Decoding LoRa: Realizing a Modern LPWAN with SDR

Matthew Knight MATT@BASTILLE.NET
Bastille Networks, 101 2nd Street Suite 510, San Francisco, CA

Balint Seeber BALINT@BASTILLE.NET
Bastille Networks, 101 2nd Street Suite 510, San Francisco, CA

Abstract

LoRa is an emerging Low Power Wide Area Network (LPWAN), a type of wireless communication technology suitable for connecting low power embedded devices over long ranges. This paper details the modulation and encoding elements that comprise the LoRa PHY, the structure of which is the result of the author's recent blind analysis of the protocol. It also introduces *gr-lora*, an open source software defined implementation of the PHY that will empower wireless developers and security researchers to investigate this nascent protocol.

1. Introduction

LoRa is an emerging Low Power Wide Area Network (LPWAN) designed to provide wireless connectivity to high endurance embedded devices over very long ranges. It enables functionality that is analogous to cellular data service, but optimized for "Internet of Things"-focused applications. LoRa excels in embedded roles by sacrificing data rates (kilobits/second) in exchange for very long ranges (several miles) and high endurance (years on AA batteries).

1.1. LoRa Technology Overview

LoRa is noteworthy for having been designed for use and compliance within the unlicensed 900 MHz ISM frequency band. This represents a dramatic departure from the cellular network paradigm, which requires network operators to license expensive exclusive spectrum in order to operate at higher power and longer or continuous duty cycles on protected channels. By complying with permissive regulatory statues, deploying LoRa infrastructure capable of providing several miles of coverage on a public or private network may soon be as accessible as deploying an 802.11 network in one's home or office.

While OFDM, FSK, and ARFz PHYs are common in modern wireless systems, LoRa leverages a unique chirp spread spectrum modulation (CSS) to achieve exceptional link budget and low power performance within these noisy and contested ISM channels.

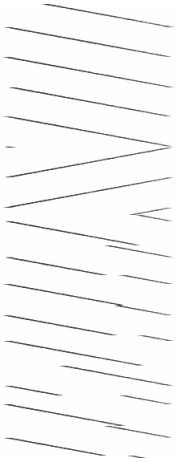

Figure 1. Spectrogram excerpt of a LoRa chirp
y-axis: Time, *x-axis:* Frequency

LoRa's CSS represents symbols as instantaneous changes in the frequency of a chirp – in essence, symbols are represented as frequency modulated chirps.

1.2. Initial Research

The LoRa PHY is closed source and proprietary, thus there were no official references or protocol specifications to base an open source implementation on. The description of the LoRa PHY layer that follows is the result of the author's blind analysis of the protocol.

Open source intelligence was instrumental in understanding the basis of the protocol. While there was no documentation available for the PHY, a reference for the Lo-

Proceedings of the 6^{th} GNU Radio Conference, Copyright 2016 by the author(s).

RaWAN network stack was available and described some elements of the PHY on a conceptual level (LoR, 2015). Additionally, several application notes for Semtech LoRa transceiver ICs described concepts implemented by the PHY (Sem, 2013b) (Sem, 2015). Finally, there were two community projects that served as useful inspiration: a partial open source implementation in the *rtl-sdrangelove* open source software project (Greb, 2016), and the *Decoding LoRa* wiki page that contained some high level observations about the PHY (Sikken, 2016).

While information gleaned from limited available documentation provided a large head start to understanding the protocol, many of the details proved to be a red herring. While Semtech Application Note AN1200.18 (Sem, 2013b) defines several whitening sequences, none of them turned out to be the one implemented in the PHY. Additionally, Semtech's European Patent (Seller & Sornin, 2015) defines a diagonal interleaver, however the interleaver implemented in silicon was a substantially different algorithm. Thus, the bulk of the research was conducted as a black box analysis. A Microchip RN2903 LoRa Mote provided reliable messages to decode, while an Ettus Research USRP B210, GNU Radio, and Python with Numpy and Scipy formed the basis of the decoding platform.

The Microchip Mote exposes a Semtech LoRa module to a USB interface, providing simple host-based radio configuration and control. While technically a LoRaWAN device, the MAC layer stack can be disabled to allow the device to send raw PHY frames without automatically populated addresses, checksums, or sequence numbers cluttering the payload (Mic, 2015). Thus, with knowledge and control of the data being modulated, messages can be crafted to reveal the structure of each encoding element. The blind signal analysis process has been thoroughly documented in recent conference presentations (Knight, 2016); this document describes the protocol features necessary for implementing the PHY in software.

The PHY mechanics revealed by the aforementioned blind analysis have been implemented as the open source GNU Radio out of tree module *gr-lora*. This module is presented to the community to empower both application developers and security researchers to interoperate with and explore the fundamentals of LoRa networks.

2. Demodulation

LoRa uses a proprietary chirp spread spectrum (CSS) modulation to encode data onto sweeped frequency chirps via instantaneous changes in frequency. A *chirp* is defined as a signal whose frequency changes at a fixed rate, which may be constant or exponential. An instantaneous change in frequency of the chirp, or lack thereof, constitutes a symbol.

LoRa encodes multiple bits of encoded information into each symbol; the number of encoded bits per symbol is referred to as the *spreading factor*, and may range from [7:12] in the US. The *chirp rate* is the first derivate of the chirp frequency, and is a function of the spreading factor and the signal bandwidth (125, 250, or 500 kHz in the US). The *chirp rate* is a function of the signal bandwidth and *spreading factor*, and is defined as (Sem, 2015):

$$\textit{chirp rate} = \frac{dfrequency}{dt} = \frac{bandwidth}{2^{spreadingfactor}}$$

The first step to demodulating LoRa is to de-chirp the signal. This is done by channelizing the complex baseband signal to its bandwidth and then multiplying the result against a locally generated chirp and its complex conjugate. This produces two IQ streams, where the chirped signals are "rotated" within the spectrum to have a chirp rate of 0, meaning each symbol resides on a unique constant frequency.

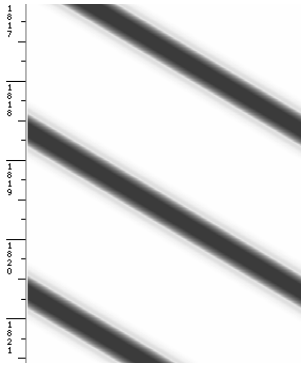

Figure 2. Spectrogram excerpt of a locally generated upchirp (Spreading Factor=8, Bandwidth=125kHz)

With the signal de-chirped, it may be treated as MFSK, where the number of possible frequencies is:

$$M = 2^{spreadingfactor}$$

By taking an M-bin wide FFT of each IQ stream at the symbol rate of the signal, the symbols may be resolved by finding the *argmax*, or bin index with the strongest component, of each FFT. Normalizing these symbols relative to the bin of the preamble produces the encoded data bitstream.

Accurate synchronization is crucial to properly resolving symbols. If synchronization is off, then when the FFT is taken each symbols' energy will be split between adjacent FFTs. Synchronization may be achieved in a computationally efficient manner by implementing the following three steps (Knight, 2016).

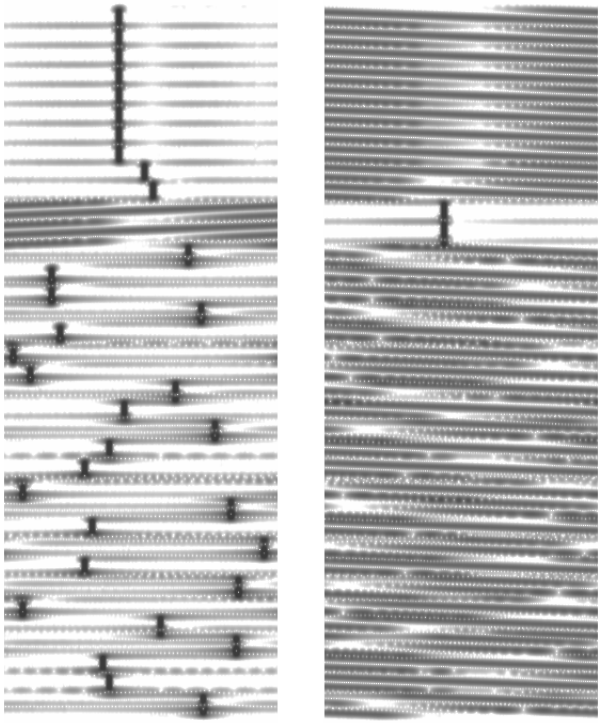

Figure 3. De-chirped LoRa signal
Left: Upchirps (preamble and body)
Right: Downchirps (start of frame delimiter)

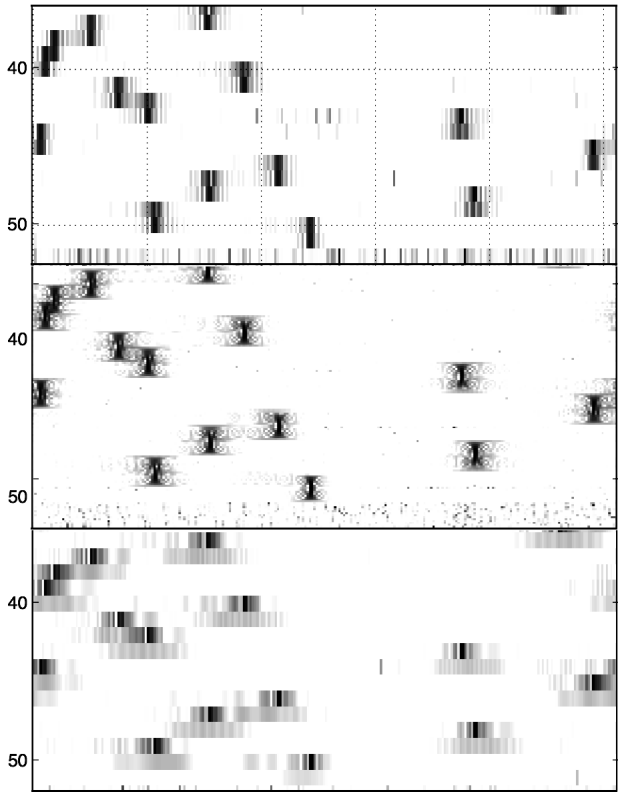

Figure 4. FFT synchronization overview
Top: Symbol collision due to poor synchronization (#39 and #50)
Middle: Overlapped FFTs
Bottom: Properly synchronized symbols

2.1. Preamble Detection

LoRa's preamble is represented by a number of continuous upchirps. Once de-chirped and passed through an FFT, a preamble may be identified if enough consecutive FFTs have the same *argmax*.

2.2. Start of Frame Delimiter Synchronization

With the preamble identified, increasing the time-based FFT resolution enables more accurate framing of the start of frame delimiter (SFD), which consists of two chirps with a negative chirp rate (downchirps). This is done by shifting the stream of de-chirped IQ samples through the FFT input buffer, processing each sample multiple times. Since computing overlapping FFTs is more computationally intensive, it is only done to frame the SFD; non-overlapped FFTs with each sample being processed exactly once are taken otherwise.

2.3. PHY Header and Service Data Unit Sampling

Once the SFD is synchronized, the PHY header and service data unit symbols may be acquired by using non-overlapping FFTs, with each sample processed once.

3. Decoding

In order to increase over the air resiliency, data is encoded before it is sent. Thus, the received symbols must be decoded in order to extract the data they represent. The decoding stages are as follows:

3.1. Gray Indexing

According to Semtech's European Patent application, encoded LoRa symbols are "gray indexed" before they are sent over the air (Seller & Sornin, 2015). This prevents off-by-one errors when resolving symbols. Gray indexing in this case actually refers to the inverse gray coding operation; thus in order to undo gray indexing the receiver must be gray code the received symbols.

3.2. Whitening

Data whitening is applied to induce randomness into the symbols to provide more features for clock recovery, should clock recovery be implemented by a receiver. The received symbols may be de-whitened by XORing them

against the same whitening sequence used by the transmitter. The whitening sequence implemented by *gr-lora* was experimentally derived during the prior blind analysis process after it was determined that the algorithms provided by Semtech's reference designs (Sem, 2013b) were not what is implemented by the protocol.

3.3. Interleaving

Interleaving is a process that scrambles data bits throughtout the packet. It is often combined with forward error correction to make the data more resilient to bursts of interference. One of Semtech's patent filings suggests that LoRa implements a diagonal interleaver and even defines a format, however experimenting revealed that the interleaver defined is not the interleaver implemented by the LoRa standard.

Thus, the interleaver was deciphered during the aforementioned blind analysis. Determining a solution involved constructing transmissions to exploit properties of the Hamming FEC codewords to map each codeword's position within the interleaver. In summary, the interleaver is a diagonal interleaver, with the most significant two bits reversed. Each diagonal word is offset, or rotated, by an arbitrary number of bits. Finally, the bits within each codeword are reversed.

3.4. Forward Error Correction

Forward error correction enables bits damaged during transmission to be recovered and corrected. It is similar to using a single parity bit or checksum, but it goes further by providing error correction under certain scenarios as well. LoRa uses Hamming FEC with a variable codeword size ranging from [5:8] bits and fixed data size of 4 bits per codeword (Seller & Sornin, 2015). The Hamming order is traditionally notated parenthetically:

$$Hamming(\#databits + \#paritybits, \#databits)$$

although LoRa nomenclature uses an equivalent fractional notation to describe the *code rate* (Sem, 2013a):

$$\frac{\#databits}{\#databits + \#paritybits}$$

A larger codeword size improves the robustness of the FEC: Hamming(5,4) and (6,4) provide error detection as a parity bit would, whereas (7,4) and (8,4) provide single bit correction with (8,4) additionally providing dual error detection. The Hamming bits are in a nonstandard order however and must be re-ordered before FEC can be applied.

4. Introducing *gr-lora*

gr-lora is an open source implementation of the LoRa PHY, presented to the community to accelerate embedded and IoT application development and security research. It defines blocks for implementing LoRa compliant receivers and transmitters using the methods described in this paper. Modulation and encoding, and consequently demodulation and decoding, are handled by separate blocks for modularity. The initial *gr-lora* release supports transmitting and receiving spreading factor 8, code rate 4/8 at all bandwidths, with additional permutations to be supported imminently. The author will maintain this module, and welcomes all contributions from the community.

gr-lora models modulation and encoding as separate blocks to allow for modularity and experimentation with different algorithms. This will enable users to use either the LoRa-compliant module described above or modulators and encoders of their own design.

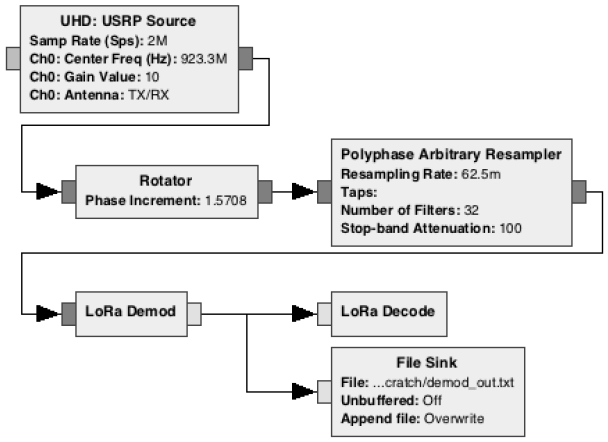

Figure 5. gr-lora flowgraph prepared for DEFCON 24, preconfigured SF=8 and CR=4/8

Since the initial publication of LoRa's PHY internals, two open source software defined radio implementations of the LoRa PHY have surfaced online. The first is LoRa-SDR for Pothos, written by Josh Blum (Blum, 2016). LoRa-SDR contains an accurate demodulator, however the decoding stages implement the incorrect red herring algorithms from Semtech's documentation, as mentioned in Section 1.2. The second implementation is a second *gr-lora* out of tree module written by github user rpp0 (rpp0, 2016). It implements a receiver in Python that uses a modified FM demodulation, however the author has not been able to successfully decode messages with it.

4.1. Future Work

Several improvements and extensions will need to be made before *gr-lora* can be considered fully featured.

1. **Additional spreading factors and code rates:** The decoder must be extended and validated for all valid spreading factors ([7:12]) and code rates ((8,4), (7,4), (6,4) (5,4)).

2. **Improve whitening sequence:** As discovered during the prior blind analysis process, the whitening algorithms described in Semtech's documentation (Sem, 2013b) do not match what is implemented by the protocol. Thus, the whitening algorithm implemented in *gr-lora* was experimentally derived. The current sequence was recovered by transmitting over the air long strings of 0x00s and determining the most commonly occurring bit patterns for each result, since the result of applying the whitening sequence to such a string would be the whitening sequence itself (Knight, 2016), but this method has shown to be lossy and error-prone. This process should be repeated using a larger sample set and a cable instead of live transmission.

3. **Header parsing:** The existing implementation does not decode and process LoRa's optional PHY header (Sem, 2013a). This is due to the author lacking hardware capable of disabling the header – since the header is always present, it was not able to be replaced with 0s to extract the portion of the whitening sequence that is applied to it. Once the full whitening sequence is established then the header structure may be evaluated and implemented.

4. **PHY CRC Handling:** The existing implementation does not handle LoRa's PHY header or PHY PDU CRCs (Sem, 2013a), due to the same whitening constraints as above. This should be implemented along with the header.

5. **Clock Recovery:** *gr-lora* implements no clock recovery by default, beyond the initial SFD synchronization. Clock recovery may be desirable should clock drift become an issue particularly with larger spreading factors, which have longer over-the-air transmission times (Sem, 2015).

6. **Implement Upper Layers:** While beyond the scope of this module, the author hopes that *gr-lora* may enable the development of open source LoRaWAN implementations and applications to network embedded devices at scale.

5. Conclusions

In summary, the composition of the LoRa PHY is now known and available for open source software defined radio integration and experimentation. *gr-lora* has been added to the software defined radio community's toolset, and may empower both the software development and security research communities to expand the Internet of Things and explore this emerging wireless technology.

References

Blum, Josh. Lora-sdr. Source Code on Github, 2016. https://github.com/myriadrf/LoRa-SDR.

Greb, John. rtl-sdrangelove. Source Code on Github, 2016. https://github.com/hexameron/rtl-sdrangelove/tree/master/plugins/channel/lora.

Knight, Matthew. Reversing lora: Exploring next-gen wireless. Jailbreak Security Summit Invited Presentation, 2016. http://jailbreaksecuritysummit.com/s/Reversing-Lora-Knight.pdf.

LoRaWAN Specification. LoRa Alliance, 2400 Camino Ramon, Suite 375, San Ramon, CA 94583, 2015. V1.0.

RN2903 LoRaTM Technology Module Command Reference User?s Guide. Microchip Technology, 2355 West Chandler Blvd., Chandler, AZ 85224, 2015. DS40001811A.

rpp0. gr-lora. Source Code on Github, 2016. https://github.com/rpp0/gr-lora.

Seller, Olivier Bernard Andre and Sornin, Nicolas. *Low Power Long Range Transmitter*. Semtech Corporation, Camarillo, CA 93012, 2015. Application Number 13154071.8/EP20130154071, Publication Number EP2763321 A1.

SX1272/3/6/7/8: LoRa Modem Designer's Guide. Semtech Corporation Advanced Communications and Sensing Products Division, 200 Flynn Road, Camarillo, CA 93012, 2013a. AN1200.13.

Implementing Data Whitening and CRC Calculation in Software on SX12xx Devices. Semtech Corporation Wireless and Sensing Products Division, 200 Flynn Road, Camarillo, CA 93012, 2013b. Application Note AN1200.18.

LoRa Modulation Basics. Semtech Corporation Wireless Sensing and Timing Products Division, 200 Flynn Road, Camarillo, CA 93012, 2015. Application Note AN1200.22.

Sikken, Bertrik. Decodinglora. Wiki Page, 2016. https://revspace.nl/DecodingLora.