The suitability of LoRaWAN for battery powered security solutions

IRINA GАLIULINA

PATRIK KARLSTÉN
Abstract

Many conventional forms of communication technology, such as Wi-Fi, 3G/4G or cable, require a lot of power. For battery powered devices that need to last a long time on a single charge, one alternative is the low-power, long range technology LoRaWAN.

This thesis tries to answer the question - how well do the properties of LoRaWAN meet the requirements for a battery powered security solution? Two identical prototype remote motion detectors were implemented for this purpose. The results show that while the prototypes do not meet the requirements for energy efficiency, LoRaWAN as a technology easily does. The results shows that if a solution to the reliability issues can be found, LoRaWAN would be well suited for battery powered security solutions.

Keywords:
LoRaWAN, Internet of things (IoT), security, battery power, motion sensor
Abstract

Många vanliga teknologier som används för kommunikation, så som Wi-Fi, 3G/4G eller fiber, kan vara väldigt strömkrävande. Ett alternativ för batteridrivna enheter som behöver kunna klara sig på en laddning under lång tid, är att använda en lågenergiteknologi med lång räckvidd - LoRaWAN.

Den här rapporten försöker att besvara frågan om hur väl LoRaWANs egenskaper tillgodoser de krav som ställs på batteridrivna säkerhetslösningar. För detta ändamål utvecklades två identiska prototyper av en batteridriven rörelsesensor. Resultaten visar på att även om prototyperna inte möter energikonsumtionsskraven, så gör själva LoRaWAN-tekniken detta. Resultaten visar att om man kan hitta lösningar på problemen med pålitligheten hos LoRaWAN, så kan LoRaWAN mycket väl vara lämpligt för batteridrivna säkerhetslösningar.

Nyckelord:
LoRaWAN, sakernas internet (IoT), säkerhet, batteridrift, rörelsesensor
# Contents

1 Introduction  
1.1 Background 3  
1.2 Problem 3  
1.3 Purpose 4  
1.4 Goal 4  
1.4.1 Benefits, Ethics and Sustainability 4  
1.5 Methodology 5  
1.6 Deliminations 6  
1.7 Stakeholders 6  
1.8 Outline of the thesis 6  

2 LoRaWAN 7  
2.1 Low Power Wide Area Networks 7  
2.2 The Things Network 9  
2.3 Device classes 10  
2.4 Data rates 12  
2.5 Activation of an end-device 13  
2.6 Technical overview and security of messages 13  
2.7 Known vulnerabilities 15  

3 Security systems, motion detection sensors and related work 17  
3.1 Grading of security and alarm systems 17  
3.2 Passive Infrared Motion Detection Sensors 18  
3.3 Microcontroller boards and Arduino 19  
3.4 Related work 20  

4 Project methods and methodology 22  
4.1 Research approach and strategy 22  
4.2 Data Collection 23  
4.3 Data Analysis 23  
4.4 Quality Assurance 23  

5 Requirements specification, practical implementation and experimental setup 25  
5.1 The requirements specification 25  
5.2 End-device prototypes 26  
5.3 The application 28  
5.4 The experimental setup for measuring power consumption 29  
5.5 Measuring and handling of readings 30
6 Experimental Results
   6.1 Current measurements ........................................ 33
   6.2 Voltage measurements ........................................ 35
   6.3 Power consumption adjusted by voltage ..................... 36
   6.4 Overall behavior of the current ............................ 37
   6.5 Applying the requirements specification .................... 38

7 Discussion
   7.1 Reliability .................................................... 41
   7.2 Power consumption ........................................... 42

8 Conclusions and future work
   8.1 Conclusions ................................................... 44
   8.2 Future work .................................................. 45

A Source code
   A.1 Main source-code for the end-device ....................... 46
   A.2 Source-code for testing 1 Byte payloads ................. 49
   A.3 Application source code .................................... 50
      A.3.1 Decoder .................................................. 50
      A.3.2 Converter ............................................... 50
      A.3.3 Validator ............................................... 50
   A.4 Source code for reading the measurement from Fluke 45 multimeter 51
1. **Introduction**

For many years, the majority of things connected through Internet have been stationary computers and devices. This is rapidly changing, with more and more different devices being connected. Many devices that are being connected today do not have a permanent power supply and are being powered by batteries.

To create small, long lived, battery powered devices that can connect through the Internet, conventional communication technologies such as cellular network or Wi-Fi face some issues when it comes to power requirements \[1\] and cost for network access \[2\][3]. In order to be able to connect more and more things through the Internet, new communication technologies are being developed. One such type of technology is low power wide area networks \[4\].

1.1 **Background**

There are several different types of low power wide area network technologies. What they all have in common is trying to increase the amount of data that can be sent over a long range, while using as little power as possible.

One such technology is LoRaWAN \[5\]. LoRaWAN is a low power, long range wireless technology made for sending small amounts of data but not too frequently \[6\]. LoRaWAN is a network protocol and architecture in one. LoRaWAN operates on top of a physical layer protocol \[5\] called LoRa \[7\][8]. LoRa enables wireless, long range communication over radio \[6\].

Various research has been done on LoRaWANs qualities in general, such as power consumption and range \[1\][9][10]. There has also been research done on LoRaWANs suitability for cases such as smart cities \[11\] or monitoring peoples health \[12\], but the area of security monitoring and alarm systems is much less explored.

1.2 **Problem**

In this modern era more and more people have the opportunity to travel or own assets in places that are not their main home. However not everyone has the means to or is willing to install a full scale security solution for their remote cabin, boat or storage unit. A full scale security solution may cost a lot to install and maintain, and usually requires a permanent supply of power, which may or may not be available. This problem can be solved with modern day technology in many ways. With LoRaWAN being a fairly new technology, there seems to be not much research done on whether or not LoRaWAN could be a suitable solution for this problem. In order to solve this with the use of LoRaWAN, several factors need
to be considered. These are energy efficiency, transmission time, reliability and vulnerabilities.

High energy efficiency is vital for a battery powered solution to be viable. When using LoRaWAN there will always be trade offs between energy efficiency, data rate and time spent transmitting [6][10][13]. Less time spent transmitting leads to a higher energy efficiency.

Transmission time is a highly relevant factor when designing and implementing security solutions. The user needs to be notified as soon as possible in case of a breach. It is also important that the message is sent fast enough for a physical interruption to be impossible or at least improbable.

The reliability of the solution depends on the solutions uptime and the solutions ability to correctly identify a breach. If the solution frequently fails to send messages or be available, then it can not be considered reliable. If the solution fails to identify breaches or falsely reports a breach when there is none, then it can not be considered reliable either.

The last factor is the vulnerabilities of the solution. How could someone disrupt or circumvent the solution? To be able to design a secure solution, possible vulnerabilities need to be taken into consideration.

With these factors in mind, the question this thesis tries to answer is as follows: How well do the properties of LoRaWAN meet the requirements for a battery powered security solution?

1.3 Purpose

This thesis presents how LoRaWAN may be used for security and alarm solutions and analyzes the different constraints and requirements. While LoRaWAN itself is considered to be very energy efficient and secure [1], the question is if LoRaWAN is suitable for a remote battery-powered security application.

1.4 Goal

The goal for this project is to determine the suitability of LoRaWAN when used in remote battery-powered security solutions.

1.4.1 Benefits, Ethics and Sustainability

This project and thesis may prove beneficial for several parties. It may be useful for people, companies and researchers who are looking into motion security systems and want to see how this can be implemented with wireless technology, as well as those looking for new wireless solutions where the main power source is a battery. While this project focuses mainly on motion sensors, many other applications require a certain level of security and reliability, which are both discussed in this thesis. The latter two properties can also be useful for further improving LoRaWAN in general.

When it comes to ethical issues, two main ones have been identified. The first one is misuse of the proposed system. While the main purpose of the prototypes developed for this project is to detect possible intruders, they can also be used against someone with the purpose of stalking. However the system does not show who has triggered the motion detector and therefore the stalker still needs to be
present to be sure of the trigger. It can therefore be concluded that the proposed system in itself does not enable stalking and is believed to bring little value while misused.

The second ethical issue is also an issue of sustainability - the production of the hardware and the product. From a social point of view, it is important that the production does not involve child labor, unfair wages and dangerous work environment. When it comes to social sustainability of the hardware used in this project, the level of the social sustainability is deemed reasonably high. The main microcontroller board used is an open-license one, which provides the freedom to manufacture the board personally, if there are doubts about the product being made in a sustainable and ethical way [14]. The motion sensor company [15] claims on their sustainability page on the website [15] to work for social and environmental sustainability, and this is assumed to be true by the authors of this thesis. The battery and other parts, like cables, can easily be chosen in such a way that they are made in an ethical way. Making sure that the production line is socially sustainable leads to economic sustainability in the long run. As more people earn more money, more children are able to go to school instead of working low paying jobs, these people also become able to afford sustainable products, instead of just cheaper versions.

Besides social and economical sustainability, it is also important to consider environmental sustainability. Most of the modern day components for the prototypes in this project can be recycled and the prototypes are expected to be operational for many years, as parts that need changing can easily be replaced individually. With waste awareness spreading more and more throughout the world, more and more research is done on recyclable and sustainable materials, meaning that in the future, the solution may be improved and implemented with materials that leave a smaller footprint on our planet. The proposed solution is therefore believed to be fairly environmentally sustainable.

1.5 Methodology

There are two main methodologies that are commonly used when conducting research - qualitative and quantitative. The quantitative approach is focused on proving a phenomenon or hypothesis through large sets of data and experiments [16]. The qualitative approach on the other hand focuses more on understanding a phenomenon and forming a hypothesis from sets of data or circumstances [16][17]. A qualitative research question often starts with a "How" or "Why" statement.

The methodology used for the research this thesis is based on is qualitative. The purpose of this thesis is to investigate how suitable LoRaWAN is for security applications. Through a qualitative approach, LoRaWAN and security systems are studied and analyzed, so that a hypothesis later can be formed to answer the research question. While the main question of this thesis is asked and answered in a qualitative way, the criteria are evaluated in a quantitative way allowing for the collected data and comparisons to be used in other studies.

The project consists of two parts, a theoretical and a practical. The theoretical part is done by a literature background study to collect data for a requirements specification for a remote battery powered security solution. Research is also done
on what kind of tests and experiments are needed to evaluate the performance of such a system. The practical part consists of implementing two identical prototype units of a remote battery powered security solution using LoRaWAN and executing the tests needed to verify the prototypes compliance with the requirements specification.

1.6 Delimitations

This thesis is focused on battery-powered security solutions that utilize motion sensors. It should be noted that only two prototypes have been created, that the two prototypes are identical in design, and that no practical tests have been done on a larger quantity of devices. Due to budget and time restrictions, the practical tests of this thesis focused only on the energy efficiency aspect of LoRaWAN.

1.7 Stakeholders

While the evaluation of LoRaWAN was requested by Slagkryssaren [18], the focus on remote motion security was decided upon by the authors of this thesis. Slagkryssaren is a Swedish Tech agency focused on software development. Slagkryssarens stake was to supply the hardware and work area for the project.

1.8 Outline of the thesis

The thesis is organized as follows. Chapter 2 presents the technical background of LoRaWAN. Chapter 3 gives a technical background of security systems, the motion sensor and the microcontroller used. Related work is also covered in Chapter 3. The research methods and strategies used for the research and work are discussed in depth in Chapter 4. The prototypes and the work are presented in Chapter 5. Chapter 6 presents the results, and Chapter 7 presents a discussion of said results, while conclusions and future work are covered in Chapter 8.
2. LoRaWAN

This chapter begins with Section 2.1 by introducing the basic functionality of low power wide area networks and the specific low power wide area network technology LoRaWAN. This is followed by Section 2.2, which introduces the specific LoRaWAN used in this thesis known as The Things Network. This section details the specific limitations imposed by The Thing Network. Section 2.3 goes on to describe the different device classes available for LoRaWAN devices and how they differ in terms of behavior and energy efficiency. Section 2.4 expands on the relationship between how fast and how far a message can be sent. This relationship is important when trying to find a balance between reliability and energy efficiency. In Section 2.5 the two methods used to connect to a LoRaWAN are explained. These two methods are important to know about in order to gain an understanding of how they impact the security aspect of a system. The security aspects of the message structure of LoRaWAN is presented in Section 2.6. The Sections 2.5 and 2.6 both provide background information necessary to fully understand the known vulnerabilities presented in Section 2.7.

2.1 Low Power Wide Area Networks

Low Power Wide Area Networks [4] are very useful for certain types of devices. The main attribute of a low power wide area network is that they are very energy efficient and therefore useful for applications that need to transmit and operate while using as little power as possible [1][3][4][9]. Two well known low power wide area network technologies are SigFox [19] and LoRaWAN [6]. The technology used in this project is LoRaWAN.

LoRaWAN is a radio-based network protocol and architecture in one. The standardization of the LoRaWAN protocol is managed by the LoRa Alliance [5], which is a non-profit, worldwide organization. The main aim of LoRa Alliance is to manage the standardization of LoRaWAN, as well as spread the low power wide area network technology. A LoRaWAN is composed of end-devices, Gateways, a Network Server and one or many Application Servers [6]. The network topology is illustrated in figure 2.1.

Anyone may set up a LoRaWAN-gateway and the owner of the gateway is responsible for establishing a connection to the network server. The connections are usually over the cellular network or Ethernet. The communication between the end-devices and the gateway on the other hand is over radio [20][21]. During transmission, the end-device broadcasts its message and all gateways within range will pick up and relay the message to their respective network server. That means that gateways connected to other networks will also pick up the message [22].
The network server does the routing and determines which application server the end-device is trying to reach and whether or not that network server has a connection to that specific application server. The message will be rejected and dropped by the network servers which do not have a connection to the right application server [20].

The network server is set up and maintained by the person, group, company or institution that is maintaining that specific LoRaWAN. Anyone can set up a private LoRaWAN. To set up a public LoRaWAN, like The Things Network [22], where others may roam freely, requires a LoRa Alliance membership [5]. The membership requirement is to make sure that the security of the public networks is not compromised.

The application server is set up and maintained by the owner of the end-devices that will be connecting to the application server. Depending on the solution and network, the network server and the application server may be handled by the same person, group, company or institution as the network server.

To join a network, a connection must be set up between the network server and the specific application server that will be receiving the messages from the end-device. Each end-device must then be registered, so that the network server knows which devices belong to which application server. An end-device can only be connected to one network at a time, but can with use of the Over The Air Activation (OTAA) join procedure roam between any networks that the end-devices application server has a connection to [13].

The wide coverage attribute of LoRaWAN is achieved by using the LoRa technology developed by Semtech [7]. LoRa is a long range protocol that operates over unlicensed radio bands [1][6][13]. LoRa is based on Chirp Spread Spectrum modulation, which was first developed in 1940s [6][8][23]. The main attribute of the modulation is that the signal frequency changes constantly, making it more resistant to interference [9]. This made chirp spread spectrum modulation very useful for various military purposes [8][23]. The different layers of LoRaWAN are illustrated in figure 2.2.
Depending on what country the end-device is located in, different radio bands are used. In Europe, the bands that are used are 863-870 MHz. The maximum number of available channels over LoRaWAN in Europe is 16, but it is mandatory for all end-devices to be able to support the following 3 channels with 125kHz bandwidth - 868.10, 868.30 and 868.50 MHz [20][21]. In the North America on the other hand, 902-928MHz bands are used instead and the number of available channels and regulations are different [5][6].

Besides restrictions on which radio bands are used, there are also restrictions on how much they can be used as well. To comply with these restrictions, set by the European Telecommunications Standards Institute [24], LoRaWAN uses a duty-cycle limitation. The limitation allows an end-device to transmit up to 1% of the time [21]. During one hours duty cycle a device is therefore allowed to transmit for 36 seconds in total. This means that all end-devices are limited in how often and how much data they can transmit [21][22].

### 2.2 The Things Network

The Things Network [22] is a world wide LoRaWAN. The Things Network consists of many communities in different cities and countries and provides access to a public LoRaWAN free of charge. The Things Network is a Lora Alliance member. The Things Industries [25] is a part of The Things Network and offer various private LoRaWAN solutions [22] for businesses.

The coverage of The Things Network relies on the communities members, which can be groups or individuals. The communities members set up the individual gateways, while The Things Network themselves provide the network server and a simple user interface [22]. The gateways need to be registered with The Things Network to be considered trusted. If a message from an end-device goes through a LoRaWAN gateway that is not considered as trusted, the message will be marked with a flag, to warn the application.

There are certain restrictions when connecting to The Things Network’s public network. The restrictions are a set of rules that all end-devices that are connected to that network should follow. This is to ensure a fair access for all users. The limits include that each end-device is allowed to use on average 30 seconds of uplink time on air and at most 10 downlink messages, including acknowledgement mes-
sages, from the application [26]. Depending on the distance between the gateway and the end-device and the message length are important factors when it comes to the average airtime needed. Solutions that will often be far away from a gateway, are in great need of bi-directional communication or have a large payload are advised to use a private LoRaWAN network or other solutions [26].

### 2.3 Device classes

There are three different end-device classes that are used within LoRaWAN - Class A, B and C. The classes were created to meet the different needs that end-devices and applications may have. All three classes allow bi-directional communication, but in very different ways and with a significant difference in energy usage [10][13][20].

Class A is the default class for all end-devices. While Class A devices allow bi-directional communication, the downlink receive opportunities are greatly limited. Class A devices can only receive during two short windows just after transmitting. If a message is sent to the end-device outside of those windows, the gateway will hold the message until the next transmission from the end-device has been received. How often an end-device is transmitting is individual for each end-device and network [13][21].

Figure 2.3 illustrates the transmit and receive windows for Class A devices. The delay lengths for the receive windows are the default recommended values as per the LoRaWAN regional parameter specification [21].

![Figure 2.3: The transmit and receive windows for Class A devices as described in LoRaWANs regional and technical specifications [13][21]. Figure was made by the authors.](image)

When it comes to the length of the receive windows, the following can be found the LoRaWAN Specification 1.1 [13] (citation):

"The length of a receive window MUST be at least the time required by the end-device’s radio transceiver to effectively detect a down-link preamble." [13]

In the technical specification it is also stated that if an end-device successfully receives a message during the first receive window, then it will not open the second receive window and a new message will have to wait until after the next transmission from the end-device [13].
Class B devices implement the same functionality for transmit and receive windows as Class A, but they also schedule additional periodical receive windows. To be able to continuously schedule another receive window, the Class B device receives a time synchronization beacon from the gateway [6][10][13]. The receive windows for Class B devices are illustrated in figure 2.4.

![Figure 2.4: The transmit and receive windows for Class B devices as described in LoRaWANs regional and technical specifications [13][21]. Figure was made by the authors.](image)

This type of communication is useful if there is bi-directional communication between the end-device and the application, where the end-device needs to repeatedly be contacted at a certain interval. The opportunities to periodically be able to contact the end device, come at a cost. Class B devices use more energy than Class A devices.

The last device class is Class C. Class C devices have a continuously open receive window, which is only closed when the end-device is transmitting. While this greatly lowers the latency between the application and the end-device, the amount of power required to almost constantly keep the receive window open makes Class C devices not suitable as battery powered applications [6][10][13]. Figure 2.5 illustrates the transmit and receive windows for class C.

![Figure 2.5: The transmit and receive windows for Class C devices as described in LoRaWANs regional and technical specifications [13][21]. Figure was made by the authors.](image)

All end-devices are assumed by the gateway and network server to be of Class A, unless anything else is specified [9][13]. For class B devices, a separate bit is set during uplink transmission to inform the gateway that the device is class B. For class C, the functionality is managed entirely on the application level by the developer of the application and the end-device.

A summary of the different classes can be seen in table 2.1. The restrictions and regulations for up-link transmission of end-devices are the same for all classes and depend on each individual network [13].
Table 2.1: The comparison of different device classes of LoRaWAN based on technical and regional specifications by LoRa Alliance [13][21]. The specifications for transmitting is the same for all classes.

2.4 Data rates

To be able to effectively scale the network depending on the number of connected devices, maximize the end-devices battery life and keep a good ratio between data rate and range, LoRaWAN implements something called Adaptive Data Rate [6][13].

Adaptive data rate means that the data rate for each end-device is frequently being reevaluated and adapted depending on the status of the network. The status of the network consists of parameters such as the total number of end-devices, the minimal time on air that each end-device needs and the quality of the signal and range between the specific end-device and the gateway [6][13].

The data rate (DR) for each end-device is managed by the network server. There are different algorithms and techniques for network servers to implement adaptive data rate in an effective and fair way. However, adaptive data rate is only implemented by the network server for relatively stationary end-devices [13]. Devices that constantly move and change their location very fast are recommended to implement this on the application level instead [5].

When talking about data rates, another factor becomes important - the Spreading Factor (SF). The spreading factor is inversely proportional to the data rate. Higher spreading factor means lower data rate. While the data rate describes how fast data is transfered, generally in bits per second, the spreading factor describes how far the message will reach and how resilient the transfer is to interference. This is commonly compared to a conversation between two people. Data rate is how fast you speak, while spreading factor is how clearly you want to say it. If two people are standing close to each other and the listener (gateway) is attentive, then the speaker (end-device) can usually speak a lot faster without fearing that the listener will miss some of the words (packetloss in a network). While if the listener (gateway) is a bit busy, then the speaker (end-device) may need to talk a bit slower (to avoid packetloss) or faster(to give others time to talk as well). There are 6 different data rates and spreading factors, DR0-DR5 and SF7-12. SF12 is used with DR0, which results the lowest and slowest data rate. SF7 is used with DR5, which means that a lot more data can be sent much faster [13]. The different SF allow several end-devices to transmit to the Gateway over the same channel,
2.5 Activation of an end-device

There are two ways for an end-device to join a LoRaWAN. Regardless of the method used, the end-device must first be registered with the network server. The method of activation is also chosen at this stage and allows the network server to establish the right connection.

The first method of activation is Over The Air Activation, commonly referred to as OTAA in documentation. The end-device sends a join-request message consisting of the end-device's device identifier, application identifier and join request counter. The message consisting of these three is then signed using an encrypted, 4-byte long field. The encryption used is a well-known symmetric encryption algorithm for 128 bits - AES128 [27]. Unlike normal messages, the join-request message in itself is not encrypted. Provided that the end-device is registered and allowed to join the network through OTAA, the network server will return a Join-Accept message. The message consists of a network identifier, the end-device's address within the current network (assigned by the network server), downlink settings and the delay parameter for the receive windows. Once the end-device receives the join-accept message, the end-device will derive 4 session keys (application session key, network session key, serving network session integrity key and forwarding network session integrity key) that will be used to encrypt and decrypt the messages between the end-device and the application server [13].

In order to use over the air activation for roaming between networks, the end-device should be configured to repeat the OTAA process at set intervals since the end-device can’t easily detect changes in available networks [13].

The second way for an end-devices to join LoRaWAN is Activation by Personalization (ABP). This activation method is usually used when an end-device will only be connected to one certain network and does not allow roaming. The difference between activation by personalization and over the air activation is how the keys are handled. When using OTAA, new keys are created each time the end-device joins the network. When using ABP however, the keys are static and do not change when the end-device disconnects and then connects to the same network again. The keys are instead stored on the end-device at the production stage. For security purposes the keys must be unique for each end-device and stored in such a way, that they are not accessible from outside [13][21].

2.6 Technical overview and security of messages

The technical specifications detailed in this section is taken from the original LoRaWAN specification v1.1 [13]. The structure of a LoRaWAN uplink message can be seen in figure 2.6. This section will only detail the parts of the LoRaWAN message structure that is directly relevant for this thesis. The full details of the message structure can be found in the LoRaWAN specification [13].

The physical layer message is used by the LoRa protocol. The physical message contains two fields of special interest to us, namely the Cyclic Redundancy Check (CRC) and the physical payload. The cyclic redundancy check is used to check the...
Figure 2.6: Uplink message structure as described in the LoRaWAN Specification 1.1 [13]. Figure was made by the authors.

...integrity of the payload in uplink messages. The physical payload contains the LoRaWAN message.

Within the MACPayload field of the LoRaWAN message is the message frame. The frame payload contains the actual data sent and is encrypted using an AES128 encryption, with the application session key as encryption key. The application session key is only known by the end-device and the application server, the contents of the payload are ensured to be end-to-end encrypted.

Within the Frame Header there is a field called Frame Counter. The frame counter starts at zero and increments for each message sent. The frame counter is used by the receiver to prevent play-back attacks by only allowing messages with a frame counter higher than the last message.

The Frame Payload contains the overhead of the LoRaWAN message. The overhead of a LoRaWAN message is 13 bytes in size, meaning that the smallest message that can be sent with LoRaWAN is 14 bytes out of which 1 byte is the Frame Payload.

LoRa distinguishes between uplink and downlink messages but they are similar in structure, the difference between uplink and downlink messages is that only the uplink messages contain the CRC field, this is so that a faulty uplink message can be discarded right away by the Gateway instead of forcing the network server to deal with faulty messages. Downlink messages do not have the CRC field since it is sufficient for the end-device to detect errors using the MIC as the end-device is the one to check the message integrity anyway.

In the LoRaWAN message the Message Integrity Code (MIC) is of special interest. The MIC is used to ensure that there are no errors in the message and that the message has not been tampered with. The security of LoRaWAN messages is largely centered around three points, the previously mentioned encryption of the frame payload, encryption of the frame options, and the generation of the MIC.
To generate the MIC the two encryptions cmacS and cmacF are used.

\[ \text{msg} = \text{Message type} | \text{Frame header} | \text{Port field} | \text{Frame Payload} \]

\[ \text{cmacS} = \text{aes128_cmac(serving network session key, B1 | msg)} \]

\[ \text{cmacF} = \text{aes128_cmac(forwarding network session key, B0 | msg)} \]

Where B1 and B0 are two different blocks of data containing frame counters, message length and transmission rates among other things. The MIC consists of the first two bytes of cmacS followed by the first two bytes of cmacF. As the generation of the MIC is done using the secure serving network session key and forwarding network session key the MIC can’t be correctly generated by an illegitimate source. Therefore depending on if the message is an uplink or downlink message either the network server or the end-device can check the MIC to make sure that the message is valid.

2.7 Known vulnerabilities

While LoRaWAN does use encryption, integrity checks and other security measures, LoRaWAN is not without its weaknesses. This section will go over three known vulnerabilities of LoRaWAN.

The first and perhaps the greatest source of vulnerabilities is ironically the frame counter meant to stop replay attacks. The frame counter is a 16 bit field that starts at zero and increments for each message sent. Only messages received with a higher frame count than the last message received get accepted [13]. The frame counter vulnerability arises when the frame counter reaches the maximum value, at which point the counter is reset to zero. If any malicious entity has picked up any previous message, that message can now be re-sent causing the receiver to expect a higher frame counter than the real end-device is sending. This causes the receiver to ignore any message sent by the real end-device [28].

The frame counter is mainly an issue when using activation by personalization since the different keys will always stay the same. While using over the air activation the end-device can be set up to rejoin the network and thereby generate new keys when the frame counter reaches maximum value. By generating new keys a replay attack with a high frame counter will not work, since the replay message will have old obsolete keys.

It is worth to note that the maximum value for the frame counter is 32768 [13] and that the 1% transmission regulation [21] makes it so that reaching the maximum frame counter will take a significant amount of time (approximately one year at a send rate of four messages per hour). The long reset time of the frame counter also works against eavesdropping attacks. If a malicious entity were to pick up three or more messages with the same keys and the same frame counter value the malicious entity could be able to decipher the messages [28]. The long time required to gather three or more messages with the same frame counter makes the risk of eavesdropping far less prevalent. However this assumes that a malicious entity does not get physical access to the end-device in which case frame counter resets could be triggered more frequently making eavesdropping a very realistic threat.
The previously mentioned replay and eavesdropping attacks can generally be mitigated through the use of OTAA joining process as those attacks abuse the fact that activation by personalization keeps using the same keys. The OTAA joining process is the second source of vulnerability as the join request sent by the end-device is not encrypted. However no research pointing to it being a significant vulnerability was found. There exists some research pointing to ways to abuse the lack of encryption in the join procedure [29][30], however those attacks would only be possible against very specific setups where the end-device sends out join requests frequently and at regular intervals [30].

The third vulnerability in LoRaWAN is that if an end-device sends a message requiring an Acknowledgement (ACK), there is nothing in the acknowledgement that specifies what message the acknowledgement is for. The acknowledgement will simply validate the last message sent by the end-device even if it was not the message that the acknowledgement was created for. This could be abused if a malicious entity were to be in control of the LoRaWAN gateway used, the acknowledgement returned by the network server could be held by the gateway and the next time the end-device tries to send a message through the gateway, the gateway discards that message and sends the previously held acknowledgement to the end-device making the end-device believe that the last message sent went through successfully [28]. Since end-devices that require an acknowledgement can be assumed to send messages that must reach the application server at a higher rate than an end-device that does not require an acknowledgement, this attack could prove to be quite harmful, although this is not the easiest attack to perform. The prevention of an acknowledgement-attack lies largely in the hands of the network provider making sure that all gateways can be trusted.
3. Security systems, motion detection sensors and related work

There are many different types of security and alarm systems. The systems can be implemented in many different ways, using a variety of sensors, platforms and computing algorithms. This chapter provides a general overview on security and alarm systems which focus on motion detection. In Section 3.1, the European grading standard of security systems is presented. The grading system makes it easier to compare different security and alarm systems depending on the requirements. Section 3.2 introduces a common type of sensors used for motion detection systems - passive infrared motion sensors. This is followed by Section 3.3, which covers a common type of microcontroller board used for small applications. The last section, Section 3.4, presents related work and research, not only on security systems and motion sensors, but also LoRaWAN.

3.1 Grading of security and alarm systems

To make it easier to compare different security systems based on requirements for functionality, reliability and resistance to interference, a grading system was introduced. The grading system is a part of a general European standard for intrusion and alarm systems (EN 50131-1:2006). The standard is used within the whole European Union and is continuously being updated. The last revision of the standard is from 2018. Due to resource access problems, the official Swedish translation for the European standard was used - SS-EN 50131-1 [31]. The Swedish translation has the same status and content as the original European standard.

The grading systems and grade guidelines are based on the risk of theft, the expected knowledge of the thief/intruder and the thief’s/intruder’s expected resources. The English translation, which was done by the authors, of the grades presented in the European Standard document in Swedish can be found below.

**Grade 1:** Low risk of theft/intrusion
The trespasser or burglar are expected to have almost no knowledge about alarm systems and to be limited to very few, easily accessible tools.

**Grade 2:** Low to medium risk of theft/intrusion
The trespasser or burglar are expected to have limited knowledge about alarm systems and have accesses to a general range of tools and portable instruments, such as multi-meter.

**Grade 3:** Medium to high risk of theft/intrusion
The trespasser or burglar are expected to be very familiar with alarm systems and have an extended range of tools and portable electronic instruments.
**Grade 4: High risk of theft/intrusion**

Used when the security is the first priority factor above all else. The trespasser or burglar are expected to have the ability or resources to plan a burglary or intrusion in detail. The trespasser or burglar are expected to have access to full range of tools and equipment, including replacement parts for the vital parts of the alarm or security system.

The scenarios described in the grading specification are general descriptions on what situation a system of that grade should be able to withstand. Grade 1 represents systems that are easy to overcome or interfere with. Grade 4 represents the most secure type of systems, which require very specific skills and knowledge to disarm.

There are also individual specifications for different types of alarm and intrusion systems. For example, there is a separate specification for systems using passive infrared motion detection sensors - SS-EN-50131-2-2 [32]. In that specification more concrete criteria are given for that type system to be considered to be of a certain grade. For example, to be considered to be Grade 1, which is the lowest grade, a system with a motion sensor only has to be able to detect intruders. For Grade 2 however, a system using passive infrared motion detector sensors needs to be able to handle events such as total loss of power supply.

### 3.2 Passive Infrared Motion Detection Sensors

To detect an intruder, a security and alarm system needs some kind of sensor. A common type of sensor that is used in security systems is motion detection sensors. Many motion detection sensors use infrared radiation to detect a change in the sensors environment.

A Passive Infrared (PIR) motion sensor uses pyroelectric sensors to detect changes in temperature through the electric reaction of certain crystals [33]. In this case the change in temperature comes from infrared radiation. A PIR sensor uses multiple pyroelectric sensors to identify movement through detecting changes in infrared radiation. For example a human body moving in front of the sensor will give off a different amount of infrared radiation than the surrounding area causing the sensor to trigger [34][35].

The standby power required for a PIR sensor that uses a pyroelectric triggers is very low [15][36]. When triggered the sensor will let power flow through, which can be used as a logical high input when connecting the sensor to a digital device [34][35].

A PIR sensor operates just as well in the dark, as when there is light and can be used for many different things [34][35][36][37][38][39]. One type of application is smart homes and offices. By using PIR sensors to determine if someone is present or not, it is possible to automate the turning on and off of lamps, air conditioning and heating [34][38][40][41]. If there is no one present in the building, the lights can be turned off and air conditioning can be lowered, thus saving energy. This is very useful in offices, where people may forget to turn off lights and air conditioning when leaving for the day or move a lot through the premises [40][41].
Another area where PIR sensors are very commonly used is security and surveillance systems. There are many different ways to use PIR sensors in security systems [35]. If several PIR are positioned correctly, they can be used to track a persons movement within a room or building [35]. Combined with a camera, PIR sensors can be used as an on trigger for the camera. When the PIR sensor is triggered, the signal will be forwarded to the camera and turn it on, to monitor what is happening. That way, the camera can be idle until something happens, saving both energy and hard drive space [42]. The sensors can also be used as a security measure to help elders that live alone in case of an accident or fall [36].

3.3 Microcontroller boards and Arduino

In a security system, a sensor is usually connected to a microcontroller board or a computer, to process the input. For small units that rely on battery power or need to be power effective, microcontrollers boards are commonly used. A common type of board used for research or home projects is Arduino.

Arduino is an open-source hardware and software platform. Arduino started out as a project at Ivrea Interaction Design Institute and was first created in 2005. The main aim is to provide a platform that is affordable and easy to program and can be used even by complete beginners [14][43].

The hardware part of Arduino consists of microcontroller circuit boards. There are many different varieties, depending on what the board will be used for. The Arduino UNO board is the simplest and most robust one, primarily targeted at beginners. The Arduino Lilypad boards are designed to be wearable and be able to be integrated with clothes. Other Arduino boards incorporate LoRa chips and make it possible to use LoRa technology by just adding an antenna to the board. All board designs are open-source and can be used by others, remodeled and improved [14].

To extend a boards functionality, one or several Shields can be added to it. While the main Arduino board contains the microcontroller, a shield is a board that extends the main board functionality. There are shields that are specifically designed for motors, design custom circuits or allow wireless access [14]. Before boards with integrated LoRa-chips were available, it was necessary to use a shield together with an Arduino board to be able to use LoRa technology. There are different boards with extended functionality being sold both by Arduino themselves, but also others. The Things Network have their own custom board, based on an original Arduino, which allows LoRa-functionality without a shield.

To write, compile and upload software to an Arduino board, a cross-platform Arduino IDE is available. The IDE is available for Windows, Mac OS and Linux, as well as online. The programming language used to write the programs for the Arduino is very similar to C and C++. At the moment it is not possible to write in other programming languages in the Arduino IDE [14]. There is however a library, called Firmata library [44], which implements the Firmata protocol [45] and makes it possible to execute software from a computer or a tablet that is connected to the Arduino board. By using the firmata library it is possible to implement and run some programs in other programming languages, like python [46].

An arduino program is called a sketch. Every sketch usually contains two key components - an initial setup function and a loop function. The setup function
contains the commands that need to be executed once at the beginning of when the program starts to run. The loop function contains the main body of the program and provides the actual functionality [14].

The board that was used in this project is called ”The Things Uno” [47]. The board was developed by The Things Network [22] and is originally based on the Arduino "Leonardo" board [48]. The Things Uno contains all of Leonardo’s functionality and has the same overall schematic. The difference is that The Things Uno contains a built-in LoRa-module and internal antenna. This means that no external shields or parts are needed to use the board for LoRaWAN connectivity. An external antenna can be added to the board for increased range. The LoRa module used by The Things Network for the the board is the Microchip RN2483 [49]. The Things Network also provides an Arduino library of their own for managing the LoRaWAN connectivity [50]. The library provided by The Things Network has the advantage of being easy to use, with predefined functions for joining networks and sending messages. The library does however have the disadvantage of being limited to the predefined functions, making any action without a predefined function difficult to perform.

3.4 Related work

Using PIR sensors for motion detection and surveillance is not a novel idea. A lot of research is being done on how to utilize motion detection sensors for a variety of purposes. The majority of the research done is based either on fixed placement of the sensors and detection systems, as well as a fixed network connection. In a conference paper from an Information Networking conference in 2008, a method of tracking a person position within a room with the help of PIR sensors was proposed [35]. In this article from 2010 [34], a wireless cluster of sensors are used to track a persons movement through a hallway. Each individual node in the cluster uses PIR sensors for motion detection and a microcontroller board to handle the inputs. A short-range low power network technology alternative to LoRaWAN was used for communication between the nodes in the cluster. And in the following conference paper for a video and signal based surveillance [42] a method for how wireless sensor nodes, using PIR sensors, can be used for surveillance and triggering of a video camera. Wireless nodes that use PIR sensors and video in combination for surveillance purposes are also covered in another article [51] from 2012.

This provides a good insight on how PIR sensors can be used for motion detection and tracking, but not much in the way of power efficiency or battery-powered systems. While power efficiency is mentioned in several of the articles above, a high power efficiency is not the main aim of any of the researches mentioned above. The sensors, while wireless, are stationary and do not need to be battery-powered only.

Power efficiency is one of the main goals of low power wide area networks. For the efficiency of a low power wide area network technology to be relevant it must work on a power efficient platform. An online article on the home-automation-community website goes in to great detail on how to make an Arduino pro mini more power efficient [52]. This is done by removing some unnecessary components such as the power LED and the voltage regulator. By making these modifi-
cations the author of the article manages to reduce the power consumption of the board from 4.74 mA in active mode and 0.90 mA in sleep mode, to 3.9 mA active and 0.054 mA sleep after only removing the LED.

A conference paper from 2017 [10] presents research done on LoRaWAN and the comparison of power consumption for Class A and Class C devices, which gives a good understanding of possibilities and limitations of both classes. An article on LoRaWANs energy performance for Class A devices [53] describes how the different data rates and transmission patterns affect the power consumption of the end-devices. There are several other works that cover various attributes of LoRaWAN and how these affect energy consumption and range. One paper compares LoRaWAN to other low power wide area technologies [1]. Another paper investigates theoretical energy efficiency of LoRaWAN for different spreading factors and ranges [54]. An article from IEEE Communications Magazine presents the different limitations of LoRaWAN, such as packet size, time on air and how it scales with different number of end-devices [9].

Besides being power efficient and accurately detecting motion, the remote battery powered security system prototypes developed for this thesis also need to be secure and reliable. Various research has been done on vulnerability and security of LoRaWAN protocol and networks. A master thesis from Delft University of Technology in the Netherlands LoRaWANs vulnerabilities and how they can be exploited [28]. A conference paper from Information Networking conference in 2017, presents countermeasures for replay attacks on LoRaWAN during join procedures. A paper from another conference describes how LoRaWANs network architecture can be improved from a security standpoint [55].

Another aspect of security is a systems reliability. There have been studies done on LoRaWANs performance indoors [56] and in urban areas [57] to gain a better understanding of the reliability of a LoRaWAN based system.
4. Project methods and methodology

Earlier in Section 1.5, two main research methodologies were briefly mentioned and explained - qualitative and quantitative. Quantitative research analyses large datasets and uses that data to try to disprove or strengthen a hypothesis. Qualitative research on the other hand uses smaller sets of data and tries to draw conclusions from the results [17]. In short, the main distinguishing factor between qualitative and quantitative research is that quantitative can be said to be numerical while qualitative is non-numerical [16]. The methodology used for this research is qualitative.

A research’s methodology shapes the methods used in the project related to the research. This chapter presents the methods and strategies used for this project. Section 4.1 presents the overall methods and approaches used for this project. The methods for data collection are covered in Section 4.2. This is followed by Section 4.3, which presents the data analysis methods used for this project. The chapter is concluded with Section 4.4, which covers quality assurance.

4.1 Research approach and strategy

There are several different ways research can be approached, the two main one’s being the inductive way and the deductive way [16][17]. In inductive reasoning, conclusions are drawn based on the data collected, while in deductive reasoning a hypothesis is tested [16][17][58]. Usually, qualitative research is approached in an inductive way, while research conducted in a quantitative way is usually uses a deductive approach.

The approach used for this research was inductive. The main research question is how well do the properties of LoRaWAN meet the requirements for a battery powered security solution? The main aim of the research is to conclude the level of suitability of LoRaWAN based on empirical studies and not verification of whether or not LoRaWAN is suitable for battery powered security solutions, making the question an inductive one.

There exists several different inductive research strategies, where case study is one [16]. The main aim of case study is to study a phenomenon through empirical studies. A research question is set at the beginning, followed by data collection aimed to answer that question and finished with analysis of the data and answer to the research question [58]. Due to the nature of goal and research question for this research, case study was a natural choice of research strategy.


4.2 Data Collection

There were two sets of data that needed to be collected for this research. The first set of data was necessary to form a requirements specification for the remote battery powered security solution. Without a requirements specification, it is impossible to answer the research question in constructive way, as "suitability" can be very subjective and depends on the observer. The data for the requirements specification was gathered in two steps. The first step consisted of interviews with the stakeholders to gather information on their potential requirements for the system. The second step was to conduct literature studies to collect information on the general requirements for security systems. The data collected from the interviews and literature studies were then compiled in a requirements specification.

The second set of data was collected through experiments on two identical end-device prototypes made by the authors. The practical experiments were conducted to collect the data about the power consumption of the prototypes and LoRaWAN. To better understand which experiments to perform, data about various attributes of the network protocol and hardware used for the prototypes was collected. This was done through literature study of specifications for the parts and technologies that were used.

4.3 Data Analysis

As previously mentioned in Section 4.2, there were two sets of data that were collected. One set of data was collected for the requirement specification and the second set of data was collected through power consumption experiments. Different data collection methods often require different data analysis methods [58].

The set of data that was collected through practical experiments is quantitative [16][17] and is therefore presented and analyzed through statistics. Statistics is a common way to present large sets of data from experiments.

The set of data that was gathered for the requirements specification was analyzed qualitatively. The attributes of the end-device prototype that was implemented in this project were compared to the requirements specification and evaluated.

The conclusions for the case study were based on the analysis made through statistics and the results of the qualitative analysis.

4.4 Quality Assurance

The main research methodology used for this research was qualitative and the research quality in general is therefore evaluated based on those criteria. The experiments that were performed to measure the power consumption of the end-device prototypes, were done in a quantitative way. The quality of the results from those experiments are therefore evaluated according to quantitative criteria.

Complete description of the construction of the prototypes, the parts used and the way the experiments were conducted can be found in Chapter 5. The code for the prototypes, readings and application is provided in Appendix A, making the experiments reproducible by a third party. Results of the experiments are presented as numerical data in Chapter 6, making it possible for others to compare their own results. In Section 6.5, the requirements specification is applied
on the results from the experiments and literature studies. By using a require-
ments specification to define what is deemed suitable, the reader is provided with
concrete criteria for the definition of suitability, decreasing the possibility of mis-
understandings of the definition due to subjectivity.

All sources and bibliography that were used for this thesis are listed in the Bibliography section, making it possible to verify the data and information gathered through literature studies.
5. Requirements specification, practical implementation and experimental setup

In order to gather data on the properties of the end-device prototypes and LoRaWAN in general, some practical work was conducted. First a requirements specification on which to base the evaluations on was created. The creation of the requirements specification is detailed in Section 5.1. Then two identical end-device prototypes were constructed, on which the requirements specification was later applied. The construction and programming process of the prototypes are detailed in Section 5.2. When the two end-device prototypes were up and running, a rudimentary application side was configured using The Things Network’s pre-built interface. That process is described in Section 5.3. In order to be able to evaluate the power efficiency of the end-device prototypes, two experimental setups were set up as described in Section 5.4. Using the experimental setup a number of tests were conducted. The details of these tests and the handling of the readings can be found in Section 5.5.

5.1 The requirements specification

A requirements specification was needed to answer the research question of this thesis - "How well do the properties of LoRaWAN meet the requirements for a battery powered motion-detection solution?".

The areas for evaluation are energy efficiency, transmission time, reliability and vulnerabilities. In order to define what would constitute an "acceptable", "not acceptable" or "preferred" grade, data collection was conducted for each of the mentioned areas. To define the grades for the energy efficiency an interview was held with the stakeholders. It was determined that for a battery powered security solution to be useful, the solution needs to be able to stay active for at least 6 months and preferably more than 24 months. Assuming a 10000mAh battery these numbers would translate to an average consumption of approximately 2.3mA to be acceptable and 0.58mA to be preferable.

The transmission time gradings were mainly defined by the time it would take an intruder to reach the battery powered security device and disable it. If an intruder was to reach the battery powered security device and destroy it before the battery powered security device could finish sending its message, the system would fail. It was estimated that an intruder without prior knowledge of the system would need at least five seconds to notice and reach the battery powered security device in an average size room. It was also estimated that an intruder with prior knowledge would not be able to reach and destroy the battery powered security device within one second. Therefore one second and five seconds were chosen as the thresholds for grading the transmission time.

Reliability in this case refers to how reliably LoRaWAN delivers the messages...
sent. As LoRaWAN generally operates without acknowledging messages received, any packet that is lost in transmission will not be re-sent and is thus permanently lost. This means that the rate of packet loss must be very low for the battery powered security device to be considered reliable. For the reliability grading of the battery powered security device the the amount of risk of an alarm not reaching the user were used as thresholds. A 1% risk of an alarm not reaching the end user was considered acceptable and 0.1% risk or lower was considered preferable.

When grading the vulnerability of the battery powered security device the European standard for alarm and intrusion systems that was introduced in Section 3.1 was used. For the evaluation of this device "Grade 1: Low risk of theft/intrusion" of the European standard was deemed to be acceptable and "Grade 2: Low to medium risk of theft/intrusion" of the European standard to be preferable. Failing to meet any of the European standard gradings was deemed not acceptable.

The final requirements specification can be seen in table 5.1.

<table>
<thead>
<tr>
<th>Area/Grade</th>
<th>Not acceptable</th>
<th>Acceptable</th>
<th>Prefferable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy efficiency (e)</td>
<td>$e &gt; 2.3\text{mA consumption}$</td>
<td>$e &lt; 2.3\text{mA consumption}$</td>
<td>$e &lt; 0.58\text{mA consumption}$</td>
</tr>
<tr>
<td>Transmission time (t)</td>
<td>$t &gt; 5$ seconds</td>
<td>$t &lt; 5$ seconds</td>
<td>$t &lt; 1$ second</td>
</tr>
<tr>
<td>Reliability (R)</td>
<td>R &gt; 1% to miss alarm</td>
<td>R &lt; 1% to miss alarm</td>
<td>R &lt; 0.1% to miss alarm</td>
</tr>
<tr>
<td>Vulnerability</td>
<td>No grade reached</td>
<td>Grade 1 reached</td>
<td>Grade 2 or higher reached</td>
</tr>
</tbody>
</table>

Table 5.1: The requirements specification which was compiled by the authors.

By evaluating the end-device prototypes using the requirements specification a conclusion could be reached as to how well this specific prototype solution meets the requirements. Based on that conclusion it was possible to extrapolate which strengths and/or weaknesses can be attributed to LoRaWAN and which strengths and/or weaknesses lies with other parts of the end-device prototypes.

5.2 End-device prototypes

Two end-device prototypes were constructed for test purposes. One end-device prototype consists of a PIR motion sensor, a "The Things Uno" microcontroller board and a 10400mAh power bank. The components of the prototype were connected as shown in figure 5.1 on the next page. The PIR motion sensor is connected to the 3.3 volt pin and the ground pin of the board for power. The middle pin of the PIR motion sensor is connected to pin 2 of the board, which is a digital input/output pin. The end-device prototypes are powered through the on-board micro-USB connector. The board is connected to the power bank via a USB to micro-USB cable.
The end-device prototypes were programmed with a behavior illustrated in the flowchart in figure 5.2.

To interact with the LoRaWAN unit on the board, The Things Network’s own LoRaWAN library was used. The Things Network’s library was used mainly for its ease of use. On startup the end-device prototype will send an over the air activation join request in order to join the network. When the end-device prototype has successfully joined the network the end-device prototype will enter a loop where it continuously checks if it is time to send a heartbeat or if the PIR sensor has been triggered. A heartbeat is a 1 byte message containing only zeros meant to indicate that the device is functioning as intended. If the PIR sensor has been triggered the end-device prototype will start by sending a LoRaWAN message containing a ‘1’ to indicate the start of an intrusion. The end-device prototype will then enter a log-mode. The log mode will log the sensor events over a period of 3 minutes with
a resolution of 6 seconds. Every 6 second interval is represented as one bit in the log. Any event during the presiding 6 seconds will result in a ‘1’ in the log. Any 6 second interval without events will be represented with a ‘0’ in the log. After 3 minutes the end-device prototype will send the collected log over the LoRaWAN. This log mode was designed in order to comply with the strict air time restrictions of LoRaWAN and The Things Network. By using the log mode, detailed information about an intrusion can be sent without exceeding the airtime restrictions. After the log has been sent, the end-device prototype will return to the loop checking the heartbeat timer and sensor trigger. The code for the end-device prototypes can be found in Appendix A.1 A summary of the different messages sent by an end-device prototype is presented in table 5.2.

Table 5.2: Message types sent by an end-device prototype.

<table>
<thead>
<tr>
<th>Message types</th>
<th>Size</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heartbeat</td>
<td>1 Byte</td>
<td>00000000</td>
</tr>
<tr>
<td>Alarm</td>
<td>1 Byte</td>
<td>00000001</td>
</tr>
<tr>
<td>Log</td>
<td>4 Byte</td>
<td>event dependent</td>
</tr>
</tbody>
</table>

5.3 The application

For this project, the application was the user interface provided by The Things Network. The goal of this thesis was to investigate LoRaWANs suitability and developing a full scale application was therefore out of scope for this thesis. This user interface provided by The Things Network is called the console. The console requires a registered user and one or more end-device to be registered to that user. After the registration of a user and devices, the user can in real time, see what messages are being sent to and from the end-devices.

The messages between the end-device and the application are per default sent as bytes and presented in the console in the hexadecimal format. However, the console interface allows the user to write simple decoding functions to present the message in other ways and if desired forward it to another application. The functions have to be written in JavaScript.

For this project, the console was used to verify the messages. Simple functions were written by the authors of this thesis to more easily distinguish the various messages sent and verify their contents. Three functions were written by the authors. The fist one was a decoder. The decoder receives the message from the LoRaWAN network server as an argument and checks if it is a log type of message or an alert or heartbeat type of message. The message is then passed on to the converter function, which converts the contents to human readable format. For logs, the format was a binary sequence to see during which intervals the sensor was triggered. For alerts and heartbeats, the message was converted to decimal base number. The last function was a validation function. The function filters the messages, so that only messages of the correct type and size are accepted. The code for all three functions can be found in Appendix A.3.
5.4 The experimental setup for measuring power consumption

In order to evaluate the power efficiency of the LoRaWAN technology a series of measurements were done on the prototypes. The devices used for the measurements were a Fluke 45 digital multimeter for the current and voltage measurement, a computer to log the measurements, the prototype devices and a USB power supply used to power the prototype devices.

Two setups were used for the measurements and can be seen in figure 5.3 and figure 5.4. In the first setup, a prototype was connected to the power source through a micro-USB interface. In the second setup the power source was connected to the voltage (V\textsubscript{IN}) and ground pins available on the board used for the prototypes. To be able to take measurements using a power supply with a USB interface, a USB cable had to be modified. For both setups, the measurements were done one prototype at a time.

![Figure 5.3](image1.png)

**Figure 5.3:** Experiment setup for measuring power efficiency, where a prototype was connected to the power source through a micro-USB cable. Figure was made by the authors.

![Figure 5.4](image2.png)

**Figure 5.4:** Experiment setup for measuring power efficiency, where the power source was directly connected to the board of a prototype. Figure was made by the authors.
For the first setup, a USB extension cable was cut open and the inner red 5 volt (V) wire was split into A1 and A2 as shown in figure 5.3. A1 was then connected to the 100 milliamperes (mA) port of the multimeter and A2 was connected to the common terminal (COM) port of the multimeter, thereby allowing measurement of the current. The inner black ground wire had the insulating plastic peeled back and B was attached to the wire. B was then connected to the Volts (VΩ) port. This connection allowed the voltage difference between A2 and B to be measured.

For the second setup, A2 was replaced by an ordinary, extra cable. A2 was then connected between the 100mA port of the multimeter and the V<sub>IN</sub> pin of the board for a prototype. A1 and B were connected as in the first setup. An extra cable was attached to the black ground wire of the USB cable and to the GND pin of the board.

The initial idea was to make concurrent measurements of both current and voltage during all tests. However for the Fluke 45 to make concurrent measurements of current and voltage, the Fluke 45 employs the use of a relay to switch between the two types of measurements. This switching effectively halves the frequency of the readings. Due to the short transmission times to be measured, halving the measurement frequency introduced an increased risk of missing values. This would mean that some of the transmissions would be missed partly or completely and provide incomplete information. Instead the voltage behavior was measured separately. By measuring the current and voltage separately, the risk of missing readings decreased, although final power consumption calculations may need to have larger margins for error. While simultaneous and sufficiently frequent measurements could be achieved using an additional multimeter, resource and time limitations caused the separate measurement strategy to be used.

In order to accurately analyze the measurements, the Fluke 45 multimeter was connected to a computer using a Serial to USB cable. The Fluke 45 was set to “print-only mode 1” and the measure rate to medium. These settings result in one measurement every 0.2 second with a resolution of 0.01 mA. The computer was set up to log the current readings from the Fluke 45 multimeter using the custom made python program found in Appendix A.4. The python program can easily be edited to set the duration of each test and the number of iterations of the test to run. The python program writes all measurements to text files, with one file per iteration. The text files was then imported in to Google Sheets where the readings were analyzed.

5.5 Measuring and handling of readings

Using the measuring setups described in Section 5.4 a number of measurements were made. Each test-case consisted of 10 iterations of tests, or measurements, on each of the two end-device prototypes. During the measurements where spreading factor 12 was used, only 3 messages were sent over a longer test duration. This is to comply with the duty-cycle limitations explained in Section 2.1 and 2.2. The code used on the end-device prototypes during the tests can be found in Appendix A.2.
The first set of measurements were made on the current. Different payload sizes and different spreading factors were used on both boards, as well as two different power connections. There were 6 different test cases, the settings for each case are listed in table 5.3.

<table>
<thead>
<tr>
<th>Payload</th>
<th>Spreading factor</th>
<th>Number of sends</th>
<th>Test duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Byte</td>
<td>SF7</td>
<td>10</td>
<td>300s</td>
</tr>
<tr>
<td>4 Bytes</td>
<td>SF7</td>
<td>10</td>
<td>300s</td>
</tr>
<tr>
<td>1 Byte</td>
<td>SF12</td>
<td>3</td>
<td>360s</td>
</tr>
<tr>
<td>4 Bytes</td>
<td>SF12</td>
<td>3</td>
<td>360s</td>
</tr>
</tbody>
</table>

Table 5.3: Settings for test cases for current measurements.

The second set of measurements were made on the voltage. Different spreading factors and power connections were used, while the payload was kept the same throughout the measurements. Due to time limitations fewer voltage measurements were made than for the current. The settings for test cases for voltage are presented in table 5.4 on the next page.

<table>
<thead>
<tr>
<th>Payload</th>
<th>Spreading factor</th>
<th>Number of sends</th>
<th>Test duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Bytes</td>
<td>SF7</td>
<td>10</td>
<td>300s</td>
</tr>
<tr>
<td>4 Bytes</td>
<td>SF12</td>
<td>3</td>
<td>360s</td>
</tr>
</tbody>
</table>

Table 5.4: Settings for test cases for voltage measurements.

After the measurements, the results were used to calculate some of the following things - overall average for each iteration of measurements, average current/voltage when idle, average current/voltage for transmission of a message, average cost of sending one message.

To calculate the cost of sending a single message in ampere hours, the formula presented in equation 5.1 was used.

\[
\text{Message cost in Ampere hours} = \frac{\text{Overall Average} - \text{Average Idle}}{\text{Number Of Sends} \times \frac{3600}{\text{Test Duration}}} \quad (5.1)
\]

First, for each iteration, the overall average current and the average current when idle were calculated. The difference between the two shows the contribu-
tion LoRaWAN made to the overall average. By dividing this contribution by the number of messages sent, the average contribution of a single message could be isolated. The last step was to divide the single message contribution by the test duration to hour ratio, which shows how many ampere hours a single transmission consumes.

Ampere hours are however not a complete measurement of power consumption as they do not account for the voltage. For a more accurate representation of the power consumption, equation 5.2 was used to represent the cost in Watt hours instead.

\[
\text{Watt hours} = \text{Ampere hours} \times \text{Voltage} \tag{5.2}
\]
6. Experimental Results

This chapter presents the results that were achieved by doing the power measurements described in Section 5.5. Through out this chapter, Board #1 and Board #2 is used to refer to the two identical end-device prototypes used. The chapter is laid out as follows: Section 6.1 presents the results from measurements of the current during transmission of a 1 Byte payload and a 4 Bytes payload respectively, for two different spreading factors, SF7 and SF12. Section 6.2 presents results from measurements of the voltage for both boards for SF7 and SF12. In Section 6.3 the power consumption based on the current and voltage measurements is presented. Section 6.4 shows the behavior of the current for transmission, receive windows and idle stages. The last section, Section 6.5 applies the requirements specification, which was first presented in Section 5.1, to the end-device prototype solution. The full set of results, including raw data can be found in [59].

6.1 Current measurements

The results from the current measurements for board #1 and #2 are summarized in table 6.1 through 6.3. The three tables are structured as follows: The row ”Average, idle” shows an average value for the current in milliampere (mA) when the boards were not transmitting and did not have open receive windows. The average value presented is an average of 10 iterations of measurements. For each iteration of measurements, the average cost for one transmission (sending a message and the two receive windows), was calculated as described in equation 5.1, in Section 5.5. The row ”Average, 1 send” shows the average cost of a single transmission in milliampere hours (mAh). In ”Max 1 send” the highest average cost of a single transmission among the 10 iterations is presented in milliampere hours (mAh). In ”Min 1 send” the lowest average cost of a single transmission among the 10 iteration is presented in milliampere hours (mAh).

In table 6.1 data from measurements for spreading factor 7 is presented. For these tests board #2 was observed to have an abnormally low average idle consumption and the reason for this abnormality remains unknown. However, the idle values for all tests on board #2 were consistent, meaning that calculations based on that measurement data are still valid.

When looking at the ”Average, 1 send” data, despite the high variation between the boards while idle, there is a clear trend pointing at 4 byte transmissions having a higher average power consumption than 1 byte transmissions, which was expected. For spreading factor 7, the difference seems to be 3-5%. In reality, the messages are 14 and 17 Bytes respectively, as an overhead of 13 bytes is always added to a LoRaWAN message. When considering that the preamble and the two receive windows do not change, a difference of 3-5% seems like a reasonable re-
Table 6.1: The results of the current measurements where the size of the payload was 1 or 4 Bytes, and SF7 was used. The boards were connected to the power through a micro-USB cable.

In table 6.1, data for spreading factor 12 is presented. When looking at table 6.2 the "Average, idle" values are much more consistent between the boards than in table 6.1. The variation of the "Average, 1 send" between the two boards are less than 2% which allows for a rather accurate average cost for one transmission to be given. In this case the average cost of one transmission of 1 byte using spreading factor 12 can be determined to be between 0.0102mAh and 0.0104mAh. Increasing the payload size from 1 byte to 4 byte can be seen to result in an increased transmission cost by about 12% for board #1 and about 13% for board #2. The fact that the increased cost between a payload of 1 byte and 4 byte is much larger when using spreading factor 12 than spreading factor 7, can be explained by the fact that the fixed costs of a transmission makes for a smaller part of the total cost when using SF12 than when using SF7.

Table 6.2: The results of the current measurements where the size of the payload was 1 or 4 Bytes, and SF12 was used. The boards were connected to the power through a micro-USB cable.

Table 6.1 and 6.2 present the results of measurements performed when the boards were powered through the micro-USB connection. The results representing the measurements where the boards were powered directly are presented in table 6.3. When powering the board directly through the \( V_{\text{IN}} \) and GND pins, the "average, idle" consumption is a little over 10 mA lower than when powering the board through micro-USB. This rather significant difference in power consumption is thought to be due to the on board USB connector circuits going in to sleep mode when not used. When comparing the results in the "Average, 1 send" row of table 6.3 to the same row in table 6.1 the results seem to suggest that the trans-
mission cost of a single message would be slightly higher when using direct power. Why this would be the case is yet unknown. One possible theory is that the USB connector circuits smooths out the consumption somewhat, resulting in slightly lower peaks when using micro-USB to power the board. The slightly higher "Average, 1 send" values aside, the difference between a 1 byte transmission and 4 byte transmission is still about 5% when powered through direct power.

### 6.2 Voltage measurements

The results from the voltage measurements for board #1 and board #2 are summarized in two tables below. The two tables are structured as follows: The row "Average, idle" shows an average value for the voltage in Volt (V) when the boards were not transmitting and did not have open receive windows. The average value presented is an average of 10 iterations of measurements. The row "Average send" shows the average voltage in Volt (V) during transmission or when the receive windows were open. For each iteration of measurements, the average voltage during transmission and two receive windows was calculated. Table 6.4 presents the results for spreading factor 7, while table 6.5 shows the results for spreading factor 12.

<table>
<thead>
<tr>
<th>SF7 - 1 byte - Direct power</th>
<th>SF7 - 4 byte - Direct power</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Board #1</strong></td>
<td><strong>Board #2</strong></td>
</tr>
<tr>
<td>Average, idle (mA)</td>
<td>27.88934347</td>
</tr>
<tr>
<td>Average, 1 send (mAh)</td>
<td>0.0009128</td>
</tr>
<tr>
<td>Max 1 send (mAh)</td>
<td>0.0009750</td>
</tr>
<tr>
<td>Min 1 send (mAh)</td>
<td>0.0008579</td>
</tr>
</tbody>
</table>

Table 6.3: The results of the current measurements where the size of the payload was 1 or 4 Bytes, and SF7 was used. The board was powered directly through the V\textsubscript{IN} and GND pins.

<table>
<thead>
<tr>
<th>SF7 - 4 byte - USB power</th>
<th>SF7 - 4 byte - Direct power</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Board #1</strong></td>
<td><strong>Board #2</strong></td>
</tr>
<tr>
<td>Average, idle (V)</td>
<td>4.6600634</td>
</tr>
<tr>
<td>Average send (V)</td>
<td>4.6109744</td>
</tr>
</tbody>
</table>

Table 6.4: The results of the voltage measurements where the size of the payload was 4 Bytes and SF7 was used. Two different ways of connection to the power source were used.

Table 6.4 and table 6.5 show that the voltages for both boards were very similar to each other when being in the same state, either idle or transmitting/keeping receive windows open, and when the same type of connection to the power source was used. However, the voltage of the boards when being connected to the power source through a micro-USB cable was 0.1V lower than when the boards were connected to the power source in the more direct way described in Section 5.4. This difference did not seem to affect the behavior of LoRaWAN.
Table 6.5: The results of the voltage measurements where the size of the payload was 4 Bytes and SF12 was used. The boards were connected to the power through a micro-USB cable.

Changing the spreading factors did not produce any significant difference in voltage during the idle state. This was expected, as LoRaWAN transmission parameters should not affect the board when not sending or receiving. There was however a significant difference in voltage when different spreading factors were used to send a message. For spreading factor 7, the difference in voltage between the idle state and the state of transmission or keeping the receive windows open was on average 0.05-0.06 V. For spreading factor 12, the difference was on average 0.26 V.

6.3 Power consumption adjusted by voltage

To get a more complete picture of the power consumption of the end-device prototypes, the power consumption is here calculated in hours (mWh) using equation 5.2. To get the values presented, the "Average, idle (mA)" From Section 6.1 was multiplied with the "Average, idle (V)" from Section 6.2, and "Average, 1 send (mAh) from Section 6.1 was multiplied with the "Average send (V) from Section 6.2.

Table 6.6: Power consumption when SF7, SF12 respectively and 1 Byte payload were used. The board was powered through micro USB cable.

Table 6.7: Power consumption when SF7, SF12 respectively and 4 Byte payload were used. The board was powered through micro USB cable.
Table 6.8: Power consumption when the board was powered directly and SF7 was used with two different payloads.

6.4 Overall behavior of the current

Figure 6.1 shows the general behavior of the current during the SF7 tests. The pattern was the same for all tests using SF7. Figure 6.1 is based on 10 iterations of measurements with 1 byte payloads sent over SF7 on board #2. The peaks in the chart are where the end-device was transmitting and the following receive windows, confirming that this is a class A device.

Figure 6.1: Chart of 10 measurements of current for Board #2 while sending a payload of 1 Byte with SF7. The payload is sent every 30 seconds. The chart was made by the authors.

Figure 6.2 gives a closer look at one of the peaks seen in figure 6.1. Figure 6.2 clearly shows where the message is sent and the following two receive windows.

Interesting to note is that for small transmissions using SF7, the total cost of the receive windows seems to outweigh the cost of the sending the message. The cost of the receive windows is the same no matter what spreading factor or payload size is used, which means that the receive windows have much less impact on the overall cost of transmitting when using higher spreading factors. This can be confirmed by figure 6.3, where a transmission of 1 byte using SF12 is shown.

While the cost of the receive windows for SF12 is clearly a much smaller part of the total cost of transmission, the cost is still significant. The end-device remains occupied from the start of the transmission to the end of the second receive window, this should be taken into account when modeling the power consumption of a prototype.
6.5 Applying the requirements specification

In this section, the requirements specification presented in Section 5.1 is applied on the end-device prototype solution for an overall evaluation. The requirements specification can be found in table 6.9 below. The four areas the specification covers are energy efficiency, transmission time, reliability and vulnerability.

Starting with the energy efficiency of the end-device prototypes made by the authors, it quickly becomes clear that the end-device prototypes made by the authors do not meet the requirements set for energy efficiency. Even using the most power efficient sleep mode puts the device at an average consumption of almost four times the acceptable level. It’s evident that the "The Things Uno" board used does not have the energy efficiency required.

Looking at the transmission time, the end-device prototypes made by the authors receives either an "acceptable" or "preferable" grade depending on the spreading factor used. By observing figure 6.2 which shows a transmission using spreading factor 7, the relevant sending part of the transmission can be seen to be less than half a second, putting the device at a "preferable" grade. When looking at fig-
Table 6.9: The requirements specification which was compiled by the authors and was presented in Section 5.1.

<table>
<thead>
<tr>
<th>Area/Grade</th>
<th>Not acceptable</th>
<th>Acceptable</th>
<th>Preferable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy efficiency</td>
<td>$e &gt; 2.3,\text{mA consumption}$</td>
<td>$e &lt; 2.3,\text{mA consumption}$</td>
<td>$e &lt; 0.58,\text{mA consumption}$</td>
</tr>
<tr>
<td>Transmission time</td>
<td>$t &gt; 5,\text{seconds}$</td>
<td>$t &lt; 5,\text{seconds}$</td>
<td>$t &lt; 1,\text{second}$</td>
</tr>
<tr>
<td>Reliability</td>
<td>$R &gt; 1%$ to miss alarm</td>
<td>$R &lt; 1%$ to miss alarm</td>
<td>$R &lt; 0.1%$ to miss alarm</td>
</tr>
<tr>
<td>Vulnerability</td>
<td>No grade reached</td>
<td>Grade 1 reached</td>
<td>Grade 2 or higher reached</td>
</tr>
</tbody>
</table>

Figure 6.3 which shows a transmission using spreading factor 12. The sending part of the transmission takes almost two seconds. While two seconds are not good enough for a "preferred" grade, two seconds are well within the requirements for an "acceptable" grade.

When trying to apply the specification for reliability, the conclusion was reached that determining the reliability of a wireless technology such as LoRaWAN is far too complex of a question to be answered with the current requirements specification. Since the rate of packet loss for a wireless technology is dependent on a large number of factors, such as distance, interference and number of gateways in range to mention but a few. With that said, based on the research of others and the undocumented observations of the authors, the end-device prototype solution in its current setup would likely receive a "Not acceptable" grade in most environments.

Finally when applying the requirements specification for vulnerabilities, the prerequisites required by an intruder to be able to bypass the system is considered. Assuming that an end-device prototype is placed in a position with a good view of the surrounding area (such as in a corner facing out), an intruder would require specialized masking equipment to avoid detection by the PIR motion sensor. Due to the battery powered nature of the end-device prototypes, disabling an end-device prototype would require direct physical access to the device. Since direct physical access to the end-device prototype would not be possible without the previously mentioned masking equipment, an intruder would have to target the LoRaWAN side of the device to prevent alarms getting through. As explained in Section 2.7, attacking the LoRaWAN transmission requires time, specialized equipment and knowledge of the system.

Having established what would be required to bypass the end-device prototype, those requirements can be compared to the gradings used by the European standard for intrusion and alarm systems presented in Section 3.1. The key factors required are specialized equipment and knowledge of the system. Looking at the European gradings, grade 3 is described as being able to withstand "The trespasser or burglar are expected to be very familiar with alarm systems and have an extended range of tools and portable electronic instruments." which seems to describe the equipment and knowledge required. Thus the end-device prototypes fail to achieve grade 3 but grade 2 which is described as being able to withstand "The trespasser or burglar are expected to have limited knowledge about alarm systems and have accesses to a general range of tools and portable instruments,"
such as multi-meter.” is clearly achieved. According to the requirements specification established in this thesis, a European grade 2 corresponds to a “preferable” grade in this thesis.
7. Discussion

When the requirements specification was applied to the end-device prototype solution developed by the authors in Section 6.5, it was evident that the prototypes did not meet all the requirements. This chapter discusses solutions on how all requirements could be met in a case where development time and funds are less restricted. Section 7.1 presents how the reliability of the device could be improved. Section 7.2 discusses a scenario where a more power efficient board is used, along with a sleep function.

7.1 Reliability

In Section 6.5 it is mentioned that the end-device prototype solution in its current state would likely receive a "not acceptable" grade in most environments. Here the issue is explored more in depth and some potential ways to solve the issue are discussed. In other research done on packet loss using LoRaWAN it does become clear that a rate of packet loss of 10-30% is likely in areas with low gateway density. 10-30% chance of an alarm not arriving is of course not acceptable. There are several ways to mitigate the packet loss which all have their own downsides. One way of mitigating the packet loss is to use acknowledgement of transmissions and re-send any messages not acknowledged. The current problem with acknowledgements lies in the downlink limitations put in place by networks owners such as The Things Network. For The Things Network specifically the limit of 10 downlink transmissions per day heavily impairs the use of acknowledgement of transmissions. Using acknowledgements for high priority transmissions only, such as alarms could be a viable approach. But any false triggers such as the owner of the device not turning off the device when returning would quickly consume the 10 allotted downlinks for that day. As this downlink limitation is in place due to the half-duplex properties of LoRaWAN gateways, the downlink limitation is unlikely to be removed until full-duplex for LoRaWAN gateways can be achieved.

Another way of mitigating a high rate of packet loss is to send each transmission multiple times. For low power devices however, any more transmissions than necessary are of course undesirable. Ignoring the low power aspect for a moment, sending each transmission multiple times can become problematic in situations where high spreading factors are required. This is due to the duty cycle limitations imposed both by the LoRaWAN specification and more strictly so by network providers such as The Things Network. To comply with the duty cycle limitation of The Things Network which is maximum on-air time of 30 seconds, a device using SF12 to send a minimal payload could send a little over 30 transmissions per day, which is especially problematic if you want to repeat each transmission a number of times.
In both the suggested ways of mitigation, the limitations put in place by the network provider are central issues. One way of circumventing those limitations is to create a private LoRaWAN. The downside of using a private Network compared to a public one is that the coverage of the network must be increased by the owner of the network. Public networks on the other hand grow on their own as any one is allowed to connect their gateway to the network. Provided that they follow the rules of the owner of the public network. With these pros and cons of public and private networks in mind, a private network is really only a good solution if the area of operation is limited and defined, or if the supplier of the device is willing to put considerable recourses in to the network.

### 7.2 Power consumption

As seen in the **Chapter 6 Results**, the end-device prototypes constructed by the authors are nowhere near an acceptable level of power consumption. This is however due to the The Things UNO boards base power consumption being far too high. Figure 7.1 isolates the costs that are created by LoRaWAN during transmission and receive windows. The LoRaWAN messaging on its own could be sustained for hundreds or even thousands of years depending on behavior. However, figure 7.1 does not include any base cost for powering the device sending the messages, but simply the cost of the messages themselves.

![Figure 7.1: This chart shows for how long LoRaWAN messages could be sent depending on behavior and spreading factor. Chart is made by the authors.](image_url)

Here we will hypothesize about a more viable device. This can be achieved by using the power consumption in the home-automation-community article that is briefly described in **Section 3.4**. In that article, an Arduino Pro Mini is reduced to a power consumption of 3.9 mA in an active state and 0.054 mA in sleep mode, by removing the power LED. For the remainder of this section “the device” will refer to the hypothetical device. In order to make the device more power efficient, the device is put in to sleep mode whenever there is nothing to do. The sleep mode is interrupted either by the motion detector triggering or a watchdog timer timing out.
In figure 7.2 the life time of the device on a single 10000 mAh battery is displayed for a number of different device behaviors, and for both SF7 and SF12. When calculating these estimations, the cost of a single message is based on the results presented in Section 6.1. For the estimated heartbeats, alerts and logs, the device is assumed to be in active mode for 3 seconds for SF7. For SF12 the device is assumed to be in active mode for 4 seconds for Heartbeats and alerts, and for 5 seconds for logs. The logging procedure the device is assumed to keep the device in active mode for the full duration of 3 minutes. These estimated active times are based on the charts presented in Section 6.1 and also additional charts available in the result spreadsheet [59]. During the active mode times, the idle current is applied to the calculation. Looking at figure 7.2 it can be seen that depending on behavior, the device would last somewhere between 2.4 and 21 years. This result would receive a “preferred” grade in this thesis even in the rightmost case of figure 7.2 which shows a behavior far more intense than what can be expected.

![Chart illustrating how long a device would last on a 10000mA battery, if sleep function was implemented when not transmitting or receiving. Chart is made by the authors.](image)

**Figure 7.2:** Chart illustrating how long a device would last on a 10000mA battery, if sleep function was implemented when not transmitting or receiving. Chart is made by the authors.

When it comes to being power efficient, LoRaWAN certainly lives up to its **Low Power** Wide Area Network category. There is however still some room for improvement. When observing figure 6.2 and 6.3 in Chapter 6 it is clear to see that for small payloads, the amount of time the device spends awake is prolonged heavily by the two receive windows. For devices designed to never receive any downlink messages a device class with no receive windows could be a great addition, and further improve energy efficiency.
8. Conclusions and future work

This chapter presents the conclusions of the authors on how well suited LoRaWAN is for battery powered security solutions (Section 8.1) and provides suggestions for future work on this topic (Section 8.2).

8.1 Conclusions

The purpose of this thesis was to answer the question "How well do the properties of LoRaWAN meet the requirements for a battery powered security solution?". A literature study and interviews were conducted to create a requirements specification for a simple battery powered security system. Two identical end-device prototypes were created to perform tests on power consumption. The end-devices were then compared to the requirements.

The end-device prototypes did not reach an acceptable level for the requirements when it came to energy efficiency and reliability. The board used for the end-device prototypes had a very high idle power consumption and no sleep function was implemented. Estimations based on the results show that with the use of a more power efficient board and software, there would be no problem to achieve not only acceptable, but even preferable grade of power efficiency. Reliability was much harder to estimate. The strict limitations of downlink messages encourages the use of non-acknowledged messages. Without acknowledged messages, the reliability of LoRaWAN becomes completely reliant on the rate of packet loss. As packet loss always is possible when dealing with wireless technologies, LoRaWAN is inherently unreliable when used without acknowledged messages. Section 7.1 presents a few ways of mitigating reliability issues but the area still requires further research.

When it comes to security attributes, the battery powered properties of the end-device, and the use of LoRaWAN for communication both proved to be great strengths. To turn off a battery powered security solution while avoiding to trigger the motion detector requires extensive prior knowledge about the device, the devices placement and the motion detection technology. When it comes to LoRaWAN, unless there is no active gateway within range or the device itself is disabled before the motion sensor is triggered, very little tampering can be done. While the end-device is not intended for use in areas with high-risk of intrusion or burglary, the end-device could potentially be used in places that require Grade 1 or Grade 2 security systems, but where a full-scale permanent solution is not a viable option. However, there needs to be more research done on reliability of such end-device to ensure that potentially poor reliability does not become a vulnerability.
In conclusion, if the reliability aspects of LoRaWAN itself can be improved or if the end-device can be implemented in such a way that the device proves to be sufficiently reliable, then LoRaWAN as a technology would be well suited for remote battery powered security solutions.

8.2 Future work

The tests done in this thesis were all done on the same setup. In order to get a more complete picture of the energy efficiency of LoRaWAN, the tests need to be repeated on a wider array of devices. By testing a wider array of devices, the power cost that can be attributed to LoRaWAN can be more accurately isolated. In this thesis the power cost of LoRaWAN was isolated from the cost of powering the microcontroller board, and used to calculate an estimated power consumption of a more power efficient board. This estimation is done with the assumption that the power consumption of LoRaWAN would be the same if used on a more power efficient board. However, this setup needs to be implemented and tested in practice in order to be confirmed.

As this thesis has mentioned, the duty cycle limitations imposed by the LoRaWAN specification, and by some public network providers makes it difficult to achieve a high level of reliability. LoRaWAN gateways with full-duplex capabilities would go a long way to improve the reliability of LoRaWAN. Full-duplex gateways would allow for more liberal use of acknowledgement, which would more or less solve the reliability issues.

While this thesis quite clearly points at the isolated cost of LoRaWAN messages being very small, there is still some improvement that could be done. As with the end-device prototypes presented in this thesis, there are many situations where devices are not designed to receive any messages outside of the join procedure. In such situations, the receive windows needlessly keep the device active when it could be in sleep mode. For a small payload sent using a low spreading factor, the device is kept active for over four times longer then needed to send the message. This means that while the isolated cost of the receive windows might be small, the device is forced to run at its idle consumption rate instead of its sleep consumption rate for a longer time. With all of this in mind, the implementation of a "send only" device class which excludes any receive windows is desirable. Such a device class would further push the limits of LoRaWANs energy efficiency.
A. Source code

A.1 Main source-code for the end-device

/* WANPIR – A remote battery powered motion detector solution using LoRaWAN

Heartbeats are sent every X minutes, unless an alarm is triggered. (X=hbInterval is given in milliseconds)

Heartbeat message is 1 byte.
Then the heartbeat timer is restarted.

Alert is triggered when the PIR–sensor is triggered.
The alert message (1 byte) should always be followed by a log message.

The log message (32 bits) contains historical information of 3 minutes from the first alert.
The information is when the sensor was triggered for each 6s interval in the last 3 minutes.
The leftmost bit of the log will always be 0.

The appKey and appEui needed for the boards have been changed to XXX in for privacy reasons.
When using the code, please substitute with correct parameters for your equipment.

Developed as a part of a Bachelor Thesis project for KTH, Sweden and in collaboration with Slagkryssaren AB

Authors: Patrik Karlstén and Irina Galiulina

v 0.02 May 2018 */

#include <TheThingsNetwork.h>

const char *appEui = "XXX"; // Application specific parameter
const char *appKey = "XXX"; // Device specific parameter
const unsigned long hbInterval = 300000; // 5 min heartbeat timer in miliseconds
const unsigned long logTime = 180000; // log mode interval in miliseconds
byte shortPayload[1]; // one byte payload array used for heartbeat and alarm
byte logPayload[4]; // four byte payload array used for eventLog
volatile bool alarmTriggered = false; // sensor trigger flag, set to true by sensor
unsigned long lastHeartBeatTimer; // timer for the heartbeats
int sensorPin = 2; // pin number of the PIR sensor input

#define loraSerial Serial1
#define debugSerial Serial
#define freqPlan TTN_FP_EU868 // European frequency standard

TheThingsNetwork ttn(loraSerial, debugSerial, freqPlan);
void setup()
{
    loraSerial.begin(57600);
    debugSerial.begin(9600);
    pinMode(sensorPin, INPUT);
    // define the sensor pin as input
    while (!debugSerial && millis() < 10000);                // wait a maximum of 10s for Serial Monitor
    debugSerial.println("Join TTN");
    ttn.join(appEui, appKey);
    // join the ttn LoRaWAN network using OTAA
    lastHeartBeatTimer = millis();                        // reset heartbeat timer
    // warmup timer; avoids instant sensor triggers
    delay(30000);
}

void loop() { // the main body of the program
    heartbeat();  // check if it’s time for heartbeat
    alarmCheck(); // check if the sensor has been triggered
}

void heartbeat()
{
    // if there has been more than hbInterval milliseconds since last message sent
    // send a heartbeat message
    if ((millis() - lastHeartBeatTimer) >= hbInterval)
    {
        shortPayload[0] = 0;                           // setting heartbeat payload
        ttn.sendBytes(shortPayload, sizeof(shortPayload)); // send heartbeat
        lastHeartBeatTimer = millis();                  // reset heartbeat timer
    }
}

void alarmCheck()
{
    getSensorState();
    // if the alarmTriggered flag is set, send an alarm message
    if (alarmTriggered)
    {
        shortPayload[0] = 1;                           // setting alarm payload
        ttn.sendBytes(shortPayload, sizeof(shortPayload)); // send alarm
        lastHeartBeatTimer = millis();                  // reset heartbeat timer
        alarmTriggered = false;                        // reset the alarm trigger
        logMode();                                     // enter log mode
    }
}

void logMode()
{
    // start the log with a ‘1’ since the log mode was started by an alarm
    unsigned long eventLog = 1;

    // store the start time of the log mode session
    unsigned long firstTriggerTime = millis();

    // OBS! (logTime/logSegment) must be <= 31 to prevent overstoring the event log
    unsigned long logSegment = 6000;
    while (millis() - firstTriggerTime <= logTime)       // log mode loop
    {
        if (!alarmTriggered)
        {
            getSensorState();    // check sensor state if not already known
        }
    }
}
```c
if (millis() - firstTriggerTime >= logSegment) // check for sensor triggers
{
    if (alarmTriggered) // alarmTriggered=true,
    {
        eventLog = (eventLog<<1) + 1;
        logSegment = logSegment + 6000;
        alarmTriggered = false;
    }
    else // if the alarmTriggered=false
    {
        eventLog = (eventLog<<1) + 0;
        logSegment = logSegment + 6000;
    }
}

// encoding the event log in to byte size segments for sending with LoRa
logPayload[0] = (byte) ((eventLog & 0xFF000000) >> 24);
logPayload[1] = (byte) ((eventLog & 0x00FF0000) >> 16);
logPayload[2] = (byte) ((eventLog & 0x0000FF00) >> 8);
logPayload[3] = (byte) ((eventLog & 0x000000FF));
ttn.sendBytes(logPayload, sizeof(logPayload)); // send event log
lastHeartBeatTimer = millis(); // reset heartbeat timer

// sets the alarmTriggered variable to true if the sensor is triggered
// sets to the variable false otherwise
void getSensorState()
{
    if (digitalRead(sensorPin) == HIGH){
        alarmTriggered = true;
    }else
    {
        alarmTriggered = false;
    }
}
```
A.2 Source-code for testing 1 Byte payloads

```
#include <TheThingsNetwork.h>

const char *appEui = "XXX"; // Application specific parameter
const char *appKey = "XXX"; // Device specific parameter

const unsigned long hbInterval = 300000; //timer, in miliseconds, currently 5 min

byte payload[1]; //one byte payload array
unsigned long lastHeartBeatTimer; //used to track when the last message was sent
//to determine when to send the next one

#define loraSerial Serial1
#define debugSerial Serial
#define freqPlan TTN_FP_EU868 // European frequency standard

TheThingsNetwork ttn(loraSerial, debugSerial, freqPlan);

void setup()
{
    loraSerial.begin(57600);
    debugSerial.begin(9600);

    while (!debugSerial && millis() < 10000); //wait a maximum of 10s for Serial Monitor
    debugSerial.println("Joining TTN");
    ttn.join(appEui, appKey); //join the ttn LoRaWAN network using OTAA
    lastHeartBeatTimer = millis(); //reset the timer
    delay(30000); //warmup timer

    debugSerial.println("Entering the loop-phase");
}

void loop()
{
    heartbeat(); //check if it’s time for a message
}

void heartbeat() //heartbeat function
{
    //if there has been more than hbInterval miliseconds since last message sent
    //send a message
    if ((millis() - lastHeartBeatTimer) >= hbInterval)
    {
        payload[0] = 0; //setting heartbeat payload
        lastHeartBeatTimer = millis(); //reset heartbeat timer
        ttn.sendBytes(payload, sizeof(payload)); //send regular payload
        //Below: send payload with SF12, no ACK and use port 1
        //ttn.sendBytes(payload, sizeof(payload), 1, false, 12);  
    }
}
```
A.3 Application source code

A.3.1 Decoder

// Decode an uplink message from a byte--array
function Decoder(bytes, port) {
    var decoded = {};
    decoded.size = bytes.length;  // length of the message received

    if (port === 1 && bytes.length === 1) {
        // (length == 1 byte -> alert OR heartbeat)
        decoded.message = bytes[0];
    } else if (port === 1 && bytes.length === 4) {
        // (length == 4 byte -> log)
        decoded.message = ((bytes[0]) << 24) + ((bytes[1]) << 16) + ((bytes[2]) << 8) + ((bytes[3]));
    }
    return decoded;
}

A.3.2 Converter

// Convert the byte--array message into human--readable information
function Converter(decoded, port) {
    var converted = decoded;

    if (decoded.size === 1) {
        // If the message is 1 byte -> check if heartbeat or alert
        if (port === 1 && converted.message === 0) {
            // message = 0 is a heartbeat
            converted.heartbeat = true;
            converted.alert = false;
            converted.log = false;
        } else if (port === 1 && converted.message === 1) {
            // message = 1 is a alert
            converted.heartbeat = false;
            converted.alert = true;
            converted.log = false;
        }
    } else if (port === 1 && decoded.size === 4) {
        // if the message size is 4 bytes, it is the log
        converted.heartbeat = false;
        converted.alert = true;
        converted.log = true;
    }
    return converted;
}

A.3.3 Validator

// Check that the message received is valid
function Validator(converted, port) {
    // return false if the message length is NOT 1 or 4 bytes long
    if (((converted.size !== 1) && (converted.size !== 4))) {
        return false;
    }
    return true;
}
A.4 Source code for reading the measurement from Fluke 45 multimeter

```python
import serial
import sys
import time
import statistics

def main():
    print("Starting the test function")

    # setup of the serial interface
    usb = serial.Serial(
        port='/dev/ttyUSB0',
        baudrate=9600,
        parity=serial.PARITY_NONE,
        stopbits=serial.STOPBITS_ONE,
        bytesize=serial.EIGHTBITS,
        timeout = 1
    )

    log = []
    print(usb.name)
    x = 0
    testNr = 1
    runtime = 6*60*5

    while (testNr <= 10):
        while (x < runtime):
            x += 1
            read = usb.readline()
            time.sleep(0.2)
            log.append(read)
            print(runtime - x)

        fileName = "log" + str(testNr) + ".txt"

        f = open(fileName,"w")
        print("file opened")
        print(testNr)
        f.writelines(log)
        print("writing done")
        f.close()
        testNr += 1
        x = 0
        log = []

    f.close()

if __name__ == "__main__":
    main()
```

Bibliography


[26] Limitations: data rate, packet size, 30 seconds uplink and 10 messages downlink per day fair access policy. [Online]. Available: https://www.thethingsnetwork.org/forum/t/
limitations-data-rate-packet-size-30-seconds-uplink-and-10-messages-downlink-per-day

[27] I. T. L. Computer Security Division. AES development - cryptographi

[28] Xueying Yang, “LoRaWan: Vulnerability analysis and practical exploita

[29] K. Feichtinger, Y. Nakano, K. Fukushima, and S. Kiyomoto, “Enhancing the
security of over-the-air-activation of LoRaWAN using a hybrid cryptosys
tem,” p. 9.

for replay attack using join request messages in LoRaWAN,” in 2017 Inter

[31] “Alarm systems - intrusion and hold-up systems - Part 1: System require
ments (SS-EN-50131-1 + A1 + A2).”

tectors – passive infrared detectors (SS-EN-50131-2-2).”

CRC Press, google-Books-ID: VXQdq0B3tnUC.

[34] P. Zappi, E. Farella, and L. Benini, “Tracking motion direction and distance
with pyroelectric IR sensors,” vol. 10, no. 9, pp. 1486–1494.

infrared motion sensors in wireless sensor network,” in 2008 Inter

[36] Q. Zhang and M. Karunanithi, “Feasibility of unobstrusive ambient sensors
for fall detections in home environment,” in 2016 38th Annual Inter
national Conference of the IEEE Engineering in Medicine and Biology Society
(EMBC), pp. 566–569.

paired,” in Digest of Papers. Second International Symposium on Wearable

location-aware system for the smart home,” vol. 52, no. 4, pp. 1311–1317.

[39] F. Erden, B. U. Toreyin, E. B. Soyer, I. Inac, O. Gunay, K. Kose, and
A. E. Cetin, “Wavelet based flickering flame detector using differential PIR


[59] Spreadsheet with results from the measurements performed for this project; Created by Patrik Karlstén and Irina Galilina. [Online]. Available: https://docs.google.com/spreadsheets/d/1QjmT6pOio1draO9ThR-UdKtC5vyLo8IfWosg8Xzo-A/edit?usp=sharing accessed on 2018-05-25.