Design and Implementation of LoRa Physical Layer in GNU Radio

Tapparel Joachim JOACHIM.TAPPAREL@EPFL.CH
Telecommunication Circuits Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland

Andreas Burg ANDREAS.BURG@EPFL.CH
Telecommunication Circuits Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland

Abstract

LoRa is the physical layer of LoRaWAN, one of the most popular low-power wide-area network (LPWAN) technologies for the Internet of Things (IoT). LoRa uses a proprietary chirp spread spectrum modulation. The modulation is used together with error correction coding and interleaving to achieve long-range communication with low energy consumption. In the past years, many reverse engineering attempts have been made and led to an overall understanding of the encoding and modulation scheme used by the physical layer of LoRa. In this paper, we present in detail all the signal processing operations required to transmit and receive a LoRa frame for all the modes that are supported by commercial devices. We further develop the synchronization methods used by our open-source implementation of a LoRa transceiver that is fully compatible and has been tested extensively with commercial LoRa devices. Finally, we evaluate the performance of our LoRa transceiver implemented in GNU Radio and available on GitHub.

1. Introduction

In the last decade, low-power wide-area network (LPWAN) technologies have gained popularity for their ability to provide long-range communication with low energy consumption. Among the many technologies that have been developed, LoRaWAN has emerged as one of the most popular and widely deployed solutions. While the medium access control (MAC) layer LoRaWAN is an open standard, its physical layer, simply named long range (LoRa) (Seller & Sornin, U.S. Patent 9 252 834, Feb. 2016), is proprietary and has motivated many reverse engineering attempts. Those attempts have led to an overall understanding of the encoding and modulation scheme used by the physical layer of LoRa. As LoRa uses a chirp spread spectrum (CSS) modulation that presents many interesting properties, it has been the subject of multiple research papers that have analyzed its performance and proposed improvements. However, the details that would allow compatibility with commercial devices are mostly overlooked or simplified for the sake of analysis. In this paper, we present a detailed description of the LoRa physical layer used by commercial devices. These insights have enabled the implementation of a LoRa transceiver that is fully compatible with commercial devices and offers new evaluation possibilities by using the open-source GNU Radio software in conjunction with off-the-shelf LoRa transceivers.

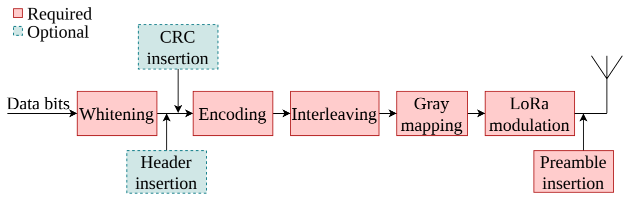

Figure 1. LoRa transmitter chain

Contributions: We provide a comprehensive overview of the LoRa physical layer, covering modulation, frame structure, and frame detection for all modes supported by commercial LoRa devices. Furthermore, we provide detailed insights into our open-source GNU Radio LoRa transceiver and assess the performance of our implementation with both simulation and experimental evaluation.

2. LoRa Frame Generation

A LoRa frame is composed of a preamble, a header, a payload, and a cyclic redundancy check (CRC) code. Figure 1 illustrates the operations required for constructing a frame. In the following, we detail each of these operations and provide the equations required to implement them in a transceiver. All following operations performed on binary values are using modulo-2 arithmetic unless specified otherwise.

Proceedings of the 14th GNU Radio Conference, Copyright 2024 by the author(s).

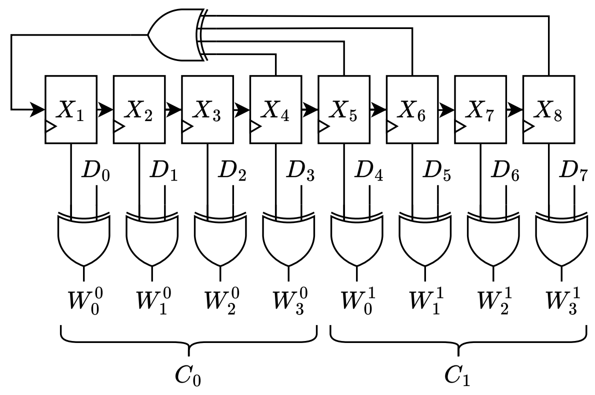

Figure 2. Whitening operation on one byte of data

L_4	L_5	L_6	L_7
L_0	L_1	L_2	L_3
C	P_0	P_1	P_2
H_4	▨	▨	▨
H_0	H_1	H_2	H_3

Figure 3. Explicit header structure, where L is the payload length in bytes, C indicates the presence of a CRC, P is the number of parity bits, and where H is the header checksum.

2.1. Whitening

The transmitter chain starts with a whitening operation that is used to randomize the data before encoding, removing any data specific binary patterns. The whitening operation consists in an XOR operation between the data and a pseudo-random sequence. As proposed in (Xu et al., 2023), the pseudo-random sequence can be generated by a linear feedback shift register (LFSR) with a feedback polynomial of $x^8+x^6+x^5+x^4+1$. For each step of the LFSR, one byte of the pseudo-random sequence is generated. Each input byte $\mathbf{D} = [D_0 \ldots, D_7]$, represented with right-MSB, is whitened as illustrated in Figure 2. After the whitening, each byte is split into nibbles, whereof the low nibble is first and the high nibble is second.

2.2. PHY Header

The PHY header is an optional field that is used to carry information necessary to decode the frame. The header is composed of the following information: the payload length, the presence of a payload CRC, the code rate, and a header checksum. The structure of the five codewords composing the header is illustrated in Figure 3. Note that contrary to the payload, the nibbles in the header are ordered high nibble first, low nibble second. The header checksum $\mathbf{h}_c$ is computed using the linear equation

$$\mathbf{h}_c = \mathcal{G} \cdot \mathbf{h}, \tag{1}$$

where $\mathbf{h}$ is a vector of the header bits $[L_7, L_6, \ldots, C]^T$, and where $\mathcal{G}$ is the generator matrix given by

$$\mathcal{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{2}$$

The presence of the header is not mandatory if all the aforementioned parameters are known in advance by the receiver. The transmission mode that does not include the PHY header is referred to as *implicit header mode*, as opposed to the *explicit header mode*.

2.3. Payload CRC

An optional cyclic redundancy check (CRC) of two bytes is computed over the payload to allow the receiver to verify the integrity of the decoded data. The CRC is computed using the generator polynomial CCITT-16, that is described by $x^{16}+x^{12}+x^5+1$. It is important to note that the CRC is only computed over the payload and not over the header. Furthermore, the last two bytes of payload are not included in the CRC computation, but instead are XORed to the CRC output.

2.4. Encoding

LoRa defines the number of parity bits $P \in \{1, \ldots, 4\}$, which corresponds to code rates $\frac{4}{4+P}$. The code parameters $(8,4)$ to $(6,4)$ are based on hamming code with cropped parity bits, while $(5,4)$ corresponds to a single bit checksum. The codeword $\mathbf{c}$ that corresponds to the whitened data $\mathbf{w} = [w[0] \ldots w[3]]$ is given by

$$\mathbf{c} = \begin{cases} \mathbf{w} \cdot \mathcal{G}_0 & \text{if} \quad \mathrm{P} = 1, \\ \mathbf{w} \cdot \mathcal{G}_1 & \text{otherwise}, \end{cases} \tag{3}$$

with the two generator matrices given by

$$\mathcal{G}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \tag{4}$$

$$\mathcal{G}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{5}$$

For $P < 4$, the codeword $\mathbf{c}$ is cropped to the first $(4+P)$ bits.

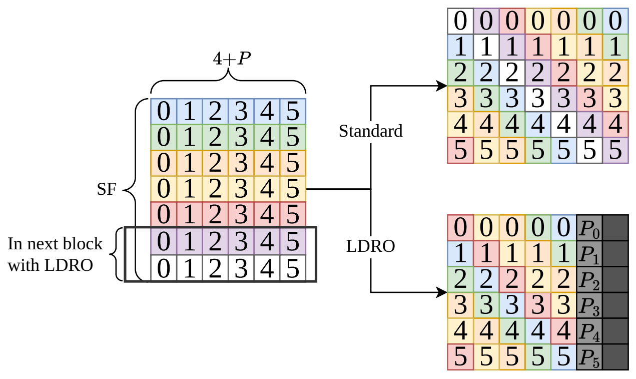

Figure 4. Illustration of interleaving of one block for SF=7 and P=2 (i.e., code rate 4/6) without and with low data rate optimization (LDRO)

2.5. Interleaving

Multiple codewords are interleaved using a diagonal block interleaver. The interleaver is used to spread the bits of a codeword over multiple symbols to improve the error correction capability of the code. The size of an interleaver block is given by the number of bits in a codeword, i.e., $(4{+}P)$, and by the spreading factor (SF) used. The interleaver takes an SF $\times$ $(4{+}P)$ binary matrix $\mathcal{D}$ in which each row is a codeword and each column is a bit of the codeword and outputs an interleaved matrix $\mathcal{I}$ of size $(4{+}P) \times$ SF. The diagonal interleaving is defined by the equation

$$\mathcal{I}_{i,j} = \mathcal{D}_{[i-(j+1)] \bmod \text{SF},i} \quad , \tag{6}$$

with $i \in \{0, \ldots, 3{+}P\}$ and $j \in \{0, \ldots, \text{SF}{-}1\}$. LoRa defines a low data rate optimized mode (LDRO) that reduces the number of bits in an interleaver block to $(\text{SF}{-}2) \times (4{+}P)$. After interleaving, two bits are appended to each interleaved codeword: a parity bit P_i computed from XORing the SF-2 bits of the interleaved codeword and a zero. The interleaving of a block with SF $= 7$ and $P = 2$ is illustrated in Figure 4.

2.6. Gray Mapping

Each row of the interleaved matrix is interpreted as gray encoded binary values. The gray mapping operation is used to convert the gray encoded values to a decimal value. After the gray mapping, each symbol s only differs by one bit from symbols $s \pm 1$. As discussed in Section 3.2.2, some hardware impairments increase the likelihood of mistaking a symbol for an adjacent one, therefore the gray mapping minimizes the number of errors when such demodulation mistakes occur. The gray mapping used by LoRa is given by

$$s = [\mathbf{I}_i \oplus (\mathbf{I}_i \gg 1) + 1] \bmod \text{SF}, \tag{7}$$

where $\mathbf{I}_i$ is the i-th row of the interleaved matrix, where $\oplus$ denotes the binary XOR operation and where $\gg$ denotes the binary right shift operation. Note that the gray mapping used by LoRa includes an arbitrary offset of $+1$ to the mapped values.

2.7. Fixed Parameters for First Interleaver Block

In order to fully decode a LoRa frame, the receiver needs to know some information contained in the header, such as the payload length, the presence of a CRC, and the code rate. However, the receiver does not know the header before decoding the first interleaver block. To solve this issue, the first block is encoded with a fixed set of parameters that are independent of the user-defined parameters. The first interleaver block is encoded with the most robust code rate of 4/8 and, for SFs larger than six, interleaved using the low data rate optimized mode. For SF five and six, LDRO is not used to allow the entire *explicit header* to fit in the first interleaver block. This choice of parameters maximizes the probability to correctly decode the header content. It is important to note that the first block always uses the parameters mentioned, even if the transmission uses *implicit header mode*, i.e., no header is transmitted.

2.8. LoRa Modulation

LoRa uses a linear chirp spread spectrum (CSS) modulation that is defined by two transmission parameters: the spreading factor (SF) and the signal bandwidth B. Each LoRa symbol is a linearly increasing frequency chirp composed of $N = 2^{SF}$ chips that carry SF $\in \{5, \ldots, 12\}$ bits of information. The frequency of a baseband symbol with value $s \in \{0, \ldots, N{-}1\}$ spans the entire signal bandwidth B during the symbol duration $T_s = \frac{N}{B}$. The frequency of the symbol starts at $\left(\frac{sB}{N} - \frac{B}{2}\right)$ and increase by $\frac{B}{N}$ for each chip until it reaches the maximum frequency $\frac{B}{2}$. The frequency then folds back to $-\frac{B}{2}$ and continues to increase linearly for each subsequent chip, as illustrated on Figure 5. When sampled at a frequency f_s, the fold in frequency occurs at the sample $n_{\text{fold}} = (N - s)\,\frac{f_s}{B}$. As described in (Afisiadis et al., 2019; Chiani & Elzanaty, 2019), the baseband signal of a LoRa symbol sampled at a frequency f_s can be written as

$$x_s[n] = e^{j2\pi\left(\frac{n^2}{2N}\left(\frac{B}{f_s}\right)^2 + \left(\frac{s}{N} - \frac{1}{2} - u[n-n_{\text{fold}}]\right)\left(\frac{B}{f_s}\right)n\right)}, \tag{8}$$

where $n \in \{0, \ldots, N\frac{f_s}{B}\}$ and where $u[n]$ denotes the unit step function. The LoRa modulation also supports linearly decreasing frequency chirps, referred to as downchirps. Downchirps are obtained by taking the complex conjugate of the upchirps $x_s[n]$. Typically, LoRaWAN uses the upchirp modulation for the uplink and the downchirp modulation for the downlink to avoid interference between the two directions.

$$N_s = \overbrace{4.25 + N_{\text{up}}}^{\text{Preamble}} + \overbrace{\underbrace{8}_{\text{First interleaver block}} + \left\lceil \frac{2D - \text{SF} + 4C + 5H + 2u(\text{SF-7})}{\text{SF} - 2L} \right\rceil \cdot (4{+}P)}^{\text{Payload}} \tag{10}$$

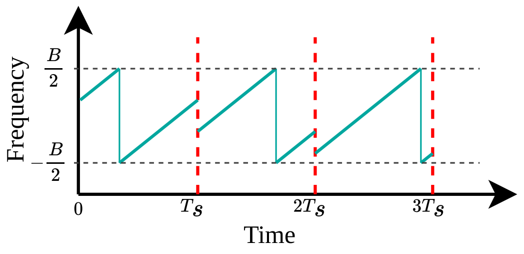

Figure 5. Spectrogram of LoRa symbols delimited by red dashed lines

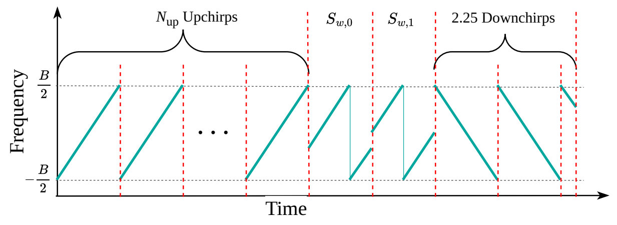

Figure 6. Spectrogram of the preamble of a LoRa frame

2.9. Preamble

For frame synchronization, the LoRa frame starts with a preamble that is composed of a sequence of symbols known by both the transmitter and the receiver. The preamble is composed of N_{up} repetitions of a reference upchirp (i.e., modulated symbol with value zero x_0) followed by two modulated symbols called *sync word* and finally two and a quarter downchirps (i.e., the complex conjugate of reference upchirp). The preamble structure is illustrated in Figure 6. The number of repetition of the reference upchirp N_{up} can be chosen in the range $\{6, \dots, 65535\}$, with eight being the most common value for LoRaWAN. The *sync word*, sometimes referred to as *network identifier*, is used to identify frames sent on the same network. Typical values for the *sync word* S_w are 0x34 for LoRaWAN and 0x12 for private networks. However, any value in the range $\{0\text{x}10, \dots, 0\text{xFF}\}$ is valid. Some transceivers define the *sync word* as a 16-bit value that is equivalent to the 8-bit representation given by the relation $0\text{xXY} = 0\text{xX4Y4}$ The value of the two modulated symbols $S_{w,0}$ and $S_{w,1}$ that correspond to S_w are given by

$$\begin{aligned} S_{w,0} &= (S_w \gg 4) \cdot 8, \\ S_{w,1} &= (S_w \land \texttt{0x0F}) \cdot 8, \end{aligned} \tag{9}$$

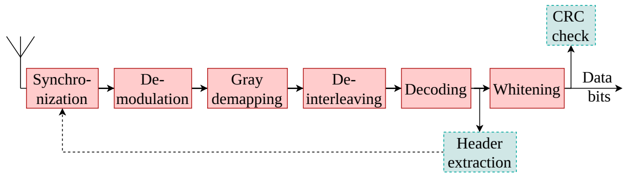

Figure 7. LoRa receiver chain

where $\land$ denotes the bitwise AND operation.

The number of symbols composing the final LoRa frame N_s can be computed using (10), where D denotes the length of data in byte, C denotes the presence of a payload CRC, H denotes the use of the *explicit header mode*, and L indicates the use of LDRO. The corresponding duration of the frame is given by

$$T_{\text{frame}} = N_s \frac{2^{\text{SF}}}{B}. \tag{11}$$

3. LoRa Frame Detection

In this section, we present the operations necessary to detect a LoRa frame. We describe the functioning of the demodulation, the frame synchronization, and the soft-decision decoding for the receiver chain illustrated in Figure 7.

3.1. Demodulation

As discussed in (Chiani & Elzanaty, 2019), the demodulation of a LoRa signal sampled at a sampling rate equal to the bandwidth can be efficiently performed in two steps: a dechirping operation followed by a discrete Fourier transform. Indeed, the symbol estimate $\hat{s}$ can be obtained by computing the correlation between the received symbol $\mathbf{y}^l$ and all possible symbol $\mathbf{x}_s$ as

$$\hat{s}^l = \underset{s \in \{0,\dots,2^{\text{SF}}\}}{\text{argmax}} \left|\text{corr}(\mathbf{y}^l, \mathbf{x}_s)\right|. \tag{12}$$

The correlation between the received symbol y and the signal of a symbol candidate x_s can be expressed as

$$\text{corr}(\mathbf{y}^l, \mathbf{x}_s) = \sum_{n=0}^{N-1} y[n] x_s^*[n] = \sum_{n=0}^{N-1} y[n] x_s^*[n] \underbrace{x_0[n] x_0^*[n]}_{=1}, \tag{13}$$

where $\cdot^*$ denotes the complex conjugate operation. After reordering of the terms and some straightforward algebraic simplifications, the correlation can be expressed as

$$\text{corr}(\mathbf{y}^l, \mathbf{x}_s) = \sum_{n=0}^{N-1} \underbrace{(y[n]x_0^*[n])}_{\text{Dechirping}} \cdot e^{-j2\pi \frac{s}{N} n}. \tag{14}$$

One can notice that each correlation with a candidate symbol $\mathbf{x}_s$ corresponds to computing one Fourier coefficient of the received signal multiplied by the conjugate of the reference symbol $\mathbf{x}_0$. This observation allows to take advantage of the fast Fourier transform (FFT) algorithm to compute all the correlations with a reduced complexity. The output bin of the DFT that has the highest magnitude provides the best estimate of the transmitted symbol. The demodulation operation is finally given by

$$\hat{s}^l = \underset{k \in \{0,\ldots,2^{\text{SF}}\}}{\text{argmax}} \left|\tilde{Y}^l[k]\right|, \tag{15}$$

$$\tilde{\mathbf{Y}}^l = \text{DFT}\left(\mathbf{y}^l \odot \mathbf{x}_0^*\right), \tag{16}$$

where $\odot$ denotes the element-wise multiplication operation, where $\mathbf{y}^l = [y[lN] \ldots y[N + lN - 1]]$ denotes the samples of the l-th received symbol, and where $\mathbf{x}_0 = [x_0[0] \ldots x_0[N-1]]$.

3.2. Frame Synchronization

The receiver chain starts with a frame synchronization operation that is used to detect the start of a LoRa frame, estimate the most critical offsets introduced by hardware impairments, and compensate digitally those offsets in the received IQ samples. In the following sections we summarize the synchronization steps required to correctly detect symbol and refer to the corresponding papers for the detailed derivations and evaluation of the different estimators.

3.2.1. Preamble Detection

The reception of a LoRa frames starts with the detection of the preamble. The receiver demodulate symbols from an arbitrary starting time offset using (16). The identification of a preamble relies on the repetition of the upchirps present in the preamble of a frame. The demodulated values will repeat themselves irrespective of the CFO and STO. However, the presence of CFO and STO introduces a higher probability to mistake a symbol value for an adjacent one. After finding repetitions of same demodulated values, within a margin of ± 1 symbol value, the receiver can perform a first rough synchronization. The rough synchronization compensates for the combination of all offsets by assuming that only STO is present. Note that this is a necessary step for the correct estimation of the STO discussed later. The receiver simply shifts the input samples by the value of the repeating symbol. After this rough synchronization the symbol repetitions in the preamble are demodulated as symbols with value zero.

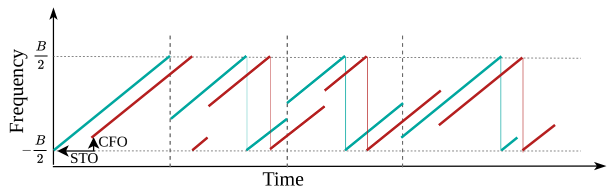

Figure 8. Illustration of the effects of CFO and STO on the LoRa spectrum for $f_s = B$

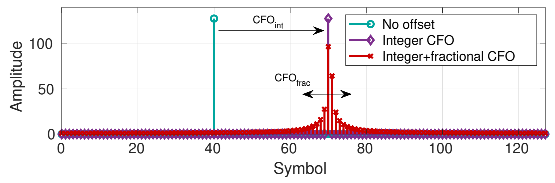

Figure 9. Illustration of the effects of integer and fractional CFO on the output of the symbol demodulation $|\tilde{Y}|$

3.2.2. Offset Estimation and Compensation

The main hardware offsets that affect the reception of a LoRa frame are the sampling time offset (STO) and carrier frequency offset (CFO). As illustrated on Figure 8 the STO shifts the received symbol in time while the CFO shifts the signal in the frequency domain. When the sampling rate is equal to the bandwidth, the CFO rotates the symbol between $-B/2$ and $B/2$. As described in (Xhonneux et al., 2022), it is convenient to separate the CFO Δf_c and STO τ into an integer and fractional components as

$$\Delta f_c = \frac{B}{N}(L_{\text{CFO}} + \lambda_{\text{CFO}}), \quad \tau = \frac{L_{\text{STO}} + \lambda_{\text{STO}}}{B}, \tag{17}$$

with $L_x \in \mathbb{Z}$ and $\lambda_x \in [-0.5, 0.5)$. Figure 9 illustrates the different effect that the integer and fractional CFO have on the output of the symbol demodulation. While the integer CFO shifts the entire spectrum, the fractional offset spreads the symbol energy into adjacent frequency bins. As the effects of both types of offsets on the despread symbol are very different, it is beneficial to estimate the integer and fractional parts separately. The STO has an impact on the received symbol similar to the CFO due to the linear chirp modulation. In the remainder of this section, we detail the methods used to estimate and compensate the fractional and integer offsets. Note that all estimators are using the despread signal, i.e., after the dechirping and DFT operations, as the received (spread) LoRa signal can be far below the thermal noise floor.

Fractional CFO Estimation (Bernier et al., 2020) proposes a method to estimate the fractional CFO $\hat{\lambda}_{\text{CFO}}$ based on the phase rotation between consecutive symbols which as been extended in (Xhonneux et al., 2022) to include all the upchirps of the preamble as

$$\hat{\lambda}_{\text{CFO}} = \frac{1}{2\pi}\angle\left(\sum_{l=2}^{N_{\text{up}}}\sum_{p=-2}^{2}\tilde{Y}^{l}[i+p]\cdot\tilde{Y}^{l-1}[i+p]\right), \quad (18)$$

where $i = \arg\max_k |\tilde{Y}^l[k]|$ and where $\angle(\cdot)$ denotes the argument of a complex number. The estimator is independent of the other offsets present in the received signal as only the fractional CFO contributes to the linear rotation of the phase of each symbol. The fractional CFO can then be compensated in the received signal using

$$y'[n] = y[n]\cdot e^{-j2\pi\frac{\hat{\lambda}_{\text{CFO}}}{N}\frac{B}{f_s}n}, \quad n\in\mathbb{N}_0. \quad (19)$$

Fractional STO Estimation After compensation of the fractional CFO, the dechirped signal only contains a pure frequency tone. However, due to the fractional STO, the frequency is located between two bins of the DFT. We use the general frequency estimator proposed in (Yang & Wei, 2011) to obtain the fractional STO based on the values of the DFT bins. When applied to our specific case, the frequency estimator can be expressed as

$$\hat{\lambda}_{\text{STO}} = \frac{N}{2\pi}\frac{P[i+1]-P[i-1]}{u(P[i+1]+P[i-1])+vP[i]}, \quad (20)$$

$$\mathbf{P} = \sum_{l=1}^{N_{\text{up}}}\left|\text{DFT}_{2^{\text{SF}}}(\mathbf{y}^l\odot\mathbf{x}_o^*)\right|^2, \quad (21)$$

$$u = \frac{64\cdot N}{(\pi^5+32\pi)}, \quad v = \frac{u\pi^2}{4}, \quad (22)$$

where $i = \arg\max_k(P[k])$, where DFT_n denotes a DFT with a zero-padding of size n. The compensation of the fractional STO can be done by oversampling the received signal by a factor f_s/B, shifting the samples by $\lceil f_s/B\cdot\hat{\lambda}_{\text{STO}}\rfloor$, and downsampling the signal to the original sampling rate with the right offset.

Integer CFO and STO Estimation After compensation of the fractional offsets, the integer CFO and STO can be estimated using the method proposed in (Bernier et al., 2020; Xhonneux et al., 2022). The preamble of LoRa includes two and a quarter downchirps at its end. While both CFO from STO cause a similar shift on the upchirps values, the effects of both offsets on the downchirps are opposite. Therefore, the demodulation of the upchirps gives a symbol value $s_{\text{up}} = (L_{\text{CFO}} + L_{\text{STO}}) \bmod N$ while the demodulation of the downchirps gives a symbol value $s_{\text{down}} = (L_{\text{CFO}} - L_{\text{STO}}) \bmod N$. Based on the demodulated values s_{up} and s_{down}, the integer parts of the CFO and STO can be estimated as

$$\hat{L}_{\text{CFO}} = \frac{1}{2}\Gamma_N\left[(s_{\text{up}}+s_{\text{down}}) \bmod N\right], \quad (23)$$

where

$$\Gamma_N[k] = k - N\cdot u(k-\frac{N}{2}). \quad (24)$$

Finally the integer STO can be estimated as

$$\hat{L}_{\text{STO}} = (s_{up} - \hat{L}_{\text{CFO}}) \bmod N. \quad (25)$$

Similarly to their fractional counterparts, the integer offsets can be compensated using (19) for the CFO and a shift of $\hat{L}_{\text{STO}}$ samples for the STO.

Sampling Frequency Offset As the duration of a LoRa frame can be up to several seconds, the receiver has to account for the sampling frequency offset (SFO) present in the received signal. The SFO causes a linear increase (or decrease) of the STO over time. While negligible for small frame durations, the SFO can lead to an STO accumulation larger than half a chip duration and cause every symbol to be mistaken by an adjacent one. As both the CFO and STO are caused by the same hardware impairments, i.e., the local oscillator imprecision, an SFO estimate $\hat{\Delta}_{f_s}$ can be obtained from the CFO estimation using the simple relation

$$\hat{\Delta}_{f_s} = \hat{\Delta}_{f_c}\frac{f_s}{f_c}, \quad (26)$$

where f_c denotes the carrier frequency of the LoRa signal. The SFO can then be compensated by either periodically dropping (or duplicating) a sample. For every sample received, the receiver accumulates a sample time offset of Δ_{f_s}/f_s. The receiver compensates the accumulated STO by dropping (or duplicating) a sample every $\left\lceil\frac{B}{2\Delta_{f_s}}\right\rfloor$ received samples. This period corresponds to the time it takes for the STO to accumulate a duration of half a sample. After the estimation of all offsets discussed previously, the receiver can compensate each of them in the remaining samples of the frame that contains data.

3.3. Frame Decoding

Each of the transmitter block has a counterpart in the receiver chain. While the demapping, deinterleaving and dewhitening are trivally reversible based on the transmitter description, the decoding require a more complex approach. The decoding can be done using hard-decision decoding or soft-decision decoding.

The hard-decision decoding of codes using 2 to 4 parity bits P is done with the help of a syndrome $\mathbf{S}$. This syndrome of each of the received codeword can be obtained by

$$\mathbf{S} = \mathbf{c}\cdot\mathcal{H}_P^T, \quad (27)$$

where $\mathcal{H}_P$ denotes a submatrix composed of the first P rows and $4{+}P$ columns of the matrix

$$\mathcal{H} = \begin{bmatrix}1&1&1&0&1&0&0&0\\0&1&1&1&0&1&0&0\\1&1&0&1&0&0&1&0\\1&1&1&1&1&1&1&1\end{bmatrix}. \quad (28)$$

Table 1. Number of detectable and correctable errors for different code rates

Code rate	Detectable	Correctable
$4/5$	1	0
$4/6$	2	0
$4/7$	1	1
$4/8$	2	1

For the code rate $4/5$, the syndrome is computed by XORing of all the bits of the codeword. Depending on the number of parity bit used, a syndrome can either be unique and allow error correction or not and only allow error detection. The number of errors that can be detected and corrected for each code rate is summarized in Table 1.

For soft-decision decoding, the receiver does not take any decision on the received bits before the decoding step. The demodulation outputs the probability of every possible transmit symbol hypothesis, instead of only the most probable symbol value. As discussed in (Tapparel et al., 2021), the log-likelihood ratio (LLR) of each bit of a row of the interleaver block $\mathbf{I}^l$ can be computed from the DFT of the dechirped signal as

$$\mathbf{I}^l[m] = \max_{\bar{s}:g_m(\bar{s})=1}\left[\log I_0\left(\frac{\sqrt{P}}{\sigma^2}\left|\tilde{Y}^l[\bar{s}]\right|\right)\right] \tag{29}$$

$$- \max_{\bar{s}:g_m(\bar{s})=0}\left[\log I_0\left(\frac{\sqrt{P}}{\sigma^2}\left|\tilde{Y}^l[\bar{s}]\right|\right)\right], \tag{30}$$

where I_0 denotes the modified Bessel function of the first kind, where $g_m(\bar{s})$ denotes the m-th bit of the symbol $\bar{s}$ in the Gray labeling, where P denotes the power of the received signal, and where σ^2 is the variance of the white noise at the receiver. The LLR of each bit of the interleaver block can then be deinterleaved using the same method as the hard-decision decoding. Finally, the LLRs is fed to a soft-input decoder such as the one proposed in (Müller et al., 2011) to obtain the decoded payload.

4. Implementation and Results

In this section, we present some particularities of the implementation of the LoRa transceiver in GNU Radio. We then present results obtained from simulated and experimental measurements.

4.1. GNU Radio Transceiver

All the previously described functionalities have been implemented in GNU Radio to build a fully functional LoRa transceiver. The implementation is an out-of-tree module for GNU Radio available on GitHub (gr-lora_sdr). All the blocks present in the block diagram of the transmitter and receiver chains are implemented as distinct GNU Radio blocks. The flowgraphs presented on Figure 10 can be used to transmit and receive LoRa frames using any SDR supported by GNU Radio. A flowgraph is a synchronous dataflow graph where each block processes a chunk of data and passes it to the next block. However, as LoRa transmits discrete frames, each block needs to know when a frame starts and ends. Furthermore, the first interleaver block is encoded using a fixed set of parameters and has to be treated differently than the other interleaver blocks. To manage these different parts of a frame, two tags are generated per frame and attached to specific samples: one at the start of the frame that contains relevant information for the processing of the whole frame and one at the end of the first interleaver block. Furthermore, the receiver needs to know the content of the *explicit header* to process the payload. While waiting for the header to be decoded, the synchronization block buffers samples until the header content has been successfully recovered. Once the header content is known, an asynchronous message is sent to the synchronization block with all the header information. The header information is then added to the tags located at the end of the first interleaver block to be propagated to the rest of the receiver chain.

4.2. Results

In this section, we evaluate our GNU Radio LoRa receiver implementation. We first present results obtained from simulated AWGN channels and then compare the results with experimental measurements obtained from two LoRa transceivers based on software-defined radios for both transmission and reception.

Figure 11 shows the frame error rate (FER) for spreading factor seven in a simulated additive white Gaussian noise (AWGN) channel using hard and soft-decision decoding. The simulations are performed on GNU Radio using the LoRa blocks described in the previous section. The FER is computed on the frame with a payload of 16 bytes for which the preamble has been detected by the receiver. We can observe the advantages that soft-decision decoding provides, as the error rate of the $(6, 4)$ code with soft-decision decoding is equivalent to the error rate of the $(8, 4)$ code that uses hard-decision decoding. Both experiments achieve similar error correction capabilities while reducing the number of transmitted bits by $\sim 25\%$.

Figure 12 shows the FER obtained from a simulated AWGN channel and from USRP measurements. The experimental measurements were performed using the GNU Radio implementation of the transceiver and NI-2920 USRPs for transmission and reception. To enable comparison with simulated performance and to allow the reproduction of the results, the USRPs are connected via a coaxial cable to restrict the transmission channel to an AWGN

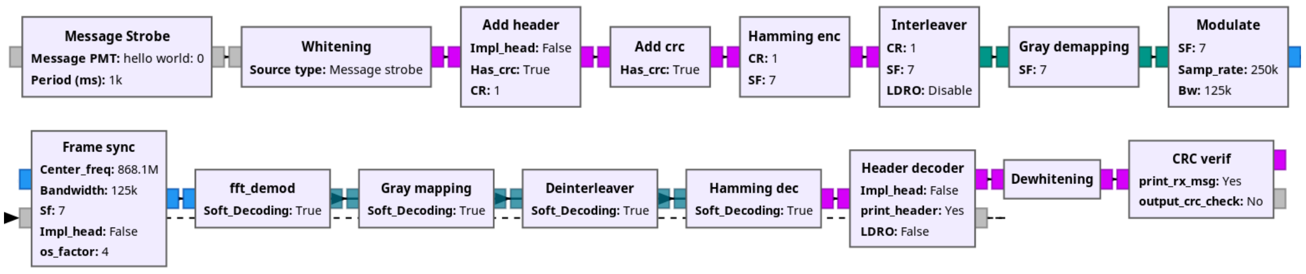

Figure 10. GNU Radio flowgraph for the LoRa transmitter (top) and receiver (bottom)

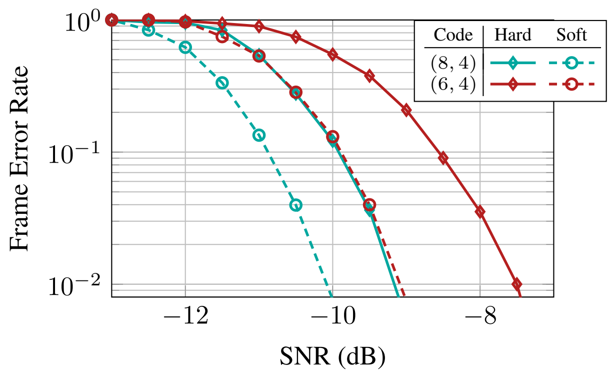

Figure 11. Frame error rate for spreading factor seven in a simulated AWGN channel using hard and soft-decision decoding

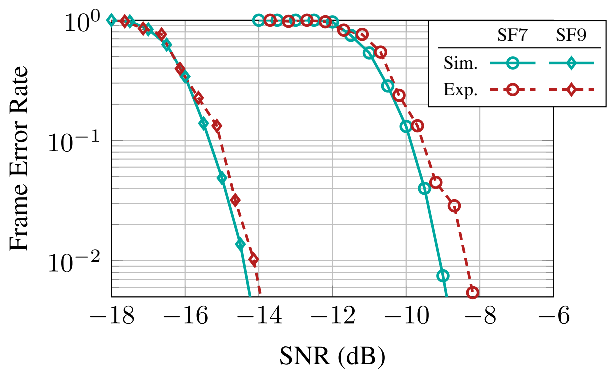

Figure 12. Frame error rate for SF7, code $(6, 4)$, and soft-decision decoding for a simulated AWGN channel and USRP measurements

channel. In addition to AWGN, both STO and CFO are present in the hardware (but are ideal in the simulation) due to the lack of synchronization between the two USRPs. We can observe that the error rates of the experimental testbed stay within 1 dB from the simulated curves, demonstrating the proper functioning of the synchronization and overall implementation of our LoRa transceiver.

5. Conclusion

In this paper, we presented the modulation and coding scheme of LoRa and detailed the operations necessary to build and detect a LoRa frame compatible with commercial devices. We then described our open-source implementation of a LoRa transceiver in GNU Radio and presented results obtained from simulated and experimental measurements. We finally showed that our GNU Radio implementation of the LoRa transceiver only suffers from a loss of 1 dB SNR between simulated AWGN and experimental measurements that included hardware impairments.

References

Afisiadis, Orion, Cotting, Matthieu, Burg, Andreas, and Balatsoukas-Stimming, Alexios. On the error rate of the LoRa modulation with interference. *IEEE Trans. Wirel. Commun.*, 19(2):1292–1304, 2019.

Bernier, Carolynn, Dehmas, François, and Deparis, Nicolas. Low complexity LoRa frame synchronization for ultra-low power software-defined radios. *IEEE Trans. Commun.*, 68(5):3140–3152, 2020.

Chiani, Marco and Elzanaty, Ahmed. On the LoRa modulation for IoT: Waveform properties and spectral analysis. *IEEE Internet Things J.*, 6(5):8463–8470, 2019.

gr-lora_sdr. A fully-functional GNU Radio software-defined radio implementation of a LoRa transceiver. URL https://github.com/tapparelj/gr-lora_sdr. Accessed: Sept. 14, 2024.

Müller, Benjamin, Holters, Martin, and Zölzer, Udo. Low complexity soft-input soft-output hamming decoder. In *2011 50th FITCE Congress - "ICT: Bridging an Ever*

Shifting Digital Divide", pp. 1–5, 2011. doi: 10.1109/FITCE.2011.6133448.

Seller, Olivier B. A. and Sornin, Nicolas. Low power long range transmitter, U.S. Patent 9 252 834, Feb. 2016.

Tapparel, Joachim, Xhonneux, Mathieu, Bol, David, Louveaux, Jérôme, and Burg, Andreas. Enhancing the reliability of dense LoRaWAN networks with multi-user receivers. *IEEE Open J. Commun. Soc.*, 2:2725–2738, 2021. doi: 10.1109/OJCOMS.2021.3134091.

Xhonneux, Mathieu, Afisiadis, Orion, Bol, David, and Louveaux, Jérôme. A low-complexity LoRa synchronization algorithm robust to sampling time offsets. *IEEE Internet Things J.*, 9(5):3756–3769, 2022. doi: 10.1109/JIOT.2021.3101002.

Xu, Zhenqiang, Tong, Shuai, Xie, Pengjin, and Wang, Jiliang. From demodulation to decoding: Toward complete lora phy understanding and implementation. *ACM Trans. Sen. Netw.*, 18(4), jan 2023. ISSN 1550-4859.

Yang, Cui and Wei, Gang. A noniterative frequency estimator with rational combination of three spectrum lines. *IEEE Transactions on Signal Processing*, 59(10):5065–5070, 2011. doi: 10.1109/TSP.2011.2160257.