Demodulation von LoRa-Signalen mit LimeSDR Mini

Yannick Kulli

Thursday, December 21, 2023

Industrieprojekt an der Hochschule Luzern - Technik & Architektur

Titel	**Demodulation von LoRa-Signalen mit LimeSDR Mini**
Studentin/Student	**Kulli, Yannick**
Bachelor-Studiengang	**Bachelor Elektrotechnik und Informationstechnologie**
Semester	**HS23**
Dozentin/Dozent	**Hunziker, Thomas**
Zweitdozentin/-dozent	**Schuster, Kilian**

Abstract Deutsch
In dieser Arbeit wird mittels eines «Software Defined Radios» und einem LoRa-Sender eine Datenübertragung an das LoRaWan eingefangen und anschliessend demoduliert. Der Fokus liegt dabei auf der zeitlichen Synchronisation zwischen dem Sender und dem Emfänger. Des weiteren kommt es zu Frequenzabweichungen gegenüber dem perfekten Frequenzband. Auch dieser Effekt führt zu einer fehlerhaften Demodulation des Signals. Zur Korrektur dieser Phänomene muss in erster Linie die Preamble detektiert und anschlissend korrigiert werden. Als Hilfestellung wird eine Masterarbeit der EPFL verwendet, welche sich mit der Rekonstruktion der Technologie beschäftigte. Mit dessen Hilfe konnte auch in dieser Arbeit eine Übertragung erfolgreich empfangen und demoduliert werden.

Abstract Englisch
In this work, a "Software Defined Radio" and a LoRa transmitter are used to capture a data transmission to the LoRaWan and then demodulate it. The focus is on the temporal synchronisation between the transmitter and the receiver. Furthermore, there are frequency deviations from the perfect frequency band. This effect also leads to incorrect demodulation of the signal. To correct these phenomena, the preamble must first and foremost be detected and then corrected. A master's thesis from the EPFL, which dealt with the reconstruction of the technology, is used as an aid. With its help, a transmission could also be successfully received and demodulated in this work.

Ort, Datum Beromünster,17.12.2023

Inhaltsverzeichnis

Appendices

Einleitung

Aufgabenstellung

Aufgabe im Rahmen einer Industriearbeit ist die Signaldemodulation eines LoRa-Senders mittels zur Verfügung gestellter Hard- und Software. Die Aufgabenstellung ist unter Appendix A zu finden.

Motivation

Seit der Entwicklung von LoRa im Jahr 2009 ist das Wachstum kaum zu stoppen. Mit über 300 Millionen Geräten ist es eines der grössten LPWANs der Welt. “LoRa Technology Overview” (2023a) Die Firma Semtech besitzt das gesamte Patent und ist somit auch der einzige Hersteller von Sender- und Empfängerchips. Deshalb ist die Technologie nicht öffentlich zugänglich. Für den sicheren Umgang mit der Technologie ist jedoch das Verständnis der Technologie und deren Funktionsweise wichtig. Das Ziel dieser Arbeit ist es, die LoRa-Technologie in einem einfachen Kontext zu erklären und verständlich zu machen. Zu diesem Zweck wird eine Masterarbeit der EPFL als Vorlage und zur Nachbildung des Prozesses verwendet. Tapparel et al. (2020)

Aufbau

In einem ersten Teil wird die notwendige Theorie von der Technologie LoRa sowie des Netzprotokolls LoRaWAN und dessen Unterschiede erklärt. Eine kurze thematische Erläuterung zu der Entstehung der Technologie, der Firma Semtech und dessen Entwicklung in den letzten Jahren hilft zu verstehen warum wissenschaftliche Arbeiten zu diesem Thema überhaupt wichtig sind. Anschliessend wird tief in die Funktionsweise der Preamble und der eines Symbols eingegangen, um anschliessend das mathematische Konzept der Demodulation zu erläutern. Das Augenmerk liegt dabei bei der korrekten Synchronisation im Zeit- und Frequenzbereich. Dabei werden dessen Problem und Möglichkeiten der Synchronisation mittels der Preamble mathematisch erklärt und dessen Umsetzung in Python gezeigt. Des weiteren wird auf die Detektion einer Preamble aus einem Signal aufgegriffen. Mittels eines “Software Defined Radio” welches als Empfangsstation dient, wird versucht ein Signal einzufangen und ihm dessen Informationsgehalt zu entlocken. Mittels GNU Radio Companion werden diese Signale gespeichert und mit Python demoduliert.

1 Hardware

1.1 LimeSDR Mini

Der LimeSDR Mini ist ein "Software Defined Radio". Es handelt sich dabei um ein externes Eingabegerät, welches eine Kabelleseübertragung im Bereich von **35Hz bis 3.5GHz** ermöglicht. Für das Senden und das Empfangen wird jeweils eine eigene Antenne verwendet. Damit kann somit gleichzeitig Daten gesendet sowie empfangen werden.

1.1.1 Senden & Empfangen

Das LMS7002M übernimmt die Modulation und Demodulation von Basis- und Trägerbändern. Es handelt sich um ein MIMO-System mit insgesamt zwei Ein- und Ausgängen, das ein Frequenzband von 100 kHz bis 3,8 GHz abdeckt. Es ist jedoch jeweils nur ein Ein- und Ausgang aus dem Chip herausgeführt, wodurch das System ein SISO-System ist. Die maximale Ausgangsleistung ist auf $0dBm = 1mW$ begrenzt. "LMS7002M" (2023)

1.1.2 Antennen

Obwohl die Sendeeinheit über einen sehr grossen Frequenzbereich verfügt, ist dies bei der Antenne nicht der Fall. Daher ist es wichtig, die richtige Antenne für einen bestimmten Frequenzbereich zu installieren. In diesem Fall wurde lediglich eine Antenne mitgegeben, welches die Auswahl vereinfachte.

1.2 The Things Node

The Things Node ist ein Endgerät für Einsteiger in die LoRa-Welt. Es wird mit 3xAAA Batterien betrieben. Das Gerät verfügt über einen Temperatursensor, einen Luftfeuchtigkeitssensor, einen Helligkeitssensor und einen Bewegungssensor. Über einen Taster kann manuell eine Messung ausgelöst und die gesammelten Daten gesendet werden. Alternativ kann die Messung auch periodisch oder bei Bewegung ausgelöst werden. Das Ganze wird durch eine LED angezeigt. Das Gerät ist seit mehr als 5 Jahren auf dem Markt. "The Things NODE" (2023)

2 LoRa

LoRa ist ein Übertragungsprotokoll, welches vor allem für die Datenübertragung kleiner Datenmengen entwickelt wurde. Es handelt sich dabei um eine Punkt-zu-Punkt-Verbindung. Die Vorteile sind ein geringer Stromverbrauch und eine hohe Reichweite.

2.1 Geschichte

LoRa ist ein eingetragenes Produkt von Semtech. “LoRa Technology Overview” (2023b) Die Technologie wurde 2009 von zwei Freunden in Frankreich entwickelt, mit dem Ziel, eine grosse Reichweite bei energiearmer Datenübertragung zu erreichen. Der angedachte Anwendungsbereich war das Übertragen von Messwerten für Gas, Wasser und Strom. Um das Rad nicht neu erfinden zu müssen, wurde ein bereits bekanntes Verfahren zur Modulation verwendet, welches bis dahin nur beim Militär zur Aufklärung in der Luft und im Wasser eingesetzt wurde. Der Unterschied besteht darin, dass es bisher nur als Informationssignal und nicht als Hilfssignal verwendet wurde.

Im Jahr 2012 wurde die Idee von Semtech mitsamt den Entwicklern aufgekauft und fortan unter eigenem Namen vermarktet. Im Februar 2015 wurde die LoRa Alliance gegründet, um die Idee des LoRaWAN-Netzwerks zu standardisieren und voranzutreiben. Slats (2023)

Laut den neuesten Zahlen des Herstellers gibt es derzeit etwa 300 Millionen Endgeräte und 5,9 Millionen Gateways weltweit, die auf insgesamt 181 verschiedene Netzwerke aufgeteilt sind. Die meisten davon erfüllen einen industriellen Zweck, wie zum Beispiel das LoRaWAN des schweizerischen Netzwerkanbieters Swisscom. “LPN LoRaWAN-Das Schweizer IoT-Netzwerk” (2023)

In diesem Projekt wird jedoch das Netzwerk von “The Things Network” verwendet, welches als eines der grössten öffentlichen, kollaborativen und freien Netzwerke gilt.

2.2 Technologie

2.2.1 Anforderungen

Um eine energiearme und reichweitenstarke Übertragung zu ermöglichen, sind folgende Überlegungen zu beachten:

Die Signalleistung sollte möglichst gering und effizient genutzt werden, um die benötigte Energie zu minimieren.

Des Weiteren sollte die benötigte Sendezeit möglichst kurz sein und zur Erzielung einer hohen Reichweite sollte das Signal ein hohes Signal-Rausch-Verhältnis (SNR) aufweisen. Zusätzlich wird das minimal erforderliche SNR durch Codierung und Bandbreitennutzung reduziert.

2.2.2 Signalaufbau

2.2.2.1 Symbol

Als Symbol dient ein sogenannter “Compressed Hight Intensity Radar Pulse” (Chirp). Dieser zeigt in seiner Grundform eine stetige steigende Frequenz.

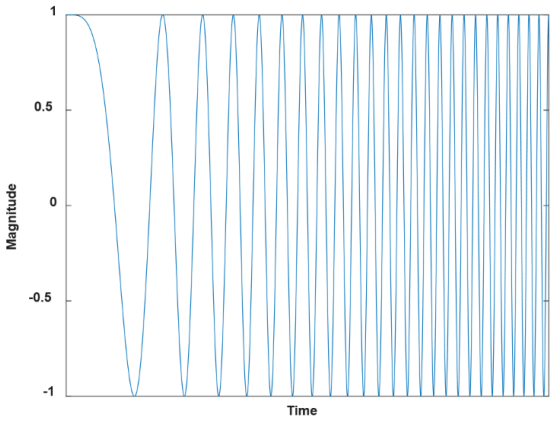

Figure 2.1: Einen Chirp im Zeitbereich

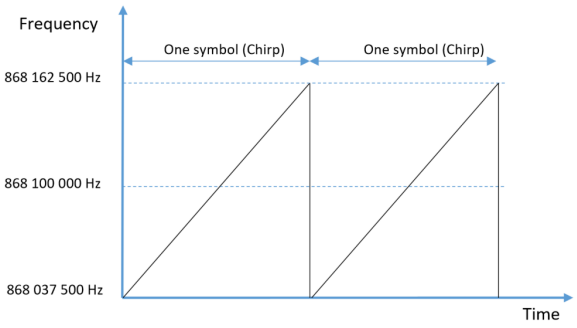

Figure 2.2: Einen Chirp im Frequenzbereich

In der Figure 2.1 ist ein Chirp im Zeitbereich ersichtlich. Dabei ändert sich lediglich die Frequenz mit der Zeit. Die Amplitude des Signals ist konstant.

In der Figure 2.2 ist die lineare Änderung der Frequenz gut ersichtlich. Am Ende des Chirps kommt es zu einer sprunghaften Änderung der Frequenz zurück zur Ausgangsposition. Es handelt sich somit um einen Sägezahn, welcher von $-\frac{fs}{2}$ bis $+\frac{fs}{2}$ auf dem Trägersignal verläuft. Montagny (2022)

Das Signal nutzt das gesamte verfügbare Spektrum. Es handelt sich jedoch lediglich um ein Hilfssignal, welches sowohl vom Militär als auch von Tieren bei der Jagd verwendet wird. Der grosse Vorteil des Signals liegt in der Übertragung aufgrund des Dopplereffekts und der Abhängigkeit von der Frequenzveränderung anstelle der absoluten Frequenz. “Chirp Spread Spectrum” (2023)

2.2.2.2 LoRa Paketstruktur

Um das Signal zu erkennen und die Zeit- und Frequenzverschiebung zu korrigieren, wird eine sogenannte Preamble als Start jeder Übertragung gesendet. Im Anschluss daran folgen die Daten mit einer vorgegebenen Länge. Die Preamble erfüllt mehrere Zwecke: Zum einen hilft sie beim Erkennen von Datenpaketen und beim Synchronisieren von Frequenz und Zeit. Zum anderen kann sie Informationen über die Paketlänge, die Coderate und die Checksumme enthalten.

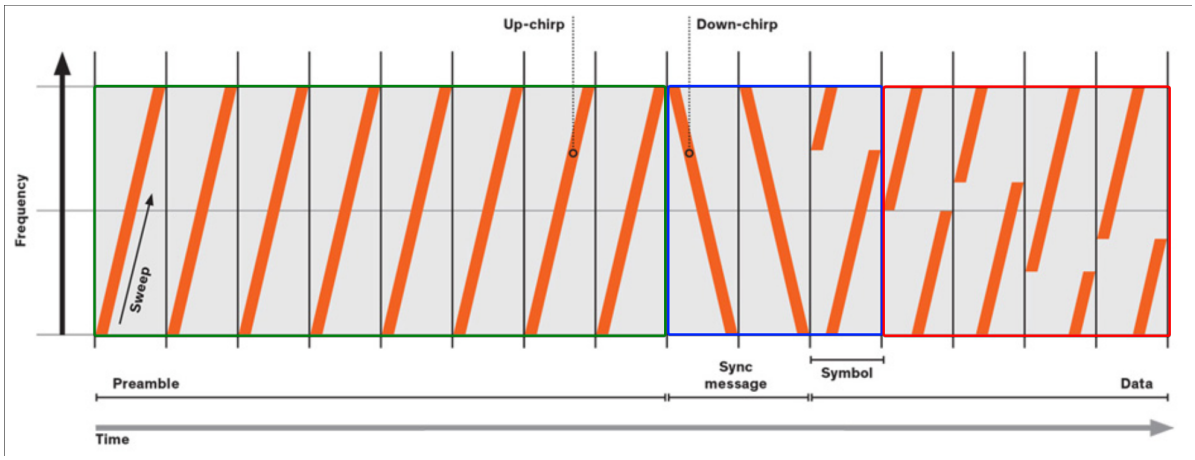

Figure 2.3: Eine Preamble im Frequenzbereich

Die ersten 8 Upchirps dienen dem Erkennen der Startsequenz und hilft bereits gewisse Zeitverschiebungen zu korrigieren. Typisch dafür ist eine Anzahl von 8 welche auch in Figure 7.1 erkennbar ist. Jedoch ist ein Bereich von $N_{pr} = \{6, \dots, 65535\}$ Nullsymbole als Start spezifiziert. Mehr Nullsymbole helfen das Signal auch bei noch schlechter SNR noch erfolgreich zu decodieren. Tapparel et al. (2020) Nach der Preamble folgen zwei und einen viertel Downchirps, welche helfen den Unterschied zwischen dem Zeit- und Frequenzoffset zu bestimmen. Anschliessend kann optional ein Header gefunden werden, in dem Informationen über das Datenpaket beinhaltet. Erst dann folgen die zu übertragenden Daten mit einer Checksumme (CRC). Die maximale Grösse der übertragbaren Daten ist auf 255 Bytes beschränkt.

2.3 Reglementierung

Wie bei jedem funktionierenden System müssen Regeln befolgt werden, die festlegen, welche Leistung und wie oft welches Frequenzband genutzt werden darf. Dadurch wird eine faire Nutzung für alle Teilnehmer gewährleistet.

2.3.1 Frequenzplan

Für das Senden und Empfangen von Daten über das LoRaWAN Netzwerk stehen in Europa folgende Frequenzbänder zur Verfügung:

Figure 2.4: Europäischer Frequenzplan für LoRaWAN

In Figure 2.4 “Frequency Plans” (2023) ist die Bestimmung der “The Things Network” über ihre Frequenzbänder. Diese Frequenzen wurden von TTN bestimmt und gelten auch nur speziell für ihr Netzwerk. “Frequency Plans” (2023) Da in diesem Projekt die Dekodierung eines solchen Signals im Vordergrund steht, sind leidlich die Uplink-Frequenzbänder relevant. Dabei wird sich zusätzlich auf die Frequenzbänder 1-8 beschränkt, da nur diese die CSS-Modulation nutzen. Des weiteren wird im europäischen Raum beinahe ausschliesslich eine Bandbreite von $125kHz$ verwendet.

2.4 Frequenzband

Das offizielle Frequenzband für den europäischen Raum wird vom “CEPT Electronic Communications Committee” verwaltet. “ERC Recommendation” (2023)

In der Figure 2.5 “Frequency Plans” (2023) ist genau dieser Frequenzplan für das europäische Spektrum abgebildet. In der Farbe grün hinterlegt ist das entsprechende Frequenzband. Dabei wichtig anzumerken, dass sich bei LoRaWAN um eine “Non-FHSS” Technologie handelt. Es kommt somit während der laufenden Übertragung zu keinem wechseln des Frequenzbands. Dabei wichtig ist die effektive Strahlungsleistung von lediglich $25mW$ und die maximale Nutzungszeit von rund 0.1%.

Frequency Band		Power / Magnetic Field	Spectrum access and mitigation requirements	Modulation / maximum occupied bandwidth	ECC/ERC Deliverable	Notes
f3	169.4875-169.5875 MHz	10 mW e.r.p.	≤ 0.001% duty cycle except for 00:00 h to 06:00 h local time where the duty cycle limit is ≤ 0.1%	Not specified	ECC/DEC/(05)02	The frequency band is also identified in Annex 10
f4	169.5875-169.8125 MHz	10 mW e.r.p.	≤ 0.1% duty cycle	Not specified	ECC/DEC/(05)02	
g1	433.05-434.79 MHz	10 mW e.r.p.	≤ 10% duty cycle	Not specified		
g2	433.05-434.79 MHz	1 mW e.r.p.	No requirement (note 3)	Not specified		Power density limited to -13 dBm/10 kHz for wideband modulation with a bandwidth greater than 250 kHz
g3	434.04-434.79 MHz	10 mW e.r.p.	No requirement (note 3)	≤ 25 kHz		
h0	862-863 MHz	25 mW e.r.p.	≤ 0.1% duty cycle	≤ 350 kHz		
h1.0	863-870 MHz (note 2)	25 mW e.r.p.	≤ 0.1% duty cycle (note 1)	≤ 100 kHz for 47 or more hop channels		FHSS. Parts of the frequency band are also identified in Annexes 2, 3, 10 and 11
h1.1	865-868 MHz	25 mW e.r.p.	≤ 1% duty cycle (note 1)	≤ 50 kHz for 58 or more hop channels		FHSS. The frequency band is also identified in Annexes 2, 3 and 11
h1.2	863-870 MHz (note 2)	25 mW e.r.p. -4.5 dBm/100 kHz	≤ 0.1% duty cycle or LBT+AFA	Not specified		Non-FHSS. Parts of the frequency band are also identified in Annexes 2, 3, 10 and 11
h1.3	863-865 MHz	25 mW e.r.p.	≤ 0.1% duty cycle or LBT+AFA	Not specified		The frequency band is also identified in Annexes 3 and 10
h1.4	865-868 MHz	25 mW e.r.p.	≤ 1% duty cycle or LBT +AFA	Not specified		The frequency band is also identified in Annexes 2, 3 and 11
h1.5	868-868.6 MHz	25 mW e.r.p.	≤ 1% duty cycle or LBT +AFA	Not specified		
h1.6	868.7-869.2 MHz	25 mW e.r.p.	≤ 0.1% duty cycle or LBT+AFA	Not specified		
h1.7	869.4-869.65 MHz	500 mW e.r.p.	≤ 10% duty cycle or LBT+AFA	Not specified		
h1.8	869.7-870 MHz	5 mW e.r.p.	No requirement (note 3)	Not specified		
h1.9	869.7-870 MHz	25 mW e.r.p.	≤ 1% duty cycle or LBT +AFA	Not specified		

Figure 2.5: Europäischer Frequenzplan für LoRaWAN

Um diese maximale Nutzungszeit nicht zu überschreiten wird hier eine Abschätzung aufgestellt. Bei einem Spreadingfaktor von $SF = 7$ und einer Bandbreite von $BW = 125kHz$ benötigt die Übertragung von einem Symbol:

$$t_s = \frac{2^{SF}}{BW} = \frac{2^7}{125kHz} = 1.024ms$$

Bei einer Datengrösse von 39 Datenbits und 8 Preambelbits mit einer Coderate von $CR = 4/5$ ergibt sich folgende Übertragungszeit:

$$t_p = \text{payload} \cdot \frac{t_s}{CR} = (39Bits + 8Bits) \cdot \frac{1.024ms}{\frac{4}{5}} = 60.61ms$$

Dies ermöglicht in einem Tag etwa $x = \frac{t_d * DC}{t_p} = \frac{864 \cdot 10^5 ms \cdot 0.1}{60.61ms} \approx 1425$ Übertragungen. Gleichmässig auf den ganzen Tag verteilt ist somit möglich pro $\frac{1425}{24 \cdot 60} \approx 1min$ ein Paket zu versenden.

3 LoRaWAN / LPWAN

"LoRa ist eine Form einer sogenannten LPWAN-Funktechnologie. LPWAN ist die Abkürzung für den englischen Begriff des Low Power Wide Area Network, der sich grob als etwa Niedrigenergieweitverkehrsnetz übersetzen lässt." Dultz (2023) . Dabei steht LoRa für die kurzform von "Long Range" welche als speziellen Vorteil der Technologie gilt. Die typische Reichweite ist stark von der Umgebung abhängig und liegt oftmals in einem Bereich von etwa 2 bis 17 Kilometern. Der momentane Weltrekord wird von einem Wetterballon in Spanien gehalten. Trots der Reichweite von über 766km konnte ein Paket mit einer $SNR = -14.25dB$ erfolgreich empfangen und übersetzt werden. "LoRaWAN Distance Record" (2023)

3.1 Aufbau

Während das LoRaWAN eine ganze Netzwerkstruktur bietet, ist LoRa nur eine kleiner Teil dieses Pfades.

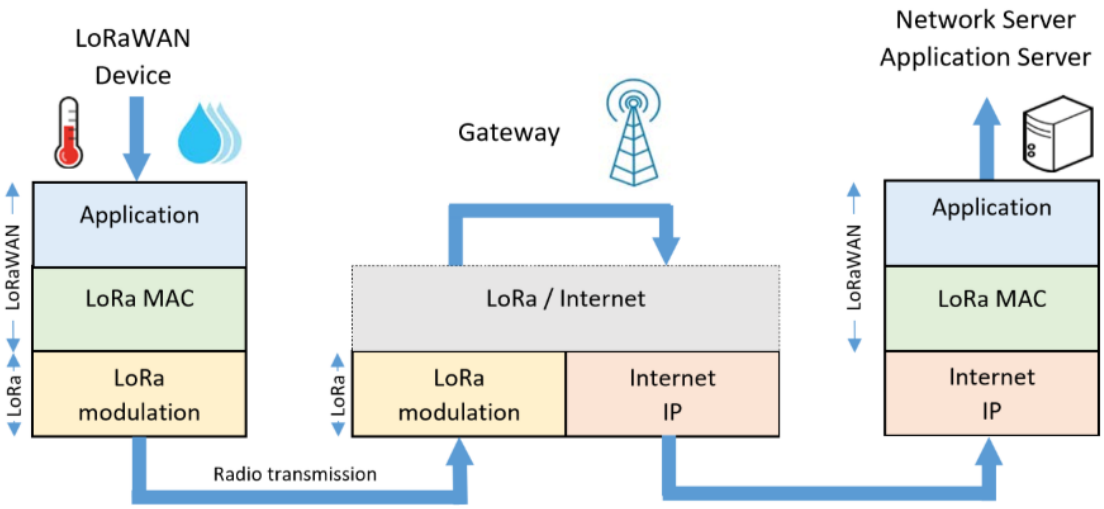

Figure 3.1: Aufbau einer LoRaWAN Verbindung

Grundlegend kann das LoRaWAN in drei Gruppen aufgeteilt werden:

- **Sender:** Das Sendegerät fungiert als Informationsquelle. Es erfasst die Daten, formatiert sie richtig und sendet sie mit Hilfe des LoRaWAN-Protokolls über LoRa in die Aussenwelt.
- **Endpunkte:** Der Endpunkt oder auch Gateway genannt, fungiert als Schnittstelle zwischen dem Sender und dem effektiven Empfänger. Das LoRa-Signal wird zwar vom Gateway über eine Empfangsstation aus der Luft empfangen und demoduliert, jedoch ist das Gateway nicht der Empfänger der Daten. Stattdessen lädt es die Daten auf einen Server hoch. Der gesamte Verkehr ist verschlüsselt und für das Gateway nicht einsehbar. Gateways benötigen daher einen Internetanschluss und sind aufgrund des Stromverbrauchs oft in der Nähe von Gebäuden zu finden. Durch den Aufbau eines Netzwerks von Gateways kann ein Netzwerk gebildet werden.
- **Empfänger:** Der Empfänger hat die Möglichkeit, auf einem Server seine bereits demodulierten LoRa-Signale zu entschlüsseln. Als Nutzer wird somit nur ein Sender benötigt und die Daten sind automatisch aus der Ferne abrufbar. Diese Daten können dann persistent gespeichert und visuell aufbereitet werden.

Um eine reibungslose Kommunikation und Zuweisung von Datenpaketen zu ihren Eigentümern zu gewährleisten, werden spezielle Daten für das LoRaWAN mitgesendet. Hierfür benötigt jedes Gerät eine einzigartige Adresse (MAC-Adresse).

3.2 The Things Network

Das TTN ist ein Zusammenschluss von Privatpersonen und Firmen, die ein internationales Netzwerk an Endpunkten aufbauen, um das Signal von LoRa-Geräten aufzunehmen und die Daten dem Eigentümer des Geräts im Internet zur Verfügung zu stellen. Die Nutzung des Systems ist kostenlos und die Daten sind bis zum Ende verschlüsselt. Allerdings gilt dies nur für die aktiv gesendeten Daten und nicht für die Verbindungsdaten. Somit können Fakten jeder Verbindung der Vergangenheit analysiert werden. Jede Person kann mit der passenden Hard- und Software eigene Endpunkte und Geräte betreiben. Derzeit kooperieren ca. 200k Personen in 153 verschiedenen Ländern. “The Things Network” (2023)

3.2.1 Schweiz

Die Schweiz ist mit insgesamt 639 Gateways stark in der Community vertreten. In Horw allein stellen zwei Contributoren insgesamt 17 Gateways zur öffentlichen Nutzung zur Verfügung. Die Position dieser Gateways ist auf einer Karte auf der Website von TTN für alle einsehbar. Die HSLU T&A besitzt ebenfalls ein Outdoor Gateway, welches regelmässig von verschiedenen Geräten genutzt wird. Von einem Gateway können alle eingegangenen Verbindungen mit ihren Parametern angezeigt werden. “Switzerland” (2023)

Die grösste Distanz eines Datenpaketes der HSLU stammt vom Sempachersee und legte dabei eine Strecke von 16.77 Kilometer zurück. Dabei wurde der grösste Spreading Faktor von 12 benutzt. Dies und weitere Informationen sind in Figure 3.2 Montagny (2022) ersichtlich.

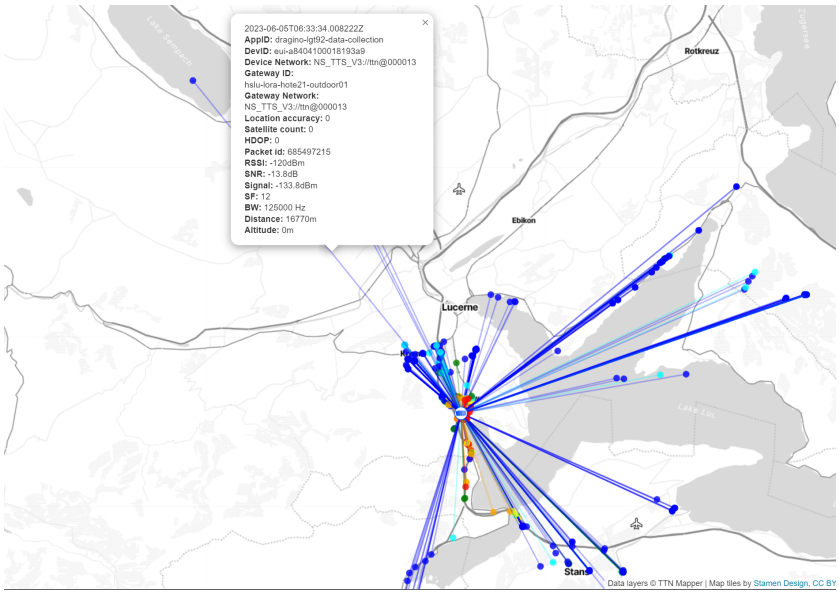

Figure 3.2: Reichweitenrekord der HSLU

4 Signalrekonstruktion

Um während des Prozesses mit generierten und perfekten Signalen ohne Verzerrungen durch den Kanal und Rauschen testen zu können, wurde bereits in einem frühen Schritt eine Preamble synthetisch nachgebaut. Um eine Preamble zu erzeugen, muss jedoch zunächst eine Möglichkeit geschaffen werden, ein Symbol zu erzeugen.

4.1 Synthetische Symbole

Um ein Symbol im Frequenzbereich zu erzeugen welches auch die korrekte Anzahl an Samples hat, muss bestimmt werden welchen **Spreading Faktor** SF verwendet werden sollte. Die **Anzahl der Symbole** $S = 2^{SF}$ ist auch gleich die Anzahl der Samples n. Das **Symbol** s welches erzeugt wird, muss in der gesamten Anzahl der Symbole S enthalten sein. Tapparel et al. (2020)

$$x_s[n] = e^{\jmath 2\pi(\frac{n^2}{2N}+(\frac{s}{N}-\frac{1}{2})n)n\in S}$$

Jedoch sorgt diese mathematische Beschreibung lediglich um eine steigt steigende Frequenz. Dabei fehlt der Frequenzsprung welcher bei $+\frac{BW}{2}$ startet und bei $-\frac{BW}{2}$ endet. Dabei steht BW für die definierte Bandbreite, welche in Europa auf $125kHz$ gesetzt ist. “Frequency Plans” (2023)

```
import numpy as np

def GetSymbol(spreading_factor:int, symbol:int)->complex:
    N = 2**spreading_factor                      # bits/symbol
    n = np.arange(N, dtype=np.float32)           # number of symbols

    idx_fold = (N - S)
    symbol = np.zeros(N, dtype=np.complex64)     # init symbolarray
    n1, n2 = n[0:idx_fold], n[idx_fold:]         # split at the fold
    symbol_f[0:idx_fold] = np.exp(2*np.pi*1j*(n1**2/(2*N) + S*n1/N - n1/2))
    symbol_f[idx_fold:] = np.exp(2*np.pi*1j*(n2**2/(2*N) + S*n2/N - 3*n2/2))

    return symbol_f
```

In diesem Codeabschnitt können einzelne Symbole generiert werden. Dabei muss der Spreadingfaktor angegeben werden, da dieser entscheidend für die Anzahl der Samples im Symbol ist. Ein höherer Spreadingfaktor führt zu mehr Samples im Symbol. Mehr Samples bedeuten jedoch auch mehr Informationen, da es mehr Punkte geben kann, an denen es zum Frequenzsprung kommen kann.

Um diesen Sprung zu erstellen, wird zuerst berechnet, wo dieser Sprung stattfindet und in `idx_fold` gespeichert. Danach wird das Array vom Start bis `idx_fold` mit der steigender Frequenz gefüllt und dem korrekten Offset versehen. Anschließend wird dasselbe mit dem Array von `idx_fold` bis zum Ende des Arrays durchgeführt. Der Rückgabewert ist dann das Array mit dem komplexen Signal.

4.2 Synthetische Preamble

Da es nun möglich ist mittels `GetSymbol()` möglich ist jedes mögliche Symbol zu erstellen, kann nun eine Preamble zusammengesteckt werden. Der Aufbau der Preamble wird bereits in Section 2.2.2.2 erläutert.

```
import numpy as np

def GetPreamble(SF, net_id=64):
    N = 2**SF
    n = np.arange(N, dtype=np.float32)

    upchirps = np.tile(GetUpchirp(SF), 8)
    sync = np.tile(GetSymbol(SF, net_id), 2)
    downchirps = np.tile(GetDownchirp(SF), 2)
    quarter_downchirp = downchirp[0:int(N/4)]

    preamble = np.concatenate([upchirps, sync, downchirps, quarter_downchirp])
    return preamble
```

Im Code ist festgehalten, wie die Preamble generiert wird. Dazu werden aus den einzelnen Blöcken sogenannte Tiles erstellt, in denen die Symbole in der korrekten Anzahl enthalten sind. Anschließend werden sie mit der Funktion `np.concatenate()` nacheinander zusammengefügt. Schließlich wird die Preamble als Antwort zurückgegeben.

5 Signaldetektion

5.1 Signalaufnahme

Bei der Aufnahme eines LORA-Signals müssen mehrere Parameter des Lime-SDR korrekt angepasst werden. Zunächst ist die korrekte Bandbreite und Frequenz entscheidend. Diese Grenzen können mit digitalen und analogen Filtern gesetzt werden. Auch die Samplingrate muss beachtet werden, um das Nyquist-Theorem nicht zu verletzen.

Insgesamt werden für LoRaWan acht Übertragungskanäle mit der Mittenfrequenz $f_0 = 867,8MHz$ verwendet, wobei ein Kanal eine effektive Bandbreite von $B = 125kHz$ hat. Zwischen den Frequenzbändern gibt es einen Abstand von $75kHz$. Jeder Kanal hat eine effektive Bandbreite von $B = 125kHz$. Daher benötigt jeder Kanal eine Bandbreite von $B = 200kHz$. Die Mittelfrequenz beträgt $f_0 = 867,8MHz$. Die absolute minimale benötigte Bandbreite zur Abtastung aller Kanäle beträgt somit:

$f_k = \text{Anzahl Kanäle} \cdot (f_s + \text{Abstand der Kanälen}) = 8 \cdot (125kHz + 75kHz) - 75kHz = 1.525MHz$

Jedoch muss die Bandbreite für das spätere Downsampling ganzzahlig teilbar durch die effektive Bandbreite sein, und sollte gewisse Reserven beinhalten:

$$k = \frac{f_k}{f_s} = \frac{1525kHz}{125kHz} = 12.2 \rightarrow 16$$

$$B_{\text{effektiv}} = k \cdot f_s = 16 \cdot 200kHz = 3.2MHz$$

Somit wäre der effektive abgetastete Frequenzbereich bei:

$$f = f_0 \pm \frac{B}{2} = 867.8MHz \pm \frac{3.2MHz}{2} \rightarrow 866.2MHz \ - \ 869.4MHz$$

Um anschliessend bei der Detektion des Signals die korrekte Anzahl an Samples pro Symbol zu haben, muss dies ebenfalls korrekt eingestellt werden. Bei einem Spreadingfaktor von $SF = 7$ sind die benötigte Anzahl der Samples pro Symbol $N = 2^{SF} = 2^7 = 128$ Samples.

Da es beim Downsampling effektiv um eine Reduktion der Samples um den Faktor k kommt, muss dies bei der Aufnahme des Signals und derer Samplingrate beachtet werden. Somit muss das LimeSDR Mini eine Abtastrate f_{SDR} von $f_{SDR} = k \cdot f_s = 16 \cdot 125kHz = 2MHz$ aufweisen.

5.2 Downsampling

Da jedes LoRaWan-Endgerät eine der genannten Frequenzen zur Übertragung nutzen kann, müssen auch alle diese Frequenzen abgetastet werden. Um dies mit einem einzigen Gerät realisieren zu können, wird hier ein unerwünschter Nebeneffekt des Downsamplings genutzt. Wird bei der Unterabtastung kein geeignetes Aliasing-Filter verwendet und damit das Nyquist-Shannon-Abtasttheorem von $f_s > 2 \cdot f_{SDR}$ verletzt, so überlagern sich die einzelnen Frequenzbänder. Da das Gerät jeweils nur eine Frequenz nutzt und die Signalstärke des eigenen Endgerätes deutlich höher ist als die der Alternativen in der Umgebung, können Interferenzen nahezu ausgeschlossen werden. Die vollständige Überlagerung aller 8 Frequenzbänder erfolgt genau bei $f_s = f_{BW}$

6 Offsetkorrektur

6.1 Problemstellung

Da es sich hier um eine drahtlose Übertragung handelt, muss eine Synchronisation stattfinden. Diese wird durch zwei verschiedene technische Probleme ausgelöst. Im ersten Fall existiert keine gemeinsame Clockleitung. Diese zeitliche Verzögerung wird in der Fachliteratur auch als 'Sample Time Offset' bezeichnet. Im zweiten Fall hat der Sender der Nachricht nicht die gleiche Übertragungsfrequenz. Dies ist auf die Fertigung von Schwingquarzen und diverser Hardware zurückzuführen. Da die Trägerfrequenz bei einer sehr hohen Frequenz von etwa $868MHz$ schwingt, entsteht eine Abweichung vom perfekten und spezifizierten Frequenzband. Diese Abweichung muss korrigiert werden. Der Frequenzfehler wird auch als 'Carrier Frequency Offset' bezeichnet.

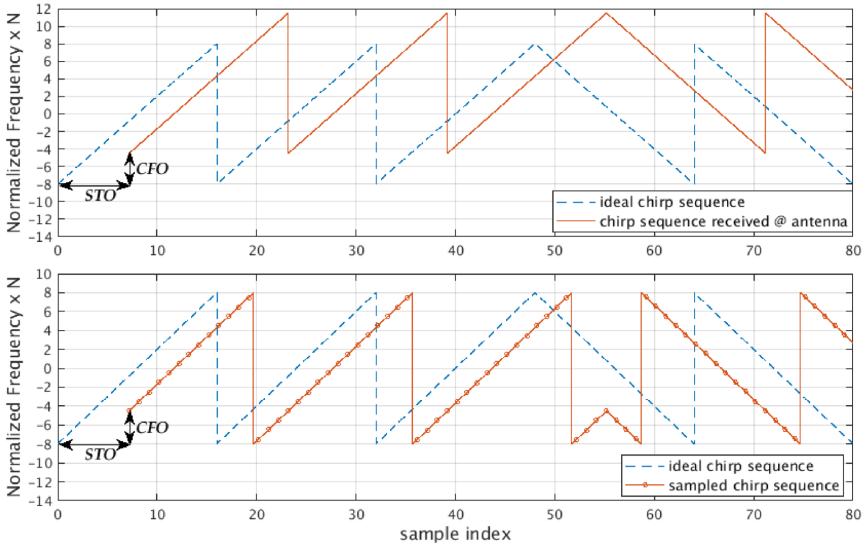

Figure 6.1: Darstellung der beiden Offsetarten und dessen Auswirkung.

Wie in Figure 6.1 zu sehen ist, ist der Unterschied zwischen CFO und STO nach der Diskretisierung nicht mehr erkennbar. Um dies dennoch zu erkennen, verfügt die Preamble über $2\frac{1}{4}$ Synchronisationssymbole. Diese sind in Figure 7.1 dargestellt. Es handelt sich dabei um zwei ganze Down-Chirps, gefolgt von einem viertel Down-Chirp. Im oberen Teil der Grafik repräsentiert Orange das Eingangssignal und Blau das Referenzsignal des Empfängers im kontinuierlichen Zeitbereich. Im unteren Teil der Grafik wurden die beiden Signale von der Antenne des Empfängers erfasst und diskretisiert.

6.2 Demodulation

Zunächst muss erkannt werden, ob eine Preamble empfangen wird. Hierfür wird ein Algorithmus zur Demodulation des Signals benötigt. Die Nullsymbole am Anfang jeder Preamble dienen dazu. Eine Möglichkeit ist die Korrelation mit einem Nullsymbol im Frequenzbereich. Allerdings ist dies nicht effizient, da eine solche Korrelation sehr rechenintensiv ist.

Eine deutlich effizientere Variante zeigt eine Arbeit der EPFL. Tapparel et al. (2020) Dabei wird das Eingangsignal $y = [y[0] \dots y[N-1]]$ in eine Blockgrösse von 2^{SF} aufgeteilt un anschliessend mit dem **komplex konjugierten Referenzsymbol** $x_0 = [y[0] \dots y[N-1]]$ punktweise multipliziert. Dafür wird jeweils das Nullsymbol als Referenzsymbol verwendet.

$$Y = DFT(y \odot x_0^*)$$

Das maximale Argument dieser Liste zeigt den Frequenzsprung im Eingangsignal an. Dieser Index $\hat{s}$ entspricht der demodulierten Zahl dieses Symbols.

$$\hat{s} = \arg\max_{k \in S}(|Y[k]|)$$

6.3 Preamble Detektion

Mithilfe der Demodulation kann lediglich die Preamble zuverlässig detektiert werden. Da zu diesem Zeitpunkt keine zeitliche Synchronisation stattgefunden hat, stimmen die Startpunkte eines Symbols zwischen den beiden Teilnehmern nicht überein. Diese zeitliche Verschiebung eines Nullsymbols führt jedoch faktisch zur Generierung von höherwertigen Symbolen. Da mehrere Nullsymbole hintereinander folgen und die Verschiebung über diese Zeit konstant bleibt, werden somit auch die gleichen Symbole demoduliert. Wenn dies mehrmals hintereinander geschieht, wurde erfolgreich eine Preamble detektiert.

Hier ist anschliessend der Codeabschnitt beschrieben, welcher diese Detektion übernimmt.

```
def preamble_detection(index:complex,symbols:complex) -> int:

    index_delta,symbols_delta = preamble_to_delta(index,symbols)
    position = 0
    number_of_detected_symbols = 0

    # As long as there is signal
    while (position + 1) < len(index_delta):

        # Index in bound of +-1
        if(index_delta[position+1] in range(index_delta
            [position-number_of_detected_symbols]-1,index_delta
            [position-number_of_detected_symbols]+1)):

            number_of_detected_symbols += 1
            position += 1

        # Index out of bound of +-1
        else:
            number_of_detected_symbols = 0
            position += 1

        # positiv endcase
        if number_of_detected_symbols >= (PREAMBLE_LENGTH - 1):
            return (position - PREAMBLE_LENGTH + 1 )

    return -1  # no signal found
```

In diesem Codeabschnitt wird die Funktion gezeigt, die zum Erkennen der Preamble geschrieben wurde. Es wird eine Schleife verwendet, in der geprüft wird, ob das aktuelle

Symbol und das folgende Symbol maximal eine Abweichung von eins aufweisen. Wenn dies der Fall ist, wird dasselbe mit dem darauf folgenden Symbol in Bezug auf das erste Symbol geprüft. Wenn dies nicht der Fall ist, wird das nächste Symbol zum neuen ersten Symbol und der Vorgang wird wiederholt. Diese Schleife endet auf eine von zwei Arten: Im positiven Fall wurden acht aufeinander folgende Symbole gefunden und die Position des ersten Symbols wird zurückgegeben. Im negativen Fall erreicht die Schleife das Ende des Arrays und für die Position wird -1 zurückgegeben.

6.4 Offsetkorrektur

Um das Signal zu demodulieren und eine Offsetkorrektur zu implementieren, muss nach dem Empfangen des Signals zwischen diskreten Signalen unterschieden werden. Hierbei sind CFO_{int} und STO_{int} lediglich ganzzahlige Sample-Korrekturen, die nur eine begrenzte Genauigkeit zulassen.

6.4.1 Referenzsignal

Wie bereits im Abschnitt über die Preamble erwähnt besteht das Signal zu Beginn aus mehreren **Up-Chirps**. Diese können mathematisch im Frequenzbereich folgendermassen beschrieben werden:

$$r[k] = A \cdot e^{\jmath 2\pi(\frac{k^2}{2N} + k(\frac{(N-STO_{int})}{N} + \frac{1}{2}))} \cdot e^{\frac{\jmath 2\pi k CFO_{int}}{BW} + \jmath\phi_0}$$

Dabei beschreibt A die Amplitude des Signals und $N = 2^{FS}$ die Anzahl der Samples pro Symbol. Hingegen ein **Down-Chirp** Signal welches doppelt in der Preamble hinter den Netzwerk-Symbolen sich befindet, ändert sich lediglich die Richtung der Frequenz. Dies ist mit einer Invertierung des Exponentialtherms realisiert.

$$r[k] = A \cdot e^{-\jmath 2\pi(\frac{k^2}{2N} + k(\frac{(N-STO_{int})}{N} + \frac{1}{2}))} \cdot e^{\frac{\jmath 2\pi k CFO_{int}}{BW} + \jmath\phi_0}$$

6.4.2 Erweiterte Demodulation

Um die erforderlichen Informationen aus den Up- und Down-Chirps zu extrahieren, muss eine Anpassung vorgenommen werden. Zunächst ist zu beachten, dass das konjugiert komplexe Ergebnis eines Up-Chirps genau dem Down-Chirp entspricht. Für die Demodulation der Up-Chirps und der restlichen Informationssignale bleibt der beschriebene Vorgang bestehen.Es wird mit dem komplex konjugierten Nullsymbol $B[k]^*$ multipliziert.

$$r[k] x B[k]^* = A \cdot e^{\jmath 2\pi k(\frac{CFO_{int}}{BW} + \frac{(N-STO_{int})}{N})} x e^{j\phi_0}$$

Jedoch um die zwei Down-Chirps zu demodulieren wird reguläre Nullsymbol $B[k]$ verwendet.

$$r[k]xB[k] = A \cdot e^{j2\pi k(\frac{CFO_{int}}{BW} + \frac{(N-STO_{int})}{N})} x e^{j\phi_0}$$

Durch diese Demodulieren mit der verbundenen DFT können die normierten Frequenzen f_{up} und f_{down} extrahiert werden. Diese sind auch in den oberen Beschreibungen der Chirps sichtbar.

$$f_{up} = \frac{CFO_{int}}{BW} + \frac{(N - STO_{int})}{N}$$

$$f_{down} = \frac{CFO_{int}}{BW} - \frac{(N - STO_{int})}{N}$$

6.5 Korrektur CFO

Der Carrier Frequency Offset ist ein Phänomen, das durch nicht perfekte Oszillatoren verursacht wird. Dabei unterscheidet sich die Trägerfrequenz des Senders von der Trägerfrequenz des Empfängers. Der Offset kann mithilfe der normierten Frequenzen eines Up- und Down-Chirps berechnet werden.

$$f_{up} + f_{down} = \frac{CFO_{int}}{BW} + \frac{(N - STO_{int})}{N} + \frac{CFO_{int}}{BW} - \frac{(N - STO_{int})}{N}$$

$$f_{up} + f_{down} = 2 \cdot \frac{CFO_{int}}{BW}$$

$$CFO_{int} = \frac{f_{up} + f_{down}}{2} \cdot BW$$

Mit dem folgenden Code konnte genau dieser “Center Frequency Offset” berechnet werden.

```
import numpy as np

DOWNCHIRP_POSITION = 7  # Last downchirp
UPCHIRP_POSITION = 10  # First upchirp
STEP_SIZE = 128
NORMALIZED_OFFSET = -0.5
BANDWIDTH_HZ = 125_000

# Get the normalized frequencies
frequency_upchirp = index_delta[start_position_preamble + UPCHIRP_POSITION]
frequency_downchirp = index_delta[start_position_preamble + DOWNCHIRP_POSITION]
```

```
frequency_upchirp_norm = frequency_upchirp/STEP_SIZE + NORMALIZED_OFFSET
frequency_downchirp_norm = frequency_downchirp/STEP_SIZE + NORMALIZED_OFFSET

sums = round(frequency_upchirp_norm + frequency_downchirp_norm)

# Center Frequency Offset calculation
if(frequency_upchirp_norm >= -frequency_downchirp_norm):
    cfo_int = sums/2*BANDWIDTH_HZ
else:
    cfo_int = sums/2*BANDWIDTH_HZ + BANDWIDTH_HZ/2

# Center Frequency Offset correction
shift_int = np.exp(2j*np.pi*np.arange(len(signal_t))*-cfo_int/BANDWIDTH_HZ)
signal_cfo_int_corrected = np.multiply(signal_t,shift_int)
```

Der vorliegende Code enthält den Abschnitt, der sich mit der Korrektur des CFOs befasst. Zunächst werden die korrekten demodulierten Werte des letzten Downchirps und des ersten Upchirps der Preamble normalisiert. Anschliessend wird die Summe berechnet und der Offset bestimmt. Schliesslich wird dieser bestimmte Offset durch einen Shift korrigiert. Die Funktion gibt ein CFO-korrigiertes Signal zurück.

6.6 Korrektur STO

Die Sample Time Offset ist eine zeitliche Verschiebung des Signals zwischen Sender und Empfänger. Diese kann ebenfalls mit den normierten Frequenzen von einem Up- und Down-Chirp berechnet werden.

$$f_{up} - f_{down} = \frac{CFO_{int}}{BW} + \frac{(N - STO_{int})}{N} - \frac{CFO_{int}}{BW} - \frac{(N - STO_{int})}{N}$$

$$f_{up} + f_{down} = 2 \cdot \frac{N + STO_{int}}{N}$$

$$STO_{int} = \left(\frac{f_{up} + f_{down}}{2} \cdot N \right) - 1$$

Der folgende Codeabschnitt trägt die Aufgabe das Signal wie oben mathematisch beschrieben, “Sample Time Offset” frei zu bekommen.

```
import numpy as np

DOWN_CHIRP_POSITION = 7  # Last position
UP_CHIRP_POSITION = 10  # First position
STEP_SIZE = 128
NORMALIZED_OFFSET = -0.5

# Get the normalized frequencies
frequency_upchirp = index_delta[start_position_preamble + UP_CHIRP_POSITION]
frequency_downchirp = index_delta[start_position_preamble + DOWN_CHIRP_POSITION]

frequency_upchirp_norm = frequency_upchirp / STEP_SIZE + NORMALIZED_OFFSET
frequency_downchirp_norm = frequency_downchirp / STEP_SIZE + NORMALIZED_OFFSET

difference = round(frequency_upchirp_norm - frequency_downchirp_norm)

# STO correction
if(frequency_upchirp_norm >= frequency_downchirp_norm):
    sample_time_offset_int = difference / 2 * STEP_SIZE
else:
    sample_time_offset_int = difference / 2 * STEP_SIZE + STEP_SIZE/2

# Cut off to the start of the preamble from signal_t
length = np.zeros(sample_time_offset_int,dtype=complex)
signal_sto_int_corrected = np.concatenate((length,signal_t),dtype=complex)
```

In diesem Code werden, wie bereits im vorherigen Abschnitt beschrieben, die normalisierten Frequenzen des Up- und Downchirps berechnet. Allerdings wird hierbei die Differenz der beiden verwendet. Anschliessend wird mithilfe des berechneten Offsets die korrekte Anzahl an Samples vor dem Signal angehängt. Dadurch befindet sich das Ausgangssignal zeitlich korrekt im Array.

7 Ergebnisse

In diesem Teil werden die erreichten Ergebnisse aufgeschlüsselt und erläutert. Dabei steht vor allem das Ziel der Arbeit, dem Demodulieren eines LoRa-Signals, im Vordergrund.

7.1 Theoretische Umgebung

Um sicherzustellen, dass die mathematischen Zusammenhänge korrekt verstanden und umgesetzt werden, wird zu Beginn kein reales Signal verwendet. Hierfür wurde eine Preamble möglichst realistisch nachgebildet.

In der Figure 7.1 ist die nachgebaute Preamble zu sehen. Sie besteht aus acht Up-Chirps, zwei Sync-Words mit der Network-ID 64 und am Ende zwei $2\frac{1}{4}$ Down-Chirps. Da für die anschliessende Demodulation ohnehin eine Möglichkeit zur Erstellung von Symbolen geschaffen werden musste, war dies keine grosse zusätzliche Arbeit. Um jedoch die CFO- und STO-Korrekturen testen zu können, muss dieser Offset zuerst erzeugt werden. Somit ist es später einfacher zu überprüfen, ob die einzelnen Bestandteile korrekt erkannt wurden.

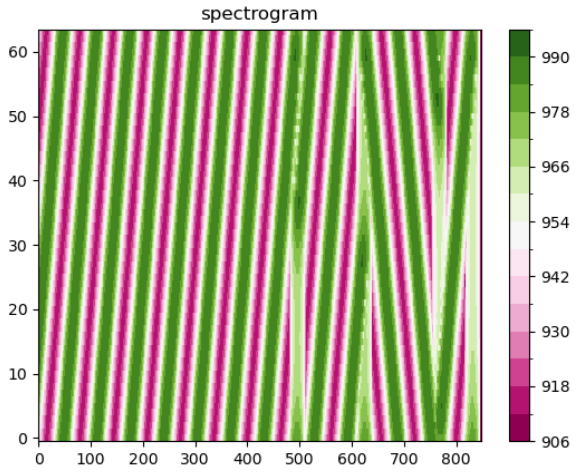

Figure 7.1: Synthetische Preamble

7.2 Versuchsaufbau

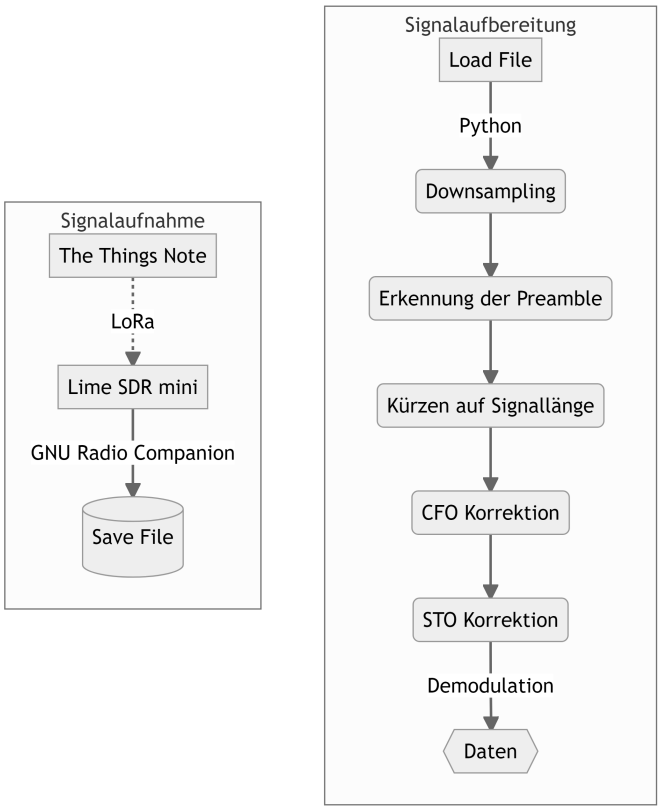

Figure 7.2: Ablaufdiagramm der Demodulation

In Figure 7.2 wird zunächst das Signal aufgenommen, um es in einem zweiten Schritt aufzubereiten. Das Signal wird als Datei übergeben, was auch der Vorgehensweise des Codes entspricht. Aus Komplexitätsgründen wurden an dieser Stelle bewusst Punkte vernachlässigt.

7.3 Reale Signale

Um dies jedoch unter realen Bedingungen mit der Verzerrung durch den Kanal und weiteren Anomalien zu testen, wurde ein solches Signal mittels SDR empfangen und abgespeichert. In einem ersten Schritt wird das Signal ohne jegliche Korrekturen demoduliert, um es später als Referenz nutzen zu können.

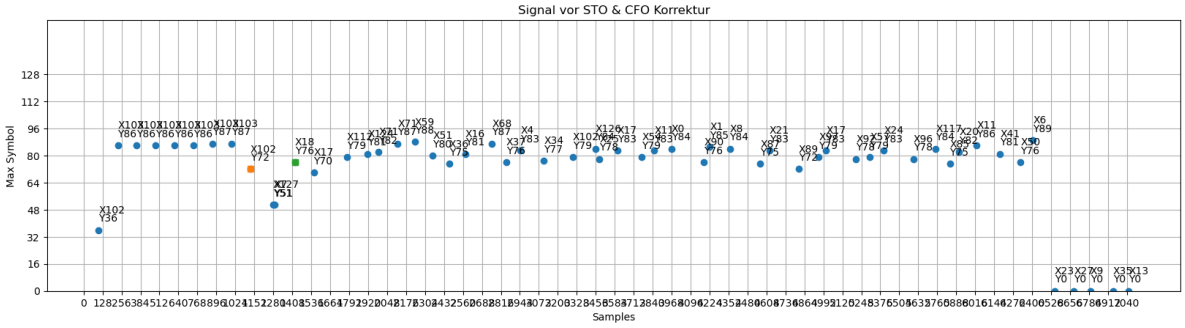

Figure 7.3: Reales Signal ohne CFO und STO Korrekturen

In Figure 7.3 ist die korrekte Startkennung bereits sichtbar. Der orange Punkt markiert hier das letzte Up-Chirp-Symbol, welches als Synchronisationspunkt für die Berechnung von CFO & STO verwendet wird. Der grüne Punkt markiert den ersten Down-Chirp, der ebenfalls als Synchronisationspunkt dient. An der X-Koordinate der Preamble ist zu erkennen, dass diese keine Schwankungen ausserhalb des ± 1 Fehlers aufweisen. Der kombinierte Offset von CFO und STO beträgt hier 103 Samples. Das Ende der Datenübertragung ist ebenfalls in dieser Abbildung zu sehen. Bei der Entscheidung, welches Symbol übertragen wird, spielt das maximale Argument aller Samples eines Symbols eine Rolle. Die Grösse des Arguments gibt dabei die Übereinstimmung an. Liegt dieses nahe bei Null, kann davon ausgegangen werden, dass kein Symbol mehr übertragen wurde.

Nun wird das Signal mittels der oben beschriebenen Offsetkorrektur angepasst. Da es sich bei LoRaWAN um ein verschlüsseltes Protokoll handelt, können die gewonnenen Daten nicht mit den Daten des LoRaWAN-Servers verglichen werden. Aus diesem Grund wird hier auf die visuellen Merkmale einer erfolgreichen Demodulation zurückgegriffen.

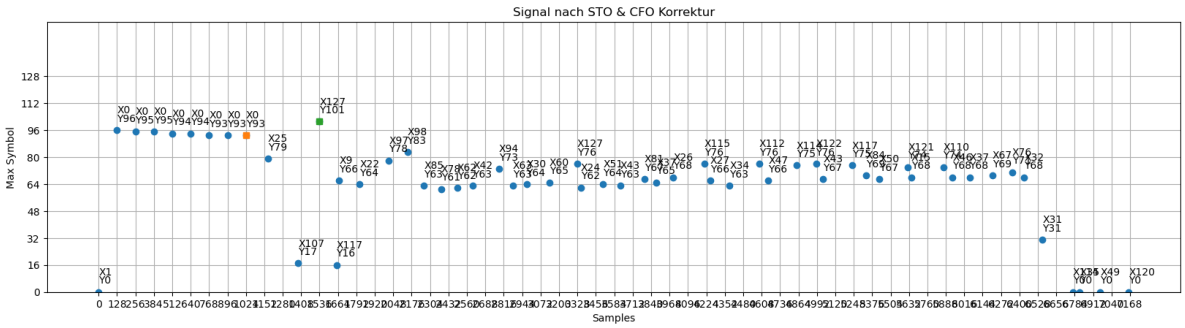

Figure 7.4: Reales Signal mit CFO und STO Korrekturen

In Figure 7.4 wurde das gleiche reale Signal wie in Figure 7.3 verwendet. Hier ist es wieder wichtig zu erwähnen, dass die Information im Signal auf der X-Achse versteckt ist. Dies kann an der X-Koordinate jedes einzelnen Punktes abgelesen werden. Auf der Y-Achse wird lediglich die Stärke der Korrelation dargestellt. Somit sind Änderungen in der Y-Achse zwischen den beiden Figure 7.3 und Figure 7.4 nicht relevant. Bei einem erfolgreich synchronisierten Signal liegen alle Upchirps der Preamble auf dem relativen Nullpunkt der X-Achse. Aufgrund der begrenzten Synchronisationsgrösse von ganzen Samples kann es hier zu Fehlern von ± 1 kommen. Aufgrund der Länge der Preamble von $12\frac{1}{4}$ ergibt sich bei der Demodulation der Datensymbolen ein konstanter Offset von $\frac{1}{4} \cdot 2^{SF}$. Dieser wird jedoch bei der Datenextraktion korrigiert.

8 Fazit

Am Ende dieser Arbeit möchte ich die Gelegenheit nutzen, meine eigene Meinung in die Ergebnisse einfliessen zu lassen. Meiner Meinung nach sind die Ergebnisse kritisch zu betrachten. Der offensichtlichste Grund dafür ist die mangelhafte Methode zur Verifizierung der Ergebnisse. Zwei Punkte helfen mir jedoch, genügend Vertrauen in diese Daten zu haben. Der erste ist die intensive Arbeit mit den synthetischen Signalen, die klar zeigt, dass es so funktioniert wie in der Masterarbeit der EPFL beschrieben. Tapparel et al. (2020) Zweitens zeigen die Daten keine Anomalien, die einen stutzig machen.

Durch das Schreiben dieser Arbeit habe ich eine neue Welt des technischen Schreibens kennengelernt, die mir in Zukunft sicher von Nutzen sein wird. Auch wenn es zusätzliche Zeit in Anspruch genommen hat, hat es sich meiner Meinung nach gelohnt, diesen Weg zu gehen. Durch diese Arbeit habe ich endlich ein klares Verständnis dafür bekommen, warum mich die LoRa-Technologie so interessiert und fasziniert. Durch diesen tieferen Einblick in die Funktionsweise und deren Umsetzung konnte ich erste Erfahrungen im Reverse Engineering von Produkten und Patenten sammeln. Die Auseinandersetzung mit vorhandenem Wissen hat mich auch dem wissenschaftlichen Arbeiten näher gebracht.

Mein Ziel für diese Arbeit ist es, eine grundlegende Basis für die Demodulation von LoRa-Paketen zu entwickeln. Dementsprechend wurde auch viel Zeit in die Dokumentation und den Code investiert. Damit soll auch späteren Studenten und Studentinnen die Möglichkeit gegeben werden, auf diesem Projekt aufzubauen.

8.1 Verbesserungen

Für zukünftige Arbeiten dieser Grössenordnung ist es wichtig, schnell eine klare Struktur sowohl für die Dokumentation als auch für die Arbeit zu finden. Die Arbeit mit dem Versionsverwaltungstool muss in Zukunft verbessert werden. Die Anzahl der Commits ist deutlich zu gering und zu unspezifisch.

8.2 Ausblick

Obwohl das Projekt meiner Meinung nach einen akzeptablen Stand erreicht hat, gibt es auch hier noch offene Arbeiten und Anknüpfungspunkte für eine weitere Industriearbeit oder eine Bachelorarbeit. Nicht alle Teile der Masterarbeit der EPFL wurden in diese Arbeit integriert. Tapparel et al. (2020) Zum Beispiel wurde die CFO & STO Synchronisation nur auf ganze Samples genau programmiert. In der Arbeit wird jedoch auch der Prozess der Offsetkorrektur für kleinere als ganze Samples erklärt. Auch das Thema Datenrekonstruktion und Fehleranfälligkeit wurde hier nicht diskutiert. So fehlt die Graycode-Kodierung sowie die Berechnung von Prüfsummen. Weiterhin ist die aktuelle Demodulation auf einen Spreadingfaktor von $SF = 7$ eingestellt und wird nicht aktiv erkannt. Das Ende einer Übertragung wird ebenfalls nicht aktiv erkannt. Zuletzt fehlt eine Möglichkeit durch eine Einbindung mittels einer API oder einer Bibliothek in z.B. GNU-Radio Companion. Eine grosse Aufgabe wäre hingegen das Senden eines LoRa-Signals durch den Lime-SDR Mini. Nur so wäre eine komplexe Validierung und Kommunikation möglich. Um das Programmierte effektiv nutzen zu können, müsste eine Möglichkeit zur Verarbeitung von Livedaten geschaffen werden.

Quellenverzeichnis

"Chirp Spread Spectrum." 2023. https://de.wikipedia.org/wiki/Chirp_Spread_Spectrum; Wikipedia.

Dultz, Johannes. 2023. "LoRa-WAN Technology." https://www.lora-wan.de/lora/.

"ERC Recommendation." 2023. https://docdb.cept.org/download/3700; CEPT CEE.

"Frequency Plans." 2023. https://www.thethingsnetwork.org/docs/lorawan/frequency-plans/; The Things Network.

"LMS7002M." 2023. https://limemicro.com/technology/lms7002m/; Lime Microsystems.

"LoRa Technology Overview." 2023b. https://www.semtech.com/lora; Semtech.

———. 2023a. https://www.semtech.com/lora; Semtech.

"LoRaWAN Distance Record." 2023. https://www.thethingsnetwork.org/article/lorawan-distance-world-record; The Things Network.

"LPN LoRaWAN-Das Schweizer IoT-Netzwerk." 2023. https://www.swisscom.ch/de/business/enterprise/angebot/iot/iot-connectivity/lpn.html; Swisscom.

Montagny, Sylvain. 2022. "LoRa-LoRaWAN and Internet of Things." https://www.univ-smb.fr/lorawan/wp-content/uploads/2022/01/Book-LoRa-LoRaWAN-and-Internet-of-Things.pdf.

Slats, Laurens. 2023. "A Brief History of LoRa: Three Inventors Share Their Story." https://blog.semtech.com/a-brief-history-of-lora-three-inventors-share-their-personal-story-at-the-things-conference.

"Switzerland." 2023. https://www.thethingsnetwork.org/country/switzerland/; The Things Network.

Tapparel, Joachim, Orion Afisiadis, Paul Mayoraz, Alexios Balatsoukas-Stimming, and Andreas Burg. 2020. "An Open-Source LoRa Physical Layer Prototype on GNU Radio." In *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE. https://doi.org/10.1109/spawc48557.2020.9154273.

"The Things Network." 2023. https://www.thethingsnetwork.org/; The Things Network.

"The Things NODE." 2023. https://www.thethingsnetwork.org/marketplace/product/the-things-node; The Things Network.

Abbildungsverzeichnis

A Aufgabenstellung

HSLU Hochschule Luzern

Horw, 18. September 2023
Seite 1/2

Industrieprojekt im Studiengang Elektrotechnik und Informationstechnologie

Aufgabe für Herrn Yannick Kulli

Demodulation von LoRa-Signalen mit LimeSDR Mini

Fachliche Schwerpunkte
Nachrichtentechnik / Signal Processing

Einleitung
Mit dem LimeSDR Mini, einem Software-Defined Radio (SDR) Modul für PCs, lassen sich Radiosignale von 10 MHz bis 3.5 GHz breitbandig senden und empfangen.

Aufgabenstellung
Das für Anwendungen im Bereich des Internet der Dinge (IoT) zur Verfügung stehende Frequenzspektrum ist begrenzt. Mit zunehmender Verbreitung wird die Effizienz der Übertragungsverfahren immer wichtiger. In der Schweiz werden Netzwerke auf Basis von LoRaWAN (Long Range Wide Area Network) genutzt. In dieser Arbeit geht es nun darum, mit dem LimeSDR eine LoRa-Verbindung zu einem vorhandenen Device oder einem Gateway aufzubauen. Auf dieser Basis sollen später IoT-Übertragungsverfahren untersucht werden, die robust bezüglich Interferenz sind.

In einem vorangegangenen Projekt wurde die Synchronisation auf ankommende LoRa-Signalpakete realisiert. Darauf soll nun aufgebaut werden. Die genauen Anforderungen werden zu Beginn des Projekts besprochen und festgelegt.

Termine

Start der Arbeit:	Montag 18.9.2023
Zwischenpräsentation:	Zu vereinbaren im Zeitraum 16.10. – 18.11.2023
Abgabe Schlussbericht:	Freitag 22.12.2023 vor 16:00 im D311 an C. Haab
Abgabe Digitale Doku:	Gemäss separater Anweisung der Studiengangleitung
Abschlusspräsentation:	Zu vereinbaren im Zeitraum 3.1. – 20.1.2024

Dokumentation

Der gebundene Schlussbericht enthält keine Selbständigkeitserklärung und ist in einfacher Ausführung zu erstellen. Er enthält zudem zwingend

- einen sehr kurzen, englischen Abstract (auf einer Seite)
- Ein Titelblatt, gleich hinter dem Deckblatt, gemäss Weisungen der Studiengangleitung
- Eine SD-Hülle, innen, auf der Rückseite des Berichtes für den Betreuer

Alle Exemplare des Schlussberichtes müssen komplett und termingerecht gemäss Angaben der Studiengangleitung abgeben werden. Zusätzlich muss eine SD-Speicherkarte mit dem Bericht (inkl. Anhänge), dem Poster und den Präsentationen, Messdaten, Programmen, Auswertungen, usw. unmittelbar nach der Präsentation abgeben werden.
Die gesamte Dokumentation ist zudem gemäss Anweisungen der Studiengangleitung elektronisch auf einen Server zu laden. Sämtliche abzugebende Teile der Dokumentation sind Bestandteile der Beurteilung.

Fachliteratur/Web-Links/Hilfsmittel

https://wiki.myriadrf.org/LimeSDR-Mini

Geheimhaltungsstufe: Öffentlich

Verantwortlicher Dozent/Betreuungsteam, Industriepartner

Dozent

	Prof. Dr. Thomas Hunziker	thomas.hunziker@hslu.ch

Experte

	noch offen	…

Hochschule Luzern
Technik & Architektur

Prof. Dr. Thomas Hunziker

B Bewertungsraster

HSLU Hochschule Luzern

Bewertungsraster PAIND (Studiengang Elektrotechnik und Informationstechnologie)

HS 2023

Titel: .....

Student/Studentin: .....

Dozent/Dozentin: ..... Experte/Expertin: .....

	Vorgängige Punktezuordnung	Erreichte Punkte
1. Prozess (30 Punkte)		0
- Erfassen der Problemstellung, Abgrenzung, Zielsetzung	5	
- Planung, Organisation, Systematik	5	
- Erwerb von neuem Wissen	5	
- Selbstständigkeit, Einsatz, Problemlösung, Motivation	10	
- Kommunikation und Teamarbeit	5	
2. Bericht und Inhalt (50 Punkte)		0
- Zielerreichung (Erfüllungsgrad)	15	
- Lösungswege überzeugend und nachvollziehbar	10	
- Innovation, Kreativität der erarbeiteten Lösungen	5	
- Validierung und Diskussion der Lösungen	5	
- Berichtstruktur (Layout, Sprache und Vollständigkeit)	15	
3. Präsentation (20 Punkte)		0
- Projektzwischenpräsentation (Kalenderwoche 10 bis 14)	5	
- Gehalt Schlusspräsentation	5	
- Schlusspräsentation: Präsentationstechnik, Stimulanz, Sprache, Auftreten	5	
- Mündliche Befragung	5	
Erreichte Punktzahl		**0**
Bewertung (A: hervorragend = 92 … 100, B: sehr gut = 84…91, C: gut = 76…83, D: befriedigend = 68…75, E: ausreichend = 60…67, F: nicht bestanden < 60)		

Verbale Gesamtbeurteilung (muss explizit in einigen kurzen aber prägnanten Sätzen formuliert werden)**:**

Horw, Dozent/Dozentin: Experte/Expertin

FH Zentralschweiz