OULU UNIVERSITY OF APPLIED SCIENCES

Ukko-Pekka Peura

LORAWAN OPTIMIZATION FOR A BATTERY POWERED SENSOR NETWORK

LORAWAN OPTIMIZATION FOR A BATTERY POWERED SENSOR NETWORK

Ukko-Pekka Peura
Bachelor's Thesis
Spring 2018
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Ukko-Pekka Peura
Title of the bachelor's thesis: LoRaWAN Optimization for a Battery Powered Sensor Network
Supervisors: Marko Hietala (Etteplan Oy), Jaakko Kaski (OAMK)
Term and year of completion: Spring 2018 Number of pages: 55 + 7 appendices

---

The aim of this bachelor's thesis was to study LoRaWAN limitations and power consumption in real life working conditions for Etteplan Oy. Etteplan Oy was interested in using LoRaWAN in its future Internet of Things (IoT) projects and thus needed to know more about its suitability on various use cases, with a strong emphasis on battery powered devices. This thesis aims to provide guidelines for designing and developing LoRaWAN based embedded devices and networks. The thesis will also outline edge cases and problems that might otherwise go unnoticed when developing such a device.

A full protocol specification and regional parameters for LoRaWAN were available from The LoRa Alliance. These documents were used to study how LoRaWAN works and how some of its features are implemented. The documentation also contains some recommendations for using this technology. For things that were not explained in the specification, a test on the actual hardware and software was performed to test how the protocol works. Real world testing for range was also performed on suburban and open areas. The software used on end devices was built with Mbed OS 5.8 with an integrated LoRaWAN stack.

The result of the study was that LoRaWAN's suitability must be carefully investigated on case basis. The device designer must know several design parameters before considering how well LoRaWAN would fit into this design. Such design parameters include: the wanted battery life, required wireless transmission range, payload size and message frequency. LoRaWAN has clear advantages when transmitting very small amounts of data infrequently, and a when good object penetration is required. On the other hand, LoRaWAN is not suitable for designs where a high bandwidth or a fast packet rate are required. Power consumption is also very dependent on the use case but power consumption can also be easily predicted if that is required by the design.

Keywords: LoRaWAN, LPWAN, LoRa, Internet of Things, IoT, Mbed, wireless communication, power consumption

PREFACE

This thesis was done during the spring of 2018 for Etteplan Oy. Etteplan has made it possible for the author to work with them since autumn 2017 and it also made it possible to write this thesis. Working almost exclusively with LoRaWAN and Mbed OS since autumn of 2017 has given me the required knowledge to make this thesis come together.

Special thanks to Marko Hietala for doing an awesome superior work at Etteplan. I would also like to thank Jaakko Kaski from Oamk who was supervising my thesis work and for his mentorship on physics and mathematics.

Oulu, 21.4.2018
Ukko-Pekka Peura

TABLE OF CONTENTS

VOCABULARY

ABP	Activation By Personalization
ADC	Analog-To-Digital Converter
ADR	Adaptive Data Rate
CSS	Chirp Spread Spectrum
EIRP	Effective Isotropic Radiated Power
ERP	Effective Radiated Power
FSK	Frequency-shift keying
ISM	Industrial, Scientific and Medical
IoT	Internet of Things
LPWAN	Low-Power Wide-Area Network
LoRa	Long Range
MAC	Media Access Control
OTAA	Over-The-Air-Activation
RTOS	Real-Time Operating System
SNR	Signal-to-noise ratio
ToA	Time on Air

1 INTRODUCTION

The number of IoT devices is growing fast, and the variety of use cases is also growing. IoT is all about connectivity and moving data between devices. As smart devices are being used almost everywhere, they still have to be able to communicate and move data. A device designer must choose between many different ways of communicating both wired and wireless. Wireless networking is a very common and flexible way for IoT appliances to transfer information. All wireless radio technologies have their own characteristics for range, available bandwidth, and power consumption. Choosing the right radio technology for the right use case is important.

LoRaWAN is a Long Range (LoRa) Low-Power Wide-Area Network (LPWAN) technology which targets at the market by providing long range and low power consumption wireless networking. LoRaWAN uses a proprietary LoRa modulation owned by Semtech. The media access control (MAC) layer defined by an open LoRaWAN specification works on top of the proprietary physical implementation. The specification is maintained by LoRa Alliance. LoRaWAN operates on an unlicensed radio spectrum and does not require any additional licencing fees to be paid. (1.)

Etteplan has been an early investor in the LoRaWAN technology by working closely with ARM and porting Semtech's LoRaWAN stack to Mbed OS in 2017. Since LoRaWAN is still a new technology, it is in Etteplan's best interest to study its suitability for upcoming projects. By knowing LoRaWAN's limitations and best use cases, Etteplan and its customers will have the benefit to utilize Mbed OS with LoRaWAN when it is deemed to be the best solution for the problem at hand.

Several methods were used in order to understand LoRaWAN's pros and cons. Studying the protocol specification, regional parameters, and regulations were the foundations for this thesis. After a throughout study of these topics, there were still a number of uncertain issues which these documents do not explain or do not make straightforward to understand. To solve these uncertainties, tests

with real hardware on real working conditions were first planned and then carried out. All findings and results were then brought together for a conclusion.

2 LORAWAN INNER WORKINGS

In order to understand what parameters affect LoRaWAN end device's power consumption and performance, one must first understand how the LoRaWAN physical layer works and what limitations and features the MAC layer imposes. This chapter introduces what duties the physical layer and the MAC layer handle.

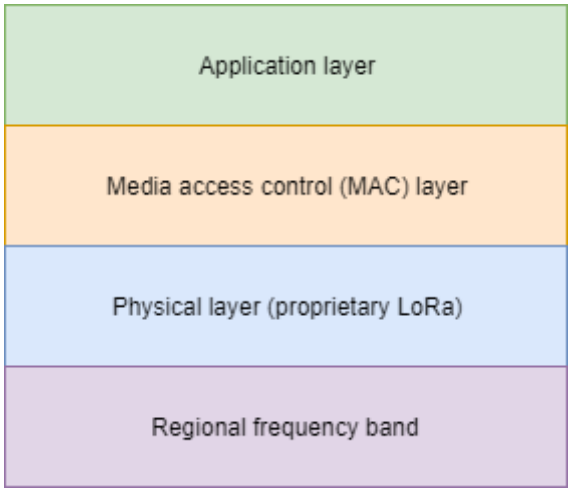

FIGURE 1. The LoRaWAN stack.

LoRaWAN is actually two technologies combined. At the physical layer, there is the proprietary LoRa modulation scheme and on top of that there is the open-standard MAC layer for the LPWAN implementation. These technologies together form the actual LoRaWAN stack implementation as seen in figure 1.

2.1 LoRa physical layer

On the physical layer, the proprietary LoRa modulation by Semtech is used. The LoRa modulation is a derivative of Chirp Spread Spectrum modulation (CSS). The modulation has constant amplitude and it sweeps across the entire bandwidth. The CSS modulation was originally developed to be used in radar systems. The modulation has few selectable parameters to tune its performance: modulation bandwidth, code rate, and spreading factor. CSS also has

relatively low transmission power and it is very resistant against jamming, multi-path propagation, and unwanted doppler effects. (2.)

The physical layer also contains gray indexing, data whitening, interleaving, and forward error correction to reduce effects of interference and poor radio conditions. An exact implementation is proprietary. (3.)

2.1.1 Modulation

LoRa modulates information into chirps. Chirps are constantly varying frequency signals. Rising frequency chirps are upchirps and decreasing frequency chirps are downchirps. The frequency bandwidth of a chirp is equal to the used channel bandwidth, meaning that a single chirp uses the entire bandwidth. (3.)

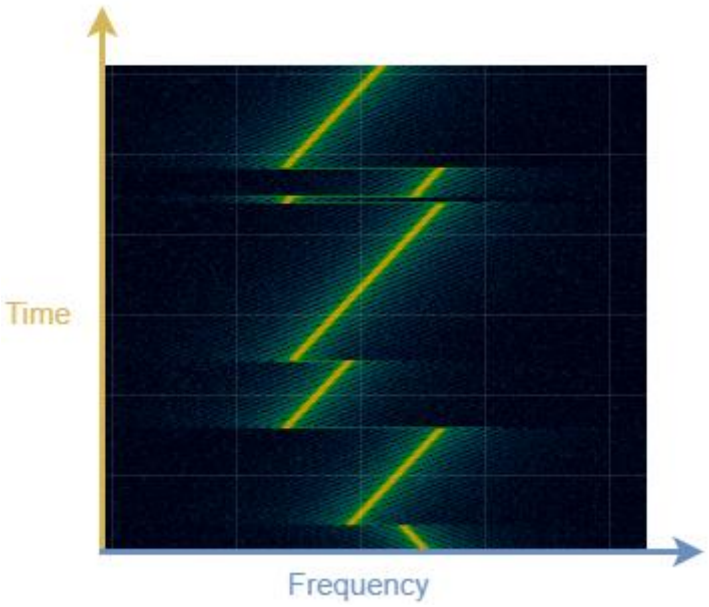

FIGURE 2. LoRa modulation as seen on a spectrogram plot. (4.)

Data is modulated into chirps by instantaneous frequency changes. Figure 2 shows the actual LoRa modulated signal received with a software defined radio. Modulation has also a preamble which consists of repeated upchirps in the beginning of a frame. The receiver will use this known sequence to adjust its receiver before demodulation. (3.)

2.1.2 Demodulation

Demodulating a spread spectrum signal requires the receiver to know how the expected signal has been spread across the spectrum. The receiver can then generate upchirp and downchirp signal patterns and then multiply the received signal with generated patterns to extract symbols. (3.)

TABLE 1. LoRa radio demodulation parameters. (5.)

SpreadingFactor (RegModulationCfg)	Spreading Factor (Chips / symbol)	LoRa Demodulator SNR
6	64	-5 dB
7	128	-7.5 dB
8	256	-10 dB
9	512	-12.5 dB
10	1024	-15 dB
11	2048	-17.5 dB
12	4096	-20 dB

Due to the spread spectrum nature of the LoRa modulation, it can be demodulated with a negative signal-to-noise ratio (SNR). Table 1 from LoRa radio module's datasheet shows the minimum SNR required for demodulation at different spreading factors. Being able to a receive signal below the noise floor is the key for LoRa's long range and good building penetration.

2.1.3 Bandwidth

A bandwidth is expressed as kHz. The LoRa modulation has no fixed bandwidth. It can operate on different bandwidth settings. The actual channel bandwidth is fixed. Semtech chipsets offer a wide bandwidth range to be used. For example the SX1276 radio chipset has a bandwidth range from 7.8 kHz to 500 kHz (6). The modulation the uses entire available bandwidth whereas upchirps or downchirps sweep across the entire channel bandwidth. Typical bandwidths specified by LoRaWAN regional specifications are 125 kHz, 250 kHz, and 500 kHz (7). A higher bandwidth means a higher data transfer rate.

2.1.4 Spreading factor

A spreading factor is an important aspect for the LoRa modulation. It represents the rate on which the signal changes frequency. The spreading factor is expressed as a number ranging from 7 – 12, representing the speed of a frequency change in a chirp. Higher numbers mean that the chirp lasts longer meaning that sweep across the bandwidth lasts longer (2.). Figure 3 illustrates how the duration of the chirp increases as the spreading factor increases.

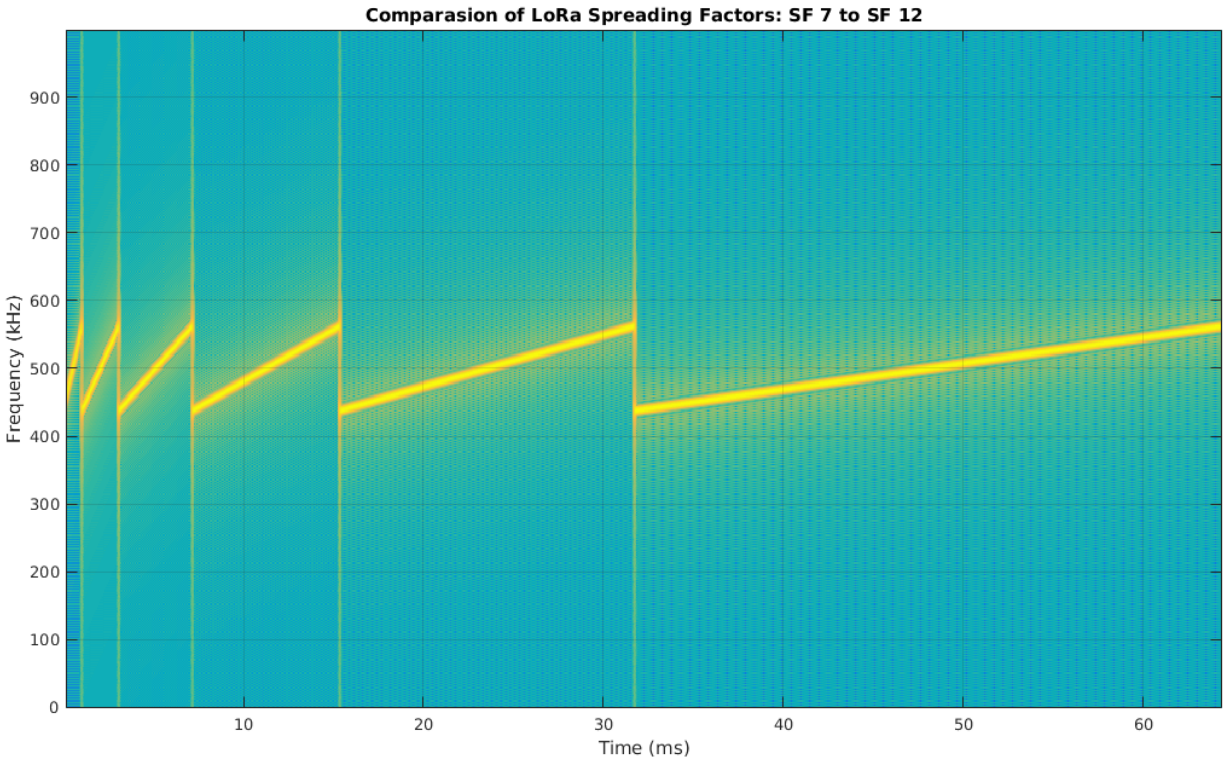

FIGURE 3. Spectrogram of different LoRa spreading factors. (8.)

The spreading factor affects the data transfer rate and demodulation SNR (Figure 2.). When data is spread more (a higher spreading factor), it can be demodulated with a lower SNR value. The spreading factor combined with the channel bandwidth forms the LoRa data transfer rate (2).

2.1.5 Code rate

LoRa contains a variable error correction scheme, which requires extra redundant data to be generated to the message. The code rate defines how many extra bits of data are encoded to a message when it is sent over the air. An increasing number of redundant bits allows error correction to correct more errors but increases a message size. (2). The code rate is expressed as k/n which means that for every k bit, the coder will generate n bits of data (9). LoRa has no default code rate but 4/5 is the most used one (10).

2.2 MAC layer

An MAC layer handles the actual LPWAN protocol work. The MAC layer protocol is defined by the LoRaWAN specification, and several implementations exist (11, 12). The MAC layer works on top of the proprietary LoRa physical layer. In most cases this means that MAC runs on a microcontroller which is connected to a LoRa radio module, which handles the physical layer.

2.2.1 Mbed OS

Mbed OS is an open source embedded operating system developed by ARM. Mbed OS is specifically designed for IoT applications. Mbed OS targets ARM Cortex-M microcontrollers and provides connectivity, hardware drivers, and a real time operating system (RTOS). (13).

Since the version 5.8, Mbed OS has an integrated software LoRaWAN stack compliant with the v1.0.2 specification. Mbed and its LoRaWAN stack are free to use and only require a development board with a LoRa radio that is supported by Mbed OS. This thesis uses Mbed OS for all LoRaWAN testing on the real hardware and only focuses on the v1.0.2 specification because there is no support for the 1.1 specification yet on Mbed. (14).

2.2.2 Device classes

As different devices have different connectivity and power requirements, the LoRaWAN specification specifies three different device classes. The classes are

named from A to C. The class A is mandatory and every MAC layer should implement it. Classes B and C are optional and do not have to be implemented. Different device classes have different communication proprieties.

- Class A
    - Bi-directional communications
    - Receives only shortly after each uplink transmission
    - Lowest power
- Class B (beacon)
    - Bi-directional communications
    - Receives shortly after each uplink transmission
    - Has scheduled receive windows
    - Extra receive windows increase power consumption
- Class C (continuous)
    - Continuous receiver (only closed when transmitting)
    - Highest power consumption

This thesis only focuses on the lowest power end device class A. (15.)

2.2.3 Receive windows

Class A devices can only receive downlinks after sending an uplink (see figure 4 below).

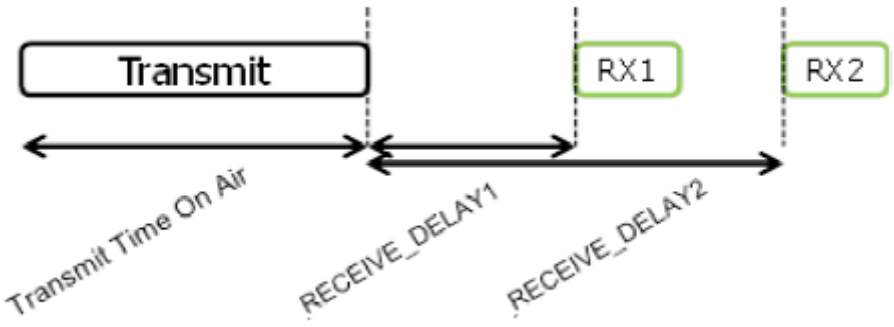

FIGURE 4. Receive window timing. (15.)

The device opens two short receive windows after each uplink even if the application layer is not waiting for any message. If a downlink is received on the first receive window, the second receive window is not opened. The first receive window uses same radio parameters as the uplink before it. It matches a spreading factor, code rate, bandwidth, and frequency. The second receive window has fixed parameters. (15).

2.2.4 Channel plan and channel selection

The LoRaWAN channel plan is a list of radio channels which a device or a gateway is using to communicate. The devices operating in Europe have the maximum of 16 channels in their channel plan. The channel plan contains information on all usable channels of the device. Each channel has the minimum and maximum usable data rate, frequency, and sub-band where the channel belongs. (7, 15.)

The channel selection must be done at random from all available channels. Channels can be unavailable or unusable if a channel's sub-band has no duty cycle left or the channel does not support the data rate that the device is trying to use. The channel might also be disabled by the network server with an MAC command. (15.)

2.2.5 Device commissioning

Device commissioning means connecting a device to a LoRaWAN network. The LoRaWAN specification has two different ways of doing this: Activation By Personalization (ABP) or Over-The-Air-Activation (OTAA). The device designer can freely choose between both methods and after the connection, devices behave exactly in the same way regardless how they were connected. (15.)

Both methods are discussed in detail in chapter 6.

2.2.6 Message types

The LoRaWAN specification has several different message types. Messages sent by the end device are called uplinks and messages sent by the gateway are downlinks. LoRaWAN has both unconfirmed and confirmed messages. Both

the end device and gateway can freely choose between unconfirmed or confirmed messages. Unconfirmed messages are only sent out once and they are never acknowledged in any way. Unconfirmed messages may be lost without any indication to the application. (15.)

Confirmed messages will always require a confirmation when they are received. When the end device sends a confirmed message, the server must respond with an acknowledge message immediately when the device opens an RX1 or an RX2 window. If the confirmation does not arrive within these windows, the message is considered lost. When the network server sends a confirmed downlink message to the end device, it will have to wait for the next uplink to receive a confirmation. The end device will not immediately respond. The confirmation bit is set on the next uplink sent by the end device. Figure 5 illustrates a confirmed message logic flow. (15.)

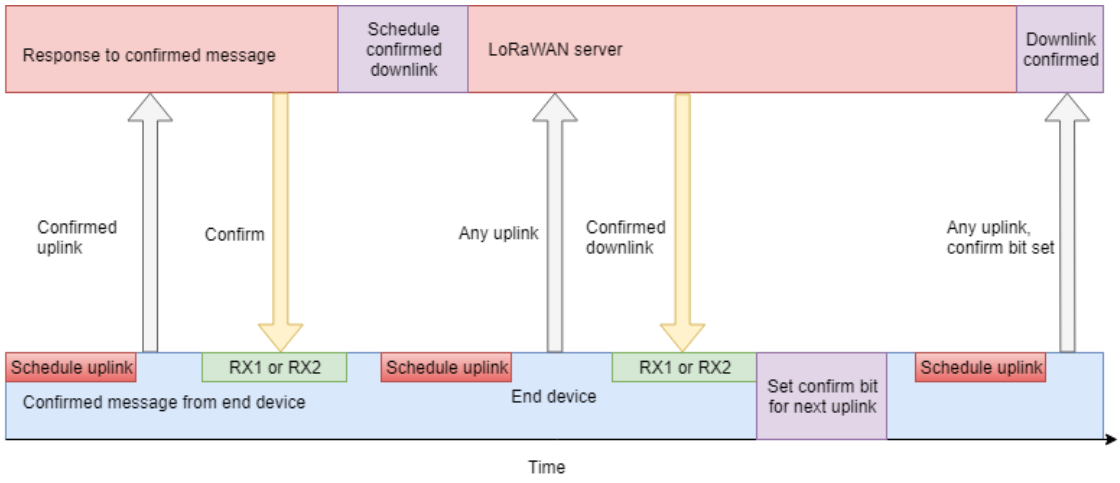

FIGURE 5. Confirmed message logic.

A protocol specification also contains proprietary messages. Proprietary messages are undefined but can be used. Proprietary messages are meant for extending the protocol with custom message implementations. If proprietary messages are being used, the device designer must ensure that both the end device and network server are running the same custom implantation that understands these messages. (15.)

2.2.7 Frame counters

In order to prevent replay attacks, LoRaWAN implements frame counters on both uplinks and downlinks. Both the device and server have to keep track of these message counters and ignore messages with the frame counter that has already been used. (15.)

2.2.8 MAC commands

LoRaWAN has a number of MAC commands that are used for network administration. MAC commands are exchanged exclusively between the network server's MAC layer and the end device's MAC layer. MAC commands are never visible to the application layer. Every MAC command has a corresponding answer and every command should be answered accordingly. The complete list and definitions for all commands can be found in the LoRaWAN specification. (15.)

2.3 Network structure

LoRaWAN uses a star-of-stars topology on its networks. Networks can have several gateways, and they all connect to a central server. Gateways only route messages from or to the central network server. Figure 6 shows a typical LoRaWAN network topology. Gateways do not run the LoRaWAN MAC layer. The central network server runs the MAC layer where all protocol functions, such as message formats, MAC commands and commissioning logic are done. The network server then uses a standard TCP/IP connection to exchange data with gateways. Gateways may be located anywhere in the world and several gateways can receive a single end device's messages. (16.)

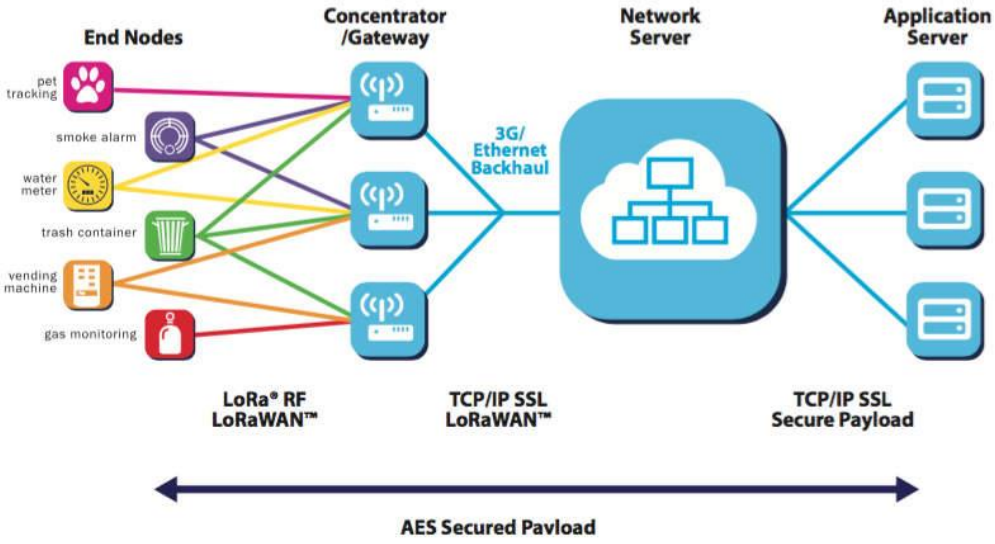

FIGURE 6. LoRaWAN network with a centralized network server. (16.)

3 REGIONAL LIMITATIONS

As LoRaWAN operates on licence free frequencies, it has different regulations depending on the region where it operates. This chapter outlines the limitations for an Europe 868-MHz industrial, scientific and medical (ISM) band. Understanding the limits of the planned operating region is crucial when attempting to determine if LoRaWAN is the right choice for this application. (7.)

3.1 Sub-bands

LoRaWAN uses the 868-870-MHz ISM band in Europe. This band has been divided into several sub-bands. Each sub-band has its own limitations set by European regulations. A radio device using these frequencies must obey these limitations with no exceptions. (7, 17.)

TABLE 2. 863 – 870MHz sub-bands and their limitations. (17.)

Band	Edge Frequencies		Field Power	Spectrum Access	Band Width
g (Note1,2)	863 MHz	870 MHz	+14 dBm	0.1% or LBT+AFA	7 MHz
g (Note2)	863 MHz	870 MHz	-4.5 dBm / 100 kHz	0.1% or LBT+AFA	7 MHz
g (Note2)	865 MHz	870 MHz	-0.8 dBm / 100 kHz	0.1% or LBT+AFA	5 MHz
	865 MHz	868 MHz	+6.2 dBm / 100 kHz	1% or LBT+AFA	3 MHz
g1	868.0 MHz	868.6 MHz	+14 dBm	1% or LBT+AFA	600 kHz
g2	868.7 MHz	869.2 MHz	+14 dBm	0.1% or LBT+AFA	500 kHz
g3	869.4 MHz	869.65 MHz	+27 dBm	10% or LBT+AFA	250 kHz
g4	869.7 MHz	870 MHz	+14 dBm	1% or LBT+AFA	300 kHz
g4	869.7 MHz	870 MHz	+7 dBm	No requirement	300 kHz
Note1: Modulation bandwidth ≤ 300 kHz is allowed. Preferred channel spacing is ≤ 100 kHz. Note2: Sub-bands for alarms are excluded (see ERC/REC 70-03 Annex 7).					

Table 2 shows that each sub-band has a frequency range, maximum transmission power, and duty cycle limitation set by regulations. The device channel plan needs to know about sub-bands and their limitations to be able to keep its radio functioning within the set regulations.

3.2 Duty cycle

A duty cycle is a mechanism for limiting a device access to radio frequencies. A duty cycle means the amount of time each device can be transmitting. This time

is known as Time on Air (ToA). A typical duty cycle limitation is 1% meaning that if the device transmits for one second, it must not transmit for the next 99 seconds. (18.)

To calculate how long the device must be silent for a given ToA, the formula 1 below can be used (19).

$T_{off} = (TimeOnAir / DutyCycle) - TimeOnAir$ *FORMULA 1*

Where

TimeOnAir = radio transmission time (s)
DutyCycle = sub-band duty cycle limitation (%)
T_{off} = time that transmitter must be silent on current sub-band (s)

A duty cycle is tied to a specific sub-band, and every sub-band in Europe is duty cycle limited. Sub-bands may have different duty cycle limitations. If a sub-band contains 3 LoRaWAN channels, their total ToA must be less than the allowed duty cycle. If a device uses all 3 channels equally and the sub-band's duty cycle limitation is 1%, each channel has a ~0.34% duty cycle limitation. If a device has two channels and they are on different sub-bands, the device has then a 2% total duty cycle limitation, if both sub-bands have a 1% limit. (18.)

All devices using these bands must obey these limits, and gateways are not an exception. If the gateway has to send lots of downlinks, it may then run out of its duty cycle reserve and be unable to send downlinks or acknowledge messages.

3.3 Transmit power

The maximum transmit power is also regulated. An LoRaWAN device operating on the 868-MHz unlicensed band is allowed to have the Equivalent Radiated Power (ERP) of +14dBm. All devices using this band must obey this rule, including the gateways for LoRaWAN. (7.)

The LoRaWAN specification has the maximum transmitter power defined as MaxEIRP where EIRP stands for the Effective Isotropic Radiated Power (EIRP). EIRP is defined as the total radiated power radiated by a hypothetical isotropic

antenna. MaxEIRP for LoRaWAN in Europe is +16dBm. The difference between the Effective Radiated Power (ERP) and EIRP is that EIRP compares the radiated power to a hypothetical isotropic antenna, and ERP uses a real antenna. Formula 2 below shows the relation between ERP and EIRP. (7, 20.)

$$ERP = EIRP - 2.15dB \qquad \textit{FORMULA 2}$$

Where

ERP = effective radiated power (dBm)
EIRP = effective Isotropic Radiated Power (dBm)

Because of the ERP limitations, devices with high gain antennas will need to reduce their transmission power to comply with these regulations. A high gain antenna will not allow the device to use any more transmission power but high gain will increase the receiver sensitivity. A higher gain antenna can help the device save power by using less power when transmitting.

3.4 Data rate and bandwidth

The available date rate and bandwidth are depending on radio regulations. LoRaWAN regional parameters define 8 different data rates to be used in Europe. As discussed earlier, the LoRa modulation has a variable spreading factor and bandwidth which form the actual data transfer rate. Regional parameters combine both and then refer to them as data rates. (7.)

TABLE 3. Data rate definition for LoRaWAN on the Europe 868-MHz band. (7.)

DataRate	Configuration	Indicative physical bit rate [bit/s]
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK: 50 kbps	50000
8..15	RFU	

Table 4: TX Data rate table

Table 3 has all available data rates for LoRaWAN in Europe. All data rates except 6 and 7 are using a 125-kHz bandwidth and only varying spreading factor to vary the data transfer rate. Data rates 6 and 7 are special. The data rate 6 requires a 250-kHz bandwidth which is not available on all sub-bands. The data rate 7 does not use the LoRa modulation at all. Instead, it uses the Frequency-shift keying (FSK) modulation for the high speed data transfer. As the LoRa modulation and FSK are two distinct modulation types, the receiver must be configured to correctly expect the incoming FSK transmission. (7.)

Data rates are denoted with a syntax DRx where x is a number from 0 to 7. The LoRaWAN specification uses this syntax when referencing to the actual data rate and to the used spreading factor and bandwidth. (7.)

3.5 Channel plan

When operating in Europe, LoRaWAN specifies 3 mandatory channels that every device must always have on its channel plan and that every gateway should be listening. These channels enable any device to establish a connection to a network without knowing its channel plan. Europe channel plan has a maximum of 16 channels. (7.)

TABLE 4. The Things Network channel plan.

Frequency	Bandwidth	SF max	SF min
868.1 MHz (JOIN)	125 kHz	12	7
868.3 MHz (JOIN)	125 kHz	12	7
868.3 MHz	250 kHz	7	7
868.5 MHz (JOIN)	125 kHz	12	7
868.8 MHz	FSK	FSK	FSK
867.1 MHz	125 kHz	12	7
867.3 MHz	125 kHz	12	7
867.5 MHz	125 kHz	12	7
867.7 MHz	125 kHz	12	7
867.9 MHz	125 kHz	12	7

Table 4 shows the channel plan for Europe the 868-MHz band used by the global LoRaWAN network The Things Network. The channel plan lists used frequencies, spreading factors, and bandwidths. All gateways on The Things Network operating in Europe should use the above channel plan. As seen in the channel plan, DR6 and DR7 are special data rates because they both only have one channel to use. (21.)

3.6 Maximum payload size

Regional limitations also govern the maximum payload size. The maximum payload size will depend on the data rate used. Higher data rates allow for bigger messages to be transmitted.

TABLE 5. LoRaWAN maximum payload sizes with the repeater support. Column M is the maximum MACPayload size. (7.)

DataRate	*M*	*N*
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	230	222
7	230	222
8:15	Not defined	

Table 7: EU863-870 maximum payload size

For LoRaWAN, it is possible to use a repeater and when doing so it adds extra information to the payload. This leads to a reduced maximum payload size. Table 5 shows the relation between the data rate and payload size. (7.)

Size (bytes)	7..22	0..1	0..*N*
MACPayload	FHDR	FPort	FRMPayload

FIGURE 7. MACPayload structure. (15.)

The MACPayload structure (seen in figure 7) contains three different fields. A frame header (FHDR) field contains information about the current frame and it

contains an “FOpts” field which is used to send MAC commands with the normal user payload. The FOpts field can be absent or it can use as much as 15 bytes of message size. An FRMPayload is the actual user specified payload. In the worst case scenario, the actual FRMPayload size is reduced by FHDR and FPort fields. The resulting FRMPayload size would be reduced by 23 bytes. (15.)

TABLE 6. LoRaWAN maximum payload size without the repeater support. Column M is the maximum MACPayload size. (7.)

DataRate	M	N
0	59	51
1	59	51
2	59	51
3	123	115
4	250	242
5	250	242
6	250	242
7	250	242
8:15	Not defined	

Table 8 : EU863-870 maximum payload size (not repeater compatible)

The repeater support is not mandatory. If a device does not plan to support repeaters, it can then increase its maximum payload size. The maximum message size without the repeater support is listed in table 6. (7.)

4 RANGE

LoRaWAN is advertised as a long range radio technology. The theoretical range can be calculated with known equations and models but they will never match real world testing. Real world testing will separate advertisements from reality and give realistic values on which the device designer can then reference when needed.

As the signals travel through air and objects, they will fade more before they reach the receiver. The LoRa modulation has a benefit of being demodulated below the noise floor with a negative SNR. The required SNR for demodulation depends on a spreading factor and the spreading factor depends on the used data rate. When testing for the range, the used data rate will have a great effect on the transmission range.

Because LoRaWAN operates on a license-free ISM band, the transmitter power output is limited by regulations. Devices operate on the maximum transmitting power by default. The device transmission power cannot be increased to increase the range. (7.)

4.1 Testing setup

To test LoRaWAN's range, the actual hardware was needed. MultiConnect mDot and MultiConnect Conduit by Multi-Tech Systems were used (33, 34). The hardware was provided by Etteplan Oy.

MDot is a small, Mbed OS enabled LoRaWAN module with an integrated radio and microcontroller. MDot uses an SX1272 LoRa radio module made by Semtech. For range testing, mDot was programmed with a testing software built with Mbed OS with an integrated LoRaWAN stack. The testing application is simple. It sets the data rate and sends a 10-byte-unconfirmed message and then waits for 10 seconds and repeats the process. See appendix 1 for testing program source code. The duty cycle was disabled for this test. MDot was powered from a battery pack. (34).

Conduit is a programmable gateway for IoT devices. A Conduit gateway can be extended using mCard accessory plug-ins. A Multiconnect LoRa mCard was used to turn Conduit into an LoRa gateway. Conduit can also be configured to work as a packet forwarder for a bigger LoRaWAN network, such as The Thing Network, or it can run a private LoRaWAN network server by itself. The later was used in this test. Running a private LoRaWAN network is quick and easy to setup, and results can be logged from the web interface. (33.)

Both Conduit and mDot were equipped with 3-dBi antennas.

4.2 Suburban range

For testing a range without line-of-sight, the gateway was positioned inside the Etteplan office building. The end device was moved around by walking in the surrounding suburban area with a smartphone logging position with timestamps. LoRaWAN messages that arrived at the gateway are automatically timestamped by the gateway. The timestamps from GPS logs and LoRaWAN messages were then matched and the best results per data rate were drawn on a map.

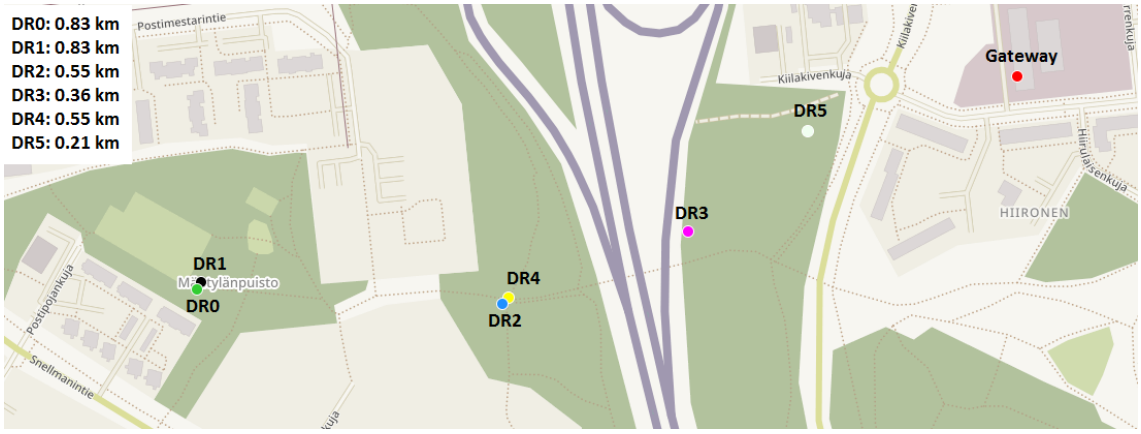

FIGURE 8. Suburban range.

Figure 8 shows the best results per data rate plotted on a map with the distance to the gateway. Results show that lower data rates get a better range on average. Data rate 4 and data rate 3 had interesting results because in theory data

rate 4 should have a shorter range than data rate 3. This is probably due to rapidly changing radio conditions as each transmission is 10 seconds apart and the device is moving between obstacles.

4.3 Line-of-sight range

The Line-of-sight range was carried out with same testing hardware as in the urban range test. The gateway was mounted on top of a stationary car. The battery powered mDot was mounted on top of another car which was driving on ice.

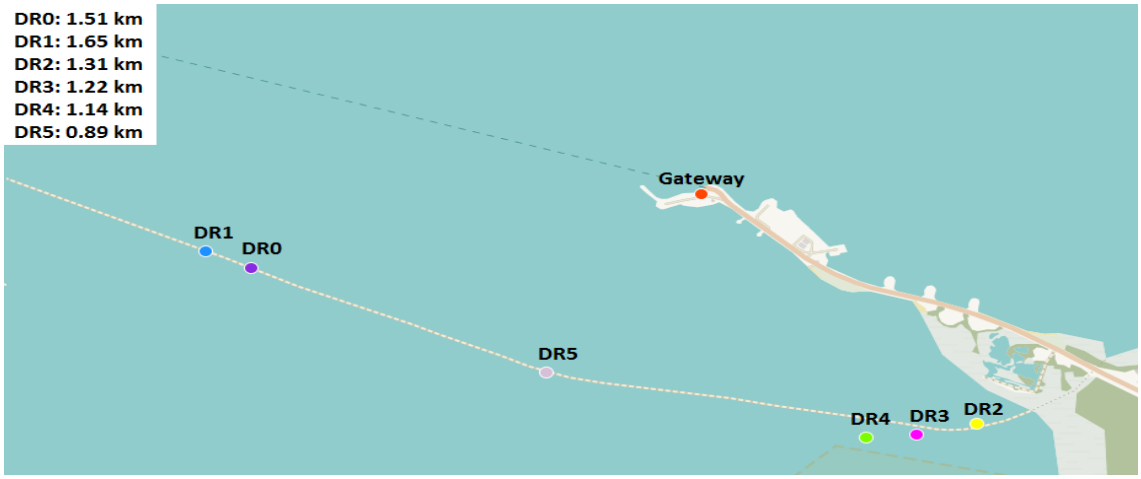

FIGURE 9. First line-of-sight test.

Figure 9 shows the gateway positioned at the end of a road. The car with mDot mounted on top of it was moving out to the sea. Getting the line-of-sight between the cars was challenging due to high snow banks. Antennas were barely over the bank most of the time which had a great effect on the achieved range. The best range was 1.51 km and it was received with data rate 1. A decision to move gateway to a different position for the second test was done.

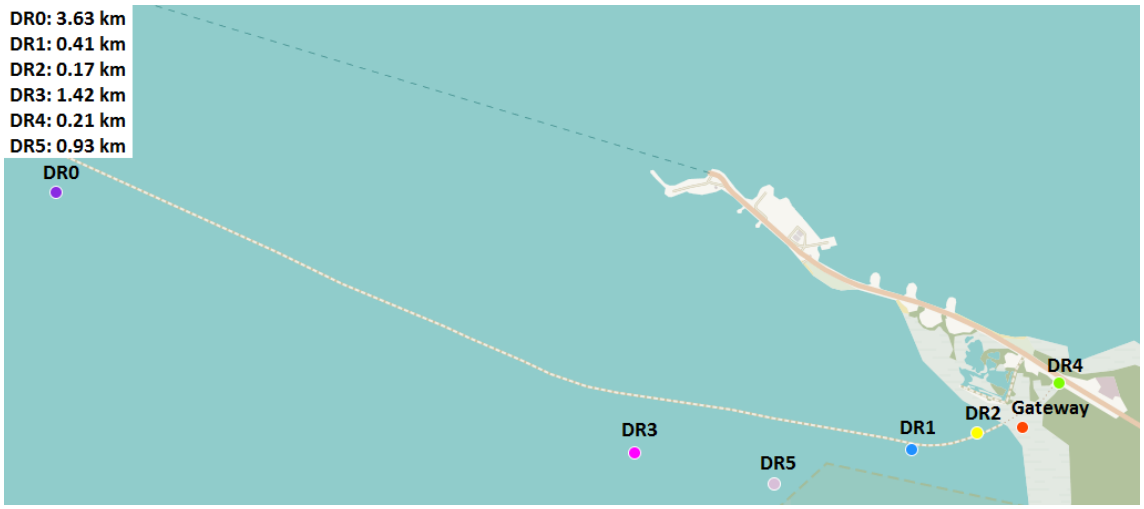

FIGURE 10. Line-of-sight test, gateway repositioned.

The gateway was positioned to a different place when the car was coming back to ashore (see figure 10). A single message with data rate 0 was received from 3.63 km away when there was a momentary line of sight before snow banks started to block it again. Again, rapidly changing radio conditions were challenging and data did not directly correlate with the expected smaller data rate which would yield a better range. More data is needed to average out errors from the rapidly changing link quality, but both test runs confirmed that data rates 0 and 1 receive the best range.

5 POWER CONSUMPTION

Low power consumption is one of the biggest selling points for the LoRaWAN technology. The battery life in excess of 10 years is being advertised. But what limitations does battery life measured in years set to a device designer? (22.)

A device designer must be able to predict how long a device can operate from different power sources and how to optimize the battery life. The LoRaWAN class A specification has been designed with a battery operated device as its main target. Implementing an ALOHA style messaging system, where the end device will only send data when it is needed and radio is operated as a receiver only shortly after each end device uplink, allows the device to sleep most of its time because it does not have to listen to any events coming from the radio downlink. (15.)

5.1 Test setup

To characterize the device power consumption, a device must be tested in real operating conditions.

Two different tests were planned. One was to determine the module power consumption without any radio usage. The other one was to test a module's power consumption while transmitting and receiving a message. The hardware used in these tests was the same that was used in range testing.

MDot has its own development board where it is inserted for programming and debugging. It is important to program a testing software using a development board and then remove it for power consumption testing. As the development board has extra electronics, measuring MDot's power consumption from the development board's power consumption would yield incorrect results. Because of this, MDot was always powered from an external power supply with a fixed voltage to keep results consistent.

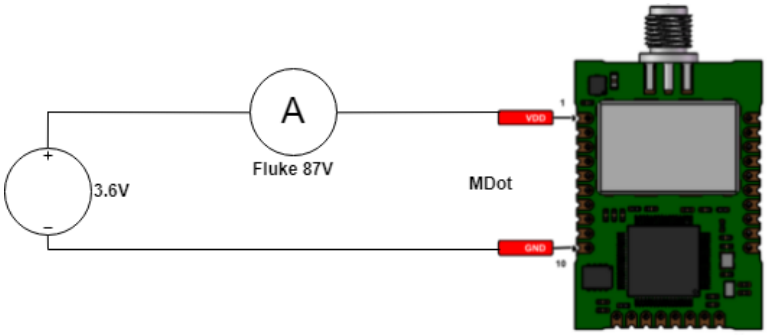

FIGURE 10. Static current measuring setup.

Measuring the power consumption of the device was done differently in each of the tests. When measuring small static currents with no radio usage, a Fluke 87V multimeter was used. This multimeter has a 0.01uA resolution with ±(0.2% + 2) accuracy. As all idle current tests draw constant power, results could be logged with just pen and paper from the multimeter screen. Figure 10 illustrates the measuring setup. (23.)

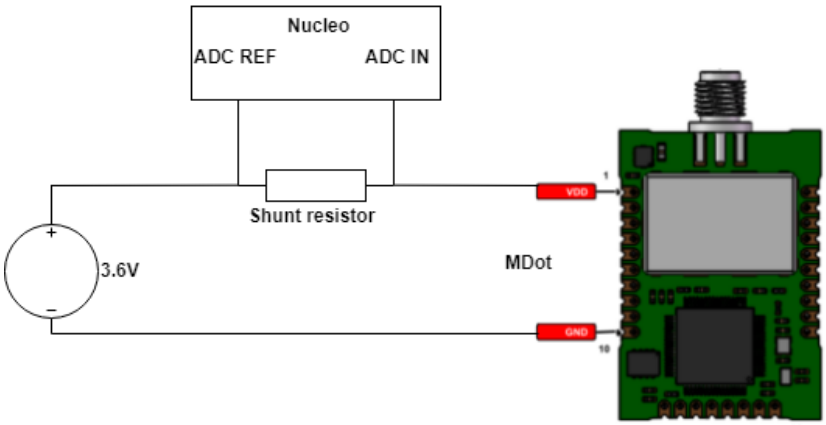

FIGURE 11. Radio power consumption measurement setup.

Testing the radio's power consumption was more complicated. Sending an LoRaWAN message is too fast for a multimeter to react and to log results. This problem was solved by using a STM NUCLEO-F411RE development board with 12-bit analog-to-digital converter (ADC) and a shunt resistor. The voltage drop over the shunt resistor was measured by connecting it to the ADC reference

voltage and to the ADC input (see figure 11). The ADC reading versus the actual current was then calibrated against the Fluke 87V multimeter and variable current power supply. The calibration function was generated by Microsoft Excel after manually collecting ADC readings and corresponding currents. The resulting function was then written into a sampling software which reads ADC at 200 Hz and then calculates the current and dumps it to a serial port for a computer to logging (see appendix 2). This setup was tested against the Fluke 87V multimeter and had sub mA accuracy. This setup allows rapidly changing currents to be accurately logged for a later analysis. (23, 24.)

5.2 Idle

Battery operated devices, which are meant to run for years, actually spend most of their time sleeping and doing nothing. Therefore, it is very important to know how much energy is being used when sleeping because this energy consumption mode is where the device spends most of its time. Modern microprocessors have several different options for power management and going through all of them is out of scope.

A simple testing software was written using Mbed OS 5.8 to test processor's operating modes for power consumption. The testing software has preprocessor flags to define what test to perform. The test software is compiled and flashed to mDot. MDot is then removed and connected to the measuring setup. This was repeated for all test cases. The program implementation can be found in appendix 3.

TABLE 7. Power consumption for different idle modes.

Operating mode	**Current (mA)**
While loop	15.05
Sleep	5.60
Deep sleep	0.42
Standby	0.034

Table 7 shows four different tests that were done. The different processor modes tested were; while loop, sleep, deep sleep and standby. The standby mode shuts down the processor and memory, and all memory contents are lost, with the exception of small a RTC backup memory area. (25.)

5.3 Radio

The radio power consumption testing was done with the shunt resistor setup described earlier. A testing software was created to send fixed size messages at fixed data rates (see appendix 1). The different test cases were: a confirmed message with a 10-byte payload on data rates 0-5, an unconfirmed message with 10-byte payload on data rates 0-5, and an unconfirmed message with 40-byte payload on data rates 0-5. The radio is operating at the default transmit power on all tests and the code rate is also default 4/5.

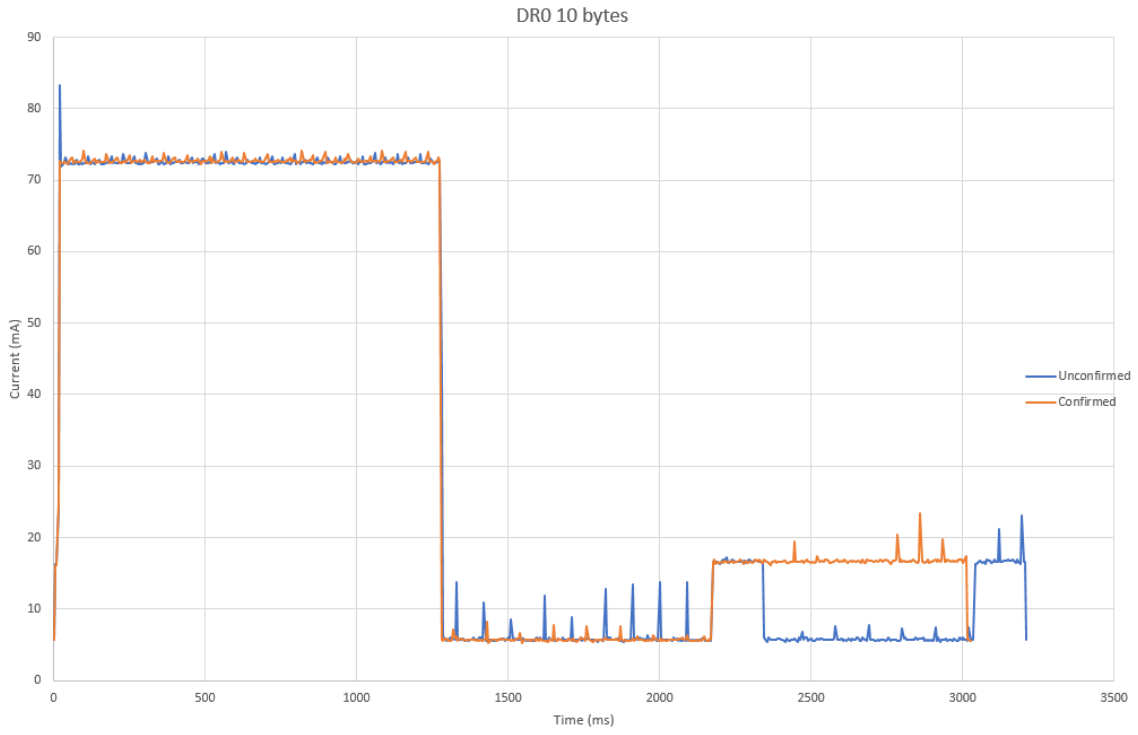

FIGURE 12. Raw data logged from serial.

All raw sampled data was logged and then processed using Microsoft Excel (see figure 12 as an example). The total energy consumption in Joules was calculated from each test scenario.

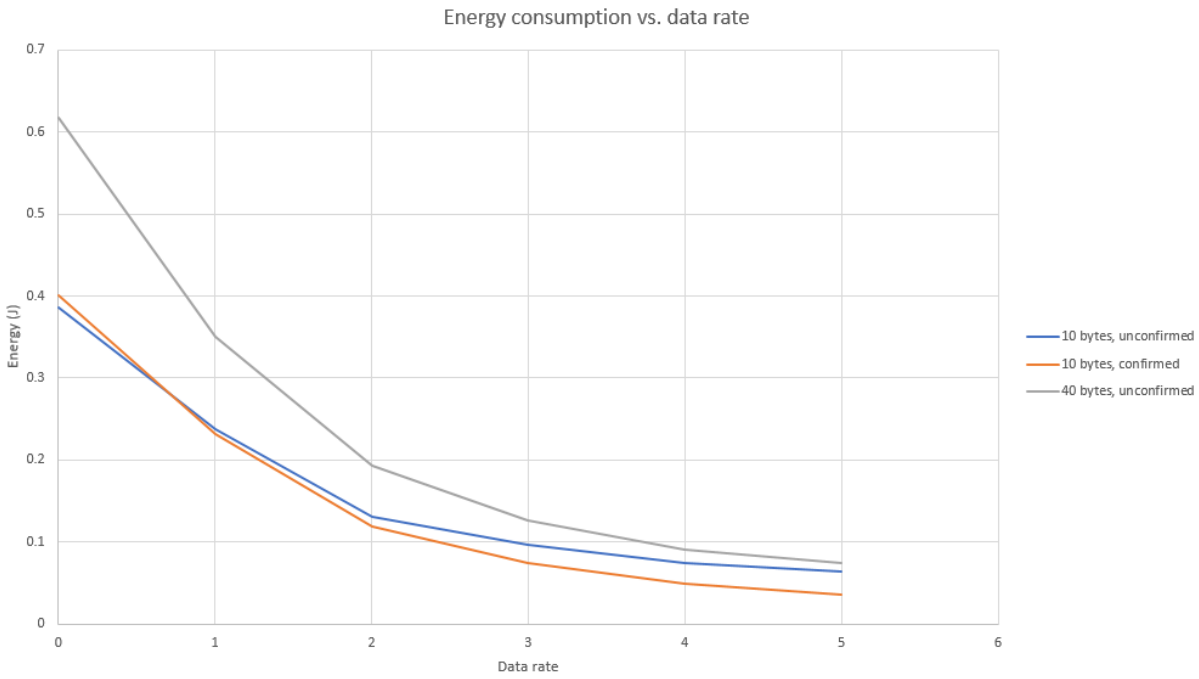

FIGURE 13. Energy consumption vs. data rate.

Figure 13 shows plotted data from all test cases. The plot shows clearly how energy required to transmit data decreases with the increasing data rate. This is because the increasing data rate shortens ToA of messages and thus the transmitter has to be active for a shorter time.

Confirmed messages actually consume slightly less energy on higher data rates because message confirmation happens on the RX1 window with the same settings as the uplink before it. The message confirmation RX1 window is shorter when the data rate increases and if confirmation is received during the RX1, the fixed data rate 0 RX2 window will not be opened. Unconfirmed messages need to open both receive windows even though no downlinks are coming and thus wasting energy.

The difference between energy used on 10- and 40-byte payloads get smaller as the data rate increases. The energy cost of transmitting 40 bytes at the data rate 5 is almost equal to sending 10 bytes at the same data rate.

5.4 Estimating power consumption

Because data rate and payload size are the two most important parameters that affect radio's power consumption, being able to estimate power the consumption for given parameters is important. As seen in the previous tests, when the radio is running at the default transmitting power and the code rate is default, its power consumption while transmitting is steady. Energy consumption is linear to the radio transmission time. By calculating the transmission time, the required energy for the given transmission can be calculated quite accurately.

the LoRa modulation has 3 parameters which affect the message ToA: Spreading factor, bandwidth, and code rate. The spreading factor and bandwidth together form a predetermined data rate specified by LoRaWAN regional parameters. The code rate has no default value but 4/5 is used by the Mbed OS LoRaWAN stack (26).

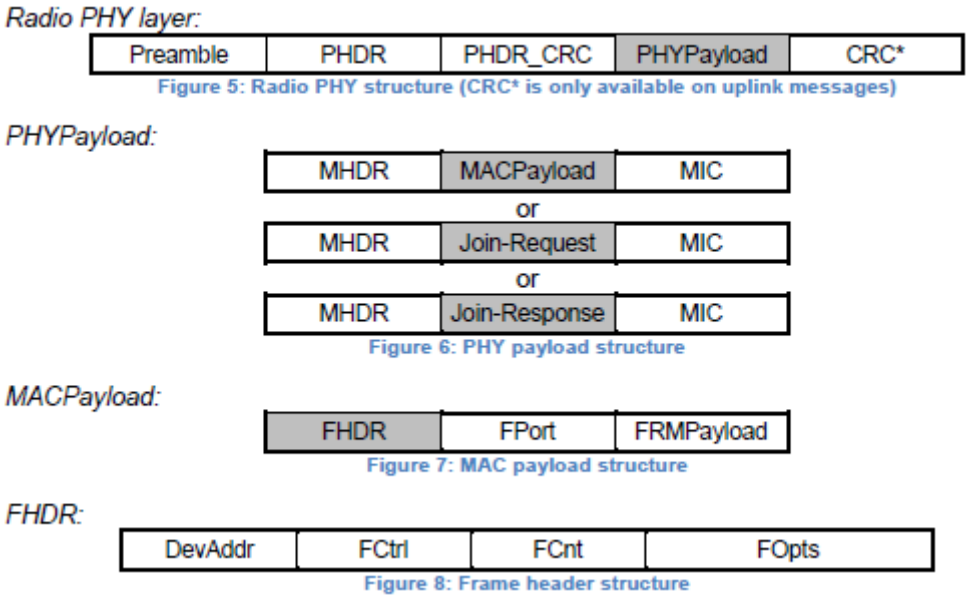

FIGURE 14. LoRaWAN message structure. (15.)

Each LoRaWAN message has a fixed format as seen in figure 14. Everything except PHYPayload is automatically added by the physical layer. PHYPayload is then broken down into several different fields. When performing an uplink message, the MACPayload field is present with its MAC header (MHDR) and

message integrity code (MIC). By determining the size of PHYPayload, the message time-on-air can be calculated and power consumption estimated. (15.)

MHDR has a fixed size of 1 byte. MIC has also a fixed size of 4 bytes. The frame header (FHDR) can vary its size if piggybacked MAC commands are present in the FOpts field. As MAC commands are rarely used, it is assumed that this frame does not contain any MAC commands. The FPort field is a fixed size of 1 byte and is always present if FRMPayload is present. (15.)

FHDR fields have the following sizes: DevAddr 4 bytes, FCtrl 1 byte, and FCnt 2 bytes. FOpts is assumed to be absent. Therefore, one can calculate the size of all MAC layer headers. By adding together all these fields, the total header length is determined to be 13 bytes. (15.)

It is now evident that the PHYPayload size is 13 bytes + the user supplied payload. To calculate the length of physical layer's fields, such as the physical layer header (PHDR), formula 3 can be used (5).

FORMULA 3

$$n_{payload} = 8 + max\left(ceil\left[\frac{(8PL - 4SF + 28 + 16CRC - 20IH)}{4(SF - 2DE)}\right](CR + 4), 0\right)$$

Where

$N_{payload}$ = total physical layer payload size (symbols)
PL = payload size (byte)
SF = spreading factor (7-12)
CRC = 1 if CRC is used, 0 if not
IH = 1 if implicit header mode, 0 if not
DE = data rate optimization, 1 if used, 0 if not
CR = code rate, 1 – 4 (4 / CR + 4)

To find out when the implicit header mode, CRC checking, and data rate optimization are used, one must check the SX1272 datasheet and the LoRaWAN specification. The SX1272 datasheet mandates that the data rate optimization is

used when using spreading factors 11 and 12 with 125-kHz bandwidth. The LoRaWAN specification mandates that all uplink messages use an explicit packet mode with CRC checking enabled. Table 8 lists all physical layer parameters required to calculate the payload length for each data rate. (5, 7.)

TABLE 8. Physical layer settings for each data rate.

Data rate	DE	IH	CRC	SF	BW
DR0	1	0	1	SF12	125 kHz
DR1	1	0	1	SF11	125 kHz
DR2	0	0	1	SF10	125 kHz
DR3	0	0	1	SF9	125 kHz
DR4	0	0	1	SF8	125 kHz
DR5	0	0	1	SF7	125 kHz
DR6	0	0	1	SF7	250 kHz
DR7	FSK	FSK	FSK	FSK	FSK

When the final physical layer payload size is known, the actual ToA can be calculated. Time on air includes a preamble which has a length of 8 symbols (7). Time on air for the payload can be calculated with formula 4 (5).

FORMULA 4

$$T_{payload} = n_{payload} \times T_s$$

Where

$T_{payload}$ = payload time on air (s)
$n_{payload}$ = length of payload (symbols)
T_s = symbol time (s)

The symbol time can be calculated by first calculating the symbol rate with formula 5 (5).

FORMULA 5

$$Rs = \frac{BW}{2^{SF}}$$

Where

Rs = symbol rate (symbols/s)
BW = bandwidth (Hz)
SF = spreading factor (7-12)

After having calculated the symbol rate, the actual symbol time can be calculated with formula 6 (5).

FORMULA 6

$$Ts = \frac{1}{Rs}$$

Where

Ts = symbol time (s)
Rs = symbol rate (symbols/s)

To complete time the on air calculation, the preamble must still be taken into account. Formula 7 can be used to calculate the time required to transmit the preamble (5).

FORMULA 7

$$T_{preamble} = (n_{preamble} + 4.25)T_{sym}$$

Where

$T_{preamble}$ = preamble time on air (s)

$n_{preamble}$ = preamble length (symbols)

T_{sym} = symbol time (s)

The total message time on air is then calculated with formula 8 (5).

FORMULA 8

$$T_{packet} = T_{preamble} + T_{payload}$$

Where

T_{packet} = total message time on air (s)

$T_{preamble}$ = preamble time on air (s)

$T_{payload}$ = payload time on air (s)

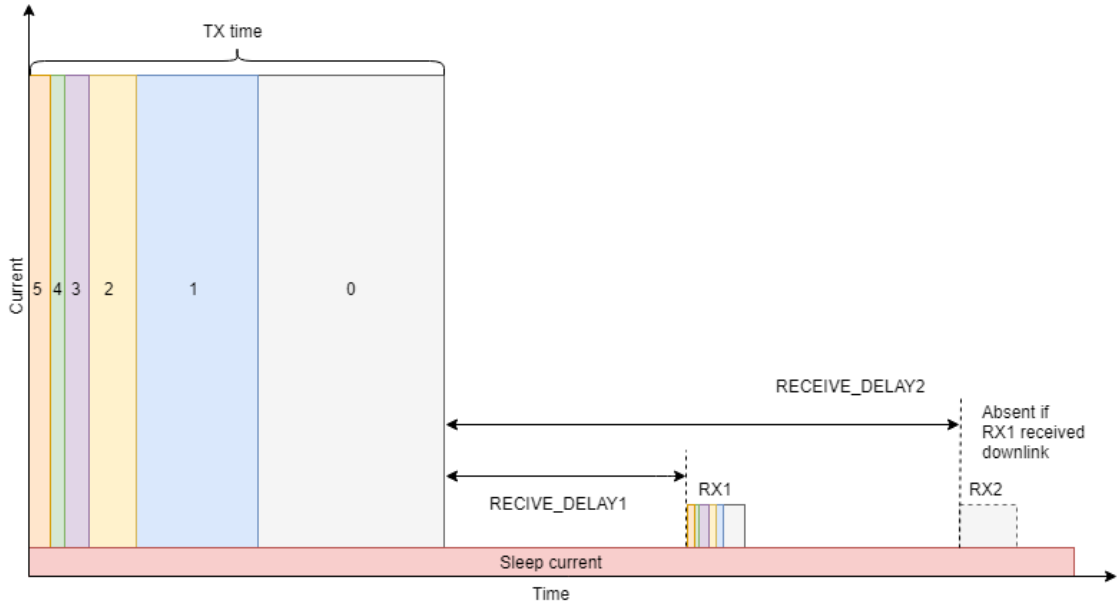

FIGURE 15. Power consumption breakdown.

To calculate the total power consumption, the receiver time also needs to be calculated. LoRaWAN will open two short receive windows after each uplink. Receive windows are named RX1 and RX2. The RX1 is the first window to be opened, and it uses the same radio settings as the uplink before it. The RX2 will

only be opened when RX1 does not receive a downlink. The RX2 has fixed radio settings. The receive windows are opened after fixed delays. Delays are defined as RECEIVE_DELAY1 for the RX1 and RECEIVE_DELAY2 for the RX2. The RX2 uses the data rate 0 by default. RECEIVE_DELAY1 is defined as 1 second, and RECEIVE_DELAY2 is defined as RECEIVE_DELAY1 + 1 second. (7, 15.)

The receiver needs 5 symbols to detect a preamble and synchronise its settings. The Time required to receive these 5 symbols from the preamble depends on the data rate used. Assuming that no downlink is received, the duration of the receive window can be calculated with formula 9. (27.)

$T_{rx} = 5 * T_{sym}$ *FORMULA 9*

Where

T_{rx} = duration of receive window (s)
T_{sym} = symbol time (s)

When both transmitter and receiver durations are calculated, the actual energy consumption can be calculated when the current consumption is known. One must also account energy consumed when the device is sleeping between receive windows. Figure 15 illustrates how the device power consumption is constructed. To calculate energy consumed by a single packet transmission, formula 10 is used. The formula 10 is only for unconfirmed messages. For estimating the confirmed message energy consumption, one must ignore the RX2 window and estimate the RX2 duration by calculating time on air for incoming message confirmation downlink.

$E_{unconfirmed} = (T_{tx} * I_{tx} + T_{rx1} * I_{rx} + T_{rx2} * I_{rx} + T_{sleep} * I_{sleep}) * U$ *FORMULA 10*

Where

$E_{unconfirmed}$ = energy used by unconfirmed message (J)
T_{tx} = transmission time (s)
I_{tx} = transmit current (A)
T_{rx1} = RX1 window duration (s)

I_{rx} = receive current (A)

T_{rx2} = RX2 window duration (s)

T_{sleep} = time spent in sleep between receive windows (s)

I_{sleep} = sleep current (A)

U = voltage (V)

When the total message energy requirement is calculated (or measured), the estimation can be done to determine how many messages would some particular battery provide. Since formula 10 gives the required energy in a standard unit (Joule) and battery capacities are usually measured in milliampere hours (mAh), a conversion is needed. Formula 11 will convert the battery nominal voltage and capacity in milliampere hours to energy.

$$E = U * Ah * 3.6 \qquad \textit{FORMULA 11}$$

Where

E = total energy stored in battery (J)

U = battery nominal voltage (V)

Ah = battery capacity (mAh)

TABLE 9. Measured energy consumption vs. calculated.

Data rate	Energy (measured) J	Energy (calculated) J	Error %	Messages (measured)	Messages (calculated)
0	0.385747549	0.413486182	6.94	117590	109701
1	0.236866731	0.238295347	0.60	191500	190352
2	0.130397358	0.118849434	9.27	347860	381659
3	0.096397827	0.075051725	24.90	470550	604383
4	0.075068944	0.050498662	39.13	604245	898242
5	0.063734225	0.036895027	53.34	711706	1229434

Table 9 illustrates 10-byte unconfirmed messages being sent. The measured energy usage is compared against the values calculated with formulas mentioned above. Errors are very small on lower data rates but increase radically when the data rate increases. A message count is also calculated from both measured and calculated values, comparing them to a 3500mAh 3.6V lithium-ion battery. The calculation assumes that all energy from this battery is used and that messages are sent immediately after each other. Even though the

message count seems large, it does not mean that the device has a long battery life. A quick calculation for data rate 0, for example, gives around 100 hours of battery life when assuming that each message takes around 3.5 seconds (transmission time + RECEIVE_DELAY2 + receive window 2 duration).

Semtech also provides a tool to calculate the LoRa modem power consumption. Previously used calculations can be double checked against this calculator. (28.)

6 POWER OUTAGE

No discussion about battery powered devices is complete without those batteries going dead. Every device will experience a power outage at a high certainty during its lifetime. It may be the actual electrical grid going down or simply running out of battery. This chapter explores issues related to these occurrences and how a device designer should work around them.

LoRaWAN has two different methods of connecting devices to a network; OTAA and ABP. The OTAA device performs a handshake with the server in order to join network. ABP devices have all encryption keys and device ids already in the device memory and the same settings have been manually applied to the network server as well. ABP devices do not do any kind of handshaking as they can just start sending and receiving messages after powering up. After connection, the OTAA device will behave exactly like the device with the ABP connection. The differences come apparent on power outage situations. (15.)

6.1 ABP power outage

Because ABP does not do any handshaking when it connects, there is no way to know when it is actually connected. When the ABP device loses its power, it will also lose its packet counters if they are not backed up to the non-volatile memory. (15.)

When the packet counters are reset back to 0 when the device powers back up, even with correct keys, the LoRaWAN server will ignore its messages until the packet counters are higher or equal the ones the server is expecting. If the device uses unconfirmed messages, the server will ignore all messages up to its current packet counter count. If the device uses confirmed messages, it will never get a confirmation as the server ignores the message. The only way to restore connections, if the device has lost its packet counters, is to reset them manually at the server. (15.)

6.2 OTAA power outage

The OTAA device will do actual handshake when it connects. Also, when connecting, both the server and end device will reset their packet counters to 0. When the device using OTAA loses power, it will have no problem of getting back to network. The OTAA connection also gives feedback to the device itself that the connection has been established. (15.)

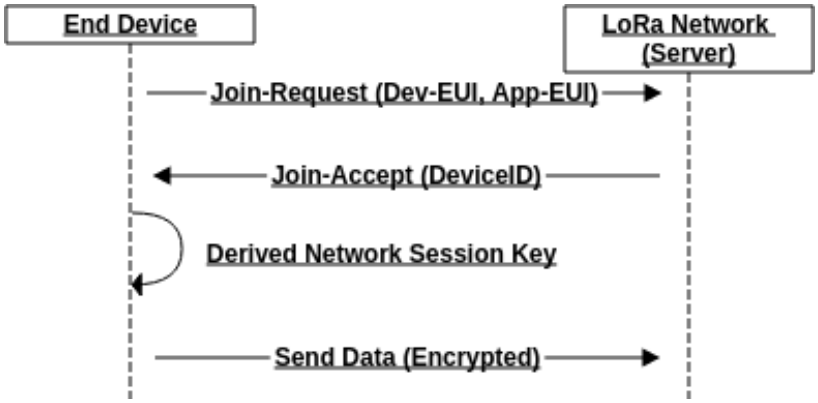

FIGURE 16. OTAA handshake. (29.)

Figure 16 illustrates the OTAA handshake flow. The end device initiates a join procedure by sending a Join-Request message to the network server with required encryption keys. If the keys are correct, the network server will respond with a Join-Accept message from which the actual encryption keys are derived. When this message arrives to the end device, it knows that the connection to the network server is working. The packet counters are now synchronized with the end device and the network server. (15.)

Because devices are likely to perform a join procedure right after powering up, there is a risk when a large amount of devices experience power outage at the same time. When the power is restored, all devices will begin joining at the same time. Devices using OTAA are probably using only 3 default join channels, thus possibly flooding them. The LoRaWAN specification implements a back-off strategy when such situation happens. A device designer can try to avoid these situations by joining after a randomized delay when powering up. (15.)

7 ADAPTIVE DATA RATE

Because the data rate has so great effect on the device radio power consumption and on air time, LoRaWAN features a built-in mechanism for the adaptive management of the end device data rate. The Adaptive data rate (ADR) aims to minimize device on air times and to reduce wasted energy. (15.)

ADR can optimise the device power consumption while ensuring that it is still being received by the gateway. When ADR is in use, the network server will indicate to the end device that it should reduce its transmission power or increase its data rate. In this way no energy and air time is wasted. (30.)

7.1 ADR implementation

ADR works by exploiting the fact that different data rates have different demodulation margins. If messages are received with too much of a margin, energy is wasted. (31.)

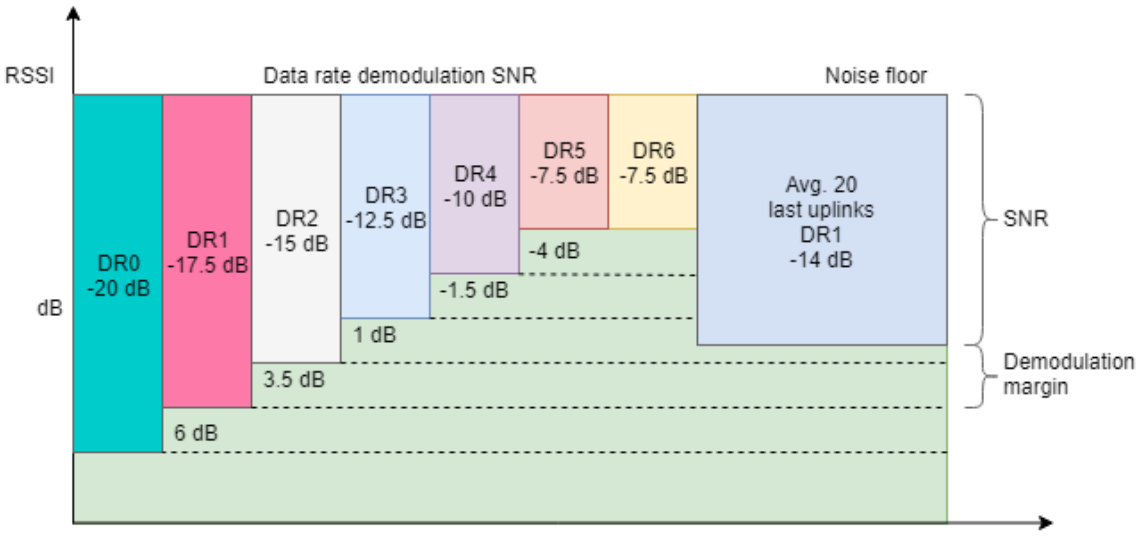

FIGURE 17. LoRa demodulation margins, and an example for ADR.

The LoRaWAN specification has a recommended implementation for an ADR algorithm. The device will decide if it wants to use ADR or if it wants to handle data rates by itself (although the device network server can also force device's

MAC layer to set or unset ADR bit). If the device wants the server to optimize its data rate, it will set the ADR bit on its packet headers. the LoRaWAN server collects 20 most recent transmissions for the device, which is requesting ADR. The server then averages all collected message SNR values and then determines how much margin there is. (30)

Figure 17 shows the LoRa demodulation SNRs and an example of the last 20 uplinks received by the server. The example uplinks are sent with DR1, and calculating the average from them, results in -14dB SNR. As seen in this figure, DR1 has a 3.5-dB demodulation margin when last uplinks were received with -14dB. DR2 would have a 1-dB demodulation margin. The network server would schedule a downlink with the MAC command for the end device to increase its data rate to DR2. (31.)

One important aspect of ADR is that the network server only sends commands to increase the end device's data rate, and the end device will only decrease its data rate by itself. When the network server calculates that the end device requesting ADR has enough demodulation to use a higher data rate, it will just send the MAC command. When the end device does not get confirmations or any downlinks from the network server while using ADR, it will reduce its data rate. The actual implementation is discussed in this chapter more in depth. (31.)

7.2 ADR challenges

ADR is not meant for every situation, and a designer preferring to use ADR must know about its limitations. Using ADR on wrong situations or use cases will lead to an excessive power consumption and a poor link quality. When ADR is used correctly, it will yield the minimum possible power consumption for the transmissions.

7.2.1 Moving devices

If the device is moving, and requesting a rate adaptation, the network server will most likely fail to adjust its data rate correctly. The radio channel of moving devices changes too fast and unpredictably to be correctly adjusted.

The LoRaWAN specification also implies that moving devices should not use ADR, but if the device can detect when it is stationary, it may then request a rate adaptation until it starts moving again. Then unsetting ADR bit will cause the network server to stop adapting its rate and to clean up previous packet measurements. (15.)

7.2.2 Unconfirmed messages

Because rate adaptation MAC commands are sent by the gateway, working up-links are required. If the device is only sending unconfirmed messages, and there is a sudden change in the link quality (e.g. something blocking line-of-sight, device relocated), the device will miss a lot of uplinks before adapting its own rate. If the device has not heard any downlink for ADR_ACK_LIMIT up-links, the device will have to set an ADRACKReq bit on its next uplinks until any downlink is received. If the network does not respond within ADR_ACK_DELAY uplinks, then the device can try to switch to a lower data rate. Every time ADR_ACK_DELAY uplinks are reached, the device can lower its data rate. ADK_ACK_LIMIT is defined as 64 and ADC_ACK_DELAY as 32 in LoRaWAN regional specifications. (7, 15, 31.)

A worst case scenario is that the device has been using ADR and is currently on DR5, and suddenly the link quality requires it to use DR0. This might result that the device is sending (ADR_ACK_DELAY * 5) + ADK_ACK_LIMIT uplinks before regaining the connectivity. (15.)

7.2.3 Confirmed messages

The ARM Mbed OS LoRaWAN stack implements the recommended backoff strategy for confirmed messages (32). When used with ADR, the network server will increase the data rate exactly like the device sending unconfirmed mes-sages, but reducing the data rate uses a different strategy than unconfirmed messages.

TABLE 10. Data rate back-off for confirmed messages. (15.)

Transmission nb	Data Rate
1 (first)	DR
2	DR
3	max(DR-1,0)
4	max(DR-1,0)
5	max(DR-2,0)
6	max(DR-2,0)
7	max(DR-3,0)
8	max(DR-3,0)

Table 10 shows the recommended data rate back-off strategy for confirmed messages. A transmission nb stands for a retransmission count for the current message. The retransmission count is configurable by the user. Using a higher retransmission setting will result in unpredictable power consumption and will possibly lead to a duty cycle starvation. If the retransmission count is not configured for more than one transmission, the end device will never reduce its data rate. The device will reduce its data rate when using confirmed messages, even when ADR is not enabled. (15.)

8 CONCLUSION

LoRaWAN technology has a narrow scope of use cases where it excels. Devices requiring long range communication with low data rates can take LoRaWAN into consideration. To achieve a long battery life, the device designer must be able to make compromises and be aware of design decisions that might decrease the battery life of device on certain conditions.

8.1 Power consumption

The end device will be spending most of its life doing nothing, sleeping and waiting for events or timers before doing anything. The most important aspect is to optimize the sleep or deep sleep power consumption, where the device is spending most of its time. Only after the sleep state power consumption is optimized, other optimizations take place. It should be understood that when sending an LoRaWAN message, most of the time is still spent waiting for receive windows. The actual radio power consumption is quite small as sending only lasts a few seconds at the best.

8.2 Connection method

OTAA should be used whenever possible. OTAA will give feedback that the device has actually connected and that it can send and receive messages. An OTAA connection will handle power outages easily, as it resets packet counters on a server when the connection has been established. When the device powers up, it has to do an OTAA handshake to be able to communicate with the network, and all power outage problems are solved. The OTAA handshake consists of only one uplink and one downlink, power consumption is similar to a single confirmed message.

8.3 Channel plan

One should use all channels that gateways on the network are listening. More channels on the channel plan also give a device more duty cycle, and decrease

the probability of packet collisions. One should only use channels that the gateways nearby are configured to listen, otherwise the device will experience a packet loss due to a random channel selection.

8.4 Confirmed and unconfirmed messages

For devices sending messages periodically, on should use mainly unconfirmed messages as they do not drain the duty cycle of a gateway and have more predictable power consumption. One should also periodically verify the connectivity with confirmed messages. When confirmed messages are being used, the designer must remember to select a retransmission count. Too many retransmissions will the drain device's battery and duty-cycle, too few will not the lower data rate when retrying.

For devices, which only send a message when something happens, for example a burglar alarm, one should use confirmed messages. Confirmed messages also allow for self-diagnostics, giving device designer more information about the connection status.

8.5 Data rate and ADR

If the device is static, one should always use ADR. ADR is meant for static devices, and a static device will reduce its radio power consumption to the minimum when using it. When using ADR with a static device, one should consider mixing confirmed messages with unconfirmed to verify the connectivity and quickly drop the data rate lower if no confirmation is received.

When the device is moving, a fixed data rate must be used. The device designer should use a data rate as low as possible, as it should have the best range. A low data rate will limit the device's maximum payload size and its data transfer rate even further with duty-cycle limitations. Payloads should be kept at the minimum, and the maximum payload size should be kept in mind when designing. Messages can be truncated if not.

The payload size should always be minimized so that it will not be fragmented. Compression methods should be used to pack as much data on a single frame

as possible. All frames have at least 13 bytes of overhead from the MAC layer and then an additional overhead from a physical layer, thus minimizing the sending of small messages and packing them up as one larger message will be beneficial for the power consumption and duty cycle.

REFERENCES

1. Ray Brian 2015. What is LoRa? Date or retrieval 9.4.2018 https://www.link-labs.com/blog/what-is-lora

2. Semtech 2015. AN1200.22 LoRa™ Modulation Basics. Date of retrieval 5.4.2018. https://www.semtech.com/uploads/documents/an1200.22.pdf

3. Knight Matt. Reversing LoRa. Date of retrieval 15.4.2018. https://static1.squarespace.com/static/54cecce7e4b054df1848b5f9/t/57489e6e07eaa0105215dc6c/1464376943218/Reversing-Lora-Knight.pdf

4. RevSpace. DecodingLora. Date of retrieval 10.4.2018. https://revspace.nl/DecodingLora

5. Semtech 2017. SX1272/73 Datasheet. Date of retrieval 10.4.2018 https://www.semtech.com/uploads/documents/sx1272.pdf

6. Semtech 2016. SX1276/77/78/79 Datasheet. Date of retrieval 10.4.2018 https://www.semtech.com/uploads/documents/DS_SX1276-7-8-9_W_APP_V5.pdf

7. LoRa Alliance 2017. LoRaWAN™ 1.0.2 Regional Parameters. Date of retrieval 11.4.2018. https://www.thethingsnetwork.org/forum/uploads/default/original/2X/d/dca7977549092f2c594a601caa739e1bacbeea59.pdf

8. Ghoslya Sakshama2017. LoRa: Symbol Generation. Date of retrieval 12.4.2018. http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html

9. Unknown. Code rate. Date of retrieval 12.4.2018. https://en.wikipedia.org/wiki/Code_rate

10. Sebastien 2017. coding rate in LoRaWAN. Date of retrieval 12.4.2018. http://semtech.force.com/lora/lc_answers_questions?id=90644000000PmBRAA0

11. IBM. ibm-lmic. Date of retrieval 12.4.2018. https://github.com/mcci-catena/ibm-lmic

12. Semtech. LoRaWAN endpoint stack implementation and example projects. Date of retrieval 12.4.2018. https://github.com/Lora-net/LoRaMac-node

13. ARM 2017. Mbed OS. Date of retrieval 13.4.2018. https://www.mbed.com/en/

14. ARM. Example LoRaWAN application for Mbed-OS. Date of retrieval 13.4.2018. https://github.com/ARMmbed/mbed-os-example-lorawan

15. Lora Alliance 2016. LoRaWAN Specification v1.0.2. Date of retrieval 13.4.2018. Available from Lora Alliance.

16. Panigrahi Prashant 2018. LoRa Architecture. Date of retrieval 16.4.2018. http://www.3glteinfo.com/lora/lora-architecture/

17. Quere Julien 2018. LoRaWan, a dedicated IoT network. Date of retrieval 16.4.2018. https://witekio.com/de/blog/lorawan-dedicated-iot-network/

18. TheThingsNetwork. Duty Cycle for LoRaWAN Devices. Date of retrieval 2.4.2018. https://www.thethingsnetwork.org/docs/lorawan/duty-cycle.html

19. TheThingsNetwork 2017. Understanding duty cycle limitations. Date of retrieval 2.4.2018. https://www.thethingsnetwork.org/forum/t/understanding-duty-cycle-limitations/5029

20. Unknown. Effective radiated power. Date of retrieval 5.4.2018. https://en.wikipedia.org/wiki/Effective_radiated_power

21. TheThingsNetwork. EU-global_conf.json. Date of retrieval 17.4.2018. https://github.com/TheThingsNetwork/gateway-conf/blob/master/EU-global_conf.json

22. Digi-Key 2016. LoRaWAN Part 1: How to Get 15 km Wireless and 10-Year Battery Life for IoT. Date of retrieval 22.4.2018. https://www.digikey.com/en/articles/techzone/2016/nov/lorawan-part-1-15-km-wireless-10-year-battery-life-iot

23. Fluke. Fluke 87V Industrial Multimeter. Date of retrieval 17.4.2018. http://en-us.fluke.com/products/digital-multimeters/fluke-87v-digital-multimeter.html

24. STMicroelectronics. STM32 Nucleo-64 development board with STM32F411RE MCU, supports Arduino and ST morpho connectivity. Date of retrieval 18.4.2018. http://www.st.com/en/evaluation-tools/nucleo-f411re.html

25. STMicroelectronics 2017. STM32F411xC STM32F411xE Datasheet. Date of retrieval 18.4.2018. http://www.st.com/resource/en/datasheet/stm32f411re.pdf

26. ARM. LoRaPHY.cpp. Date of retrieval 6.4.2018. https://github.com/ARMmbed/mbed-os/blob/master/features/lorawan/lorastack/phy/LoRaPHY.cpp

27. Semtech 2015. AN1200.23 SX1272 Settings for LoRaWAN. Date of retrieval 21.4.2018. https://www.semtech.com/uploads/documents/an1200.23.pdf

28. Semtech. LoRa Calculator: fast evaluation of link budget, time on air and energy consumption. Date of retrieval 21.4.2018. https://www.semtech.com/uploads/documents/SX1272LoRaCalculatorSetup1_1.zip

29. Nazmul, Atlants Embedded. Diving Deeper in LoRaWan – Episode 2. Date of retrieval 23.4.2018. http://atlantsembedded.com/2017/10/16/diving-deeper-in-lorawan-episode-2/

30. TheThingsNetwork. LoRaWAN Adaptive Data Rate. Date of retrieval 23.4.2018. https://www.thethingsnetwork.org/docs/lorawan/adr.html

31. Ghoslya Sakshama 2017. How does LoRaWAN Adaptive Data Rate work? http://www.sghoslya.com/p/how-does-lorawan-nodes-changes-their.html

32. ARM. LoRaMac.cpp. Date of retrieval 23.4.2018. https://github.com/ARMmbed/mbed-os/blob/master/features/lorawan/lorastack/mac/LoRaMac.cpp

33. MultiTech. MultiConnect® ConduitTM. Date of retrieval 24.4.2018. https://www.multitech.com/brands/multiconnect-conduit

34. MultiTech. MultiConnect® mDotTM. Date of retrieval 24.4.2018. https://www.multitech.com/brands/multiconnect-mdot

```
//Based on https://github.com/ARMmbed/mbed-os-example-lorawan
//Test settings
#define USE_ADR             false
#define PAYLOAD_LENGTH      10
#define ROTATE_DATARATE     false
#define DATARATE            DR_0
#define SEND_INTERVAL       20000
#define MESSAGE_TYPE        MSG_CONFIRMED_FLAG

#include <stdio.h>

#include "mbed.h"
#include "lorawan/LoRaWANInterface.h"
#include "lorawan/system/lorawan_data_structures.h"
#include "events/EventQueue.h"

#include "mbed_trace.h"

#define TRACE_GROUP "APP"

#include "trace_helper.h"
#include "lora_radio_helper.h"

using namespace events;

uint8_t tx_buffer[LORAMAC_PHY_MAXPAYLOAD] = { 0 };


#define MAX_NUMBER_OF_EVENTS            16
#define CONFIRMED_MSG_RETRY_COUNTER     3


static EventQueue ev_queue(MAX_NUMBER_OF_EVENTS * EVENTS_EVENT_SIZE);
static void lora_event_handler(lorawan_event_t event);
static LoRaWANInterface lorawan(radio);
static lorawan_app_callbacks_t callbacks;


int main (void)
{
    // setup tracing
    setup_trace();

    // stores the status of a call to LoRaWAN protocol
    lorawan_status_t retcode;
```

```
    // Initialize LoRaWAN stack
    if (lorawan.initialize(&ev_queue) != LORAWAN_STATUS_OK) {
        printf("\r\n LoRa initialization failed! \r\n");
        return -1;
    }

    printf("\r\n Mbed LoRaWANStack initialized \r\n");

    // prepare application callbacks
    callbacks.events = mbed::callback(lora_event_handler);
    lorawan.add_app_callbacks(&callbacks);

    retcode = lorawan.connect();

    if (retcode == LORAWAN_STATUS_OK ||
        retcode == LORAWAN_STATUS_CONNECT_IN_PROGRESS) {
    } else {
        printf("\r\n Connection error, code = %d \r\n", retcode);
        return -1;
    }

    printf("\r\n Connection - In Progress ...\r\n");

    // make your event queue dispatching events forever
    ev_queue.dispatch_forever();

    return 0;
}


/**
 * Sends a message to the Network Server
 */
static void send_message()
{
    int16_t retcode;
    static uint8_t rate = 0;

     // Enable adaptive data rate
    #if USE_ADR
    if (lorawan.enable_adaptive_datarate() != LORAWAN_STATUS_OK) {
        printf("\r\n enable_adaptive_datarate failed! \r\n");
        return;
    }
    printf("\r\n Adaptive data  rate (ADR) - Enabled \r\n");
    #elif !USE_ADR
```

```
    if (lorawan.disable_adaptive_datarate() != LORAWAN_STATUS_OK) {
        printf("disable_adaptive_datarate failed! \r\n");
        return;
    }
    printf("ADR DISABLED\r\n");

    #if ROTATE_DATARATE

    if (rate > 5) rate = 0;

    if (lorawan.set_datarate(rate) != LORAWAN_STATUS_OK) {
        printf("set_datarate failed! \r\n");
        return;
    }

    rate++;

    #else

    if (lorawan.set_datarate(DATARATE) != LORAWAN_STATUS_OK) {
        printf("set_datarate failed! \r\n");
        return;
    }

    #endif
    printf("SET_DATARATE OK\r\n");
    #endif

    retcode = lorawan.send(MBED_CONF_LORA_APP_PORT, tx_buffer, PAYLOAD_LENGTH,
                           MESSAGE_TYPE);

    if (retcode < 0) {
        retcode == LORAWAN_STATUS_WOULD_BLOCK ? printf("send - WOULD BLOCK\r\n")
                : printf("\r\n send() - Error code %d \r\n", retcode);
        return;
    }

    printf("\r\n %d bytes scheduled for transmission \r\n", retcode);
}

static void lora_event_handler(lorawan_event_t event)
{
    switch (event) {
        case CONNECTED:
            printf("\r\n Connection - Successful \r\n");
```

```
            ev_queue.call_every(SEND_INTERVAL, send_message);
            break;
        case DISCONNECTED:
            ev_queue.break_dispatch();
            printf("\r\n Disconnected Successfully \r\n");
            break;
        case TX_DONE:
            printf("\r\n Message Sent to Network Server \r\n");
            break;
        case TX_TIMEOUT:
        case TX_ERROR:
        case TX_CRYPTO_ERROR:
        case TX_SCHEDULING_ERROR:
            printf("\r\n Transmission Error - EventCode = %d \r\n", event);
            break;
        case RX_DONE:
            printf("\r\n Received message from Network Server \r\n");
            break;
        case RX_TIMEOUT:
        case RX_ERROR:
            printf("\r\n Error in reception - Code = %d \r\n", event);
            break;
        case JOIN_FAILURE:
            printf("\r\n OTAA Failed - Check Keys \r\n");
            break;
        default:
            MBED_ASSERT("Unknown Event");
    }
}
```

```
#include "mbed.h"

AnalogIn input(A0);
Serial pc(USBTX, USBRX);

int main()
{
    pc.baud(115200);

    while(1) {
        float ma = (-233.21f * input.read()) + 233.11f;
        pc.printf("%f\r\n", ma);
        wait_ms(5);
    }
    return 0;
}
```

```
#define TEST_LOOP

#include "mbed.h"

#include "SX1272_LoRaRadio.h"
#include "lorawan_data_structures.h"

void sleep_radio() {
    SX1272_LoRaRadio radio(LORA_MOSI, LORA_MISO, LORA_SCK, LORA_NSS, LORA_RESET,
        LORA_DIO0, LORA_DIO1, LORA_DIO2, LORA_DIO3, LORA_DIO4,
        LORA_DIO5, NC, NC, LORA_TXCTL, LORA_RXCTL, NC, NC);

    radio_events_t test;

    radio.init_radio(&test);
    radio.sleep();
}

void sleep_flash() {
    SPI flash(SPI3_MOSI, SPI3_MISO, SPI3_SCK);
    flash.format(8, 0);
    DigitalOut flash_cs(SPI3_CS);

    flash_cs = 0;
    flash.write(0xB9);
    flash_cs = 1;
}

int main() {
    //Put external peripherals to sleep
    sleep_radio();
    sleep_flash();

    #ifdef TEST_LOOP
    while(1) {

    }
    #endif

    #ifdef TEST_SLEEP
    while(1) {
        wait(100);
    }
    #endif

    #ifdef TEST_DEEPSLEEP
```

```
    while(1) {
        hal_deepsleep();
    }
    #endif

    #ifdef TEST_STANDBY
    while(1) {
        HAL_PWR_EnterSTANDBYMode();
    }
    #endif

    return 0;
}
```