# Firmware Development of a LoRaWAN Multi-Sensor Generic Node: an Industrial IoT Empirical Study

Master of Science in Technology Thesis
University of Turku
Department of Future Technologies
Embedded Systems Laboratory
2019
Ahmed Mohamed Elsalahy Mohamed

Supervisors:
PhD (Tech) Tomi Westerlund
PhD (Tech) Tuan Nguyen
MSc Jorge Peña Queralta
MSc Johan Stokking

UNIVERSITY OF TURKU
Department of Future Technologies

AHMED MOHAMED ELSALAHY MOHAMED : Firmware Development of a LoRaWAN Multi-Sensor Generic Node: an Industrial IoT Empirical Study

Master of Science in Technology Thesis, 74 p., 5 app. p.
Embedded Systems Laboratory
July 2019

---

Connectivity is the defining property of the Internet of Things (IoT). Multiple technologies and techniques allow embedded devices to transmit and receive data. The intersection between connectivity demands, physical and environmental limitations is what triggers the use of a specific technology. Low Power Wide Area (LPWA) wireless communication technologies such as LoRa and LoRaWAN are showing practicality and ease of the use in the field of IoT. LoRa modulation ability to provide devices with longer communication ranges make it an attractive choice in multiple IoT use cases. The energy efficiency and scalability aspects of LoRaWAN protocol trigger the research curiosity around the challenges and opportunities of using the technology.
In this study, we provide an extensive overview of the firmware development of an industrial LoRaWAN device. An empirical analysis of the device capabilities provides a deeper understanding of the technology potential and the possible areas of improvements. Energy-efficient firmware design practices are explored, analysed and implemented to provide a foundation for future developments in the field. Furthermore, we propose and evaluate a new LoRaWAN application design that explores over the air firmware configuration in runtime a novel micro update scheme. We devise a simple LoRaWAN energy estimation model and apply it to the proposed application. The same model is used to get an indication of LoRaWAN firmware updates over the air (FUOTA) practicality. The applied model highlighted useful energy optimisation techniques for an improved LoRaWAN firmware development.

Keywords: IoT, LoRa, LoRaWAN, Firmware, Design

# Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **SW** | Software |
| **HW** | Hardware |
| **IoT** | Internet of Things |
| **LPWA** | Low Power Wide Area |
| **LPWAN** | Low Power Wide Area Network |
| **UNB** | Ultra Narrow Band |
| **MAC** | Medium Access Control |
| **ISM** | Industrial, Scientific and Medical |
| **TTN** | The Things Network |
| **TTI** | The Things Industries |
| **GN** | Generic Node |
| **MCU** | Micro-Controller Unit |
| **UHF** | Ultra High Frequency |
| **LoRa** | Long Range Wireless Communication Protocol |
| **RF** | Radio Frequency |
| **LoRaWAN** | Long Range Wide Area Network |
| **FSK** | Frequency-shift keying |
| **SS** | Spread Spectrum |
| **CSS** | Chirp Spread Spectrum |

**SF** Spreading Factor

**ADR** Adaptive Data Rate

**AES** Advanced Encryption Standard

**ABP** Activation by Personalization

**OTAA** Over The Air Activation **MIC!** (**MIC!**)Message Integrity Check

**PTC** Peripheral Touch Controller

**PCB** Printed Circuit Board

**RSSI** Received Signal Strength Indicator

**SNR** Signal to Noise Ratio

**RTOS** Real Time Operating System

**CMSIS** Cortex M Software Interface Standard

**FUOTA** Firmware Update Over The Air

**GNMA** Genric Node Micro Application

# Chapter 1

# Introduction

In our current age, Internet of Things (IoT) is becoming more relevant than ever before. It is mesmerising how a few years ago, humans were struggling to establish a reliable method of communication between themselves, and now we are determined to connect the "Things" around us. IoT is changing and improving the way we live, whether we notice it or not. IoT involvement in agriculture, manufacturing, healthcare and many other areas is paving the road for a digital transformation that will redefine a new roadmap for human evolution.

With more than twenty IoT platforms [1] supporting all sorts of applications, it is clear how the technology research and development is starting to pay off. However, IoT technologies attempt to solve complex problems in multiple sectors, and the adoption of IoT can only be facilitated if the technology is offering a viable solution to an existing problem.

## 1.1 IoT and Low Power Wide Area Networks

Communication technologies allow for the creation of Internet-connected networks. Wirelessly connected devices allow the network to grow as it overcomes the challenges imposed by the need for wired infrastructure.

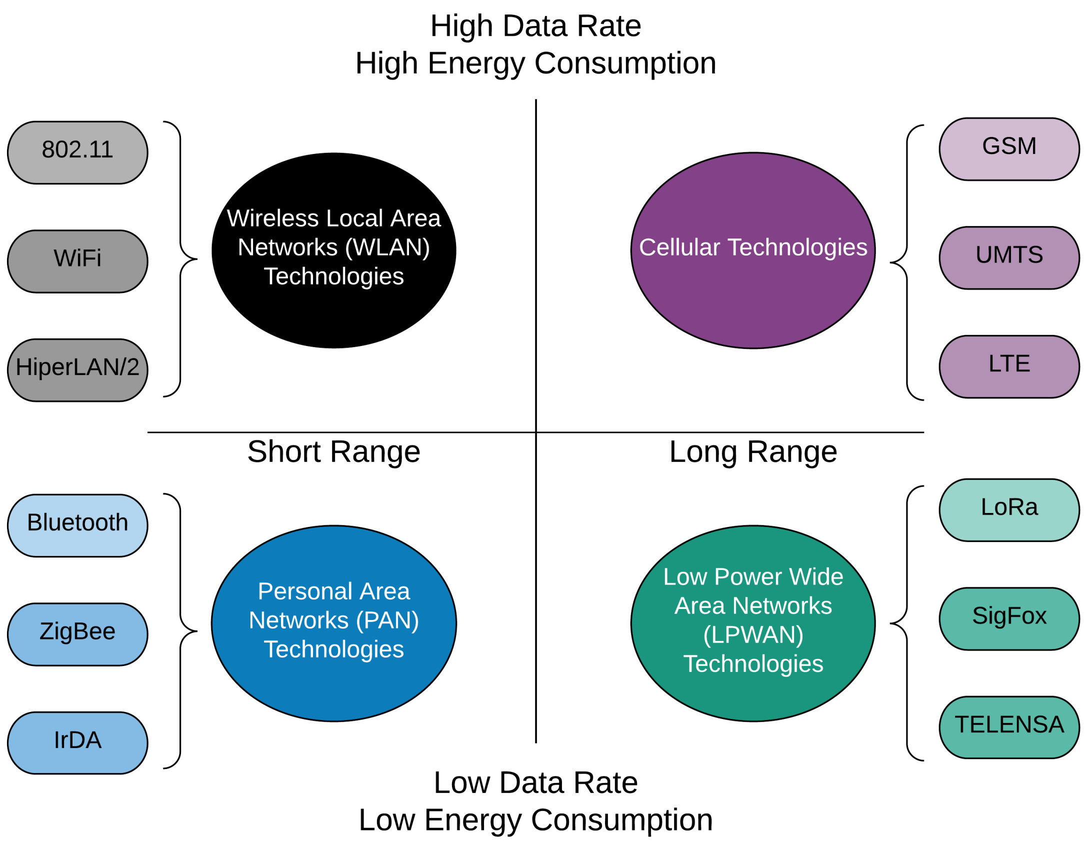


Figure 1.1: Wireless communication technologies.

Wireless communication networks are split into multiple categories, as depicted in Figure 1.1. The first category is short-range and high bandwidth with medium to high energy consumption like WiFi. Similarly, the second category shares a similar short-range with a focus on reducing data rates and energy consumption, and it is usually observed in Bluetooth. The third category noted in cellular networks such as GSM, UMTS and LTE. These networks are known for being longer in range with average to high bandwidth that is often limited by the energy consumption aspect. The fourth category is known as Low Power Wide Area (LPWA) networks, and it is found in proprietary technologies such as LoRa, SigFox, INGENU RPMA, TELENSA and QOWISIO [2]. LPWANs fill the gap by offering long-range, energy-efficient low data rate networks.

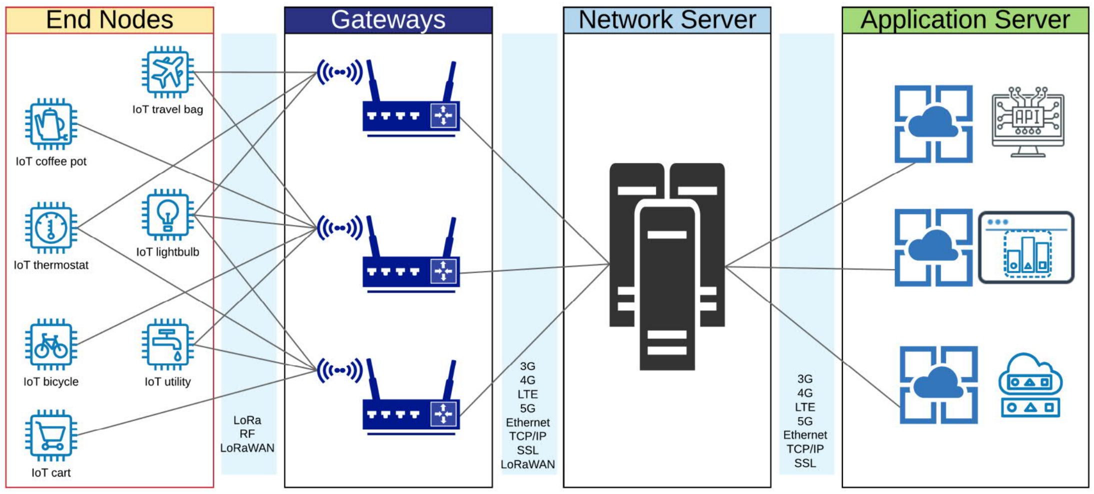


Figure 1.2: LoRaWAN network architecture.

Many IoT devices are either battery operated or relaying on an energy harvesting solution that allows them to operate for long periods [3]. LPWANs allow for inexpensive low power transceivers, which makes them ideal for many IoT applications such as smart cities, agriculture, industrial monitoring, logistics, home automation and many more [2].

## 1.2 LoRa and LoRaWAN

Long Range Wireless Communication Protocol (LoRa) is a term coined to describe the physical (radio) layer of an LPWA technology. Alternatively, Long Range Wide Area Network (LoRaWAN) is a communication protocol that builds upon the LoRa physical layer by specifying the Medium Access Control (MAC) and application layers. The LoRa alliance [1] introduced the protocol first in 2015 [4] to allow for usability, scalability and security of the connected devices while maintaining the LoRa physical modulation fundamental characteristics such as long-range, low bandwidth and extended battery life.

The network architecture depicted in Figure 1.2 highlights the four major components. End nodes describe the connected "things". They transmit and receive information to and

[1] www.lora-alliance.org/

from the gateways using LoRa Radio Frequency (RF), and in turn, they act as translators and communication bridges that allow the network and application servers to remain general-purpose entities that are modulation/topology independent.

## 1.3 The Things Network

The network architecture proposed by the LoRa alliance could be considered simple and easy to implement, but one of the main IoT properties is scalability, and here is where The Things Network (TTN) [2] comes into the picture. TTN is a global distributed crowd-funded [3] internet of things intuitive that provides an infrastructural service in a decentralised manner (Distributed Infrastructure Initiative [5]) building upon the LoRaWAN standard and providing a public, free and easy to use IoT network.
The idea behind a crowdsourced public network has many advantages and breaks the barriers of entry that exist with most IoT networks and platforms. The collaboration exists on multiple levels; mainly, it can be analysed based on the four major components of the network architecture. Volunteers connect their gateways to the network server and allow any end node within range to transmit and receive packets. TTN handles and maintains the gateway, network and application servers while providing an open-source network stack to allow for transparency and community collaboration.

TTN is pushing IoT and LoRaWAN forward in many aspects, and this can be easily observed in a number of initiatives and papers that use The Things public network to do research that helps define the technology promises and limitations.
Several research papers from multiple universities and various worldwide locations leverage the existence of the public network to research the use cases and improve upon the existing technology.
In [6], air quality monitoring system for smart cities, in [7], an object tracking solution,

[2]www.thethingsnetwork.org

[3]www.kickstarter.com/projects/419277966/the-things-network

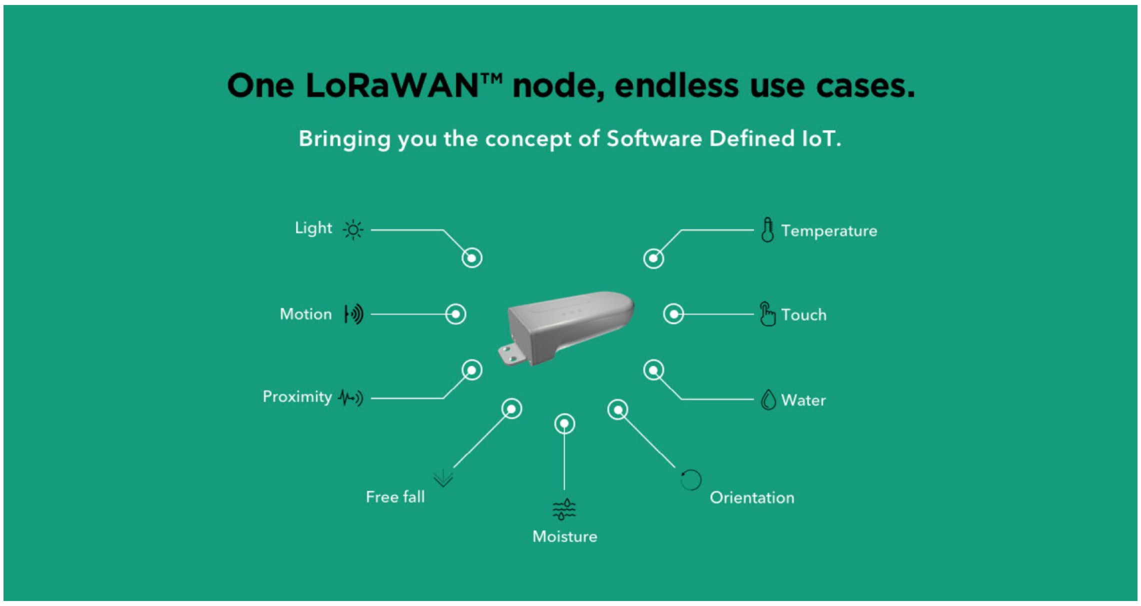

One LoRaWAN™ node, endless use cases.
Bringing you the concept of Software Defined IoT.
Light
Temperature
Motion
Touch
Proximity
Water
Free fall
Moisture
Orientation

Figure 1.3: Generic Node sensing abilities [15].

in [8], [9], e-health and agriculture systems are developed and evaluated. Improving on the technology is observed in [10] by reducing energy consumption using an artificial neural network, in [11], a LoRaWAN testbench, in [12], improved gateway server design, in [13], edge analytics evaluation and in [14] , the integration of LoRaWAN and 4G/5G for industrial IoT. The papers mentioned provide an insight into the usability and effectiveness aspects of LoRaWAN and how TTN is helping accelerate the development and integration of the new technology.
On the business side, The Things Industries (TTI) [4] helps businesses to build upon the usability of LoRaWAN and allows companies to integrate the functionalities into their core products.

## 1.4 The Generic Node (GN)

End nodes are the indispensable members of the architecture (Figure 1.2). All the other components are only useful if the end nodes are providing practical and advantageous

[4]www.thethingsindustries.com

use-cases.

In January of 2019 at the things conference [5] in Amsterdam, TTN/TTI unveiled a multi-sensor LoRaWAN end node called Generic Node (GN)[6]. It acts as a reference design for a next generation LoRaWAN end device. It is a low power design, contains a replaceable battery, supports firmware over the air updates, contains a secure element and it is designed to support ARM Mbed OS platform. It contains four versatile sensors which together drive around 20 to 30 use cases some of which are depicted in Figure 1.3.

Building upon the same community collaboration approach. The firmware of the device and all related software components are available as open-source repository on Github. Furthermore, the device schematics will be publicly available and can be commercially licensed.

Software-defined IoT is an effort to eliminate the Hardware (HW) and Software (SW) challenges by providing an easy to use development platform that allows developers to focus on building applications, and ultimately accelerating the adoption of the technology.

Due to the young nature of the standard and the usual long development and production times of end nodes, only a few end devices exist on the market that supports the standard. This creates an unwanted monopoly in a young and immature industry. Almost all commercially available end nodes are closed source, and they provide limited functionality with a high purchase cost.

TTN always had a history of supporting the LoRaWAN ecosystem by providing some of the essential components such as indoor and outdoor gateways. GN is part of the endeavour to eliminate the barrier to entry.

It is a welcomed addition to the market as it targets providing a multi-sensor, cost-effective, battery efficient and securely connected open source LoRaWAN end node to

[5] www.thethingsnetwork.org/conference/

[6] www.genericnode.com

Companies and individuals.

## 1.5 Research Focus and Goals

There is no doubt that the GN is a unique device, and the fact that it is an open-source project makes it even more valuable. Being a firmware engineer at TTN/TTI allowed for the unique opportunity to be part of a small team tasked with the design and implementation of an open-source GN firmware.

From an embedded engineer perspective, the device is considered the ultimate challenge. It offers a new design and a multitude of sensors. It can communicate over long-ranges and demands an efficient battery life. Furthermore, it utilises a young new standard (LoRaWAN) and builds upon it with the addition of a secure element and Firmware Update Over The Air (FUOTA).

### 1.5.1 Research Hypotheses

Due to LoRaWAN standard young nature and the competitive and secretive market built around IoT end devices, publicly available research regarding end nodes capabilities and possible future improvements is almost non-existent.

The research and development of GN establish multiple hypotheses that are verified, validated and extensively analysed throughout the development phase; some of them are:

- Multi-sensor hardware design complexity can have a negative effect on sensors measurements.
- IoT devices generic firmware design can allow for a multitude of use cases with minimal configuration overhead.
- General purpose multi-sensor IoT end devices can achieve low energy consumption and maintain long battery life.

- GN will enhance the security profile of LoRaWAN end devices with the inclusion of an onboard cryptographic co-processor.
- Efficient implementation of FUOTA can be achieved with minimal impact on battery life.

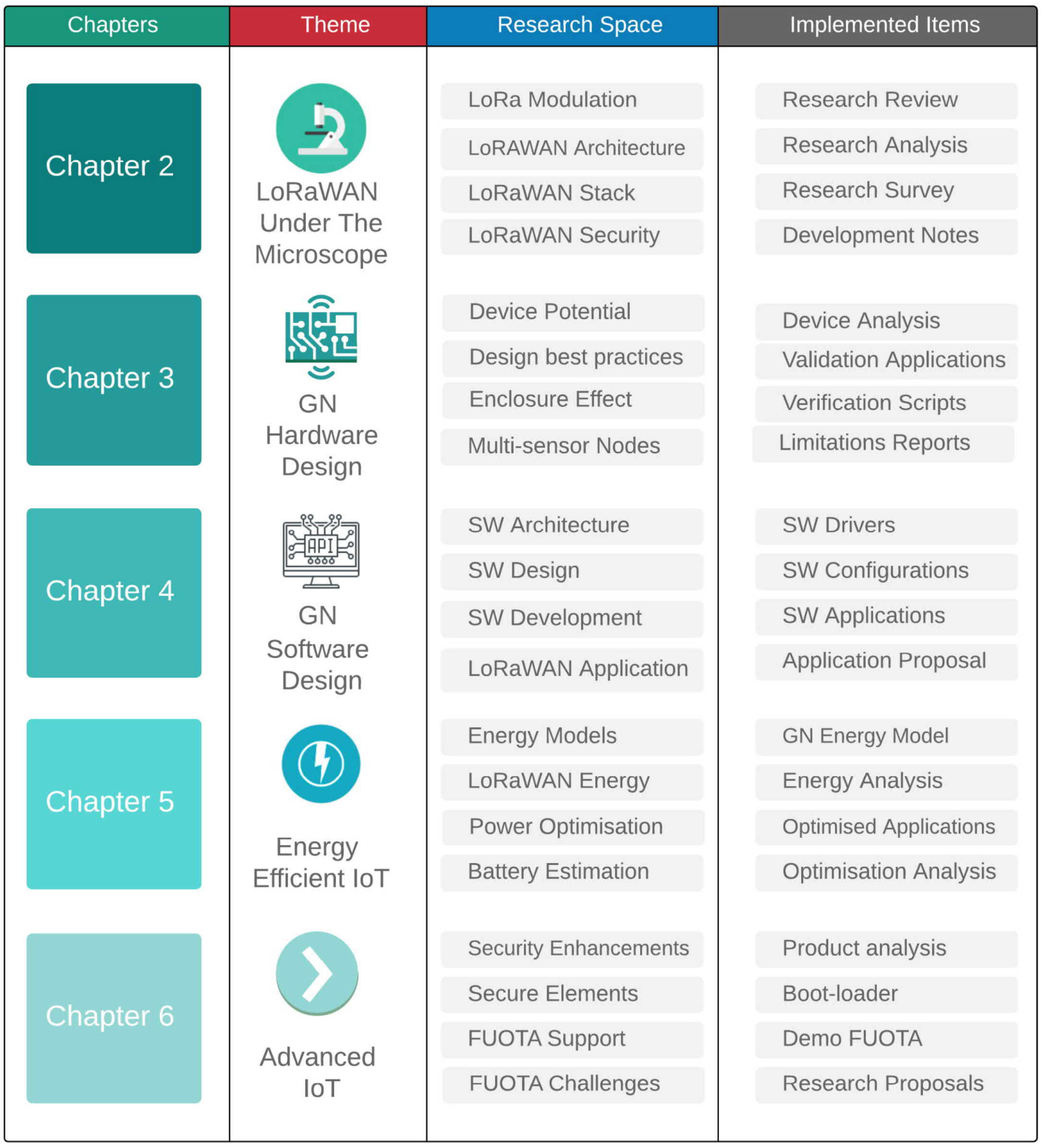
### 1.5.2 Research Methodology

Building a general-purpose firmware focuses mainly on abstracting the HW capabilities of the device and making them accessible through SW allowing developers and companies to use the device in various applications.

The information collected throughout the development process provides an insight into the state of the art of multiple technologies due to the reference design nature of the device. The challenges and opportunities of building the device are highlighted, analysed and recorded in an empirical approach. Related research is referenced, compared and applied to relevant challenges to provide an overall view of current issues in the field of IoT. Furthermore, the empirical findings are publicly shared in this document to provide an insight into the recent development problems allowing for further research.

### 1.5.3 Thesis Structure and Organisation

The development covers several areas and subjects depicted in Figure 1.4. Each chapter focuses on a specific research space. Implemented items refer to relevant reports, scripts, test cases and SW components that are developed in order to aid in the analysis of the corresponding research space.



| Chapters | Theme | Research Space | Implemented Items |
|---|---|---|---|
| Chapter 2 | LoRaWAN Under The Microscope | LoRa Modulation | Research Review |
| | | LoRAWAN Architecture | Research Analysis |
| | | LoRaWAN Stack | Research Survey |
| | | LoRaWAN Security | Development Notes |
| Chapter 3 | GN Hardware Design | Device Potential | Device Analysis |
| | | Design best practices | Validation Applications |
| | | Enclosure Effect | Verification Scripts |
| | | Multi-sensor Nodes | Limitations Reports |
| Chapter 4 | GN Software Design | SW Architecture | SW Drivers |
| | | SW Design | SW Configurations |
| | | SW Development | SW Applications |
| | | LoRaWAN Application | Application Proposal |
| Chapter 5 | Energy Efficient IoT | Energy Models | GN Energy Model |
| | | LoRaWAN Energy | Energy Analysis |
| | | Power Optimisation | Optimised Applications |
| | | Battery Estimation | Optimisation Analysis |
| Chapter 6 | Advanced IoT | Security Enhancements | Product analysis |
| | | Secure Elements | Boot-loader |
| | | FUOTA Support | Demo FUOTA |
| | | FUOTA Challenges | Research Proposals |

Figure 1.4: Chapters outline.

# Chapter 2

# LoRaWAN: Under the Microscope

In IoT, communication technologies impose functional and non-functional requirements on any network component. End nodes inherited connectivity capabilities are typically considered the defining characteristics of the device. To explain further, any device using Bluetooth is typically designed around power efficiency, limited range and average to high data rates.
In the case of GN, LoRa and LoRaWAN are not different, as any device using LPWA technology is expected to operate for several years on a single battery with a minimum communication range in kilometres [2], all of which is achievable by decreasing the data rates.

Understanding the communication technology requirements and specifications is critical to building a compliant device that delivers on the technology proposed vision. In the upcoming sections in this chapter, we take a brief overlook at research relevant technical terms and specifications that must be considered in the design and implementation of any LoRaWAN end node.

## 2.1 LoRa: Terms and Concepts

LoRa is a proprietary wireless modulation technology that acts as a physical layer for LoRaWAN specification.

### 2.1.1 LoRa Modulation

Low Power Wide Area Network (LPWAN) technologies are commonly divided into two communication techniques, Ultra Narrow Band (UNB) and Spread Spectrum (SS) [3]. LoRa physical layer uses frequency shift chirp modulation patented technique [16] developed by Cycleo (later acquired by Semtech).
LoRa is considered a SS technique where signals are modulated by chirp pulses (Chirp Spread Spectrum (CSS)) which is ultimately a frequency varying sinusoidal pulses [17] (Appendix A Figure A.1).

In [3], UNB and SS are reviewed and compared with respect to interference, capacity, link budget and coexistence. From an end node design and implementation perspectives, the analysis provides a valuable evaluation of what is expected if one of the modulation technologies is employed.
If we consider **interference**, in the case of LoRa, the utilization of CSS indicates a number of points:

- The modulation is resistant to external interferences and jamming signals (most interference signals lack the SS key, so most are rejected thus increasing the resistance property), such property improves the range and reliability aspects.
- In ideal situations, LoRa transmission range can reach up to 15 KM. Data rate is the deciding parameter, lower data rates deliver longer range and vice-versa.
- Dense environments populated with devices using the same modulation technique are considered an interference hazard due to self-noise, this means nearby nodes are

considered a source of interference for other nearby devices, such problem highlights the need for LoRaWAN from a scalability perspective [18].

**Capacity** refers to the amount of usable data (without any physical layer overheads or redundant noise offset data), and in the case of SS, it is significantly lower (theoretically five times lower) than UNB modulation techniques (challenged by network planning example in [19]). The capacity indicates the throughput of a modulation technique, and in the case of SS, it signifies the importance of designing an efficient communication protocol that takes into account the limited capacity. We can observe how the reduction of data is demanded from the interference, range and capacity aspects, and this becomes critically relevant when designing the application layer.

**Link budget** is a sophisticated equation that reflects the strength of a transmitted signal arriving at a receiver, the equation takes into account all the parameters such as antenna gain, path loss, propagation loss, receiver sensitivity, system dependant gains and modulation technique to be able to quantify the system communication capabilities.
The equation ultimately indicates the coverage of the communication technology, higher link budget typically means better coverage and since LPWAN is a constrained network (power and cost-wise), it is critical to accurately asses the link budget to arrive at an ideal network design that delivers on the promised functionalities, moreover the link budget is critical to any firmware design to help avoid wasting precious energy and in achieving reliable operation.

Using [19], a possible LoRa link budget could be 151 dB (calculation shown in Appendix A), and higher link budget can reach up to 164 dB as observed in [20], theoretically this would allow for a coverage of 800 KM (pure free space loss = 150 dB). A LoRaWAN world record of 766 KM was recorded [21], confirming similar link budget coverage calculations, but wireless reach in reality is affected by multiple phenomena's that fall under signal propagation theory [20], and further in depth exploration would be outside of the thesis scope.

**Coexistence** refers to the capability of using multiple LPWA networks in close proximity of each other. The analysis conducted between multiple LPWAN deployments scenarios shows increased interference problems that highlight the existence of a scalability problem.
Multiple solutions such as group splitting or power control coordination are recommended [3], but in case of GN design, the analysis highlights the need for end nodes to employ a reliable fallback mechanism that is able to detect transmission failures for long periods, and diagnostically report the issue when possible to allow network maintainers to adjust transmission rates and synchronise accordingly.

## 2.1.2 LoRa Spreading Factor

In digital modulation, some physical layer parameters are configurable and controlled at the application level such as **Baud Rate** in UART, and in the case of LoRa, the **Spreading Factor (SF)** acts similarly providing control of multiple LoRa transmission characteristics such as data rate and power consumption. To explain the relation, we define some related terms:

**Symbol** is an RF energy state that represents a cretin quantity of information (a symbol can represent a bit or more). A symbol rate or modulation rate is most commonly known as the baud rate, and it depicts the number of symbol changes over a period of time. From [19], we can evaluate the symbol rate using the following formula:

$$R_s = \frac{BW}{2^{SF}} \ symbols/sec$$

Where **Bandwidth (BW)** is considered the width of the spectrum occupied by a chirp. The **SF** represents the chirp signal spreading over the spectrum (the amount of frequency variation over the spectrum)[19], this ultimately depicts the number of bits encoded per symbol, or at a more granular level, $2^{SF}$ represents the number of chirps used to encode a bit [4], and in LoRa, SF parameter is configurable ranging from SF7 to SF12 (SF7 means

128 chirps per bit) We can observe that the data rate is inversely proportional to SF. SF12 corresponds to the lowest data rate, and the decreased data rate means more chirps per bit, allowing the signal to travel for longer distances (lower sensitivity and higher link budget) [19]. The spreading factor is also directly proportional to data transmission time, which is similarly proportional to energy consumption [22].

Understanding SF and its effect on energy consumption and range is critical in the development of any LoRa device. All loRa/LoRaWAN related research account for SF in their analysis and findings, moreover, some research is fundamentally built around the configurability aspect of the parameter as observed in [23], [24], [25], [26] and [27]. LoRa devices such as GN can control SF in runtime, and in devices that are promising long-range and prolonged battery life, the responsibility falls upon the firmware developer to properly configure SF to meet data rate, energy consumption and range requirements.

### 2.1.3 LoRa Operation

LoRa physical layer operates in the Industrial, Scientific and Medical (ISM) unlicensed frequency bands. The bands are region-specific, and their usage is regulated and supervised to ensure operability. Transmitting devices must comply with region-specific regulations such as max transmission power and duty cycle.
In Europe, LoRa devices are allowed to operate in EU 433/863-870 MHz ISM bands with a maximum transmission power of 14 dBm and a limited duty cycle up to 1% (every 1 second of transmission means the device cannot transmit for 99 seconds)[28].

The regional regulatory specifications act as functional requirements for LoRa capable devices, the implications of limited transmission power and duty cycle are significant from SW and HW perspectives and must be accounted for in multiple design/implementation areas. Devices such as GN are expected to operate in multiple regions, and SW configuration capabilities are demanded to allow for compliance in the various regions.

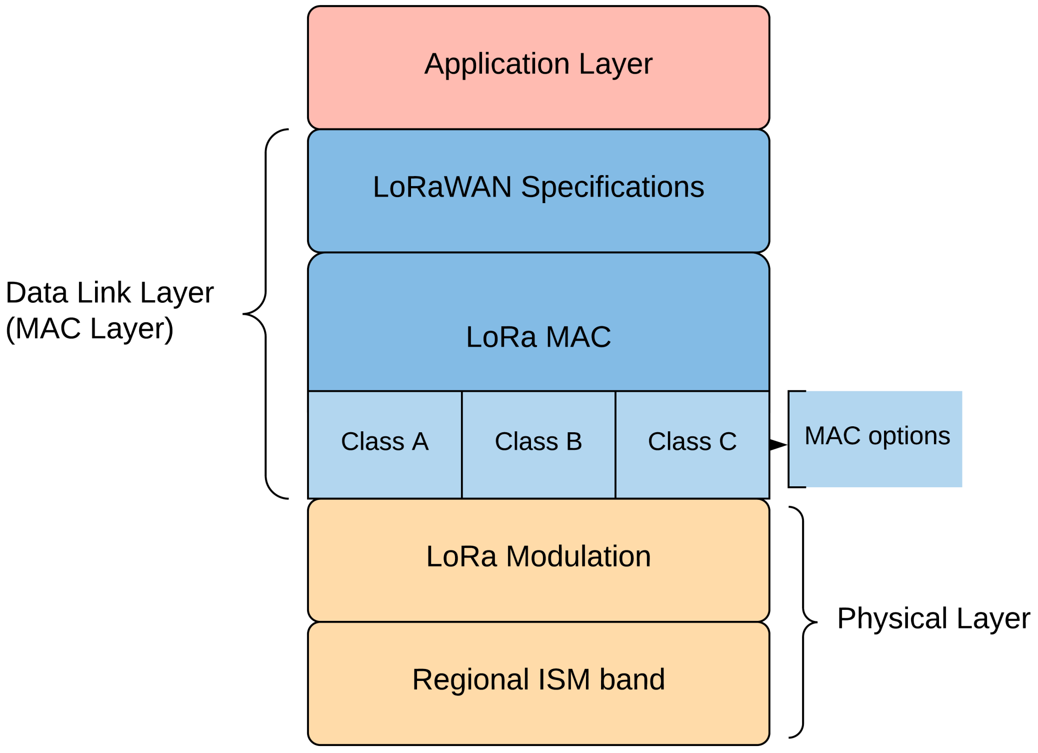


Figure 2.1: LoRaWAN Stack.

## 2.2 LoRaWAN: Network Stack and Architecture

LoRaWAN network architecture initially introduced in Chapter 1 Figure 1.2 highlights the four major components (end nodes, gateways, network and application servers) ins a star-of-stars topology.

The specifications [29] refers to messages transmitted and received by end devices as uplink and downlink messages. End devices send uplink messages to the network server by transmitting a LoRa message to one or several gateways which in turn relay the message to the network server. Similarly, the network server sends a downlink to an end device by relaying the message to a single gateway that transmits the LoRa message to the end device. The network server is responsible for routing device packets to the associated application server and handling application server request and scheduled downlink messages.

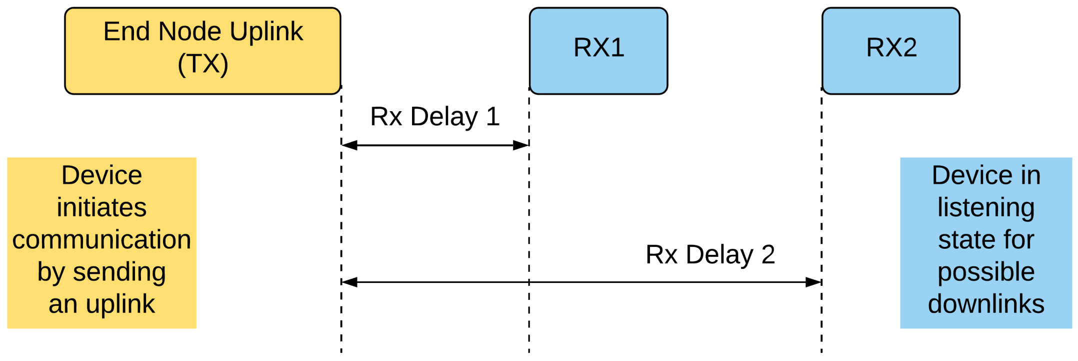

End Node Uplink (TX)
RX1
RX2
Rx Delay 1
Device initiates communication by sending an uplink
Rx Delay 2
Device in listening state for possible downlinks

Figure 2.2: LoRaWAN Class A communication scheme.

### 2.2.1 LoRaWAN MAC layer

As depicted in Figure 2.1, LoRaWAN builds upon the physical layer and defines the MAC layer and the specifications for the data link layer[30]. The LoRa alliance maintains a detailed and updated specification [29] that target organizing the communication of battery-powered LoRa end devices.

LoRaWAN recognizes three different classes of devices (MAC options); class A is the baseline that all LoRaWAN devices must implement to comply with the specification. Classes B and C are optional, depending on the end device required functionality and power constraints. Class A is designed to be the lowest energy consuming class. The energy efficiency is achieved by allowing all end devices to enter a deep sleep state most of the time. The longevity of the deep sleep state is achieved by partially omitting the downlink messages (end device is not listening for new messages), and therefore, the device can efficiently enter deep sleep. Only when the end node sends an uplink, it waits for downlink in a predefined receive window as depicted in Figure 2.2.
In comparison, classes B and C consume more energy to allow for flexible receive windows and downlinks.

A comprehensive analysis done in [31] shows the implication of varying SF and device class on total power consumption. Devices such as GN are class A devices and the firmware should account for class A uplink and downlink communication scheme, furthermore, the application layer is expected to employ a cretin level of intelligence to properly utilize the enforced mac layer leaving room for further research and improvements [32].

### 2.2.2 Adaptive Data Rate

Building upon the offered controllability of the physical layer, end devices can optionally allow the network server to control the end node SF (data rate) and transmission power. Since the network server and the gateways are not power constrained, they can use the communication metadata to optimise the network traffic and end nodes energy consumption in runtime.
End nodes can opt-in the Adaptive Data Rate (ADR) by setting uplink ADR bit and can detect if the network is actively controlling the data rate by checking the ADR downlink bit [29]. The ADR runtime algorithm at the end node is considered relatively simple and offloads all the overhead to the network server[33]. The network traffic optimisation opportunity offered by LoRaWAN stack is the subject and focus of multiple research papers [34] [35].

GN can be used in various applications and use-cases, therefore, activating ADR is an important application decision that must be critically evaluated before various deployment scenarios. Using ADR is not recommended [36] when the link budget calculations are known in the deployment scenario, furthermore, if the end node is expected to be mobile, using ADR is shown to have negative impact due to the implied overhead [4] and a reduced packet delivery ratio [36].

## 2.3 LoRaWAN Security

IoT security is a major research focus due to the recent technology development. Moreover, the research areas are extensively diverse and require in-depth analysis that is typically associated with multiple proposals and countermeasures. Extensive surveys show an exhaustive categorisation of various IoT vulnerabilities [37]. Most IoT platforms and protocols share similar security vulnerabilities, and LoRaWAN is no different. End nodes inherit the security profile of the utilised communication protocol. Firmware designers must comply with standard specifications (functional requirements) and employ research best practices and countermeasures.

In this section, we inspect LoRaWAN (version 1.1) security specifications and highlight relevant research. The analysis provides background and a basic understanding of the current security profile of a typical LoRaWAN end node. GN HW and SW improves upon LoRaWAN typical end node security profile, which is further illustrated and analysed in Chapter 6.

### 2.3.1 Authentication and Data Encryption Frameworks

LoRaWAN is built around security as it relies on symmetric cryptography with authentication and secure communication frameworks [38]. LoRaWAN employs a lightweight [39] key-based encryption scheme based on 128 bit Advanced Encryption Standard (AES).

Two methods of activation are possible: Activation by Personalization (ABP) and Over The Air Activation (OTAA). In the case of ABP there is no joining procedure, and all the authentication data is hard-coded [40] such as DevAddr and four session keys.
The method is considered to be prone to replay message attacks and the use of OTAA is a recommended alternative [41].
In OTAA, the communication between the end node and network server is session-based. Each session must be activated for end node-server communication to take place, this is

| Key | Origin |
|---|---|
| NwkKey | Stored before session activation |
| AppKey | Stored before session activation |
| JSIntKey | NwkKey + DevEUI |
| JSEncKey | NwkKey + DevEUI |

Table 2.1: LoRaWAN v1.1 before activation keys (OTAA)

done using pre-shared and derived keys scheme [33] [29][40].

In Table 2.1, we show the root keys needed before activation in LoRaWAN v1.1. The end node derives the session keys after a successful joining procedure using the Join Accept message and the previously known activation keys as depicted in Figure 2.3. The generated keys are used to check uplink and downlink message integrity codes (FNwkSIntKey and SNwkSIntKey), and encrypt every MAC and payload data packets using NwkSEncKey and AppSKey, respectively.

### 2.3.2 Vulnerabilities

LoRaWAN V1.1 introduces new security countermeasures to vulnerabilities discovered in the initial stranded specification (V1.0). End Nodes such as GN are expected to follow and comply with the latest specification. Countermeasures and best practices from older research similar to [41], which is done on the older specifications should be revaluated and only applied if the vulnerability still exists. New detailed analysis and exploration of the new specifications is done in [42].

The attack tree describes several vulnerabilities and their countermeasures. End nodes firmware design can apply described best practices such as avoiding using ABP to counter keystream reuse and denial of service attacks.

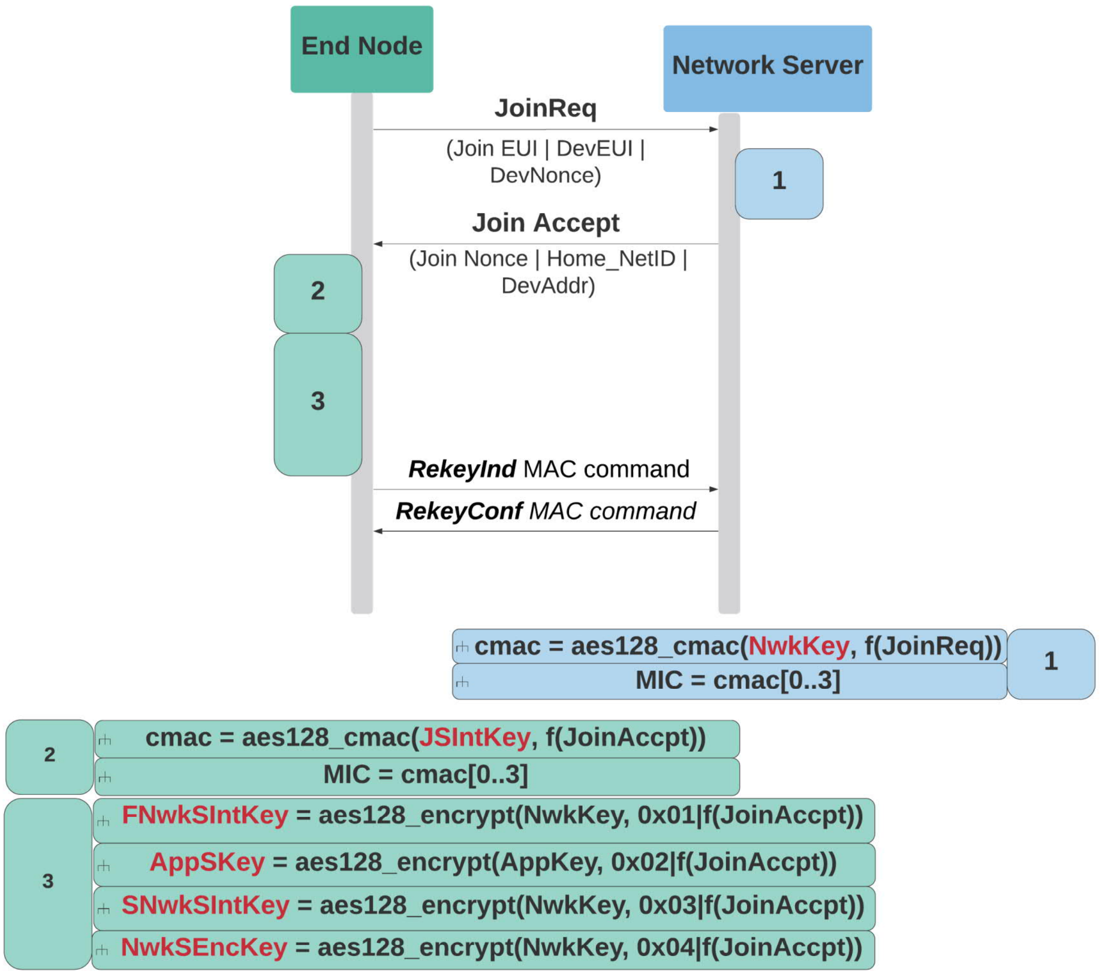

End Node
Network Server
JoinReq
(Join EUI | DevEUI | DevNonce)
1
Join Accept
(Join Nonce | Home_NetID | DevAddr)
2
3
RekeyInd MAC command
RekeyConf MAC command
cmac = aes128_cmac(NwkKey, f(JoinReq))
MIC = cmac[0..3]
1
2
cmac = aes128_cmac(JSIntKey, f(JoinAccpt))
MIC = cmac[0..3]
3
FNwkSIntKey = aes128_encrypt(NwkKey, 0x01|f(JoinAccpt))
AppSKey = aes128_encrypt(AppKey, 0x02|f(JoinAccpt))
SNwkSIntKey = aes128_encrypt(NwkKey, 0x03|f(JoinAccpt))
NwkSEncKey = aes128_encrypt(NwkKey, 0x04|f(JoinAccpt))

Figure 2.3: Over The Air Activation in LoRaWAN 1.1.

Action 1,2: Message Integrity Check (MIC). Action 3: derivation of session keys.

# Chapter 3

# Generic Node Hardware Design

Hardware design is one of the most challenging aspects of the development life cycle of an embedded device. Collaboration between multiple vendors and hardware manufacturers is encouraged to overcome all the challenges that accompany building and designing a new device. GN is no different, as to achieve the vision of TTN, a collaboration between TWTG, Microchip and ARM took place where they targeted building a reference design for a multi-sensor general-purpose LoRaWAN end device. The target of such a device is to supplement the LoraWAN ecosystem; furthermore, reduce the complexity of designing LoRaWan end device while still pushing the industry in multiple areas such as security and battery life. Microchip and ARM engineers developed the hardware requirements for the device (MCU, external memory, number of sensors, secure element) and worked closely with TWTG to design the schematics and the PCB.

TTN role as part of the collaboration is to build a firmware for the GN and design software sample applications that show the benefits of using this reference design. One of the initial steps in the firmware development is identifying the hardware components and analysing the capabilities and limitations of the device. From a hardware perspective, GN provides a multitude of challenges to any embedded firmware developer; this is a result of the fact that the node accommodates many sensors and onboard components. The firmware development of such device resembles developing several independent IoT

devices.HW validation is one of the activities undertaken by TTN acting as an entry point for the firmware development.
The validation includes the following set of activities:

1. Analyse the device schematics and provide TWTG (HW designers) with feedback related to any functional issues and required improvements.

2. Write sample SW applications to test the sensors and all the onboard components.

3. Validate the usability of GN and the effect of the device enclosure.

The validation of GN HW design provides a comprehensive overview of the device functionally and details the issues that face HW and SW designers in the development of similar IoT devices. In the upcoming sections in this chapter, we target the following:

- Highlight the components of a next-generation LoRaWAN IoT device.

- Show the promising aspects of using certain HW components.

- Discuss relevant HW design issues based on generated findings, thus opening the door for future research that builds upon these findings.

- Provide best practices and relevant research solutions that could apply to the development of similar devices.

## 3.1 Device Components

The GN is constructed to function as a multi-sensor gerenal purpose IoT device that is operated by single double A battery. As seen in Figure 3.1, the device contains the following components:

- SAMR34/35 Microchip system in package which integrates a 32-bit ARM Cortex M0+ Micro-Controller Unit (MCU) with an Ultra High Frequency (UHF) transceiver

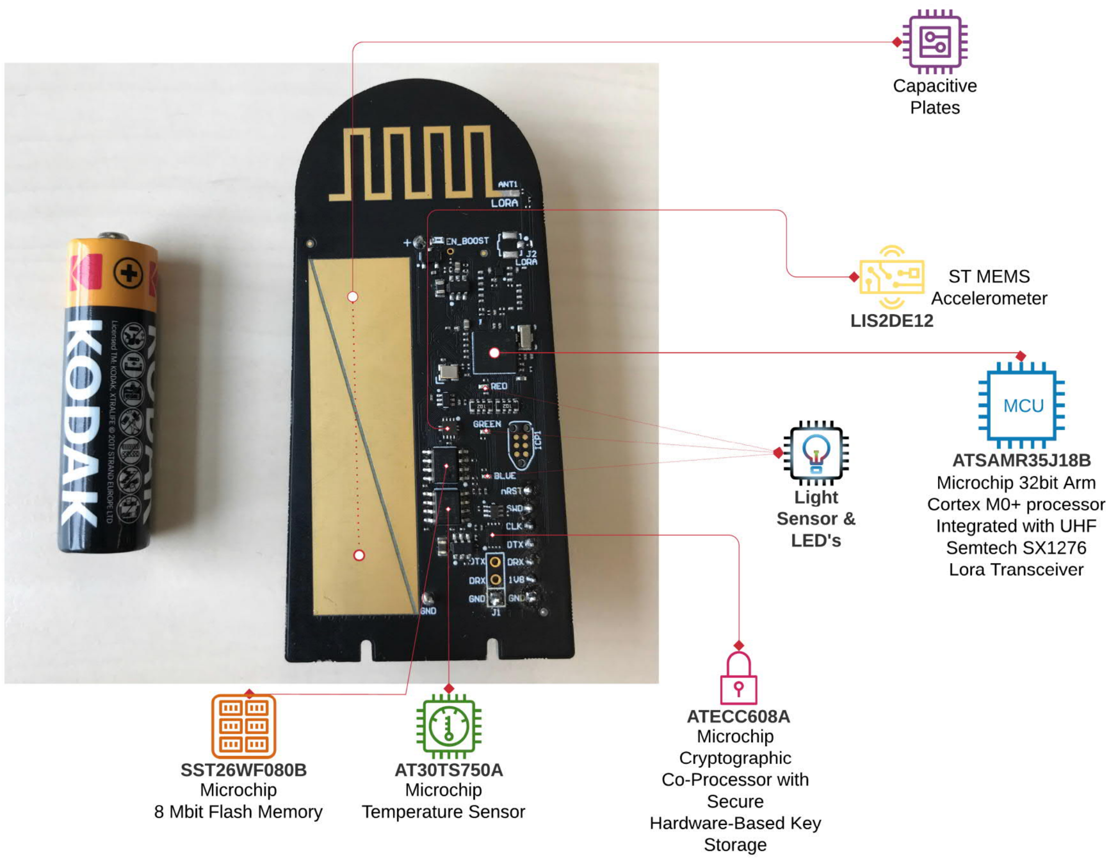

Capacitive Plates
ST MEMS Accelerometer
LIS2DE12
MCU
ATSAMR35J18B
Microchip 32bit Arm Cortex M0+ processor Integrated with UHF Semtech SX1276 Lora Transceiver
Light Sensor & LED's
ATECC608A
Microchip Cryptographic Co-Processor with Secure Hardware-Based Key Storage
SST26WF080B
Microchip
8 Mbit Flash Memory
AT30TS750A
Microchip
Temperature Sensor

Figure 3.1: Generic Node components.

communication interface (Semtech SX1276) that supports LoRa and Frequency-shift keying (FSK) modulation.

- Four sensors which are Accelerometer, Temperature, Light (backward LED) and Capacitive plates (proximity, touch, slide and moisture sensor).
- ATECC608A cryptographic coprocessor (referred to later as secure element).
- SST26WF08 8 Mbits of external flash memory.
- Three LEDs (RED led functions as a light sensor as well).
- Translucent Polycarbonate IP64 rated enclosure depicted in Figure 3.2
- Other miscellaneous components such as regulators and crystal oscillators.

The components mentioned above construct a reference design that allows new products (such as the MCU and the secure element) to establish a presence in the market while still reducing the cost and saving on power consumption through various techniques.

## 3.2 HW Design Best Practices

Reducing energy consumption and cost of production is the main design objectives in IoT HW design, and every approach that can empower achieving such objective is encouraged and highlighted in the research. GN HW design incorporates many of the best practices that make it a useful reference for any future development. Some of these practices are mentioned below:

1. Dedicated MCU interrupt lines for sensors. (Temperature, Accelerometer and Capacitive plates) providing alert functionality for certain events.

   **Positive impact**

   - Improved battery life as the MCU can sleep most of the time and wake up only when the interrupts are triggered.
   - Allows application design to become event-driven.

   **Negative impact**

   - Occupies valuable MCU pins even if one of the sensors is never utilised for a specific use case.
   - Application dependant practice that might be unnecessary for some use cases such as interval-based data collection.
   - Interrupt lines are prone to noise and might unnecessarily trigger the system in noisy environments.

2. Double usage of PCB mounted LED as a light sensor, taking advantage of the physical properties of a photodiode acting as a capacitor in case of reverse bias and

measuring the time of capacitor discharge time as an indicator of the surrounding amount of light [43].

**Positive impact**

- Cost effective approach of implementing a light senor, elimnating the need for a didacted light senor.

**Negative impact**

- No interrupt line, requiring all sensor measurements to be done by polling.
- Interval based light sensor measurements requires the processor to fetch the data every time, reducing the number of use cases that can be implemented with this sensor.
- Reduced accuracy and sensitivity in comparison with a dedicated sensor.

3. Power switching MCU pins/lines that allow the user to power on/off the sensors depending on the application.

   **Positive impact**

   - Improved battery life and increased power management capabilities.
   - Allow for sensor specific applications without compromise on energy.

   **Negative impact**

   - Can be complex to implement and requires a dedicated power distribution load switches. More on the topic in Chapter 5.
   - Requires multiple GPIO pins and load switches per each sensor, making it an expensive practice. Sensor grouping (power up several sensors simultaneously) is one approach to reduce the number of pins and switches needed, but this comes with the added sacrifice of application modularity.

4. Battery voltage indicator monitoring using an ADC pin is a common practice that is critical for any battery-powered IoT device and combining this practice with the previously mentioned sensor power switches makes it possible to implement low battery modes that prolong the battery life.

## 3.3 Enclosure Effect

Battery operated end devices such as GN are typically used in various environmental surroundings ranging from typical indoor locations such as homes and offices to harsh outdoor locations such as farmlands and animal husbandries. A general-purpose device such as GN must be able to function properly and survive a number of environmental conditions for several years. This means that having an enclosure is critical to the functional operability of the device.

The benefits of using an enclosure are overshadowed by the complexity of designing one. Since GN will always be used with the protective casing, this means that it is considered a critical HW component that must be appropriately verified and validated. When you combine a multi-sensor device with an enclosure, we can get an idea of the challenges that are faced in IoT HW design and the existing limitations and compromises that are currently existent in the industry.
Considering the fact is that each sensor can be a separate production device on its own; analysing the enclosure effect on all of the onboard sensors is an excellent research opportunity. On the other hand, it also highlights the problems that accompany building a general-purpose end device as the enclosure design needs to satisfy multiple requirements that don't necessarily exist in other one sensor IoT devices.

As seen in Figure 3.2 and in Appendix B Figure B.1, the enclosure covers the entire body of the PCB allowing GN to be IP64 certified device. The material of the enclosure

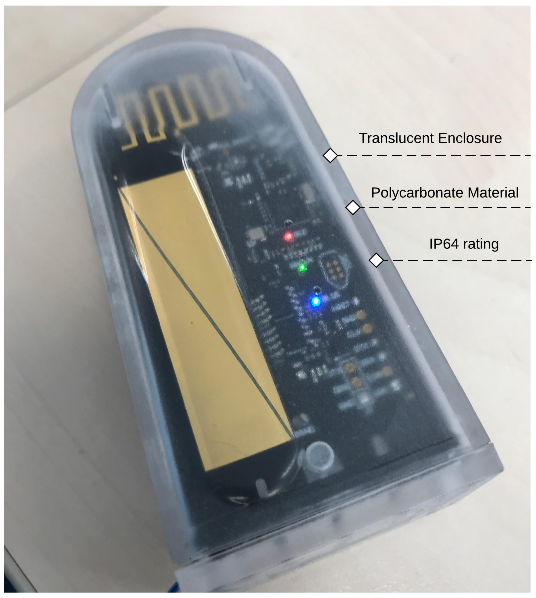


Figure 3.2: Generic Node enclosure.

is made of polycarbonate material which is typically transparent and used to protect the device in harsh environments [44]. It also can act a heat insulator/deflector, which makes it a focus on the process of HW validation. Using a polycarbonate material, in this case, has positive and negative effects that will be highlighted per each sensor.

### 3.3.1 Test Procedure

#### Surrounding environment

Due to the nature of GN and its inherent capabilities of sensing environmental changes, the nodes test environment (either indoor or outdoor) depended on the sensor under test. To perform boundary value testing, extra equipment such as hot air blower, fridge and a flashlight helped in generating the needed conditions.

#### Test setup

Tests conducted on GN sensors included the following components:

- One Generic node with the polycarbonate enclosure, see Appendix.
- One Generic node without the polycarbonate enclosure.
- Two serial to USB converters (FTDI232), connected to both nodes with jumper wires to allow for increased sample rate and stress tests.
- Two external batteries connected to each node with jumper wires, to allow for synchronous power switching (since access to the battery is blocked by the enclosure), see Appendix B Figure B.1
- Ubuntu machine with two separate serial terminals for data logging.

Multiple software test applications allowed for testing the enclosure effect on each sensor. The test application focused on isolating and testing only one sensor allowing for concise and discrete test results.

### 3.3.2 Effect On Temperature Sensor

Multiple tests have been conducted on the temperature sensor in order to validate the HW, these tests provide meaningful observations from a theoretical and practical stand points. The Temperature sensor accuracy is the most affected by the enclosure as seen in Figures 3.3, 3.4

#### Test application

The applications uploaded on the nodes were sampling a new temperature readings every five seconds. Both devices were started simultaneously. Every serial message was received by the serial terminal on the Ubuntu machine and logged with a corresponding timestamp.

**Test 1: 15.02.2019 to 18.02.2019**

As seen in Figure 3.3, this test targeted exposing the devices to various temperature

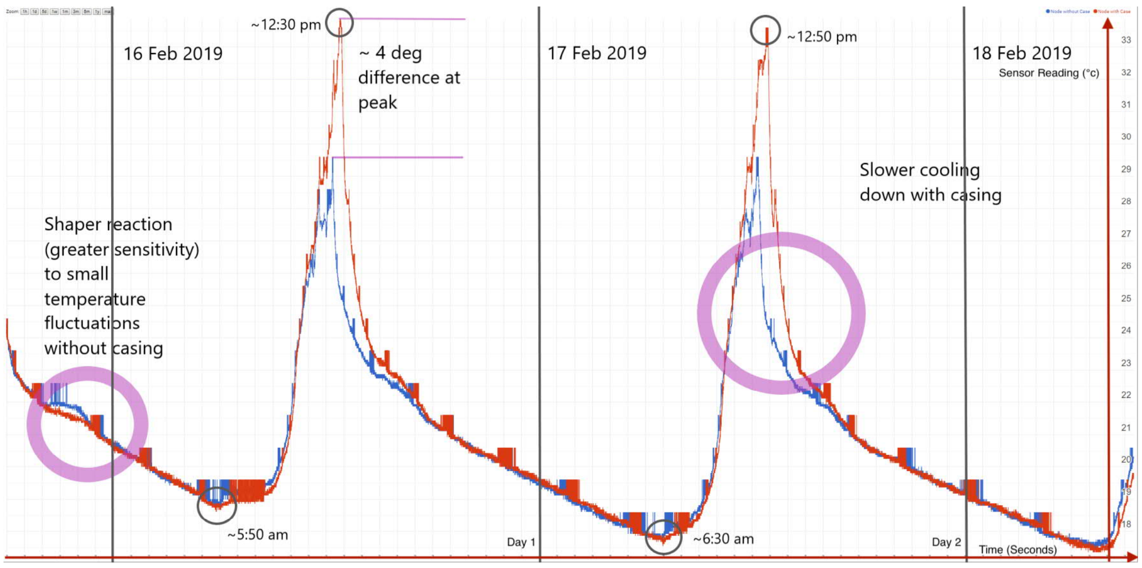


Figure 3.3: Test 1: Enclosure effect on temperature measurements.

changes, both during the night and during the day, this was done by placing the nodes right next to the office windows. The test lasted for approximately three days (weekend period) taking 45000 sample measurements, in this period the devices did not move and had no external interference from any individuals.

**Test 2: 22.02.2019 to 25.02.2019**

As seen in Figure 3.3 (Blue is a node with enclosure and Red is Node without enclosure), the Second test was intended to omit direct sunlight exposure and measure temperature in a normal room climate. The test setup was placed on a table in the middle of the room, and devices had a carton covering to ensure sunlight does not heat up the enclosure. Similar to Test 1, The setup was left untouched for a weekend period.

### Test impressions and overall take

Each of the tests separately gives an idea of how big of a role the casing plays in temperature readings. In both cases it can be seen, that the sensitivity of the sensor is reduced (less sharp reaction to small temperature fluctuations), heating in the peak day temperature is significantly higher, thus cooling down takes more time. In the test conducted near

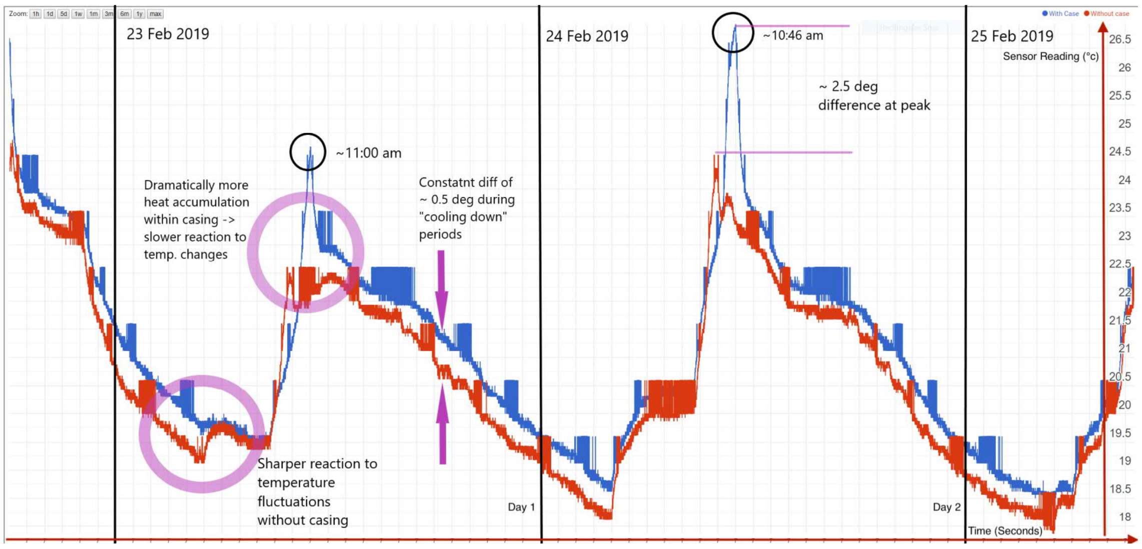


Figure 3.4: Test 2: Enclosure effect on temperature measurements.

Blue plot indicates sensor measurements with an enclosure.

Red plot indicates sensor measurements without enclosure.

the window (Test 1), where temperature changes are more significant due to the sensors being closer to the outside, some noticeable details can be seen:

1. Due to direct sun exposure, heating in peak daytimes, where the sun effect is the biggest, the enclosure overheats and inaccurately show an inaccurate increased temperature reading of approximately four degrees in comparison to the node with no enclosure.

2. The sensitivity is slightly less with the casing than without the casing.

3. After cooling down to the evening temperature, both devices readings even up showing similar readings until the next day where inaccurate measurements reoccur due to the enclosure effect.

In the test conducted in the middle of the room (Test 2), where surrounding temperature changes are less frequent, and devices were not exposed to direct sunlight, some noticeable details can be seen:

1. The sensitivity with the enclosed node is significantly less as some temperature fluctuations are barely recorded (due to the fact that room temperature changes are not persistent, allowing the node without the enclosure to pick up the changes while the enclosed one unable to detect the change).

2. Heat accumulation problem still exists during the day temperature peak times, same as Test 1; although the inaccuracy in peak readings is reduced from four degrees Celsius to two and a half degrees.

3. A constant difference in temperature readings of a half degree can be observed throughout the whole test, except for the smallest regions where the outside temperature starts to rise. This means that the extra heat accumulated by the casing is not dissipated in the room temperature environment.

As a conclusion, the enclosure dramatically affects the temperature readings. Moreover, the measurement overshoot is inconsistent and depends drastically on the surrounding environment. The analysis shows the need for a SW solution to solve the inaccuracies induced by using an enclosure (more on that in Chapter 3) as HW changes will increase the cost of development and production leading to the increased final cost of the device.

### 3.3.3 Effect On Capacitive Plates

The enclosure material acts as an insulator making testing the capacitive plates with the casing a priority to ensure the functionality of this critical component. The procedure of testing involved increasing the sensitivity of the Peripheral Touch Controller (PTC) to allow for touch detection.
As seen in Figure 3.5, with the current enclosure design, it is essential that the distance X between the enclosure surface and the Printed Circuit Board (PCB) mounted capacitive plates is within the max sensitivity range. Adjusting the channel touch detection threshold and hysteresis with relation to the plates touch surface area is necessary to condition the

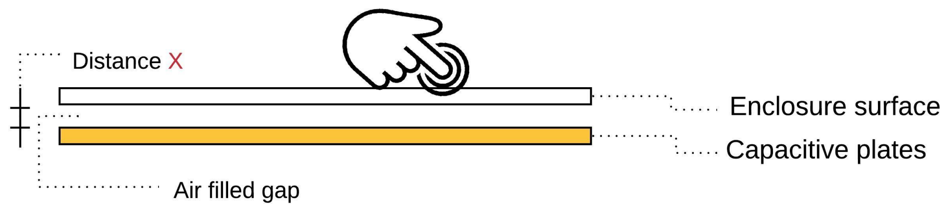


Figure 3.5: Enclosure effect on the capacitive plates.

| Reference light level (lux) | Light (%) without enclosure | Light (%) with enclosure |
|---|---|---|
| 0 | 0% | 0% |
| 231 | 3.179% | 2.987% |
| 570 | 7.387% | 7.152% |
| 2400 | 40.276% | 38.137% |
| 4300 | 78.291% | 76.254% |
| 6200 | 95.332% | 92.142% |

Table 3.1: Enclosure effect on the light sensor.

plates to act as proximity sensors allowing for the detection of any charges on the surface of the enclosure.

### 3.3.4 Effect On Light Sensor

One of the benefits of using a polycarbonate enclosure is the capability of producing an almost translucent casing as seen in Appendix B figure B.1 that doesn't block light making it possible to mount the light sensor (red LED) on the PCB, reducing the casing impact significantly on the light-sensing capabilities. Tests conducted with and without the casing seen in Table 3.1 show the enclosure material minimum impact on the light-sensing capabilities.

### 3.3.5 Effect On Lora Radio and Accelerometer

Lora radio antenna is one of the most indispensable HW components and analysing the Received Signal Strength Indicator (RSSI), and the Signal to Noise Ratio (SNR) information obtained from the connected gateways was important in evaluating the effects of the enclosure (Appendix C Figure C.2). Based on multiple testing procedures done by another team member, we concluded that the enclosure material does not affect the transmission and receiving capabilities of GN. Furthermore, the accelerometer functionality is also not influenced in any way by the addition of an enclosure, making it the only sensor that does not require any calibration/ adjustment of any sorts thus allowing the user to use and trust the data generated from the sensor.

### 3.3.6 Impressions and Possible Solutions

As observed with the analysis of the enclosure effect on the different sensors and HW components, we can observe how the capacitive plates use cases (water, moisture and touch detection) are complemented by the usage of an enclosure. On the other hand, we can see how using an enclosure in this design affects the sensitivity and accuracy of the temperature and light sensors.

This goes to show the complexity of designing and validating a general-purpose device and the need for research-oriented solutions to fix the design challenges highlighted and incurred by the usage of several components such as temperature sensor calibration done in [45]. It is also worth highlighting that the enclosure effect analysis only shows the limitations induced by the usage of the casing; this effect becomes more significant when considering the measurement limitations of the sensors.

# Chapter 4

# Generic Node Software Design

General-purpose devices such as GN introduce several challenges from a software design perspective. There are multiple areas where the hardware intersects with the software, and a proper SW design needs to allow for the usability of all HW components. Being a reference design device increased the need for a modular software that allows any user from any background to utilise the device and be able to benefit from its functionality.

Embedded SW design has matured significantly throughout the years, and it is safe to assume that no one needs to rebuild the wheel every time a new device is produced. A huge number of references and books exist in the field, allowing any industry such as IoT to move forward building upon previous experiences.

GN as a device is a mixture of multiple individual components and sensors, all of which are not considered new in any way, this fact alleviates some of the challenges and allows for building upon previous knowledge.

Only by approaching SW design with this mentality, we can build a device that pushes the industry a step forward in the right direction. Since the device is targeted to push the envelope of what is possible in the field, it is also necessary to approach the SW design with a cost-effective mentality similar to any IoT production device, this means focusing on fixing some of the challenges that face manufacturers and designers in the field and showing possible solutions to these challenges.

The approach to building the firmware of GN treats it as a production device; this means planning and executing said plans gives a clear overview of the practical issues that face engineers building similar products.

## 4.1 SW Development Plan

As shown in Chapter 3, The first step was validating the HW components from a firmware development perspective, and while this showed practical issues and noticeable HW best practices, any action taken to fix some of these issues proved to be either very costly or impractical to implement in the HW side.

Since TTN is fully responsible for the SW development of the device, we approach it by taking into account the HW capabilities and limitations and devising a SW development plan that adheres to all the requirements and addresses some of the issues.
To approach SW design, it is required to define a set of goals that act as targets in the development phase.
Some of the high priority goals being:

- Reduce SW development time in any way possible while maintaining quality.
- Allow for SW modularity and re-usability taking into account possible future developments (For example the possibility of a new HW revision with a change in MCU or any of the sensors).
- Show the capabilities of the device by building sample applications that show the promise of using a LoRaWAN end device (long-range and low power).

Building on the above-defined goals, we adapted the V-model SW development process. The benefits of using this model and combining it with an agile approach are vital in achieving the goals we set for this project. While discussing the benefits of such a model is out of our scoop, it is essential to mention it and indicate the need for a clearly defined

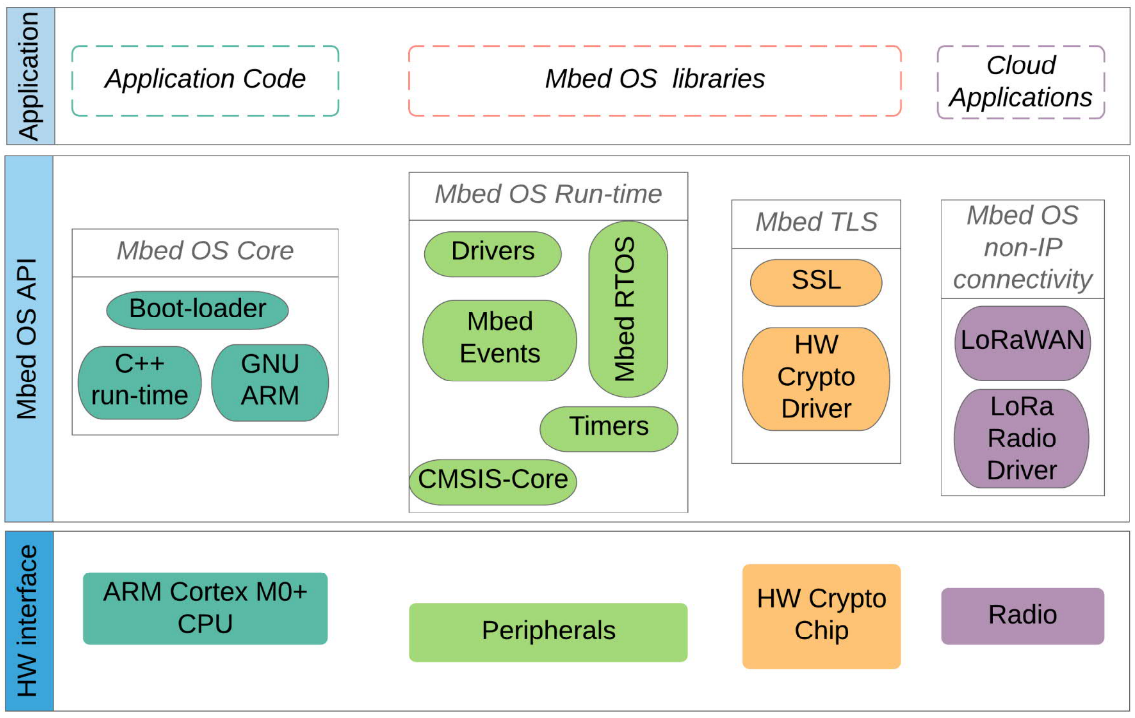


Figure 4.1: Software architecture and components.

process in the development of any IoT product.

In the subsequent sections, we show various aspects of the SW development cycle and highlight critical approaches and findings.

## 4.2 SW Components

Similar to HW, SW can be divided into multiple components; each of these components is a critical part of the final product. Several decisions are made when choosing the different SW components in the development of any new device. These decisions dramatically affect the development process and the end product functionality.

### 4.2.1 Embedded Operating System (Mbed OS)

As we approach an era where micro-controllers are becoming less constrained memory-wise, we are observing the increased usage of sophisticated real-time operating systems that take advantage of the increased memory. Deciding whether to use an RTOS or opting for a bare-metal approach is highly application dependent decision that should be considered in requirements and architecture definition phase. In the case of GN, it was clear that a simple Real Time Operating System (RTOS) is needed to ensure application versatility.

GN is a new device in a growing market, Microchip and ARM were involved in every development step of the way to guide TWTG and TTN to overcome the challenges of building a LoRaWAN end device successfully. One of the initiatives established by ARM to accelerate the development of the IoT industry is the development of an open-source embedded operating system called Mbed OS. Mbed OS provides a middle ground between using a bare-metal scheduler and an overly complicated memory-intensive RTOS. The benefits of using it are immense and significantly accelerated the development time of GN while still adhering to all the goals set for product development.

Using Mbed OS was a great decision that showed its benefits throughout the development cycle of GN, and while it would be lengthy to illustrate all the benefits, it is beneficial to mention some critical aspects that assisted in the development:

1. All Mbed OS related SW (scheduler, drivers, libraries) are unit and integration tested with system testing done on multiple targets. Any SW that is generated as part of GN development will be similarly tested, increasing the testing coverage of the entire system significantly in comparison to any other system that does not use Mbed OS.

2. Modularity and reconfigurability being an essential characteristic of Mbed OS makes it perfect for a general-purpose device such as GN.

3. The communication protocols drivers are implemented in an open-source form. reducing the cost of development, and the amount of resources and knowledge needed to build a product. This allows any user to focus on building applications that solve real-world issues and not be burdened by the complexity of the SW development

needed to bring a product to the market.

4. The OS is built from the ground up with support for the various IoT requirements such as low power modes and efficient data transfers. This eliminates the need for optimisation to the prebuilt drivers, and switches the focus towards optimising all the newly developed SW.

5. Mbed OS is released under an Apache 2.0 licence; this means it can be used in commercial products with minimal to no fees. This makes it ideal for young companies that are unable to pay the hefty licensing fees for the commercially available real-time operating systems. Complemented by the community and manufacturers support of the drivers, it is definitely helping push IoT forward and can accelerate the time to market of LoRaWAN end devices.

A simplified SW architecture and an illustration of the several SW components can be seen in Figure 4.1, the SW development can be divided to three different layers (HW interface, Mbed OS API, Application), each component within a layer should be easily replaceable with minimum changes to the neighbouring components. Each layer should be abstracting the layer before it, and if one of the layers is changed for any reason, only minimum SW changes should be done without the need for a complex software architecture redesign.

If we observe the Mbed OS API layer, we get an understanding of how Mbed OS provides the flexibly needed in an agile fast-paced IoT development process. To further illustrate, in the development of GN there is only need for building and configuring some MCU peripheral drivers in the Mbed OS runtime sublayer, every other component within the layer works seamlessly even with a new MCU (SAMR34/35) that was never used before in the market.

This also means that the LoRaWAN stack is compatible with any MCU architecture supported by Mbed OS and the fact that it is modular and easy to configure encourages developers to experiment and build applications in a smooth and straight forward manner. By using Mbed OS in the development of GN, it was easy to build upon previous SW layers and focus all the resources in building the Application layer and Mbed OS libraries that take advantage of GN capabilities.

### 4.2.2 Software Drivers (Sensors and Flash)

The usage of Mbed OS accelerates the development of any MCU related SW component since Mbed OS leverages the shared architecture design between the various targets. This allows it to support multiple targets (different vendors) with minimum need for SW change.

Sadly this concept does not apply for the onboard sensors as each sensor will have its unique architecture and HW interface even if it was designed by the same HW manufacturer.

By design, sensors target collecting analogue data and acting as a vessel by converting the collected data and delivering it in a digital form to a receiver that will analyse it and maybe act upon it. Due to the versatility of methods to collect and measure environmental data, sensors are inherently incapable of sharing the same architecture/design. The only common component between sensors is usually the HW interface, more specifically the communication protocol such as SPI, I2C and UART.

From a SW development perspective, this means that each sensor often needs a set of commands to either activate certain functionality or to deliver the measured data, all of which are occurring on one of the previously mentioned buses.

Having to develop sensor drivers can be a daunting and challenging task that should only be undertaken if the sensor is unique, and the generated data is needed. Multiple sensor HW manufacturers already support their sensors with standardised platform-independent C language SW drivers such as ST, and this allows the firmware developer to integrate the method of communication (bus used usually is platform dependent) and then

it becomes a matter of activating the relative functions using the provided standardised SW driver.
Some other manufacturers such as Microchip provide sample drivers that are function-specific and platform-dependent (Atmel start, MPLAB and ASF); this makes integrating their sample drivers with a different platform a challenging, time-consuming task.
The same applies to other components such as Flash memory and not only limited to sensors (although the support varies).

By using Mbed OS we overcome a number of challenges. But, since most of GN sensors are not supported by the Mbed OS platform, it is clear that developing the drivers for some of the sensors is not something that can be escaped.
In mature embedded industries such as automotive, tier-one suppliers usually focus their resources on developing applications and only interact with the drivers when optimisation is needed, as building a driver is typically a task that concerns the HW manufacturer of the device.

Building Mbed OS compatible GN sensor drivers can be considered an opportunity, as the drivers can be designed from the start to support LoRaWAN various transmission modes/approaches with a focus on enhancing the battery life.
In the next subsections, we highlight some of the approaches taken in building GN SW drivers and implementation practices that lead to improvements to the device functionality.

#### Development of Temperature, Accelerometer and Cryptographic Co-processor drivers

In building drivers, it is important to understand the architectures that are expected to utilize it and target building a modular SW interface that is flexible, efficient and supports multiple development platforms with minimal configurations.
An example of modularity is explained by understanding the capabilities of Cortex M0+ architectures as they target conserving energy over processing capabilities.

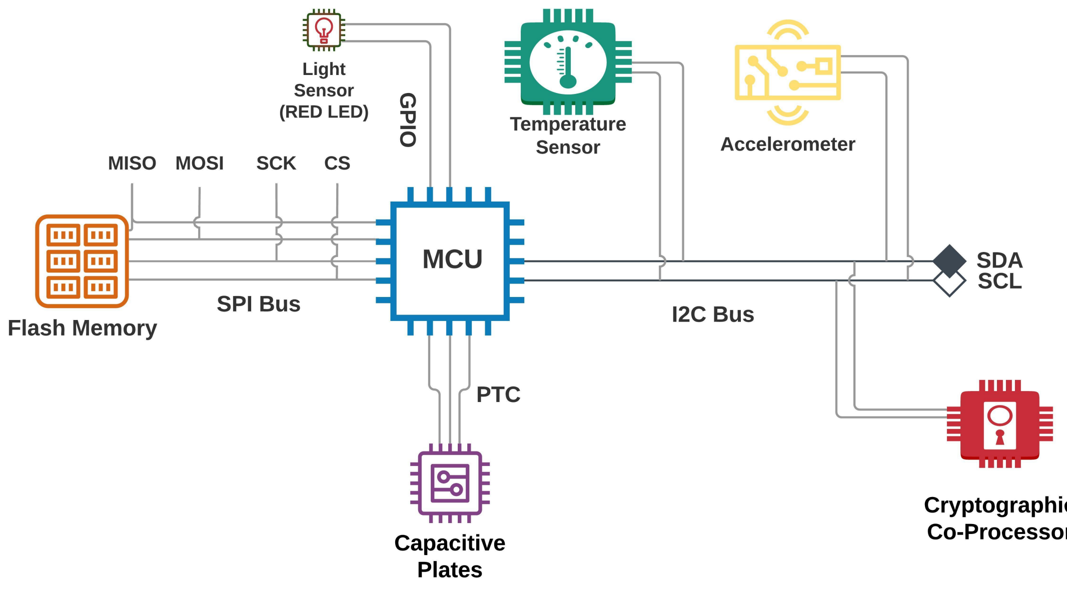


Figure 4.2: Generic Node sensors and flash MCU connections.

Best practices to achieve modularity and flexibility of sensor SW drivers are:

1. Reduce MCU-sensor communication to a minimum.

   **Justification:** As seen in Figure 4.2, both accelerometer, temperature sensor and the cryptographic co-processor communicate with the MCU using a shared I2C bus and acting as slaves. This approach has the obvious benefit of reducing the number of PINs needed for a multi-sensor device such as GN.

   Activating any functionality of the sensors requires the MCU to be in an awakened state (more on the topic in Chapter 5) and sending several bytes and waiting for a response. From a battery lifetime improvement perspective, if we reduce the amount of time taken to retrieve the data, we can return back to sleep faster, which ultimately reduces the overall energy consumption.

2. Allow for modularity by separating data processing functions from data retrieval functions.

   **Justification:** LoRa capable devices strive to achieve long-range and low power, and this comes at the cost of bandwidth. This means reducing the amount of data sent and received as much as possible is a desirable feature; furthermore, the separation provides the application layer with a choice of either formatting the received sensor data or opting for sending the data in its raw form.

3. When processing is not needed it should always be omitted.

   **Justification:** Edge devices such as GN can operate with respect to application needs, and in most application use cases, cloud computing is the preferred model in favour of edge computing. The topic is discussed in more detail in Chapter 5, but it is important to understand that passive devices such as GN usually act as data collectors and rely on other devices to make sense of the data and act upon it.
   So maintaining long term battery survivability is more critical than any other feature.

   A straightforward example is GN temperature sensor and the representation of temperature measurements sent from the sensor to the MCU through the I2C bus. By referring to the sensor datasheet [46] we see that positive numbers and negative numbers have special 16-bit representation that needs to be formatted for the measurements to become representative (**1111 0101 1110 0000** represents -10.125 °C). By sending the raw data without formatting and building cloud libraries that parse the data, we save a considerable amount of energy, and we increase the battery life significantly.

The benefits of applying some of these best practices are highlighted in Chapter 5.

### Development of Flash driver

External memories have multiple usage scenarios when used within devices such as GN as they can be incredibly useful in storing the sensor data or act as an enabler for firmware updates over the air.

Building the flash driver benefits from all the previously mentioned best practices with the only variable being, the use of the SPI bus for Flash-MCU communication instead of I2C as seen in Figure 4.2.
Reducing the amount of SPI communications needed to write/erase a sector/block of data is the key idea [47].

An example would be using block or chip erase commands instead of sector erase commands if it is suitable to do so (application dependent), this mindset lowers the amount of energy consumed in the long run, and the modular design approach (if applied) facilitates the application layer energy optimisation techniques.

### Development of LED-based Light sensor drivers

In Figure 4.2 we observe the light sensor and the capacitive plates are utilising PT C and GPIO as the methods of interacting with While PTC and GPIO are not serial communication protocols (typical MCU supported peripherals), they still allow the MCU to communicate with external components and get information similar to what a dedicated sensor would provide.

In the case of the light sensor, as we established in Chapter 3, the red LED acts as a light sensor exploiting the physical properties of a photodiode in a reverse-biased situation [43]. This means applying no voltage to the positive terminal and applying voltage to the negative terminal of the photodiode, in this reverse bias situation, the diode acts as a capacitor charging up. Another property of the photodiode is that it reacts to the amount of light and in reverse biased condition it resembles an optically induced photocurrent. This means that when we stop applying the reverse bias, the photodiode discharges the capacitor charge based on the amount of surrounding light, and by measuring the time it takes to fully discharge we get an indication to the amount of light (hence acting as a light sensor by switching the digital I/O pin connected to the negative photodiode terminal to input), fast discharge time means an increased amount of surrounding light and vice versa. This procedure is what enables an LED to acts as a light sensor and describes how the MCU uses the GPIO pins to get an indication of the light.

The driver of the light sensor is responsible for applying this procedure every time light measurements are needed. This means using an MCU timer to measure the discharge time. Here it is critical in reducing the amount of time it takes the MCU to do one measurement as the entire procedure of measurement is done while the processor is awake and it cannot go to sleep until it finishes the measurement. Also eliminating any processing of the measured time (converting it to light percentage) is critical, and this is done by sending the time measurements to the cloud for processing and appropriate representation.

Building a driver for the light sensor requires proper understanding of HW physical properties and the knowledge of several SW techniques, and in the case of the light sensor, it proved to be easy and straight forward, but it is worth to mention that it is highly platform/design dependent. The driver also shows the exciting aspect of HW and SW working together to achieve a new and different functionality from the one intended. This leads to the reduction of the number of HW components on the board (no need for a dedicated light sensor).

### Development of Mbed OS compatible Touch sensor driver

Capacitive plates are usually connected to their own processing chip, and the measurements information are relayed to the MCU similar to how the temperature and accelerometer operate.

However, in the case of GN, the touch capabilities come from the support of the MCU PTC seen in Figure 4.2. This makes it a target-specific feature and explaining the principle of operation behind it should be enough as PTC is a Microchip HW peripheral that is not supported by other HW manufacturer.

The idea is for any pins supporting PTC is to perform capacitive touch sensor measurements autonomously. The PCB capacitive plates are connected to an analogue charge integrator (built-in HW that facilitates accurate measurements of touch) using the I/O pins.

To explain in a simpler manner, PTC has HW and SW (firmware) that reduces the processing power needed by the MCU for touch detection and measurements and allows any MCU that supports it to get accurate and sensitive measurements without the need for complex processing.
This is done by the integration of individual HW components and channelling all the measurements to a dedicated SW library called Qtouch that is designed to understand what these measurements represent. After doing some complex post-processing and filtering for the measurements, it can accurately make sense of the event that occurred to the touch plates and properly triggers the MCU.

To get an understanding of the PTC measurements (the data in the registers), there is a need to integrate Qtouch library acting as the SW driver for the touch sensor. The library is closed source and only allows for high-level configurations. This provides a set of challenges that are hard to overcome:

1. The benefits of driver modularity and flexibility in data processing are unattainable as the application has a limited set of closed source functionalities that can be invoked, limiting the application capabilities to optimise energy consumption.

2. Integrating the library with Mbed OS proved to be a complex task. Even after successful integration, the application proved complex to develop in comparison to using the Microchip supported platforms such as Atmel Start and MPLAB as they have a set of tools that support Qtouch, unlike Mbed OS.

3. Qtouch code size (cannot be reduced since it is closed source), this acts as a critical constraint for devices with small flash memories.

4. Using unofficial reverse engineered open source libraries available on Github limits

the device capabilities and opens the door for more problems in the future.

The overview of the previously mentioned design tips and challenges goes to show the importance of HW/SW planning and co-design in reducing development times and time to market.

## 4.3 LoRaWAN Applications Design

IoT devices require a SW application that takes advantage of all the HW can offer. Most applications target adding value to the user by configuring the functionality of the HW to serve a specific purpose. This means that most applications focus on solving a specific set of problems, and defining these problems (use cases) is the starting point. Designing and implementing GN applications is critical to show the capabilities of the device, the application is where all the best practices mentioned beforehand take effect, and all the benefits come to the surface.

### 4.3.1 Requirements and Use Cases

The use cases GN can help solve act as requirements for the application layer, and it allows the device to solve some of the problems facing the users. The device ability to implement a numerous number of use cases makes it essential to implement a set of applications that act as a reference for the users. All use cases mentioned in the Chapter 1 share a number of requirements that should be considered when building any application. By defining these requirements, we get an idea of how LoRaWAN devices function and what are the areas of focus that need to be considered when designing the application.

Application requirements (functional and non-functional) that should be considered when using LPWAN communication protocols (LoRaWAN):

- Transmission should target data delivery rather than event delivery.

  If the sensor is used to count the occurrence of certain events (for example, how many times a door was opened), a simple counter should increment and later sent every specific interval (every x hours for example) as there is no need to send a packet per event.

- Data delivery acknowledgements form server side should be reduced/eliminated. They are considered unnecessary and resource-intensive as acknowledgements do not guarantee data delivery since packet loss might still occur even if the node receives an acknowledgement, they should be only be used in certain situations like probing. Usage of streaming codes is recommended instead.
- Application must provide a method for detecting connection instability. An example would be if connection problems are detected, the application would resolve to send periodic short payload probes to check for a functional connection before transmitting huge chunks of data that would be otherwise wasted due to the instability.
- Fallback mechanism is demanded from the application in case of long connection loss (keep collecting data, while still trying to join).

Application requirements (functional and non-functional) that target improving battery life:

- Reduce the number of transmitted bytes (payload size) whenever possible.
- Activate deep sleep for all onboard devices. (Applies for any sleep capable component on the device such as MCU and sensors).
- Use interrupt-driven applications and reduce the polling approach to a minimum.
- Power off inactive sensors/ components once when they are not used.
- Use ADR algorithm when application payload length is indeterministic, and the distance between the gateway and end device location is unknown.

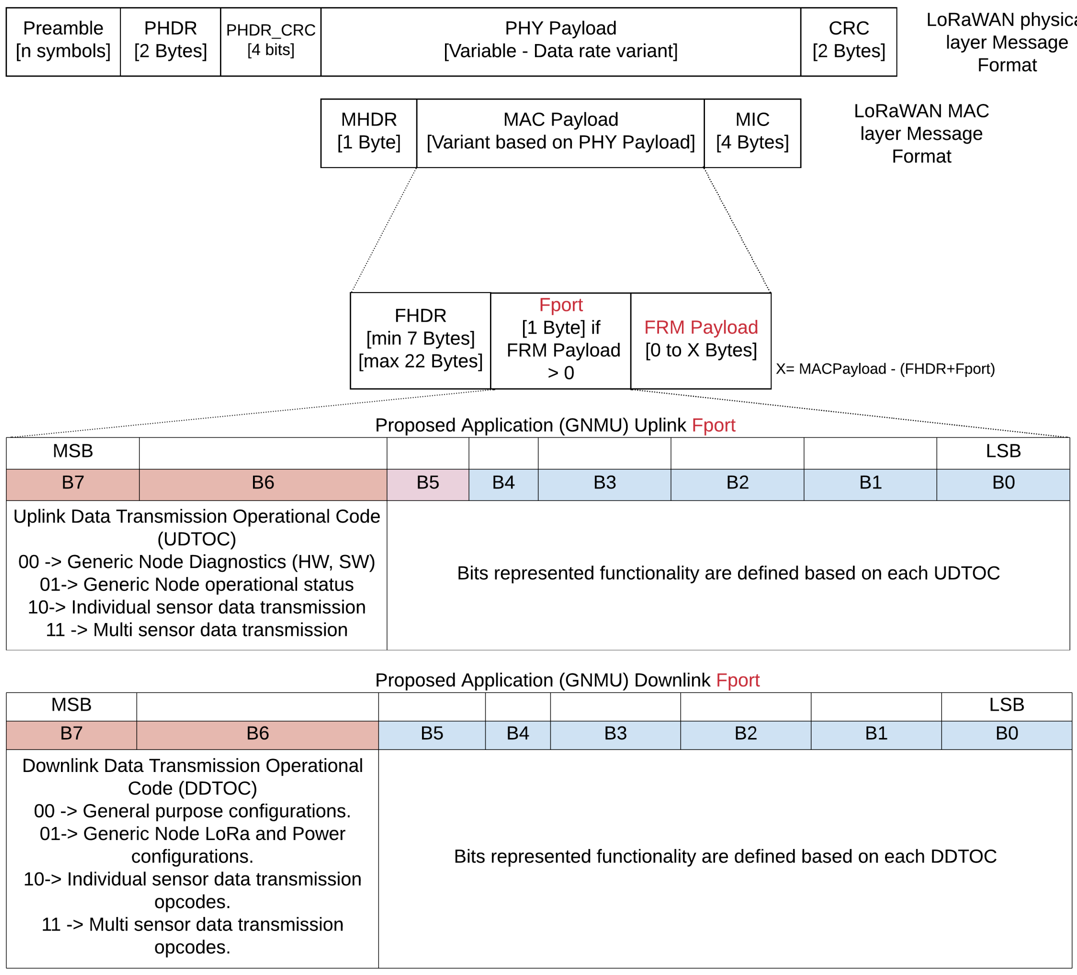
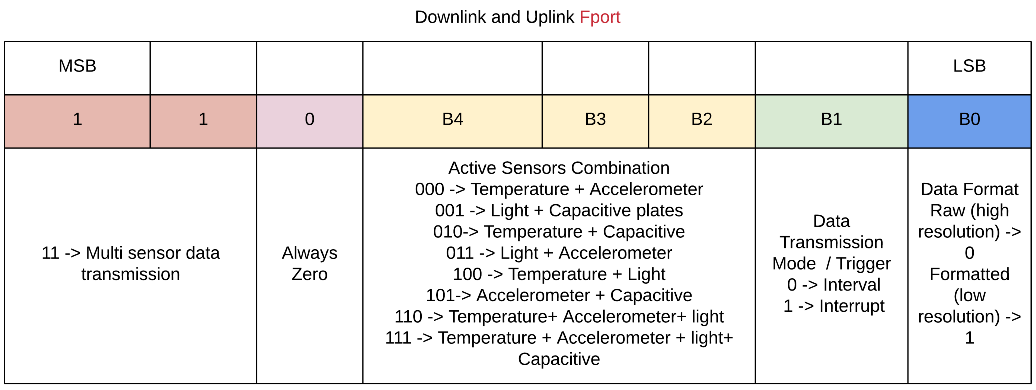

Preamble [n symbols]
PHDR [2 Bytes]
PHDR_CRC [4 bits]
PHY Payload [Variable - Data rate variant]
CRC [2 Bytes]
LoRaWAN physical layer Message Format
MHDR [1 Byte]
MAC Payload [Variant based on PHY Payload]
MIC [4 Bytes]
LoRaWAN MAC layer Message Format
FHDR [min 7 Bytes] [max 22 Bytes]
Fport [1 Byte] if FRM Payload > 0
FRM Payload [0 to X Bytes]
X= MACPayload - (FHDR+Fport)
Proposed Application (GNMU) Uplink Fport
MSB
LSB
B7 B6 B5 B4 B3 B2 B1 B0
Uplink Data Transmission Operational Code (UDTOC)
00 -> Generic Node Diagnostics (HW, SW)
01-> Generic Node operational status
10-> Individual sensor data transmission
11 -> Multi sensor data transmission
Bits represented functionality are defined based on each UDTOC
Proposed Application (GNMU) Downlink Fport
MSB
LSB
B7 B6 B5 B4 B3 B2 B1 B0
Downlink Data Transmission Operational Code (DDTOC)
00 -> General purpose configurations.
01-> Generic Node LoRa and Power configurations.
10-> Individual sensor data transmission opcodes.
11 -> Multi sensor data transmission opcodes.
Bits represented functionality are defined based on each DDTOC

Figure 4.3: Generic Node proposed application Fport design.

- Choose appropriate spreading factor when payload length and device location is known.

### 4.3.2 Proposed Application Design

Applying the requirements and best practices is critical to any application design. Furthermore, any application needs to provide a certain level of flexibility and reconfigurability to allow the user to utilise the device capabilities. Devices such as GN demand even more flexibility; therefore, a special application is designed by taking into account the device architecture. It is important to highlight that any GN user can remove the application and


Downlink and Uplink Fport

| MSB | | | | | | | LSB |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | B4 | B3 | B2 | B1 | B0 |
| 11 -> Multi sensor data transmission | | Always Zero | Active Sensors Combination<br>000 -> Temperature + Accelerometer<br>001 -> Light + Capacitive plates<br>010-> Temperature + Capacitive<br>011 -> Light + Accelerometer<br>100 -> Temperature + Light<br>101-> Accelerometer + Capacitive<br>110 -> Temperature+ Accelerometer+ light<br>111 -> Temperature + Accelerometer + light+ Capacitive | | | Data Transmission Mode / Trigger<br>0 -> Interval<br>1 -> Interrupt | Data Format<br>Raw (high resolution) -> 0<br>Formatted (low resolution) -> 1 |

Figure 4.4: Downlink and uplink operational code example.

replace it with any application that satisfies the use case special requirements, but since the device is targeted as a reference design, there is a need to provide an application that highlight's the node capabilities and possible uses. To put it in easier terms, GN will be shipped with a SW application that will run out of the box and provide a specific set of features. The proposed application is called Genric Node Micro Application (GNMA), and it is designed to take advantage of the underlying LoRaWAN MAC layer. This design is a personal effort (made for GN and thesis) to explore the flexibility of LoRaWAN application layer. Moreover, it shows that proper understanding of the protocol and the device architecture can be extremely beneficial.

As depicted in Figure 4.3, we break down the physical and MAC message formats based on LoRaWAN specification [29]. Fport and FRM Payload are the most relevant in terms of application development. The specifications mandate that Fport (1 byte with allowed values from 1 to 223) is sent whenever there is any data in FRM Payload. In GNMA we define a bitmask for Fport which maps bits to functionalities as observed in Figure 4.4.

The application design treats Fport as an operational code and follows the sequence depicted in 4.5. The sample sequence behaviour is as followed:

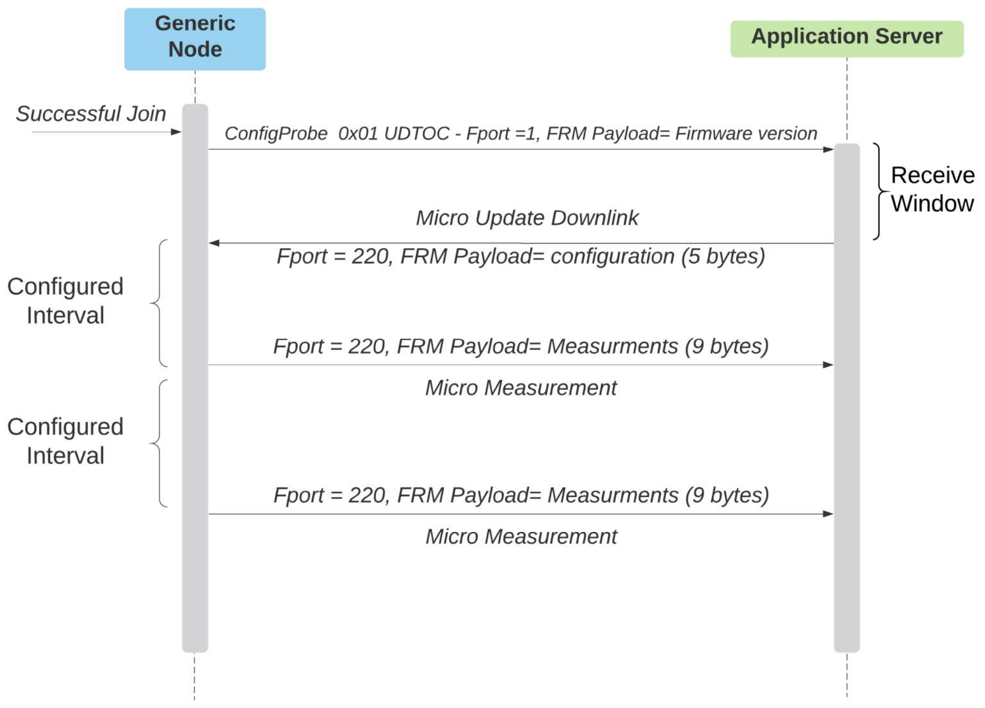

Generic Node
Application Server
Successful Join
ConfigProbe 0x01 UDTOC - Fport =1, FRM Payload= Firmware version
Receive Window
Micro Update Downlink
Fport = 220, FRM Payload= configuration (5 bytes)
Configured Interval
Fport = 220, FRM Payload= Measurments (9 bytes)
Micro Measurement
Configured Interval
Fport = 220, FRM Payload= Measurments (9 bytes)
Micro Measurement

Figure 4.5: Application server and Generic Node communication sequence.

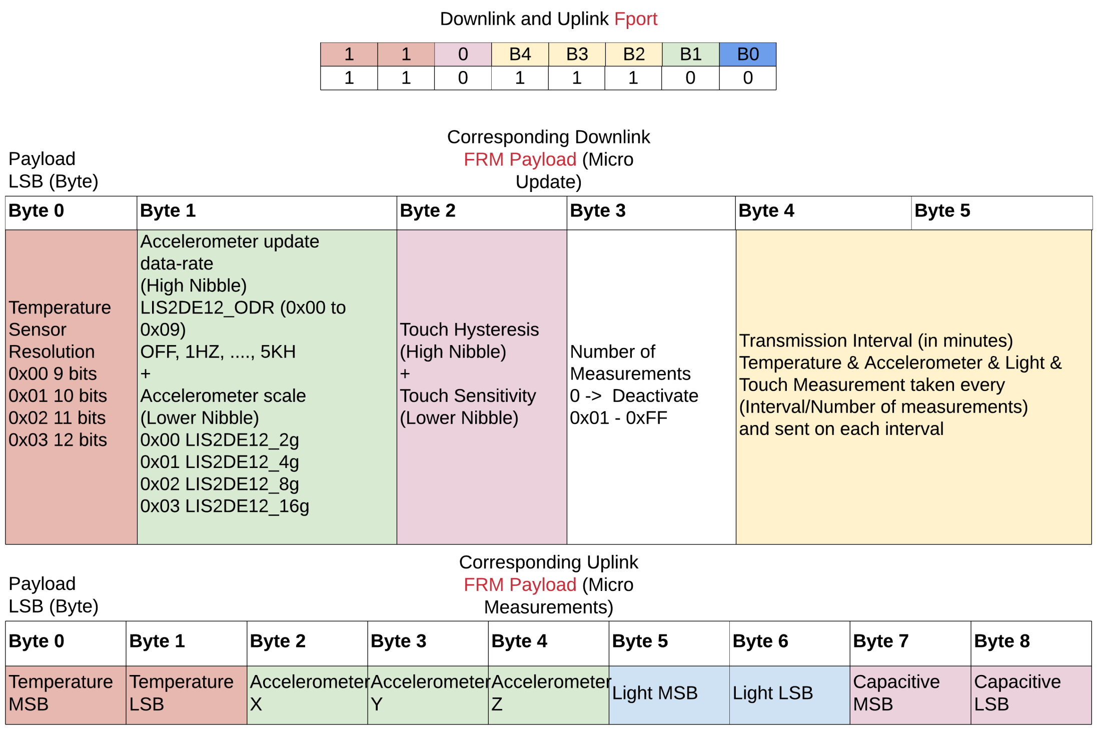
1. The node triggers communication by sending a probe uplink message indicating the need for configurability.

2. Application server is expected schedules a downlink micro update message in any receive window with suitable configurations as depicted in Figure 4.6.

3. GN parses the received FRM Payload according to the specified application Fport and configures the device operation accordingly.

4. If the node is configured correctly, in the next uplink, the node transmits the information using a micro measurement on the same Fport, which acts as an acknowledgement of the micro update.

5. The node firmware concatenate the data in the FRM Payload according to the Fport,



Downlink and Uplink Fport

| 1 | 1 | 0 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |

Payload LSB (Byte)

Corresponding Downlink FRM Payload (Micro Update)

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 |
|---|---|---|---|---|---|
| Temperature Sensor Resolution<br>0x00 9 bits<br>0x01 10 bits<br>0x02 11 bits<br>0x03 12 bits | Accelerometer update data-rate<br>(High Nibble)<br>LIS2DE12_ODR (0x00 to 0x09)<br>OFF, 1HZ, ...., 5KH<br>+<br>Accelerometer scale<br>(Lower Nibble)<br>0x00 LIS2DE12_2g<br>0x01 LIS2DE12_4g<br>0x02 LIS2DE12_8g<br>0x03 LIS2DE12_16g | Touch Hysteresis<br>(High Nibble)<br>+<br>Touch Sensitivity<br>(Lower Nibble) | Number of Measurements<br>0 -> Deactivate<br>0x01 - 0xFF | Transmission Interval (in minutes)<br>Temperature & Accelerometer & Light & Touch Measurement taken every<br>(Interval/Number of measurements)<br>and sent on each interval | |

Payload LSB (Byte)

Corresponding Uplink FRM Payload (Micro Measurements)

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
|---|---|---|---|---|---|---|---|---|
| Temperature MSB | Temperature LSB | Accelerometer X | Accelerometer Y | Accelerometer Z | Light MSB | Light LSB | Capacitive MSB | Capacitive LSB |

Figure 4.6: Sample Fport 220 downlink and uplink specifications.

and transmits it to the application server which in turn can parse the data used according to the predefined uplink payload specifications seen in Figure 4.6.

### 4.3.3 Proposed Application Implementation and Analysis

The proposed application takes into account all the requirements and possible uses cases. Moreover, it pushes the boundary by applying a new design pattern that can be useful if properly integrated. The novelty of the application comes from the introduction of over the air HW configurability seen in the micro-updates as it shows a deep understanding of the device, specifications and SW structure. The application also highlights the need for further research exploration in possible LoRaWAN applications that efficiently utilise the limited data rates.

The design of the application can seem complex to implement at first glance, especially for people who are not aware of the specifications and GN various HW configura-

| Application Metric | Empirical Finding | GN Limit |
|---|---|---|
| Application number of lines | 352 lines | No limit |
| Application layer Flash size (text+data) | 1.411 KB | 256 KB |
| Application layer static RAM utilisation (data+bss) | 4.054 KB | 32 KB |
| Overall Firmware Flash size (text+data) | 114.286 KB | 256 KB |
| Overall Firmware static RAM utilisation (data+bss) | 11.784 KB | 32 KB |

Table 4.1: Sample Application metrics using Fport 220

tion capabilities.

The complexity linearly increases with respect to the number of supported Fports. The application can optionally support up to 223 Fports with a minimum of 4.

Beginners can opt for understanding and activating simple opcodes, on the other hand, businesses who want to take full advantage of what the node can offer are also able to customise how the node application behaves without having to reprogram the device or waste effort designing a new application.

The Fport 220 design depicted in Figure 4.6 was implemented to get an idea of the overhead and the complexity of applying the proposed application. Some data obtained from developing the application is observed in Table 4.1 More detailed information on application flash utilisation can be found in Appendix C Figure C.1.
With Mbed OS and the internally designed and developed drivers, the application implementation complexity is considered average with 352 lines of C++ application code that invokes several underlying drivers. The analysis shows a need for simplifying the design to allow ease of development.

Furthermore, application size is expected to increase with the introduction of more Fports and with 114 KBytes for only one Fport implementation, it is recommended to cautiously observe the text size in further development. From the application server-side, it is important to highlight that between TTN console ease of use and the interesting similarity between JavaScript and C++, it was fairly simple to build a Javascript decoder on the application server side that parses and decodes the received RAW data on the cloud side (refer to Figures C.2, C.3 in Appendix C).

# Chapter 5

# Energy Efficient IoT

Energy efficiency is one of the most researched topics in our current age, and in IoT, the research is more significant than ever. Wireless communication is burdened by several energy constraints, and IoT dependency on wireless technologies transfers the burden and draws more attention to the problem. LoRaWAN devices aim to provide energy-efficient long-range communication by compromising on the data rate. The LoRa modulation explored in Chapter 2 has proven to be capable of delivering promised ranges with minimum energy consumption. LoRaWAN end nodes need to build upon LoRa physical layer energy efficiency by incorporating battery-friendly HW components. GN HW design includes a low power MCU, multiple sensors, external flash and cryptographic co-processor that make it an ideal device to study and evaluate. Several SW components are implemented with a focus on reducing energy consumption. In this chapter, we provide a simple empirical analysis and discuss briefly the indications of the findings.

## 5.1 LoRaWAN Energy Consumption Models

Researchers are interested in evaluating the limits of LoRaWAN [17]. Nevertheless, most work explores and evaluates the scalability aspect, network infrastructure and network deployments. The work done in analysing end devices performance and energy consumption under LoRaWAN is surprisingly small in comparison and typically tends to be application-specific with no emphasise on possible optimisation and best practices. Furthermore, the evaluation is typically done on evaluation boards with commercial of the shelf transceivers making the end device energy consumption evaluations subject to inaccuracies and false representation. The research observations mentioned are obtained by analysing most recent comprehensive and exhaustive surveys such as [33] and exploring various research databases. However, new studies such as [22], [48] and [49] define an extensive energy consumption models, and use the developed framework to empirically evaluate and profile LoRaWAN end devices energy consumption patterns in various scenarios and applications. The work is extremely relevant to any LoRaWAN end device firmware designer and acts as a source of inspiration to the energy consumption analysis done in this chapter. However, the work falls short in multiple areas, some of which are:

- The energy consumption evaluations are done on development boards, which in itself is problematic and provides an incomplete picture of the challenges.
- The boards used for evaluation have high leakage current due to the usage of high voltage (3.3 V), which is typically employed by manufacturers to make it commercially easy to use. Furthermore, the boards use weak, power inefficient and old MCUs such as ATmega328p.
- The analysis uses a maximum of one sensor for an interval-based application, which is not useful for devices with multiple sensors and several operational modes.
- The power optimisation techniques explored are directed towards the transceiver, which is understandable, but the analysis does not explore sensor measurements and the effect of one bad sensor operation model in the expected longevity of a LoRaWAN end device.
- Battery behaviour and limitations are not considered from a practical perspective such as behaviour in various temperatures, or the withdrawal of high current for

extended periods. The shock typically arrives when the device operates differently in real-life scenarios.

## 5.2 Generic Node Energy Consumption

The energy consumption model proposed in [48] is scenario-based, which in turn goes to show that energy profiling of embedded devices is highly application dependent. To reduce the complexity of this chapter, we choose to analyse the simplest GN application possible. The application used in evaluating the power consumption is the one proposed in Chapter 4 with uplinks from Fport 220 observed in Figure 4.6. The use of this application ensures the thesis flow and allows us to proceed with the analysis without having to explain a new application; thus the application flow depicted in Figure 4.5 provides a basic understanding of the application operation.

### 5.2.1 Operational States

The application under analysis goes through different operational states energy-wise. Figure 5.1 visualises the energy-consuming states throughout node lifetime using a simple interval-based application, furthermore, we use the measurements done in [22], [48], [49] and GN measurement to better understand the amount of power and time spent in each state. It is important to highlight that time and power percentages are rough estimations based on the work mentioned, and that it is highly device, application and clock configuration dependant.

### 5.2.2 Generic Node Battery Life Estimation

Equipped with an understanding of the operational states, GN HW and SW, we can evaluate the energy consumption of Fport 220 application and get a battery life estimation.

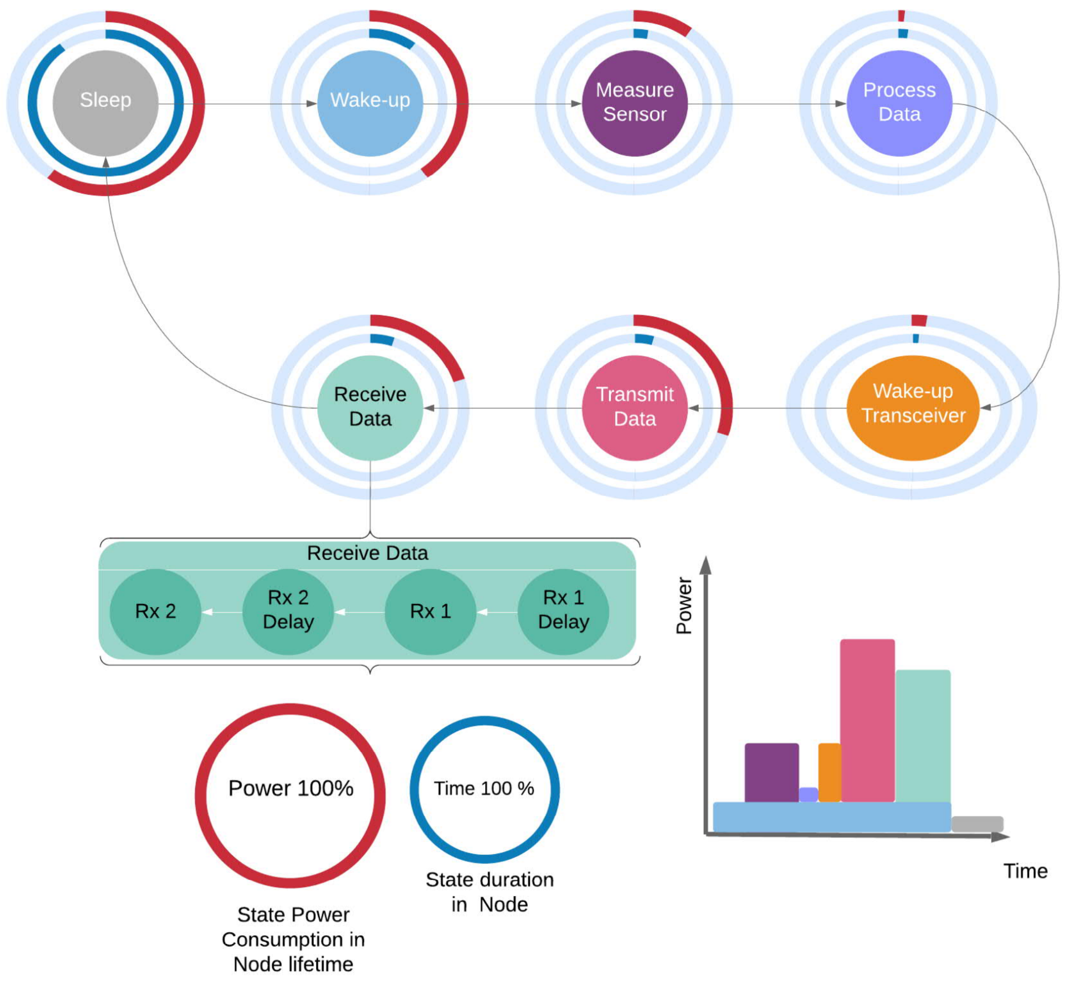


Figure 5.1: Energy state diagram of an interval-based LoRaWAN class A application.

### Battery Estimation Setup

The setup included the following components:

- Generic node.
- Microchip power debugger, see Appendix D Figure D.9.
- Microchip Data Visualizer windows application.
- Single slot battery case for input power and easy switching, see Appendix D Figure D.9.

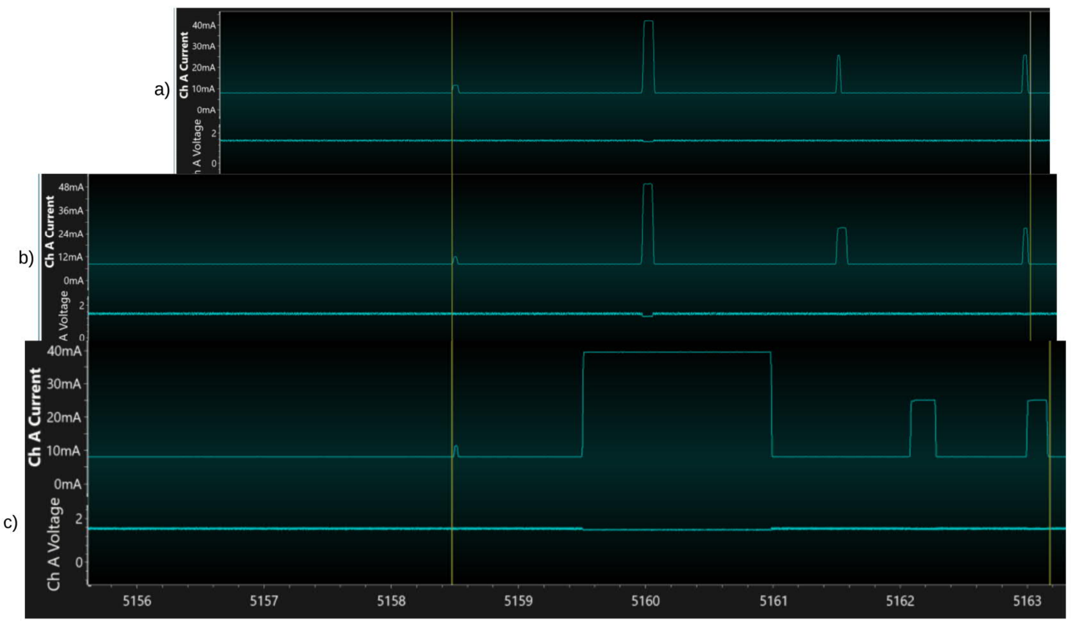


Figure 5.2: Generic Node class A LoRaWAN energy consumption state capture.

Figure (a) represents SF7, BW125 Uplink (22 bytes) with no Downlink.

Figure (b) represents SF7, BW125 Uplink (22 bytes) with Downlink (5 bytes).

Figure (c) represents SF12, BW125 Uplink (22 bytes) with Downlink (5 bytes).

- Single new double-A Kodak Alkaline battery (1.5V, 2300 mAh).

**Application Notes**

- The main MCU clock frequency used for measurements is 48 MHz using the DFLL48M source (uses external 32.768 kHz crystal) [50]. The usage of the highest speed provides a better estimation as it shows the current consumption ceiling.
- Sleep state time can be only calculated by defining how many Uplinks are done by the device, which is application dependent.
- GN sleep state current consumption is being optimised even further, and it is expected to be lower than the value measured. If lower sleep current is achieved, the estimation will require an update to measurements found in this document.

- Wakeup transceiver state is delayed by 1 sec from measure sensors state in all captures; this is to be able to septate the states visually and reduce analysis complexity.

**Measurements Notes**

- The measurements are the most recent results obtained at the time of writing this document, more on the optimisations done will be discussed in the upcoming section 5.3.
- All measurements are done on GN prototype with a real-time optimised application running on Mbed OS, and no datasheet reading was used in the analysis, which is intended to provide accurate representation to the state of the technology reached with GN.
- The device is suspended from sleep in some measurements to be able to verify and validate the measurements multiple times. Nevertheless, a detailed capture of the device sleeping current is provided for reference.

## Measurements Technique and Accuracy

Microchip Data Visualizer[1] application is used in conjunction with Microchip Power Debugger [2] to act as a digital oscilloscope allowing for capturing the energy consumption with a maximum reported accuracy error margin of 3% [51]. All measurements are digitally recorded and attached as image captures for reference in Appendix D. The images are time-based energy measurements of the application behaviour, and all the current and time values are manually extracted from the snapshots. The measurement was repeated four times and the findings obtained are similar with small variations (0.47% due to the need to manually align and adjust the oscilloscope curser).

[1]https://www.microchip.com/mplab/avr-support/data-visualizer

[2]https://www.microchip.com/developmenttools/ProductDetails/atpowerdebugger

| GN State | Payload (bytes) | Time (ms) | Current (mA) | Reference Capture |
|---|---|---|---|---|
| Wake-up | - | 2043 | 8.39 | Appendix D, Figure D.1 |
| Measure Sensor + Process Data | 9 | 23.73 | 11.61 | Appendix D, Figure D.2 |
| Wake-up Transceiver + Transmit Data (SF7, BW125) | 9 | 69.65 | 42.25 | Appendix D, Figure D.3 |
| Wake-up Transceiver + Transmit Data (SF12, BW125) | 9 | 1521 | 38.69 | Appendix D, Figure D.4 |
| Receive Data (Rx1 + Rx2) | 0 | 62.12 | 19.05 | Appendix D, Figure D.5 |
| Receive Data (Rx1 + Rx2) ( Micro Update) | 5 | 94.13 | 24.45 | Appendix D, Figure D.6 |
| Sleep | - | $T_{sleep}$ | 0.1019 | Appendix D Figure D.7 |

Table 5.1: Generic Node operational states energy consumption

### Findings

We measure GN energy consumption and the node behaviour in several LoRaWAN scenarios as depicted in Figure 5.2. Our measurements capture GN energy consumption and the state transitions depicted in Figure 5.1. The Measurements observed in Table 5.1 represent the current consumption and duration of all GN various operational states and scenarios using Fport 220 application specification. $T_{sleep}$ represents all the time the node spends not being in any other state. The duration and the current consumed in sleep sates are considered the key parameters that define the usability of the device as a LoRaWAN device. If the sleep current is low, $T_{sleep}$ will increase accordingly, and the number of entered states in the node lifetime will increase accordingly. $T_{sleep}$ will reduce according to the amount of activity the node undertake. Therefore, firmware designers should optimise with the purpose of increasing $T_{sleep}$ either by reducing the sleep current or reducing the amount of activity undertaken in other states. Understanding LoRaWAN is key to properly evaluating the performance, the Fport 220 specification forces GN to send 9 bytes FRM payload, but it is important to highlight that the LoRaWAN MAC and LoRa physical layer add a minimum overhead of 13 bytes to any uplink. This indicates that the energy consumption represent the transmission of 22 bytes (9 bytes Application +13 MAC and PHY metadata).

**Interval Application Estimated Battery Life**

Using the empirical data retrieved, we can estimate GN application battery lifetime for any interval-based application using the following equations:

- GN Battery Capacity (mAh) $= B$
- GN Requested Uplink Hourly Rate $= H_r$
- State Time (ms) $= T_S$ (available per each state in Table 5.1)
- State Current (mA) $= I_S$ (available per each state in Table 5.1)
- State Time (hour), calculated per each GN State $S$,

$$TH_S = \frac{T_S \times H_r}{1000 \times 60 \times 60}$$

- State Discharge Rate (mAh), calculated per each GN State $S$,

$$DH_S = TH_S \times I_S$$

- Sleep State Time (hour)

$$TH_{Sleep} = 1 - \sum_{S=Wake-up}^{S=ReceiveData} TH_S$$

- Sleep State Discharge Rate (mAh)

$$DH_{Sleep} = TH_{Sleep} \times I_{Sleep}$$

- Application Discharge rate (mAh)

$$DH_{Application} = \sum_{S=Sleep}^{S=ReceiveData} DH_S$$

- Application battery Lifetime (hour)

$$\frac{B}{DH_{Application}}$$

| GN Application (22 bytes uplink every 15 minutes) | Battery Discharge Rate (mAh) | Battery (years) |
|---|---|---|
| SF7 and no downlinks | 0.1255870518 | 2.090 |
| SF12 and no downlinks | 0.1875391462 | 1.4 |
| SF7 with 5 bytes downlink (micro updates) | 0.1268257526 | 2.07 |
| Sf12 with 5 bytes downlink (micro updates) | 0.188942172 | 1.389 |

Table 5.2: Generic Node battery life with 100uA sleep current and heavy load application

Application transmission interval $H_r$ is usually sent as part of the micro update configuration downlink observed previously in Figure 4.5. As depicted in Table 5.2, which assumes the application server requests the node to transmit 4 uplinks per hour in a micro update using the proposed application architecture (Fport 220). GN is estimated to approximately survive on one battery for 2 years using SF7 and 1.4 years using SF12 with measurements of all sensors and transmitting 22 bytes (9 Application + 13 LoRaWAN MAC metadata) in each transmission. The detailed calculations are available in Appendix D, Figure D.8.

## 5.3 Generic Node Power Optimisations

LoRaWAN advertised battery efficiency is a product of the physical layer energy efficient modulation and the MAC layer reduced listening time with Class A devices. Several power optimisation techniques can be implemented in the HW and SW to allow for an enhanced battery survivability. On HW level, the optimisations exploit techniques that reduce the standby leakage current which ultimately target the reduction of sleep state current. However, on the SW side, the optimisation space expands even further by targeting the reduction of active states current and duration.

GN firmware is responsible for employing several power-saving techniques based on an understanding of the HW design. Furthermore, the energy measurements showed in Table 5.1 give us an understanding of the areas where further optimisation can be beneficial. In the upcoming sections, we explore the power optimisation techniques employed

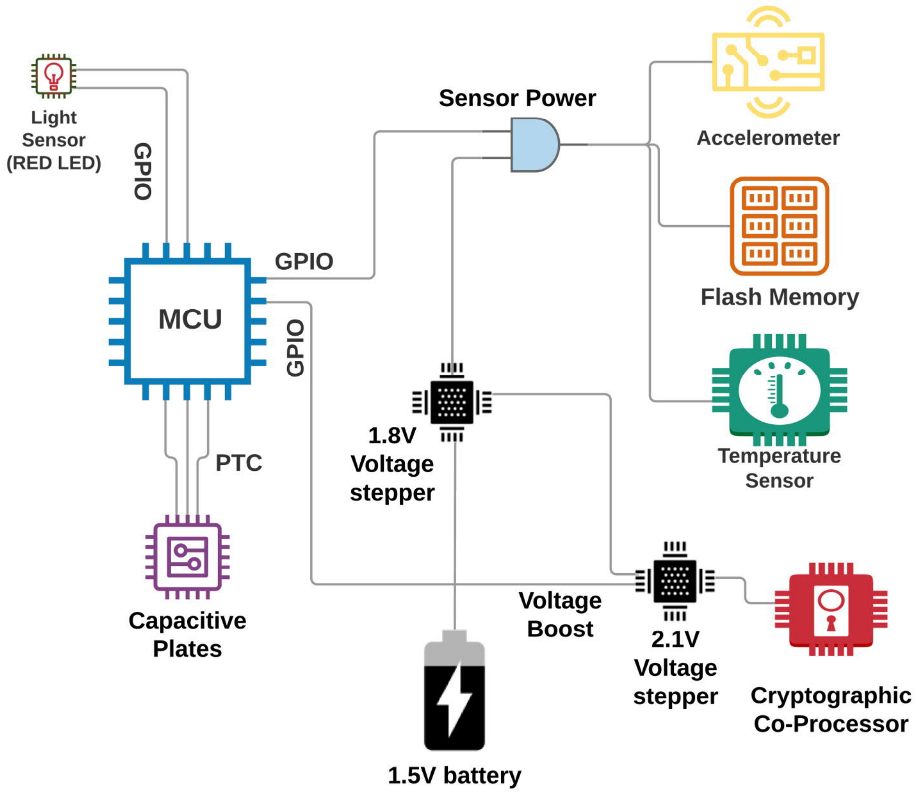


Figure 5.3: Simplified form of Generic Node power circuit.

in GN HW and SW. Moreover, we analyse the impact of the employed optimisations on battery life.

### 5.3.1 Energy Efficient Hardware

The device HW design defines the energy efficiency profile and establishes how all components interact. Therefore, firmware designers are always encouraged to explore the device HW design and get an understanding of the capabilities and limitations. Some GN HW design decisions have proven to be indispensable for a LoRaWAN device.

#### Components Power Gating

The concept of power gating is important for any multi-sensor IoT device and in the case of LoRaWAN [52], the importance increases due to the reliance on sleep for longer periods. GN employes power gating control for all external components on the board as depicted in Figure 5.3.

**Results after utilising Power Gating:**

GN design allows the MCU to power ON and OFF all the external components in runtime, which allowed the node to reach the measured 101uA sleep current with all of these components on board. One issue that arises with GN design is the inability to power off individual sensors. If the external flash needs to be used, the temperature sensor and accelerometer will need to be powered as well, which is not ideal for some use-cases.

### Power Efficient Components

All components found in GN board target power efficiency. Each device supports entering sleep modes if they are powered on and not used. As an example, the MCU can request the temperature sensor to enter shutdown mode to save energy while the sensor is powered and not being utilised.

**Results after utilising Sensor Sleep:**

When this mode was activated in the wake-up state, the current consumed by the state dropped by 0.1 mA, Appendix D Figures D.10 and D.11. To put it in context, this minor change increases GN Fport220 application operation by 36 hours with only one sensor configuration. The effect of using such an approach increases as more sensors enter sleep modes in the wake-up state. If applications target longer battery life, then this feature has proven to be beneficial for multi-sensor LoRaWAN devices.

Moreover, the MCU supports multiple different sleeping modes that can make GN survive longer if properly utilised [50]. The current withdrawn by the MCU is typically high, and proper configuration depending on the application will allow the device to last for longer periods. It's worthy of mentioning that MCU deep sleep modes enforce some tradeoffs to achieve lower sleep currents. As an example, one of the deep sleep configurations will give up RAM retention, this is not ideal for some applications and must be considered deeply by the designer before choosing the MCU. In the case of GN, simple interval applications can give up RAM retention for better battery life as sensors data are measured and updated in each interval. GN operational states depicted in Figure 5.1 stands to make use of better sleep current reduction over RAM retention.

### 5.3.2 Energy Efficient Software

On the SW side, there are multiple areas that can benefit from further optimisation.

#### Low Power Ticker

Real-time operating systems such as Mbed OS, FreeRtos and many others allow users to attach the kernel scheduler to a low power ticker, thus, effectively activating a feature referred to as Tickless Scheduling. The feature uses a low power timer interrupt as a ticker that awakens the system instead of actively waking up the system periodically to check the queued tasks. This allows the system to enter the sleep state for longer periods in between tasks, which in turn reduces the current of all other operational states.

**Results after activating Tickless Scheduling:**

Activating this feature for GN had interesting results. The wake-up state current reduced from 11.86 mA to 4.897 mA, Appendix D Figures D.12 and D.13. This means an improvement of 41.63% which is a significant impact that will increase battery life by 50,5 days which is remarkable as the feature only required developing the low power ticker driver for the target and porting it to Mbed OS.

#### Optimised SW Drivers

GN SW development was done from the beginning with a focus on energy efficiency, as indicated in Chapter 4. All drivers developed were used in the assessments observed in this chapter. However, some optimisations had a better impact on the battery than expected:

1. Raw data uplinks instead of sending formatted data

   Using the temperature sensor as an example. Inefficient bare-metal drivers forcibly convert any sensor reading to a float value as part of the driver. From LoRaWAN perspective, this means a 2-byte measurement is now a 4-byte one with the same exact information, which is an overhead of 100%. Furthermore, the node LoRaWAN stack will need to process the float and break it into 4 bytes to be able to transmit it. The overhead continuous in this scenario, as the application server, need to use the uplinked 4 bytes and decode it back to a float again.

   **Key takeaway:** It is not power-efficient to convert the measurement obtained from the sensor at the node side. This is due to the fact that the application server will need to process the data anyway (Appendix D Figure C.2). The idea is to offload any computation from the node side to the cloud side.

2. Efficient serial bus communication.

   All sensor readings translate to increased duration of measure sensor and wake-up states, which leads to consuming more energy. Reducing the SPI or I2C bus traffic by optimising the driver has proved to be useful in reducing energy consumption. At the start of development, the bus traffic was high due to the use of inefficient SW taking up to 1.5 seconds to retrieve driver information(Appendix D Figure D.14). The optimisation reduced the measure time to 0.023 ms using the same clock (Appendix D Figure D.2).

It is necessary to highlight that all optimisations done are not device-specific and can be employed by any LoRaWAN device that targets energy efficiency. Furthermore, all the mentioned points are backed by real-time measurements (before and after improvements).

## 5.4 Generic Node Limitations

The process of developing a new product such as GN highlights some problems and device limitations that need to be considered and preferably improved in future developments. GN HW design done by TWTG [3] targets providing an energy efficient multi-sensor LoRaWAN device that caters to the power demands of various components. The device operates on a single, double-A battery with a voltage range of 1.5 to 1.7V. The use of one battery complicates the design and increases the complexity of the power delivery system, as observed in Figure 5.3. To comply with the licensing agreement and protect TWTG and TTN intellectual property rights we reframe from providing details about the components used in the power delivery system, but thankfully due to the open-source SW nature of the device, we can abstractly discuss HW complexity of the power delivery system.

### Complex Power Delivery System

Figure 5.3 shows a simplified block diagram of GN circuit. The complexity of the power delivery system arises from the different voltage operational domains needed by the external components. It is worthy to note the use of two different voltage steppers to power the MCU, sensors, flash and the cryptographic co-processor. The MCU operates on 1.8V similarly to all the sensors and the flash, however, the cryptographic co-processor requires 2.1V to operate. The voltage variation increases the design complexity as the entire system runs on a single 1.5V battery.
As a result, component failures are hard to debug and analyse.
Furthermore, profiling the node energy consumption and estimating battery life becomes highly application dependant.

[3]www.twtg.io

### Battery Discharge Rate

GN power source is a single double-A battery that can be bought off the shelf from any store, and while this is an attractive feature from a user standpoint, it makes predicting the device survivability in harsh environments extremely battery dependent [53] and further complicates any power profiling endeavour. This can be counteracted by making the node report the runtime battery voltage to predict the device survivability but requires a sophisticated battery estimation model implementation on the application server-side.

### Limited Transmission Power and Reset Issues

The Semtech SX1276 LoRa transceiver [54] can transmit in two power modes, the lower power mode (when active) withdraws a peak of 42mA in SF7 transmissions, and the high power mode consumes a 96 mA peak in SF7 transmissions Appendix D Figure D.15. Unlike the low power mode which works well, the higher power transmission mode tends to cause GN to enter continuous reset state when the device is powered by batteries that are low on charge (rated at 20%). This is caused by a sudden voltage drop across several power delivery components rendering them to OFF state and ultimately forcing the MCU to reset.

The issue is still being investigated and can only be concerned to a cretin prototype patch, but it can also be an indication to the node incapability to use high power transmission modes. This can be a limitation of the power delivery system complex design.

# Chapter 6

# Advanced IoT

The last chapter establishes the device potential and proves with concrete findings the device capability to deliver on LoRaWAN promises and vision. In the final chapter, we explore how the device pushes the boundaries of what is possible in IoT. Furthermore, we highlight future research topics that fit the device profile and provide a conclusion to work done in the document.

## 6.1 Secure End Nodes

In Chapter 2, we explore how LoRaWAN is designed with a focus on security. GN HW features a cryptographic co-processor chip (ATECC608A) [55] that adds a new layer of security.

### 6.1.1 Problem Description

As we previously established, LoRaWAN allows devices to communicate with the network server after a successful activation either through ABP or OTAA. In both cases, the node and the network server secure the radio communication by using several 128-bit AES keys which are used to encrypt and decrypt the communication.

For LoRaWAN 1.1, the OTAA process (analysed in Chapter 2 Figure 2.3) relies on two main root keys (AppKey and NwkKey). The root keys are used to create session keys for each joining request. The handling of the root keys is the issue here since the keys must be stored in the end node as well as the network/join servers. To further illustrate, any mishandling of the keys by any involved party will leave all the device manufactured borne to hijacking, impersonation and sniffing vulnerabilities [41].

### 6.1.2 Device Operation

1. ATECC608A securely stores a unique set of root keys for each device at the chip manufacturing facility.
2. Microchip transfers device-specific keys to a trusted LoRaWAN join Server.
3. On activation, the ATECC608A calculates and provides the session keys based on requests over the I2C bus (Previously highlighted in Chapter 4 Figure4.2).

### 6.1.3 Solution Benefits

ATECC608A acts as a secure vault for LoRaWAN root keys. The keys enter the vault when the device is produced and never leave again. Microchip handles the delivery of the keys to the trusted join server based on the buyer preference, therefore, reducing production keys management issues. Microchip integrates the silicon with a service to add a new layer of security to LoRaWAN end devices.

## 6.2 LoRaWAN and Firmware Updates Over The Air

LoRaWAN devices such as GN operate for several years and will require a firmware update to deploy firmware improvements and comply with the latest requirements and safety standers. LoRaWAN FUOTA specifications was released recently in 2018 [56].

The recent release show how updates can be handled by different device classes. The reduced data rates of LoRaWAN and the duty cycle regulations make FUOTA a challenge that requires deep analysis. At the time of writing, GN FUOTA demo application is still under development, and it provides some insights that might be beneficial for future LoRaWAN devices that plan to support FUOTA.

### 6.2.1 Device Support

- In theory, any device can support FUOTA, but in practicality, it typically requires an external flash to store the firmware update (delta or full) as the internal flash is typically consumed by the bootloader and the application.
- The updates require a cretin amount of run time RAM to be able to check the validity and authenticity of the received data before storing it in external flash. This typically causes memory pressure that leads to stack overflows and faults in devices low on memory.
- The use of cryptographic co-processors in packet verification can reduce memory pressure in the update process due to the reduced use of SW crypto libraries.

### 6.2.2 Challenges

- High energy consumption

  FUOTA updates cause significant current discharges over long periods as the device temporarily becomes a class C device, entering listening state all the time until the update is finished. In case of GN, that means increasing wake-up current to 24.45 mA for the entire update period.
- Large application sizes

  Smallest application size reached in GN development was 16KByte which means that if a full application update is required, this means splitting the application into

80 fragments of 200 bytes at SF7, assuming no packet loss, the update will have a significant impact on the battery. Simple interval application reached sizes of 114 KBytes, which indicates the need for delta updates.

- Packet loss and self noise

  We established in Chapter 2 how LoRa modulation is borne to interference from nearby LoRa devices. The specification [56] attempts to solve the issue by forcing the devices being updated to stop sending uplinks in the duration of FUOTA. However, nearby devices that are not being updated will be a source of interference, not to mention the noise from the updating gateways downlinks on nearby devices and gateways.

# Chapter 7

# Conclusion and Future Research

Embedded hardware designers are pushing the limits of what is possible in the field of IoT. Firmware developers need to take note of what the hardware is aiming to achieve and push the boundaries even further with an innovative exploration of what is possible. The opportunity to develop the firmware of a next-generation LoRaWAN end-node has proven to be an exciting and extremely challenging task.

In the document, we empirically proved the capabilities of LoRaWAN devices to achieve long battery life (2 years) with multiple on-board sensors. We highlighted the sensing limitation incurred by the use of multiple sensors and showed possible alternatives. Firmware development time was reduced by the use of open-source platforms such as Mbed OS. Energy-efficient design can be applied to the hardware and software sides to achieve optimal operation. In Chapter 4, we propose a novel way of configuring GN in runtime via small micro-updates that allow the device to behave according to the needs of the user without the need to do FUOTA. The empirical analysis done in Chapter 5 shows that industrial IOT nodes are capable of surviving for longer periods. The research proves that LoRaWAN end nodes can achieve low battery operation with the assistance of the hardware and firmware designs. It also showed how the proposed application improves usability with minimum impact on the device battery. For devices that are faced with FUOTA challenges, the micro update scheme can be the solution. Further research will focus on the energy and traffic overhead of using micro-updates in comparison to using FUOTA.

## 7.1 Future Research

GN can be an asset in evaluating LoRaWAN. Several other research areas can benefit from using the device such in areas such as energy estimation models, machine learning sensor calibrations and predictive maintenance. However, research on optimizing LoRaWAN end devices can definitely benefit from an open-source device such as GN. A number of research topics can be explored even further with the assistance of such device.

### 7.1.1 Lossless Compression for Improving Performance

IoT Devices with sensors can typically benefit from compressing the measured data and sending it [57], but with LoRaWAN low data rates, lossless compression [58] stands to make an even more significant impact. GN can be used to evaluate how the compression allows devices to send twice the amount of data with minimum energy overhead. The research needs to evaluate the energy consumed in compression against the energy consumed in transmission for the same payload.

### 7.1.2 Edge and Cloud Computing

GN can collect data and transmit all the information without any processing, or it can save the data internally and evaluate it at the edge side. The research area is still being explored [13], but no analysis was done to compare the energy efficiency from LPWAN point of view, in particular, the energy cost of processing the data on edge devices in comparison to the energy cost of sending large amounts of data to the cloud. It should be considered as an analysis of energy wasted by edge processor against energy wasted by multiple transmissions to the cloud.

# References

[1] H. Hejazi, H. Rajab, T. Cinkler, and L. Lengyel, “Survey of platforms for massive iot”, in *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*, Jan. 2018, pp. 1–8. DOI: 10.1109/FIOT.2018.8325598.

[2] U. Raza, P. Kulkarni, and M. Sooriyabandara, “Low power wide area networks: An overview”, *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 855–873, Secondquarter 2017, ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2652320.

[3] N. Naik, “Lpwan technologies for iot systems: Choice between ultra narrow band and spread spectrum”, in *2018 IEEE International Systems Engineering Symposium (ISSE)*, Oct. 2018, pp. 1–8. DOI: 10.1109/SysEng.2018.8544414.

[4] N. Blenn and F. A. Kuipers, “Lorawan in the wild: Measurements from the things network”, *CoRR*, vol. abs/1706.03086, 2017. arXiv: 1706.03086. [Online]. Available: http://arxiv.org/abs/1706.03086.

[5] J. Kramer, J. M. van der Werf, J. Stokking, and M. Ruiz, “A blockchain-based micro economy platform for distributed infrastructure initiatives”, in *2018 IEEE International Conference on Software Architecture (ICSA)*, Apr. 2018, pp. 11–1109. DOI: 10.1109/ICSA.2018.00010.

[6] M. Y. Thu, W. Htun, Y. L. Aung, P. E. E. Shwe, and N. M. Tun, “Smart air quality monitoring system with lorawan”, in *2018 IEEE International Conference on*

*Internet of Things and Intelligence System (IOTAIS)*, Nov. 2018, pp. 10–15. DOI: 10.1109/IOTAIS.2018.8600904.

[7] C. Schmitt, J. Meier, M. Diez, and B. Stiller, “Otiot — a browser-based object tracking solution for the internet of things”, in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Feb. 2018, pp. 445–451. DOI: 10.1109/WF-IoT.2018.8355157.

[8] M. T. Buyukakkaslar, M. A. Erturk, M. A. Aydin, and L. Vollero, “Lorawan as an e-health communication technology”, in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, Jul. 2017, pp. 310–313. DOI: 10.1109/COMPSAC.2017.162.

[9] D. Davcev, K. Mitreski, S. Trajkovic, V. Nikolovski, and N. Koteli, “Iot agriculture system based on lorawan”, in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, Jun. 2018, pp. 1–4. DOI: 10.1109/WFCS.2018.8402368.

[10] R. Kromes, A. Russo, B. Miramond, and F. Verdier, “Energy consumption minimization on lorawan sensor network by using an artificial neural network based application”, in *2019 IEEE Sensors Applications Symposium (SAS)*, Mar. 2019, pp. 1–6. DOI: 10.1109/SAS.2019.8705992.

[11] F. Torres, J. Soriano, and G. Riva, “First steps in the development of a lorawan test-bench”, in *2018 Ninth Argentine Symposium and Conference on Embedded Systems (CASE)*, Aug. 2018, pp. 7–12. DOI: 10.23919/SASE-CASE.2018.8542160.

[12] P. A. Barro, M. Zennaro, and E. Pietrosemoli, “Tltn – the local things network: On the design of a lorawan gateway with autonomous servers for disconnected communities”, in *2019 Wireless Days (WD)*, Apr. 2019, pp. 1–4. DOI: 10.1109/WD.2019.8734239.

[13] H. Truong, "Enabling edge analytics of iot data: The case of lorawan", in *2018 Global Internet of Things Summit (GIoTS)*, Jun. 2018, pp. 1–6. DOI: 10.1109/GIOTS.2018.8534429.

[14] J. Navarro-Ortiz, S. Sendra, P. Ameigeiras, and J. M. Lopez-Soler, "Integration of lorawan and 4g/5g for the industrial internet of things", *IEEE Communications Magazine*, vol. 56, no. 2, pp. 60–67, Feb. 2018, ISSN: 0163-6804. DOI: 10.1109/MCOM.2018.1700625.

[15] ThethingsIndustries. (2019). Generic node sensing abilities, [Online]. Available: https://www.genericnode.com (visited on 08/24/2019).

[16] L. Vangelista, "Frequency shift chirp modulation: The lora modulation", *IEEE Signal Processing Letters*, vol. 24, no. 12, pp. 1818–1821, Dec. 2017, ISSN: 1070-9908. DOI: 10.1109/LSP.2017.2762960.

[17] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan", *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, Sep. 2017, ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600613.

[18] K. Mikhaylov, J. Petäjäjärvi, and J. Janhunen, "On lorawan scalability: Empirical evaluation of susceptibility to inter-network interference", in *2017 European Conference on Networks and Communications (EuCNC)*, Jun. 2017, pp. 1–6. DOI: 10.1109/EuCNC.2017.7980757.

[19] Semtech. (2015). Lora modulation basics an1200.22, [Online]. Available: https://www.semtech.com/uploads/documents/an1200.22.pdf (visited on 07/06/2019).

[20] A. Ikpehai, B. Adebisi, K. M. Rabie, K. Anoh, R. E. Ande, M. Hammoudeh, H. Gacanin, and U. M. Mbanaso, "Low-power wide area network technologies for internet-of-things: A comparative review", *IEEE Internet of Things Journal*, vol. 6,

no. 2, pp. 2225–2240, Apr. 2019, ISSN: 2327-4662. DOI: 10.1109/JIOT.2018.2883728.

[21] Thethingsnetwork. (2019). Lorawan distance world record broken, twice. 766 km (476 miles) using 25mw transmission power, [Online]. Available: https://www.thethingsnetwork.org/article/lorawan-distance-world-record (visited on 07/30/2019).

[22] L. Casals, B. Mir, R. Vidal, and C. Gomez, "Modeling the energy performance of lorawan", *Sensors*, vol. 17, no. 10, 2017, ISSN: 1424-8220. DOI: 10.3390/s17102364. [Online]. Available: https://www.mdpi.com/1424-8220/17/10/2364.

[23] M. Babaee and S. Sharifian, "An improved spread factor assignment method for large-scale lorawan deployments in iot", in *2018 4th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, Dec. 2018, pp. 183–187. DOI: 10.1109/ICSPIS.2018.8700554.

[24] T. Elshabrawy and J. Robert, "Analysis of ber and coverage performance of lora modulation under same spreading factor interference", in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1–6. DOI: 10.1109/PIMRC.2018.8581011.

[25] F. Cuomo, J. C. C. Gámez, A. Maurizio, L. Scipione, M. Campo, A. Caponi, G. Bianchi, G. Rossini, and P. Pisani, "Towards traffic-oriented spreading factor allocations in lorawan systems", in *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, Jun. 2018, pp. 1–8. DOI: 10.23919/MedHocNet.2018.8407091.

[26] M. El-Aasser, T. Elshabrawy, and M. Ashour, "Joint spreading factor and coding rate assignment in lorawan networks", in *2018 IEEE Global Conference on Inter-*

*net of Things (GCIoT)*, Dec. 2018, pp. 1–7. DOI: 10.1109/GCIoT.2018.8620147.

[27] A. Farhad, D. Kim, P. Sthapit, and J. Pyun, "Interference-aware spreading factor assignment scheme for the massive lorawan network", in *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, Jan. 2019, pp. 1–2. DOI: 10.23919/ELINFOCOM.2019.8706416.

[28] L. Alliance. (2017). Lorawan® regional parameters v1.1rb, [Online]. Available: https://lora-alliance.org/sites/default/files/2018-04/lorawantm_regional_parameters_v1.1rb_-_final.pdf.

[29] L. Alliance. (2017). Lorawan® specification v1.1, [Online]. Available: https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf.

[30] A. Lavric and A. I. Petrariu, "Lorawan communication protocol: The new era of iot", in *2018 International Conference on Development and Application Systems (DAS)*, May 2018, pp. 74–77. DOI: 10.1109/DAAS.2018.8396074.

[31] P. S. Cheong, J. Bergs, C. Hawinkel, and J. Famaey, "Comparison of lorawan classes and their power consumption", in *2017 IEEE Symposium on Communications and Vehicular Technology (SCVT)*, Nov. 2017, pp. 1–6. DOI: 10.1109/SCVT.2017.8240313.

[32] J. Lee, W. Jeong, and B. Choi, "A scheduling algorithm for improving scalability of lorawan", in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct. 2018, pp. 1383–1388. DOI: 10.1109/ICTC.2018.8539392.

[33] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of lorawan for iot: From technology to application", *Sensors*, vol. 18, no. 11, 2018, ISSN: 1424-

8220. DOI: 10.3390/s18113995. [Online]. Available: https://www.mdpi.com/1424-8220/18/11/3995.

[34] K. Q. Abdelfadeel, V. Cionca, and D. Pesch, “Fair adaptive data rate allocation and power control in lorawan”, in *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Jun. 2018, pp. 14–15. DOI: 10.1109/WoWMoM.2018.8449737.

[35] S. Kim and Y. Yoo, “Contention-aware adaptive data rate for throughput optimization in lorawan”, *Sensors*, vol. 18, no. 6, 2018, ISSN: 1424-8220. DOI: 10.3390/s18061716. [Online]. Available: https://www.mdpi.com/1424-8220/18/6/1716.

[36] J. M. Marais, R. Malekian, and A. M. Abu-Mahfouz, “Evaluating the lorawan protocol using a permanent outdoor testbed”, *IEEE Sensors Journal*, vol. 19, no. 12, pp. 4726–4733, Jun. 2019, ISSN: 1530-437X. DOI: 10.1109/JSEN.2019.2900735.

[37] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, “Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations”, *IEEE Communications Surveys Tutorials*, pp. 1–1, 2019, ISSN: 1553-877X. DOI: 10.1109/COMST.2019.2910750.

[38] J. de Carvalho Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic, and A. L. L. Aquino, “Lorawan — a low power wan protocol for internet of things: A review and opportunities”, in *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, Jul. 2017, pp. 1–6.

[39] N. Hayati, K. Ramli, M. Suryanegara, and Y. Suryanto, “Potential development of aes 128-bit key generation for lorawan security”, in *2019 2nd International Conference on Communication Engineering and Technology (ICCET)*, Apr. 2019, pp. 57–61. DOI: 10.1109/ICCET.2019.8726884.

[40] R. Sanchez-Iborra, J. Sánchez-Gómez, S. Pérez, P. J. Fernández, J. Santa, J. L. Hernández-Ramos, and A. F. Skarmeta, "Enhancing lorawan security through a lightweight and authenticated key management approach", *Sensors*, vol. 18, no. 6, 2018, ISSN: 1424-8220. DOI: `10.3390/s18061833`. [Online]. Available: `https://www.mdpi.com/1424-8220/18/6/1833`.

[41] B. Oniga, V. Dadarlat, E. De Poorter, and A. Munteanu, "Analysis, design and implementation of secure lorawan sensor networks", in *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, Sep. 2017, pp. 421–428. DOI: `10.1109/ICCP.2017.8117042`.

[42] X. Yang, E. Karampatzakis, C. Doerr, and F. Kuipers, "Security vulnerabilities in lorawan", in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, Apr. 2018, pp. 129–140. DOI: `10.1109/IoTDI.2018.00022`.

[43] K.-T. Lau, S. Baldwin, M. O'Toole, R. Shepherd, W. J. Yerazunis, S. Izuo, S. Ueyama, and D. Diamond, "A low-cost optical sensing device based on paired emitter–detector light emitting diodes.", *Analytica Chimica Acta*, vol. 557, no. 1/2, pp. 111–116, 2006, ISSN: 00032670. [Online]. Available: `http://search.ebscohost.com.ezproxy.utu.fi/login.aspx?direct=true&db=aph&AN=19466369&site=ehost-live`.

[44] P. S. Nasirabadi, M. Jabbari, and J. H. Hattel, "Numerical simulation of transient moisture and temperature distribution in polycarbonate and aluminum electronic enclosures", in *2016 17th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE)*, Apr. 2016, pp. 1–6. DOI: `10.1109/EuroSimE.2016.7463382`.

[45] J. Rivera, M. Carrillo, M. Chacón, G. Herrera, and G. Bojorquez, "Self-calibration and optimal response in intelligent sensors design based on artificial neural net-

works", *Sensors*, vol. 7, no. 8, pp. 1509–1529, 2007, ISSN: 1424-8220. DOI: `10.3390/s7081509`. [Online]. Available: `https://www.mdpi.com/1424-8220/7/8/1509`.

[46] Microchip. (2014). At30ts750a datasheet, [Online]. Available: `http://ww1.microchip.com/downloads/en/devicedoc/Atmel-8855-DTS-AT30TS750A-Datasheet.pdf` (visited on 07/06/2019).

[47] Microchip. (2017). Sst26wf080b/080ba datasheet, [Online]. Available: `http://ww1.microchip.com/downloads/en/DeviceDoc/20005283C.pdf` (visited on 07/06/2019).

[48] T. Bouguera, J.-F. Diouris, J.-J. Chaillout, R. Jaouadi, and G. Andrieux, "Energy consumption model for sensor nodes based on lora and lorawan", *Sensors*, vol. 18, no. 7, 2018, ISSN: 1424-8220. DOI: `10.3390/s18072104`. [Online]. Available: `https://www.mdpi.com/1424-8220/18/7/2104`.

[49] B. Thoen, G. Callebaut, G. Leenders, and S. Wielandt, "A deployable lpwan platform for low-cost and energy-constrained iot applications", *Sensors*, vol. 19, no. 3, 2019, ISSN: 1424-8220. DOI: `10.3390/s19030585`. [Online]. Available: `https://www.mdpi.com/1424-8220/19/3/585`.

[50] Microchip. (2017). Sam r34/r35 low power lora sub-ghz sip datasheet, [Online]. Available: `http://ww1.microchip.com/downloads/en/DeviceDoc/SAMR34-R35-Low-Power-LoRa-Sub-GHz-SiP-Data-Sheet-DS70005356B.pdf` (visited on 07/06/2019).

[51] Microchip. (2016). Power debugger user guide, [Online]. Available: `http://ww1.microchip.com/downloads/en/DeviceDoc/Power-Debugger_UserGuide.pdf` (visited on 07/06/2019).

[52] M. Hayashikoshi, H. Noda, H. Kawai, and H. Kondo, "Low-power multi-sensor system with normally-off sensing technology for iot applications", in *2016 Inter-*

*national SoC Design Conference (ISOCC)*, Oct. 2016, pp. 195–196. DOI: 10.1109/ISOCC.2016.7799854.

[53] A. B. da Cunha, B. R. de Almeida, and D. C. da SilvaJr., "Remaining capacity measurement and analysis of alkaline batteries for wireless sensor nodes", *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 6, pp. 1816–1822, Jun. 2009, ISSN: 0018-9456. DOI: 10.1109/TIM.2009.2013660.

[54] Semtech. (2016). Sx1276/77/78/79 datasheet, [Online]. Available: https://www.mouser.com/ds/2/761/sx1276-1278113.pdf (visited on 07/06/2019).

[55] Microchip. (2017). Atecc608a datasheet, [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/40001977A.pdf (visited on 07/06/2019).

[56] L. Alliance. (2018). Lorawan remote multicast setup specification v1.0.0, [Online]. Available: https://lora-alliance.org/sites/default/files/2018-09/remote_multicast_setup_v1.0.0.pdf.

[57] T. Adegbija, A. Rogacs, C. Patel, and A. Gordon-Ross, "Microprocessor optimizations for the internet of things: A survey", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 7–20, Jan. 2018, ISSN: 0278-0070. DOI: 10.1109/TCAD.2017.2717782.

[58] F. Marcelloni and M. Vecchio, "A simple algorithm for data compression in wireless sensor networks", *IEEE Communications Letters*, vol. 12, no. 6, pp. 411–413, Jun. 2008, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2008.080300.

# Appendix A

# LoRaWAN

## LoRa link budget sample calculation:

- Tx Power: 14 dBm
- Bandwidth: 125 kHz = 10log(125000) = 51
- SNR: -20 dB
- Noise Figure: 6 dB
- Sensitivity= -174 + 10log10(BW) + NF + SNR -174 + 51 + 6 – 20 = -137 dBm
- Link Budget= Tx power - Sensitivity = 14 + 137 = 151 dB

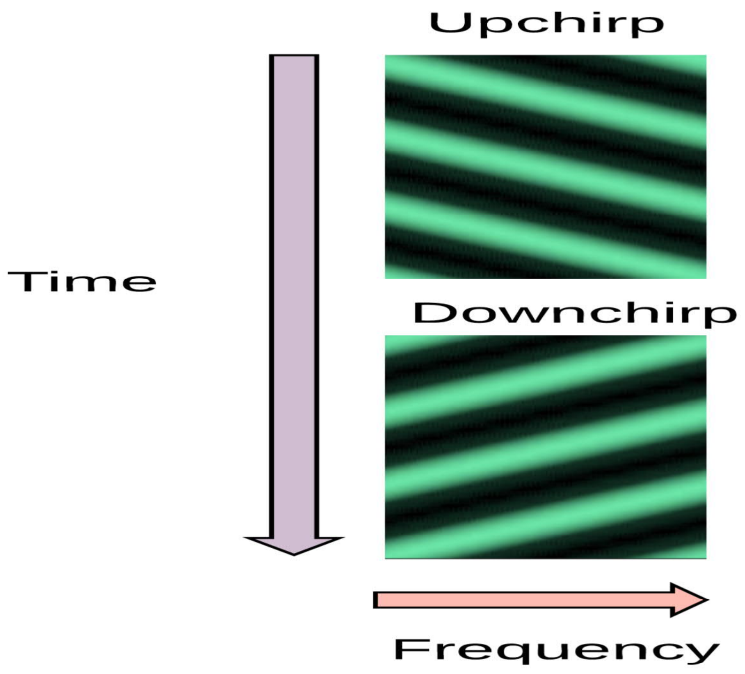


Figure A.1: Chirp pulse form with time and frequency change.

# Appendix B

# HW Design

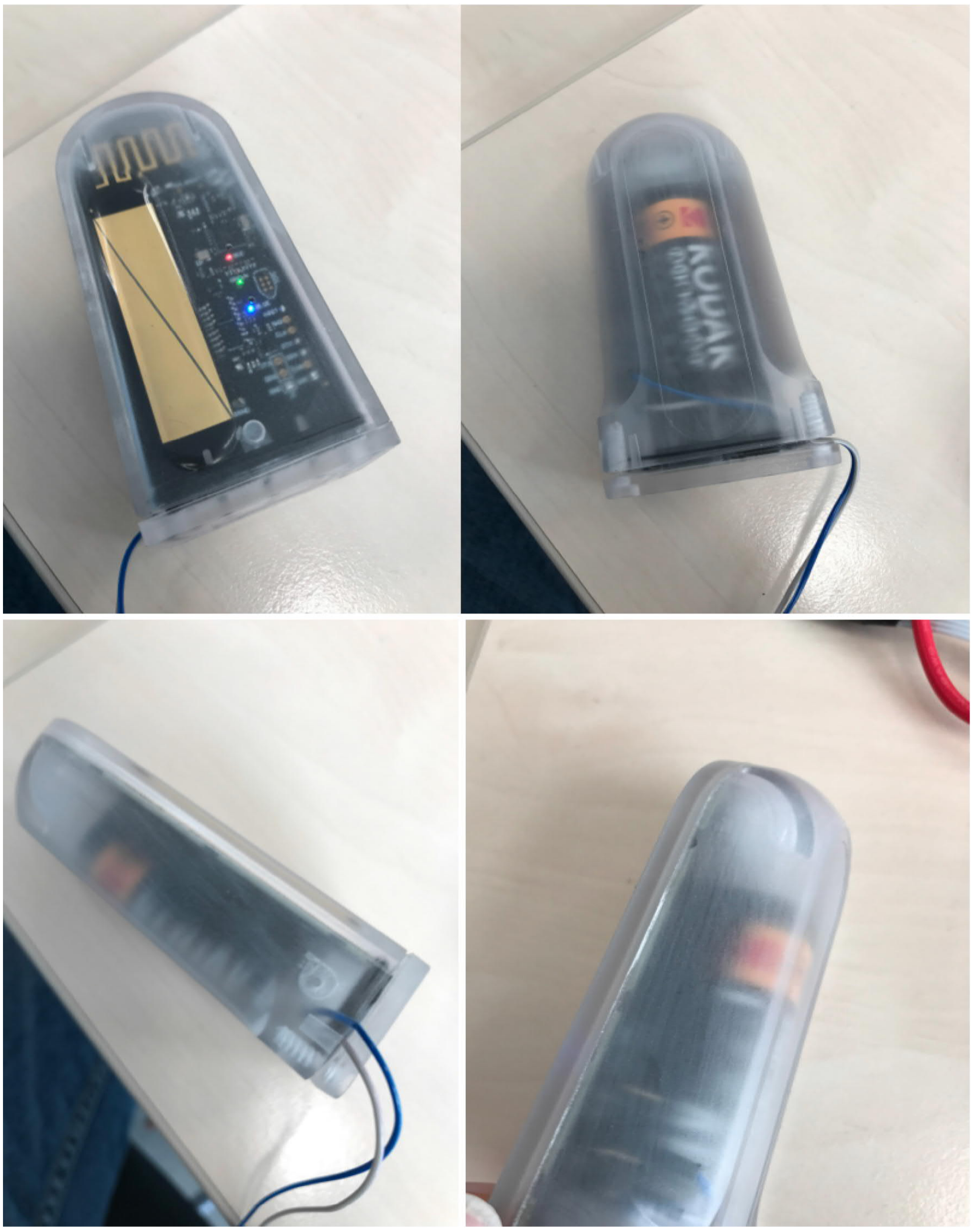
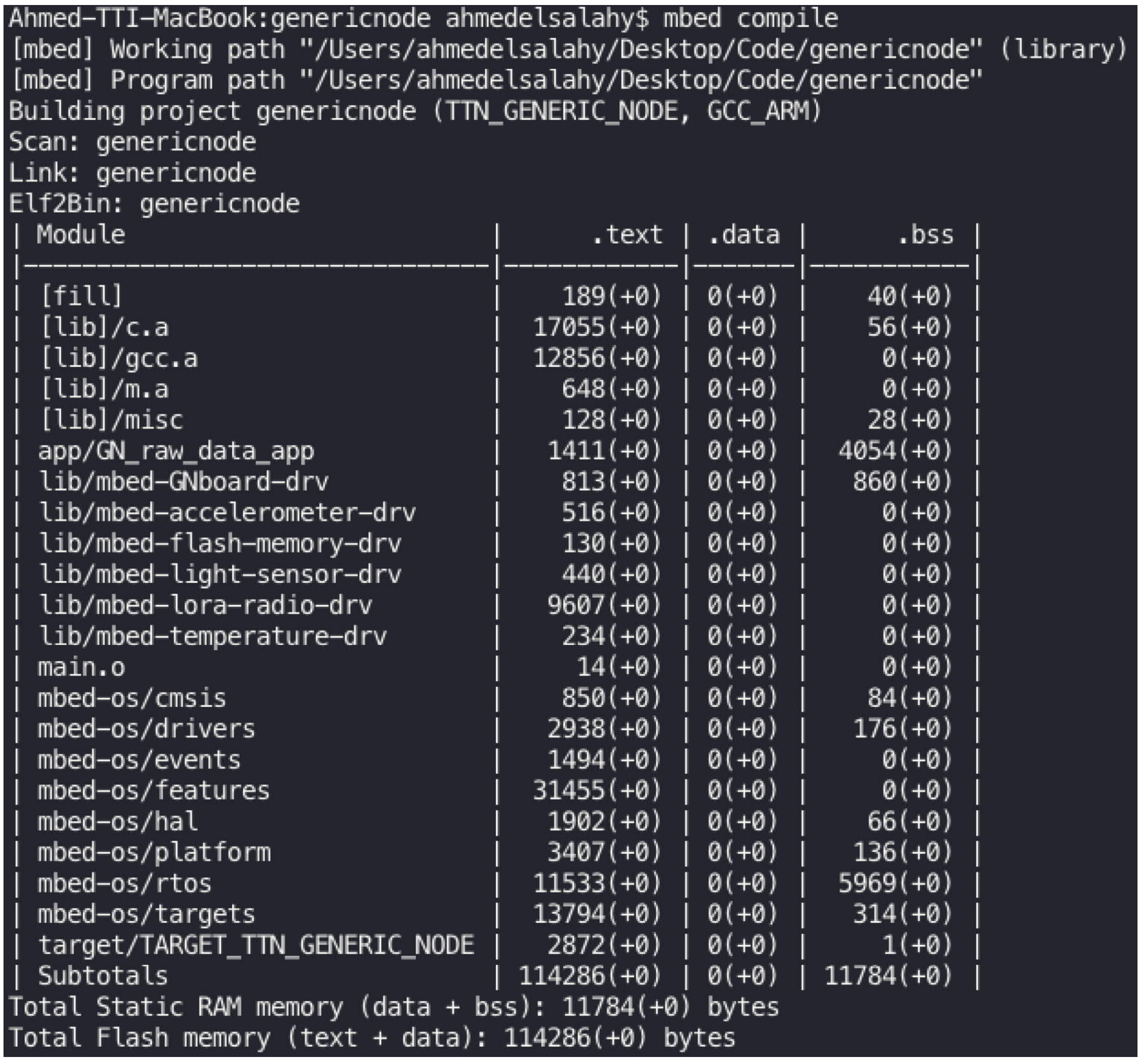


Figure B.1: Multi-angle view of Generic Node enclosure.

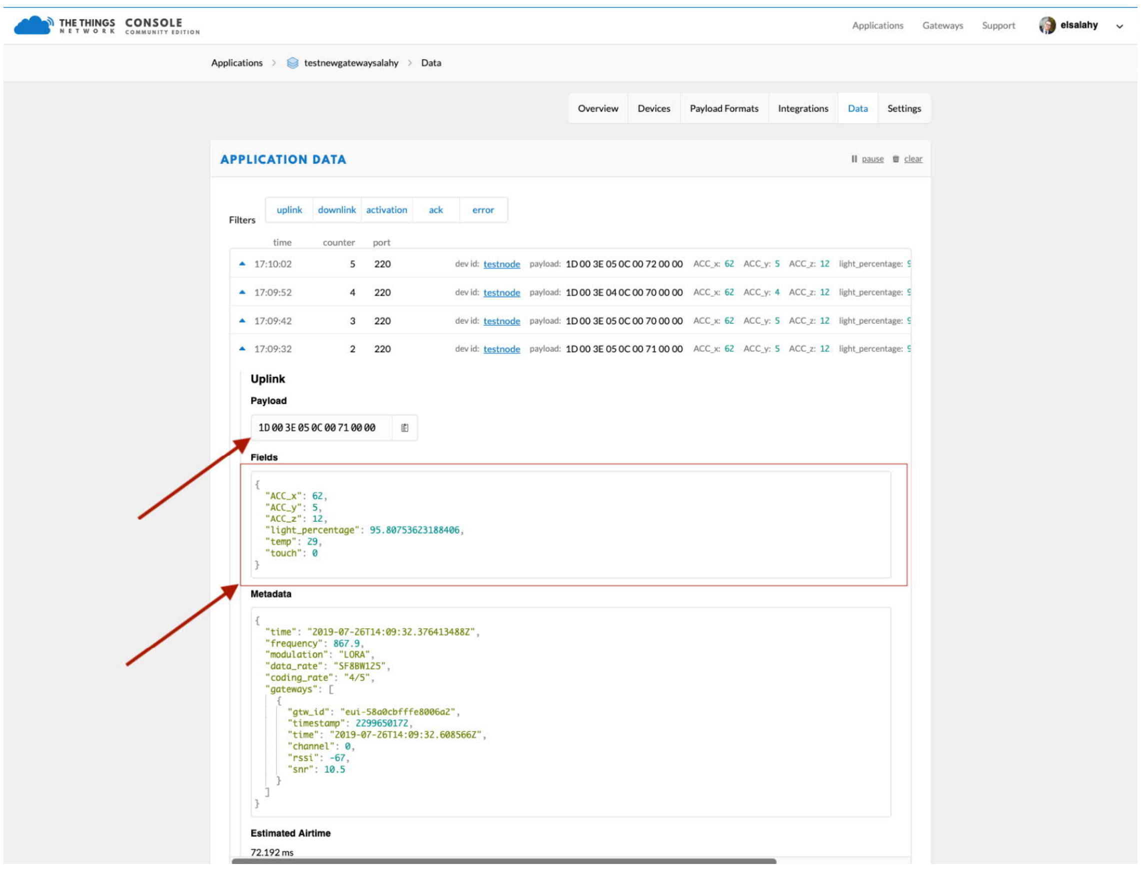
# Appendix C

# SW Design



```
Ahmed-TTI-MacBook:genericnode ahmedelsalahy$ mbed compile
[mbed] Working path "/Users/ahmedelsalahy/Desktop/Code/genericnode" (library)
[mbed] Program path "/Users/ahmedelsalahy/Desktop/Code/genericnode"
Building project genericnode (TTN_GENERIC_NODE, GCC_ARM)
Scan: genericnode
Link: genericnode
Elf2Bin: genericnode
| Module                          |      .text | .data |      .bss |
|---------------------------------|------------|-------|-----------|
| [fill]                          |    189(+0) | 0(+0) |    40(+0) |
| [lib]/c.a                       |  17055(+0) | 0(+0) |    56(+0) |
| [lib]/gcc.a                     |  12856(+0) | 0(+0) |     0(+0) |
| [lib]/m.a                       |    648(+0) | 0(+0) |     0(+0) |
| [lib]/misc                      |    128(+0) | 0(+0) |    28(+0) |
| app/GN_raw_data_app             |   1411(+0) | 0(+0) |  4054(+0) |
| lib/mbed-GNboard-drv            |    813(+0) | 0(+0) |   860(+0) |
| lib/mbed-accelerometer-drv      |    516(+0) | 0(+0) |     0(+0) |
| lib/mbed-flash-memory-drv       |    130(+0) | 0(+0) |     0(+0) |
| lib/mbed-light-sensor-drv       |    440(+0) | 0(+0) |     0(+0) |
| lib/mbed-lora-radio-drv         |   9607(+0) | 0(+0) |     0(+0) |
| lib/mbed-temperature-drv        |    234(+0) | 0(+0) |     0(+0) |
| main.o                          |     14(+0) | 0(+0) |     0(+0) |
| mbed-os/cmsis                   |    850(+0) | 0(+0) |    84(+0) |
| mbed-os/drivers                 |   2938(+0) | 0(+0) |   176(+0) |
| mbed-os/events                  |   1494(+0) | 0(+0) |     0(+0) |
| mbed-os/features                |  31455(+0) | 0(+0) |     0(+0) |
| mbed-os/hal                     |   1902(+0) | 0(+0) |    66(+0) |
| mbed-os/platform                |   3407(+0) | 0(+0) |   136(+0) |
| mbed-os/rtos                    |  11533(+0) | 0(+0) |  5969(+0) |
| mbed-os/targets                 |  13794(+0) | 0(+0) |   314(+0) |
| target/TARGET_TTN_GENERIC_NODE  |   2872(+0) | 0(+0) |     1(+0) |
| Subtotals                       | 114286(+0) | 0(+0) | 11784(+0) |
Total Static RAM memory (data + bss): 11784(+0) bytes
Total Flash memory (text + data): 114286(+0) bytes
```

Figure C.1: LoRaWAN raw data transmission application build metrics with Fport 220.



```
time      counter  port
17:10:02  5        220    dev id: testnode  payload: 1D 00 3E 05 0C 00 72 00 00  ACC_x: 62  ACC_y: 5  ACC_z: 12  light_percentage:
17:09:52  4        220    dev id: testnode  payload: 1D 00 3E 04 0C 00 70 00 00  ACC_x: 62  ACC_y: 4  ACC_z: 12  light_percentage:
17:09:42  3        220    dev id: testnode  payload: 1D 00 3E 05 0C 00 70 00 00  ACC_x: 62  ACC_y: 5  ACC_z: 12  light_percentage:
17:09:32  2        220    dev id: testnode  payload: 1D 00 3E 05 0C 00 71 00 00  ACC_x: 62  ACC_y: 5  ACC_z: 12  light_percentage:
```

**Uplink**

**Payload**

```
1D 00 3E 05 0C 00 71 00 00
```

**Fields**

```
{
  "ACC_x": 62,
  "ACC_y": 5,
  "ACC_z": 12,
  "light_percentage": 95.80753623188406,
  "temp": 29,
  "touch": 0
}
```

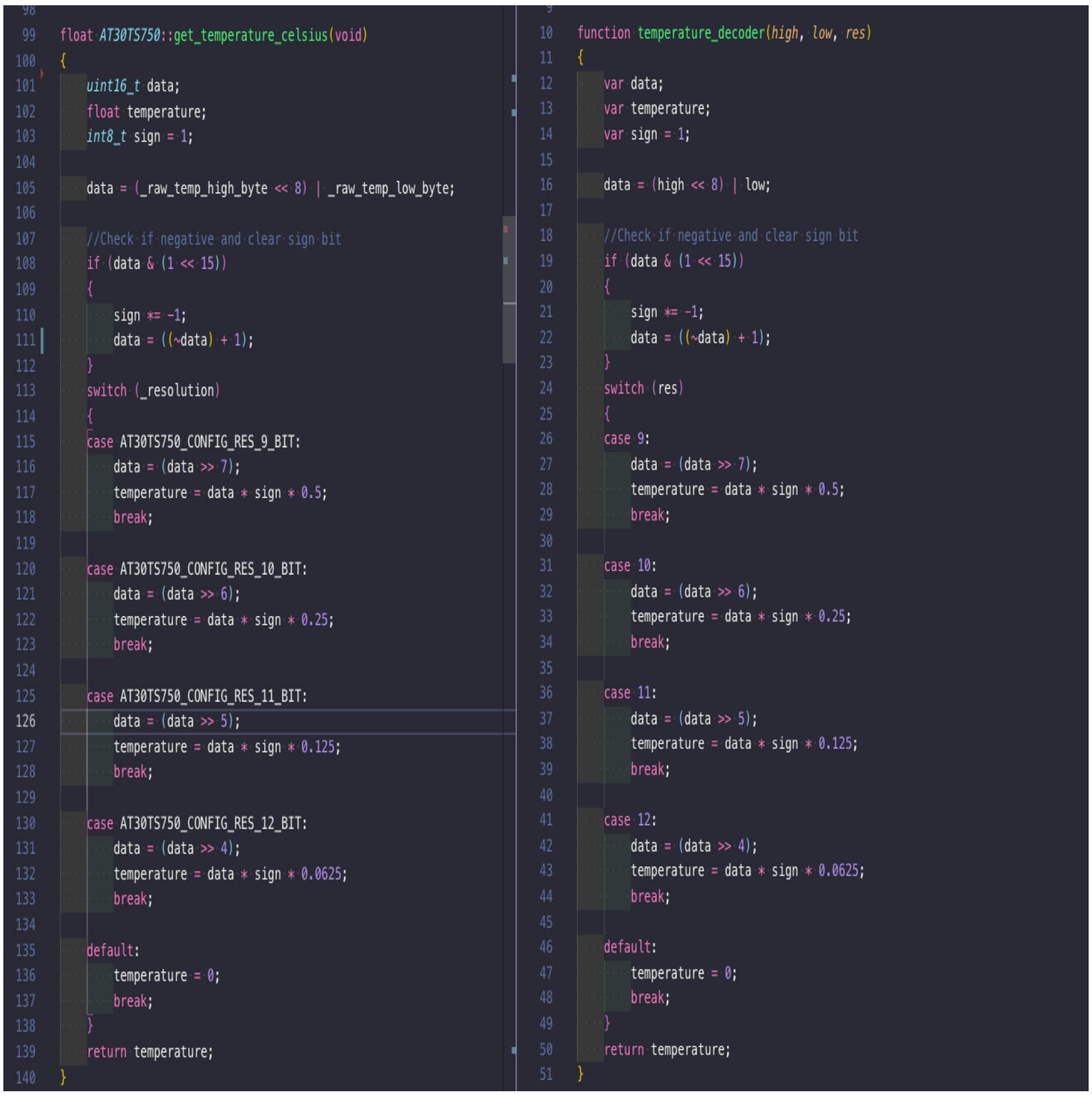
**Metadata**

```
{
  "time": "2019-07-26T14:09:32.376413488Z",
  "frequency": 867.9,
  "modulation": "LORA",
  "data_rate": "SF8BW125",
  "coding_rate": "4/5",
  "gateways": [
    {
      "gtw_id": "eui-58a0cbfffe8006a2",
      "timestamp": 2299650172,
      "time": "2019-07-26T14:09:32.608566Z",
      "channel": 0,
      "rssi": -67,
      "snr": 10.5
    }
  ]
}
```

**Estimated Airtime**

72.192 ms

Figure C.2: Application server side decoding raw payload uplink.



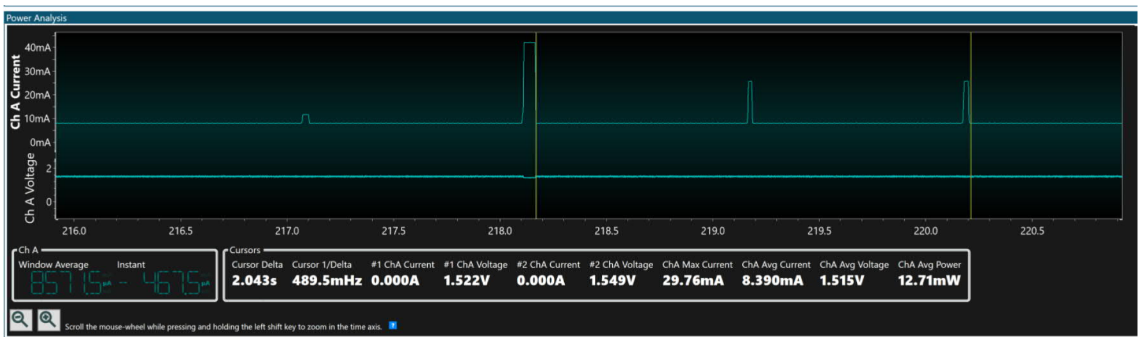
```cpp
float AT30TS750::get_temperature_celsius(void)
{
    uint16_t data;
    float temperature;
    int8_t sign = 1;

    data = (_raw_temp_high_byte << 8) | _raw_temp_low_byte;

    //Check if negative and clear sign bit
    if (data & (1 << 15))
    {
        sign *= -1;
        data = ((~data) + 1);
    }
    switch (_resolution)
    {
    case AT30TS750_CONFIG_RES_9_BIT:
        data = (data >> 7);
        temperature = data * sign * 0.5;
        break;

    case AT30TS750_CONFIG_RES_10_BIT:
        data = (data >> 6);
        temperature = data * sign * 0.25;
        break;

    case AT30TS750_CONFIG_RES_11_BIT:
        data = (data >> 5);
        temperature = data * sign * 0.125;
        break;

    case AT30TS750_CONFIG_RES_12_BIT:
        data = (data >> 4);
        temperature = data * sign * 0.0625;
        break;

    default:
        temperature = 0;
        break;
    }
    return temperature;
}
```

```javascript
function temperature_decoder(high, low, res)
{
    var data;
    var temperature;
    var sign = 1;

    data = (high << 8) | low;

    //Check if negative and clear sign bit
    if (data & (1 << 15))
    {
        sign *= -1;
        data = ((~data) + 1);
    }
    switch (res)
    {
    case 9:
        data = (data >> 7);
        temperature = data * sign * 0.5;
        break;

    case 10:
        data = (data >> 6);
        temperature = data * sign * 0.25;
        break;

    case 11:
        data = (data >> 5);
        temperature = data * sign * 0.125;
        break;

    case 12:
        data = (data >> 4);
        temperature = data * sign * 0.0625;
        break;

    default:
        temperature = 0;
        break;
    }
    return temperature;
}
```

Figure C.3: Raw temperature sensor measurement decoding function with C++ on GN firmware compared against similar functionality Java Script code for application server side.

# Appendix D

# Energy



Ch A

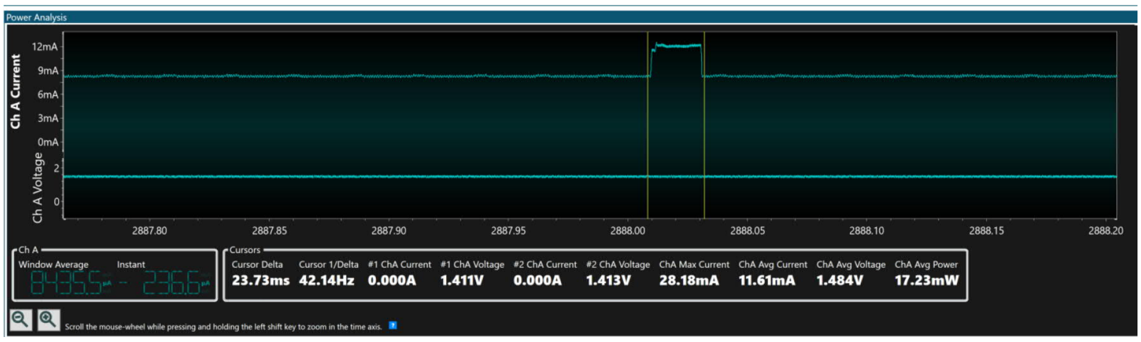
| Window Average | Instant |
|---|---|
| 8571.5µA | 4675µA |

Cursors

| Cursor Delta | Cursor 1/Delta | #1 ChA Current | #1 ChA Voltage | #2 ChA Current | #2 ChA Voltage | ChA Max Current | ChA Avg Current | ChA Avg Voltage | ChA Avg Power |
|---|---|---|---|---|---|---|---|---|---|
| 2.043s | 489.5mHz | 0.000A | 1.522V | 0.000A | 1.549V | 29.76mA | 8.390mA | 1.515V | 12.71mW |

Scroll the mouse-wheel while pressing and holding the left shift key to zoom in the time axis.

Figure D.1: GN Wake-up state Fport 220.



Ch A

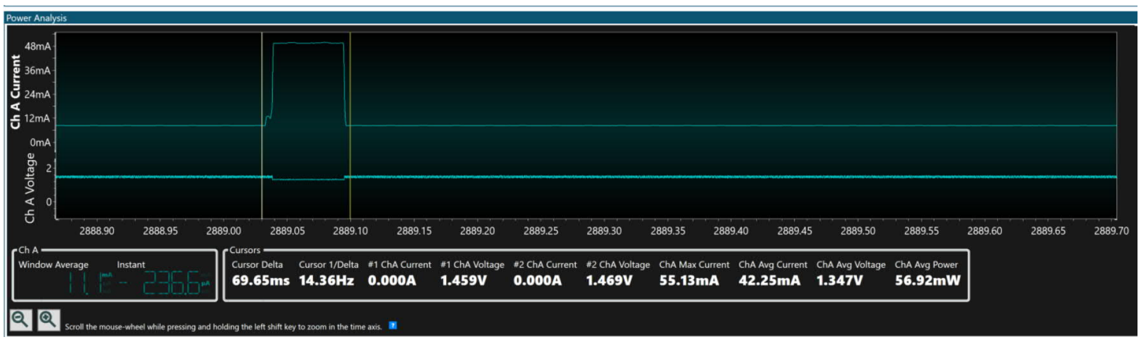
| Window Average | Instant |
|---|---|
| 8435.5µA | 236.6µA |

Cursors

| Cursor Delta | Cursor 1/Delta | #1 ChA Current | #1 ChA Voltage | #2 ChA Current | #2 ChA Voltage | ChA Max Current | ChA Avg Current | ChA Avg Voltage | ChA Avg Power |
|---|---|---|---|---|---|---|---|---|---|
| 23.73ms | 42.14Hz | 0.000A | 1.411V | 0.000A | 1.413V | 28.18mA | 11.61mA | 1.484V | 17.23mW |

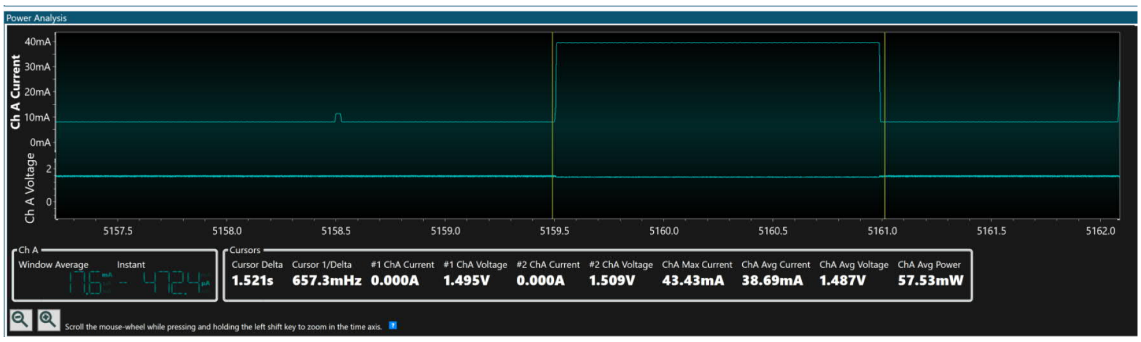
Scroll the mouse-wheel while pressing and holding the left shift key to zoom in the time axis.

Figure D.2: GN Measure Sensor + Process Data states Fport220.



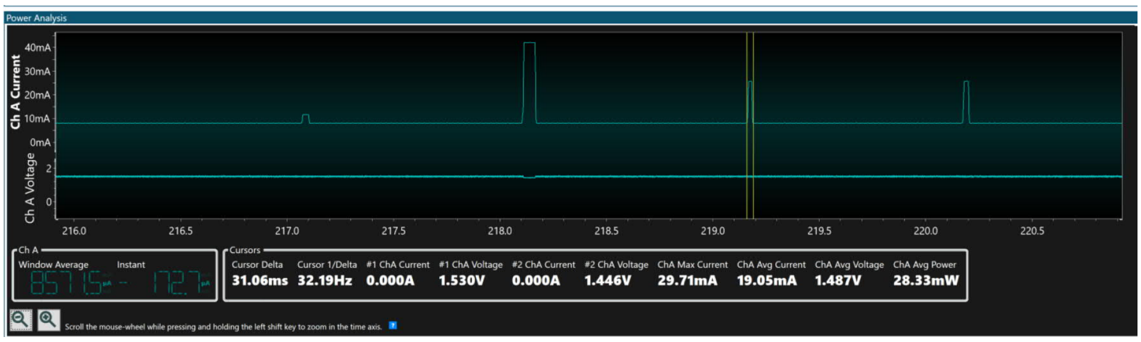
| Cursor Delta | Cursor 1/Delta | #1 ChA Current | #1 ChA Voltage | #2 ChA Current | #2 ChA Voltage | ChA Max Current | ChA Avg Current | ChA Avg Voltage | ChA Avg Power |
|---|---|---|---|---|---|---|---|---|---|
| 69.65ms | 14.36Hz | 0.000A | 1.459V | 0.000A | 1.469V | 55.13mA | 42.25mA | 1.347V | 56.92mW |

Figure D.3: GN Wake-up Transceiver + Transmit Data (SF7, BW125) states Fport220.



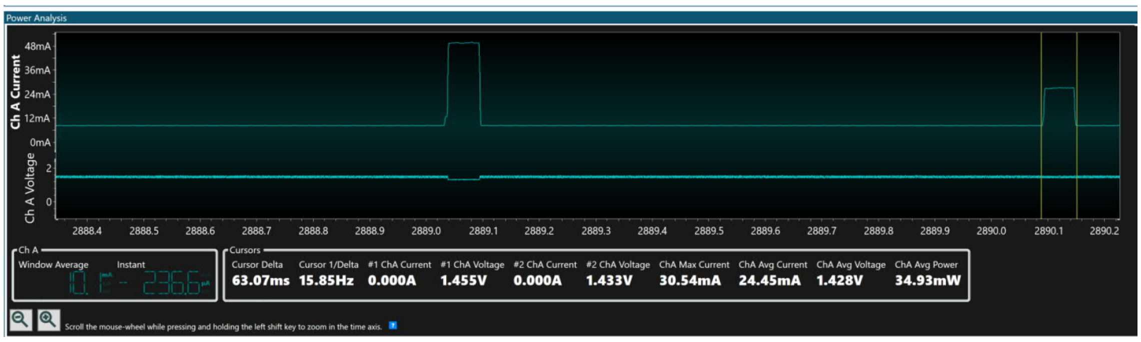
| Cursor Delta | Cursor 1/Delta | #1 ChA Current | #1 ChA Voltage | #2 ChA Current | #2 ChA Voltage | ChA Max Current | ChA Avg Current | ChA Avg Voltage | ChA Avg Power |
|---|---|---|---|---|---|---|---|---|---|
| 1.521s | 657.3mHz | 0.000A | 1.495V | 0.000A | 1.509V | 43.43mA | 38.69mA | 1.487V | 57.53mW |

Figure D.4: GN Wake-up Transceiver + Transmit Data (SF12, BW125) states Fport220.



| Cursor Delta | Cursor 1/Delta | #1 ChA Current | #1 ChA Voltage | #2 ChA Current | #2 ChA Voltage | ChA Max Current | ChA Avg Current | ChA Avg Voltage | ChA Avg Power |
|---|---|---|---|---|---|---|---|---|---|
| 31.06ms | 32.19Hz | 0.000A | 1.530V | 0.000A | 1.446V | 29.71mA | 19.05mA | 1.487V | 28.33mW |

Figure D.5: GN Receive Data (Rx1 + Rx2) (0 bytes) state Fport 220.



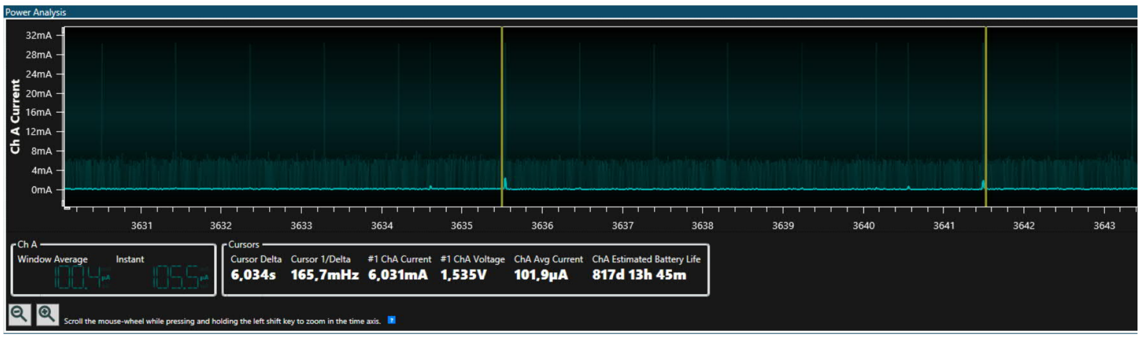
Ch A

| Window Average | Instant |
|---|---|
| 10.1 mA | 236.6 µA |

Cursors

| Cursor Delta | Cursor 1/Delta | #1 ChA Current | #1 ChA Voltage | #2 ChA Current | #2 ChA Voltage | ChA Max Current | ChA Avg Current | ChA Avg Voltage | ChA Avg Power |
|---|---|---|---|---|---|---|---|---|---|
| 63.07ms | 15.85Hz | 0.000A | 1.455V | 0.000A | 1.433V | 30.54mA | 24.45mA | 1.428V | 34.93mW |

Figure D.6: GN Receive Data (Rx1 + Rx2) (Micro Update) state Fport 220.



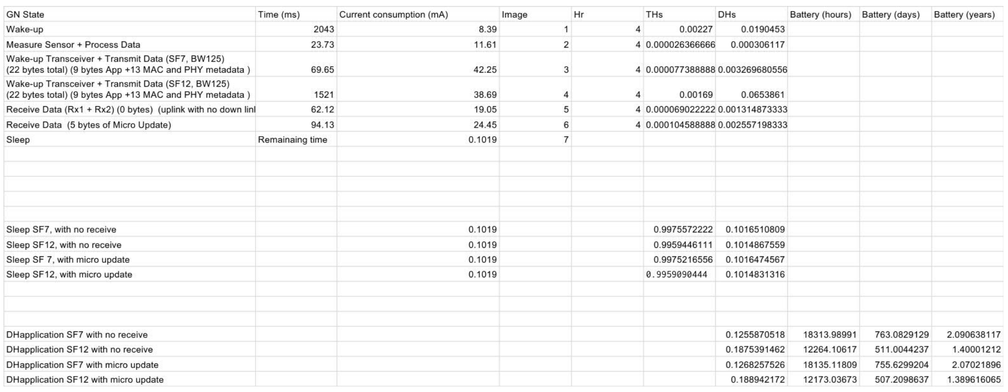
Ch A

| Window Average | Instant |
|---|---|
| 100.4 µA | 105.5 µA |

Cursors

| Cursor Delta | Cursor 1/Delta | #1 ChA Current | #1 ChA Voltage | ChA Avg Current | ChA Estimated Battery Life |
|---|---|---|---|---|---|
| 6,034s | 165,7mHz | 6,031mA | 1,535V | 101,9µA | 817d 13h 45m |

Figure D.7: Sleep state Fport 220.



| GN State | Time (ms) | Current consumption (mA) | Image | Hr | THs | DHs | Battery (hours) | Battery (days) | Battery (years) |
|---|---|---|---|---|---|---|---|---|---|
| Wake-up | 2043 | 8.39 | 1 | 4 | 0.00227 | 0.0190453 | | | |
| Measure Sensor + Process Data | 23.73 | 11.61 | 2 | 4 | 0.000026366666 | 0.000306117 | | | |
| Wake-up Transceiver + Transmit Data (SF7, BW125)<br>(22 bytes total) (9 bytes App +13 MAC and PHY metadata ) | 69.65 | 42.25 | 3 | 4 | 0.000077388888 | 0.003269680556 | | | |
| Wake-up Transceiver + Transmit Data (SF12, BW125)<br>(22 bytes total) (9 bytes App +13 MAC and PHY metadata ) | 1521 | 38.69 | 4 | 4 | 0.00169 | 0.0653861 | | | |
| Receive Data (Rx1 + Rx2) (0 bytes) (uplink with no down linl | 62.12 | 19.05 | 5 | 4 | 0.000069022222 | 0.001314873333 | | | |
| Receive Data (5 bytes of Micro Update) | 94.13 | 24.45 | 6 | 4 | 0.000104588888 | 0.002557198333 | | | |
| Sleep | Remainaing time | 0.1019 | 7 | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| Sleep SF7, with no receive | | 0.1019 | | | 0.9975572222 | 0.1016510809 | | | |
| Sleep SF12, with no receive | | 0.1019 | | | 0.9959446111 | 0.1014867559 | | | |
| Sleep SF 7, with micro update | | 0.1019 | | | 0.9975216556 | 0.1016474567 | | | |
| Sleep SF12, with micro update | | 0.1019 | | | 0.9959090444 | 0.1014831316 | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| DHapplication SF7 with no receive | | | | | | 0.1255870518 | 18313.98991 | 763.0829129 | 2.090638117 |
| DHapplication SF12 with no receive | | | | | | 0.1875391462 | 12264.10617 | 511.0044237 | 1.40001212 |
| DHapplication SF7 with micro update | | | | | | 0.1268257526 | 18135.11809 | 755.6299204 | 2.07021896 |
| DHapplication SF12 with micro update | | | | | | 0.188942172 | 12173.03673 | 507.2098637 | 1.389616065 |

Figure D.8: Applied GN battery estimation calculations based on devised equations in Chapter 5

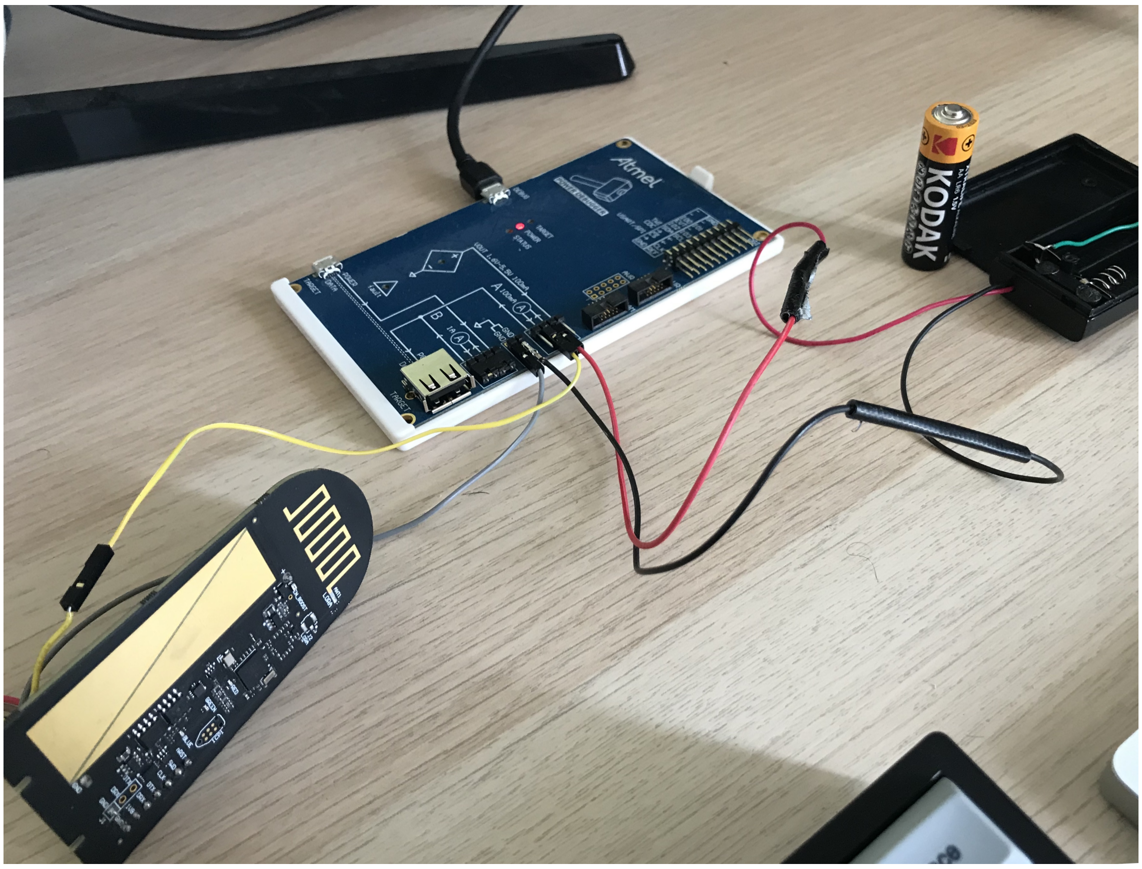
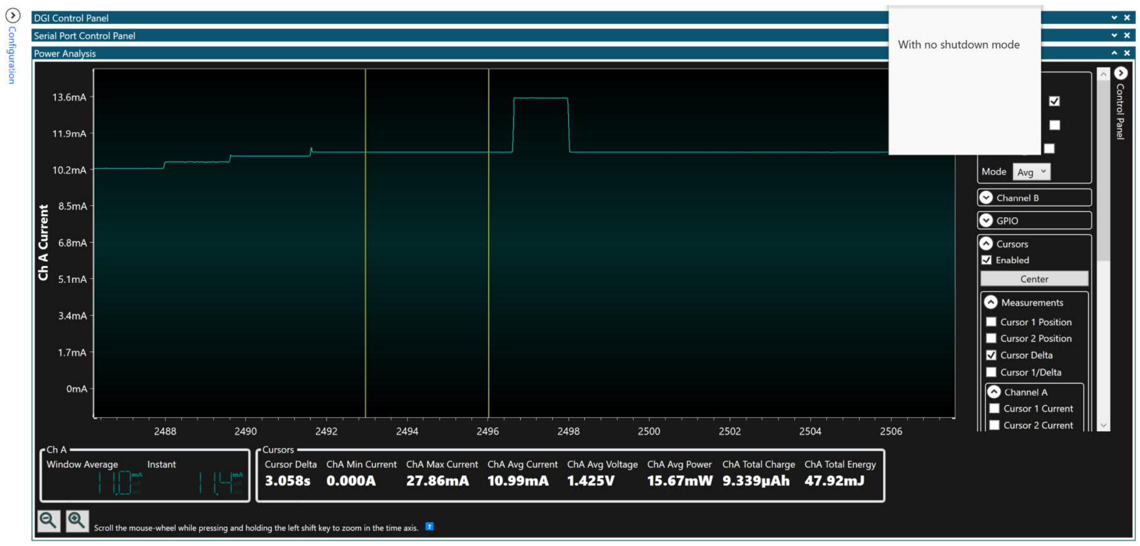


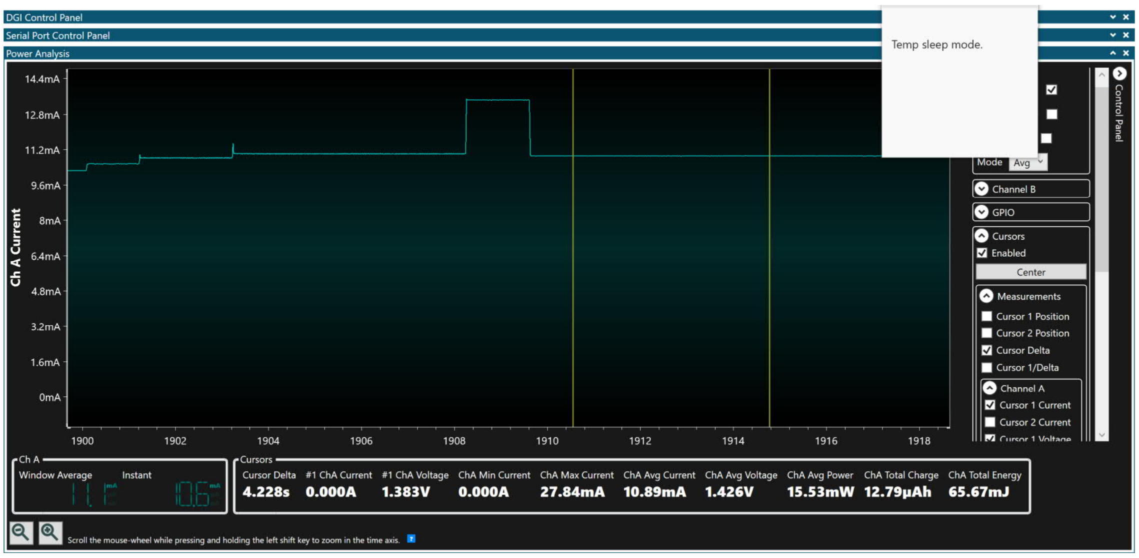
Figure D.9: Energy measurement setup.



Cursors

| Cursor Delta | ChA Min Current | ChA Max Current | ChA Avg Current | ChA Avg Voltage | ChA Avg Power | ChA Total Charge | ChA Total Energy |
|---|---|---|---|---|---|---|---|
| 3.058s | 0.000A | 27.86mA | 10.99mA | 1.425V | 15.67mW | 9.339µAh | 47.92mJ |

Figure D.10: Average current capture before enabling sensor shutdown mode.



Temp sleep mode.

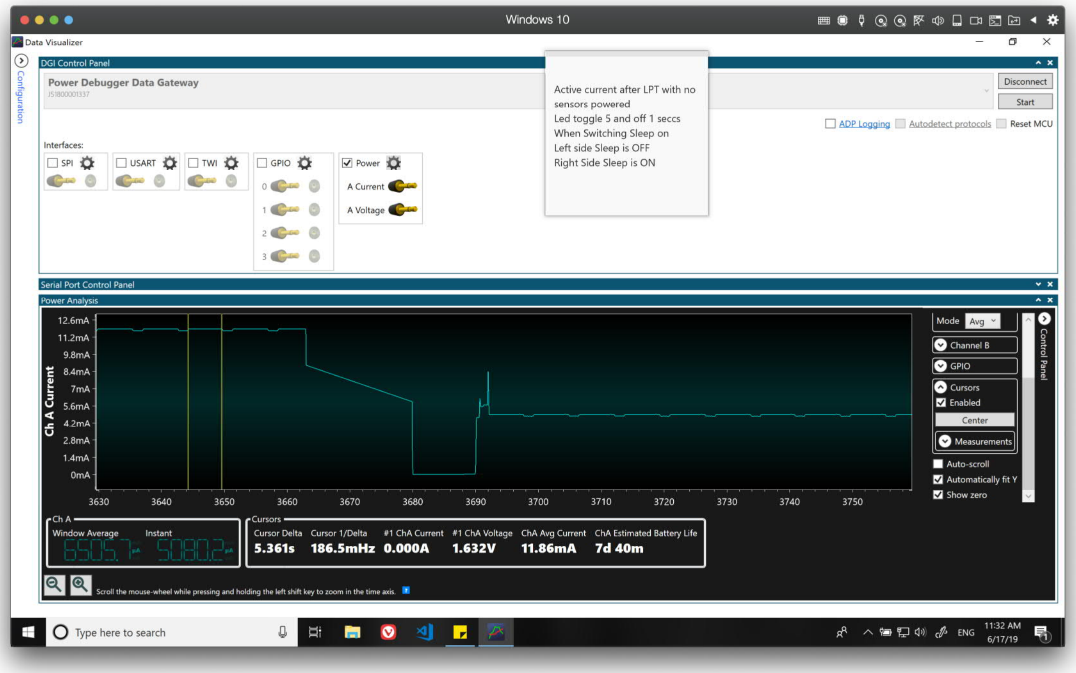
Ch A
Window Average 11.1 mA
Instant 10.6 mA

Cursors

| Cursor Delta | #1 ChA Current | #1 ChA Voltage | ChA Min Current | ChA Max Current | ChA Avg Current | ChA Avg Voltage | ChA Avg Power | ChA Total Charge | ChA Total Energy |
|---|---|---|---|---|---|---|---|---|---|
| 4.228s | 0.000A | 1.383V | 0.000A | 27.84mA | 10.89mA | 1.426V | 15.53mW | 12.79µAh | 65.67mJ |

Scroll the mouse-wheel while pressing and holding the left shift key to zoom in the time axis.

Figure D.11: Average current capture after enabling sensor shutdown mode.



Active current after LPT with no sensors powered
Led toggle 5 and off 1 seccs
When Switching Sleep on
Left side Sleep is OFF
Right Side Sleep is ON

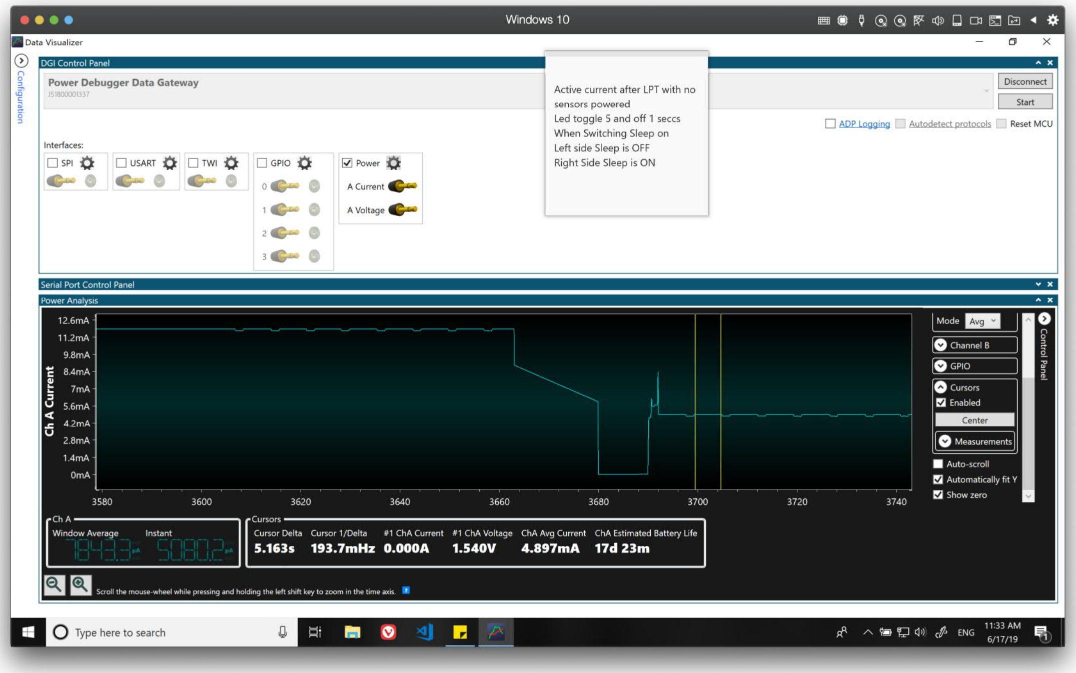
Ch A
Window Average 6505.7 µA
Instant 5080.2 µA

Cursors

| Cursor Delta | Cursor 1/Delta | #1 ChA Current | #1 ChA Voltage | ChA Avg Current | ChA Estimated Battery Life |
|---|---|---|---|---|---|
| 5.361s | 186.5mHz | 0.000A | 1.632V | 11.86mA | 7d 40m |

Scroll the mouse-wheel while pressing and holding the left shift key to zoom in the time axis.

Figure D.12: Average current capture before enabling low power ticker.


Active current after LPT with no sensors powered
Led toggle 5 and off 1 seccs
When Switching Sleep on
Left side Sleep is OFF
Right Side Sleep is ON

Ch A

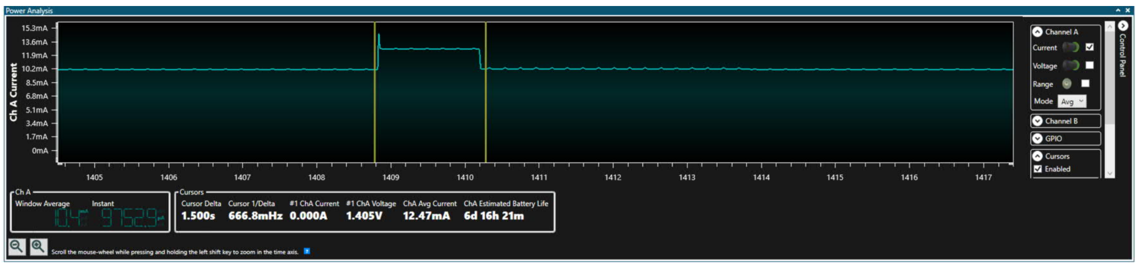
Window Average 7843.3 µA   Instant 5080.2 µA

Cursors

| Cursor Delta | Cursor 1/Delta | #1 ChA Current | #1 ChA Voltage | ChA Avg Current | ChA Estimated Battery Life |
|---|---|---|---|---|---|
| 5.163s | 193.7mHz | 0.000A | 1.540V | 4.897mA | 17d 23m |

Scroll the mouse-wheel while pressing and holding the left shift key to zoom in the time axis.

Figure D.13: Average current capture after enabling low power ticker.


Ch A

Window Average 10.4 mA   Instant 9752.9 µA

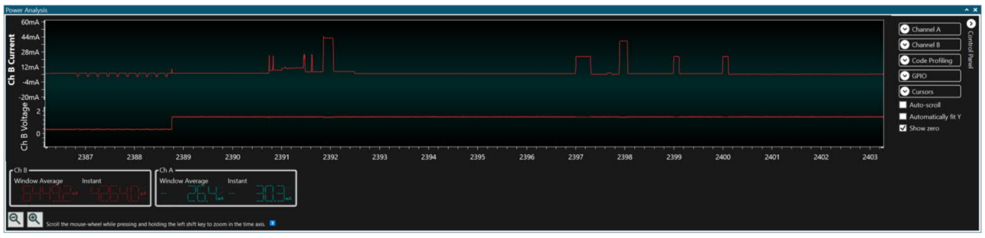
Cursors

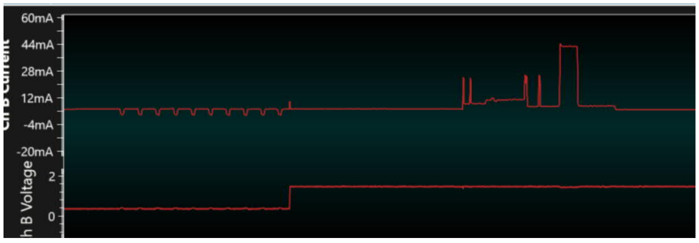
| Cursor Delta | Cursor 1/Delta | #1 ChA Current | #1 ChA Voltage | ChA Avg Current | ChA Estimated Battery Life |
|---|---|---|---|---|---|
| 1.500s | 666.8mHz | 0.000A | 1.405V | 12.47mA | 6d 16h 21m |

Scroll the mouse-wheel while pressing and holding the left shift key to zoom in the time axis.

Figure D.14: Inefficient SW driver measurement time capture.

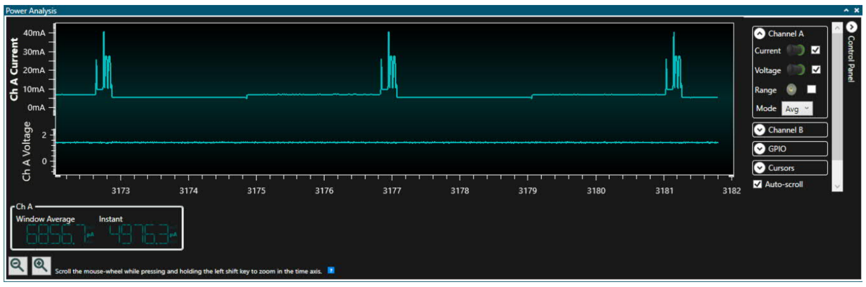
Imaged below: Lora Joining procedure with RFO_LF transmission mode and 20% full battery with no issues (working with no reset issues)





As you can see it consumes (highest point) around 45mA, which is acceptable for a low battery.

Imaged below: Lora Joining procedure with PA_BOOST transmission mode and 20% full battery (same battery) with Reset issues



Zoomed in with correct peek current level reach 95 mA.

Figure D.15: Device reset issue deep analysis report.