ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

DIGITAL COMMUNICATIONS AND NETWORKS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Study and implementation of LoRa Physical Layer

Stratidakis Georgios

Master Thesis

April 2018

Supervisor: Angeliki Alexiou, Associate Professor, Department of Digital Systems, University of Piraeus

Abstract

In the future of telecommunications, every object will be connected to the Internet. These objects will be forming a network that senses and collects data from their environment. The name of this network is Internet of Things (IoT). It opens many new possibilities for applications, such as automating certain, previously manual, activities. Considering the impact that the classical Internet had on our lives, one can imagine the magnitude of the change that the IoT will bring.

In order to succeed, it needs the development of new communication protocols, that suits its requirements. It does not necessarily need high speeds, as the sheer number of devices connected to it, will provide huge amounts of data. Low Power-Wide Area Networks (LP-WAN) is a novel communication paradigm, that addresses the requirements of IoT for long range communications and low energy consumption by sacrificing the high data rates.

The purpose of this thesis is to study and implement LoRa, a LP-WAN communications technology, proposed by Semtech. It is low power, bandwidth scalable, resistant to interference, fading and the Doppler effect. Furthermore, it can be used for long range transmissions. These properties make LoRa ideal for the IoT.

Περίληψη

Στο μέλλον των τηλεπικοινωνιών, κάθε αντικείμενο θα είναι συνδεδεμένο στο διαδίκτυο. Θα σχηματίζουν ένα δίκτυο, το οποίο θα "αισθάνεται" και θα συλλέγει δεδομένα από τον περιβάλλον τους. Το όνομα του δικτύου είναι Internet of Things (Διαδίκτυο των Πραγμάτων). Το Internet of Things ανοίγει νέες δυνατότητες για εφαρμογές, όπως είναι η αυτοματοποίηση συγκεκριμένων, πρώην χειροκίνητων ενεργειών. Λαμβάνοντας υπόψη τον αντίκτυπο που είχε το κλασικό διαδίκτυο στις ζωές μας, μπορούμε να φανταστούμε το μέγεθος της αλλαγής που θα επιφέρει το IoT.

Για να επιτύχει, είναι αναγκαία η ανάπτυξη νέων πρωτοκόλλων επικοινωνίας, τα οποία θα ανταποκρίνονται στις απαιτήσεις του. Δεν χρειάζεται απαραίτητα υψηλές ταχύτητες, καθώς το πλήθος των συσκευών που συνδέεται σε αυτό, θα προσφέρει τεράστιες ποσότητες δεδομένων. Τα δίκτυα χαμηλής ισχύος ευρείας περιοχής (Low Power Wide Area Networks-LPWAN), είναι ένα νέο παράδειγμα επικοινωνίας, που καλύπτει τις απαιτήσεις του IoT για επικοινωνίες μεγάλης εμβέλειας και χαμηλή κατανάλωση ενέργειας, θυσιάζοντας τις υψηλές ταχύτητες.

Σκοπός της διπλωματικής εργασίας είναι η μελέτη και υλοποίηση του LoRa, ενός πρωτοκόλλου επικοινωνίας, το οποίο προτάθηκε από τη Semtech. Παρέχει χαμηλή κατανάλωση ενέργεια, κλιμακωτό εύρος ζώνης και είναι ανεκτικό στις παρεμβολές, τις διαλείψεις και το φαινόμενο Doppler. Επιπλέον, μπορεί να χρησιμοποιηθεί για επικοινωνίες μεγάλης εμβέλειας. Αυτές οι ιδιότητες, το κάνουν ιδανικό για το IoT.

Table of Contents

List of figures

List of Tables

1 Introduction

In recent years, the Internet of Things (IoT) technologies are growing fast and the number of objects that are connected to the Internet is growing fast. IoT has a large number of potential applications in areas such as telecommunications, security, home automation and health. Therefore, there is a need for the design for a new wireless network. This new network must be able to deliver different services, while having good quality and low energy consumption.

“LoRa is a spread spectrum modulation scheme, that is derived from Chirp Spread Spectrum and trades data rates for sensitivity. It implements a variable data rate by using orthogonal spreading factor. This allows the system designer to trade data rates for range or power. By doing this they can optimize the performance of the network” [12].

LoRaWAN is an emerging LP-WAN, a type of wireless communication technology suitable for connecting low power embedded devices over long ranges. It enables functionality that is analogous to cellular data service but is optimized for IoT-focused applications. LoRaWAN excels in embedded roles by sacrificing data rates in exchange for very long ranges and high endurance.

1.1 Thesis Objectives

The research objective of the thesis is the study and implementation of the physical layer of LoRa systems and to implement it in MATLAB. In particulat, this thesis will:

- provide information about the physical layer of LoRa, which includes the modulation, the coding scheme and the interleaver,
- introduce gr-lora, an open implementation of the physical layer based on GNU Radio,
- provide a new implementation of the LoRa physical layer in MATLAB,
- demonstrate experiments with over-the-air commercial LoRa modules and software radio-based LoRa implementations.

1.2 Thesis Structure

The second chapter provides an introduction to the IoT technologies, some use cases and other IoT technologies.

The third chapter analyzes the physical layer of LoRa.

The fourth chapter presents an experimentation setup using commercial hardware from pycom.

The fifth chapter presents an experimentation setup using the Software Defined Radio (SDR) technology and the open source gr-lora solution.

The sixth chapter presents the implementation of the physical layer of LoRa in MATLAB.

The seventh chapter describes further work that is needed and concludes the thesis.

2 Introduction to IoT

2.1 Background

The Internet of Things is the idea that every object, will be connected to the Internet. It enables the possibility of machines gathering information without our help and using it wherever it is needed, greatly reducing waste and cost. In IoT, as in every wireless technology compromises must be made. A choice must be made between two of three characteristics: long distance, high speed and low power (fig. 1).

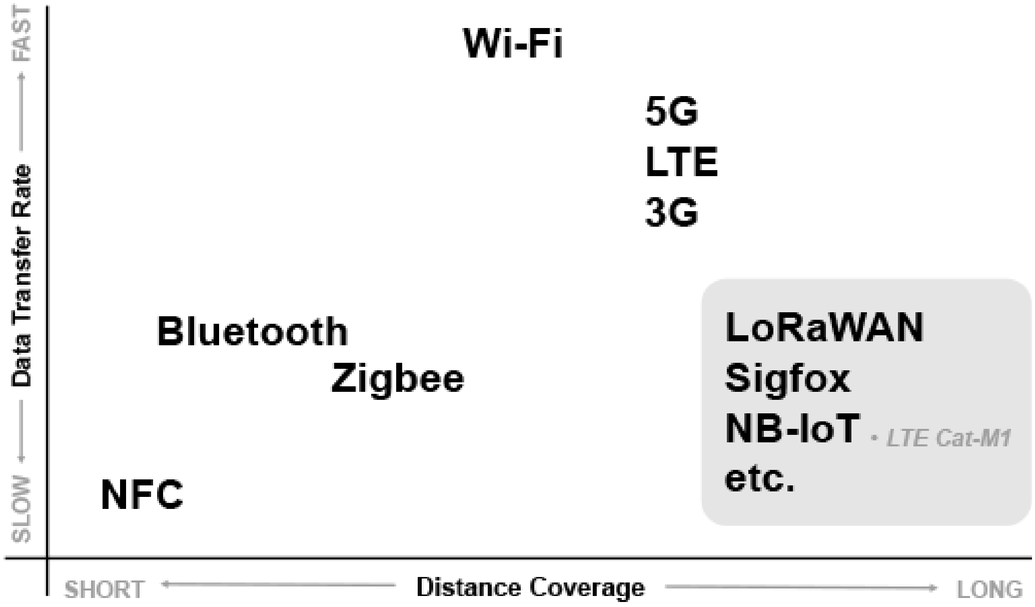


Figure 1. Compromises in wireless communications [24]

2.2 Use cases

As mentioned earlier, IoT has many areas of application. These include but are not limited to agriculture, logistics, metering, security, automation and health, that are presented below.

Smart Agriculture

The farming industry is a sector with small margins. The way to survive is by optimizing the production of corps and livestock. The use of sensors to optimize the mix of nutritions in the feeding of cattle, or collect data on weather, soil and crop quality, improves the yield of farming.

Logistics tracking

Every day, millions of shipments are being moved. The connection of these shipments along a supply chain and the analysis of the data generated from these connections, will unlock higher levels of operational efficiency and enable new customized, automated and dynamic services.

Smart Metering

Smart metering remotely collects electricity, water and gas meter data over the cellular network. It will help cut down the cost and manpower from manual metering.

Security and emergencies

Security has always been a very important matter. People want to feel safe at all times. Alarms and event detectors will help inform the users about home intrusions, fire outbreaks, or other emergencies even when they are far from their home. It will not only offer information about detected events, but also offer intelligence about events that may lead to an emergency like a sudden rise in temperature, or smoke.

Smart Home: Home automation

Smart home is an emerging and growing technology. It integrates many different technologies in a home network to make the quality of life better. The idea is to connect "everything" in home to the Internet in order to be able to check on the and control them remotely.

Smart Health

Smart Health is a new development in healthcare. Smart Health systems will provide services using a network between the devices that monitor the patients and computers. This will make possible the update of the information about the patients in real time.

2.3 Alternative IoT technologies

2.3.1 3GPP Long Term Evolution for the Machines (LTE-M)

LTE-M is a new cellular radio access technology specified by 3GPP to address the expanding market for low power wide area connectivity. It is a LPWAN technology that supports IoT with low device complexity, extended coverage and battery lifetime up to 10 years. Unlike other IoT technologies, where devices connect to their base stations, devices using LTE-M, connect to the LTE infrastructure. As a result, it is supported by all major mobile equipment and will coexist with 2G, 3G and 4G mobile networks, but the price of hardware is higher and it is dependent on mobile network operators.

2.3.2 3GPP NB-IoT

Narrowband IoT (NB-IoT) is a 3GPP feature that reuses various principles and building blocks of the LTE physical layer and higher protocol layers [25]. This makes the standardization, the product development and the network upgrade much faster. The complexity of NB-IoT devices will be even lower than that of GSM devices, as the synchronization signals and the physical layer procedures will be redesigned. NB-IoT offers extended coverage compared to GSM networks, improved uplink capacity for users in bad coverage area, using single tone single tone transmissions and guarantees 10 years of battery life.

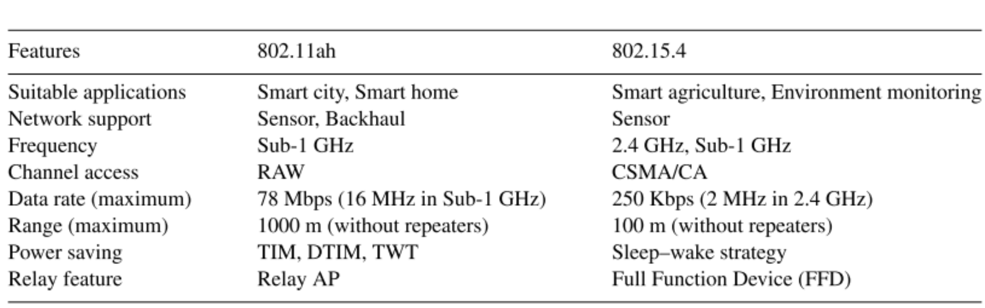
2.3.3 802.15.4 Low Rate- Wireless Personal Area Networks (LR-WPANs)

A Wireless Personal Area Network (WPAN) is used to convey information over relatively short distances and involve little or no infrastructure. 802.15.4 is a standard proposed to address the needs of LR- WPANs by maintaining a high level of simplicity, low cost and low power implementation. It operates in the 2.4 GHz ISM band, the 915 MHz and the 868 MHz frequency bands, using Direct Sequence Spread Spectrum (DSSS), which can accommodate data rates up to 250 Kbps [1],[2].

2.3.4 IEEE 802.11.ah

The IEEE 802.11.ah standard combines the advantages of Wi-Fi and low power sensor network communication technologies. The 802.11.ah standard enables single hop communications over distances up to 1000 m. Relay access points are used to extend the connectivity to Access Points (APs) that are two hops away.

Different Modulation and Coding Schemes (MCSs) are proposed based on the available channel bands, in order to provide different data rates. [11].



Features	802.11ah	802.15.4
Suitable applications	Smart city, Smart home	Smart agriculture, Environment monitoring
Network support	Sensor, Backhaul	Sensor
Frequency	Sub-1 GHz	2.4 GHz, Sub-1 GHz
Channel access	RAW	CSMA/CA
Data rate (maximum)	78 Mbps (16 MHz in Sub-1 GHz)	250 Kbps (2 MHz in 2.4 GHz)
Range (maximum)	1000 m (without repeaters)	100 m (without repeaters)
Power saving	TIM, DTIM, TWT	Sleep–wake strategy
Relay feature	Relay AP	Full Function Device (FFD)

Figure 2. 802.11.ah vs 802.15.4 [11]

2.3.5 SigFox

Sigfox is mostly widespread in Europe. Messages are transmitted using Ultra-Narrow Band (UNB) modulation with Differential Binary Phase-Shift Keying (DBPSK). It operates in the 200 KHz of the publicly available band to exchange radio messages over the air. Each message is 100 Hz wide and transferred at 100 or 600 bits per second, depending on the region. As a result, long distances can be achieved while being very robust against noise.

Sigfox uses a lightweight protocol to handle small messages. By sending less data, less energy is consumed, which means longer battery life can be achieved. An uplink message has up to 12 bytes payload and takes an average of 2 seconds over the air to reach the base stations looking for UNB signals to demodulate.

3 LoRa/LoRaWAN and SDR

3.1 LoRa/LoRaWAN Standard

LoRaWAN is a LP-WAN specification, intended for use in wireless objects that are battery powered, in a regional, national or global network. It targets secure bi-directional communication, mobility and localization services, all of which are key requirements of IoT.

“LoRaWAN defines the communication protocol and system architecture for network while the LoRa physical layer enables the long-range communication link. The protocol and network architecture have the most influence determining the battery lifetime of a node, the network capacity, the quality of service, the security and the variety of applications served by the network” [15].

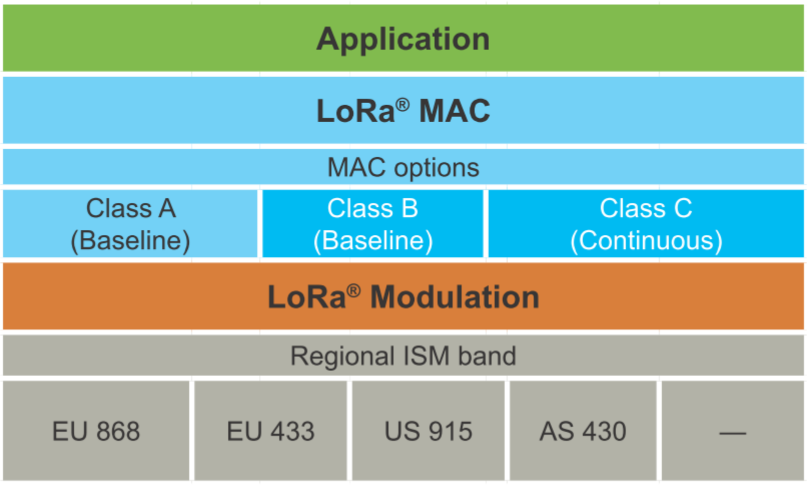


Figure 3. LoRaWAN Protocol Stack [16]

LoRa is a physical layer protocol developed by Semtech. It provides the infrastructure to implement IoT applications. Unlike LoRaWAN, which is open source, LoRa is property of Semtech and is only partially open source and information about its design or implementation is not readily available.

3.1.1 Key properties of LoRa

According to Semtech [7], the key properties of LoRa are:

Bandwidth Scalable

The modulation that LoRa uses is bandwidth and frequency scalable. It can be easily adapted for narrowband frequency hopping and wideband direct sequence applications, with a few simple configuration changes.

Constant Envelope/ Low-Power

LoRa, like FSK, is a constant envelope modulation scheme. Therefore, low-cost, low-power and high-efficiency power amplifier stages can be used. Additionally, the processing gain helps reduce the transmitting power, while maintaining the same or better link budget, compared to FSK.

High Robustness

LoRa is very resistant to in-band and out-of-band interference due to the high Bandwidth-Time (BT>1) product.

Multipath/fading resistant

The chirp pulse, is relatively broadband. Thus LoRa is multipath and fading resistant and ideal for use in urban and suburban environments, where multipath and fading are dominant.

Doppler resistant

Doppler shift causes a small frequency shift in the pulse. This introduces a negligible shift in the time axis of the baseband signal. The frequency offset tolerance mitigates the requirement for tight tolerance reference clock sources.

Long range capability

LoRa exceeds the link budget of FSK for fixed transmitting power and throughput. This in conjunction with the previous properties of LoRa, translates to significant improvements in range.

Enhanced Network Capacity

The LoRa modulation employs orthogonal Spreading Factors (SF), which enables multiple signals to be transmitted at the same time and the same channel. Signals with different SF appear as noise to the receivers and can be treated as such.

Ranging/Localization

A property of LoRa is the ability to discriminate linearly between frequency and time errors. LoRa, is derived from Chirp Spread Spectrum, which is used in radar applications. Consequently, LoRa is suited for ranging and localization applications.

3.1.2 Physical Layer

Spread Spectrum

Spread Spectrum is a means of transmission in which a signal is transmitted on a bandwidth much larger than the signal information bandwidth. To achieve this, a a pseudo-noise code is employed to “spread” the signal. At the receiver the signal energy is “despread” using the same pseudo-noise code.

Chirp Spread Spectrum

Chirp Spread Spectrum was developed for radar applications in the 1940’s. It is traditionally used in a number of military and secure communications applications.

A chirp pulse is frequency modulated pulse. the waveform is a chirp pulse, which is a frequency modulation pulse. In its duration the frequency changes in a monotonic manner from a lower value to a higher value (Upchirp), or the opposite (Downchirp) as shown in figures 4 and 5. The difference of these frequencies is approximately the bandwidth of the chirp pulse.

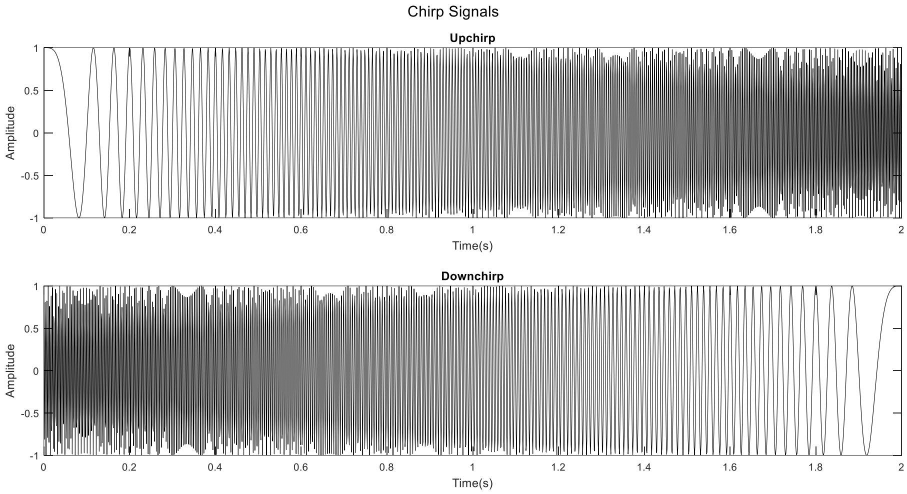


Figure 4. Chirp signals: Upchirp and Downchirp

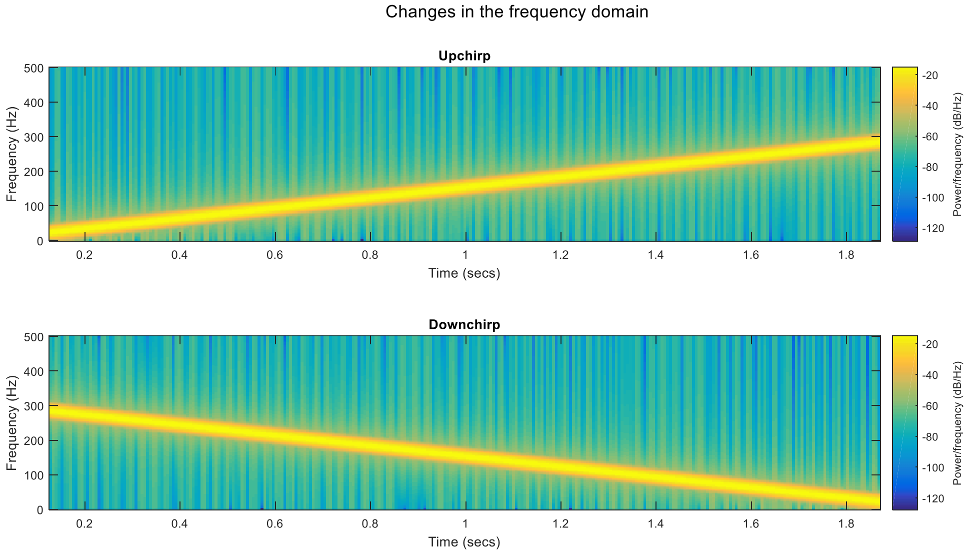


Figure 5. Changes in the frequency domain

Properties of chirp pulses

- High robustness: Due to the high BT product (Bandwidth x Time) chirp pulses are highly resistant to disturbances.
- Constant uniform Power Spectral Density (Fig.6)
- Multipath resistance: Due to the pulses being broadband, CSS is very resistant to multipath fading.
- Low Power consumption: CSS allows the designer to choose an analog implementation which often consumes less power.
- Low Latency: CSS does not need synchronization. New wireless connections can be established quickly.

- Resistance to Doppler effect: The Doppler effect causes a frequency shift of the chirp pulse, which introduces a negligible shift of the baseband signal on the time axis.
- Immunity to in-band interference: The scalable processing gain, enables selection of appropriate immunity level against in-band interference.
- Improved receiver sensitivity: Due to the timing and frequency offset being equivalent between the transmitter and the receiver [7].

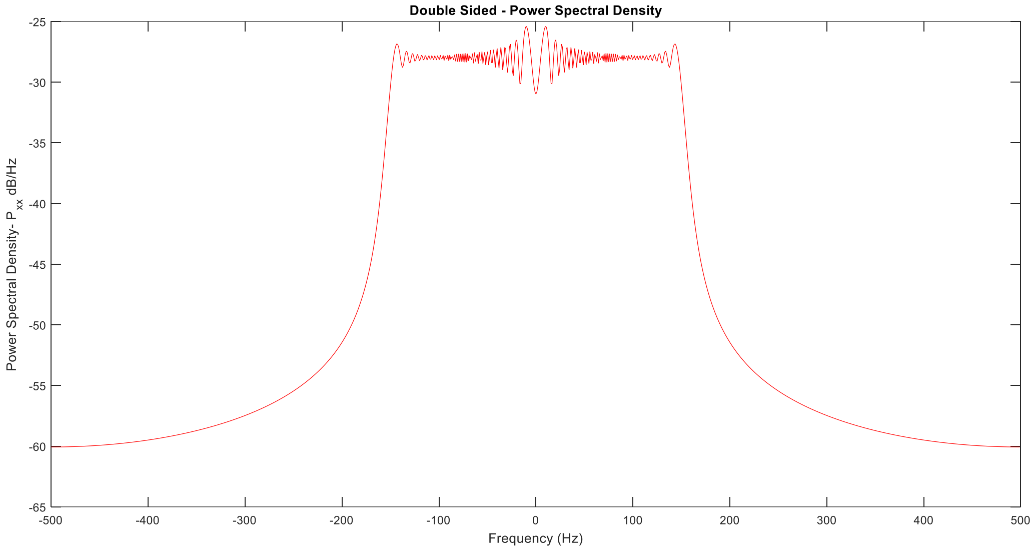


Figure 6. Power Spectral Density of chirp pulses

LoRa Spread Spectrum

In Semtechs LoRa modulation, the spreading of the spectrum, is achieved by generating a chirp signal that continuously varies in frequency [7]. By doing this, the timing and frequency offsets between the transmitter and the receiver are the same, which reduces the complexity of the receiver. In LoRa, the downchirp is the conjugate of the upchirp. Each symbol is a sweep over the whole bandwidth and represents a number of bits equal to the Spreading Factor. The starting frequency of each symbol can take one of 2^{SF} positions. By determining this starting position, the original codeword can be obtained. It should be noted that by raising the spreading factor, the range increases, while the data rate decreases.

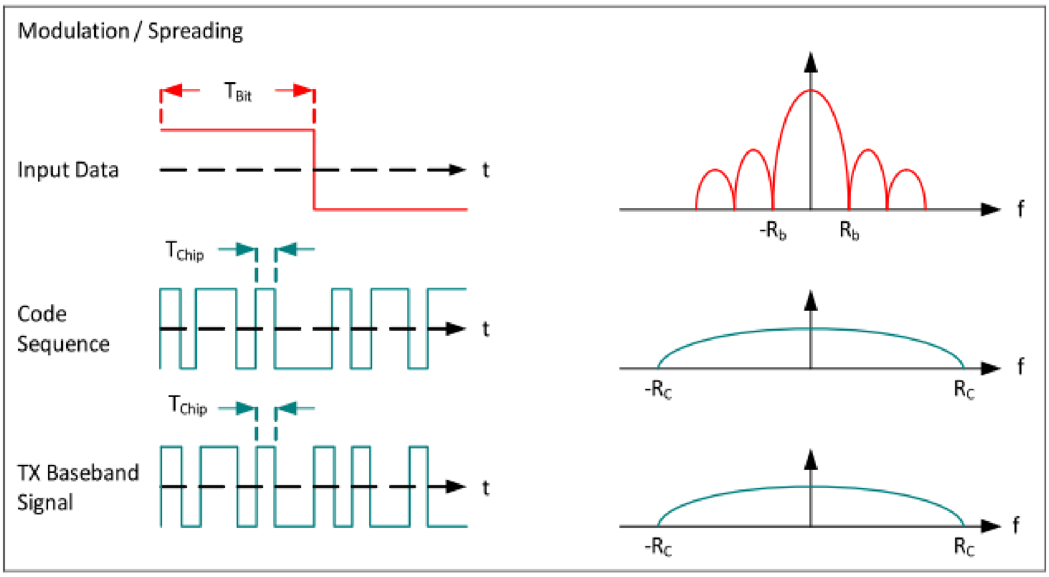


Figure 7. Modulation/ Spreading Process [7]

Demodulation

The demodulation is done by channelizing the complex baseband signal, to its bandwidth and then multiplying it with the downchirp. The result of this multiplication has constant frequency and is called dechirping. The symbols are derived by finding the index of the maximum value of the short-time Fourier transform of the dechirped signal.

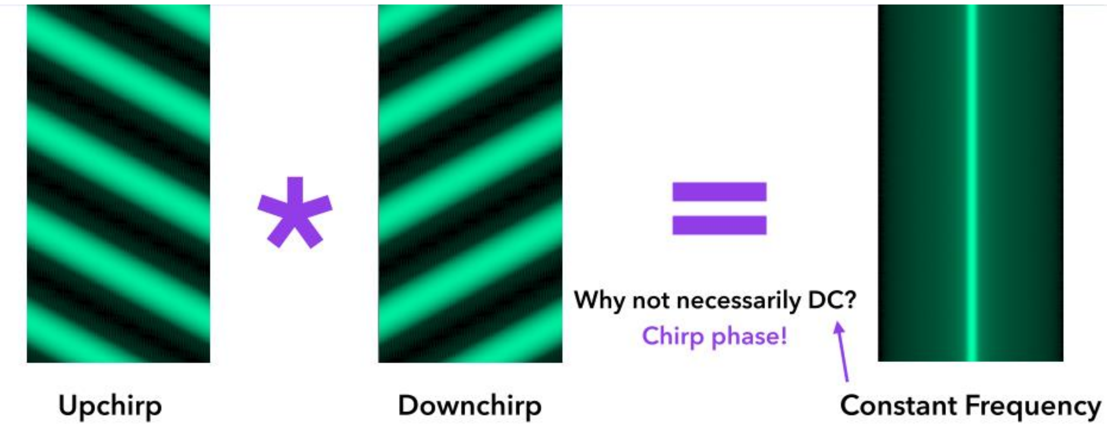


Figure 8. Dechirping [17]

Sensitivity

The noise floor is given by:

$$Noise\ Floor = 10 * \log_{10}(kTB * 1000)\ [dBm] \qquad (1)$$

Where: Noise Floor= equivalent noise power (dBm), k= Boltzmann's Constant (~1.38*10^{-23}),

T= 293 Kelvin ("room temperature"),

B= channel bandwidth (Hz) and

1000= scaling factor from Watts to milli-Watts

It can be simplified as:

$$Noise\ Floor = -174 + 10 * \log_{10} B \qquad (2)$$

The sensitivity of a radio receiver at room temperature is given by:

$$S = -174 + 10\log_{10} BW + NF + SNR \qquad (3)$$

The first term is due the thermal noise in 1 Hz bandwidth, and it can be influenced only by changing the receiver's temperature.

BW is the receiver's bandwidth.

NF is the receiver's Noise Figure.

SNR is the Signal-to-Noise-Ratio required by the modulation scheme used.

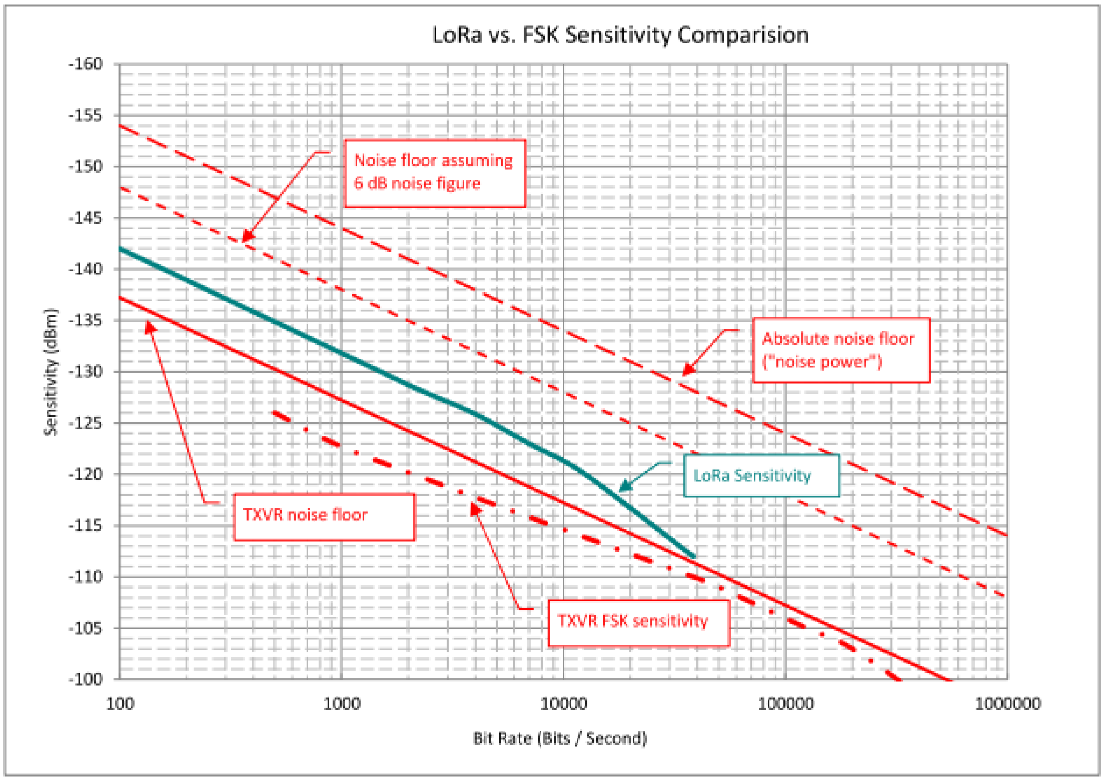


Figure 9. Comparison of LoRa and FSK Sensitivity

SNR and Spreading Factor

By increasing the bandwidth of the signal, the degradation of the SNR can be compensated.

In Spread Spectrum, each bit of information is encoded as multiple chips. This process is shown in figure 7 and is achieved by multiplying the wanted data with a spreading code, known as a chip sequence. The relationship between chip rate and bit rate is given by:

$$R_c = 2^{SF} R_b \qquad (4)$$

Where SF is the Spreading Factor, R_c is the chip rate (Chips per second) and Rb (Bits per second) is the bit rate.

The “amount” of spreading is dependent on the ratio of chips per bit and is referred as processing gain (G_p) and is given by:

$$G_p = 10 \log_{10} \left(\frac{R_c}{R_b} \right) \qquad (5)$$

The processing gain enables the receiver to recover the data signal correctly, even when the SNR value of the channel is negative. The interfering signals are also reduced by the processing gain of the receiver. They are spread beyond the desired information bandwidth and can removed easily with the use of filters [7].

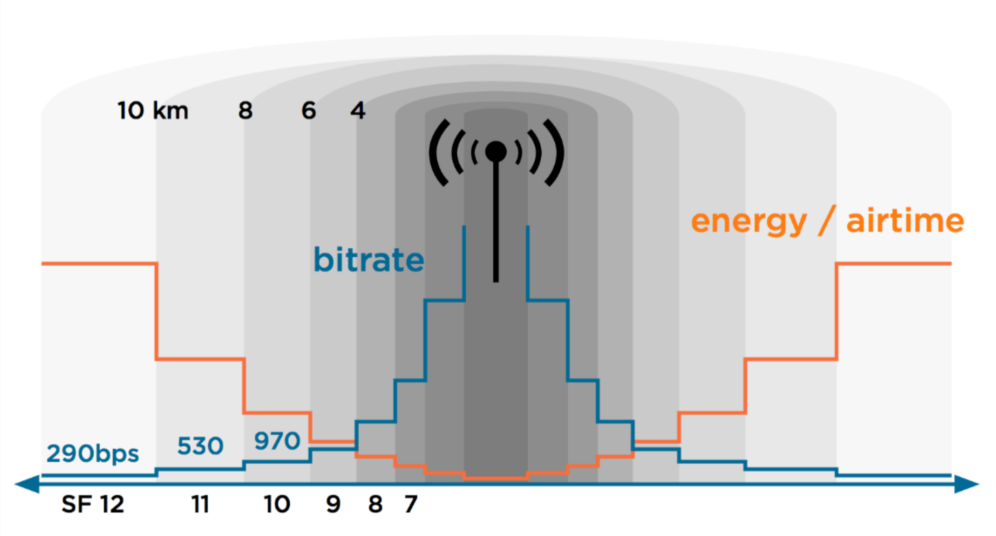


Figure 10. LoRa range, adaptive bit rate and time on air [19]

Data Whitening

The whitening process is built around a 9-bit LSFR which is used to generate a random sequence [12]. In the transmitter, the payload is XORed with this sequence. In the receiver the whitened payload is XORed again with this sequence to retrieve the original payload. Data whitening is used to induce randomness into the symbols to reduce the number of consecutive same bits.

According to [12], the LFSR polynomial that is used is $x^9 + x^5 + 1$ and the CCIT whitening structure provided is shown in fig. 11. [13] found that this sequence is not the one that is actually implemented. The whitening sequence can be found manually by sending a stream of zeros, as xor(whitening sequence, 00000000)= whitening sequence.

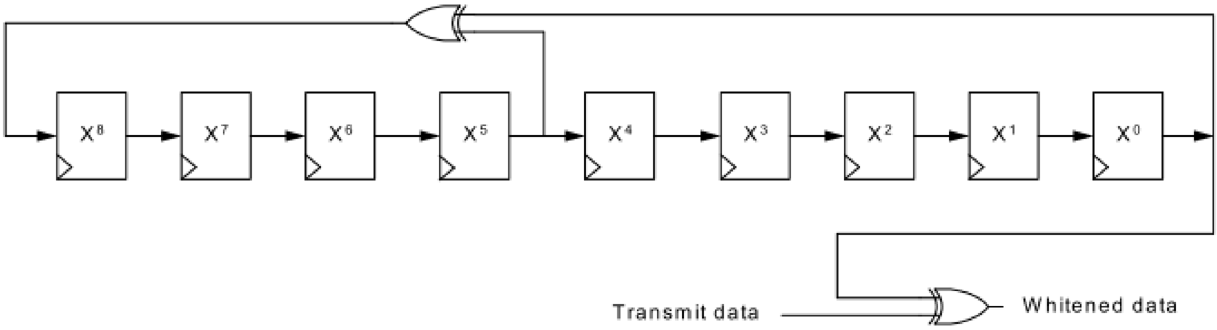


Figure 11. CCIT whitening sequence [12]

Coding Scheme

The LoRa modem employs a form of Forward Error Correction (FEC) that permits the recovery of bits of information due to the corruption by interference. It uses Hamming coding for FEC. It is a simple linear block code algorithm that is easy to implement [6]. LoRa offers code rates of (5,4), (6,4), (7,4) and (8,4) as shown in table 1. To use it, a small overhead of additional encoding of the data, is required to be transmitted. Depending upon the Coding Rate selected, the additional robustness attained in the presence of thermal noise alone is shown in figure 12. Coding rates (5,4) and (6,4) offer 1-bit error detection. Error correction is introduced in coding rate (7,4), which can correct up to 1 bit and (8,4), which can correct 1 bit and detect 2 bit errors. This makes them somewhat counterproductive as they increase the packet length from 25% to 50% without error correction and 75% to 100% to correct 1 bit. Also the number of messages and the time on air are limited, which means fewer acknowledgement and signaling messages. This makes error correction a necessity.

Coding rate	Error detection	Error correction
(5,4)	1 bit	0
(6,4)	1 bit	0
(7,4)	1 bit	1 bit
(8,4)	2 bits	1 bit

Table 1. Coding Rate improvements

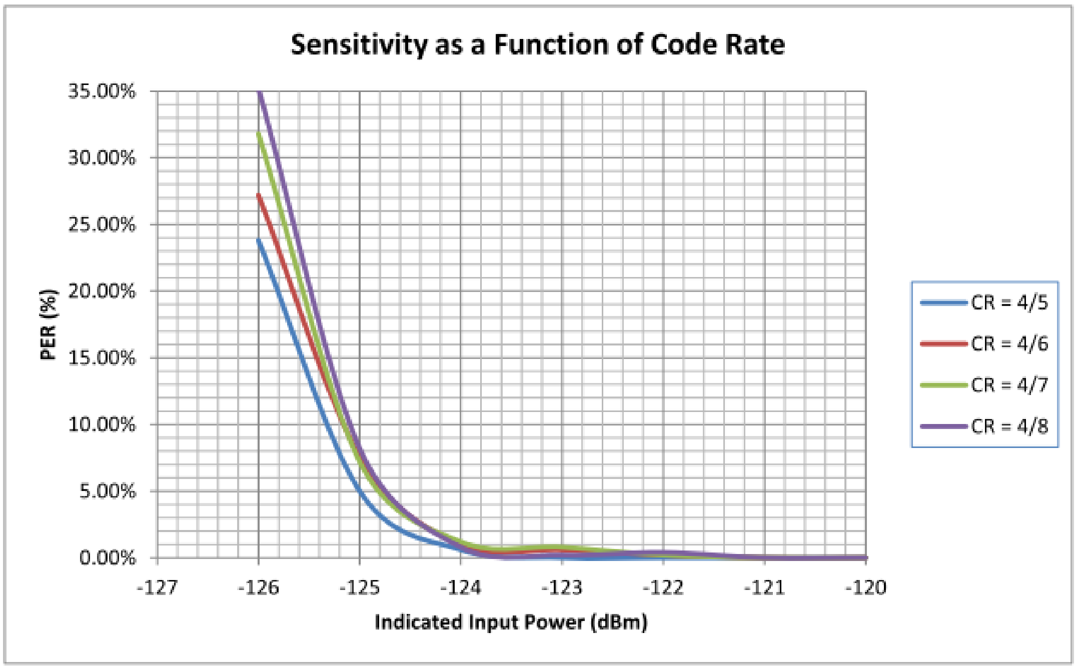


Figure 12. Influence of Coding Rate on Sensitivity [6]

Interleaver

Interleaving is the process that scrambles the bits in a specific way throughout the packet. In LoRa the interleaver scrambles the output of FEC to make the code more resistant against errors. The interleaver is not only present to help with the ±1 position demodulation errors, but also to prevent the errors in case of interference. If interference is present, while the transmission is ongoing, whole symbols can be in error. The Semtech European patent [18], defines a diagonal interleaver.

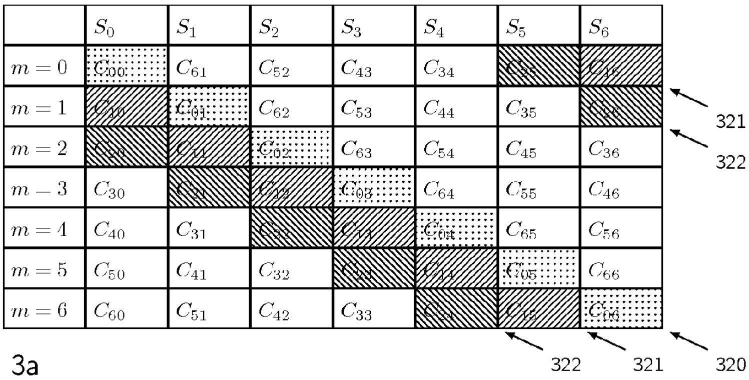
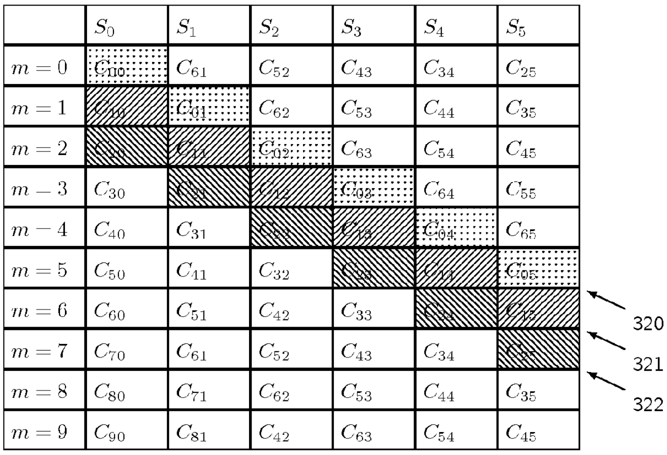

321
322
322
321
320

Fig. 3a

	S_0	S_1	S_2	S_3	S_4	S_5	S_6
$m = 0$	C_{00}	C_{61}	C_{52}	C_{43}	C_{34}	C_{25}	C_{16}
$m = 1$	C_{10}	C_{01}	C_{62}	C_{53}	C_{44}	C_{35}	C_{26}
$m = 2$	C_{20}	C_{11}	C_{02}	C_{63}	C_{54}	C_{45}	C_{36}
$m - 3$	C_{30}	C_{21}	C_{12}	C_{03}	C_{64}	C_{55}	C_{46}
$m = 4$	C_{40}	C_{31}	C_{22}	C_{13}	C_{04}	C_{65}	C_{56}
$m = 5$	C_{50}	C_{41}	C_{32}	C_{23}	C_{14}	C_{05}	C_{66}
$m = 6$	C_{60}	C_{51}	C_{42}	C_{33}	C_{24}	C_{15}	C_{06}

Fig. 3b



	S_0	S_1	S_2	S_3	S_4	S_5
$m = 0$	C_{00}	C_{61}	C_{52}	C_{43}	C_{34}	C_{25}
$m = 1$	C_{10}	C_{01}	C_{62}	C_{53}	C_{44}	C_{35}
$m = 2$	C_{20}	C_{11}	C_{02}	C_{63}	C_{54}	C_{45}
$m - 3$	C_{30}	C_{21}	C_{12}	C_{03}	C_{64}	C_{55}
$m - 4$	C_{40}	C_{31}	C_{22}	C_{13}	C_{04}	C_{65}
$m = 5$	C_{50}	C_{41}	C_{32}	C_{23}	C_{14}	C_{05}
$m = 6$	C_{60}	C_{51}	C_{42}	C_{33}	C_{24}	C_{15}
$m = 7$	C_{70}	C_{61}	C_{52}	C_{43}	C_{34}	C_{25}
$m = 8$	C_{80}	C_{71}	C_{62}	C_{53}	C_{44}	C_{35}
$m = 9$	C_{90}	C_{81}	C_{42}	C_{63}	C_{54}	C_{45}

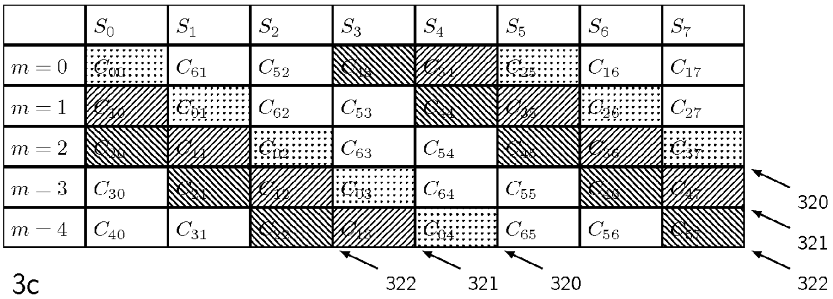

320
321
322
322
321
320

Fig. 3c

	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
$m = 0$	C_{00}	C_{61}	C_{52}	C_{43}	C_{34}	C_{25}	C_{16}	C_{17}
$m = 1$	C_{10}	C_{01}	C_{62}	C_{53}	C_{44}	C_{35}	C_{26}	C_{27}
$m = 2$	C_{20}	C_{11}	C_{02}	C_{63}	C_{54}	C_{45}	C_{36}	C_{37}
$m - 3$	C_{30}	C_{21}	C_{12}	C_{03}	C_{64}	C_{55}	C_{46}	C_{47}
$m - 4$	C_{40}	C_{31}	C_{22}	C_{13}	C_{04}	C_{65}	C_{56}	C_{57}

Figure 13. The interleaver as defined in the patent [18]

According to [23] the algorithm that the patent provides is the one that is implemented. On the other hand, according to [13], the interleaver that is described in the European patent is not the one that is actually implemented. A different interleaver has been deciphered during their blind analysis. The interleaving process is described in tables 2-4.

	Coding Rate								Bits row
Spreading Factor	1	0	0	0	0	1	0	1	1st
	0	0	1	0	1	1	1	0	2nd
	0	0	0	1	1	0	1	0	3rd
	0	0	0	0	1	1	0	1	4th
	0	1	0	0	0	0	1	1	5th
	0	0	1	0	1	1	1	0	6th
	0	0	1	0	1	1	1	0	7th
	1	1	1	1	1	1	1	1	8th

Table 2. Original bits

	Coding Rate								Bits row
Spreading Factor	0	0	1	0	1	1	1	0	2nd
	1	0	0	0	0	1	0	1	1st
	0	0	0	0	1	1	0	1	4th
	0	0	0	1	1	0	1	0	3rd
	0	0	1	0	1	1	1	0	6th
	0	1	0	0	0	0	1	1	5th
	0	0	1	0	1	1	1	0	7th
	1	1	1	1	1	1	1	1	8th

Table 3. First part of interleaver

	Coding Rate								Bits order
Spreading Factor	1	0	0	1	0	1	0	0	2nd
	1	1	1	0	1	1	0	1	1st
	0	1	0	1	0	1	0	0	4th
	0	1	1	1	0	0	0	0	3rd
	1	1	1	1	0	0	0	0	6th
	1	1	1	0	0	0	0	0	5th
	0	0	1	1	0	0	1	0	7th
	0	1	1	1	1	1	1	1	8th

Table 4. The interleaver as described in [13]

Gray Indexing

The encoded LoRa symbols are gray indexed before they are transmitted, which actually refers to the inverse gray coding. It is used to make sure that two adjacent symbols differ by one bit, which increases the probability that decoder will correct the possible bit errors.

Data rates

The modulation bit rate is defined as:

$$R_b = SF * \frac{BW}{2^{SF}} \text{, where} \qquad (6)$$

SF is the spreading factor (7…12) and

BW is the bandwidth (Hz)

The period of the symbols is defined as:

$$T_s = \frac{2^{SF}}{BW} \tag{7}$$

The symbol rate is defined as:

$$R_s = \frac{1}{T_s} = \frac{BW}{2^{SF}} \tag{8}$$

The chirp rate is defined as

$$R_c = R_s * 2^{SF} = \frac{BW}{2^{SF}} * 2^{SF} = BW \tag{9}$$

The nominal bit rate that includes the error correction scheme is defined as:

$$R_b = SF * \frac{\left(\frac{4}{4+CR}\right)}{\left(\frac{2^{SF}}{BW}\right)} \text{, where} \tag{10}$$

CR is the coding rate (1…4) and the rate code is $\frac{4}{4+CR}$

BW [KHz]	SF	Ts [ms]	Rbit [bps]	Rsymbol [Hz]
125	7	1.024	6835.94	976.562
125	8	2.048	3906.25	488.28
125	9	4.096	2197.26	244.14
125	10	8.192	1220.7	122.07
125	11	16.384	671.38	61.035
125	12	32.768	366.21	30.51
250	7	0.512	13671.9	1953.1
250	8	1.024	7812.5	976.56
250	9	2.048	4394.5	488.28
250	10	4.096	2441.4	244.14
250	11	8.192	1342.8	122.07
250	12	16.384	732.42	61.03

Table 5. Some LoRaWAN data rates

Packet Format

There are two modes for the LoRa packet format. One is the explicit mode, where the packet is composed of the preamble, the header (with its CRC), the payload and the CRC. The other is the implicit mode and does not include the header.

The number of preamble symbols can be programmed in the transmitter.

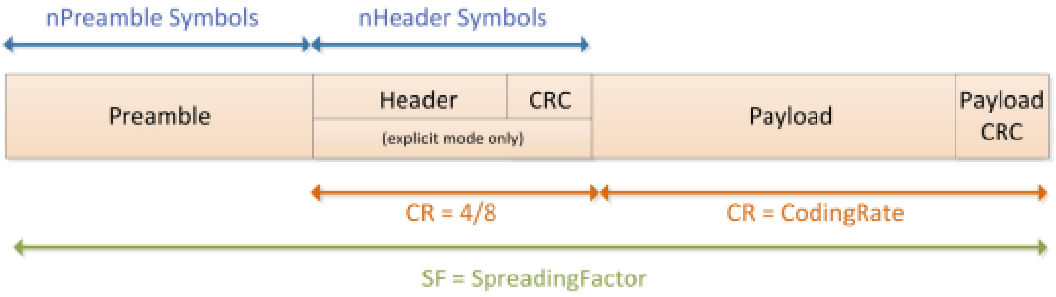


Figure 14. LoRa modem packet formatting [6]

Preamble

The preamble is used to synchronize the receiver with the incoming signal. It is composed Its length is configurable and can be set from 8 to 65539 symbols. The minimum length of 8 symbols is enough for all communications.

Header

As mentioned earlier, there are two modes for the header. The explicit, that includes it and the implicit that does not.

In the explicit mode the header contains information about the number of bytes in the payload, the coding rate and whether the optional 16-bit CRC is used or not. It is transmitted with an error correction code of 4/8 coding rate and has its own CRC.

In the implicit mode, the size of the payload is calculated from the size of the packet, while the coding rate and the CRC have to be set identically on the transmitter and the receiver.

Payload

The payload contains the actual data, that is transmitted over the air. This data is coded as specified earlier.

Packet on-air time

The total on-air time of a packet can be calculated by:

$$t_{packet} = t_{preamble} + t_{payload} \, , \tag{11}$$

where t$_{preamble}$ is the time it takes to transmit the preamble and t$_{payload}$ is the time it takes to transmit the payload.

$$t_{preamble} = (n_{preamble} + 4.25)\&t_s \ , \tag{12}$$

$$t_{payload} = n_{payload} * t_s, \tag{13}$$

$$t_s = \frac{2^{SF}}{BW} \ , \tag{14}$$

where n$_{preamble}$ is the configurable parameter that affects the length of the preamble,

4.25 (2 extra preamble symbols+ 2.25 symbols of Start of Frame Delimiter-SFD) is the number of symbols needed to reach the start of the payload,

n$_{payload}$ is the number of payload symbols and

t$_s$ is the duration of each symbol.

Start of Frame Delimiter (SFD) is two consecutive downchirp symbols that signify the end of the preamble symbols and the start of the payload as shown in figure 15.

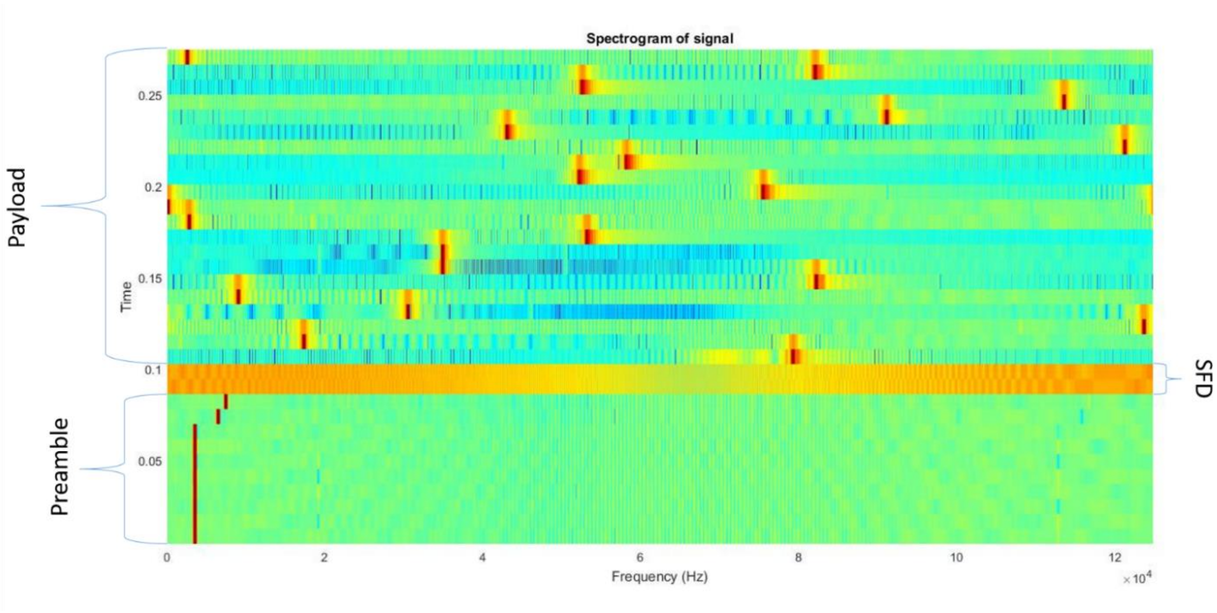


Figure 15. The preamble, the SFD and the payload as they appear in the spectrogram

The number of symbols that make up the packet is given by:

$$payloadSymbNb = 8 + max\left(ceil\left(\frac{8PL-4SF+28+16-20H}{4(SF-2DE)}\right)(CR+4), 0\right), \qquad (15)$$

where

PL is the number of payload bytes

SF is the Spreading Factor

H= 0 when the explicit mode is used and 1 when the implicit mode is used

DE= 1 when the low data rate optimization is enabled and 0 when it is disabled

CR is the coding rate from 1 to 4 (CR+4,4)

3.1.3 Network Infrastructure

LoRaWAN typically uses the star-of-stars (tree) topology, in which gateways is a transparent bridge that relay messages between end-devices and a central network server [14]. Gateways are connected to the network server via IP connections. In LoRaWAN, the end devices are not associated with a specific gateway and the transmitted signals are received by multiple gateways. As a result, each gateway will forward the received packet to the network server. When a gateway receives a frame, it checks the Physical CRC, and then forwards it.

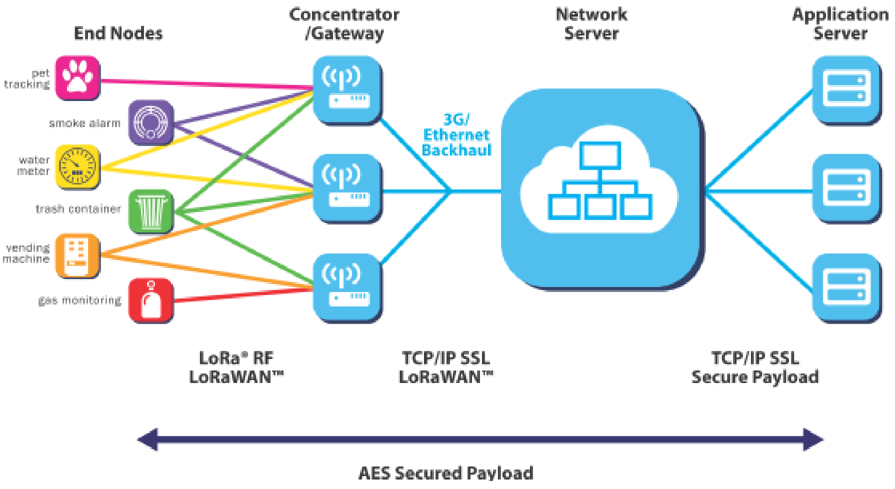


Figure 16. LoRaWAN architecture [15]

3.2 Software Defined Radio Technology

Software Defined Radio (SDR) is a radio communication technology that is based on software defined wireless communications protocols instead of hardwired implementations [9]. Components that have typically been implemented in hardware, such as filters, amplifiers, modulators and demodulators can now be implemented in software.

3.2.1 Hardware

A basic SDR must include the radio front-end, the modem, the cryptographic security function and the application function. Some radios will include support fornetwork devices, that allow the radio to provide network services and to be controlled remotely over Ethernet. Some radios provide for control of external radio frequency (RF) analog functions such as antennas, power amplifiers, coax switches and special-purpose filters.

To support the receive mode, the RF front-end consists of antenna matching unit, low noise amplifier (LNA), filters, local oscillators, and analog-to-digital converters (ADCs), to capture the desired signal and suppress the undesired signals to an extent.

To support the transmit mode, the RF front-end includes digital-to-analog converters (DACs), antenna matching units, local oscillators, filters and power amplifiers. The important property of these circuits, in transmit mode, is to synthesize the RF signal without introducing noise and false emissions at other frequencies that can interfere with other users in the spectrum [10].

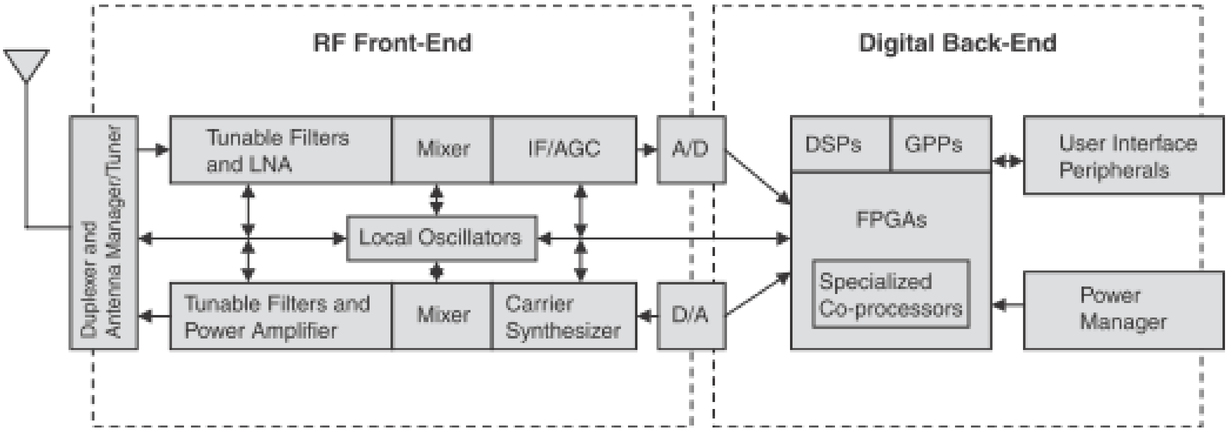


Figure 17. Basic hardware architecture of a modem SDR [10]

3.2.2 Software

The objective of the software architecture in an SDR is to place waveforms and applications onto a software-based radio platform in a standardized way. These waveforms and applications are installed, used and replaced by other applications as required by the user to achieve their objectives. To make the waveforms and application interfaces standardized, it is necessary to make the hardware platform present a set of highly standardized interfaces. This way, the vendors can develop their waveforms, without knowledge of the underlying hardware. The hardware developers can develop a radio with standardized interfaces [10]. The SDR device manufacturers typically provide drivers, APIs and wrappers for standard programming languages such as C, C++ and Python, as well as for development platforms such as MATLAB and GNU Radio.

4 Implementation of a LoRa Experimental Setup using Commercial Hardware

4.1 Hardware and Software Analysis

The LoPy is a triple network development board. It supports LoRa, Wifi and Bluetooth and can act as both a LoRa Nano Gateway and a multi-bearer development board. It can function as a simple node, that has a maximum range of 40 km, and as a gateway that can support up to 100 nodes.

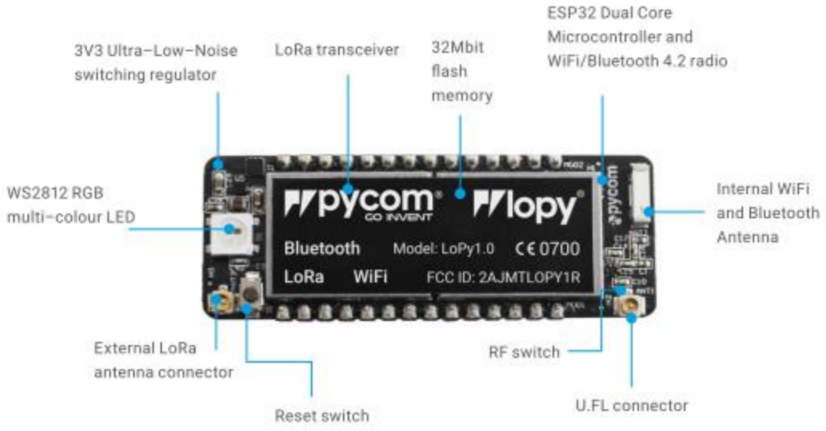
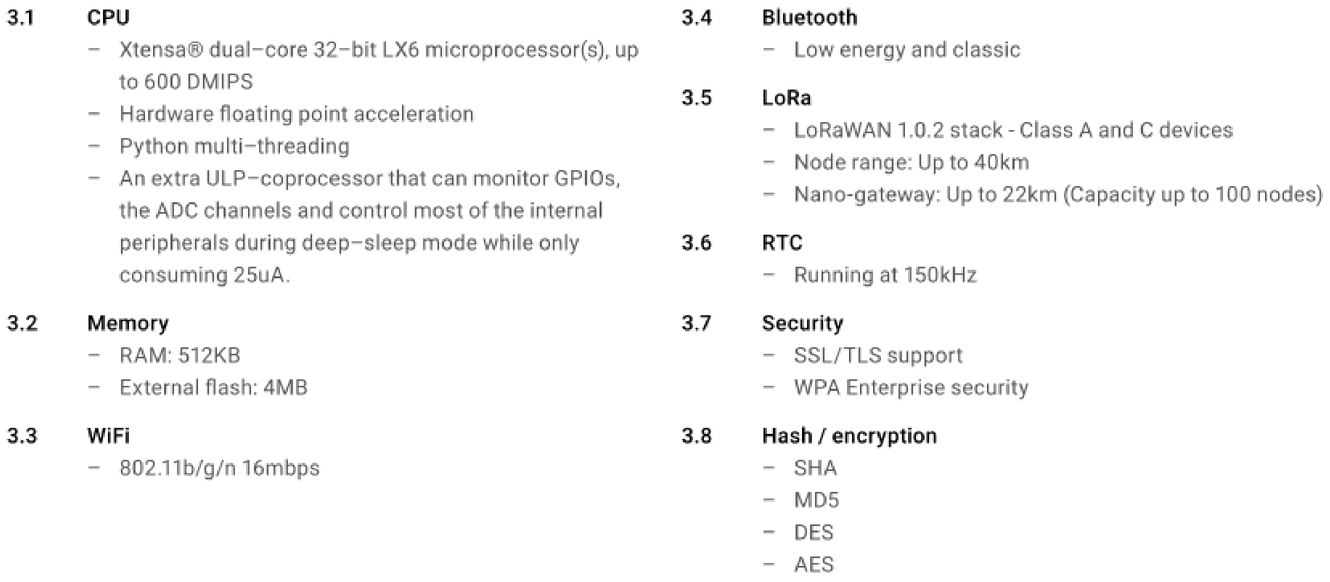

3V3 Ultra–Low–Noise switching regulator
LoRa transceiver
32Mbit flash memory
ESP32 Dual Core Microcontroller and WiFi/Bluetooth 4.2 radio
WS2812 RGB multi–colour LED
Internal WiFi and Bluetooth Antenna
pycom GO INVENT
lopy
Bluetooth
LoRa
WiFi
Model: LoPy1.0
CE 0700
FCC ID: 2AJMTLOPY1R
External LoRa antenna connector
RF switch
Reset switch
U.FL connector

Figure 18. Pycom LoPy [22]



3.1 **CPU**

- Xtensa® dual–core 32–bit LX6 microprocessor(s), up to 600 DMIPS
- Hardware floating point acceleration
- Python multi–threading
- An extra ULP–coprocessor that can monitor GPIOs, the ADC channels and control most of the internal peripherals during deep–sleep mode while only consuming 25uA.

3.2 **Memory**

- RAM: 512KB
- External flash: 4MB

3.3 **WiFi**

- 802.11b/g/n 16mbps

3.4 **Bluetooth**

- Low energy and classic

3.5 **LoRa**

- LoRaWAN 1.0.2 stack - Class A and C devices
- Node range: Up to 40km
- Nano-gateway: Up to 22km (Capacity up to 100 nodes)

3.6 **RTC**

- Running at 150kHz

3.7 **Security**

- SSL/TLS support
- WPA Enterprise security

3.8 **Hash / encryption**

- SHA
- MD5
- DES
- AES

Figure 19. LoPy specifications [22]

LoPy is programmable using MicroPython. MicroPython is an implementation of the Python 3 programming language that is optimized to run on microcontrollers and in constrained environments.

The “main” file in LoPy, is the one that contains the instructions that LoPy follows. It can be updated using a usb cable or by connecting to its network with wifi. Then there are two options. The first is to connect to the telnet server that is running on port 23 and the other is to connect to the FTP server, which is un-encrypted with the credentials:

User: micro

Password: python

4.2 Implementation

LoPy, as shown in the code below, allows the user to configure the transmitter to their needs. To do this, the first step is to import the libraries. After that the user can optionally choose to make it connect to wifi, and to adjust the time by connecting to a server like “pool.ntp.org”.

The connection to wifi allows the LoPy to connect to the Internet. This connection enables LoPy to adjust the time and upload the received data to a Cloud platform.

The time adjustment allows the transmitter to keep track of time and, by extension the transmission at specific times.

The next step is to choose what configurations the system will implement. The configurations include:

- the mode that LoPy will implement (LoRa or LoRaWAN),
- the center frequency (863-870 MHz in the 868 band, or 902-928 MHz in the 915 band),
- the region (AS923, AU915, EU868 and US915),
- the transmitting power (2 and 14 dBm for the 868 band or 5 and 20 dBm in the 915 band),
- the spreading factor (7-12),
- the coding rate (4/5-4/8),
- the bandwidth (125 and 250 KHz in the 868 band or 250 and 500 KHz in the 915 band), and
- the number of preamble symbols (default is 8 symbols).

The setup for the experiment is shown in table 6. The message “Ping Pong” is transmitted every one second. This is specified in the “time.sleep(1)” command, which means that LoPy will enter a sleep mode for 1 second before transmitting the message again. When the message is transmitted, the terminal returns the message “Sent packet”.

Mode	LoRa
Center frequency	868.1 MHz
Region	-
Bandwidth	125 KHz
Transmitting power	-
Spreading factor	7
Coding rate	4/7
Preamble symbols	8

Table 6. Setup for experiment

Code for transmitter

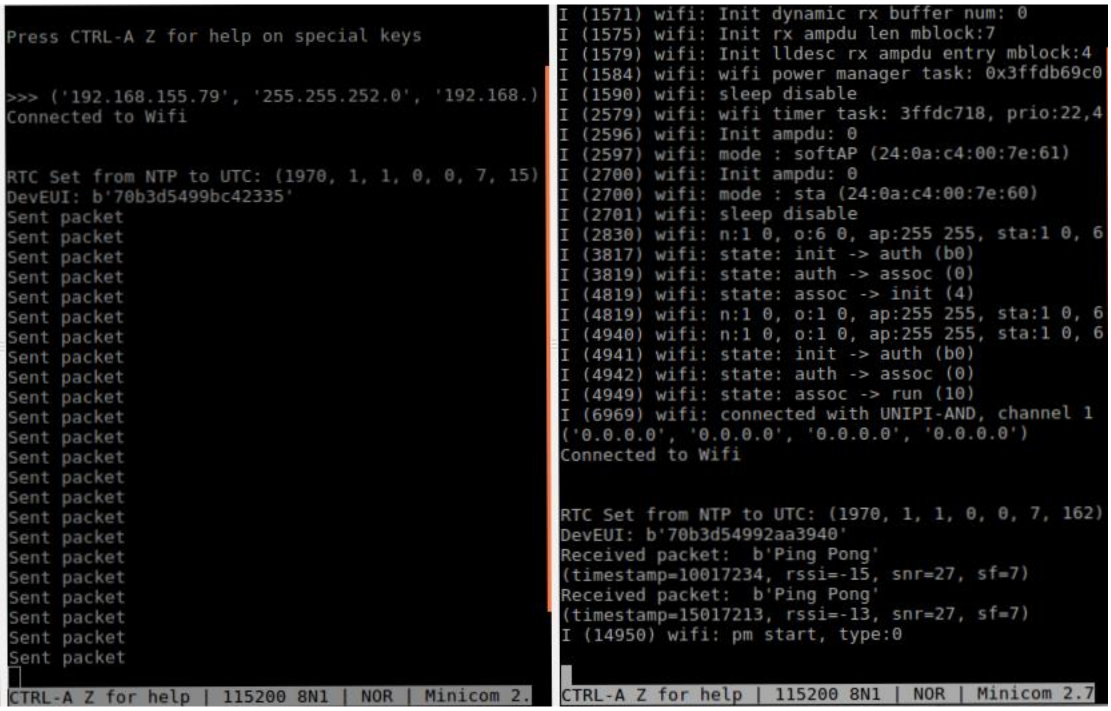
```
import machine
from network import WLAN
from network import LoRa
import socket
import time
import utime
import binascii

# connect to wlan
wlan = WLAN(mode=WLAN.STA)
wlan.ifconfig(id=0, config='dhcp')
wlan.connect(ssid='name', auth=(WLAN.WPA2, 'password'), timeout=5000)
time.sleep_ms(5000)
print(wlan.ifconfig())
print("Connected to Wifi\n")

# adjust time
rtc = machine.RTC()
rtc.ntp_sync("pool.ntp.org")
utime.sleep_ms(1000)
print('\nRTC Set from NTP to UTC:', rtc.now())

# setup lora
lora = LoRa(mode=LoRa.LORA, frequency=868100000,
bandwidth=LoRa.BW_125KHZ, sf=7, preamble=8,
coding_rate=LoRa.CODING_4_7)
s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
s.setblocking(False)
# get device id
DevEUI = binascii.hexlify(LoRa().mac())
print("DevEUI:",DevEUI)

# tx
while True:
    s.send('Ping Pong')
    print('Sent packet')
    time.sleep(1)
```

The receiver has been configured with only the mode, the center frequency and the spreading factor. At the very least these three must be setup and of course be the same as in the transmitter.

Code for receiver

```
import machine
from network import WLAN
from network import LoRa
import socket
import time
import utime
import binascii

# connect to wlan
wlan = WLAN(mode=WLAN.STA)
wlan.ifconfig(id=0, config='dhcp')
wlan.connect(ssid='name', auth=(WLAN.WPA2, 'password'), timeout=5000)
time.sleep_ms(5000)
print(wlan.ifconfig())
print("Connected to Wifi\n")

# adjust time
rtc = machine.RTC()
rtc.ntp_sync("pool.ntp.org")
utime.sleep_ms(1000)
print('\nRTC Set from NTP to UTC:', rtc.now())

# setup lora
lora = LoRa(mode=LoRa.LORA, frequency=868100000, sf=7)
s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
s.setblocking(False)
# get device id
DevEUI = binascii.hexlify(LoRa().mac())
print("DevEUI:",DevEUI)

# start receiving
while True:
    data = s.recv(64)
    if len(data) > 0:
        print('Received packet: ', data)
        print(lora.stats())
        time.sleep(5)

#if s.recv(64) == b'Ping':
```

The terminal of the transmitter returns the message “Sent packet”, as programmed. The receiver returns the stats: timestamp, Received Signal Strength Indicator (RSSI) in dBm, the SNR value and of course the received message in dB.



```
Press CTRL-A Z for help on special keys


>>> ('192.168.155.79', '255.255.252.0', '192.168.)
Connected to Wifi


RTC Set from NTP to UTC: (1970, 1, 1, 0, 0, 7, 15)
DevEUI: b'70b3d5499bc42335'
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet
Sent packet

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.
```

```
I (1571) wifi: Init dynamic rx buffer num: 0
I (1575) wifi: Init rx ampdu len mblock:7
I (1579) wifi: Init lldesc rx ampdu entry mblock:4
I (1584) wifi: wifi power manager task: 0x3ffdb69c0
I (1590) wifi: sleep disable
I (2579) wifi: wifi timer task: 3ffdc718, prio:22,4
I (2596) wifi: Init ampdu: 0
I (2597) wifi: mode : softAP (24:0a:c4:00:7e:61)
I (2700) wifi: Init ampdu: 0
I (2700) wifi: mode : sta (24:0a:c4:00:7e:60)
I (2701) wifi: sleep disable
I (2830) wifi: n:1 0, o:6 0, ap:255 255, sta:1 0, 6
I (3817) wifi: state: init -> auth (b0)
I (3819) wifi: state: auth -> assoc (0)
I (4819) wifi: state: assoc -> init (4)
I (4819) wifi: n:1 0, o:1 0, ap:255 255, sta:1 0, 6
I (4940) wifi: n:1 0, o:1 0, ap:255 255, sta:1 0, 6
I (4941) wifi: state: init -> auth (b0)
I (4942) wifi: state: auth -> assoc (0)
I (4949) wifi: state: assoc -> run (10)
I (6969) wifi: connected with UNIPI-AND, channel 1
('0.0.0.0', '0.0.0.0', '0.0.0.0', '0.0.0.0')
Connected to Wifi


RTC Set from NTP to UTC: (1970, 1, 1, 0, 0, 7, 162)
DevEUI: b'70b3d54992aa3940'
Received packet:  b'Ping Pong'
(timestamp=10017234, rssi=-15, snr=27, sf=7)
Received packet:  b'Ping Pong'
(timestamp=15017213, rssi=-13, snr=27, sf=7)
I (14950) wifi: pm start, type:0


CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7
```

Figure 20. Left is the terminal of the transmitter and right is the terminal of the receiver

5 Implementation of LoRa Experimental Setup based on SDR technology and gr-lora open source solution

In this chapter an implementation of LoRa from the gr-lora project in [21] is presented. “The gr-lora project aims to provide GNU-Radio blocks for receiving LoRa modulated radio messages using a SDR” [21]. Gr-lora supports most of the features of LoRa except (1) the CRC checks of the header and payload and (2) the decoding of multiple channels simultaneously.

5.1 Hardware and Software Analysis

In the experiment two pieces of hardware are used. The transmitter is the LoPy introduced in the previous chapter and the receiver is the RTL-SDR RTL2832U in fig.

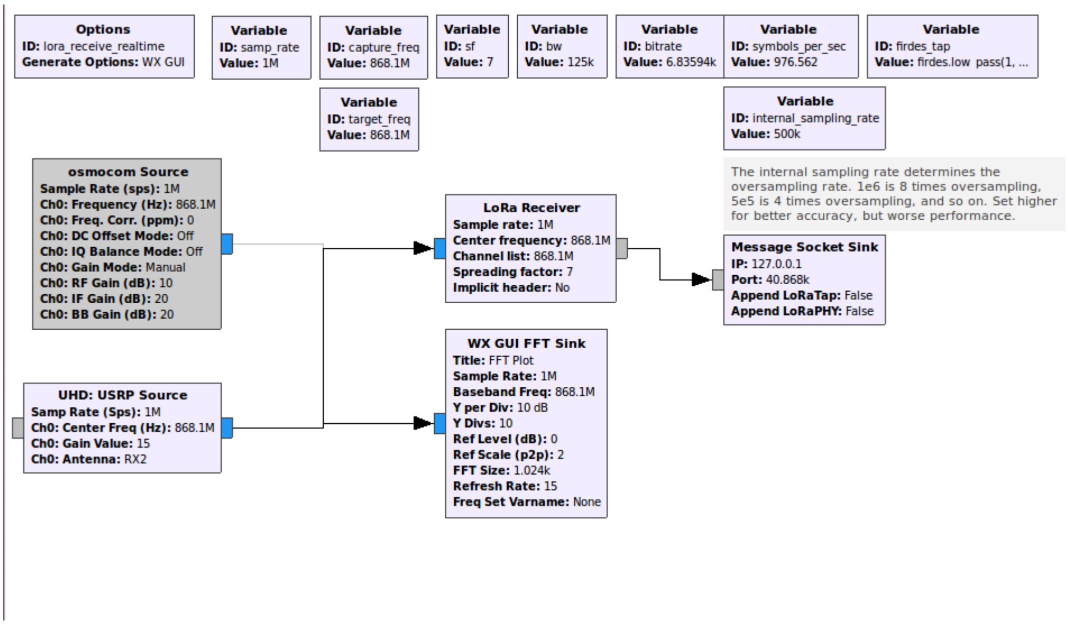
21. “The RTL2832U is a DVB-T Coded Orthogonal Frequency Division Multiplexing (COFDM) demodulator that supports a USB 2.0 interface. It supports 2K (1705 carriers) and 8K (6817 carriers) transmission modes with 6,7 and 8 MHz bandwidth. It also supports tuners at Intermediate Frequency (IF 36.125 MHz), low-IF (4.57 MHz), or Zero-IF output, using a 28.8 MHz crystal. It includes FM/DAB/DAB+ radio support and is has an integrated advanced Analog-to-Digital Converter. It also features propriety algorithms of Realtek, which include channel estimation, co-channel interface rejection, long echo channel reception and impulse noise cancellation, and provides a solution for a wide range of applications for PC-TV” [26].

Gr-lora is for the most part written in python, with some parts written in C++.

RTL-SDR.COM
QUICKSTART SETUP GUIDE: RTL-SDR.COM/QSG
DVB-T+DAB+FM+SDR
RTL2832U R820T2 TCXO+BIAS T+HF
V.3

Figure 21. RTL-SDR RTL2832U

The configuration for the receiver is shown in fig. 22. The input of gr-lora is the signal, the capture frequency, the bandwidth, the spreading factor and the header mode used in the transmitter. The received packets in gr-lora are in hexadecimal format and need to be converted in decimal and then in ASCII.



Options
ID: lora_receive_realtime
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 1M

Variable
ID: capture_freq
Value: 868.1M

Variable
ID: sf
Value: 7

Variable
ID: bw
Value: 125k

Variable
ID: bitrate
Value: 6.83594k

Variable
ID: symbols_per_sec
Value: 976.562

Variable
ID: firdes_tap
Value: firdes.low pass(1, ...

Variable
ID: target_freq
Value: 868.1M

Variable
ID: internal_sampling_rate
Value: 500k

The internal sampling rate determines the oversampling rate. 1e6 is 8 times oversampling, 5e5 is 4 times oversampling, and so on. Set higher for better accuracy, but worse performance.

osmocom Source
Sample Rate (sps): 1M
Ch0: Frequency (Hz): 868.1M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

LoRa Receiver
Sample rate: 1M
Center frequency: 868.1M
Channel list: 868.1M
Spreading factor: 7
Implicit header: No

Message Socket Sink
IP: 127.0.0.1
Port: 40.868k
Append LoRaTap: False
Append LoRaPHY: False

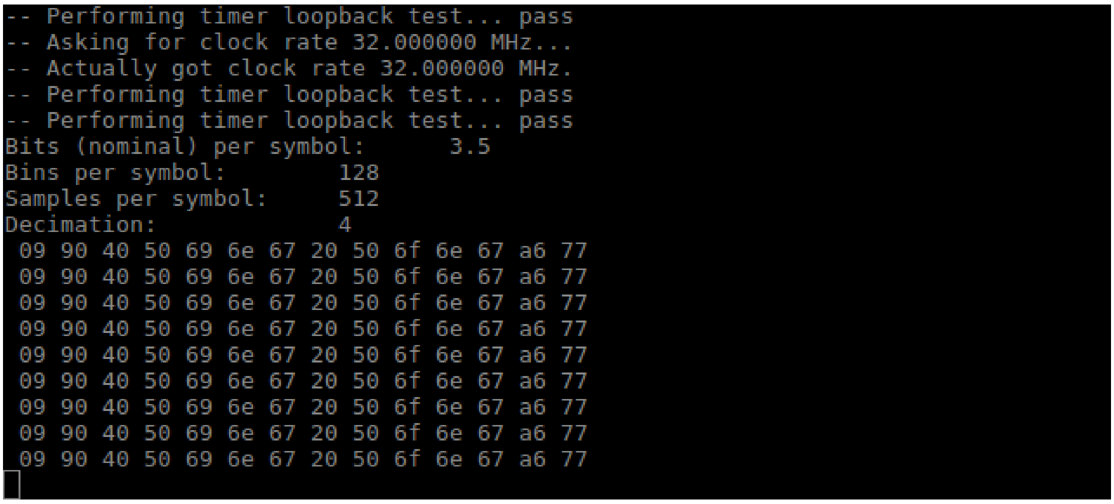
WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 1M
Baseband Freq: 868.1M
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Freq Set Varname: None

UHD: USRP Source
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): 868.1M
Ch0: Gain Value: 15
Ch0: Antenna: RX2

Figure 22. gr-lora implementation

5.2 Implementation

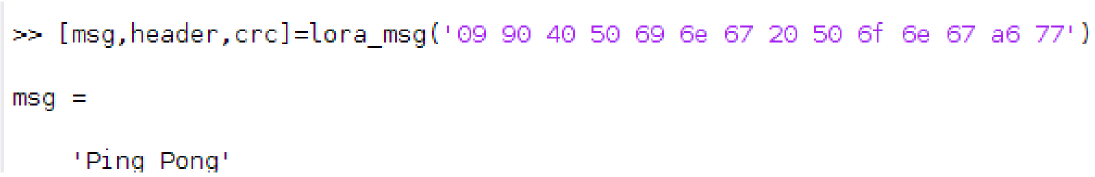
The code for the transmitter remains the same in this chapter. As shown in figures 23 and 24, the receiver functions correctly, as the received message is exactly the same as the transmitted one.



```
-- Performing timer loopback test... pass
-- Asking for clock rate 32.000000 MHz...
-- Actually got clock rate 32.000000 MHz.
-- Performing timer loopback test... pass
-- Performing timer loopback test... pass
Bits (nominal) per symbol:      3.5
Bins per symbol:         128
Samples per symbol:      512
Decimation:              4
 09 90 40 50 69 6e 67 20 50 6f 6e 67 a6 77
 09 90 40 50 69 6e 67 20 50 6f 6e 67 a6 77
 09 90 40 50 69 6e 67 20 50 6f 6e 67 a6 77
 09 90 40 50 69 6e 67 20 50 6f 6e 67 a6 77
 09 90 40 50 69 6e 67 20 50 6f 6e 67 a6 77
 09 90 40 50 69 6e 67 20 50 6f 6e 67 a6 77
 09 90 40 50 69 6e 67 20 50 6f 6e 67 a6 77
 09 90 40 50 69 6e 67 20 50 6f 6e 67 a6 77
 09 90 40 50 69 6e 67 20 50 6f 6e 67 a6 77
```

Figure 23. Received packets

The received message in figure 23 is in hexadecimal format and needs to be converted.



```
>> [msg,header,crc]=lora_msg('09 90 40 50 69 6e 67 20 50 6f 6e 67 a6 77')

msg =

    'Ping Pong'
```

Figure 24. Hexadecimal to character conversion

The received signal after the conversion is identical to the one sent.

6 Implementation of LoRa receiver using MATLAB

In this chapter an attempt to implement a receiver for LoRa. The difficulty in this, is that LoRa is not open source. There are some implementations that explain the method to create a receiver but they disagree on the block diagram and the interleaver. The block diagram according to [23] is shown in figure 25. The modulation function includes gray indexing. This implantation includes the first seven steps in figure 25.

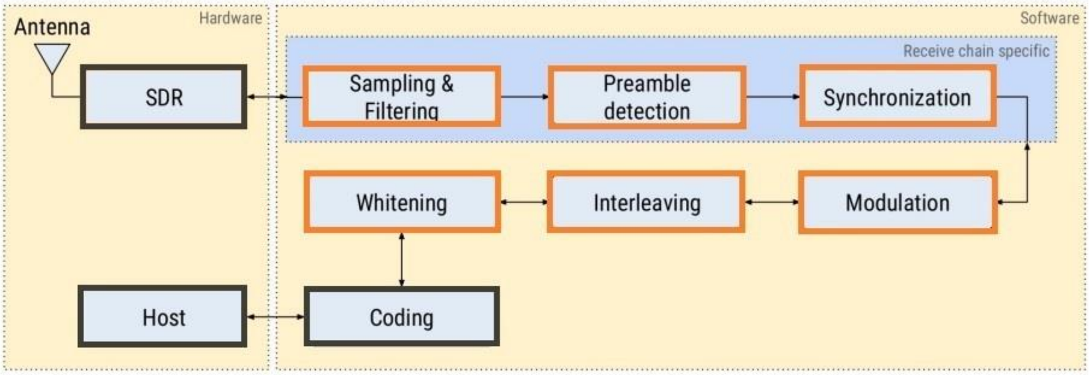


Figure 25. LoRa block diagram [23]

In [13] and [27] there is a difference in the order of the functions. In [13] the order (in the receiver) is:

1. Gray indexing
2. Whitening
3. Interleaving
4. Forward Error Correction

In [27] the order is:

1. Gray indexing
2. Interleaving
3. Forward Error Correction
4. Whitening

The input signal has the below parameters:

Sampling Rate= 1 GHz
Spreading Factor= 10
Bandwidth= 125 KHz;
Center frequency = 868.1 MHz
Center frequency with offset= 869 MHz
Symbol time = $\frac{2^{SF}}{BW} = 0{,}008192$ sec

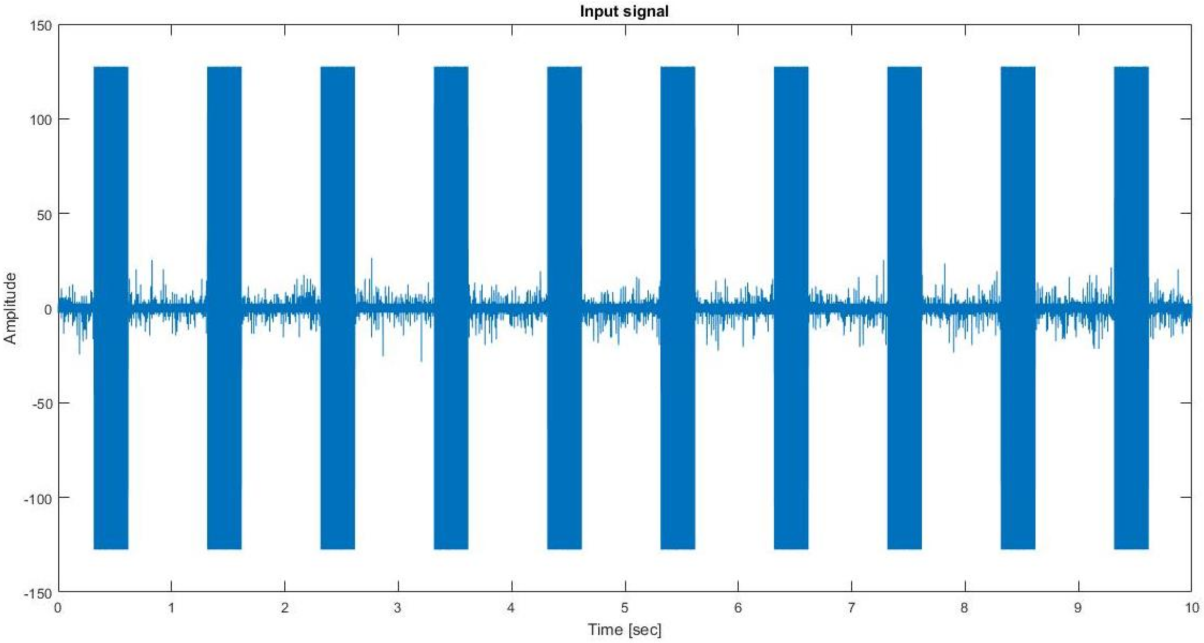


Figure 26. Input signal

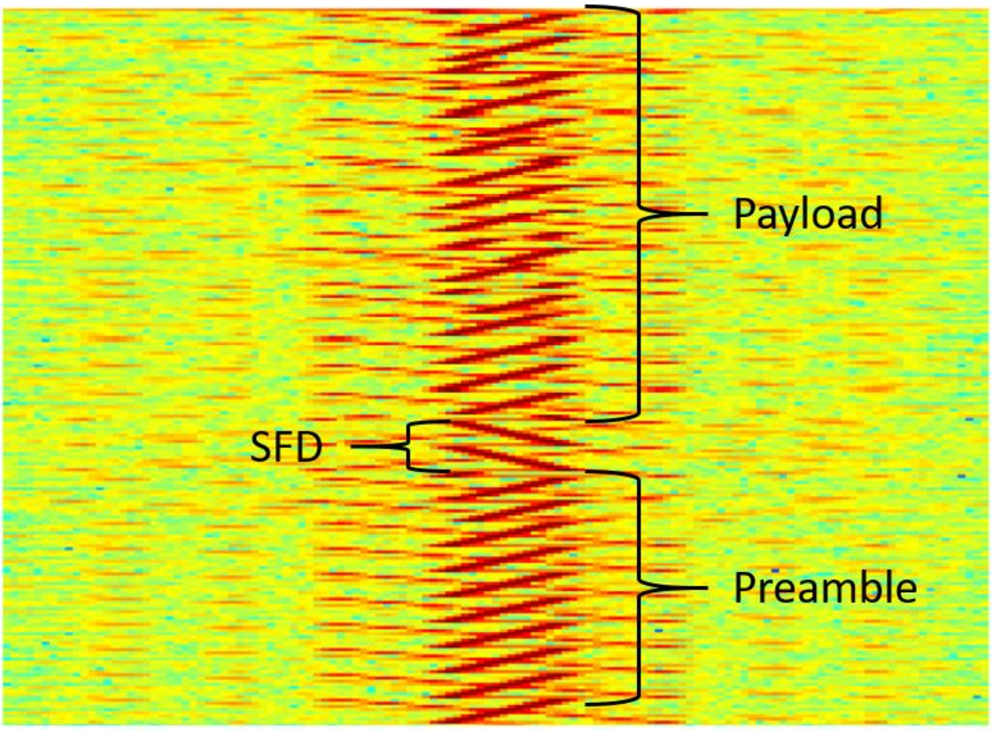


Figure 27. Spectrogram of the wanted signal

The next step is to find the start and the end of the signal. This is accomplished with the use of an energy detector.

Energy Detector

```
% energy of noise.
% The end of the signal is where the the received energy is lower than the
% energy of the signal.
% If the start of the signal is in the middle of a signal,
% it searches for the next signal

start_index= find(abs(x).^2>0.5);
start_index= start_index(1);
xx= x(start_index:end);
if start_index== 1
    start_index= find(abs(xx).^2<0.05);
    start_index= start_index(1);
    xx= xx(start_index:end);
    start_index= find(abs(xx).^2>0.5);
    start_index= start_index(1);
    xx= xx(start_index:end);
end
end_index= find(abs(xx).^2<0.05);
end_index= end_index(2);
xx= xx(1:end_index);
```

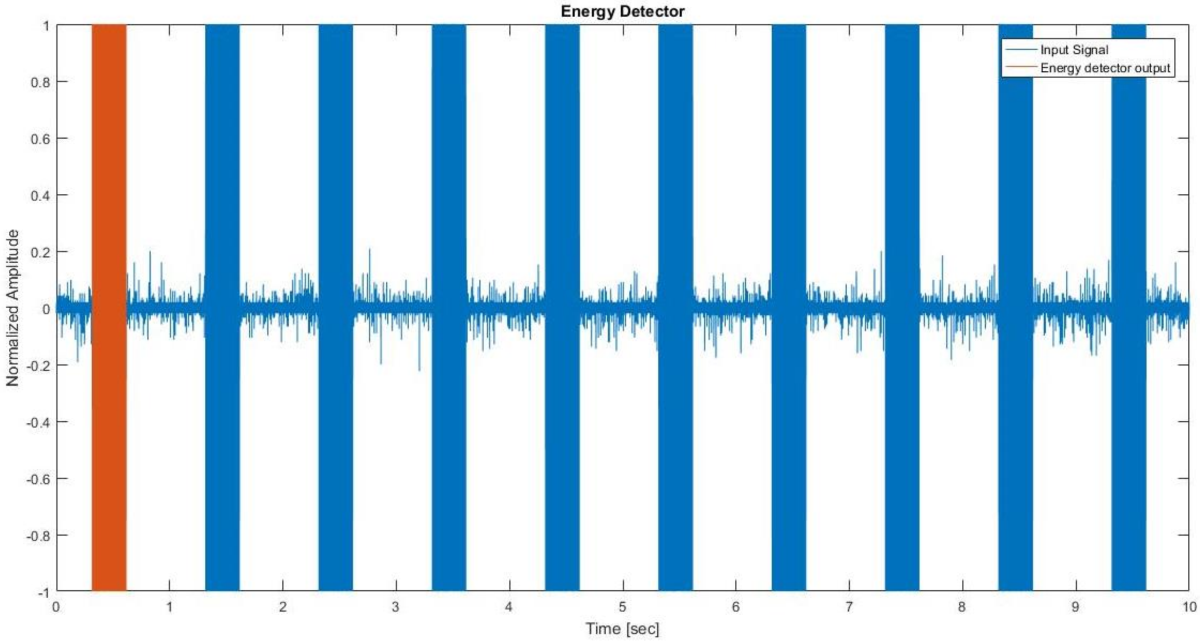


Figure 28. Energy Detector

To demodulate the signal, it must be first de-chirped. To begin de-chirping the signal, it must be channelized to its bandwidth.

Channelization

```
t= (1:length(xx))./Fs;

xx = xx.*exp(-1i*2*pi*Fc*t); %  the new signal is baseband

% Resampling
xx = resample(xx, BW, Fs); % Output sampling frequency is BW
Fs = BW;
```

Chirp generation

```
f0 = -BW/2; % starting frequency of chirp
f1 = BW/2; % end frequency of chirp
t = 0:1/Fs:symbol_time - 1/Fs;

% The chirps are complex signals
chirpI = chirp(t, f0, symbol_time, f1, 'linear', 90);
chirpQ = chirp(t, f0, symbol_time, f1, 'linear', 0);
upChirp = complex(chirpI, chirpQ);

% downchirp is the conjugate of upchirp
downChirp= conj(upChirp);
```

The MATLAB function chirp(t,f0,t1,f1,'method') creates a linear swept frequency cosine, at the time defined in t, where f0 is the starting frequency and f0 is the ending frequency. t1 is the duration of the symbol. The method specifies the sweep option. In this case the linear method is used [28].

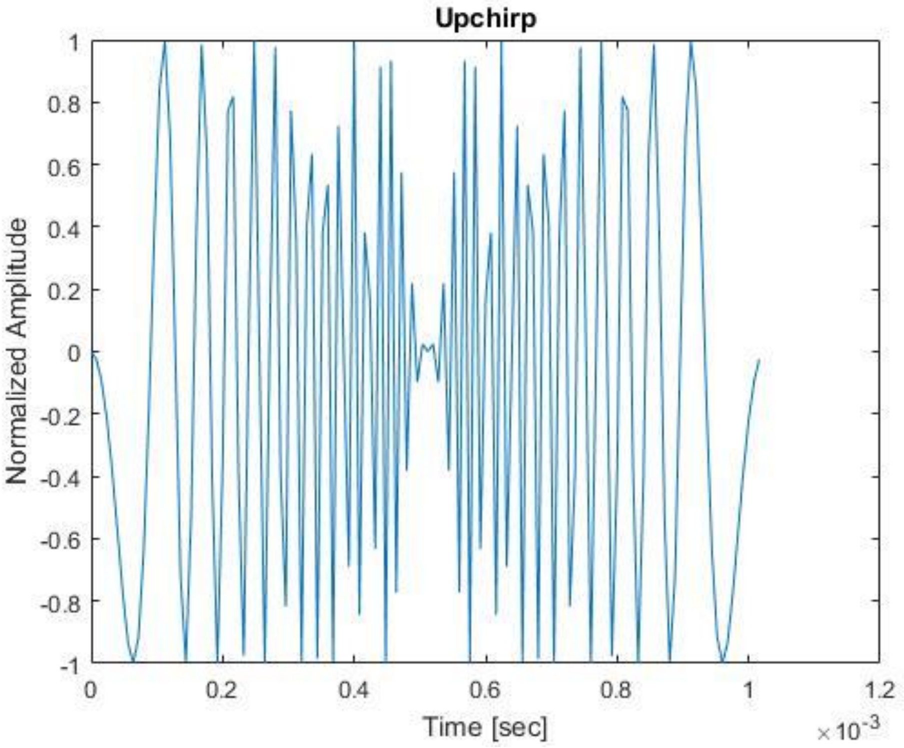


Figure 29. Baseband upchirp from the MATLAB function

The synchronization and plotting of the signal is done, with the use of the spectrogram function of MATLAB. The spectrogram function returns the short-time Fourier transform of the input signal.

De-chirping, symbol retrieval and plot

```
% replicates the downchirp, in order to have the
% same length as the signal and then multiplies them

downChirp= repmat(downChirp,1,ceil(length(xx)/length(downChirp)));
downChirp= downChirp(1:length(xx));
de_chirped= xx.*downChirp;

% Spectrogram computation
% To create a spectrogram 'grid' of symbols, these are the parameters.
signal = de_chirped;
Nfft = 2^SF;
window_length = Nfft; % same as symbol_time*Fs;
[s, f, t] = spectrogram(signal, blackman(window_length), 0, Nfft, Fs);
```

```
% Spectrogram plotting
figure;
surf(f,t,20*log10(abs((s).')),'EdgeColor','none')
axis xy; axis tight; colormap(jet); view(2);
ylabel('Time');
xlabel('Frequency (Hz)');
ylim([0.0001 t(end)])
title('Spectrogram of signal')
```

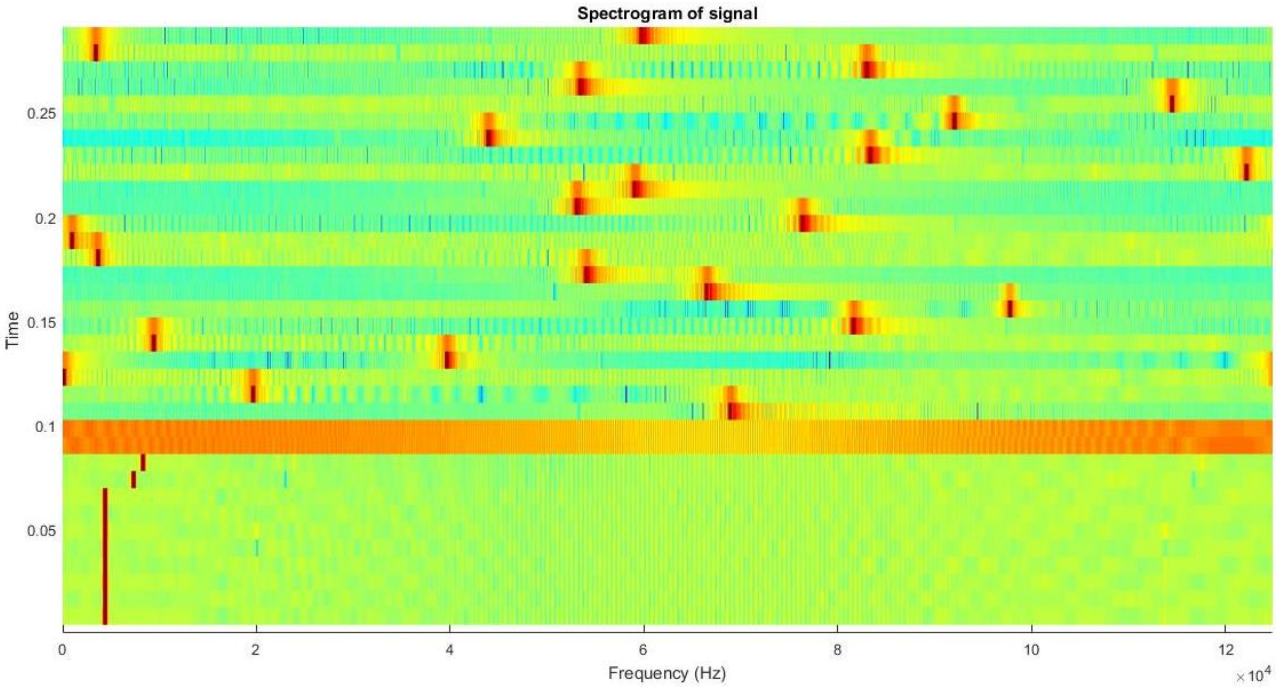


Figure 30. Spectrogram of the de-chirped signal

Due to poor synchronization the symbols appear to have twice their original duration (fig.30).

Preamble detection and synchronization to the payload

```
% finds the number of consecutive identical symbols
[~, symbols_ind] = max(abs(s));


number_preambles= 1;
while symbols_ind(number_preambles)== symbols_ind(number_preambles+1)
    number_preambles= number_preambles+1;
end

% the preamble consists of a number of identical consecutive symbols
% and two different symbols
de_chirped= de_chirped((number_preambles+2+2.25)*Nfft:end);
% The signal is synchronized to the start of the payload
% The payload starts 2.25 symbols after the start of the SFD
```

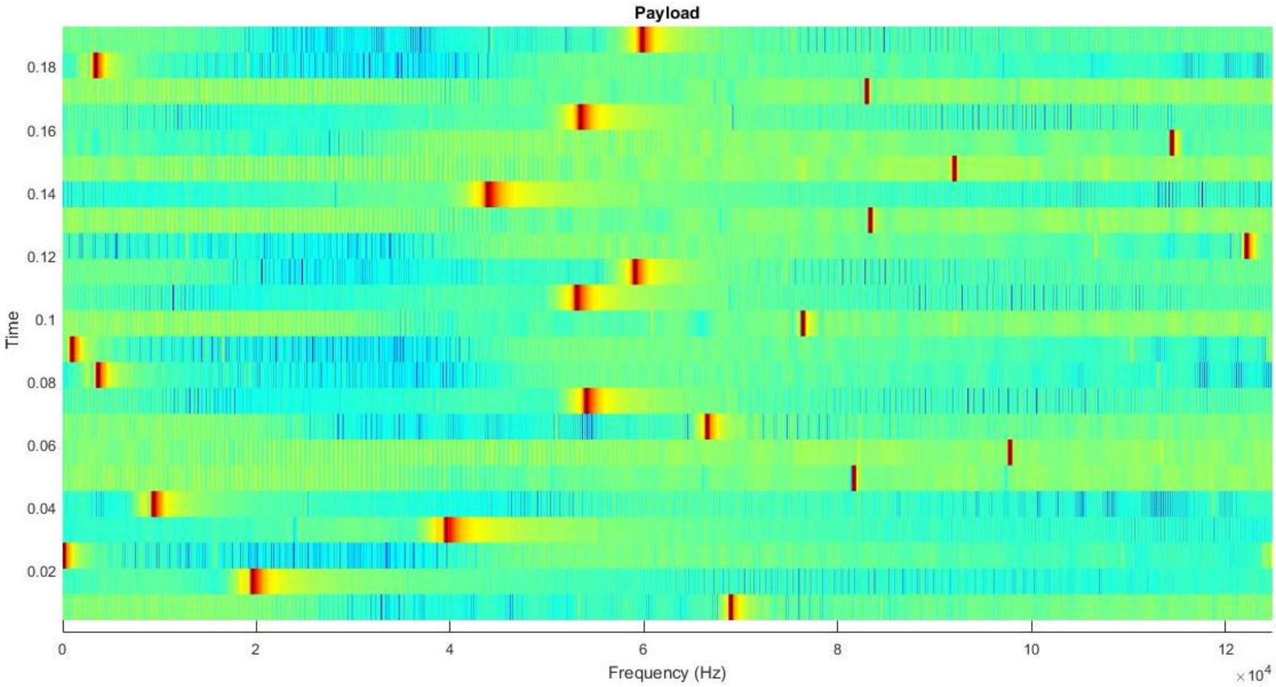


Figure 31. Synchronized signal

By skipping the payload, the SFD and 0.25 symbols, the payload symbols are obtained and the synchronization is fixed. As a result, the duration of the symbols is now correct.

SFD plotting

```
% The SFD is two downchirp symbols
upChirp = repmat(upChirp,1,ceil(length(xx)/length(upChirp)));
upChirp = upChirp(1:length(xx));
de_chirped_SFD = xx.*upChirp;

signal = de_chirped_SFD;
Nfft = 2^SF;
window_length = Nfft;
[s, f, t] = spectrogram(signal, blackman(window_length), 0, Nfft, Fs);
    figure;
surf(f,t,20*log10(abs((s).')),'EdgeColor','none')
axis xy; axis tight; colormap(jet); view(2);
ylabel('Time');
xlabel('Frequency (Hz)');
ylim([0.0001 t(end)])
title('Start of Frame Delimiter')
```

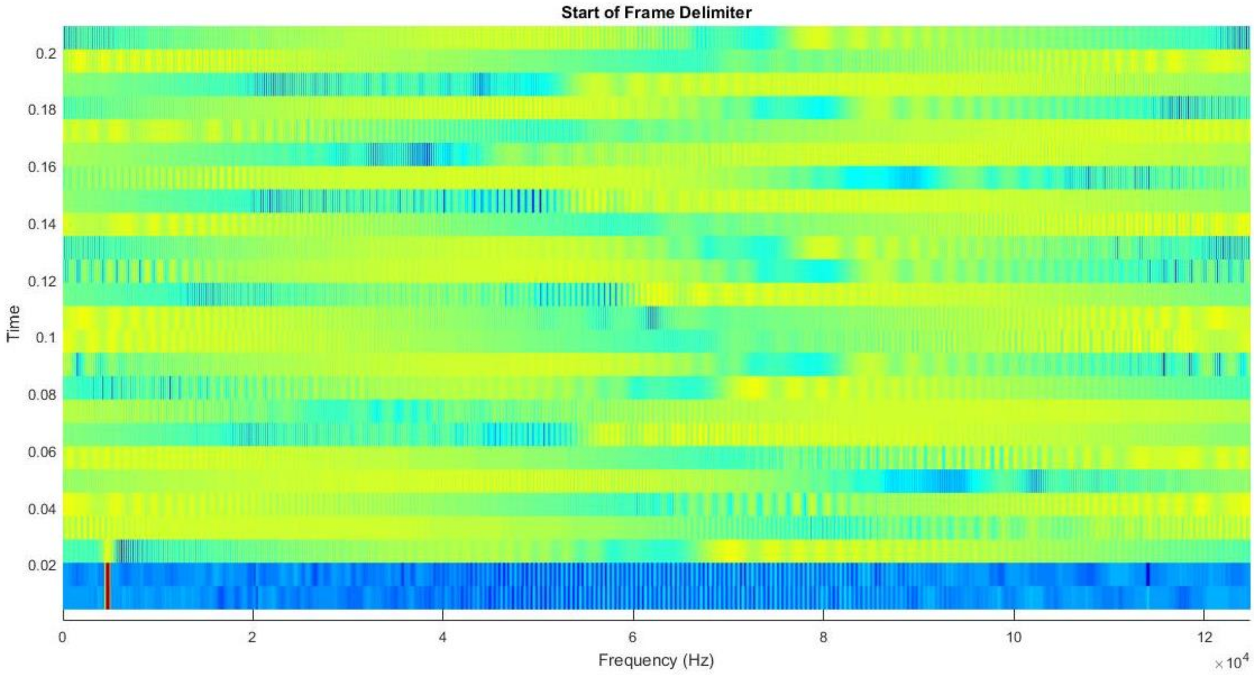


Figure 32. Start of Frame Delimiter

The next step is to demodulate the signal. The demodulation process includes 2 steps. The first step is to obtain the symbols and convert them to bits. The second step is to convert the bits to gray code.

The first step is done using the short-time Fourier transform again on the de-chirped symbols.

Demodulation

```
[~, symbols_ind] = max(abs(s));
bits= dec2bin(symbols_ind,SF);

% gray indexing
bits2= bits';
bits2= str2num(bits2(:));
bits2= reshape(bits2,SF,[])’;
gray_bit= bi2gray(bits2); % converts the bits to gray code
```

The symbols are the frequency bins of the absolute maximum value of the short-time Fourier transform. The short-time Fourier transform of a symbol is shown in figure 33. After obtaining the symbols, they need to be converted to binary format and then in gray code.

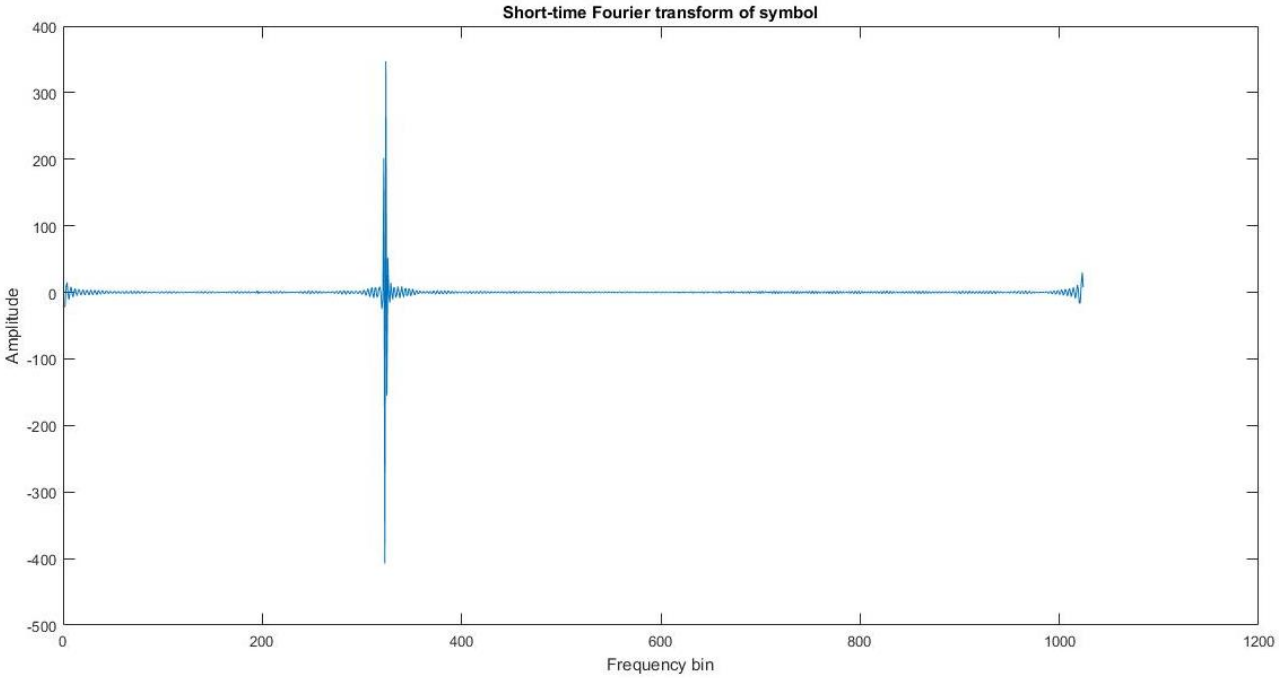


Figure 33. Short-time Fourier transform of symbol

The de-interleaving process that is defined in the patent [18], is described in Tables 7-10.

	Spreading Factor							
Coding Rate	0	0	1	1	0	0	1	1
	1	0	1	1	1	1	1	0
	1	1	1	1	1	0	1	0
	1	1	0	1	1	1	0	1
	1	0	0	0	0	0	1	0
	1	0	0	0	0	1	1	1
	1	1	0	0	0	0	0	0
	1	0	0	0	0	0	1	0

Table 7. Original bits

	Coding Rate							
Spreading Factor	1	0	0	1	0	1	0	0
	1	1	1	0	1	1	0	1
	0	1	0	1	0	1	0	0
	0	1	1	1	0	0	0	0
	1	1	1	1	0	0	0	0
	1	1	1	0	0	0	0	0
	0	0	1	1	0	0	1	0
	0	1	1	1	1	1	1	1

Table 8. Rotation counter clockwise by 90 degrees

	Coding Rate							
Spreading Factor	1	0	0	1	0	1	0	0
	1	1	1	0	1	1	0	1
	0	1	0	1	0	1	0	0
	0	1	1	1	0	0	0	0
	1	1	1	1	0	0	0	0
	1	1	1	0	0	0	0	0
	0	0	1	1	0	0	1	0
	0	1	1	1	1	1	1	1

Table 9. Diagonal De-interleaver

0	0	1	0	1	1	0	0
1	0	0	0	0	1	1	1
0	0	0	0	1	1	0	1
0	0	0	0	1	0	1	0
0	0	0	1	1	1	1	0
0	1	1	0	0	0	1	1
0	1	1	1	1	1	1	1
1	0	1	0	1	1	1	0

Table 10. De-interleaver output

In order to start the de-interleaving process, the bits are separated into tables of size Coding Rate x Spreading factor (table 7). The tables are then rotated counter-clockwise by 90^{o} (table 8) and the de-interleaver is used (table 9). The last step is to reverse the tables (table 10).

Function for the de-interleaver described in the patent

```
function de_interleaved= lora_de_interleaver_dok(bits,SF,CR)
% Diagonal de-interleaver from Semtech patent for lora
% Input is the bit sequence, the spreading factor and the coding rate
% Input must be of size (CRxSF,1)
% If the coding rate is (8,4), CR= 8
% Output is de-interleaved bits in an array of size CRxSF


bits_prep= reshape(bits,SF,[])'; % table 7


de_interleaved_prep= fliplr(bits_prep).'; % table 8
de_interleaved_prep2= fliplr(de_interleaved_prep); % table 9
% de_interleaved_prep prepares the input for use of the diag function of matlab


aa= 1;
bb=CR;
```

```
    cntr= CR;
    k=0;
for ii= 1:SF
    if CR>SF
        de_interleaved(ii,:) = [diag(de_interleaved_prep2,k)',diag(temp,cntr-1)'];
    else
        temp= diag(de_interleaved_prep2,k)';
        if length(temp)<CR
            de_interleaved(ii,:) = [temp,diag(de_interleaved_prep2,cntr-1)'];
            cntr= cntr-1;
        else
            de_interleaved(ii,:) = temp;
        end
    end
    k= k-1;
end
end
```

The Hamming decoder is already implemented in Matlab. The decoder function is decode(data,n,k,'hamming/binary'), where data is the input data, n is the code length and k is the message length [28].

7 Conclusions- Extensions

In this thesis, the physical layer of LoRa has been explored. It is shown that the properties of LoRa are ideal for long range IoT applications, although the FEC that is used is lacking. There is no error correction capability for two of the four rates, while one rate is capable of correcting only one bit and increases the payload length by 75%, in comparison with no FEC at all.

In the experiments, it is shown that the gr-lora solution functions correctly, like a commercial receiver. The only functions that are not implemented to date are the CRC checks of the header and the payload and the simultaneous decoding of multiple channels.

The Matlab implementation includes preamble detection, synchronization, demodulation, interleaving and whitening. The energy detector is adequate for the purpose of identifying the signal, but cannot identify the start of the payload. Some difficulties were encountered, as LoRa is not open source. The most important is the interleaver, as there are different opinions about whether the one in the patent is the one implemented or not.

Bibliography

[1] IEEE Std 802.15.4™-2011

[2] http://www.ieee802.org/15/pub/TG4.html

[3] LTE-M Deployment Guide to Basic Feature set Requirements

[4] https://www.sigfox.com

[5] Project: IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)

[6] SX1272/3/6/7/8: LoRa Modem, Designer's Guide, AN1200.13

[7] AN1200.22 LoRa Modulation Basics

[8] On the Coverage of LPWANs: Range Evaluation and Channel Attenuation Model for LoRa Technology

[9] http://www.ti.com/solution/software-defined-radio-sdr-diagram

[10] Bruce A. Fette, Cognitive Radio Technology

[11] N. Ahmed, H. Rahman, Md.I. Hussain: A comparison of 802.11ah and 802.15.4 for IoT

[12] AN1200.18 Implementing Data Whitening and CRC Calculation in Software on SX12xx Devices

[13] Matt Knight, Bastille Networks: Reversing LoRa: Exploring Next-Generation Wireless

[14] https://www.lora-alliance.org/technology

[15] LoRa Alliance: A technical overview of LoRa ® and LoRaWAN ™

[16] Challenges and Experiment with LoRaWAN

[17] Matt Knight: Decoding LoRa, a Wireless Network for the Internet of Things

[18] Patent EP2763321 A1

[19] https://blog.surf.nl/en/lora-the-internet-of-things/

[20] https://docs.pycom.io/

[21] Pieter Robyns, Peter Quax, Wim Lamotte, William Thenaers. (2017). gr-lora: An efficient LoRa decoder for GNU Radio. Zenodo. 10.5281/zenodo.853201

[22] https://pycom.io

[23] Pieter Robyns: LoRa Reverse Engineering and AES EM Side-Channel Attacks using SDR

[24] https://www.murata.com

[25] GSMA WHITE PAPER: 3GPP LOW POWER WIDE AREA TECHNOLOGIES

[26] http://www.realtek.com.tw

[27] https://myriadrf.org/blog/lora-modem-limesdr/

[28] www.mathworks.com