Jamming LoRa and Evaluation of Ease of Implementation

Jonas Stenholt Melchior Jensen[1], Bjørn Alexander Wade Patterson[2], Tomasz Blaszczyk[3(✉)], and Birger Andersen[4]

[1] Cibicom, Ballerup, Denmark
[2] Danish Data Protection Agency, Valby, Denmark
[3] DIS Study Abroad in Scandinavia, Copenhagen, Denmark
tomb@dis.dk
[4] Technical University of Denmark, Kongens Lyngby, Denmark

Abstract. The number of wirelessly connected devices are on the rise. With increasingly capable devices becoming cheaper and more accessible, the threat towards these radio communications is steadily growing. One of the more common technologies in the low power and long range realm of these devices is LoRaWAN. With a selection of different readily available hardware, we have examined if it is possible to conduct jamming attacks on LoRa. We have also evaluated the ease of implementation, and found that it is possible for less technically proficient persons to carry out effective jamming attacks on LoRa of different complexities using affordable hardware and open-source software.

Keywords: LoRa · LPWAN · SDR · ESP32 · SODAQ ExpLoRer · Ettus USRP E312 · HackRF One · GNU Radio · Jamming

1 Introduction

The Internet of Things (IoT) as topic under Internet of Everything has attracted much attention over the last years due to relevance in connecting devices all over the world. Automation plays a large role in this, since homes and even entire cities are looking to automate tedious tasks using the concept of IoT. A fundamental aspect of IoT is wireless communication technology whether short range communication in home automation environments or long range communication in larger scale IoT projects. Wireless Personal Area Networks (WPANs) such as ZigBee, Bluetooth, or Z-wave and Wireless LANs (WLANs) such as WiFi all provide for short range communication. When it comes to long range communication different technologies exist such as cellular communication, e.g., 5G, while lower powered solutions such as Low Power Wireless Area Networks (LPWAN) provide for longer battery life for end-devices with data rate as the trade off. An overview of the different protocols can be seen in Fig. 1.

When researching different LPWANs, it becomes clear that the architecture of the different protocols differ a lot both in terms of the PHYsical (PHY) layer,

T. Pereira et al. (Eds.): IOECON 2023, LNICST 551, pp. 19–41, 2024.
https://doi.org/10.1007/978-3-031-51572-9_3

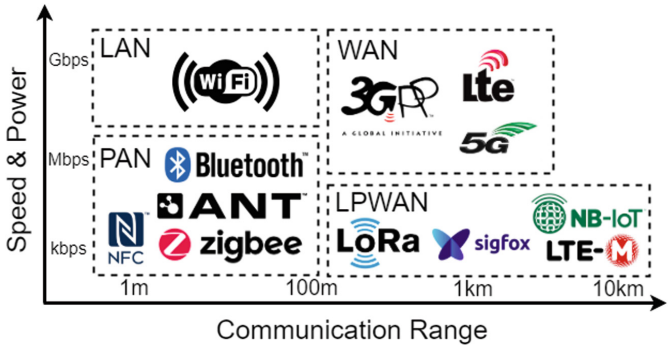

Fig. 1. Comparison of Wireless Protocols from [1].

Medium Access Control (MAC) layer and network layer. SigFox PHY is based on an ultra narrow band (200 Hz) and the entire network architecture is operated by SigFox (now unabiz), which means end users cannot deploy their own base stations/gateways and need to connect their end-devices to the SigFox back-end. In contrast to this, LoRa modulation [2] uses chirp spread spectrum utilizing a much wider frequency band, while LoRaWAN [3] allows for end users to deploy their own gateways and network servers which gives the opportunity to create private networks. Due to these features, LoRa has attracted a lot of attention and research on how to expand IoT environments [4,5], including in a context of sustainability [6].

With more data being transmitted via radio by IoT devices and these being used increasingly in sensitive/critical fields such as infrastructure monitoring, it is important to make sure that the focus on IT-security is continuous [7].

The concept of radio communication jamming encompasses a variety of methods designed to either interfere, disrupt or entirely block communications. These can be of low complexity such as random noise attacks or of a higher complexity such as a reactive attack. A simple Gaussian noise based attack is generally indiscriminate in nature. These attacks do not utilize any form of signal modulation, and are designed to blanket a selected frequency interval, interfering with the reception of the legitimate transmissions. For them to be effective, they often have to be continuous and high powered. This in turn makes it easier to track down and are often limited in their mobility due to need of a large power source. Reactive attacks are directed attacks that only act during a legitimate transmission. These jamming transmissions can also be modulated like the legitimate transmission, which can be more disruptive to technologies that work well in low Signal to Noise Ratio (SNR).

Electronic devices of a given capability generally decrease in price over time as technology advances. Along with the increased interest surrounding IoT implementation, this means that there are a lot of different IoT devices available to the public, including those with radio communication capabilities. This has the unintentional down side of making devices able to jam transmissions readily

available to the public as well. Selecting the same or similar device to perform the attack allows for access to the used modulation, and the current availability makes getting a hold of them easy.

The evolution of Software Defined Radio (SDR) has resulted in a wide variety of cheap products that can quite easily be set-up by lower skilled technical persons, for analysis of any transmission that they encounter. Devices such as the Great Scott Gadgets HackRF One are able to operate between 1MHz to 6GHz which covers most of the bands used by devices in the realm of IoT [8] and even technologies such as WiFi. These devices play a large role in the implementation of jamming methods, as they can be used as the main tool for determining the parameters of a given signal, of which the jamming methods can be tuned towards. They can also perform attacks themselves, as some of them also have transmitting capabilities.

In our work we evaluate the ease-of-implementation regarding being able to perform jamming attacks on legitimate LoRa PHY layer transmissions. This is done with the help of an array of set-ups using different publicly available hardware (IoT devices and SDRs) and jamming techniques of various complexities. We also discus the potential security impacts this may have on the current IoT landscape.

2 Background

In the following sections, an explanation of LoRa PHY and LoRaWAN principles will be provided. Prior work regarding jamming attacks on LoRa PHY will also be discussed.

2.1 LoRa PHY

At the physical layer LoRa uses its own modulation technique maintained by Semtech [2], based on Chirp Spread Spectrum (CSS). A chirp can be characterized as a sinusoidal wave, which either increases (up-chirp) or decreases (down-chirp) linearly over time in the frequency spectrum. In Fig. 2, a chirp in the time domain is visualized. Note that this is how the modulated signal in CSS will look in time domain.

What LoRa does differently is that the frequency shifting depends on the symbol transmitted which adds discontinuity to the modulated signal in the time domain. A thorough mathematical description of LoRa modulation was introduced in [9], which also raises the idea that LoRa might be better described as Frequency Shift Chirp Modulation (FSCM). The modulated signal c is expressed as

$$c = \frac{1}{\sqrt{2^{SF}}} e^{i2\pi[(s(nT_s)+k)2^{SF}]kT\frac{B}{2^{SF}}}$$

where SF is the spreading factor and the number of possible symbols is $2^{SF} - 1$. The symbol to be transmitted is represented as $s(nT_s)$ where T_s is the period of transmitting one full symbol. What is worth noticing, is that the frequency is in

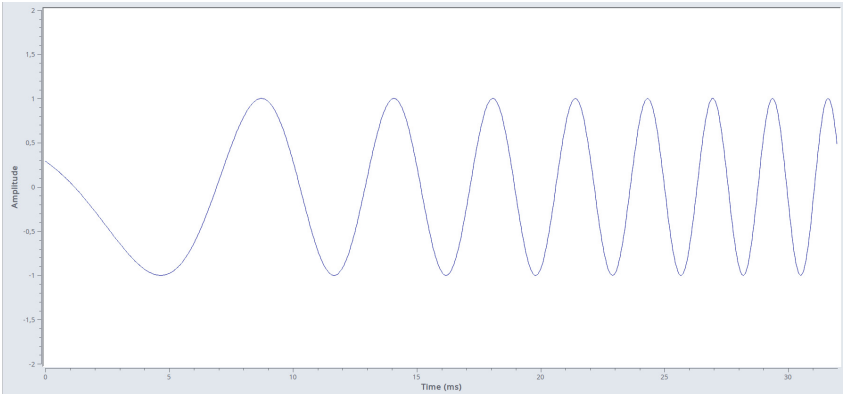

Fig. 2. A chirp in time domain.

fact increasing linearly with k as the time index. A discontinuity is introduced in term 2^{SF}, hence the name frequency shift chirp modulation. These features can bee seen in Fig. 3, where the symbol dictates the starting frequency, after which it increases linearly until it reaches the bandwidth limit and then returns to the lowest frequency within the given bandwidth increasing to the starting frequency again.

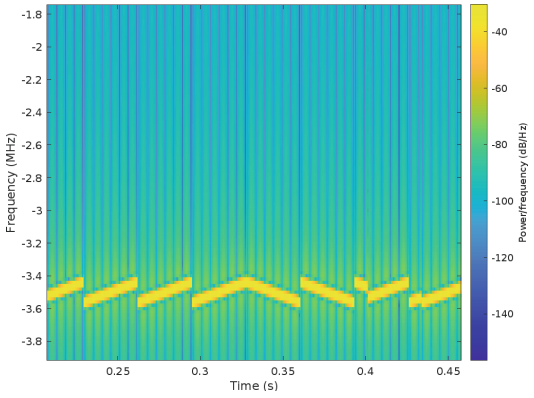

Fig. 3. LoRa chirps in frequency domain.

From [2] we have bit rate R_b on the output of the modulator expressed as

$$R_b = SF \cdot \frac{1}{\frac{2^{SF}}{BW}}$$

where BW is the modulation bandwidth in Hz. What can be derived from this, is that with a larger spreading factor of a given bandwidth (usually 125 KHz in LoRa), the bit rate decreases, meaning longer transmission times for larger spreading factors which is a key property of LoRa.

What happens at the receiving end of LoRa, is that to detect an incoming signal, the transmitter will send a preamble usually of 8 up-chirps followed by the actual data symbols. To demodulate the signal, it is first multiplied by an inverse chirp resulting in constant frequencies of the symbols. Conceptually after "de-chirping", a similarity check is performed between the received symbols and the base symbols representing perfect chirp representations of the different symbols. The functionality of this is performed by Discrete Time Fourier Transform (DTFT) - in most receiver designs Fast Fourier Transform (FFT), which outputs the signal in frequency domain to be correlated with the frequency components of the base chirps. The most significant frequencies output by FFT, represent the transmitted symbols at least for low noise levels. Figure 4 illustrates exactly this, where DTFT is performed on a de-chirped signal in which the highest amplitudes of the different frequency components will be considered the frequencies of the de-chirped transmitted symbols.

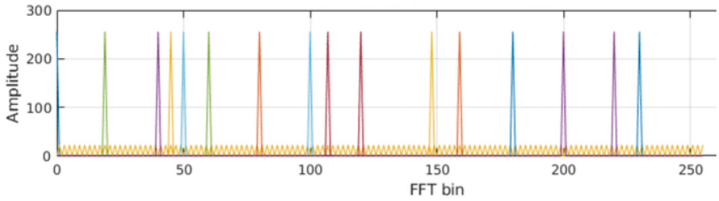

Fig. 4. LoRa FFT [10].

CSS (and DTFT) is a part of the explanation on why LoRa is robust against high noise levels and can even be demodulated from below the noise floor due to exactly this correlation check. As an example, with a spreading factor of 12 reliable communication can be achieved at SNR of −20 dB. In fact you can have a link budget of approximately 151 dB for a spreading factor of 12 with the receiver sensitivity being −137 dBm. A complete list of the link budget for the different spreading factor with a bandwidth of 125 kHz and fixed transmission power can be seen in Table 1 from [2].

Table 1. Link budget for different spreading factors in LoRa.

Mode	Equivalent bit rate (kb/s)	Sensitivity (dBm)
LoRa SF = 12	0.293	−137
LoRa SF = 11	0.537	−134.5
LoRa SF = 10	0.976	−132
LoRa SF = 9	1757	−129.7
LoRa SF = 8	3125	−126
LoRa SF = 7	5468	−123

2.2 LoRaWAN

LoRaWAN is the network protocol present in the MAC layer specified in [3]. The general architecture of a LoRaWAN network can be seen in Fig. 5 where different end-devices communicate with gateways using LoRa on the physical level. The gateways are connected to a network server through the TCP/IP protocol (secured by TLS) which has the responsibility of sending the payloads to various applications. Payload is further secured end-to-end by AES encryption.

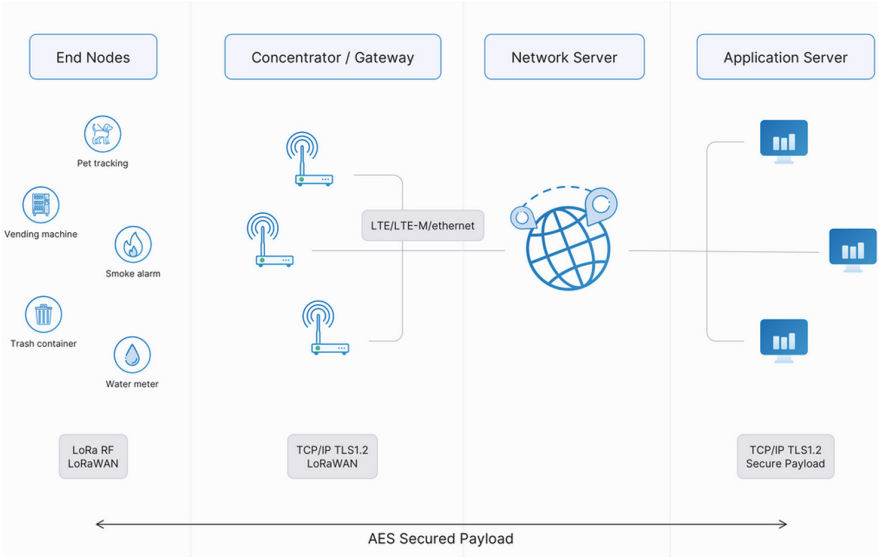

Fig. 5. LoRaWAN architecture [11].

One of the major delimiters of this specification is the different device classes where an end-device can either be of class A, B or C. Uplink communication is labeled as communication from end-device to network server while downlink communication is labeled as communication from network server to end-device. Class A is the least power consuming, since each device only listens for downlink messages during two receive windows after each uplink transmission (RX1 and RX2). For class B devices, the gateways within the network will synchronously broadcast a beacon which is the reference used by end-devices to open receive windows periodically synchronized with exactly these beacon signals. Class C devices will continuously listen for downlink traffic when not transmitting, being the most power consuming class.

For a device to be activated, i.e., joined to network, there are two methods of activation - Activation By Personalization (ABP) and Over-The-Air Activation (OTAA). In general OTAA is recommended, since it depends on the derivation of AES keys rather than fixed hard coded keys in the end-device as in ABP

as specified in [3]. In LoRaWAN 1.1, 5 keys are derived from two root keys, AppKey (Application key) and NwkKey (Network Key) which are assigned to the end-device during fabrication (in case of ABP). From NwkKey, 3 network session keys are derived: FNwkSIntKey, SNwkSIntKey and NwkSEncKey. From AppKey, 1 application session key is derived: AppSKey. All keys are AES-128 symmetric keys where the network session keys are used for integrity check from end-device to network server using AES-128 CMAC mode while the application session key is used for confidentiality from end-device to application server using AES-128 CTR mode. In prior versions to LoRaWAN 1.1, only two sessions keys were derived: Network session key (NwkSKey) and application session key (AppSKey).

An important consideration for LoRaWAN networks are the duty cycles associated with the license-free bands regulated in [12], which specifies how many times a device may transmit per time unit. In EU863-870, the duty cycles are specified for 5 sub-bands depicted in Table 2. All radio devices in the network must follow these constraints to be compliant with LoRaWAN, i.e., both end-devices and gateways. Some end-devices have radio modules which enforce the duty cycles automatically and will throw error messages if a duty in a given sub-band is exceeded.

Table 2. Duty cycles in EU863-870.

Bands	Duty cycles
Sub-band 1 (863.0–868.0 MHz)	1%
Sub-band 2 (868.0–868.6 MHz)	1%
Sub-band 3 (868.7–869.2 MHz)	0.1%
Sub-band 4 (869.4–869.65 MHz)	10%
Sub-band 5 (869.7–870.0 MHz)	1%

Table 3. Frequency Plans for TTN.

Frequency (MHz)	SF and bandwidth delimiters
868.1	SF7BW125 to SF12BW125
868.3	SF7BW125 to SF12BW125 and SF7BW250
868.5	SF7BW125 to SF12BW125
867.1	SF7BW125 to SF12BW125
867.3	SF7BW125 to SF12BW125
867.5	SF7BW125 to SF12BW125
867.7	SF7BW125 to SF12BW125
867.9	SF7BW125 to SF12BW125
868.8	SFK
869.525	SF9BW125 (RX2)

The different channel frequencies used in LoRaWAN depend on the community network, however compliant with the LoRaWAN regional parameters [13]. The network used in this project for testing purposes is The Things Network (TTN), which also specify the channel frequencies in EU863-870, depicted in Table 3. Note that the channel used for downlink communication in RX1, is a function of the channel and data rate used in the initial uplink transmission.

2.3 LoRa Jamming

Due to the demodulation scheme, LoRa is robust to interference and noise. In [14], jamming with Gaussian noise was possible at the expense of transmission power and demonstrated that LoRa PHY is robust to noise compared to other wireless technologies. In [7] was simulated a scenario in which the jamming device and legitimate device were placed 500 m and 2000 m away respectively from the gateway and Package Error Rates (PERs) above 10^{-1} was achieved using a Gaussian noise source with the jamming node only transmitting at 40% compared to the power of the legitimate node.

In [16,17] is shown that simultaneous LoRa transmissions with same frequency and spreading factor can collide with each other and not be resolved. A premise for this is that the jamming chirps must have a higher transmission power than the legitimate ones [15]. In such a scenario, the FFT performed in a given demodulation window would result in the jamming chirps having higher amplitudes rendering the legitimate chirps invalid. This jamming technique was utilized in [18] using commercial off-the-shelf LoRa devices. In [14] is suggested a similar approach in which the chirps also should be well aligned in the time domain and mimic legitimate chirps in the frequency domain to bypass collision recovery methods such as [19]. In order to align the chirps well and perform effective jamming with low probability of detection, reactive jamming is often sufficient which is also tested in [18]. The general setup for such an approach is to scan for a LoRa preamble with a specific spreading factor on a specific channel. Since most end-devices only allow for scanning a single channel for a preamble with a certain spreading factor at a time, channel hopping can be deployed as in [20], sequentially scanning channels to detect the preamble after which the jamming signal will be transmitted.

For our study, 4 jamming techniques derived from above discoveries will be used:

1. *Single Channel jamming (SC):* Jamming with LoRa modulation on one channel, transmitting continuously with a fixed spreading factor.
2. *Channel Hopping jamming (CH):* Jamming with LoRa modulation in which jamming device will jump between the channels (downlink) specified in Table 3 with a fixed spreading factor.
3. *Reactive Jamming (RJ):* The jamming device will utilize channel hopping when listening for a preamble on all downlink channel frequencies with a fixed spreading factor and start transmitting a LoRa signal when the preamble has been detected.
4. *Gaussian Noise jamming (GN):* The noise floor will be raised, such that the legitimate transmitted LoRa symbols can not be demodulated.

In an attempt to jam LoRa PHY, different experimental setups have been used. SDRs provide a great analytical tool and also have the capability of jamming with different RF signals as opposed to a standard LoRa only module. In [14,21], SDRs were used with the GNU Radio open-source software where the jamming signals were implemented. When going for an on-board LoRa module and Micro Controller (MCU), the SX127x LoRa module is popular and has good library support. In [22] was deployed an ESP32 embedded with an SX1276 module and [20] used an Arduino UNO connected to an external SX1276 module - In both cases, SX1276 with MCU acting as the jamming devices.

3 Materials and Methods

To test our selection of hardware's ability to jam LoRa transmissions, we set up a LoRa end-device to send legitimate join requests and data transmissions to a LoRa gateway, and attempted to jam them (Table 4).

Table 4. List of utilized hardware.

Module	Action	Modes
SODAQ ExpLoRer (RN2483 [23])	Transmitter	fixed SF12
Kerlink Wirnet iFemtoCell indoor LoRaWAN Gateway	Receiver	–
Great Scott Gadgets HackRF One [8]	Jammer	SC
ESP32 [24] with SX1276 [25]	Jammer	SC, CH, RJ
Ettus Research USRP E312 [26]	Jammer	GN

The jamming transmissions were all LoRa CSS signals at the same spreading factor as the legitimate transmissions. Both the legitimate LoRa end-device and LoRaWAN gateway were registered at on TTN, allowing us to track received transmissions (feature provided by TTN).

3.1 SODAQ ExpLoRer

The SODAQ ExpLoRer board was our real-world end-device analogue, used to send all transmissions during our experiments. To increase the chances of collision during our test and experiments, we forced the board to transmit using spreading factor 12. This would give us transmission times of up to 1500 ms depending on payload size [18], giving us more time to react and a better ability to follow jamming attempts visually, via a waterfall viewer in the AngelSDR software. In our case our payload was a single incremental digit, representing the number of the transmission (0–6). We used sub-band 1, described in Fig. 2. This was done with a simple c++ program, using the Sodaq_RN2483 library[1].

[1] www.github.com/SodaqMoja/Sodaq_RN2483, accessed on the 15th of January 2023.

These configuration changes were made for practicality reasons, but we believe that our source of legitimate transmissions is close enough to a real world implementation for us to assess the real world application of our techniques. The transmission power was set using an inbuilt function, and was set to a value of 5, equalling a transmission power of below 14 dBm = 25.12 mW. This is the lowest transmission power available for use on the 868 MHz band [23]. OTAA was used as the activation method during all tests and experiments. Due to duty cycle compliance we had a limit on how often we could transmit our legitimate LoRa transmissions. This resulted in us only being able to send up to 7 data transmissions after each successful join-request. For practicality reasons and time constraints we used this limit as the basis for how many transmissions we would send per set during our experiments.

The end-device was set up in such a way, that if it experienced 3 failed join requests, the program would halt stopping further attempts.

3.2 HackRF One

For the HackRF One jamming implementation, GNU Radio was utilized to implement SDR. Since we are using LoRa signals to jam LoRa PHY for this device, a GNU Radio SDR LoRa transceiver implementation [27] was used to generate our jamming LoRa signals. To connect the HackRF One to GNU Radio, we used Osmocom blocks from the library[2]. The block diagram used in our experiments can be seen in Fig. 6 in which *SC* jamming with center frequency 867.3 MHz was performed.

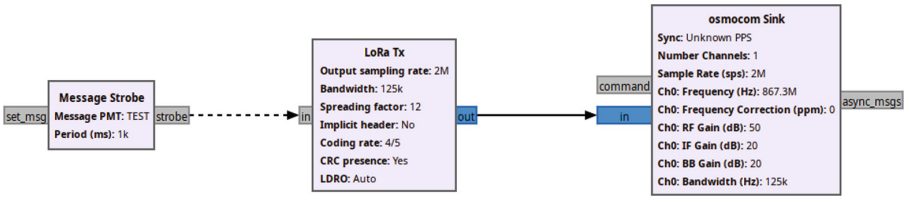

Fig. 6. HackRF One GNU Radio jamming *SC* implementation.

3.3 ESP32 with SX1276 LoRa Module

For the ESP32 embedded with an SX1276 LoRa module, several LoRa libraries were used in relation to the different jamming techniques.

[2] www.github.com/osmocom/gr-osmosdr, accessed on the 19th of January 2023.

SC Implementation. For the *SC* jamming implementation, an Arduino library[3] for the SX1276 was used to transmit LoRa signals on the frequency 867.1 MHz for this implementation. The associated code can be seen below in Listing 1 where the Serial Peripheral Interface (SPI) between the ESP32 and SX1276 module is established and the LoRa PHY parameters are set afterwards.

```
#include <SPI.h>
#include <LoRa.h>

// Pinout to SX1276
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DIO0 26

#define BAND 867.1E6

int frequency = BAND;

void setup() {
  //SPI interface to SX1276
  SPI.begin(SCK, MISO, MOSI, SS);
  //setup SX1276
  LoRa.setPins(SS, RST, DIO0);

  if (!LoRa.begin(BAND)) {
    while (1);
  }

  LoRa.setFrequency(frequency);
  LoRa.setSpreadingFactor(12);
  LoRa.setTxPower(14);
  LoRa.setSignalBandwidth(125E3);
  delay(2000);
}

void loop() {
  LoRa.beginPacket();
```

[3] www.github.com/sandeepmistry/arduino-LoRa, accessed on the 18th january 2023.

```
35    LoRa.print("jamming");
36    LoRa.endPacket();
37  }
```

Listing 1. SC Source Code.

CH Implementation. For the *CH* jamming implementation, the same library as in *SC* was used. The way channel hopping can be achieved is simply by incrementing the center frequency by 0.2 MHz and make sure, that we do not exceed the bands specified in Table 3. The only changes made to the code is in the loop function (Listing 2).

```
1   void loop() {
2     // Wrap around
3     if (frequency == 868.7E6) {
4       frequency = BAND;
5     }
6
7     LoRa.setFrequency(frequency);
8
9     LoRa.beginPacket();
10    LoRa.print("jamming");
11    LoRa.endPacket();
12
13    frequency += 0.2E6;
14  }
```

Listing 2. CH Source Code - setup equivalent to Listing 1.

RJ Implementation. For reactive jamming, RadioLib was used[4] which has preamble detection functionality for the SX1276. The code written for this implementation can be seen in Listing 3, where channel hopping again is utilized in the sense that the SX1276 is listening sequentially on all channels in the band and transmits whenever a preamble is detected.

```
1   #include <RadioLib.h>
2
3   #define LORA_SCK        05
4   #define LORA_MISO       19
5   #define LORA_MOSI       27
```

[4] https://github.com/jgromes/RadioLib, accessed on the 21st January.

```
#define LORA_SS         18
#define LORA_DIO0       26
#define LORA_DIO1       33
#define LORA_RST        14

float frequency =       867.1;

SX1276 radio = new Module(LORA_SS, LORA_DIO0, LORA_RST, LORA_DIO1);

void setup() {

  // Setup SX1276 wiring to ESP32
  int state = radio.begin();
  if (state != RADIOLIB_ERR_NONE) {
    while (true);
  }

  radio.setPreambleLength(8);
  radio.setOutputPower(14);
  radio.setCodingRate(5);
  radio.setSpreadingFactor(12);
  radio.setSyncWord(34);
  radio.setGain(1);

  delay(100);
}

void loop() {
  radio.setFrequency(frequency);

  if (radio.scanChannel() == RADIOLIB_PREAMBLE_DETECTED) {
    radio.transmit("");
    delay(100);
  }
  frequency = frequency + 0.2;
  if (frequency > 868.5) {
    frequency = 867.1;
  }
}
```

Listing 3. RJ Source Code.

3.4 Ettus USRP E312

For the Ettus USRP E312, many possibilities exist regarding how to transmit the jamming signal. Since it runs an embedded OS, it has built-in commands which can generate different signals. A more popular and common approach is however to develop the program on a host PC usually in GNU Radio and connect the USRP through the UHD driver, which needs to be of at least version 4.0 on host and target. The GNU Radio implementation with a Gaussian noise source can be seen in Fig. 7.

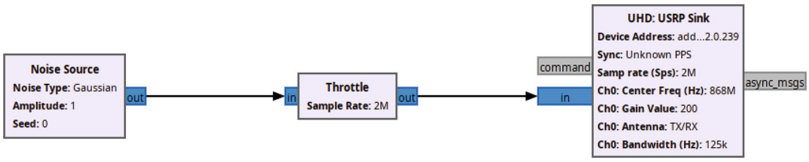

Fig. 7. USRP E312 Gnu Radio jamming *GN* implementation.

3.5 Experimental Methonds and Set-Ups

Each experiment consisted of three local devices: The legitimate transmitter (SODAQ ExpLoRer), the receiver (Kerlink LoRaWAN Gateway) and the jamming device and the LoRaWAN network and application server (TTN) which allowed for monitoring.

The experiments were performed indoor at with a static set-up. For all SC, CH and RJ experiments, there was approximately 57 m direct line-of-sight between the SODAQ transmitter and the gateway, and the jammer was approximately 2 m from the gateway. Along the transmission path the signal passed through an interior wall and an interior floor. For the GN experiment, there was approximately 47 m direct line-of-sight between the SODAQ transmitter and the gateway, and the jammer was approximately 1 m from the gateway. Along the transmission path the signal passed through 2 exterior walls.

To determine if a transmission was successfully jammed we used the live data feed from TTN. This displays the up- and down-links from the LoRaWAN application server. We visually confirmed new entries in the list to determine if the data transmission had made it through or not. The number of outgoing transmissions was counted in the software on the SODAQ board and printed to screen via a serial monitor, and we also visually confirmed that the transmission was put on the air. This was done by monitoring the waterfall visualisation of the sub-band in AngelSDR using a HackRF One as a receiver. Comparing the number of logged outgoing transmissions versus the number of received transmissions on the application server, we could calculate a jamming rate (Fig. 8):

$$\frac{jammed\ tx}{total\ tx} \cdot 100 = jamming\ rate\%$$

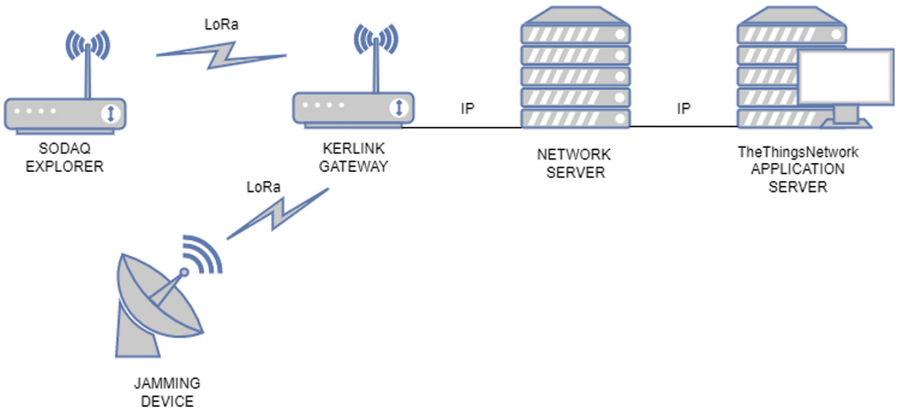

Fig. 8. Test bed diagram.

Our tests consisted of 7 sets of 7 transmissions. The first 5 sets of 7 were performed with the jamming device active, and the final 2 acted as a control, with the jamming device turned off. If the test resulted in a jamming rate $> 0\%$ it would validate our jamming technique as a successful proof-of-concept. If a jamming device was so effective as to block all join-request attempts, we would limit that set to the 3 join-request attempts the SODAQ board would attempt before halting. As join requests default to a spreading factor of 12, this lined up with the fixed spreading factor used as a fixed value throughout our experiments.

4 Experimental Results

As mentioned before, the jamming techniques (exceptGaussian noise) are based on jamming LoRa with LoRa. What this looks like in practice can be seen in Fig. 9 where the chirps with highest amplitude (yellow) are the jamming signal and overlaps with the legitimate chirps (green). Note that the jamming signal in fact mimics the legitimate signal both with spreading factor and channel center frequency while occupying close to the exact bandwidth of the legitimate signal.

When jamming with Gaussian noise as the source, Fig. 10 shows the most significant frequency components overlapping with the center frequency of LoRa channels in Table 3, which did in fact jam the communication.

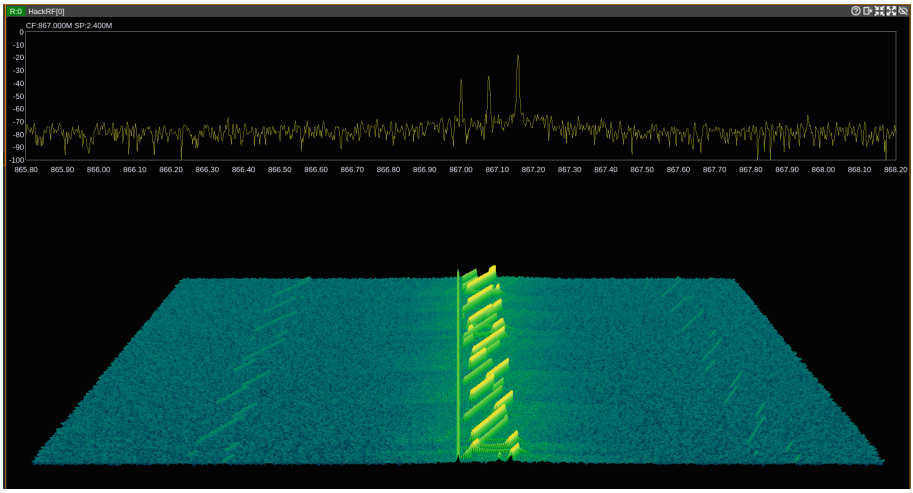

Fig. 9. Jamming LoRa with LoRa, captured with HackRF One in AngelSDR. (Color figure online)

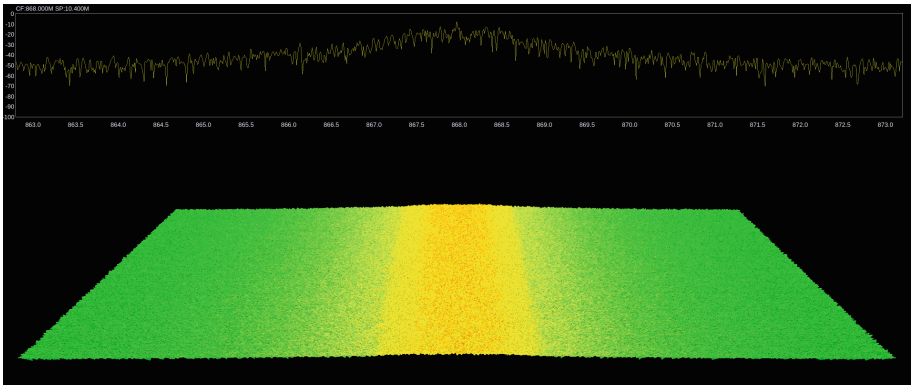

Fig. 10. Jamming LoRa with Gaussian noise, captured with HackRF One in AngelSDR. (Color figure online)

4.1 Jamming Results

Here we present the jamming results in Tables 5, 6, 7, 8, and 9. Note that all jamming devices' transmission power are at 14 dBm. All experiments consisted of 7 data transmission attempts, after a successful join-request attempt except for the experiments with the ESP32 and E312. Due to the effectiveness of the jamming techniques, we were not able to perform a successful join-request. Therefore the 3 join-request attempts are noted instead.

Table 5. HackRF One, Single Channel jamming results.

Run	Type	Jamming Rate
#1	Jamming	$1/7 \approx 14\%$
#2	Jamming	$2/7 \approx 29\%$
#3	Jamming	$2/7 \approx 29\%$
#4	Jamming	$1/7 \approx 14\%$
#5	Jamming	$1/7 \approx 14\%$
#6	Control	$0/7 = 0\%$
#7	Control	$0/7 = 0\%$

Table 6. ESP32 with SX1276, Single Channel jamming results.

Run	Type	Jamming Rate
#1	Jamming	$1/7 \approx 14\%$
#2	Jamming	$1/7 \approx 14\%$
#3	Jamming	$1/7 \approx 14\%$
#4	Jamming	$1/7 \approx 14\%$
#5	Jamming	$1/7 \approx 14\%$
#6	Control	$0/7 = 0\%$
#7	Control	$0/7 = 0\%$

Table 7. ESP32 with SX1276, Channel Hopping jamming results.

Run	Type	Jamming Rate
#1	Jamming	$2/7 \approx 29\%$
#2	Jamming	$2/7 \approx 29\%$
#3	Jamming	$3/7 \approx 43\%$
#4	Jamming	$1/7 \approx 14\%$
#5	Jamming	$1/7 \approx 14\%$
#6	Control	$0/7 = 0\%$
#7	Control	$0/7 = 0\%$

Table 8. ESP32 with SX1276, Channel Hopping jamming results.

Run	Type	Jamming Rate
#1	Jamming	$3/3 = 100\%$
#2	Jamming	$3/3 = 100\%$
#3	Jamming	$3/3 = 100\%$
#4	Jamming	$3/3 = 100\%$
#5	Jamming	$3/3 = 100\%$
#6	Control	$0/7 = 0\%$
#7	Control	$0/7 = 0\%$

Table 9. USRP E312, Gaussian Noise jamming results.

Run	Type	Jamming Rate
#1	Jamming	3/3 = 100%
#2	Jamming	3/3 = 100%
#3	Jamming	3/3 = 100%
#4	Jamming	3/3 = 100%
#5	Jamming	3/3 = 100%
#6	Control	0/7 = 0%
#7	Control	0/7 = 0%

5 Discussions

As can be seen in the jamming experiment results, *SC* both the HackRF and ESP32 with SX1276 jammed approximately one packet in every run. This relates to the fact that LoRaWAN utilizes pseudo-random channel hopping to avoid collisions, which is why one could expect a jamming rate of approximately 1/8 (8 downlink channels in Table 3), provided that continuous transmission is performed by the jamming device. *CH* jamming using the ESP32 with SX1276 performed a little better due to the fact that an ongoing legitimate transmission sometimes partially collides with a jamming signal. As expected, the *RC* jamming results gave a 100% jamming rate since the method is based on jamming the exact channel, the legitimate device is transmitting on whenever a preamble is detected. The simpler approach of *GN* jamming also resulted in a 100% jamming rate since the legitimate chirps were submerged by the Gaussian noise at such a level that the de-modulation was not able to distinguish the legitimate transmission.

It should be stated that *SC*, *CH* and *GN* jamming techniques all are easily detectable since they all exceed the regulated duty cycles and all use continuous transmission with no random element, at least in our implementation. A point worth raising is also, that jamming is indeed a malicious action and bypassing regional restrictions might not be of great significance to the adversary, rather the probability of being detected. *RJ* is on the contrary a much more careful technique. Since the jamming transmissions overlap with the legitimate transmissions and in our case mimic the exact channel and bandwidth of the legitimate signal, they are much harder to detect.

5.1 Countermeasures

To avoid jamming and in general minimize the chance of collision in LoRaWAN, a good approach is to utilize Adaptive Data Rate (ADR), meaning end-devices will vary their SF, bandwidth and transmission power in order to optimize data rate and energy consumption and ensuring that gateways can demodulate the transmitted signals. For a lot of jamming implementations including ours, ADR

is in fact sufficient to establish decent communication since a fixed spreading factor is used for the jamming device. ADR combined with channel hopping for legitimate devices limits jamming possibilities, since the range of channel frequencies for every spreading factor need to be iterated when jamming, whether *CH* or *RJ* is deployed.

In general, it is a good idea to look into the default behavior of an end-device and the libraries used to program it. Not all end-devices use channel hopping and ADR by default which sometimes have to be implemented by the user. Furthermore, ADR has to be implemented in LoRaWAN and if an end-device is not connected to a bigger community network such as, e.g., TTN, ADR might not be available, since it is the responsibility of the network server to indicate whether lower or higher data rates should be used.

More targeted countermeasures do however also exist. In [19] is proposed a parallel decoding technique (FTrack) for simultaneous LoRa transmissions at same frequency and same SF utilizing the fact, that the chirps might be and often are slightly misaligned in time domain. Hardware imperfections might also result in two simultaneous transmission differ slightly in frequency, where [28] (Choir) uses exactly these differences in frequency to distinguish different signals from each other.

5.2 Board Accessibility

The SODAQ ExpLoRer, HackRF One and ESP32 devices are all sub $500, as of January 2023, and we evaluate that they can be used by persons with slightly above average technical knowledge and skills. They all can be programmed and deployed with a consumer grade laptop/desktop computer. The software necessary to program them, such as Arduino IDE, GNU Radio, AngelSDR and the code libraries mentioned in the article are all free and open source. We evaluate that amount and quality of supportive material online, in the form of step-by-step guides on sites such as YouTube.com, GitHub.com and StackOverflow.com, is very high. This allows users to implement jamming capabilities, without deep technical knowledge of the underlying maths or physics. We believe that the average person interested in executing the attacks that we have demonstrated would have the skills to perform them.

The Ettus USRP E312 retails at around $5,000, as of January 2023, and we believe the price makes this device less accessible to the average person, compared to the devices mentioned above. For a person to use this device effectively, we believe that they will need a good understanding of UNIX type operating systems, local area networking, serial communication, radio communication and potentially the compilation of necessary software and drivers. The supportive material online is of a higher technical complexity, making it difficult to work with. Based on these assumptions we believe that this device requires at least the equivalent technical knowledge of a second year technical university student to program and implement as a LoRa jamming device.

5.3 Evaluation of Ease of Implementation

For the array of jamming devices used in our test setups it is relevant to make a high level evaluation on the difficulty of implementing these devices in practise. In relation to above explanation and experiences when implementing the devices' functionalities, we came up with Table 10[5]. The configuration parameter encapsulates installation of software implementation of, e.g., GNU Radio LoRa transceiver functionality [27], required UNIX experience, and in general tweaks needed to perform the jamming. Note that programming might be relevant on the HackRF and USRP, but a more popular approach is however to utilize GNU Radio functionality, which is why we rendered this parameter as *N/A* for these

Table 10. Evaluation of ease of implementation.

Device	Configuration	Programming	Flexibility	Compat.	Price
HackRF One	Medium	N/A	Medium High	Low	Low
ESP32 + SX1276	Easy	Easy	Low	High	Low
USRP E312	Hard	N/A	High	Low	High

devices in an adversary use-case. Flexibility denotes how flexible the devices are in terms of different jamming types which of course has to be compatible with the radio module in the devices. For both of the SDRs any signal can really be generated and the USRP E312 has enough transmission power and bandwidth to generate sufficient noise to jam LoRa communication. It might seem ambiguous, that the ESP32 with SX1276 has low flexibility, since we conducted both *SC*, *CH* and *RJ* for this device, but in the context of jamming with different signals, the SX1276 is indeed a LoRa module only able to jam LoRa with LoRa. Jamming LoRa with LoRa however, seems like the most efficient method also suggested in previous work, which is why the ESP32 with SX1276 is an easy to setup and efficient jammer in this context. The SDRs are however not as compatible with LoRa PHY, since external implementations of LoRa in GNU Radio need to be installed and configured.

As an overall assessment, everything taken into consideration, we have ranked the three devices in terms of the easiest jammer to implement against LoRa PHY in an adversary use-case:

1. ESP32 with SX1276
2. HackRF One
3. Ettus USRP E312

6 Conclusions

As more and more LPWAN capable devices are being fielded, and devices are being trusted with more sensitive tasks, the security surrounding these systems

[5] "Compat." means compatibility in terms of LoRa PHY.

must not lag behind. The LoRa protocol is very resilient towards low SNR and simultaneous transmissions of non LoRa modulation. However, it is not immune towards attacks utilizing the same type of CSS modulation.

We have put together 4 combinations of hardwares and jamming approaches to demonstrate this fact. With these experiments we have demonstrated that it is easy for a hostile actor to mount a LoRa jamming attack using cheap and readily available hardware, adding these to the already existing pool of more capable but also more expensive hardware. These range from a disruption of singular transmissions to a full DoS attack. Without much work a jamming attack can be improved to become a reactive attack, which is a lot more clandestine in nature. These types of attacks have a higher success rate, as the randomized nature of trying to catch a channel hopping signal is removed. We believe that due to the availability and low level of technical knowledge needed to perform such attacks, we could see a rise of LoRa device attacks in the future. If these devices are used as a part of infrastructural systems, this could have devastating results.

7 Future Work

The legitimate transmitter in our experiments was limited to the use of spreading factor 12, for practical reasons. One can argue that our analogue of an average LoRaWAN end-device is not truly representative of the targets an hostile actor would encounter in the field. A continuation of the work where the end-device is subjected to ADR, and other such methods of signal optimization, is required to be sure that our methods would still be viable. A more rigorous set of experiments with a larger set of parameters, such as more spreading factors, channels, signal powers, etc., should be realized.

The reactive attack could be extended to include an element of selectivity, seeking out only to jam transmissions from a specific end-device. This would raise the difficulty of detection even further, making fixing the situation that much harder. This type of attack would need a slightly deeper knowledge of the inner workings of the LoRa packet composition, but this information is readily available, and does not require a large leap in technical understanding to learn.

References

1. Jiang, X., et al.: Hybrid low-power wide-area mesh network for IoT applications. IEEE Internet Things J. **8**(2), 901–915 (2021)
2. AN1200.22 LoRa™ Modulation Basics. https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf. Accessed 15 Jan 2023
3. LoRaWAN™ 1.1 Specification. https://hz137b.p3cdn1.secureserver.net/wp-content/uploads/2020/11/lorawantm_specification_-v1.1.pdf?time=1673563697. Accessed 15 Jan 2023
4. Msaad, M., Hambly, A., Mariani, P., Kosta, S.: Mobile and delay tolerant network for LoRa at sea. In: CoNEXT 2020: Proceedings of the Student Workshop. ACM (2020)

5. Fargas, B., Petersen, M.: GPS-free geolocation using LoRa in low-power WANs. In: GIoTS 2017 - Global Internet of Things Summit, Proceedings (2017)
6. Orfanidis, C., Dimitrakopoulos, K., Fafoutis, X., Jacobsson, M.: Towards battery-free LPWAN wearables. In: ENSsys 2019 - Proceedings of the 7th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems, pp. 52–53 (2019)
7. Prasad, N., Lynggaard, P.: LoRaWan sensitivity analysis and prevention strategies against wireless DoS attacks. Wirel. Pers. Commun. **126**, 3663–3675 (2022). https://doi.org/10.1007/s11277-022-09884-8
8. Great Scott Gadgets, HackRF One specification. https://hackrf.readthedocs.io/en/latest/index.html. Accessed 15 Jan 2023
9. Vangelista, L.: Frequency shift chirp modulation: the LoRa modulation. IEEE Sig. Process. Lett. **24**, 1818–1821 (2017)
10. Liando, J., Jg, A., Tengourtius, A., Li, M.: Known and unknown facts of LoRa: experiences from a large-scale measurement study. ACM Trans. Sens. Netw. **15**, 1–35 (2019)
11. LoRaWAN architecture. https://www.thethingsnetwork.org/docs/lorawan/architecture/. Accessed 21 Jan 2023
12. ETSI En 300 220-1 V3.1.1. https://www.etsi.org/deliver/etsi_en/300200_300299/30022001/03.01.01_60/en_30022001v030101p.pdf. Accessed 18 Jan 2023
13. RP2-1.0.3 LoRaWAN® Regional Parameters. https://hz137b.p3cdn1.secureserver.net/wp-content/uploads/2021/05/RP002-1.0.3-FINAL-1.pdf?time=1674070505. Accessed 18 Jan 2023
14. Hou, N., Xia, X., Zheng, Y.: Jamming of LoRa PHY and countermeasure. In: Proceedings - IEEE INFOCOM (2021)
15. Hou, N., Xia, X., Zheng, Y.: Jamming of LoRa PHY and countermeasure. ACM Trans. Sens. Netw. **19**(4), 1–27 (2023). Article No. 80
16. Reynders, B., Meert, W., Pollin, S.: Range and coexistence analysis of long range unlicensed communication. In: 2016 23rd International Conference On Telecommunications, ICT (2016)
17. Aras, E., Small, N., Ramachandran, G., Delbruel, S., Joosen, W., Hughes, D.: Selective jamming of LoRaWAN using commodity hardware. In: ACM International Conference Proceeding Series, pp. 363–372 (2017)
18. Aras, E., Ramachandran, G., Lawrence, P., Hughes, D.: Exploring the security vulnerabilities of LoRa. In: 2017 3rd IEEE International Conference on Cybernetics (CYBCONF), pp. 1–6 (2017)
19. Xia, X., Zheng, Y., Gu, T.: FTrack: parallel decoding for LoRa transmissions. IEEE/ACM Trans. Networking **28**, 2573–2586 (2020)
20. Perković, T., Rudeš, H., Damjanović, S., Nakić, A.: Low-cost implementation of reactive jammer on LoRaWAN network. Electronics **10**, 864 (2021)
21. Wadatkar, P., Chaudhari, B., Zennaro, M.: Impact of interference on LoRaWAN link performance. In: Proceedings - 2019 5th International Conference on Computing, Communication Control and Automation, ICCUBEA 2019 (2019)
22. Ruotsalainen, H.: Reactive jamming detection for LoRaWAN based on meta-data differencing. In: ACM International Conference Proceeding Series (2022)
23. RN2483 LoRa Transceiver Module. http://ww1.microchip.com/downloads/en/DeviceDoc/RN2483-Low-Power-Long-Range-LoRa-Technology-Transceiver-Module-Data-Sheet-DS50002346D.pdf. Accessed 21 Jan 2023
24. ESP32 Technical Reference Manual. https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf. Accessed 20 Jan 2023

25. SX1276-7-8-9 Datasheet. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE. Accessed 20 Jan 2023
26. USRP E312 Datasheet. https://www.ettus.com/wp-content/uploads/2019/01/USRP_E312_Datasheet.pdf. Accessed 20 Jan 2023
27. Tapparel, J., Afisiadis, O., Mayoraz, P., Balatsoukas-Stimming, A., Burg, A.: An open-source LoRa physical layer prototype on GNU radio. In: 2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC) (2020)
28. Eletreby, R., Zhang, D., Kumar, S., Yagan, O.: Empowering low-power wide area networks in urban settings. In: SIGCOMM 2017 - Proceedings of the 2017 Conference of the ACM Special Interest Group on Data Communication, pp. 309–321 (2017)