geringen Energieverbrauch aufweisen, schließlich befinden sich die Sensorknoten die meiste Zeit in dieser Phase.

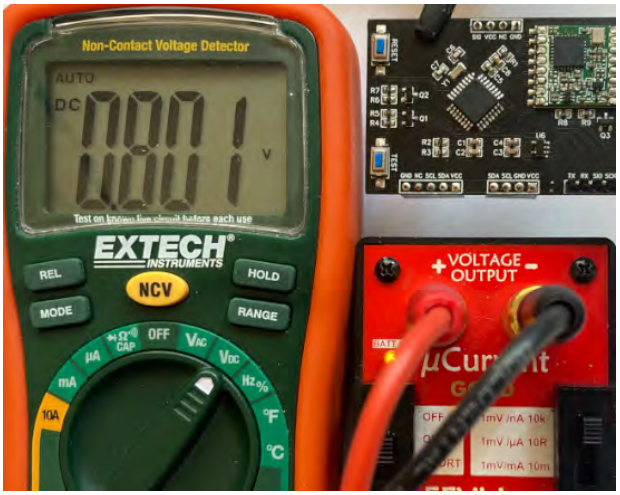

Abbildung 7.1: Strombedarf eines Sensorknotens in der Schlaf-Phase in Millivolt pro Nanoampere

In Abbildung 7.1 ist ein Ausschnitt des Messaufbaus zum Ermitteln des Strombedarfs der Sensorknoten in der Schlaf-Phase zu sehen. Am µCurrent Gold ist die Einheit auf Nanoampere eingestellt. Da auf dem Multimeter-Display in Abbildung 7.1 0,801 V und somit 801 mV ausgegeben werden, besitzt ein Sensorknoten demnach einen Strombedarf von 801 nA während der Schlaf-Phase. Der ermittelte Strombedarf von 801 nA ist nah an den

in Unterabschnitt 5.1.1 vermuteten 1 µA für die Schlaf-Phase, was zeigt, dass die vorherigen Berechnungen auch in der Praxis zutreffen. Da der verwendete 32768 kHz Schwingquarz nicht dem eigentlich vorgesehenen TCXO entspricht, ist zu vermuten, dass der Strombedarf beim Einsatz des TCXOs etwas über einem Mikroampere liegen wird[3].

7.1.2 Strombedarf in der Wach-Phase

In Unterabschnitt 5.1.1 wurde beschrieben, dass die Dauer der Wach-Phase der Sensorknoten möglichst gering sein muss. Der Einfachheit halber wurde in Unterabschnitt 5.1.1 angenommen, dass während der gesamten Wach-Phase der maximale Strom eines SX1276-LoRa-Moduls beim Senden mit 14 dBm benötigt wird. In der Praxis werden vor dem Senden jedoch einige Aufgaben ausgeführt, die weniger Strom benötigen, wie beispielsweise das Ermitteln der Sensorwerte. Demnach sind die in Unterabschnitt 5.1.1 exemplarisch ermittelten 222 ms als Richtwert für die maximale Zeit der Wach-Phase anzusehen und im Rahmen einer Messung mit tatsächlicher Hardware anzupassen. Durch die Ergebnisse der Messung kann im Anschluss in Unterabschnitt 7.1.3 die Laufzeit der Sensorknoten ermittelt werden.

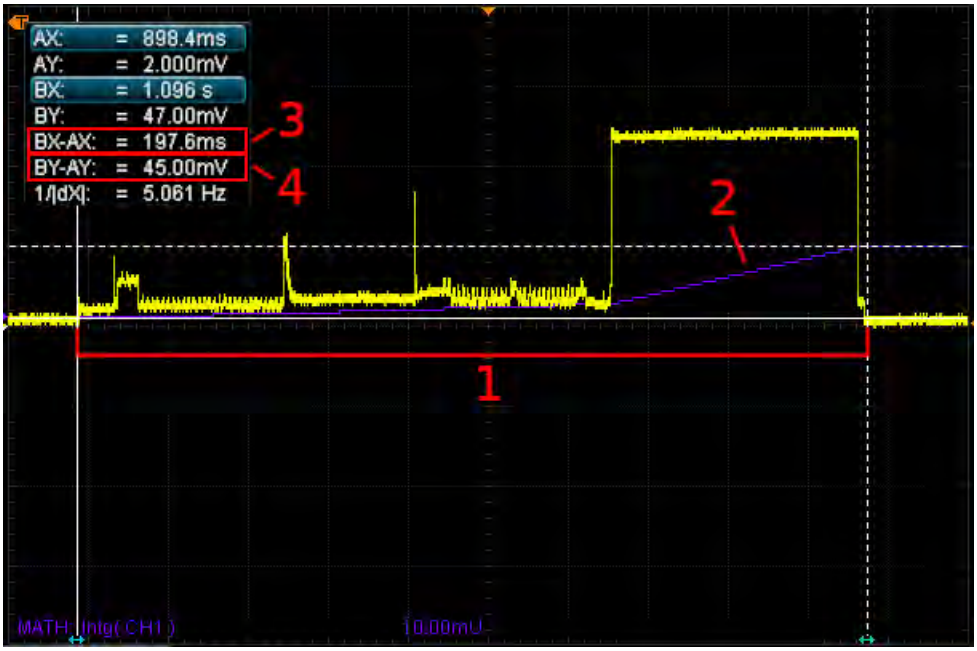

Abbildung 7.2: Strombedarf während der Wach-Phase eines Sensorknotens

[3] Siehe beispielsweise ECS-327TXO von ECS Inc.: https://ecsxtal.com/store/pdf/ECS-327TXO.pdf (Abgerufen am 2021-05-07)

In Abbildung 7.2 ist der Strombedarf während einer Wach-Phase eines Sensorknotens zu sehen. Für die Messung wurde, wie bereits in Unterabschnitt 7.1.1 beschrieben, ein µCurrent Gold verwendet. Die Einheit wurde am µCurrent Gold auf Milliampere festgelegt. Statt die resultierende Spannung mit Hilfe eines Multimeters auszulesen, wurde für das Ermitteln des Strombedarfs während der Wach-Phase ein Oszilloskop verwendet. Der Vorteil des Oszilloskops liegt darin, dass im Rahmen einer Messung sowohl die Dauer der Wach-Phase als auch der Strombedarf ermittelt werden kann. Trotz Konfigurieren eines hochauflösenden Modus am Oszilloskop ist in Abbildung 7.2 jedoch Rauschen festzustellen. Somit weicht der ermittelte Strombedarf vermutlich um einige Milliampere ab.

In Abbildung 7.2 ist der Spannungsverlauf des µCurrent Gold zu sehen. Der in Abbildung 7.2 mit Markierung 1 gekennzeichnete Bereich entspricht der Wach-Phase des Sensorknotens. Beim Betrachten des Spannungsverlaufs fallen die unterschiedlichen Spitzenwerte auf. Die Spitzenwerte sind insbesondere dann festzustellen, wenn Sensorwerte ermittelt oder versendet werden. Insgesamt werden für die Wach-Phase, also dem Aufwachen aus der Schlaf-Phase, dem Initialisieren der Hardware-Komponenten sowie dem Ermitteln und Versenden der Sensorwerte, etwa 197,6 ms benötigt (siehe Abbildung 7.2, Markierung 3). Um im Vergleich zur exemplarischen Berechnung in Unterabschnitt 5.1.1 eine präzisere Aussage über den Strombedarf während der Wach-Phase treffen zu können, wurde eine Kurve des Integrals des Spannungsverlaufs mit Hilfe des Oszilloskops ermittelt (siehe Abbildung 7.2, Markierung 2). Der Höchstwert der Kurve entspricht demnach dem Integral der gesamten Wach-Phase und beträgt 45 mV (siehe Abbildung 7.2, Markierung 4). Da ein µCurrent Gold im Messaufbau verwendet wurde, können die 45 mV direkt als 45 mA abgelesen werden. Dementsprechend werden während der Wach-Phase des Sensorknotens mit einer Dauer von 197,6 ms in etwa 45 mA Strom benötigt. Wie zuvor erwähnt, kann der tatsächliche Wert in der Praxis aufgrund des Rauschens bei den Messungen um einige Milliampere abweichen. In Abbildung 7.2 wurde keine Synchronisation der Uhr durch das DCF77-Modul vorgenommen. In der Praxis wäre der Einfluss jedoch gering, schließlich werden während der Synchronisation nur einige hundert Mikroampere benötigt. Des Weiteren findet die Synchronisation der Uhr in sehr großen Zeitabständen statt.

7.1.3 Laufzeit mit einer Batterie

Mit Hilfe von Gleichung 5.4 und Gleichung 5.5 kann nun berechnet werden, wie viele Jahre der Sensorknoten in etwa mit einer Batterie betrieben werden kann. In Gleichung 5.4 werden für I_w die in Unterabschnitt 7.1.2 ermittelten 45 mA, für I_s die in Unterabschnitt 7.1.1 ermittelten 801 nA, für t_w die in Unterabschnitt 7.1.2 ermittelten 197,6 ms und für t_s $3600000 - (5 \cdot t_w)$ eingesetzt[4]. Demnach hat ein Sensorknoten pro Stunde einen Strombedarf von 0,013 mA. Wird der Strombedarf pro Stunde in Gleichung 5.5 für I_h und die Kapazität der Lithium-Batterie aus Unterabschnitt 6.1.5 in Höhe von 2200 mAh für C eingesetzt, so ergibt sich, dass die Sensorknoten in etwa 15,28 Jahre mit einer Batterie bei fünf Wach-Phasen pro Stunde eingesetzt werden können. Somit ist die Bedingung, dass die Sensorknoten mindestens acht Jahre mit einer Batterie betrieben werden können, erfüllt.

7.2 Kosten pro Sensorknoten

In Tabelle 7.1 ist die Auflistung der Kosten pro Sensorknoten zu sehen. In der Auflistung der Kosten wurde ein Mengenrabatt bei einer Abnahme der jeweiligen Hardware-Komponente von 1000 Stück berücksichtigt, um so eine praktische Umsetzung möglichst realitätsnah abzubilden[5]. Die einzige Hardware-Komponente, für die kein Mengenrabatt auf einschlägigen Plattformen ermittelt werden konnte, ist das DCF77-Modul. Demnach wurde der tatsächliche Einkaufspreis für das DCF77-Modul in Tabelle 7.1 aufgenommen. Die Summe der Kosten für die Hardware-Komponenten pro Sensorknoten lag zum Zeitpunkt der Arbeit bei 29,52 €. Im Vergleich zu Vega-Rodríguez et al. [12] wurden pro Sensorknoten 71,58 € eingespart (vgl. Abschnitt 1.2)[6]. Es ist jedoch zu berücksichtigen, dass in der Auflistung der Kosten in Tabelle 7.1 noch kein wetterfestes Gehäuse für die Sensorknoten einberechnet wurde. Das Potential, die Kosten weiter zu senken, ist insbesondere in den verwendeten Modulen, wie beispielsweise dem SX1276-LoRa-Modul, zu finden. Wird auf die Module verzichtet und die Komponenten stattdessen direkt auf dem PCB platziert, so ist zu vermuten, dass etwa 30–40 % der Kosten pro Modul eingespart werden können.

[4] Für t_s werden $3600000 - (5 \cdot t_w)$ eingesetzt, da sich der Strombedarf der Schlaf-Phase auf fünf Wach-Phasen pro Stunde bezieht.

[5] Für einige Komponenten war zum Zeitpunkt der Thesis lediglich ein Mengenrabatt bis zu einer Anzahl von unter 1000 Stück möglich. In dem Fall wurde der höchstmögliche Mengenrabatt verwendet.

[6] Vega-Rodríguez et al. [12] haben keinen Mengenrabatt in ihrer Auflistung angewendet.

Anz.	Name	Beschreibung	Stückpreis	Summe
1	STM32L062K8T6	Mikrocontroller	1,84 €	1,84 €
1	SX1276-Modul	LoRa-Modul	4,92 €	4,92 €
1	ABS07-120-32.768KHZ-T	Ext. Schwingquarz	0,73 €	0,73 €
1	AP7354-33W5-7	Spannungsregler	0,16 €	0,16 €
2	Taster 3,7 x 6,1 x 2,5mm	Tastschalter	0,05 €	0,10 €
1	SI2333DDS-T1-GE3	MOSFET SX1276-Modul	0,14 €	0,14 €
2	MMBT2222A-TP	Transistoren	0,02 €	0,04 €
17	Widerstände/Kondensatoren	Passive Bauelemente	0,005 €	0,09 €
1	RST-MA16-5008-23M-FY-001	Antenne (SMA)	2,17 €	2,17 €
1	RF2-04A-T-00-50-G	SMA-Steckverbinder	0,58 €	0,58 €
1	Tadiran Batteries SL-760S	Lithium-Batterie	3,79 €	3,79 €
1	BME280-Modul	Sensor	3,81 €	3,81 €
1	MAX30105-Modul	Sensor	6,10 €	6,10 €
1	DCF77-Modul	Zeitsynchronisation	4,90 €	4,90 €
1	Platine	-	0,15 €	0,15 €
			Summe:	29,52 €

Tabelle 7.1: Auflistung der Kosten der Hardware-Komponenten pro Sensorknoten

Es wird deutlich, dass die Kosten pro Sensorknoten bereits im Rahmen des prototypischen Aufbaus gering ausfallen. Somit sind die Sensorknoten beispielsweise auch für Projekte einsetzbar, in denen das finanzielle Budget eingeschränkt ist. Für ein TiReX-Netz, in dem die RNs mit SF9 senden, können bis zu 492 Sensorknoten eingesetzt werden (vgl. Tabelle 5.5). Demnach müssten für die Sensorknoten bei einem solchen Netz 14 627,16 € investiert werden[7]. In Abschnitt 7.1 wurde gezeigt, dass die Sensorknoten beim Verwenden von SF7 und fünf Wach-Phasen pro Stunde etwa 15 Jahre mit einer Batterie mit 2200 mAh Kapazität betrieben werden können. Demnach relativieren sich die Kosten für das gesamte TiReX-Netz, schließlich muss die Investition in Relation zur Laufzeit gesetzt werden. Dadurch, dass die RNs bezüglich der Hardware im Rahmen dieser Arbeit nicht weiter optimiert worden sind, müssen die Kosten nach einer Optimierung der Hardware-Komponenten hinzugerechnet werden. In Anbetracht dessen, dass die Anzahl notwendiger RNs im Vergleich zu der Anzahl der Sensorknoten gering ist, ist der Anteil der Kosten für die RNs vermutlich marginal.

[7] Die Auflistung der Kosten beinhaltet noch kein Gehäuse für die Sensorknoten.

7.3 Testlauf mit einem Relay Node

Da die finanziellen Möglichkeiten im Rahmen dieser Arbeit eingeschränkt waren, war es lediglich möglich, ein RN sowie zwei vollständige Sensorknoten für einen Testlauf einzusetzen. Da kein LoRaWAN-Gateway und somit keine Verbindung in ein bestehendes LoRaWAN, wie beispielsweise dem TTN, möglich war, wurden Logs von dem RN sowie ein Software Defined Radio (SDR) als Grundlage zur Evaluation verwendet. Die Sensorknoten wurden mit den Adressen *c0ffee00* und *c0ffee01* und mit SF8 konfiguriert[8]. Das RN hat die Nachrichten mit SF9 weitergeleitet. In einer tatsächlichen Umsetzung wäre vermutlich die Kombination aus SF7 für die Sensorknoten und SF9 für die RNs gewählt worden, was bereits in Unterabschnitt 5.1.3 deutlich wurde.

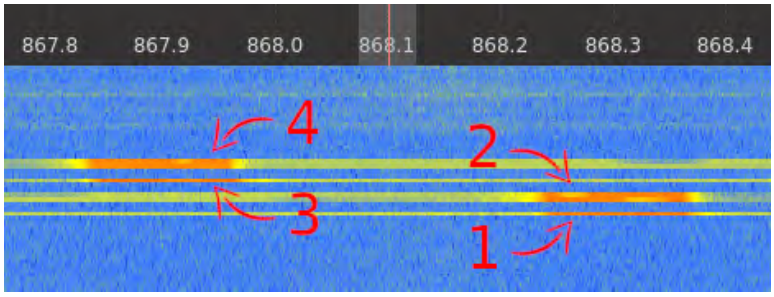

Abbildung 7.3: Bildschirmaufnahme der SDR-Software nach einer Testdurchführung mit zwei Sensorknoten auf zwei verschiedenen Frequenzen und einem Relay Node

In Abbildung 7.3 ist ein Testlauf mit zwei Sensorknoten und einem RN zu sehen. In Abbildung 7.3 ist ein Wasserfall-Diagramm abgebildet, in welchem die Zeit vertikal verläuft. Auf der horizontalen Achse ist die Frequenz in Megahertz zu sehen. Ein solches Diagramm wurde im Zusammenhang mit LoRa-Signalen bereits in Abschnitt 2.1 genauer betrachtet. Die jeweiligen Markierungen in Abbildung 7.3 kennzeichnen die einzelnen LoRaWAN-Nachrichten. Markierung 1 zeigt auf die LoRaWAN-Nachricht eines Sensorknotens, der auf einer Frequenz von 868,3 MHz gesendet hat. Markierung 3 zeigt auf die LoRaWAN-Nachricht des zweiten Sensorknotens, der auf einer Frequenz von 867,9 MHz gesendet hat. Dementsprechend wurde, wie in Unterabschnitt 5.2.4 gefordert, auf zwei unterschiedlichen Subbändern gesendet (vgl. Tabelle 2.1). Die zeitliche Verzögerung zwischen den Nachrichten der Sensorknoten wurde für eine bessere Übersicht absichtlich herbeigeführt, da der parallele Empfang in der SDR-Software undeutlich war. In der Praxis würden beide Sensorknoten parallel senden. Die Markierungen 2 und 4 zeigen

[8] Im Testlauf wurde SF8 für die Sensorknoten gewählt, da SF7 auf der Bildschirmaufnahme der SDR-Software zu undeutlich war.

auf die weitergeleiteten Nachrichten durch das RN. Zwischen den Nachrichten der Sensorknoten und den beiden Weiterleitungen ist ebenfalls eine Verzögerung zu erkennen, welche ebenso aus Gründen der Übersichtlichkeit absichtlich herbeigeführt wurde. In der Praxis würde direkt nach dem Empfang der LoRaWAN-Nachricht der Sensorknoten das Weiterleiten durch die RNs durchgeführt werden. In Abbildung 7.3 fällt auf, dass die benötigte Zeit für die Nachrichten der Sensorknoten und den Nachrichten der RNs variiert. Dieser Unterschied wurde durch die verschiedenen SFs hervorgerufen, was bereits in Unterabschnitt 2.1.1 erläutert wurde.

```
[2021-05-04 15:20:06]: RX from child node with DevAddr=c0ffee01
[2021-05-04 15:20:06]: TX (relay for DevAddr=c0ffee01)
[2021-05-04 15:20:06]: RX from child node with DevAddr=c0ffee00
[2021-05-04 15:20:06]: TX (relay for DevAddr=c0ffee00)
```

Abbildung 7.4: Ausschnitt des Logs des RNs nach einer Testdurchführung mit zwei Sensorknoten auf zwei verschiedenen Frequenzen

In Abbildung 7.4 ist zusätzlich eine Log-Ausgabe des RNs zu sehen. In der Log-Ausgabe wird deutlich, dass zuerst die LoRaWAN-Nachricht des Sensorknotens mit der Adresse *coffee01* empfangen und nach der zuvor erwähnten Wartezeit versendet wurde. Im Anschluss wurde die LoRaWAN-Nachricht des zweiten Sensorknotens empfangen und ebenso weitergeleitet. Der Ablauf in der Log-Ausgabe in Abbildung 7.4 entspricht demnach dem Ablauf, der durch die SDR-Software in Abbildung 7.3 bereits deutlich wurde.

Dieser Abschnitt hat verdeutlicht, dass das Prinzip der Weiterleitung von LoRaWAN-Nachrichten durch RNs in der Praxis umgesetzt werden kann. Der Testlauf wurde aufgrund finanzieller Einschränkungen lediglich mit einer minimalen Anzahl von TiReX-Teilnehmern durchgeführt. Trotz dessen ist deutlich geworden, dass eine Umsetzung der Weiterleitung von LoRaWAN-Nachrichten in der Praxis ohne großen Aufwand möglich ist.

8 Fazit

Im Rahmen dieser Arbeit wurde ein LoRaWAN-basiertes Sensornetz zur Bewertung einer Waldbrandgefahr entwickelt sowie eine Umsetzung mit Prototypen der Teilnehmer des Netzes evaluiert. Die vorgestellte LoRaWAN-Architektur wurde mit dem Ziel entworfen, sowohl die begünstigte Ausbreitung eines Waldbrandes vorherzusagen als auch einen Waldbrand durch geeignete Sensoren über eine Fläche von mehreren Quadratkilometern zu erkennen. Im folgenden Abschnitt wird die Arbeit zunächst kurz zusammengefasst, wobei im Anschluss mögliche Weiterentwicklungen und Optimierungansätze vorgestellt werden.

8.1 Zusammenfassung

Um Waldbrände mit Hilfe von Sensoren erkennen zu können, werden großflächige Netze benötigt. Solche Netze können hunderte Sensorknoten beinhalten, weshalb die Skalierbarkeit im Fokus steht. Ein mögliches Protokoll, das für solche großflächigen Netze eingesetzt werden kann, ist LoRaWAN. Zu Beginn der Arbeit war nicht bekannt, inwieweit ein LoRaWAN-basiertes Sensornetz für die Bewertung einer Waldbrandgefahr skaliert. Deshalb wurde zum Überprüfen der Skalierbarkeit von LoRaWAN der Simulator ns3 mit dem LoRaWAN-Modul von Van den Abeele et al. [8] verwendet. Um eine solche Simulation durchführen zu können, muss die Ausbreitung von LoRa-Signalen durch ein geeignetes Modell abgebildet werden. Da ein solches Modell für LoRa und typische europäische Wälder nicht ermittelt werden konnte, wurde ein empirisches Pfadverlustmodell eingesetzt, welches auf Basis eigener Messwerte in den Simulator ns3 implementiert wurde. Da in Wäldern mit einer schlechten Internetanbindung zu rechnen ist und die Anzahl einsetzbarer Gateways somit minimal sein muss, wurde die LoRaWAN-Erweiterung TiReX entwickelt. Mit TiReX ist es möglich, ein LoRaWAN in Gegenden mit schlechter Internetanbindung über eine Fläche von mehreren Quadratkilometern aufzuspannen. Die Entwicklung von TiReX war notwendig, da die aufgezeigten Alternativen, wie beispielsweise das Nutzen von ADR, die Skalierbarkeit des Netzes negativ beeinflusst hätten. Mit TiReX kann exakt geplant werden, wie viele Teilnehmer am LoRaWAN unter Betrachtung verschiedener Parameter, wie beispielsweise dem maximalen Duty Cycle, der maximal abdeckbaren Fläche und den jeweiligen LoRa-Parametern, teilnehmen können. Um die abdeckbare Fläche eines LoRaWANs mit lediglich einem Gateway erhöhen zu können, wurde in TiReX ein neuer LoRaWAN-Teilnehmer eingeführt, der als Relay Node (RN) bezeichnet wird. Ein RN hat die Aufgabe, LoRaWAN-Nachrichten von den Kindern seines Routing-Baums an das Gateway weiterzuleiten. Die Weiterleitung entspricht einer Kopie des Paketes, sodass beispielsweise auch die Adresse des Sensorknotens im Paket be-

stehen bleibt. Dadurch können Pakete im Verlauf der Verarbeitung zugeordnet werden. Da das resultierende LoRaWAN unter Umständen sehr umfangreich bezüglich der Anzahl der Sensorknoten sein kann, wird in TiReX zusätzlich der Medienzugriff der Sensorknoten durch eine Zeitschlitz-Erweiterung angepasst. Die Zeitschlitz-Erweiterung wurde unter der Bedingung entwickelt, dass der maximale Duty Cycle insbesondere der RNs ausgereizt, aber unbedingt eingehalten wird. Der Ablauf der Zeitschlitz-Erweiterung orientiert sich am Routing-Baum der Multi-Hop-Erweiterung und nutzt genügend große Toleranzzeiten, um das Einhalten des Duty Cycles gewährleisten zu können. Des Weiteren profitiert die Zeitschlitz-Erweiterung von parallelen Sende- und Empfangspfaden, die auf mehreren Subbändern aufgeteilt sind. Zusätzlich zur Protokollerweiterung wurde im Rahmen dieser Arbeit der Fokus insbesondere auf die Hardware-Komponenten der Sensorknoten gelegt. Für diesen Zweck wurde ein PCB entworfen, auf welchem alle Hardware-Komponenten des Sensorknotens platziert werden können. Da der Energieverbrauch der Sensorknoten insbesondere in der Schlaf-Phase möglichst niedrig sein sollte, wurden entsprechende Maßnahmen, wie beispielsweise das Trennen der Spannungsversorgung von einigen Modulen, auf dem PCB umgesetzt und anschließend evaluiert. Es hat sich herausgestellt, dass ein Sensorknoten mit den in Abschnitt 6.1 vorgestellten Hardware-Komponenten in etwa 15,28 Jahre mit lediglich einer Batterie betrieben werden kann, wobei die höchstmögliche Sendeleistung von 14 dBm und SF7 vorausgesetzt wurden. Ebenso wurden die Kosten in Höhe von 29,52 € pro Sensorknoten vorgestellt. Werden die Kosten für ein ganzes TiReX-Netz in Relation zur berechneten Laufzeit von 15,28 Jahre gesetzt, so wird deutlich, dass der finanzielle Aufwand auch für Projekte mit eingeschränktem finanziellen Budget akzeptabel ist. Zusätzlich wurde ein minimales TiReX-Netz aufgebaut und evaluiert. Es hat sich gezeigt, dass die theoretischen Ideen auch in der Praxis mit wenig Aufwand umgesetzt werden können.

8.2 Ausblick

Diese Arbeit deckt verschiedene Themenbereiche ab, wobei manche Themenbereiche einen interdisziplinären Charakter aufweisen. In zukünftigen Arbeiten können diese Themenbereiche weiter optimiert und evaluiert werden. In den folgenden Absätzen werden mögliche Aspekte vorgestellt, durch die die LoRaWAN-Architektur zusätzlich optimiert und evaluiert werden kann.

In Abschnitt 3.1 wurde beschrieben, dass für das Tewari-Modell Messungen in einem exemplarischen Wald durchgeführt wurden. Wie bereits in Abschnitt 3.1 erwähnt, wurden die Messungen aus zeitlichen Gründen lediglich in einem kleinen Umfang durchgeführt.

Es ist sehr wahrscheinlich, dass die ermittelten Parameter für die betrachtete Umgebung im Wald überangepasst sind. Des Weiteren wurde während der Messungen lediglich der SF verändert. Weitere Parameter, die in zukünftigen Messungen angepasst werden sollten, sind beispielsweise die Code Rate (CR), die Bandbreite und die Sendeleistung. Unabhängig der LoRa-Parameter ist auch anzumerken, dass Höhenunterschiede von Sender und Empfänger während der Messungen nicht weiter betrachtet wurden. Demnach wäre es von Interesse, inwieweit sich die Ergebnisse bei variierender Höhe unterscheiden. Werden die zuvor genannten Parameter ebenso während der Messungen verändert und die Messungen in verschiedenen Wäldern mit unterschiedlichen Witterungsbedingungen durchgeführt, so wäre es möglich, ein allgemeineres Tewari-Modell für LoRaWAN in Wäldern zu definieren. Möglicherweise ist die Simulation auch insoweit erweiterbar, dass topografische Eigenschaften in Form digitaler Geländepläne der Wälder berücksichtigt werden können. Eine Simulation auf Basis digitaler Geländepläne würde bei der Planung eines solchen Netzes viele Vorteile bieten und den Nutzen der Simulation weiter stärken.

In Abschnitt 1.1 wurde ein theoretisches Modell eines Waldbrandes erläutert und auf das angestrebte Netz übertragen. Inwieweit dieses theoretische Modell auch in der Praxis in unterschiedlichen Wäldern zutrifft, ist jedoch nicht endgültig geklärt. Ein geeigneter Ansatz wäre, das Waldbrand-Modell mit Forscher*innen aus den jeweiligen Fachgebieten genauer zu betrachten. Möglicherweise ist das angestrebte Sensornetz, das in Abschnitt 1.1 erläutert wurde, zu grob- oder engmaschig. Für das Sensornetz ist es aus finanzieller Sicht essentiell zu wissen, wie sich Waldbrände ausbreiten können, da sich sowohl die abdeckbare Fläche als auch die Anzahl der Sensorknoten aus dieser Information ableitet.

Auch in TiReX sind weitere Optimierungen möglich. In TiReX besitzen RNs einen statischen Routing-Baum. Damit jedes LoRaWAN-Gateway ohne Anpassungen im Zusammenhang mit TiReX verwendet werden kann, werden die Pakete der Sensorknoten nicht verändert. Daraus folgt wiederum, dass ein Gateway beim strikten Einhalten der Größe einer Region, Pakete von Sensorknoten doppelt erhalten kann. Dieses Phänomen tritt dann auf, wenn die konfigurierte Sendeleistung zu groß oder Bedingungen für eine bessere Funkwellenausbreitung existieren, wie beispielsweise bei einer dünneren Bewaldung in Richtung des Gateways. Dieser Fall tritt bei den Sensorknoten auf, die sich in Regionen befinden, die an die Gateway-Region grenzen. Sollen alle Regionen gleich groß sein, so müsste mit der derzeitigen TiReX-Version die Sendeleistung der Sensorknoten insoweit angepasst werden, dass nur das vorgesehene RN das Paket empfängt. Eine Alternative wäre, dass die Software auf dem Gateway TiReX-spezifisch

ist und das in Unterabschnitt 2.2.5 beschriebene FPort-Feld insofern verwendet wird, dass bei einer Weiterleitung durch ein RN ein spezifisches Bit gesetzt wird. Dadurch könnte das Gateway Pakete verwerfen, die nicht aus der Gateway-Region stammen. Das Verwerfen der Pakete könnte verhindert werden, indem sich das TiReX-Netz beim initialen Start dynamisch anpasst. Dazu müssten alle Sensorknoten beim initialen Start ein Paket senden, wobei überprüft wird, ob das Gateway ein Paket doppelt empfangen hat. Ist dies der Fall, müsste das Gateway mit dem für den Sensorknoten verantwortlichen RN kommunizieren. Auf Seite des RNs würde dann die Liste der zugeordneten Sensorknoten um den Sensorknoten verringert werden, dessen Nachrichten direkt vom Gateway empfangen werden können.

Eine weitere Optimierung von TiReX ist in der Synchronisation der Zeit möglich. Die prototypischen Sensorknoten, die in Abschnitt 6.1 vorgestellt wurden, besitzen alle ein DCF77-Modul. In Abschnitt 5.2 wurde beschrieben, dass zur besseren Flexibilität die Synchronisation der Zeit kein fester Bestandteil von TiReX sein soll. Möglicherweise ist eine optionale Erweiterung denkbar, in der vorausgesetzt wird, dass jedes RN eine Möglichkeit zur Synchronisation der Zeit besitzt. Sofern ein RN genügend häufig eine Möglichkeit zur Synchronisation bietet, könnte die Synchronisation der Zeit regionintern erfolgen. So würden die Kosten pro Sensorknoten zusätzlich gesenkt werden. Um diese optionale Erweiterung zu ermöglichen, muss zunächst evaluiert werden, welche Parametrisierung welchen Einfluss auf die Batterielaufzeit der Sensorknoten hätte. Eine Evaluation möglicher Optimierungen von TiReX ist Bestandteil zukünftiger Arbeiten.

Bezüglich der Hardware-Komponenten sind ebenso Optimierungen denkbar. Insbesondere der MAX30105, der zur Erkennung von Rauch in der Umgebungsluft eingesetzt wird, sollte ersetzt werden. Der Grund liegt darin, dass der Sensor mittlerweile nicht mehr produziert wird und die Verfügbarkeit am Markt stark eingeschränkt ist. Ein Vorbild für die kostengünstige Raucherkennung kann in handelsüblichen Rauchmeldern gefunden werden. Da der Vergleich der Sensoren unter Betrachtung unterschiedlichster Aspekte durchaus umfangreich ausfallen kann, bleibt dieser Themenbereich für spätere Arbeiten offen. Des Weiteren besitzen die Sensorknoten bisher keine Überwachung des Ladezustands der Batterie. Demnach sollte überprüft werden, inwieweit der Ladezustand bei den verwendeten Lithium-Thionylchlorid-Batterien ermittelt werden kann. Ebenso können die Sensorknoten um weitere Sensoren erweitert werden. Dadurch, dass in TiReX kontinuierlich gesendet wird, sind die ermittelten Daten vermutlich auch für wissenschaftliche Auswertungen über Umgebungsparameter in Wäldern von Interesse. Die Erweiterung der Sensorknoten um zusätzliche Sensoren kann für Forscher*innen von Nutzen sein und einen kontinuierlich wachsenden Datenbestand ermöglichen.

Im Rahmen dieser Arbeit war es aufgrund zeitlicher Einschränkungen nicht möglich, einen Langzeit- oder Stresstest bei unterschiedlichen Witterungsbedingungen für die Sensorknoten durchzuführen. Das Testen ist jedoch essentiell, schließlich sollen die Sensorknoten über mehrere Jahre bei unterschiedlichsten Witterungsverhältnissen eingesetzt werden. Das Testen der Software und Hardware sollte demnach in Zukunft im Fokus liegen.

Die Hardware der RNs stand im Rahmen dieser Arbeit nur geringfügig im Fokus. Zunächst existieren Möglichkeiten, leistungsärmere Hardware für die RNs einzusetzen. Zwar ist der Raspberry Pi Zero, der in Abschnitt 6.2 als Grundlage für das prototypische RN verwendet wurde, bereits ein Einplatinencomputer der leistungsärmeren Kategorie, jedoch könnte auch ein Mikrocontroller mit LoRa-Modulen für diesen Zweck ausreichen. Ein weiterer Aspekt, der kaum betrachtet wurde, ist die Spannungsversorgung der RNs. Für diesen Zweck sollte ein Modell über den Energieverbrauch der RNs entwickelt werden, um so geeignete Akkugrößen und Möglichkeiten zum Laden der Akkus, wie beispielsweise der Einsatz von Energy Harvesting, evaluieren zu können.

Es wird deutlich, dass viele Themenbereiche, die in dieser Arbeit behandelt wurden, weiter evaluiert und optimiert werden können. Es existieren demnach genügend Ansätze, um die vorgestellte Architektur zur Waldbranderkennung zu verfeinern und zu verbessern. Zukünftige Arbeiten können auch interdisziplinär erfolgen, um so den Nutzen der Architektur höchstmöglich ausreizen zu können.

Glossar

Activation By Personalization (ABP) Aktivierungsmethode in einem LoRaWAN. 16, 88

Adaptive Frequency Agility (AFA) Sieht vor, dass Teilnehmer eines Netzes die Frequenzen bei jeder Übertragung dynamisch wechseln. 14

Adaptive Data Rate (ADR) Verfahren in einem LoRaWAN, um die konfigurierte Datenrate automatisch zu optimieren. 4, 16, 17, 18, 42, 46, 47, 50, 55, 84, 114, 125, 129

Advanced Encryption Standard (AES) Blockchiffre, welche Ende 2000 als Nachfolger von DES veröffentlicht wurde. 15

Beacon Kleines Datenpaket, das in regelmäßigen Abständen versendet wird. 15

Breadboard Steckplatine für Hardwarekomponenten zum prototypischen Aufbau einer Schaltung. 24

Chirp Spread Spectrum (CSS) Modulationsverfahren, welches mit Chirp-Impulsen auf einem definierbaren Frequenzbereich arbeitet. 9, 24

Code Rate (CR) Definiert die FEC für ein LoRaWAN-Paket. 11, 12, 27, 42, 51, 116

Constrained Device Geräte, die stark eingeschränkt hinsichtlich ihrer Ressourcen, wie Speicher, Rechenleistung und Energieverbrauch, sind. 1, 13, 55, 65

Cyclic Redundancy Check (CRC) Prüfverfahren, um Fehler beispielsweise in einem Paket erkennen zu können. 12

DCF77 Zeitzeichensender in Mainflingen (Deutschland). 69, 70, 95, 96, 97, 98, 101, 102, 107, 109, 110, 117, 147

Destination-Sequenced Distance Vector routing (DSDV) Tabellenbasierter Routing-Algorithmus. 59

Duty Cycle Zeitabschnitt, in dem ein Gerät eines Funknetzes sendet. 14, 37, 41, 49, 55, 56, 58, 59, 60, 61, 62, 63, 65, 67, 69, 72, 73, 82, 85, 87, 114, 115, 127

Energy Harvesting Möglichkeit, elektrische Energie aus der Umgebung zu gewinnen. 56, 118

ESP32 Kostengünstige Mikrocontrollerfamilie von Espressif. 24, 136

Espressif Chinesische Firma, die Mikrocontroller und Bauteile herstellt. 119

Forward Error Correction (FEC) Verfahren, um Bitfehler aufgrund einer Übertragung von (digitalen) Daten korrigieren zu können. 11, 119

Frame Header (FHDR) Header in einem LoRaWAN-Paket, der Informationen vom Sender enthält. 18, 64, 65, 125

Frequency Division Multiple Access (FDMA) Multiplexverfahren, bei dem die Sender auf verschiedene Frequenzen aufgeteilt werden. 65

General Purpose Input/Output (GPIO) Kontaktstift an einem IC, der durch logische Programmierung als Eingabe- oder Ausgabekontakt bestimmt wird. 92, 103, 120

Global Positioning System Globales Navigationssatellitensystem zur Positionsbestimmung. 6, 24, 26, 27, 28, 30, 31, 70, 136

Integrierter Schaltkreis (IC) Elektronische Schaltung auf einem Halbleiter und ummantelt mit einem Füllmaterial. 12, 20, 27, 34, 51, 91, 98, 120

Inter-Integrated Circuit (I^2C) Serieller Datenbus. 147

Internet Protocol Version 6 (IPv6) Standardisiertes Verfahren zur Übertragung von Daten in paketvermittelnden Netzen. 58

Internet of Things (IoT) Beschreibt Geräte mit oft eingeschränkten Ressourcen, die meist an einem Netz angebunden sind. 1, 13, 58, 99, 121, 122

Joint Test Action Group (JTAG) Synonym für den Standard IEEE 1149.1 zum Testen und Debuggen von integrierten Schaltungen. 122

Light Emitting Diode (LED) Leuchtdiode aus anorganischem Halbleitermaterial. 94, 100

Listen Before Talk (LBT) Sieht vor, dass Teilnehmer eines Netzes vor dem Senden überprüfen, ob der Übertragungskanal frei ist. 14

Lithium-Ionen-Akku (Li-Ion-Akku) Akkumulator, dessen Grundbaustein Lithium ist. 52, 53

Long Range (LoRa) Im LoRaWAN Protokoll-Stack für die Aufgaben der Bitübertragungsschicht verantwortlich. 1, 9, 10, 11, 12, 13, 17, 20, 22, 24, 25, 26, 27, 35, 36, 37, 38, 40, 42, 46, 50, 53, 54, 75, 77, 98, 103, 104, 105, 108, 110, 112, 114, 116, 118, 122, 125, 126, 129, 136, 137

Long Range Wide Area Network (LoRaWAN) LPWAN-Netzprotokoll spezifiziert von der LoRaAlliance. 1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 27, 33, 35, 36, 38, 39, 40, 41, 42, 44, 47, 48, 49, 50, 55, 56, 57, 58, 59, 60, 62, 63, 64, 65, 66, 67, 70, 71, 74, 76, 84, 85, 87, 89, 101, 102, 104, 112, 113, 114, 115, 116, 119, 120, 121, 123, 125, 127, 129, 136

LoRa Alliance Non-Profit-Bündnis verschiedener Unternehmen mit dem Ziel, LoRaWAN weiterzuentwickeln. 13, 120

Low Dropout Regulator (LDO) Linearer Spannungsregler. 54

Low Data Rate Optimization Verfahren von Semtech, um die Anzahl zu übertragender Bits für hohe Spreading Factors zu reduzieren. 51

Low Power Wide Area Network (LPWAN) Typ von Netzprotokollen für das kabellose Senden von Daten über lange Distanz mit geringem Energieverbrauch und geringer Datenrate. 1, 2, 49, 65, 120

Message Integrity Code (MIC) Häufig verwendetes Synonym für einen Message Authentication Code, welcher die Integrität und Authentizität einer Nachricht gewährleistet. 15, 18

Methode der kleinsten Quadrate (MKQ) Methode, um auf Basis von Datenpunkten eine Funktion bestmöglich nachzubilden. 32

NMEA 0183 Standard der National Marine Electronics Association für die Kommunikation von GPS-Modulen mit Endgeräten. 70

Over The Air Activation (OTAA) Aktivierungsmethode für IoT-Geräte, welche zum Beispiel in LoRaWAN verwendet werden kann. 16, 17, 18, 88

Packet Delivery Ratio (PDR) Kennzahl, für die die Anzahl versendeter Pakete mit der Anzahl erfolgreich empfangener Pakete ins Verhältnis gesetzt wird. 44, 45, 46, 47, 48, 65, 84, 85, 86, 127, 143

parts per million (ppm) Anteil pro Million, wobei $1ppm = 1 * 10^{-6}$ entspricht. 69

Printed Circuit Board (PCB) Leiterplatte zum Befestigen und Verbinden von elektronischen Bauteilen. 24, 25, 26, 93, 96, 97, 100, 110, 115, 121, 125, 126, 136, 148

Real Time Clock (RTC) Eine meist in Hardware implementierte Echtzeituhr. 68, 89, 102, 107

Received Signal Strength Indication (RSSI) Indikator, um die Empfangsstärke in einem Funknetz anzugeben. 27, 31, 32, 34, 137

Relay Node (RN) Dedizierter Multi-Hop-Knoten in TiReX. 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 102, 103, 104, 105, 106, 111, 112, 113, 114, 115, 116, 117, 118, 125, 126, 127, 129

RIOT Speziell für den IoT-Bereich entwickeltes Betriebssystem, welches auf die Anforderungen in diesem Bereich zugeschnitten ist. 26, 89, 90, 99, 100, 102, 127, 146

Rückflussdämpfung Verhältnis von gesendeter zu reflektierter Leistung. 25

Semtech Halbleiterhersteller ansässig in Kalifornien, USA. 9, 38, 50, 51, 70, 89, 121

Serial Wire Debug (SWD) Alternative Schnittstelle für das Protokoll JTAG. 98, 122

Short Range Devices (SRD) Funkgeräte, die eine geringe Sendeleistung aufweisen und deshalb nur über geringe Distanzen senden können. 13, 14, 49, 55

Signal to Noise Ratio (SNR) Verhältnis zwischen dem Nutzsignal und dem Störsignal. 36

Software Defined Radio (SDR) Hochfrequenz-Sender und -Empfänger, in denen ein Teil der Signalverarbeitung in Software statt in Hardware implementiert ist. 112, 113, 126

Spreading Factor (SF) Spreizfaktor des LoRa-Signals. 4, 5, 10, 11, 12, 16, 17, 26, 27, 29, 30, 31, 32, 33, 34, 35, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 57, 58, 61, 62, 63, 67, 68, 71, 72, 77, 80, 84, 85, 86, 87, 104, 111, 112, 113, 115, 116, 125, 127, 129, 137, 143

Stehwellenverhältnis Maß für die stehende Welle, die durch eine Reflexion im Leiter entsteht. 25

SubMiniature Version A (SMA) Schraubenkopplung für das Anschließen einer Antenne. 24

Temperaturkompensierte Quarzoszillatoren (TCXO) Quarzoszillatoren mit analoger oder digitaler Frequenzanpassung auf Basis von Temperaturmessungen. 69, 71, 97, 108

Temperaturstabilisierte Quarzoszillatoren (OCXO) Quarzoszillatoren mit einem Thermostat zur Stabilisierung der Temperatur und somit Erhöhen der Ganggenauigkeit. 69

The Things Network (TTN) Weit verbreitete und freie Umsetzung von LoRaWAN. 14, 49, 104, 112

Time-slotted Region-based Extension for LoRaWAN (TiReX) Erweiterung von LoRaWAN mit einem Multi-Hop-Ansatz und einem Zeitschlitz-Verfahren. 73, 75, 85, 86, 87, 88, 95, 97, 101, 106, 111, 113, 114, 115, 116, 117, 122, 127, 129, 147

Time Division Multiple Access (TDMA) Multiplexverfahren, bei dem ein Sender zu definierten Zeitpunkten senden darf. 65

Time-Slotted Channel Hopping (TSCH) Methode für ein Zugriffsverfahren in Netzen (siehe IEEE 802.15.4e). 65

Time on Air (TOA) Zeit, die ein Teilnehmer in einem Funknetz für eine Übertragung benötigt. 11, 14, 16, 42, 45, 46, 47, 59, 63

Unique Identifier (UID) Eindeutige Folge von Zeichen für Identifikationszwecke. 65

Universal Asynchronous Receiver Transmitter (UART) Serielle Schnittstelle zum Senden und Empfangen von Daten. 98

Vector Network Analyzer (VNA) Messgerät, um die Leistung eines Gerätes (z.B. eine Antenne) oder Netzes auf einer bestimmten Frequenz zu messen. 25

Wireless Personal Area Network (WPAN) Im Standard IEEE 802.15 definiertes Netzkonzept für das Tragen von Kommunikationsgeräten im persönlichen Bereich. 9, 123

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Literaturverzeichnis

[1] Bormann, Carsten; Ersue, Mehmet; Keränen, Ari (2014): Terminology for Constrained-Node Networks. RFC 7228; RFC Editor. (= Request for Comments). Online unter: https://rfc-editor.org/rfc/rfc7228.txt. DOI: 10.17487/RFC7228.

[2] Raza, U.; Kulkarni, P.; Sooriyabandara, M. (2017): Low Power Wide Area Networks: An Overview. In: IEEE Communications Surveys Tutorials 19, S. 855–873. DOI: 10.1109/COMST.2017.2652320.

[3] LoRa-Alliance (2017): LoRaWAN™ 1.1 Specification. Online unter: https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf (Abgerufen am 2020-11-25).

[4] Semtech (2015): LoRa™ Modulation Basics. Online unter: https://www.semtech.com/uploads/documents/an1200.22.pdf (Abgerufen am 2020-12-14).

[5] Abramson, Norman (1977): THE ALOHA SYSTEM: another alternative for computer communications. In: Fall Joint Computer Conference 37, S. 281–285. DOI: 10.1145/1478462.1478502.

[6] Anderson, Hal E. (1983): Predicting wind-driven wild land fire size and shape.

[7] Varsier, N.; Schwoerer, J. (2017): Capacity limits of LoRaWAN technology for smart metering applications. In: 2017 IEEE International Conference on Communications (ICC). S. 1–6.

[8] Van den Abeele, F. et al. (2017): Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3. In: IEEE Internet of Things Journal 4, S. 2186–2198.

[9] Adelantado, Ferran et al. (2017): Understanding the limits of LoRaWAN. In: IEEE Communications Magazine 55. DOI: 10.1109/MCOM.2017.1600613.

[10] Farhad, A.; Kim, D.; Pyun, J. (2019): Scalability of LoRaWAN in an Urban Environment: A Simulation Study. In: 2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN). S. 677–681. DOI: 10.1109/ICUFN.2019.8806140.

[11] Semtech (2019): Understanding the LoRa® Adaptive Data Rate.

[12] Vega-Rodríguez, R. et al. (2019): Low Cost LoRa based Network for Forest Fire Detection. In: 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS). S. 177–184.

[13] Adnan et al. (2018): Forest Fire Detection using LoRa Wireless Mesh Topology. In: 2018 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT). S. 184–187. DOI: 10.1109/EIConCIT.2018.8878488.

[15] Rizanov, S.; Stoynova, A.; Todorov, D. (2019): System For Early Warning And Monitoring Of Wildfires. In: 2019 IEEE XXVIII International Scientific Conference Electronics (ET). S. 1–3.

[15] Rizanov, S.; Stoynova, A.; Todorov, D. (2019): System For Early Warning And Monitoring Of Wildfires. In: 2019 IEEE XXVIII International Scientific Conference Electronics (ET). S. 1–3.

[16] Knight, Matthew W.; Seeber, Balint (2016): Decoding LoRa: Realizing a Modern LPWAN with SDR.

[17] Robyns, Pieter et al. (2018): A Multi-Channel Software Decoder for the LoRa Modulation Scheme. S. 41–51. DOI: 10.5220/0006668400410051.

[18] Semtech (2013): Implementing Data Whitening and CRC Calculation in Software on SX12xx Devices. Online unter: https://www.semtech.com/uploads/documents/AN1200.18_STD.pdf (Abgerufen am 2021-03-09).

[19] IEEE (2007): IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs): Amendment 1: Add Alternate PHYs. In: IEEE Std 802.15.4a-2007 (Amendment to IEEE Std 802.15.4-2006), S. 1–210. DOI: 10.1109/IEEESTD.2007.4299496.

[20] Liando, Jansen et al. (2019): Known and Unknown Facts of LoRa: Experiences from a Large-scale Measurement Study. In: ACM Transactions on Sensor Networks 15, S. 1–35. DOI: 10.1145/3293534.

[21] Europäisches Institut für Telekommunikationsnormen: Short Range Devices (SRD) operatingin the frequency range 25 MHz to 1.000 MHz; Part 1: Technical characteristics and methods of measurement. Online unter: https://www.etsi.org/deliver/etsi_en/300200_300299/30022001/03.01.01_60/en_30022001v030101p.pdf (Abgerufen am 2020-12-21).

[22] Croce, Daniele et al. (2017): Impact of Spreading Factor Imperfect Orthogonality in LoRa Communications. In: Piva, Alessandro/Tinnirello, Ilenia/Morosi, Simone (Hg.): Digital Communication. Towards a Smart and Secure Future Internet. Cham: Springer International Publishing. S. 165–179.

[23] Bundesnetzagentur: Allgemeinzuteilung von Frequenzen zur Nutzung durch Funkanwendungen geringer Reichweite (SRD). Online unter: https://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Frequenzen/Allgemeinzuteilungen/2018_05_SRD_pdf.pdf (Abgerufen am 2021-01-09).

[24] IEEE (2006): IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). In: IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003), S. 1–320. DOI: 10.1109/IEEESTD.2006.232110.

[25] Iwata, Tetsu et al. (2006): The AES-CMAC Algorithm. RFC 4493; RFC Editor. (= Request for Comments). Online unter: https://rfc-editor.org/rfc/rfc4493.txt. DOI: 10.17487/RFC4493.

[26] Wehrle, Klaus; Günes, Mesut; Gross, James (2010): Modeling and Tools for Network Simulation. Berlin Heidelberg: Springer Science & Business Media.

[27] Jazebi, S.; de León, F.; Nelson, A. (2020): Review of Wildfire Management Techniques—Part I: Causes, Prevention, Detection, Suppression, and Data Analytics. In: IEEE Transactions on Power Delivery 35, S. 430–439. DOI: 10.1109/TPWRD.2019.2930055.

[28] San-Miguel-Ayanz, J. et al. (2012): Comprehensive Monitoring of Wildfires in Europe: The European Forest Fire Information System (EFFIS). In: Approaches to Managing Disaster - Assessing Hazards, Emergencies and Disaster Impacts. DOI: 10.5772/28441.

[29] Chaparro, David et al. (2016): Predicting the Extent of Wildfires Using Remotely Sensed Soil Moisture and Temperature Trends. In: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 9, S. 2818–2829. DOI: 10.1109/JSTARS.2016.2571838.

[30] Gustrau, Frank (2013): Hochfrequenztechnik: Grundlagen der mobilen Kommunikationstechnik. München: Hanser.

[31] Azevedo, J. A. R.; Santos, F. E. S. (2011): An Empirical Propagation Model for Forest Environments at Tree Trunk Level. In: IEEE Transactions on Antennas and Propagation 59, S. 2357–2367. DOI: 10.1109/TAP.2011.2143664.

[32] Palaios, A.; Labou, Y.; Mähönen, P. (2014): A Study on the Forest Radio Propagation Characteristics in European Mixed Forest Environment. In: 2014 IEEE Military Communications Conference. S. 376–381. DOI: 10.1109/MILCOM.2014.68.

[33] Tewari, R. K.; Swarup, S.; Roy, M. N. (1990): Radio wave propagation through rain forests of India. In: IEEE Transactions on Antennas and Propagation 38, S. 433–449. DOI: 10.1109/8.52261.

[34] Lee, H.; Ke, K. (2018): Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation. In: IEEE Transactions on Instrumentation and Measurement 67, S. 2177–2187. DOI: 10.1109/TIM.2018.2814082.

[35] Karney, Charles F. F. (2012): Algorithms for geodesics. In: Journal of Geodesy 87, S. 43–55. DOI: 10.1007/s00190-012-0578-z.

[36] Bankov, D.; Khorov, E.; Lyakhov, A. (2017): Mathematical model of LoRaWAN channel access. In: 2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM). S. 1–3. DOI: 10.1109/WoWMoM.2017.7974300.

[37] Semtech: SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver. Online unter: https://www.mouser.com/datasheet/2/761/sx1276-1278113.pdf (Abgerufen am 2021-01-09).

[38] Mekki, K. et al. (2018): Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT. In: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). S. 197–202. DOI: 10.1109/PERCOMW.2018.8480255.

[40] Ma, Shuai et al. (2018): Temperature effect and thermal impact in lithium-ion batteries: A review. In: Progress in Natural Science: Materials International 28, S. 653–666. DOI: 10.1016/j.pnsc.2018.11.002.

[40] Ma, Shuai et al. (2018): Temperature effect and thermal impact in lithium-ion batteries: A review. In: Progress in Natural Science: Materials International 28, S. 653–666. DOI: 10.1016/j.pnsc.2018.11.002.

[41] Deline, C. et al. (2019): Field-Aging Test Bed for Behind-the-Meter PV + Energy Storage. In: 2019 IEEE 46th Photovoltaic Specialists Conference (PVSC). S. 1341–1345. DOI: 10.1109/PVSC40753.2019.8980775.

[42] Tanenbaum, Andrew S. (2012): Computernetzwerke. Pearson Studium.

[44] Brandt, A. et al. (2012): RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC Editor. Online unter: https://www.rfc-editor.org/info/rfc6550. DOI: 10.17487/rfc6550.

[44] Brandt, A. et al. (2012): RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC Editor. Online unter: https://www.rfc-editor.org/info/rfc6550. DOI: 10.17487/rfc6550.

[45] Sartori, B. et al. (2017): Enabling RPL multihop communications based on LoRa. In: 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). S. 1–8. DOI: 10.1109/WiMOB.2017.8115756.

[46] Osorio, A. et al. (2020): Routing in LoRaWAN: Overview and Challenges. In: IEEE Communications Magazine 58, S. 72–76. DOI: 10.1109/MCOM.001.2000053.

[47] Dias, José; Grilo, António (2018): LoRaWAN multi-hop uplink extension. In: Procedia Computer Science 130, S. 424–431. DOI: 10.1016/j.procs.2018.04.063.

[48] IEEE Standard for Low-Rate Wireless Networks. In: IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011), S. 1–709. DOI: 10.1109/IEEESTD.2016.7460875.

[49] Zorbas, Dimitrios et al. (2020): TS-LoRa: Time-slotted LoRaWAN for the Industrial Internet of Things. In: Computer Communications 153, S. 1–10. DOI: 10.1016/j.comcom.2020.01.056.

[50] Zorbas, Dimitrios (2020): Design Considerations for Time-Slotted LoRa(WAN). In: Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks on Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks. USA: Junction Publishing. S. 271–276. (= EWSN '20).

[51] Rost, Manfred; Wefel, Sandro (2013): Elektronik für Informatiker. Deutschland: Oldenbourg.

[52] ECS Inc.: ECS-3X8X, 2X6X,1X5X 32.768 KHz Tuning Fork Crystal. Online unter: https://ecsxtal.com/store/pdf/ECS-3x8X (Abgerufen am 2021-01-15).

[53] Neubig, Bernd; Briese, Wolfgang (1997): Das grosse Quarz-Kochbuch – Quarze, Quarzoszillatoren, Quarz- und Oberflächenwellenfilter (SAW), Messtechnik. München: Franzis.

[54] Piester, Dirk; Hetzel, Peter; Bauch, Andreas (2004): Zeit- und Normalfrequenzverbreitung mit DCF77. In: PTB-Mitteilungen 114, S. 345–368.

[55] Micro Crystal Switzerland: Methoden zur Verbesserung der Ganggenauigkeit bei Echtzeituhren (RTC). Online unter: https://www.all-electronics.de/wp-content/uploads/migrated/article-pdf/113375/174ag0206.pdf (Abgerufen am 2020-12-21).

[56] Mokhtari, Z; Holé, S; Lewiner, J (2013): Study of an ionic smoke sensor. In: Measurement Science and Technology 24, S. 055006. DOI: 10.1088/0957-0233/24/5/055006.

[57] Kruell, Wolfgang et al. (2013): Analysis of Dust Properties to Solve the Complex Problem of Non-fire Sensitivity Testing of Optical Smoke Detectors. In: Procedia Engineering 62, S. 859–867. DOI: https://doi.org/10.1016/j.proeng.2013.08.136.

[58] Litton, Charles D. (2009): Laboratory evaluation of smoke detectors for use in underground mines. In: Fire Safety Journal 44, S. 387–393. DOI: 10.1016/j.firesaf.2008.08.008.

[59] Wu, Fan; Rüdiger, Christoph; Yuce, Mehmet (2017): Real-Time Performance of a Self-Powered Environmental IoT Sensor Network System. In: Sensors 17, S. 282. DOI: 10.3390/s17020282.

[60] Dittrich, Thomas et al. (2012): Lithium Batteries for Wireless Sensor Networks.

Anhang

A Schaltplan für die Platinen der Messdurchführung

In Abbildung A.1 ist der Schaltplan für die Platinen zur Messdurchführung aus Kapitel 3 zu sehen. Auf der Platine wird ein ESP32 als Mikrocontroller verwendet. Zum Versenden der LoRaWAN-Nachrichten wird ein SX1276-LoRa-Modul verlötet. Des Weiteren sind Anschlüsse für ein GPS-Modul vorhanden.

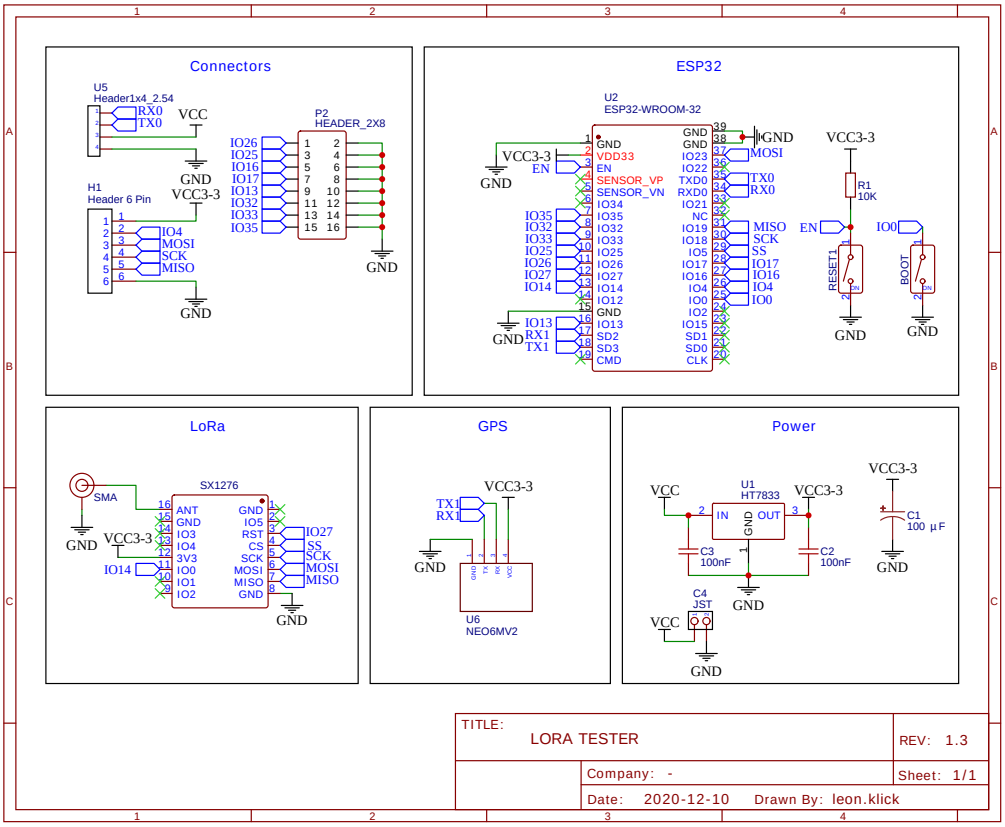

Abbildung A.1: Schaltplan für PCBs der Messdurchführung mit ESP32 und SX1276 LoRa-Modul

B Messergebnisse

Im Folgenden sind die Messergebnisse der in Kapitel 3 beschriebenen Messungen für alle sechs SFs (SF7–SF12) zu sehen. Des Weiteren ist pro Abbildung das aus den Messergebnissen abgeleitete Tewari-Modell dargestellt. Der Schnittpunkt, der durch die beiden gestrichelten Linien auf dem Tewari-Modell entsteht, ist die vermutete maximale Reichweite auf Basis des minimalen RSSI-Wertes für das SX1276-LoRa-Modul.

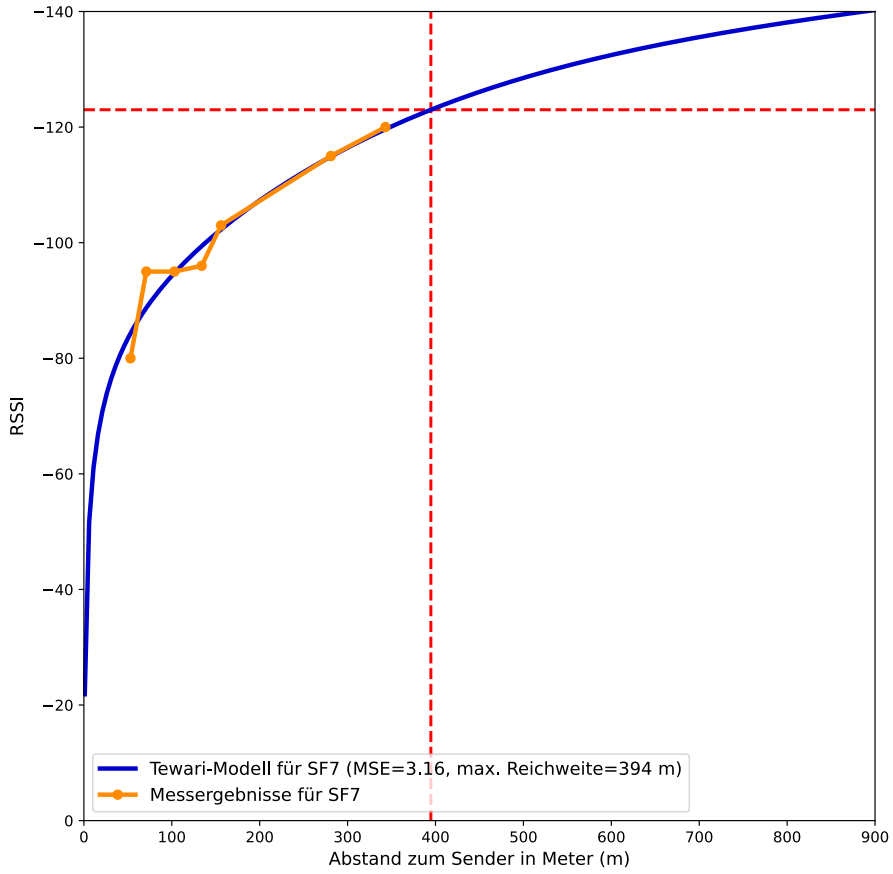

Abbildung B.2: Ergebnis der Messung für SF7 und das Tewari-Modell mit angepassten Parametern A, B und α

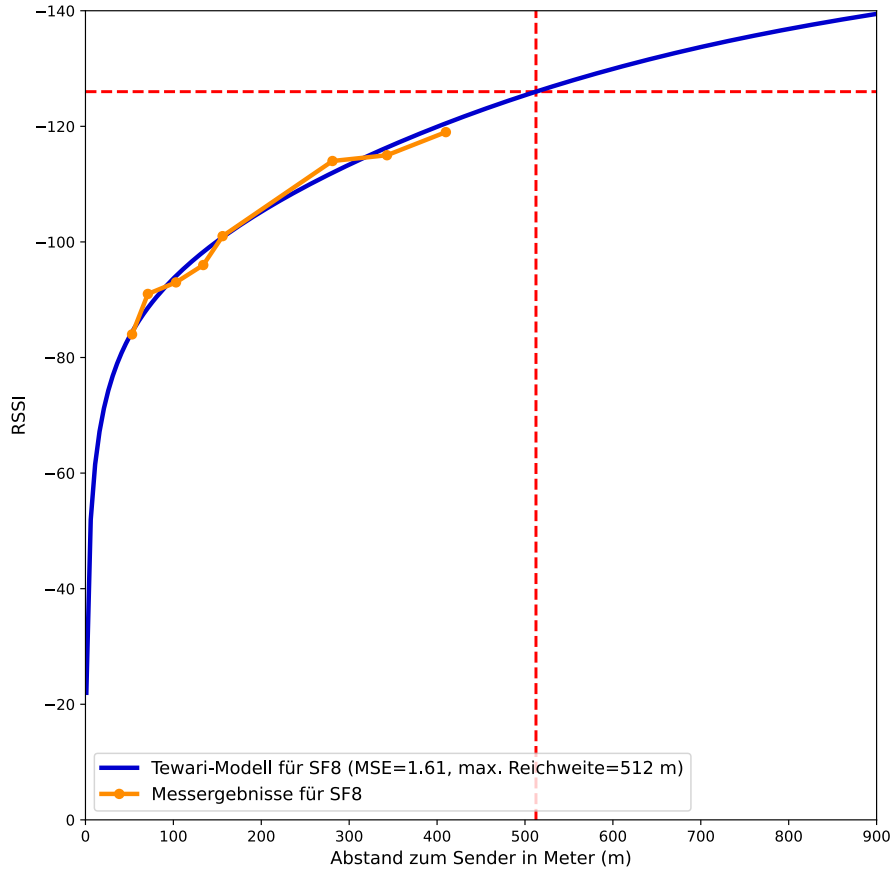

Abbildung B.3: Ergebnis der Messung für SF8 und das Tewari-Modell mit angepassten Parametern A, B und α

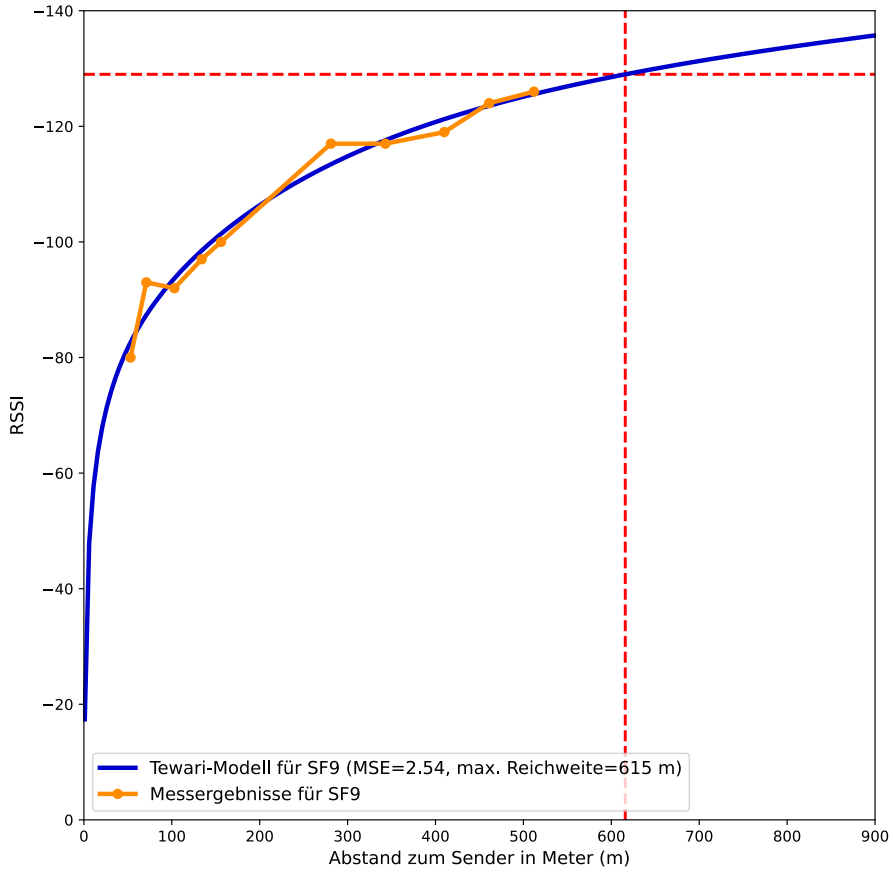

Abbildung B.4: Ergebnis der Messung für SF9 und das Tewari-Modell mit angepassten Parametern A, B und α

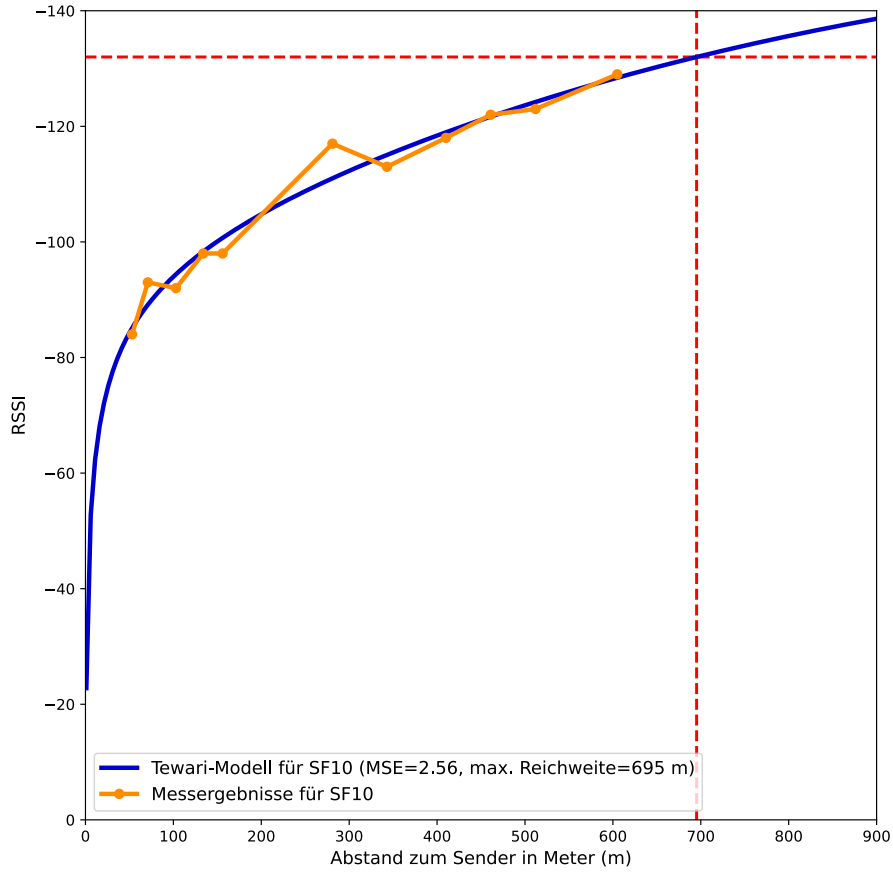

Abbildung B.5: Ergebnis der Messung für SF10 und das Tewari-Modell mit angepassten Parametern A, B und α

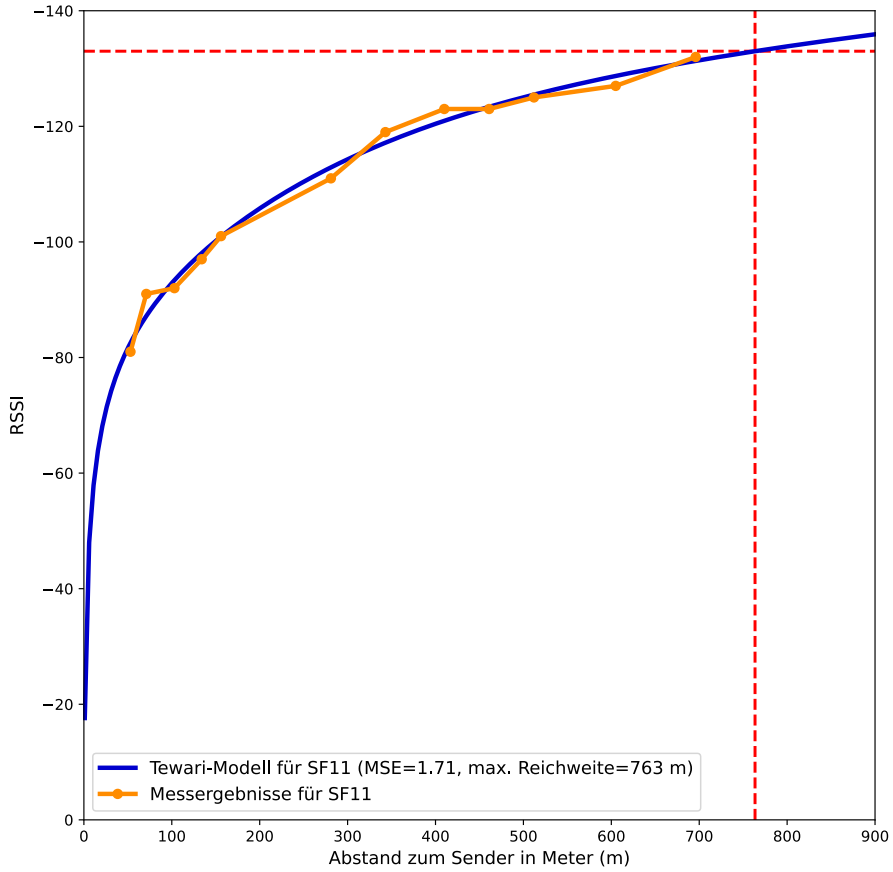

Abbildung B.6: Ergebnis der Messung für SF11 und das Tewari-Modell mit angepassten Parametern A, B und α

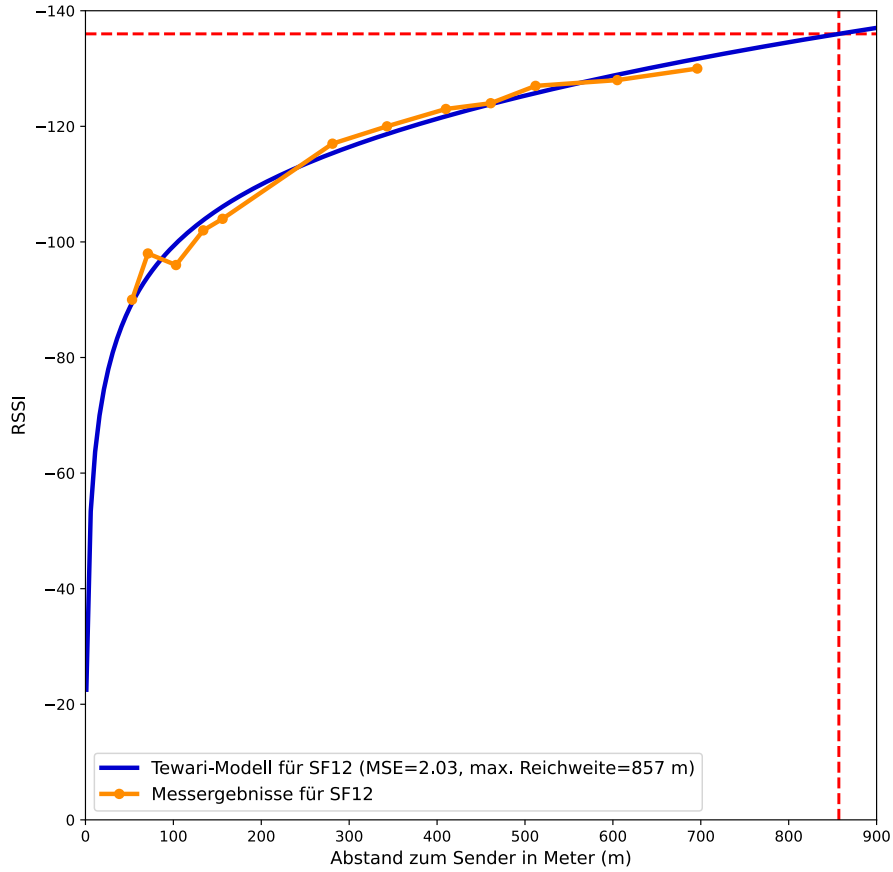

Abbildung B.7: Ergebnis der Messung für SF12 und das Tewari-Modell mit angepassten Parametern A, B und α

C Ergebnisse der initialen Simulationsdurchläufe

Im Folgenden sind die Ergebnisse der Simulationsdurchläufe pro SF (SF7–SF12) aus Kapitel 4 aufgelistet. Pro Listing ist zu erkennen, wie viele Pakete vom Gateway versendet wurden (*nPackets_macTx*), wie viele Pakete von den Sensorknoten in Summe empfangen wurden (*nPackets_macRx*), welche PDR aus dem Verhältnis der beiden zuvor genannten Kennzahlen folgt und wie viele Pakete nicht von den Sensorknoten empfangen werden konnten (*nPackets_phyRxDrop*).

Listing C.1 Ergebnis der Simulation für SF7

```
PDR STATS:
    nPackets_macRx=5600
    nPackets_macTx=6522
    PDR=0.858632

ADDITIONAL STATS:
    nPackets_phyRxDrop=922
```

Listing C.2 Ergebnis der Simulation für SF8

```
PDR STATS:
    nPackets_macRx=5177
    nPackets_macTx=6522
    PDR=0.793775

ADDITIONAL STATS:
    nPackets_phyRxDrop=1345
```

Listing C.3 Ergebnis der Simulation für SF9

```
PDR STATS:
    nPackets_macRx=4535
    nPackets_macTx=6522
    PDR=0.695339

ADDITIONAL STATS:
    nPackets_phyRxDrop=1987
```

Listing C.4 Ergebnis der Simulation für SF10

```
PDR STATS:
    nPackets_macRx=3289
    nPackets_macTx=6522
    PDR=0.504293

ADDITIONAL STATS:
    nPackets_phyRxDrop=3233
```

Listing C.5 Ergebnis der Simulation für SF11

```
PDR STATS:
    nPackets_macRx=2440
    nPackets_macTx=6522
    PDR=0.374118

ADDITIONAL STATS:
    nPackets_phyRxDrop=4082
```

Listing C.6 Ergebnis der Simulation für SF12

```
PDR STATS:
    nPackets_macRx=1428
    nPackets_macTx=6522
    PDR=0.218951

ADDITIONAL STATS:
    nPackets_phyRxDrop=5094
```

D Anforderungen an den Mikrocontroller

In Tabelle D.1 ist ein Vergleich zwischen den in RIOT verfügbaren Mikrocontrollern zu sehen. Mikrocontroller, die zum Zeitpunkt der Hardwarebeschaffung nicht beim Online-Händler Digi-Key verfügbar waren, wurden zur besseren Übersicht aus Tabelle D.1 entfernt. Der Vergleich fand auf Basis der jeweiligen Datenblätter statt. Sofern ein Datenblatt keine Informationen zu einem Attribut liefern konnte, wurde ein Fragezeichen in die jeweilige Zelle eingesetzt.

Mikrocontroller	Preis (€)	RTC?	$\geq$ 64 KiB Flash	Ruhestrom (µA)
Atmel ATmega1284p	4,18	ja	ja	0,6
Atmel ATmega128rfa1	4,57	ja	ja	0,25
Atmel ATmega2560	9,37	ja	ja	4
Atmel ATmega256rfr2	4,77	ja	ja	0,7
Atmel ATmega328p	1,03	ja	nein	1,3
Atmel ATmega32u4	3,23	ja	nein	?
Atmel SAM3	2,53	ja	ja	1
Atmel SAMD21	1,36	ja	ja	5
Atmel SAML21	2,51	ja	ja	?
Espressif ESP32	2,06	ja	ja	5
Espressif ESP8266	2,48	nein	ja	20
Microchip SAMD5x	3,23	ja	ja	3
Microchip SAML1x	1,78	ja	ja	0,5
NXP Kinetis (K32L)	3,94	ja	ja	1,96
Nordic nRF51	2,25	ja	ja	1,2
Nordic nRF52	1,32	ja	ja	1,5
MT PIC32MX	2,61	ja	ja	44
MT PIC32MZ	6,41	ja	ja	?
ST STM32L0	1,84	ja	ja	0,8
ST STM32L1	2,27	ja	ja	1,38
ST STM32L4	1,72	ja	ja	0,95
ST STM32L5	4,54	ja	ja	0,2
Silicon Labs's EZR32	5,07	ja	ja	0,7
TI CC2538	3,7	ja	ja	1,3
TI CC13x0	2,86	ja	ja	0,7
TI CC13x2	3,34	ja	ja	0,85

Tabelle D.1: Auszug der unterstützten Mikrocontroller in RIOT sowie Überprüfung der Anforderungen an den Mikrocontroller eines Sensorknotens

E Schaltplan für die Platinen der Sensorknoten

In Abbildung E.8 ist der Schaltplan für die Sensorknoten aus Kapitel 6 zu sehen. Als Grundlage dient ein Mikrocontroller STM32L062K8T6, über den beispielsweise mit den Sensoren kommuniziert wird. Des Weiteren werden Schnittstellen für zwei Sensoren (BME280, MAX30105) bereitgestellt. Beide Sensoren kommunizieren mit Hilfe des Protokolls I^2C. Für die Umsetzung der Zeitschlitzerweiterung in TiReX wird ein externer 32768 kHz Schwingquarz genutzt, wobei die Synchronisation der Uhrzeit mit Hilfe eines DCF77-Moduls durchgeführt wird.

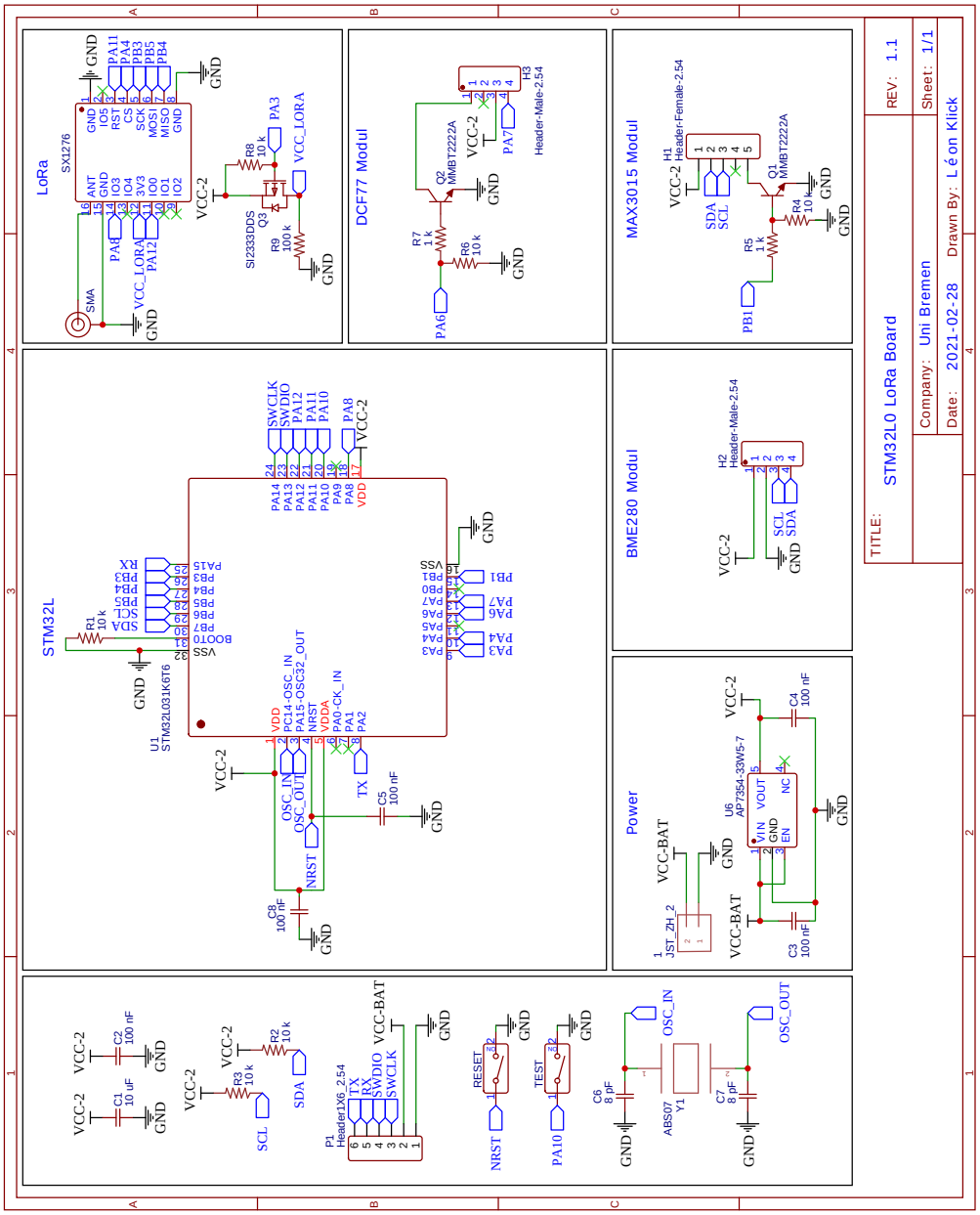

Abbildung E.8: Schaltplan für PCBs der Sensorknoten