Development of Low-cost LoRaWAN Gateway for Private Deployments

Nuno Gil Polónia Manita Nico
nuno.nico@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2017

Abstract

The main problem addressed in this thesis is the development of a Low-cost LoRaWAN gateway. To develop this gateway LoRa single-frequency modules have been used. Using modules that are only capable of receiving in one frequency simultaneously creates two main issues. The first is the vulnerability to jammers. A jammer can cause intentional interference and affect the proper operation of the system. To solve this issue, a frequency-hopping mechanism was implemented. Using this mechanism, the communications are constantly changing the frequency to avoid jamming. To solve this issue, were also implemented on the gateway two LoRa modules, each one capable of operating in different frequencies.

The second issue was to assure the correct sending of priority messages. This was done by implementing a time-slot reservation mechanism. Using this mechanism, a device can reserve a period in time where only it can send messages. The performance of the developed system was tested experimentally. The gateway range was tested and transmissions between an end-device and the gateway with up to 4.9 kilometers were achieved. The randomness of the generated sequences was tested, and the results show that it has enough randomness for this application. Was also tested the implemented extensions to LoRaWAN. Concluding that the developed mechanisms are effective.

Keywords: IoT, LPWAN, LoRaWAN, Gateway

1. Introduction

Nowadays, the interest on the Internet of Things (IoT) has been increasing. IoT is a concept whereupon all the devices are connected to the Internet.

In order to support IoT communications, several technologies have been developed. Low Power Wide Area Network (LPWAN) are low power consumption technologies that provide long range communications. One of these technologies is LoRaWAN.

The focus in this thesis is the development of a Low-cost LoRaWAN gateway. This device receive data from end-devices, like sensors, and sends it to the cloud.

To develop this gateway LoRa single-frequency modules have been used. Using modules that are only capable of receiving in one frequency simultaneously creates two main issues, presented below.

The first is the vulnerability to jammers. A jammer can cause intentional interference and affect the proper operation of the system. To solve this issue, a frequency-hopping mechanism was implemented. Using this mechanism, the communications are constantly changing the frequency to avoid jamming. To solve this issue, were also implemented on the gateway two LoRa modules, each one capable of operating in different frequencies.

The second issue was to assure the correct sending of priority messages. This was done by implementing a time-slot reservation mechanism. Using this mechanism, a device can reserve a period in time where only it can send messages.

2. Background

LPWAN are communication technologies that allow a long range communication with low power consumption. These technologies are essential to IoT applications because they need a robust communication links to send their data, and the communication cannot spend much power as IoT devices are typically battery powered.

The frequency band is an important characteristic of any wireless technology, namely LPWAN technologies. There are frequencies that are free for anyone to use in any application, frequencies that are also free but reserved to certain application and even frequencies that need to be bought from the country regulator to be used.

On the other hand, there are technologies which are open source and can be used by everyone without licensing, such as LoRaWAN and NarrowBand-IoT (NB-IoT). Sigfox is an example of a proprietary
technology, therefore can only be used with licensing.

Long Term Evolution (LTE) LPWAN are technologies developed by the 3rd Generation Partnership Project (3GPP) to use licenced bandwidths. NB-IoT is the more recent technology and the one that fits better with IoT as it provides the largest coverage and low power consumption.

Sigfox is a company that provides a communication service with a proprietary protocol. Sigfox protocol uses the free 868MHz band (Europe) with a Ultra Narrow Band (UNB), providing a maximum bitrate of 100 bits per second. Sigfox also limits the number of uplink messages sent per day to 140.

LoRa stands for Long Range and is a modulation protocol designed to provide long range communication, using a variation of Chirp Spread Spectrum (CSS) with integrated Forward Error Correction (FEC). By spreading the signal in the time domain, it is possible to reduce the Bit Error Rate (BER) and achieve long distance communications. LoRa is an open source protocol, in other words anyone can develop a LoRa network without a license.

LoRaWAN is a Media Access Control (MAC) protocol for high capacity networks. A typical LoRaWAN network is composed by four components:

1. End-device: Collects the data using sensors and sends it to the gateway.
2. Gateway: Receives the data from the network server and sends this data through the air to the end-device and vice versa.
3. Network server: Manages several gateways assuring that no downlink duplicates are sent to the air and avoiding collisions.
4. Application server: Processes and stores the data. This component can send data to the end-devices through the network server/gateway.

LoRaWAN defines protocols to implement security from the end-device to the application server. Several classes of devices according with their power consumption is also defined by LoRaWAN protocols. LoRaWAN defines several channels in the frequency domain. Each channel is independent and communications in one channel do not interfere with other channels.

There are two kinds of gateways. The first type that has the ability to listen to all the LoRa channels. The second type are the low-cost gateways that use a single channel module, therefore can only receive on one channel at a time.

A low-cost gateway with multi-channel receiving ability, by using several single frequency modules was built. As a starting point for this project a DIY low-cost gateway tutorial developed at the French University of Pau and Pays de l’Adour [7] has been used. This is a single-channel low-cost gateway.

3. Implementation
This chapter presents the development of the LoRa gateway, including the gateway-driven frequency-hopping scheme, as well as the time-slot reservation algorithm that guarantees collision-free access for higher priority traffic.

3.1. System Architecture
The developed system has four components as defined by LoRaWAN. In this implementation network server also manages the frequency-hopping and time-slot reservation algorithms, but has no multi-gateway management function. Therefore, the network gateway was incorporated in the gateway, as expressed in Figure 1.

![Figure 1: System architecture.](image)

3.2. Channel Management Algorithms and Protocols
This thesis adds two important extensions to LoRaWAN, gateway-driven frequency-hopping and time-slot reservation. To add these extensions to LoRaWAN, the related control messages are sent with a message type (MType) different than the one used for data messages. MType is a 3 bits MAC header field defined by LoRaWAN and can take different values. Table 1 presents the values the MType can take according with LoRaWAN.

<table>
<thead>
<tr>
<th>MType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Join Request</td>
</tr>
<tr>
<td>001</td>
<td>Join Accept</td>
</tr>
<tr>
<td>010</td>
<td>Unconfirmed Data Up</td>
</tr>
<tr>
<td>011</td>
<td>Unconfirmed Data Down</td>
</tr>
<tr>
<td>100</td>
<td>Confirmed Data Up</td>
</tr>
<tr>
<td>101</td>
<td>Confirmed Data Down</td>
</tr>
<tr>
<td>110</td>
<td>RFU</td>
</tr>
<tr>
<td>111</td>
<td>Proprietary</td>
</tr>
</tbody>
</table>

Table 1: MType message types according with LoRaWAN (extracted from [17]).
Control messages regarding these extensions use MType 111, which refers to proprietary messages. Data messages use 010 for uplink and 011 for downlink. For data messages that require acknowledgement 100 is used for uplink and 101 for downlink.

3.2.1 Time-Slot Reservation Algorithms and Protocols

The end-devices use the beacons sent by the gateway to synchronize. This means that the end-devices count the time since the last beacon to calculate when they can send messages. The implemented protocol divides the time into equal cycles. A new cycle starts when a beacon is received. The end of each cycle is reserved for end-devices who reserved time-slots.

The average Time on Air (ToA) for each message was calculated as 1155 ms. Taking into account a duty cycle of 1%, a device should send a message every 115.5 or more seconds.

The gateway cycles should have at least twice this value, to provide enough time for uplink messages, so 240 seconds (4 minutes) is the period selected for each cycle. From these 240 seconds, 10% were reserved for TDMA, that is, during this period only authorized end-devices can transmit. Figure 2 represents the superframe structure.

![Figure 2: Proposed superframe structure.](image)

According to the used transmission parameters, the mean time on air of each message is 1155 ms. To avoid collisions between different time-slots a security gap between time-slots has been set. Therefore, each time-slot has a duration of 2 seconds. The 240 seconds available for time-slots divided by these 2 seconds makes 12 time-slots available for reservation.

According with the defined protocol the end-devices know that 216 seconds after the beacon they can only send messages if they have a reserved time-slot. After receiving a new beacon, they are allowed to send messages again for 216 seconds.

To reserve a time-slot the end-device must send a request to the gateway. After this request, the gateway should answer with the time-slot the device should use.

The request message must include the end-device identification. In this way, the gateway knows which device is requesting a time-slot. LoRaWAN specification includes this as DevAddr (Device Address). The second field of the request is Period, which is an 1-Byte integer. This field corresponds with the number of cycles the end-device need the time-slot to be reserved. Table 2 represents the structure of the time-slot request.

<table>
<thead>
<tr>
<th>Table 2: Request time-slot message structure.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Bytes</td>
</tr>
<tr>
<td>DevAddr</td>
</tr>
</tbody>
</table>

A soft reservation scheme is adopted, whereby the time-slot reservation is only valid for a maximum of 3600 seconds, i.e. 15 cycles of 240 seconds. This maximum assures that a time-slot doesn’t stay reserved even if the end-device for any reason stops working. For this reason, there is no message to free the time-slot as it will be automatically freed. If the end-device intends to keep the time-slot reservation, it will have to send a request message before the end of the expiration time of the previous reservation.

The reply message must contain the Device Address (DevAddr) and the assigned time-slot (TimeSlotAss). The TimeSlotAss field is a 1 Byte unsigned integer, with values from 0 to 12. If this field take the value 0, it means that there are no time-slots available. The values from 1 to 12 correspond to the time-slot reserved for that device. For example, number 2 means that the device can send a message on the second time-slot, that is, 226 to 228 seconds after the beacon. Table 3 represents the structure for the reply of the time-slot request.

<table>
<thead>
<tr>
<th>Table 3: Time-slot request reply structure.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Bytes</td>
</tr>
<tr>
<td>DevAddr</td>
</tr>
</tbody>
</table>

3.2.2 Frequency-Hopping Algorithms and Protocols

The gateway will have one or more LoRa modules, each module working on a different channel. A module sends a beacon with the frequency-hopping channel and stays tuned to the new channel for 240 seconds. After this period, the module sends a new beacon and switches to other channel, keeping the loop.

Two communications can happen simultaneously in the same channel as long as they use different spreading factors. This way two operating modes were implemented:

1. Shared channels mode: The different modules have different spreading factors and each module can operate on all channels available without interference. This system provide the best spreading factor to each sensor. For example a sensor far from the gateway will need a...
higher spreading factor to reduce BER. On the other hand, if the sensor is close to the gateway, the spreading factor can be lower, reducing the transmission times.

2. Non-shared channels mode: The different modules have the same spreading factor, but different channels. The descriptions that follow describe how the channels are divided to ensure that two modules are never in the same channel.

LoRaWAN defines a message to be sent from the gateway to the end-device containing the channel changing command. The name of this is NewChannelReq and its message structure is represented in Table 4.

Table 4: NewChannelReq structure according with LoRaWAN (extracted from [17]).

<table>
<thead>
<tr>
<th>1 Byte</th>
<th>3 Bytes</th>
<th>1 Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChIndex</td>
<td>Freq</td>
<td>DrRange</td>
</tr>
</tbody>
</table>

The Channel Index (ChIndex) is the number of the channel being created. It adopts values between 0 and 15. The frequency (Freq) is an unsigned integer of 24 bits. The real value of frequency is Freq x 100, allowing a frequency range between 1.67 GHz to 100MHz in 100 Hz steps. The Date Range (DrRange) specifies the allowed bandwidths. This Field splits in two 4 bits indexes represented in Table 5.

Table 5: DrRange Structure according with LoRaWAN (extracted from [17]).

<table>
<thead>
<tr>
<th>MinDr</th>
<th>MaxDr</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bits</td>
<td>4 bits</td>
</tr>
</tbody>
</table>

Minimum Data Range (MinDr) is the minimum Bandwidth this channel can support. Maximum Data Range (MaxDr) is the maximum Bandwidth this channel can support. For example, for SF12 / 125 KHz the data rate is DR0. If this is the only configuration available, then both MinDr and MaxDr will be DR0. This way DrRange will take the value 0x00.

3.3. Security

LoRaWAN defines a protocol to implement security from the end-device to the application server. This protocol uses the AES-128 algorithm to encrypt the message with two different keys.

The first key (NwkSKey) encrypt all the message between the end-device and the network server. The second key (AppSKey) encrypt only the data between the end-device and the application server. In the stated architecture, the network server and the gateway have no access to the data encrypted with AppSKey. Therefore, control messages as the beacons are encrypted using only the NwkSKey. Messages containing data, for example, sensor readings from the end-device are encrypted with the both keys.

3.4. Gateway Implementation

This subsection presents the gateway hardware components, and the software implemented algorithms.

3.4.1 Gateway Hardware

The LoRa gateway has multiple single channel LoRa modules, each one giving the ability to receive in one channel. A Raspberry Pi was used, which communicates with the LoRa modules by SPI and sends the data to the cloud through the Ethernet cable. Figure 3 represents this components.

Figure 3: Hardware architecture.

SPI (Serial Peripheral Interface) is the protocol used to communicate between LoRa modules (SPI slave devices) and the Raspberry Pi (SPI master device). In the SPI architecture, a master device that communicate with each slave must exist – slaves cannot communicate to each other.

The SPI architecture defines four electronic pins:

1. SCLK or SCK (Serial Clock) pin that synchronizes all the devices: this pin is shared by all the SPI slaves.
2. MOSI (Multiple Output Single Input) and MISO (Multiple Input Single Output) pins used for data transfer: these pins are shared between all the SPI slaves.
3. Chip Select (CS) pin, also named as Slave Select (SS): used for coordinate between several masters, so they don’t use the shared pins simultaneously.

According with the SPI architecture CS pin could also be shared between different modules. However, to make it simpler to implement, it was decided to adopt the implementation stated before. CS pins
are controlled by software to select each module the gateway is communicating with.

3.4.2 Gateway Software

As a starting point for this project a DIY low-cost gateway tutorial developed at the French University of Pau and Pays de l’Adour [7] has been used. This tutorial is about building a gateway with a Raspberry Pi and a single LoRa module. It is not resilient to channel jamming, as multi-channel, frequency-hopping and time-slot reservation capabilities are not developed which makes it very unreliable in situations of high load, where channels are often busy.

This tutorial’s code uses a library to control generic LoRa modules such as the RFM95 and SX1276 made for Arduino (SX1272) with a layer that adapts this library to work on Raspberry Pi (ArduPi). The main software (LoRa_gateway) uses a Setup function to configure the module and then a Loop function to check for incoming messages. Both functions had to be adapted to support multiple modules. The main software also sends commands to a Python program, through a pipe (Post_processing) that will decrypt incoming messages. Figure 4 represents the several libraries and the way they relate.

![Figure 4: LoRa gateway libraries architecture.](image)

The main software first task is to launch another thread, the Beacon Processing Thread. The task of the latter, as the name implies, is to process the beacons. After this, the main software will setup the different modules (in this case there are two), and then enter an infinite loop, checking for new messages, received from the both channels available.

The Setup function used in main software configures the LoRa module with all the parameters defined by the network Server, which should be used to receive messages such as the frequency, the spreading factor, the bandwidth, the preamble length, the power among others.

The first step of the Loop function is to call a SX1272 procedure that assures the correct module is being controlled – this procedure will be detailed later. The second step is to restart a board if an error is detected. After ensuring the board is working normally, the Loop function checks for incoming messages. The message received is send through a pipe to Post_processing to be decrypted. Then, if there is a downlink message to be sent, the Loop function will send it. There is no risk of losing messages while processing other modules, as they are temporarily stored in the RFM95 module.

The Beacon Processing Thread is part of the network Server and is responsible for processing the beacons according with the specified rules. There is a time counter that assures that the beacon transmission procedure is called every 240 seconds, for each module. This thread is also responsible for deciding which module the beacon will be sent from and the next channel will be taken by the module. This thread only sends the beacon transmission orders – the process of sending the beacon is taken by the gateway.

The first step is to initialize the module counter. This counter will save the next module to be processed. After this, the timeCounter will also be initialized. This counter will save the time since the last beacon.

The second step is to wait for the time counter to reach 120 seconds. By doing this, the gateway will change a module channel every 120 seconds, alternately. There are two modules, therefore, each module will change channel every 240 seconds.

After ensuring the timeCounter reached 120 seconds, the CalculateChannel procedure will be called. After calculating the channel, a beacon command with the calculated channel will be sent. After sending the beacon the gateway will change the channel.

To finish the procedure, the ModuleCounter is toggled and the loop restarts.

3.5. End-Device Implementation

In order to test the developed gateway, an end-device was also implemented. This end-device presents the following features:

1. Was developed using Arduino software found on tutorial developed at the French University of Pau and Pays de l’Adour [7].
2. Changes the communication channel according to the beacon instructions. This frequency-hopping mechanism can be disabled.
3. Can reserve time-slots and only sends data in a time-slot if that time-slot is reserved for it.
4. Only sends data if the last beacon was correctly received.

If no beacon is received within 240 seconds, the end-device knows that a beacon is missing, therefore it will not send messages until receiving a beacon. To avoid being tied on a channel for not receiving the beacon, the end-device will automatically change channel after 5 cycles, i.e., 1200 seconds.
This procedure is only taken when the frequency-hopping mechanism is enabled. This implementation will be used further to test the time-slot reservation and frequency-hopping mechanisms.

The RFM95 module works with 3.3 Volts. This is the same voltage as the Raspberry Pi, but most Arduino boards work with 5 Volts. Therefore, there was the need for an Arduino board capable of working with 3.3 Volts and with a considerable processing power, as the low range Arduino has not power enough to encrypt and decrypt the messages in real time. Arduino M0 was found to suitably these requirements.

4. Results
A functional and performance evaluation of the developed system. The first set of tests aims to evaluate the communications range. The second set of tests aims to evaluate the randomness characteristics of the frequency-hopping sequences. The third set of tests aims to evaluate the implemented time-slot reservation and frequency-hopping mechanisms.

4.1. Communications Range Evaluation
Range evaluations were made to assess the gateway’s ability to receive messages over long distances. These evaluations were divided into two. The first evaluation measures how the signal power decays with the distance. The second evaluation measures the maximum distance an end-device can be from the gateway and still send successfully messages to it.

As mentioned before, one of the possible applications of LoRa technologies is the farm monitoring across vast areas. This application was taken into account and the tests were done in rural environment. These areas, typically have no buildings. As there are no attenuation factors, large ranges are expected.

4.1.1 Decay of Received Signal Power with Distance
To measure how the signal power decays with the distance one end-device and the developed gateway were used. The end-device was configured to send messages periodically. These messages were received by the gateway, where the received power was measured.

To perform this evaluation, the gateway was placed in a static position and the end-device was displaced at several positions, increasing the distance from the gateway. For each position, 5 signal power measurements were taken, and the mean values have been registered. The graph on Figure 5 shows the measured values. There is also represented in grey a trendline. Equation 1 shows the trendline equation. Note that the trendline follows the Free Space Path Loss (FSPL) equation. This was expected as the transmissions were done with line of sight.

\[
\text{Trendline} = -21.02 - 20 \log_{10} \text{Distance} \tag{1}
\]

The measurements made show that in rural environments, where there are no obstacles. Communications with distances up to 1 km can be easily achieved. For these distances, even with small obstacles like trees, the communications can be done with success.

4.1.2 Maximum Range
The maximum distance an end-device can be from the gateway and still send messages to it was measured. LoRa technology specifications claim a 15 km range in rural areas.

To perform this evaluation, the gateway was placed in a static position and the end-device was displaced at several positions, increasing the distance from the gateway. Note that points distanced several kilometers without obstacles are not common. Therefore, in order to test the success of the communications across long distances a previous planning had to be done. This planning consisted in looking for end-device positions where it was possible to get line of sight to the gateway.

Several high altitude spots have been pre-selected. After this, a Google Maps script has been used. This script was used to measure the altitude in several points in the path between that spot and the gateway, for each of the pre-selected zones.

The path with the longest distance, and clear from obstacles, that was found was 4.9 km long. Figure 6 shows the altitude graph of the path. The lowest point represents the location of the gateway. The highest point represents the location of the end-device.
The gateway was placed at 6 meters higher than the ground level as it was placed inside a house, with the antenna on a window. The end-device altitude was one meter higher than the ground as it was being held by a person.

Figure 6 shows that there are no obstacles to the line of sight. Therefore this point was tested and messages were successfully sent to the gateway. The signal power measured at the gateway shows that even bigger distances could be tested.

4.1.3 Data Analysis

The results taken from this evaluation show that in a rural environment ranges with one kilometer can be easily achieved, even with small obstacles like trees. Larger ranges up to 5 km can be achieved with line of sight.

With the range 1km assured, it is possible with a single gateway to cover a 314 ha farm. This proves that the developed gateway is suitable to receive data from sensors spread across vast areas, in rural environments.

4.2. Assessment of the Random Characteristics of the Frequency-Hopping Sequences

The randomness of the frequency-hopping sequences is important to prevent jammers from easily predicting them. This would make the frequency-hopping worthless to prevent jamming attacks.

In this section, tests will be presented, which were made regarding the randomness of frequency-hopping sequences, addressing two aspects. The first aspect is related with how uniform the distribution is. The second aspect is the randomness of the results in a graphical view – this second aspect is more qualitative.

Two modes of operation were introduced above. The first mode uses shared channels and different spreading factors for each LoRa module. The second mode uses the same spreading factor and different channels. These two models were tested to assess randomness of the generated frequency-hopping sequences.

4.2.1 Uniform Distribution Results

This subsection presents the uniform distribution results of two alternative random number generators: based on current timestamp, and based on thermal noise on Raspberry Pi pins.

To analyze if the results random sequences match a uniform distribution, chi-squared test was used. This test is represented by Equation 2, where lower values mean a more uniform distribution.

\[ \chi^2 = \sum_{i=1}^{k} \left( \frac{Y_i - \frac{n}{k}}{\frac{n}{k}} \right)^2 \]  

In the equation, \( k \) is the number of available options, which are 8 in shared channel mode, and 4 in the non-shared channel mode. \( n \) is the sum of the generated random numbers, which in this example is 400. \( Y_i \) takes the number of times each channel number has been randomly generated.

Table 6 summarizes the chi-squared results. The results show that the thermal noise based algorithm is slightly more compliant with the uniform distribution.

<table>
<thead>
<tr>
<th></th>
<th>Shared channel</th>
<th>Non-shared channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time based algorithm</td>
<td>8.8</td>
<td>6.92</td>
</tr>
<tr>
<td>Thermal noise based algorithm</td>
<td>3.88</td>
<td>7.24</td>
</tr>
</tbody>
</table>

4.2.2 Randomness Graphical View

It is important to bear in mind that the chi-squared function measures the compliance of the results with the uniform distribution. The results can be perfectly distributed, without being random. Take for example the binary sequence 01010101, which is perfectly distributed, but not random. This difference can be easily seen graphically. Figures 7 and 8 show an example taken from [21] of the difference between a real random algorithm, like the based on thermal noise on Raspberry Pi, and the custom based algorithm, like the based on the current timestamp.

![Figure 7: Real random algorithm (extracted from [21]).](image)
In order to graphically assess the randomness of both algorithms proposed in this dissertation, the results were transformed in a graphical representation. This was done using an open source software named Processing. Generated channel numbers are represented in a gray scale, considering channel 0 as white and channel 7 as black, the other channels taking different shades of grey.

To achieve representative results, the samples must be much more than the used on the chi-squared tests. Each test comprised 8100 cycles, resulting in 90 pixel squares. Note that a 8100 cycle test takes several days to complete. For this reason, and because the main purpose here is to test the random algorithm, only the shared channels mode was tested.

Figures 9 and 10 show the results of the random algorithm based on timestamp and the random algorithm based on thermal noise, respectively.

Despite the number of samples was widely increased, the results still show no evidences that the thermal noise random algorithm is better than the classic random algorithm available for the Raspberry Pi. None of the above graphical representations show patterns as the examples presented in Figures 9 and 10. This is an indicator that both algorithms generate numbers with a good level of randomness.

4.3. Performance Evaluation of the Time-slot Reservation and Frequency-Hopping Mechanisms

In this section, the time-slot reservation and frequency-hopping mechanisms were evaluated. There were two different evaluations. The first evaluates the time-slot reservation mechanism. The second evaluates the frequency-hopping mechanism. These evaluations were done by comparing the number of successfully received messages.

For each mechanism, evaluation different scenarios were considered in the tests.

As mentioned before, a beacon is sent by the gateway every 240 seconds. This 240 seconds corresponds to a cycle. If the end-device does not receive the beacon, it does not send any message in that cycle.

For each scenario, 5 tests were done, each one with a duration of 100 cycles.

4.3.1 Time-Slot Reservation Results

The success rate when sending a message outside a time-slot is compared against the success rate when using a reserved time-slot.

For this evaluation, the frequency-hopping mechanism was disabled. As such, it was possible to compare the success rates of the time-slot mechanism itself. With the frequency-hopping mechanism disabled, beacons are only used by the end-device to synchronize with the gateway. This synchronization is used by the end-device to calculate time-slot schedule.

As each 100 cycles test take almost 7 hours to complete, the tests were automated. The messages sent by the end-devices contained an identification code. These identification codes are saved and processed by the application server, which calculates the success rates presented below.

For this evaluation, the scenario considered two types of devices:

- The sender end-device, sending useful messages to the gateway. After receiving a beacon this end-device sends two messages, one on the reserved slot, the other on the contention-based time interval. The string "reserved" is the identification code for messages sent on reserved time-slots and the string "normal" is the identification code for messages sent outside reserved time-slots.
- The interfering device, which is at the same time, sending two messages per cycle. These
messages are sent at a random time, during non-reserved periods, to simulate the real use of the gateway. The messages sent by this end-device contain the identification code "other", and a number to identify the interfering device that send the message.

To evaluate the time-slots reservation mechanism, three scenarios were be tested. In the first scenario, there were no interfering devices, in the second scenario a single interfering device was used and in the third scenario two interfering devices were used.

Table 7 summarize the time-slot reservation results. It is clear that, when the network is being used by more than one end-device, the time-slot reservation is effective assuring the correct transmission of priority messages in the presence of additional end-devices.

The tests show that, when more than one device is operating on the network, the packet delivery rate of messages in reserved time-slots is much higher than using the contention-based periods. In fact, the success is 16.2 and 22.2 percent higher, for networks with one and two interfering devices, respectively. This allows extrapolating that with higher end node densities and higher traffic rates, the performance improvement due to the time-slot reservation mechanism will tend to increase quite significantly.

Table 7: Summary of the results in the three scenarios tested.

<table>
<thead>
<tr>
<th>Reserved time-slots</th>
<th>Contention-based periods</th>
</tr>
</thead>
<tbody>
<tr>
<td>No interfering device</td>
<td>96.8%</td>
</tr>
<tr>
<td>One interfering device</td>
<td>89.4%</td>
</tr>
<tr>
<td>Two interfering devices</td>
<td>85.8%</td>
</tr>
</tbody>
</table>

4.3.2 Frequency-Hopping Results
For the frequency-hopping evaluation, two scenarios have been tested. In the first scenario, the frequency hopping was disabled and the end-device was using the same channel as the jammer. For the second scenario, the frequency-hopping was enabled.

For this evaluation, the time-slot reservation mechanism was disabled. As such, it was possible to compare the success rates of the frequency-hopping mechanism itself. For this evaluation, two devices were used:

- The end-device, which sent messages to the gateway. After each beacon is received, this end-device sent two messages, at a random time.
- A jammer implemented as an end-device sending tiny messages in a channel. The purpose of the jammer was only to cause interference.

Therefore, the messages it sent were not LoRaWAN messages and could not be be not received by the gateway.

The channel targeted by the jammer was the same used by the end-device and the gateway, in the scenario without frequency-hopping. This simulates a situation where a jammer is being used, and allows to evaluate the gain of a frequency-hopping algorithm in such situations. In this scenario, no messages from the end-device have been received. The jammer successfully destroyed all the messages.

Table 8 shows the number of received messages with, and without the frequency-hopping mechanism. The number of received messages corresponds directly to success rate as a percentage.

Table 8: Number of received messages with, and without the frequency-hopping mechanism.

<table>
<thead>
<tr>
<th>Frequency-hopping enabled (m)</th>
<th>Frequency-hopping disabled (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>36</td>
</tr>
<tr>
<td>Test 2</td>
<td>39</td>
</tr>
<tr>
<td>Test 3</td>
<td>45</td>
</tr>
<tr>
<td>Test 4</td>
<td>22</td>
</tr>
<tr>
<td>Test 5</td>
<td>53</td>
</tr>
<tr>
<td>Average</td>
<td>39 (±17)</td>
</tr>
</tbody>
</table>

Table 8 shows that the success rate in the scenario with frequency-hopping enabled is higher than the success rate with the frequency-hopping disabled. Nevertheless, 39% success rate is low. This can be solved by implementing a mechanism to detect jammers, allowing the gateway to avoid those channels. This improvement is left for future work.

5. Conclusions
This dissertation addresses the problem of LoRaWAN implementation in small IoT installations, such as sensor networks for agriculture monitoring and control. To solve this problem, a low-cost gateway architecture is proposed. In order to improve the system resiliency and quality of service, the proposal also entails the development of time-slot reservation and frequency-hopping mechanism, which are identified in the literature as needed improvements to the LoRaWAN specification. Time-slot reservation was proposed to ensure that priority data is sent without danger of collisions. Frequency-hopping was proposed to avoid interference and jamming.

Several LPWAN technologies were analyzed with special focus on LoRa and LoRaWAN which were the selected for this proposal. Compliance with the current LoRaWAN specification was kept, whenever possible.

The gateway has been tested regarding its performance. The performance tests address the commu-
niations range, the randomness of the frequency-hopping scheme, as well as the performance improvements introduced by the time-slot reservation and frequency-hopping schemes.

Tests made with the gateway also shown that it can provide communications with line of sight within at least 4.9 kilometers. These tests have also shown that, for distances lower than 1 kilometer, communications can be done through small obstacles like trees.

Results made to both extensions made show that their use solve the problems they were implemented for.

Time-slot reservation turned out to be effective improving the packet delivery rate when more than two end nodes are concurrently transmitting to the gateway. For this situation, sending a message on a reserved time-slot offers a 22.2% higher success rate, which is expected to be even higher as the number of contending devices increases.

Frequency-hopping results show that, whenever using frequency-hopping and there is a jammer affecting one of the two available channels, the success rate is 39%.

5.1. Future Work
As mentioned before, whenever using frequency-hopping and there is a jammer, the success rates are just 39%. To improve this, a mechanism could be developed to detect the jammed channels, allowing the gateway to avoid them.

The number of channels the gateway is listening to is limited by the number of LoRa modules used. If the number of end-devices is too large, so that the gateway can not receive all the messages, more LoRa modules can be added. This can increase the cost the low-cost gateway, but one of the many LoRa modules from the end-devices can be used.

The developed time-slot reserving mechanism is not flexible. Only 12 time-slots can be reserved and the cycles have always the same length. An improvement could be done in a way that would be possible to reserve a time-slot at any time, and not just at the end of a 240 seconds cycle.

References
[8] Link Labs: Low Power Wide Area Networks (LPWAN) and LTE-M for IoT (December 2016). Retrieved from https://www.link-labs.com/