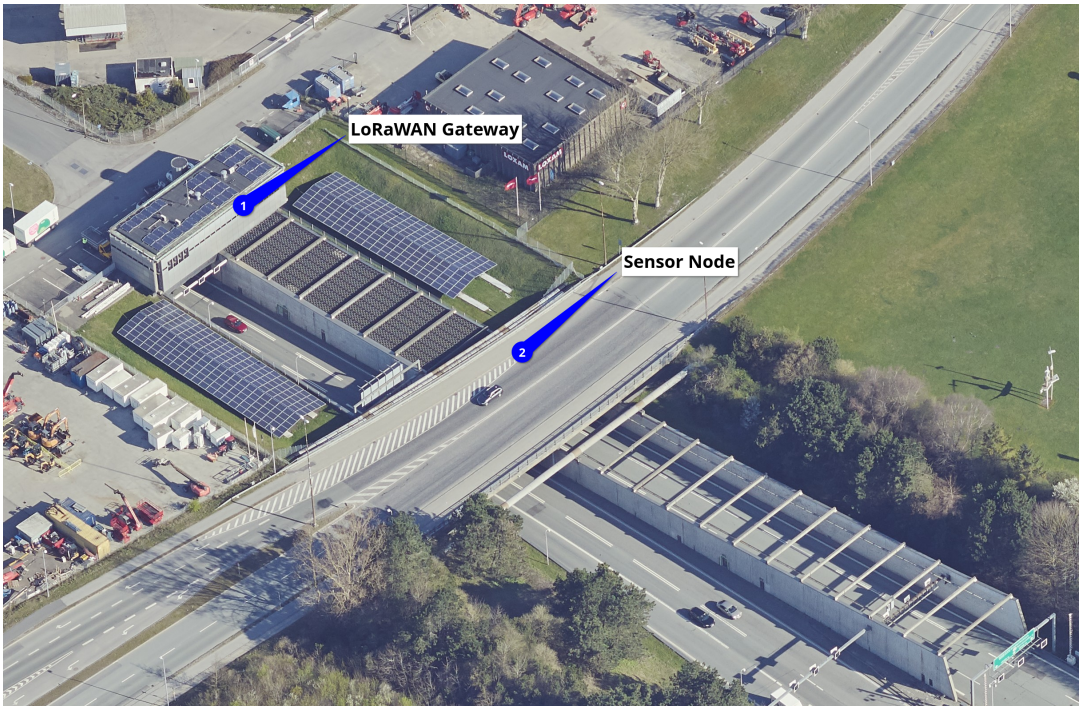
Applied Network Calculus for Optimization of LoRaWAN Communication using Insights from Real-world Measurements of Energy Harvesting Performance



MASTER'S THESIS

GROUP 1046

THOMAS GUNDGAARD MULVAD

RASMUS SIBBERN FREDERIKSEN

COMPUTER ENGINEERING - NETWORK AND DISTRIBUTED SYSTEMS

AALBORG UNIVERSITY

2024

AALBORG UNIVERSITY
STUDENT REPORT

The Technical Faculty of IT And Design
Computer Engineering - Specialisation in Network and Distributed Systems
Fredrik Bajers Vej 7
9220 Aalborg Øst

Title: Applied Network Calculus for Optimization of LoRaWAN Communication using Insights from Real-world Measurements of Energy Harvesting Performance

Project:

10th semester - Master's Thesis

Project Period:

February 2024 - May 2024

Project Group:

1046

Participants:

Thomas Gundgaard Mulvad
Rasmus Sibbern Frederiksen

Supervisors:

Tatiana Kozlova Madsen
Jimmy Jessen Nielsen

Number of pages: 85
Appendix: 2
Date of Completion: 31 May 2024

Abstract:

Structural Health Monitoring (SHM) use cases require fit-and-forget wireless sensors for scenarios where maintenance is not possible. Such sensors must operate indefinitely using Energy Harvesting (EH). However matching variations in EH productivity with reliable wireless transmissions is non-trivial, as time characteristics are not well captured by the average energy produced and consumed. In this work a method for providing the best approximation for achievable reliable throughput of a wireless system, for a given energy source, in this case a solar cell, is found using Deterministic Network Calculus (DNC). The method is validated using real-world measurements on communication reliability and energy harvesting, through the implementation of an EH wireless sensor node, which is used to perform an accelerated measurement campaign. The sensor node emulates multiple LoRaWAN devices, in order to test multiple communication parameters at the same time. The measurement campaign revealed that a specific LoRaWAN configuration outperforms when taking a fixed reliability and the energy consumption into consideration, it is also shown that longer packets weren't necessarily better as they were prone to collisions in the ISM band. Using flows for energy harvesting and energy consumption the system was analyzed using network calculus to provide maximum throughput.

Preface

We would like to thank Niels Gustav Jørgensen from Vejdiktoratet for making it possible to conduct the measuring campaign at the Limfjords tunnel.

Throughout the rapport when using graphing tools for illustration purpose these were made possible by using the xkcd font [1].

The full code for the rapport can be found at the github repository here:
`https://github.com/rasmussibbern88/P10`

Reading Guide
When reading references for figures, listings of code, equations, and tables the first number is the chapter and the last numbers are the sections in that chapter. All references can be clicked on to be taken to the corresponding chapter and section.

Source references use a numbered system [Number] with the sources listed in the bibliography section in numerated order at the end of the report.

Rasmus Sibbern Frederiksen

Thomas Gundgaard Mulvad

Acronyms

ADC analog-to-digital converter
ADR Adaptive Data Rate

CCA Clear Channel Asessment
CID command identifier

DNC Deterministic Network Calculus
DR Data rate

EH Energy Harvesting
ERP Effective Radiated Power

I2C Inter-Integrated Circuit
IoT Internet of Things
ISC short circuit current
IV current-voltage

LoRa Long Range
LoRaWAN Long Range Wide Area Network

MAC Media Access Control
MHDR MAC header
MIC message integrity code
MPPT Maximum Power Point Tracking

PCB printed circuit board
PDR Packet Delivery Ratio

RSSI Received signal strength indicator

SNR Signal-to-noise ratio
SPI Serial Peripheral Interface

ToA Time on Air

VOC circuit voltage

Contents

´

1 Introduction

1.1 Motivation

According to the EnABLES EU project [2], by 2025, up to 78 million batteries will be produced and discarded every day [3], and the ubiquitous growth of IoT (Internet of Things) and connected wireless devices creates an even greater demand.

Batteries are used to power IoT devices because it allows for freedom in choosing where to install the devices, without worrying about electrical installations. However as the amount of connected devices grows it becomes more apparent, that the total cost of battery operated IoT devices is also substantial. Batteries require regular recharging or replacement, and this still enforces a limit on the applications for IoT, as a fleet of IoT devices still has to be maintainable.

Maintenance costs of IoT devices can also vary depending on where they are installed, as they can be placed in irregular locations that are hard to get to or simply in so diverse locations that replacing them all takes to long time[4, 5, 6]. They can also be lost or forgotten when their batteries run out after years. And there are even some applications where having a fit and forget mindset is not just a useful property, but necessary for the long term system functionality [7].

Applications where fit and forget solutions are required could be installations of IoT devices in asphalt or similar structural monitoring use cases, where the devices are embedded into the structure or are otherwise wildly inaccessible.

The demand for maintenance-free operation in Structural Health Monitoring could grow to match the lifetime of the object it is monitoring. As an example road ways are expected to last for 15 years [8] and the great belt bridge, originally built to last for 100 years, is now expected to last for 200 years [9].

The best hope for powering IoT devices in fit and forget applications is providing the sensor with a renewable energy source, and as these sources are often artificial lighting, shade daylight, air-surface temperature gradients or vibrations caused by traffic, they are minuscule, compared to the power generation on the electric grid. Sensors using these energy sources are instead referred to as EH (Energy Harvesting) devices.

As many EH sensors are expected to operate perpetually, a new perspective on power management must be introduced. Battery lifetime estimations often disregard when and how much power is required by the sensor and instead calculations are base on the average power

consumption. In EH sensor nodes, the average power consumption is still a useful tool when determining whether the energy harvesting capacity meets the application demand, yet if a battery powered sensor device exceeds its expected average power consumption, it will still function, with a shortened lifetime, while in EH devices this may violate end user requirements, this is illustrated in Fig. 1.1. The recourse's, for an EH node with a power deficit, are to transmit less data, less frequently and less reliably.

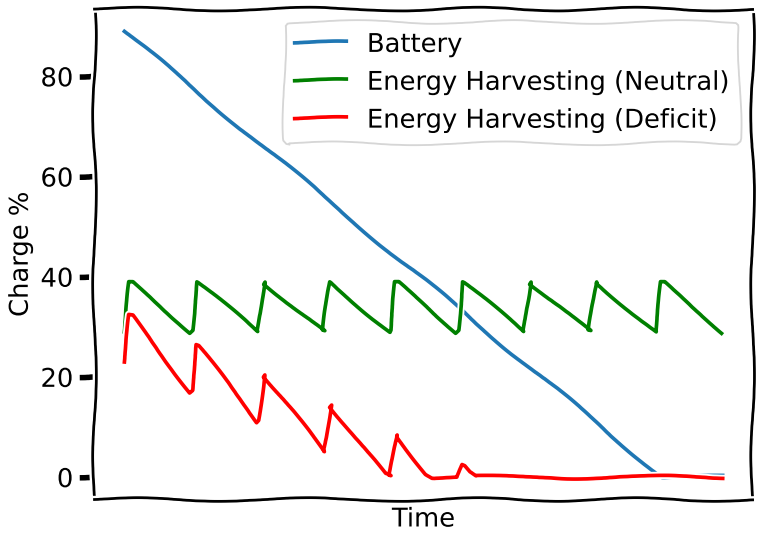


Figure 1.1. Illustration of the issue with operating in power deficit.

The wireless communication used by EH sensor nodes also further complicates the ways exact power consumption can be determined, as the energy consumption can vary based on external factors, that are unpredictable during the design phase, such as the distance of the wireless link, the network load, interference and the propagation environment.

Additionally, power used for wireless data transmission depends on the L1 and L2 protocols used, e.g. what modulation schemes are used and whether retransmissions or repetitions of the signal are applied. Reliability of data delivery is directly connected with the energy used on data transmission: the more energy is used, the more reliable the transmission can be. For EH-based systems it is desirable to find configuration and transmission schemes that use the minimum energy possibly while still achieving the desired reliability. These considerations lead to the initial problem statement: how an IoT system with a wireless data collection utilizing EH as a source of energy can be designed?

1.2 Delimitation

In this project an energy harvesting sensor node is designed and installed in conditions equivalent to a structural health monitoring use case, in particular monitoring of reinforced concrete corrosion in an overpass bridge[10]. The focus on the sensor node is to install

and perform a measuring campaign collecting real world data, as real world data includes environmental influences that can have impact on both energy harvesting but also the communication channel. These factors and variations are difficult to replicate in simulations so getting data from an actual measuring campaign is prioritized in the project.

During the measurement campaign the purpose of the sensor node is two-fold: 1) to collect time varying information on the energy harvesting and 2) to estimate the quality of data transmissions and asses the reliability of transmission for a chosen communication system. The data collected during this project is specific to the location of the measurement campaign. While the data is limited to a location it is possible to provide useful insight into the Structural Health Monitoring scenario and can serve as a basis for future work in these scenarios.

There are numerous technologies that can be used for both energy harvesting and low power communication when designing and implementing a sensor node. In this project the most important aspect is that the technologies chosen resemble real world choices and that they are simple to analyze and implement, as such a solar cell based energy harvesting solution was chosen in combination with LoRaWAN (Long Range Wide Area Network), serving as the low power communications solution. Both of these technologies are viable, widely available and easy to start and experiment with.

The pattern for packet transmissions will be mimicking the use case for structural health monitoring applications. These scenarios are characterized by periodic data collection without strict real time requirements and with reliable data delivery. The period of data transmissions may vary from once per day to once per month depending on the application requirements and the physical phenomena under observation, e.g. corrosion or humidity. Given the limited duration for a measuring campaign, the frequency of transmissions will be set higher, while obeying duty cycle requirements for LoRa (Long Range) transmissions in the ISM band.

The data collected from the measurement campaign will be used to create models for both 1) The energy produced by EH and 2) Characterization of energy consumption related to a wireless communication module and the reliability of data transmissions.

The nontrivial question addressed in this project is how the energy produced by a solar cell can be spent efficiently by a LoRa communication module and how to provide the guarantees that the designed system is operational. For the latter, use of the modeling toolbox: DNC (Deterministic Network Calculus) is envisioned, allowing for comparisons between harvested energy and energy used for communication and thus providing the tools required to make guarantees that the chosen communication strategy is supported by the EH module.

Network calculus uses min-plus algebra to provide worst-case deterministic guarantees [11] on, traditionally packet switched networks. However it has also been introduced to model the behaviour of the electrical grid [12]. In this project however the traditional packet analysis is only partially replaced.

As the energy harvesting works in units of power and energy, while the LoRaWAN communication at a higher level works with packets of data. A model for converting the data transmission

of LoRa packets into units of energy is required. This project assumes a simplified model of one way communication, and thereby reduces the mapping to the per packet energy of LoRa communication with a fixed reliability, which will allow for mapping the parameters of a communication system, to the energy harvesting requirements.

DNC captures the time varying characteristics, which the average power consumption metric that is typically used lacks [13]. However it is frequently used for analysis of hard real time systems, as it does not model random packet loss, and as such another assumption must be introduced, similarly to DNC in packet switched networks, that a sufficiently high reliability is accepted and ignored in the analysis.

By utilizing DNC (Deterministic Network Calculus) with the collected data the final problem statement is formulated for the project.

How can real-world data on energy harvesting and wireless transmissions be modeled using network calculus to characterize the time-dependent relationship between the harvested energy and consumption required for wireless transmissions in structural health monitoring scenarios?

A few sub questions are stated, to help in answering the problem statement.

- How can the deterministic aspects of network calculus be used with the variable aspects of energy harvesting and reliable communication?
- How can a sensor node be designed to support the measurement campaign described?
- How is energy production of an EH module modeled in Network Calculus and what impact do the parameters, such as time of the day, temperature and the presence of artificial lights, have?
- What impact does different LoRa settings and transmission patterns have on the energy consumption of a communication module?

The report and the following chapters is structured as following:

Chapter 2: The technical analysis, design and implementation of the sensor node. It explores and analyzes the technologies while also explaining the design choices and implementation of multiple iterations of the sensor node. The chapter also highlights interesting problems that were solved during the iterative progress of designing and implementing a sensor node.

Chapter 3: This chapter focuses on the data collected for the energy harvesting side of the sensor node. It will present the data and also provide additional data processing to make the data viable for use in network calculus.

Chapter 4: This chapter is structured similar to chapter 3, but focuses on the reliability data collected when transmitting LoRa packets from the sensor node. Furthermore it compares the energy efficiency of repetition of LoRa packets, as a method for improving reliability.

Chapter 5: The Network calculus chapter first introduces the theory behind network calculis, then integrates the results from chapter 3 and 4, in order to derive flow models, and finally the network calculus results are found.

Chapter 6: Provides a discussion on the sources of error in the measuring campaign, the design choices of the sensor node, and the assumptions underlying the analysis

Chapter 7: The possibilities for future work in applying NC for the design of EH systems is outlined and discuseed.

Chapter 8: Provides the project conclusion, and a summary of the project in relation to the problem statement.

Technical Analysis, Design and Implementation of Sensor Node 2

This chapter presents the technical analysis, design and implementation of the sensor node developed for the measurement campaign conducted during the project. It also covers the supporting infrastructure required for the sensor node to communicate using LoRa.

The chapter begins with an overview and then delves into the technical details for each component of the sensor node. The sections cover the technical analysis, design and how these technologies were implemented in the sensor node.

Throughout the project, multiple iterations of the sensor node were developed. Additionally this chapter also covers some of the challenges encountered and the subsequent modifications made to address these issues.

2.1 Overview

The sensor node is the device developed for the measuring campaign at the Rørdals bridge / Limfjords tunnel. The goal of the sensor node is to gather information by conducting a measurement campaign about energy harvesting from a solar cell and information about transmissions using LoRa. Transmitting LoRa packets have the goal to test the reliability of the tunnel environment when there is a changing environment, the changes could E.g. be temperature or traffic. Good estimates of the reliability is necessary for selecting energy efficient transmission settings.

Throughout the project multiple iterations of the sensor node was produced for separates tests, the two main versions were:

1. Strip board version
2. PCB (printed circuit board) version

Both versions have undergone multiple variations over time during the project.

To better understand the system's structure a system overview of the infrastructure and subsystems for the sensor node is presented as seen in Figure 2.1. The system is comprised

of the sensor node and its supporting infrastructure. At the heart of the sensor node is the micro controller, adjacent and to the left is the energy harvesting part of the system. The energy harvesting includes the solar cell, Maximum Power Point Tracking (MPPT) and the capacitor that functions as a rechargeable battery. The microcontroller communicates with the MPPT but also monitors the energy harvesting system to collect data points independently of the MPPT itself.

Additionally, the microcontroller is connected to an illumination sensor to gather information about light intensity in the environment. The microcontroller is logging the data collected to an SD Card, and employs LoRaWAN technology to transmit packets to the infrastructure.

In terms of infrastructure a gateway was deployed to receive and respond to incoming packets from the sensor node. The gateway forwards LoRa packets to and from the LoRaWAN network server, which is responsible for decoding packets, drafting responses and storing data in a persistent database. An alert system is also integrated to monitor the values received from the sensor node, enabling quick notifications whenever data points deviate from the expected values.

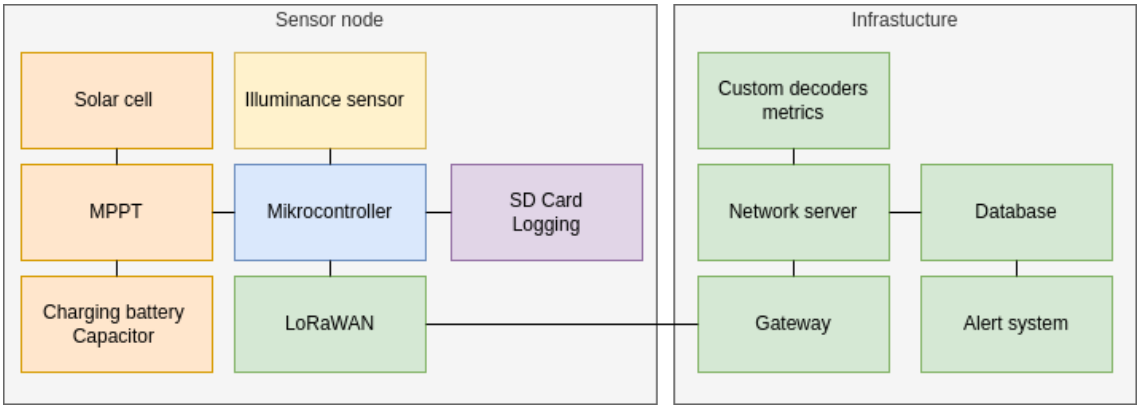


Figure 2.1. Block diagram illustrating the subsystems of the sensor node and the infrastructure for this project

2.1.1 Microcontroller and Platform

There are numerous options to consider when choosing a microcontroller for the sensor node. These options include specific project requirements such as peripheral needs, power consumption, but also include the desired development experience.

The microcontroller serves as the heart of the system, responsible for all data collection, data processing before either storing the data locally or sending the information using LoRa communication. Another important part of the microcontroller is the power management, as effective power management reduces the average power consumption and extend the sensor nodes lifetime. This is even more crucial when used in an system that relies solely on energy harvesting, as the available energy changes over time.

In this project two different microcontrollers were used. The initial version referred to as the

strip board version, employed a Raspberry Pi Pico. In the second and final version of the sensor node a Wio-E5 mini board was selected, primarily due to the simplified design with an integrated LoRa module. The Wio-E5 board utilizes a STM32WLE5JC chip that includes an SX126x LoRa Transceiver in the same package. This makes it a perfect fit for the sensor node developed in this project [14].

Coding Platform

When using these microcontrollers there is a choice for the coding platform, multiple platforms are available to chose from, depending on coding language and support it provides for the chip from libraries, that abstract the hardware throug Hardware Abstraction Layers (HAL). Some of the platforms that were considered for this project and their respective languages are:

1. Tinygo (Go)
2. Embassy (Rust)
3. Micropython (Python)
4. Libopencm3 (c)
5. STMCube with STM32Hal (c, additional graphical interface)

The support for the microcontroller varies between each platform, and the chosen platform for the versions are also depended on the abstraction level of the language and prior experience with the platform.

As the first version served as a prototype to validate that the sensor node would function correctly, the platform used was Micropython. Micropython makes it easy to execute code on the Raspberry Pi Pico, including support for interactive mode which allows for quick issue resolution and code development.

The second version of the sensor node utilized Tinygo to better meet the requirements for LoRa and had improved power management. The initial LoRa support was not great but the foundation for interacting with the LoRa transceiver was made available though drivers, and modifications here were made it possible to expand the functionality to the needs of the project. With use of Tinygo the programming language is Go and also the preferred language of choice making it easier to control the chip, especially when unknown problems occurs. The tinygo language offers a good tradeoff between abstraction for fast firmware development, firmware size and speed.

2.2 Strip Board Version

This section introduces the strip board version of the sensor node designed and implemented in this project. The strip board version was developed and installed at the bridge to test if it was possible to get valuable information from the design at the bridge. The section shows the

process going from a mock up board to a design that can be placed inside the mounting device and be used for initial testing.

2.2.1 Components

Before connecting anything the right components for the sensor node needed to be selected. Due to this being a prototype some of the components are also chosen with this in mind. A list of the components used in the strip board version can be seen in Table 2.1. Below is an introduction to each of the components.

One of the sensor node requirements is to measure the energy harvested, and in this project doing so from a solar cell, so for this a TJ GaAs Solar Cell Assembly 3g30A was used, this solar cell is usually used together with cube sats and was borrowed from AAUSAT for this project.

To control the solar cell a MPPT (Maximum Power Point Tracking) controller is needed to get the best results as described further in Section 2.6 on page 22. The solar energy click 2 board is used for this purpose, this board is detailed in Section 2.5. Also from Mirkoe Click is the Click Current 7 board that is added to measure the current produced by the solar cell. A voltage divider is used to measure the voltage to enable the calculations of the energy harvested. One last click board, the Illuminance Click was added to measure the light intensity. All of these click boards were tested individually before added to the breadboard mock up.

In this version there were no communication and the data was stored on a SD card. Everything is controlled by the Raspberry Pi Pico as described in 2.1.1.

Component
TJ GaAs Solar Cell Assembly 3g30A
Solar Energy 2 click
Click Current 7
Illuminance Click
SD card

Table 2.1. Table of components in the strip version of the sensor node

2.2.2 Breadboard Mockup

A breadboard assembly of the sensor node was made to ensure the correct integration between all the components, the assembly can be seen in 2.2. The breadboard allowed for easy access to testing points for components, to verify they were supplied with the correct voltages at the correct states. Outside the frame in 2.2 a solar cell was connected to the energy harvester. During testing, the transistors used to control the power lines were observed to have a voltage drop that was deemed to significant, an alternative solution of MOS-FETs were chosen, and instead of gating the supply line the return line was gated instead with low side switching.

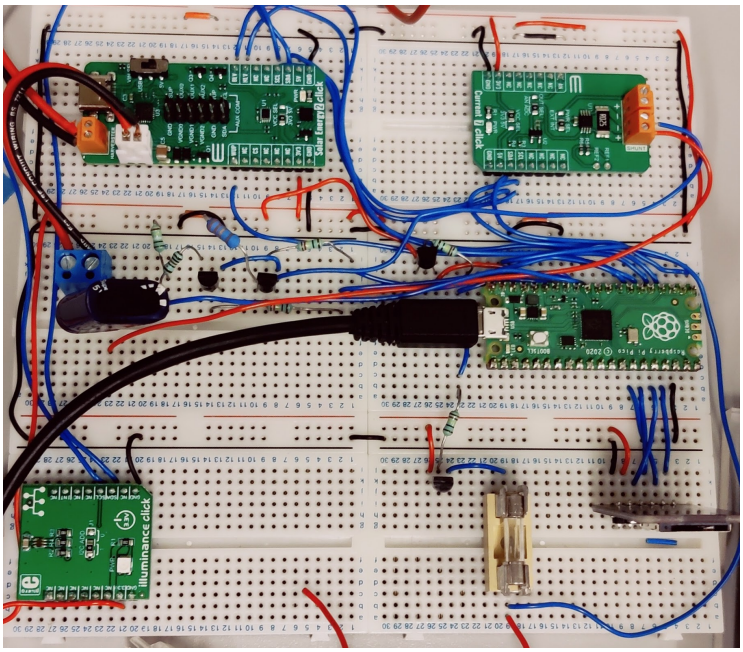


Figure 2.2. Component integration test on breadboard

2.2.3 Initial testing

After the integration test the sensor node were moved to a strip board were the components are soldered directly on or is connecting using sockets. The strip board were designed to be placed inside the container selected for the measuring campaign and the test fit of the strip board in the enclosure can seen in Figure 2.3.

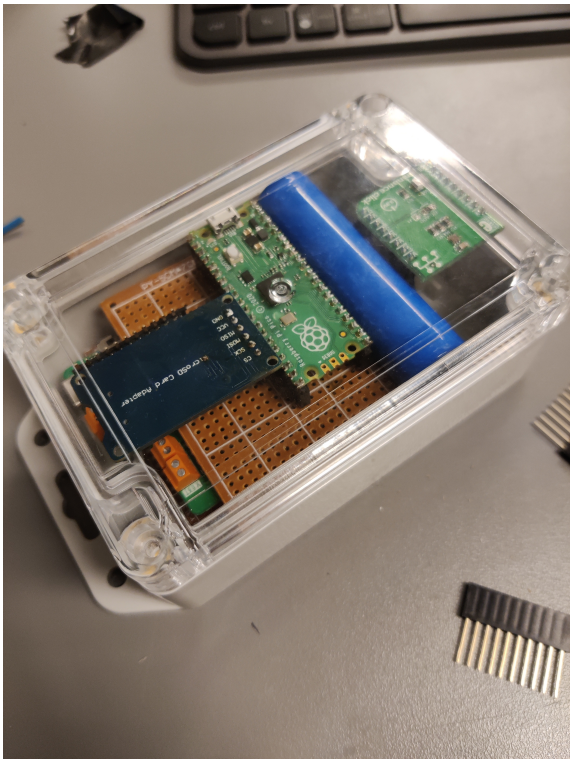


Figure 2.3. Test fitting of the strip board inside the enclosure

The enclosure with the strip board version with the solar cell attached can be seen in Figure 2.4. Here the sensor node is mounted to the underside of the bridge and is pointing towards the traffic to gather information.

2.2.4 Voltage Divider Issue

The voltage divider circuit on the first iteration of the sensor prototype, was faulty in the software and hardware, an error during production of the sensor left the ground connection of the voltage divider unconnected, effectively the capacitor was still connected to the ADC (analog-to-digital converter), but the voltage was not divided. A second error was found in the conversion formula, that caused a multiplication instead of a division. This was identified in the measurements collected, where the expected range was 0-5V, but instead it was actually calculated in the range of 50V. The incorrect conversion could be corrected in post-processing as it was a linear mapping, however during operation the sensor node used the calculated value, to limit the voltage range of the capacitor to 0-5V, and the capacitor was effectively drained for every wake-up of its operation.

2.3 Sensor Node Hardware

This sections goes into the hardware used for the sensor node. The section focuses on the second version, that uses the same components as the strip board version as seen in Section 2.2 with minor differences. The biggest change is the microcontroller that was exchanged with the Wio-E5-mini board instead that enabled transmitting LoRa.

The section presents the design of the sensor node hardware, explains the modifications, and provides the reasoning behind them.

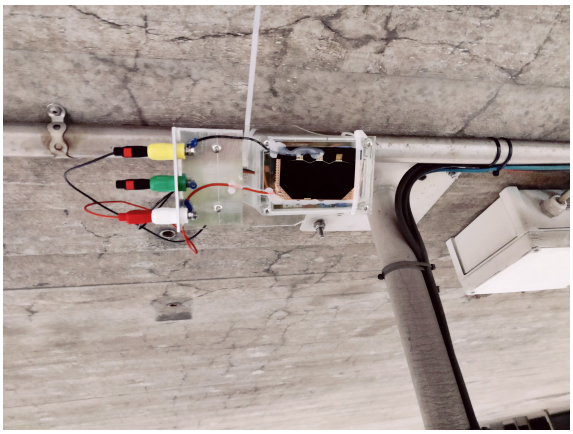


Figure 2.4. Strip board version of sensor node mounted to the underside of the Rørdal bridge

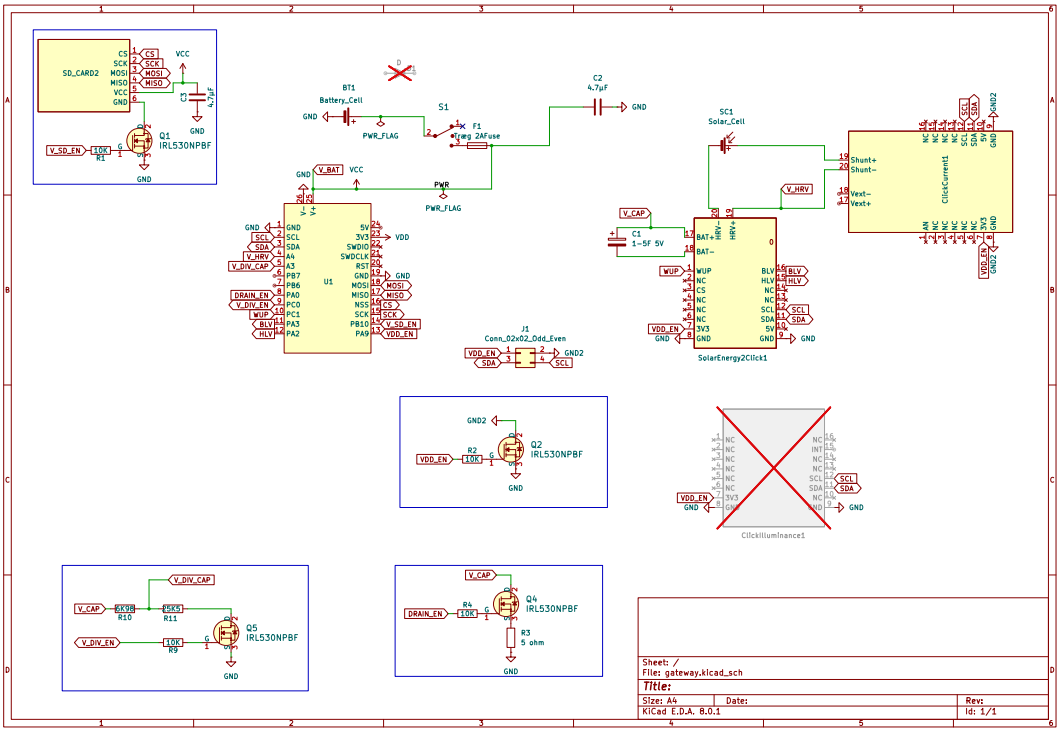


Figure 2.5. Schematic designed with KiCad

2.3.1 Schematic

The schematic for the sensor node is simple, as it is based around the Mirkoe Click boards, that have standard pin layout and dimensions in 3 predefined sizes makes it easy to expand functionality around the wio-e5-mini, as the limited amount of digital and analog pins, means that heavy use of the I2C (Inter-Integrated Circuit) and SPI (Serial Peripheral Interface) interface should be used. The click boards package a large amount of functionality, that can be accessed on the I2C bus with addressing, which is not available on the SPI bus without taking up more digital IO pins. The remaining components on the board consist of safety and convenience functions for the battery supply and circuitry to reduce sleep power consumption. As a voltage divider may carry too much current through the ground path. Some features are not present that would be critical in production grade hardware, such as battery protection circuits, to prevent over discharge.

2.3.2 WIO-E5 Modifications

The Wio-E5-Mini / LoRa-E5-Mini board[14] is designed as a compact version of the non mini breakout version, The board design provides access to large variety of peripherals, as not all the connections of the microcontroller can be exposed in the small form factor. The board was designed as a development board and not as final application, some considerations towards ease of use and low cost were made.

The board exposes one SPI bus, one I2C bus, 2 UART's, 2 Analog input channels and SWD connections for programming custom firmware, and the choice between using a U.FL or SMA Antenna port. Some modifcations can be done to the board to better support low power operations. The components of interest in this report are highlighted in red in figure 2.6. The I2C Pullup resistors, when connected to devices with their own Pullup reistors, can leak current to the voltage supply line of each of the devices. The Pullup resistors are each 4700Ω, and as they are in parallel across the clock and data line, and in series from the I2C master voltage supply and the I2C slave voltage supply, their effective leakage is $(\frac{3.3V}{(4700ohm+4700ohm)})^2 0.7mA$. The current would increase for each additional I2C slave device with a pullup resistor. As the desired sleep current is in the range of microamps, and since there is not a way to disable and re-enable the pullup on the board, it is nescessary to remove the supply of current, and this was done by desoldering the 2 resistors on the board.

The WIO-E5-Mini includes 2 antenna ports, however only one is connected to the RF output of the microcontroller at a time. A junction with 0Ω resistor selects which antenna port is connected. By default the SMA port is connected the RF output pin. However when fitting the board inside a mounting enclosure while keeping the anntenna on the outside of the enclosure it is beneficial to be able to use thin flexible wires inside the enclosure. This meant moving the 0Ω resistor from the SMA antenna port to the U.FL port instead.

Finally the development board includes a 3.3V regulator, that allows for cheap and easy DC-DC conversion for supplying the microcontroller with 3.3V from a 5V USB supply. However regulators are inefficient in their DC-DC conversion compared to switched power supplies, especially when there is a significant voltage difference between the input and the desired output in combination with a high load. While regulators may not be ideal in applications where the voltage supply may vary significantly, it was determined to be suitable for the project.

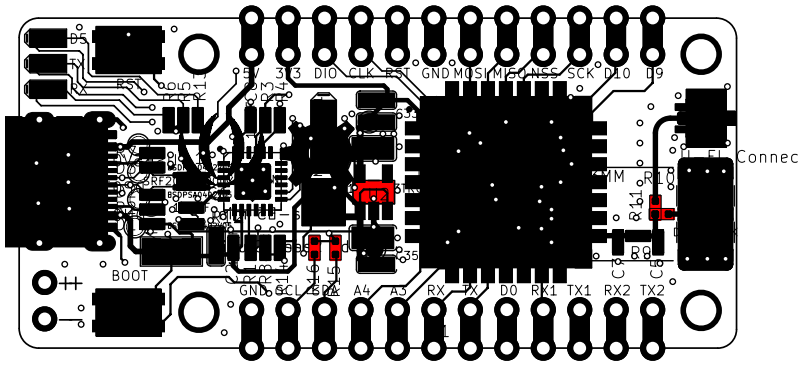


Figure 2.6. WIO-E5[14] Highlighted Components, I2C resistors (lower left), 3.3V regulator (middle), Antenna Port selection (Right). The board file is provided by Seeed Studio[15]

Modifications for I2C pullup

This section expands on the discovery of the I2C leaked current and the chosen solution. During one of the integration tests on the sensor end node, the power draw was higher than expected. When examining the power line used to enable the click boards, the voltage was still 2.2V at sleep, whereas the intended value was 0V. This caused the click boards to still be active when the microcontroller was at sleep.

By process of elimination on one of the click boards, it was determined that they could be powered on without supply voltage but instead using the I2C bus pulled high from the microcontroller, this means that the supply voltage of the microcontroller was connected to the supply line of the click boards through 2 pairs of I2C pull-up resistors.

The microcontroller schematic showed that the I2C bus pins SCL and SDA were pulled up to 3.3V through a 4.7K resistor. Having these pull-up resisters follow the standard of the I2C bus[1]. The pull-up resistors ensure the bus is high when a master releases the bus. No device is allowed to force a high, and they need to back off if the line is low, ensuring that the bus never runs into communication issues.

With the I2C bus connected to the three click boards as slaves and the microcontroller as the master. With only one master on the I2C bus it was decided to test if removing the pull resistors presented any immediate issues. The I2C peripheral was tested with the pull-up resistors removed from the microcontroller, no immediate issues were found, and the issue of the enable pin being high was resolved. Thus, the decisions was made to desolder the resistors from the microcontroller board. Using the schematic provided for the microcontroller, the location of the resistors can be found, as shown in Figure 2.6.

The lack of pullup resistors on the I2C host is not expected to cause any issues, as each device on the I2C bus has the pullup resistors needed for the proper functionality of the bus[16], and as the distance between the components is small it will not cause issues for the master device.

2.3.3 PCB Layout and Production

The design considerations for the PCB layout are detailed in this section, some of the primary considerations for the design were compactness and safety.

The PCB is powered by a battery, which is fed through a screw terminal as the input. To reduce the impact of a short in the circuit, a fuse is installed to limit the amount of current. However to ensure the current does not bypass the fuse, the trace should be kept short and far away from the ground, and as such the fuse has been placed close to the positive battery supply terminal, on the opposite side of the ground plane. One poor design decision was the use of the 2x1 screw terminal, as this places the positive and negative leads of the battery needlessly close together. The ground connections were made through a copper pour on the back plane of the 2 layer PCB, with additional through hole vias to connect the disjoint sections of the ground plane.

The ground back-plane was chosen, as it represents one of the nets with the most common connections, dedicating the back-plane for those connections reduces the amount crossing traces and thereby simplifies the routing.

As the mechanical requirements for the PCB, the size of the enclosure has limited the PCB dimensions to 70mm x 90mm (not including the protrusions), the trace widths had to be as small as possible. The net class defined in Table 2.2 was used. Additionally the clearance was set to 0.3 mm. The trace widths exceed what is required in terms of load capacity by a factor of 2 to 8, as a safety margin for the electronics and to allow for some safety margin above the minimum trace width possible on the mill. Similarly to the trace width limitations, there are a set of available tool sizes, that have to be matched to the components. This means mapping 40mil pin holes to 1mm, 1.5, 2mm etc. drill sizes.

Net Class	Trace Width [mm]
Default	0.45
Analog	0.65
Digital	0.45
Power	0.75
Minimum	0.30

Table 2.2. Showing the chosen trace width for the 4 defined net classes and the minimum trace width, that could be produced reliably

The decoupling capacitors were laid out close to the micro controller power supply input and the SD card connector supply pin, as they have a high power draw with switching circuits, this has the effect of reducing noise and improving the supply stability for these components.

The dimensions of the super capacitor allows for placement below one of the Click boards, between the pin headers. This area was later kept free from vias and other overlapping components.

A cutout for the battery was added in the PCB, to allow for placing the battery inside the PCB, as the mounting brackets that were available could not fit inside the enclosure. The cutout can be seen in the PCB layout in Figure 2.9, and the rendered result in Figure 2.10. The PCB was produced on a LPKF E44 circuit board plotter.

The initial PCB versions shown in figures 2.7 and 2.8, are based on the strip board version. The first version included all the features of the original strip board version, with an added power switch. After a review of the available components, and requirements for battery size, the PCB was modified to V1.03, and finally a redesign was made to add better deep sleep functionality, and LoRa communication.

With the new microcontroller a cascade of changes were made to reduce the number of voltage dividers from 3 to 1, the micro SD card interface was replaced with a breakout board, as the chosen connector was not available. Finally an external interface for the I2C bus along with power rails were provided for the Mirkoe Illuminance Click board. The PCB design that was used for manufacturing is shown in Figure 2.9

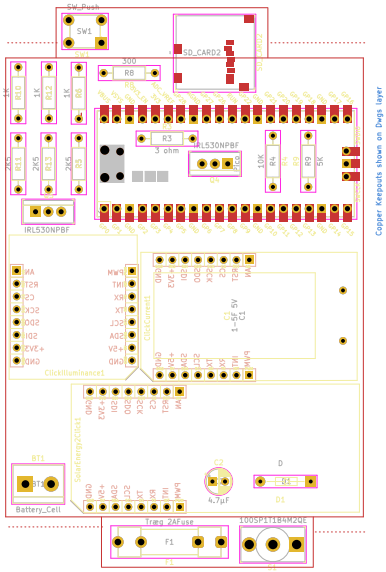


Figure 2.7. PCB V1.02

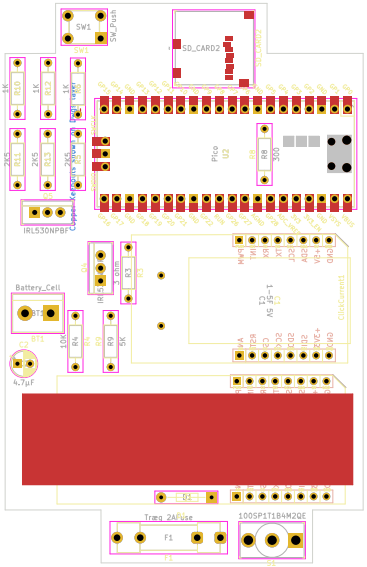


Figure 2.8. PCB V1.03

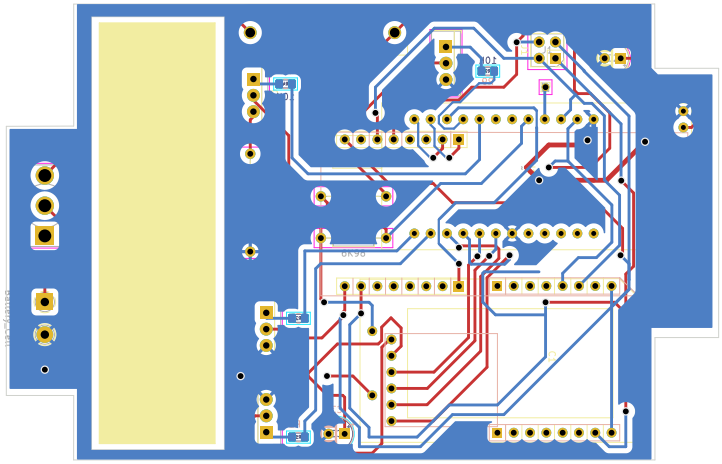


Figure 2.9. PCB V2.01

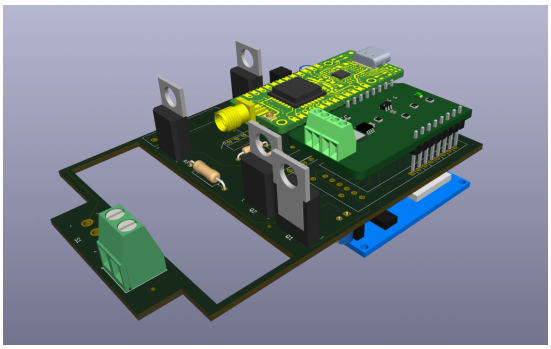


Figure 2.10. PCB 3D View Bottom. The micro controller is located in the center of the PCB, at its side is the Click Current, and on the same side is the majority of the supporting components e.g. for the power distribution

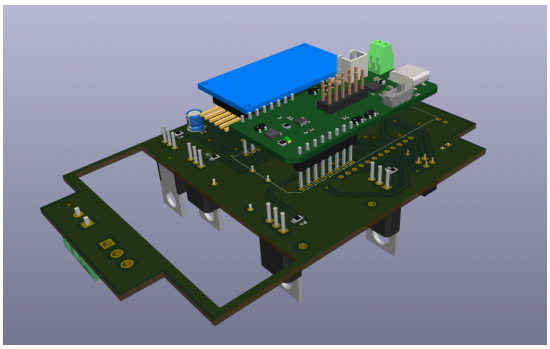


Figure 2.11. PCB 3D View Top. The blue component is the micro SD Card connector, the other add-on board is the Mikroe Solar Click 2 breakout board.

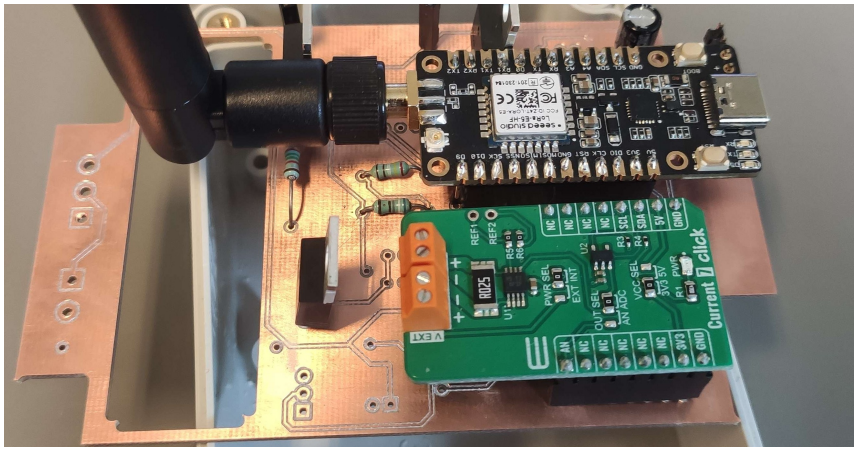


Figure 2.12. PCB Test Fit

PCB Patches/Modifications

Some modifications were made after production of the PCB, because they were easier to repair than, to redesign. These issues are outlined below

- Some through hole components and pin headers were routed to their respective mounting side, which means that soldering has to be done below the components. This was possible on some capacitors, but not on pin headers. The fix involved identifying a via along the signal path and using a wire to connect to the other side of the PCB.
- Gating the ground connection on the I2C devices, enabled power savings, but made disjoint ground planes between the battery, super capacitor and the solar cell. An alternative solution was implemented, where power was sourced from the MOS-FET Enable pin itself, as the power draw mas measured to be around 1/20 of the power limit on the digital pin.
- The ADC Harvester Connection was added through a wire on a pin header, as the harvester is not connected to the PCB, but with the Solar Energy Click board.
- The Super Capacitor was soldered to the PCB directly but also required a separate connection to the Solar Energy Click board.

- Some PCB trace widths were extremely thin after production, however they were still functional and the resistance was still acceptable. However for more reliable production, the traces between pin headers should be avoided.
- After the first operating period the board was modified to include pull down on the MOS-FET's as they were left in a floating state, when the micro controller went to sleep. The purpose was to reduce the random leakage through the MOS-FET gated drain and voltage dividers.

2.4 SD Logging

In the measurement campaign the logs are saved to a SD card integrated in the sensor node. The SD card functions as a backup of the data also being send using LoRa communication. This section outlines the challenges encountered and the various options considered when implementing the SD card for the sensor node is presented.

2.4.1 File Systems for Embedded SD card storage

A number of systems for filing logged data were tested on the sensor node. File system are used to allow for indexed lookup of data on a block device, such that read operations are not dependent on scanning the block device until a set of data is detected. While the file system uses some amount of overhead to store file indexes, it can also more efficiently use existing blocks, by rewriting incomplete blocks for better block level utilization, and make use of deduplication, compression and other schemes to improve storage efficiency and access latency. The application requirements for the sensor node consist of less than a hundred bytes of data every 5 minutes, with only a single read operation, when retrieving the logged data after testing is completed, thus the requirements suggest that a high performance filesystem is not required. However using a filesystem that has implementations on both the embedded sensor node and the computer used to process the logged data is hugely beneficial for data analysis.

A few options for implementing a file systems were available in tinygo, the options are discussed in the following sections.

CGO Littlefs and FAT32

The official community implementations of Littlefs and FAT32 were initially tested. The community implementations rely on C to GO (CGO) bindings, that wrap and link functions signatures in go to their respective C implementations. The littlefs implementations allowed for writing and reading files, however inspecting the SD card on a PC, with littlefs-fuse and raw memory, showed that littlefs was not able to sync data from its buffer to the SD card.

A second attempt was made using the FAT32 implementation from the same community implementation, the benefit of a FAT32 filesystem is that it is very common, and has great support on most operating systems. The added usability comes at the cost of reliability in the case of a power outage.

The test results for the FAT32 filesystem indicated that it was capable of write operations, as a filesystem header was observed on the SD card after initialization of the filesystem. The FAT32 CGO implementation was also able to write files to the filesystem, with no issues syncing the data to the SD card. However rebooting the device caused some issues, as a change in the generic filesystem interface was introduced in a recent version of GO that was not compatible with the C implementation, this change removed support for truncating files among other issues. Effectively no data could be overwritten on the SD Card, and when encountering duplicate filename the microcontroller was stuck in a busy waiting loop. This issue could be avoided, by ensuring unique filenames were always used, instead the uncertainty around other potential undiscovered bugs, led to the decision to seek out an alternative solution.

Fat32 Native GO

Since issues were discovered in the CGO implementations, an alternative implementation was sought ought. Using an implemented version of the FAT32 filesystem to native go using a transpiler on the C code, thanks to Whittingslow [17] implementation. However at the time of this project it had not seen wide or testing yet. The implementation was not integrated with the SD Card driver yet, so implementation of the read, write and erase - block interface functions were implemented, to test the implementation. While promising at first, some reliability issues were found after a few successive writes, as a suspected memory leak caused a stack overflow in the micro controller.

Block Device Storage

In the absence of a working filesystem, a more basic solution was needed. The logs can be encoded to bytes manually and stored on the SD Card in raw blocks of 512 Bytes. The concept of files is lost, and as such the maximum size a data point can take is 512 Bytes, and updating blocks becomes unreliable, however for our use case it is acceptable to append only. As such we rely on having a sufficient amount of blocks in the SD Card to map each data point to a unique block, selected by the wake up counter on the micro controller. To ensure there is sufficient capacity with low utilization of the bytes in each block, the amount of blocks in the SD Card is determined and compared to the expected amount of data points for the lifetime of the measurement campaign.

The information required can be found in the The CSD register of the SD Card, However a simpler method of dividing the labelled storage size by the block size gives the following:

$$32GiB/512B = 62500000blocks$$

The actual number of blocks may differ from this value, but not by an amount that would be an issue. That means there is space for around 60 million record or data points, when storing the data in 512B blocks. For a 14 day measurement campaign the maximum sample rate would be $51Hz$, and it is expected to be around once every 5 minutes.

2.4.2 SPI bit-bang

The SPI peripheral on the STM32, paired with standard library implementations, is frequently incompatible with flash storage devices, some workarounds are possible to utilize the SPI perihperal, however as multitasking and speed is not required, as the read/write workload is small, it was instead decided to use a bit-bang implementation for SPI, for better control and simplicity.

The bit bang method works by controlling 4 pins (chip select, data out, data in and clock) individually and sequentially to process read and write operations. On read operations bits are read and accumulated between cycles on the clock pin, while the chip enable / chip select pin is held. The write operations is also implemented in this scheme were small delays are introduced to ensure sufficient propagation time on the bus. Finally a function is defined that implements simultaneous transmit and receive, which is used when pipe-lining communication between the master and slave interfaces.

2.4.3 Sensor Node V1 Filesystem

In the first iteration of the sensor node, the FAT32 filesystem was used in micropython. The simple peripherals, combined with wide adoption of and first party support for micropython meant that stable implementations of SD card, SPI and filesystem interfaces were available.

The micropython filesystem event with its first party support, had issues with seeking in files, and as such common data formats such as a csv file was not possible. Instead each data point had to be seperated out into ordered files.

The naming scheme for files was deterministic, meaning that on a reboot of the microcontroller files on the SD card would be overwritten from the beginning. An alternative improved choice could have been made to ensure that in the boot sequence a new series prefix for the files would be written allowing for data retention during brownouts while operating.

2.5 Click

Click boards from MikroElektronika is used for some of the functionality in the sensor node. The click board system is a modular system of plug and play add on boards. Each click board is equipped with various sensors, communication or other modules, commonly each board serve one specific purpose. The boards are then standardized to fit the mikroBUS interface

that consist of a 16 pin layout. For this project every board used is using the I2C interface that are part of the layout on the mikroBUS. With the standard layout it is easy to prototype and implement using these boards [18].

MikroElektronika also provides a extensive library of code for the click boards that makes it easy to start using them. As the library is provided in C these are ported to be used both in version 1 and 2 of the sensor node and this means a porting from C to Python and Go for each board used.

In the sensor node these three boards is used:

1. Click Current 7
2. Solar Energy 2 Click
3. Illuminance click

The following sections go into detail about each board, describing functionalities they contribute to the sensor node.

2.5.1 Click Current 7

The Click Current 7 is a add on board that add the functionality to measure the current on the solar cell harvesting energy. The click board features the INA282 Current shunt monitor that sense the voltage drops over the shunt.

The Click Current 7 is an add-on board that provides the functionality to measure the current in solar cell energy harvesting systems, such as the sensor node. This click board features the INA282 Current Shunt Monitor, which senses voltage drops across a shunt resistor. To provide accurate current measurements. The measurements are then provided either directly on a an analog pin on the microBUS or can be read using the I2C communication enabled by the MCP3221 chip.

2.5.2 Solar Energy 2

The Solar Energy 2 Click is an add on board designed to optimize the output of a solar cell (photovoltaic), and are able to work in ranges from microwatts to milliwatts. It utilizes the EM8500 Power Management Controller, one of key functions of the chip is the MPPT that track the solar cell to optimize the energy harvesting. The board also control the charging of an attached battery, in the sensor node this is a super capacitor [19, 20].

The click board is able to go into sleep mode where the MPPT still functions but the communication over the I2C is disabled, this is essential to being able to managing energy consumption. To wake from the sleep the board needs enough energy based on its start-up curve and if the capacitor falls below this point the I2C communication becomes unavailable,

but the MPPT should still continue working but without the I2C bus the status of this can not be retrieved.

2.5.3 Illuminance

The Illuminance click board is a component to make light measurements, offering high sensity and 16 bit resolution when detecting light intensity. The board uses the TSL2583 sensor that is a light to digital converter, meaning transforming light intensity into a digital signal. The sensor utilizes two analog to digital converrs to convert the photodiode currents into these digital outputs. These currents represent the irradiance measured, capturing both visible and infrared light [21, 22].

In the project the click board placed next to the solar cell to measure the light intensity that hit the sensor node. The board utilizes the I2C connection on the microBUS to communicate with the microcontroller.

2.6 Solar Cell

In this project, a solar cell is used as the source of power generation. The solar cell is not expected to be in direct sun light, and as such it will be operating outside its design parameters. This is also further worsened, as the selected solar cell is designed for use in space. The solar cell is an Azur Space 3G30A Solar Cell assembly, which is a triple junction solar cell, with an area of $30.18cm^2$ and an efficiency of 29%. A solar cell with multiple junctions, have an improved efficiency, as they, in simplified terms, can collect energy from a wider range of frequencies of light.

One of the aspects of interest, when placing the solar cell near a tunnel entrance, is to determine, how the amount of collected energy changes, as the amount of light is varying throughout the day. The different light sources daylight, artificial road illumination and vehicle headlights, each have their own time varying characteristics. Furthermore each type of light may have different frequency spectrum's, which could have an impact on the amount of energy that can be collected from each light given the same intensity.

There are many different types and designs for solar cells. However for this low light use case, where a sensor should be operating, ideally maintenance free, a significant investment can be done. This is why the triple junction cell was used in this project, as it is difficult to produce it is also expensive. For electrical grid power generation mono and polychrystalline solar panels are used because they are cheaper to manufacture, and the amount of area available is much greater, as such a loss of efficiency can be compensated by a larger installation while still being a far cheaper solution. One final type of cell that could be of interest are amorphous silicon solar cells, as they are renowned for being more efficient in low or diffuse light, as they are less dependent on the angle of arrival of light, than mono and poly-crystalline solar cells. The

efficiency improvement of multi junction solar cells compared to single junction solar cells, is that current is only generated for incident light with energy (frequency dependent) that exceeds the band gap of the junction. This sets a lower bound on the frequencies of light that can be collected, yet there is also an issue at the high side of the spectrum, as incident light with too much energy, end up loosing their excess energy as heat instead of current. The spectral response, beyond serving as the basis for choosing efficient solar cells is not within the scope of the project.

Multi-junction solar cells layer several "diode" junctions of varying materials to both increase the total available spectrum of energy, but also to reduce the maximum residual energy in the light.

2.6.1 Diode Law

The power output of the cells are characterized by a IV (current-voltage) curve, showing the relationship between current and voltage for a given temperature and illumination. The curve can be used to determine the optimal impedance (characteristic resistance) that, maximizes the power output of the cell. There are 3 interesting points on an IV curve, firstly the the open circuit describes the maximum voltage of the cell when it is not connected to a circuit, secondly the short circuit current typically describes the maximum current that the cell can provide when it is connected with a resistance close to 0Ω, the short circuit current is also frequently interchanged with the photo current. Finally the point of maximum power output is the point on the curve where the product of the current and voltage is at its highest value, this is usually at the "knee" of the curve.

The IV curve is the result of modeling a solar cell using the ideal diode law Eq. (2.1), yet it is not sufficient to model the power generation, as the light generated current is not take into account, the full model is shown in Eq. (2.2). The solar cell acts in reverse bias when power is generated. The diode blocks displaced electrons from passing through the diode in reverse, and they are instead forced to traverse an attached electrical circuit. When the diode is operating in reverse bias it has a nonzero leakage current which is known as the reverse saturation current or I_S. The reverse saturation current in normal operating conditions is insignificant.

$$I = I_s(\mathrm{e}^{\frac{qU}{kT_k}} - 1) \tag{2.1}$$

Where: I_s is the reverse saturation current, which is the current when the diode is in reverse bias, e.g. a negative voltage less than the reverse breakdown voltage is applied. The reverse saturation current should be very close to zero. In addition to

Shockley diode equation

$$I = I_L - I_s(\mathrm{e}^{\frac{qU}{nkT_k}} - 1) \tag{2.2}$$

Where I_L is the light generated current / short circuit current. The short circuit current varies with V_t and the light incident on the solar cell. e is eulers number, q is the physical constant elementary charge, n is the ideality factor, which aids in modeling the behaviour of the diode outside the ideal regions, e.g. at low light and extreme light, U is the voltage drop across the solar cell and T_k is the temperature of the diode in the kelvin scale. The ideality factor is a value greater than or equal to 1.

Several models for estimating the short circuit current exist, primarily focusing on including properties of reflection and wavelength dependent absorption, and power varying with wavelength. However for this project, simplifying assumptions are made concerning angle of arrival [23] and the spectrum of light.

The method used to estimate the current assumes, that the power spectrum is identical to the manufacturers specifications, and adjusts for the change in temperature. The model method divides out the reference irradiance and multiplies by the new irradiance, and finally applies the temperature coefficient to achieve the new estimated current. The reference current for Azur space solar cell assemblies specify an AM0 spectrum, as there is no attenuation from the atmosphere in their use case. However and AM1.5 or worse spectrum should be the expected spectrum for the purpose of this project. After applying the current estimation method, the model was accepted as being accurate enough with a calibration for lux values measured in direct sunlight.

The equation used for calculating the short circuit current of the solar cell uses the reference values for short circuit current and temperature under standardized test conditions, the new value is then derived from the insolation denoted by E_{new} and the temperature difference between the reference conditions and the new value. TC_{ISC} is the short circuit current temperature coefficient.

$$I_L = \frac{I_{ref}}{E_{ref}} * E_{new} * (1 + TC_{ISC} * (T - T0)) \tag{2.3}$$

2.6.2 Estimating IV Curves

Good estimates of an IV cures requires knowledge of the saturation current, irradiance, temperature and the ideality factor. To demonstrate the variations in the IV curve for each parameter an illustration is made. Figure 2.13 shows 4 single parameter variations to the baseline IV curve shown in blue along the red line. Changes in the ideality factor n have the effect of shifting the curve horizontally, as can be seen by the green line. varying the reverse saturation current I_0, shifts the IV curve to a higher voltage and slightly higher current. The temperature coefficient is small, as can be seen by the small deviation in the red curve with a temperature difference of 20C. Finally the insolation has the effect of shifting the IV curve in, primarily the vertical axis, but also the horizontal axis.

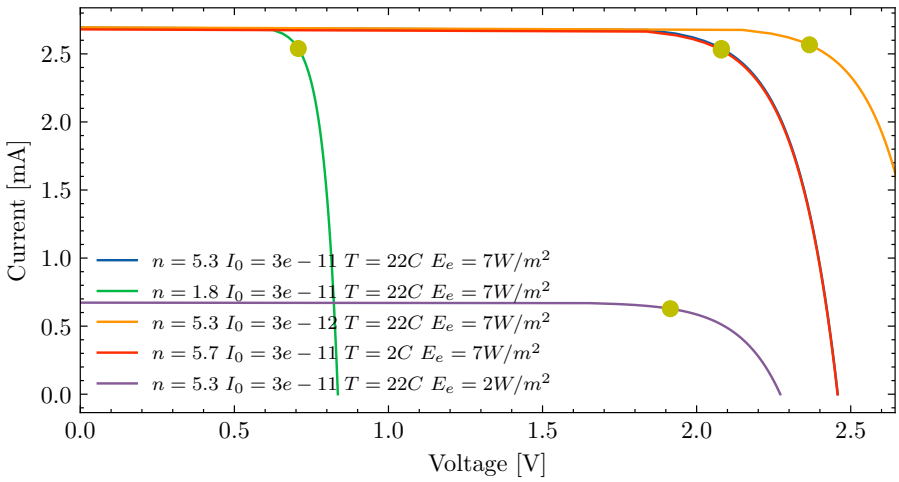


Figure 2.13. The IV curve for 5 sets of parameters are shown, to illustrate the effect of changing each parameter. The red and blue line have only a small variation

VOC-ISC Method

The VOC-ISC method presented by Meyer [24] can be used to inexpensively determine the ideality factor and the reverse saturation current from measurements of the open VOC (circuit voltage) and ISC (short circuit current) over a range of illumination of the solar cell assembly. If the curve is log linear, then the assumptions may hold. This method was used to determine the ideality factor and reverse saturation current for low light irradiance. The results from the experiment are shown in Fig. 2.14, and The experiment setup is described in test journal Appendix A on page 80. using the slope and intercept of the linear fit from 2.14, the ideality factor and reverse saturation current can be calculated[24] using equations 2.5 and 2.6

$$v_t = \frac{kT_k}{q} \tag{2.4}$$

$$n = \frac{a}{v_t} \tag{2.5}$$

$$I_0 = e^{\frac{-b}{nv_t}}) \tag{2.6}$$

$$T0 = 22v_t = 0.025V = \frac{k(273.15 + T0)}{q} \tag{2.7}$$

Using the equations for ideality factor and reverse saturation current, we get

$$n = 5.29$$

$$I_0 = 3.187 * 10^{-11}$$

As expected the ideality factor n is significantly larger than 1, indicating that the cell is not operating as an ideal diode.

2.6.3 Maximum Power Point

Using good estimates of the ideality factor, reverse saturation current, temperature and irradiance, along with some reference values from the datasheet for short circuit current under

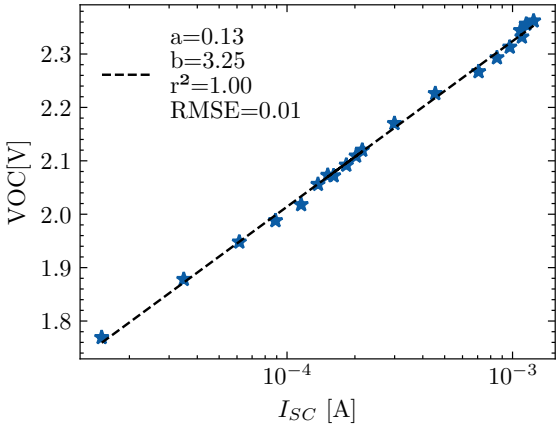


Figure 2.14. The figure shows the well known log linear relationship between the open circuit voltage and the short circuit current. The data fits well to the linear regression, as indicated by the r² value of 1.00, which also indicates that the cell is not damaged[24]. a denotes the slope of the curve, and b the intercept.

laboratory conditions, we can calculate the IV curve for arbitrary values of temperature and irradiance. The function for power can be found as either a function of voltage or current e.g. $P(v) = I(v) * v$, yet it has no good solution when determining the maximum point on the function. The Newton Rhapson method is commonly used to determine the maximum power point. However for the analysis in this project the power function was instead sampled at a high resolution to determine the maximum power. An attempt at solving the function analytically was done using the lambert W function however, when using sampled real world data the function was too unstable to provide reliable results.

2.6.4 Vehicle measurements

The energy captured from vehicle headlights is expected to be small, as their power output is small relative to sunlight. Halogen H1 light bulbs were tested. Each light bulb is rated at 55W, which is a large amount of power compared to what is required by the sensor node. However the chain of power loss, means that only a little power is left at the solar cell. Initially the bulb has some inefficiency in converting power to light, as it is heating a filament thread, and not all of the heat is dissipated as light. Secondly the light emitting from the bulb is shaped in a dipped beam, so most of the time the light will not be captured by the solar cell, and when it is captured it will only represented a fraction of the irradiated light.

A simple test was conducted to determine the effect of changing the horizontal and vertical distance from a car's headlight and the solar cell. The tests were conducted separately along a vertical and horizontal axis, the solar cell was oriented directly towards the vehicle, at the same height as the bulb. The data collected for the vertical measurements suggested, due to lack of tools to create a large height difference, That the beam was very narrow along the vertical axis, and that less than a meter above the height of the headlamp almost no energy could be collected. Figure 2.15 Shows the effect that changes in the horizontal distance to the car has on the light collected current. The current is measured through a constant resistance of 5Ohm,

connected in series with multi-meter, that is measuring the current.

2.7 Tinygo Ecosystem

The section explores the use of the Tinygo ecosystem in developing the sensor node firmware. Initially, it examines the initialization functions, these functions are a feature in the go programming language designed to configure settings at startup. When employed in firmware, these functions can introduce some side effects, which are discussed in detail here. Secondly an overview of drivers and external libraries are presented. Some of the drivers were modified to include functionality important in the firmware, these modifications are highlighted within this section.

2.7.1 Initialization Function Side Effects

When exiting deep sleep, several peripherals are reset while RAM is retained; as such, any setup of the peripherals needs to be performed again. Some packages in Go use a hook to run initializing (init) functions before the main function is scheduled for execution[25, 26]. The init functions are called as a part of the import system in go, and these init functions are not called when the system wakes from sleep. Thus, multiple issues are present after a sleep period. Additionally, all init functions are, by definition, private functions and, as such, can not be called from external code.

The issue is resolved by examining each package to identify any init functions and determine whether any initialization process has been overridden by going to sleep and waking again.

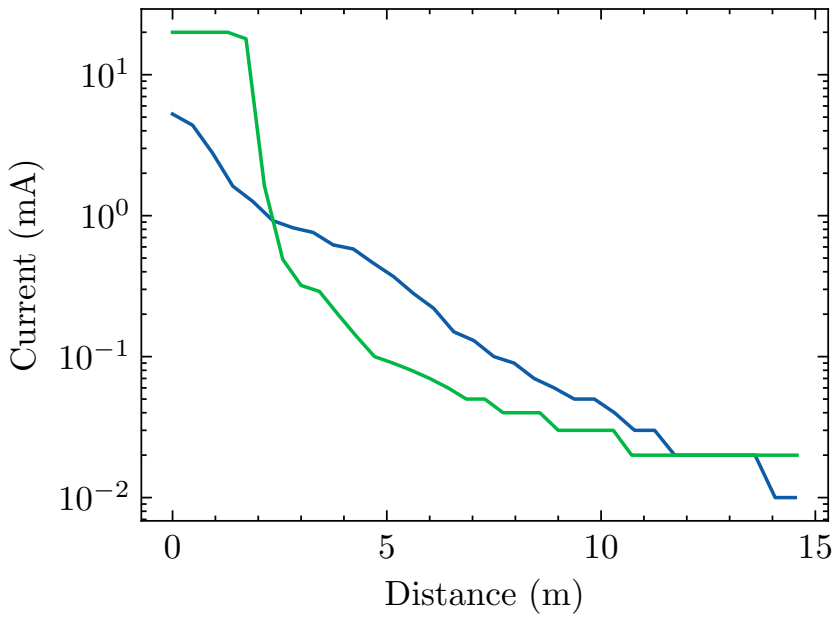


Figure 2.15. The relationship between light generated current and distance for a fixed resistance. The light source is a dipped beam car headlight, and the distance is varied in the horizontal plane.

Then, there is a need to redo this initialization upon wake or before using the package again. A careful inspection of the init functions of concern allows for implementing reinitilization code either in the imported package itself, or in the main loop, depending on the complexity of the required initialization and whether local variables are modified.

These init functions were identified in the libraries for LoRaWAN, the Drivers package and the stm32 runtime / platform support code itself, and an illustration of a straightforward workaround is given in 2.1.

Sys Tick Timer Hooks

One detailed example of the init function issue is the sys tick timer. After waking the processor, the systick timer is reset, and the interrupt handler has to be re-mapped to the TinyGO scheduler, as it is required by the go runtime for proper functionality of soft sleep and threads. As the runtime at the point of writing does not have native support for deepsleep on the STM32 platform, it is necessary to fork the tinygo stm32 runtime and modify it. The approach taken in this project was to expose the initialization logic in the runtime components to the application.

```
package main

import PackageWithRogueInit

func main() {
    for true {
        sleep(5 * time.Minutes)
        PackageWithRogueInit.Init()
    }
}
EOF

package PackageWithRogueInit

func init() { // lowercase private function
    initializeRadioPort()
    runOnlyOnce()
}

func Init() {  // Uppercase exported / public function
    initializeRadioPort()
}

```

Listing 2.1. Code Illustration of rogue init problem

2.7.2 Drivers And External Libraries

The firmware makes use of existing drivers, when available, and some modifications have been done in order to support the low power use case and LoRaWAN multiplexing of the radio transceiver. The libraries consist of a LoRaWAN packet decoder and packages included in the extended standard library "drivers" package.

- The drivers/lora package provides LoRa and SubGHzRadio functionality, such as abstractions between different LoRa transceiver versions.
  - The lora package was modified to accept the new coding rate channel config in the latest version of chirpstack, that is CR 4 5 instead of CR 4 7.
  - The driver initially did not support the receive windows from the LoRa specification [27] so these were implemented for use in the project.
  - To add support for multiplexing of the radio transceiver the method for setting active radio between, channels were changed. By default it always set a channel before sending, but with multiplexing only some settings needs to be updated, and this procedure were updated.
  - Additional data information were added to the driver to support storing the information for multiplexing.
- The brocaar/lorawan package provides functions to decode and encode LoRaWAN mac layer packets
- The drivers/sdcard package provides an implementation of the SD Card commands required to read, write and erase blocks. It also allows for reading the SD card metadata, such as the storage size. The sdcard package was forked and modified to accept a software SPI interface definition instead of the platform SPI implementation.

2.8 RTC, Power and battery Estimation

The microcontroller is designed for low power sleep, and it provides this functionality with a lot of choices. The most consequential choice is whether to use standby or stop mode functionality. Critically the stop mode provides ram and register retention, at the cost of a higher sleep current. The application can use ram retention, to retain the program state between sleep periods, while the standby mode requires the device to either include persistent storage or programming, that is independent of state. For this project stop mode was chosen, as no external memory data was available, and data retention was necessary to keep the LoRaWAN session keys available.

2.8.1 Power Profiler Kit II

The Power Profiler Kit II (PPK2) device can measure and optionally supply current for the setup during testing. The device can measure currents from 200 nA to 1A, making it suitable for testing the current draw from the sensor node [28, 29].

The PPK2 is utilized as the power supply for the sensor node, enabling precise real-time monitoring of the current drawn. It is beneficial in multiple different tests, whether sleep current is measured or figuring out the amperage over time, and figuring out how long time measurements take, combining these to calculate battery life.

2.8.2 MCU Debug Register

STM32WlEJ5C microcontroller has registers to override the power control for the processor when debugging sleep mode, setting bits in the DBGMCU_CR register Allows the debugger to stay connected during sleep modes, which means wake-up events can be captured using the debugger. This is particularly useful when configuring the wake up sources. However for correct validation of the power consumption this mode must be disabled.

2.8.3 WFE/WFI

The microcontroller can enter sleep through the "WFE" (Wait For Event) and "WFI" (Wait For Interrupt) instructions, and optionally on return from an interrupt service routine. The WFE instruction, makes the core sleep until it receives an event or interrupt, the event can be one of a predefined set of triggers typically mapped to peripherals on the microcontroller, an event may be sent from a seperate core of the microcontroller, allowing for cooperative energy saving mechanisms. The WFI instruction wakes the microcontroller when an interrupt reaches the core. This allows the microcontroller to wake from most of the peripherals, and from any external pin configured as an interrupt. After an interrupt fires, the microcontroller will wake and enter the corresponding interrupt service routine, where the interrupt source will need to be cleared. If the interrupt is not cleared, then the next WFI instruction will act as a "NOP" instruction instead. The more complicated nature of synchronizing the clearing of interrupts between the seperately clocked RTC and microcontroller, led to the choice of using event based wakeup for deepsleep.

2.8.4 RTC Configuration

This section details the chosen configuration for the RTC peripheral on the STM32

- The clock prescalers were set to 255 and 127, meaning that when the RTC is fed with the LSE (Low Speed External) clock which oscilates at 32.768 kHz the resulting divided clock is approximately one second $\frac{32.768kHz}{255 \cdot 127} = 0.988s$.
- Having a clock increment period of close to 1 second, makes it easy to program the wake-up timer. However it also works to increase maximum duration that the microcontroller can sleep uninterrupted. This is because the timer has a fixed maximum counter, so a tradeoff must be done between the maximum sleep period and the minimum sleep period combiend with the resolution.

- The divided clock is fed to 3 configurable timers, 2 alarms and one wakeup timer. Each timer can wake up the processor. The wakeup timer features an auto reload function, that simplifies the application code, as it does not have to handle interrupts and reconfigure the timer at every wakeup.
- The RTC wake-up clock can be selected in the stop mode wake-up clock register, and was set to the 16MHz High Speed Internal clock (HSI16). After wakeup, the code initializes the main High Speed External clock, as it matches the correct bus timings for using SPI and I2C with tinygo. There may be some benefit in using a lower speed clock like the MSI or HSI clocks in more mature firmware.
- The wake up timer logic seems to run at every rtc clock cycle, and as such the minimum time before the wake up source is cleared is also 1-2 periods of the clock. meaning that the minimum time that the processor must stay awake is also 1-2 seconds. This could create some issues in artificially increasing the duty cycle of the sleep schedule. However there are several alternative solutions, such as using the lowest clock speed after all processing has completed, or using a higher resolution clock, that can sleep for several cycles between each application loop execution. Finally for more demanding applications an external RTC may be desired. External RTC can also feature better drift by automatically adjusting the clock frequency.

2.8.5 Power-Up/Down Sequencing

In order to ensure the correct order of initialization and deinitialization of peripherals, a simple interface was was made, such that each module in the firmware provided functions to initialize and deinitialize, with an associated run-level. This topology of layers is a simple solution, and as such it does not allow for running with the minimum amount of required peripherals at all times. One solution to the above issue could be implementing a graph of peripheral dependencies. However it was not required for this project, as the tasks could be run synchronously without spending too much time awake.

2.9 LoRaWAN

This section explore how LoRaWAN is used for transmitting data from the sensor node to a network server. The section describe the LoRa and the LoRaWAN protocol is used focusing on the design choices and the implementation and setup to get LoRaWAN working on the sensor node. This include both the firmware and the infrastructure.

One of the most important parts of when testing LoRa with the sensor node is to have it appear as multiple devices for LoRa network, this makes it easy to distinct between different settings, and each device have one specific use case. The sensor node will emulate multiple devices for this purpose, see subsection 2.9.7 for more about this implementation.

2.9.1 EU868 P channel

The 868MHz region in the EU, has a band designated by P, which is in the frequency range 869.400MHz to 869.650MHz (250KHz Bandwidth), with a max ERP (Effective Radiated Power) of 500mW and <10% Duty Cycle or polite spectrum access. The polite spectrum access is using CCA (Clear Channel Asessment) or a similar mechanism, to avoid interrupting other users of the channel. band plan originates from EC 2017/1438/EU (54) [30]. Since the channel has the highest duty cycle among the channels available to LoRa it would be useful, as it provides a much higher throughout than channels with 1% or 0.1% duty cycle. However the P channel is, for typical LoRaWAN implementations, used as the downlink channel for the gateway exclusively. In order to determine whether the P channel could be repurposed as an uplink chanenl, in an environment with little radio activity, the hardware had to be tested. After reading the documentation for the sensor node, gateway and the network server a likely solution was found. Howevere the changes were determined to be too extensive, and the throughput on the 1% duty cycle band sufficient, as the communication scenario did not include the long range spreading factors above SF10.

2.9.2 MAC commands

A crucial component of LoRaWAN networks is the MAC (Media Access Control) commands, which enable the transmission of instructions to and from end devices. These commands, sent over the air, allow the network server to maintain and dynamically adjust the fleet of end devices without physical interventions. The MAC commands are instrumental in managing end devices by allowing changes to parameters such as DR (Data rate), power level, and channel configuration. This section introduces the structure of a LoRaWAN packet that can include MAC commands, and introduces the functionality of various MAC commands available in the specification, that are relevant for the sensor node.

Before going into how the MAC commands are sent, the format of a LoRa frame is presented. Figure 2.16 illustrates how all LoRaWAN uplink and downlink packets are structured. They all consist of a MHDR (MAC header), followed by the MAC Payload and they end with a MIC (message integrity code). The MHDR is one octet long as seen in the illustration, the MAC Payload are at least 7 octets long but can be bigger, and the MIC is 4 octets long. This makes the smallest packet size in LoRaWAN 12 octets long.

When joining the Network the MACPayload is replaced by the Join Request or the Join Accept commands as indicated, these commands have their own unique structure that is tailored for joining the network. All other frames uses the MAC Payload that consist a frame header, illustrated by the blue color, that consist of a 4 octet end device address, a frame control octet, a two octet frame counter and up to 15 octets of frame options to carry MAC commands. The remainder of the MAC payload consist of an optional FPort and then the FRMPayload that can be up to N octets. N is regional specific and depends on the data rate being used. In the EU868 band N ranges from 51 to 241 octets [31, 27].

This makes the maximum length of packet to be 255 octets long, this is important as the amount of data changes the time on air, and thus the energy consumption. Later in the project the length of packets are specified to be constant when sending data, but are within these boundaries.

The FRM Payload can also include MAC commands, this depends on if the command is piggybacked in the FOpts or a separate data frame for the MAC command. A MAC command consists of a CID (command identifier) that are one octet, specifying the command, followed by command specific payload.

An interesting MAC command for the sensor node is the NewChannelReq, NewChannelAns that are used to create or modify the radio channel. Another interesting MAC command is the LinkADRReq, LinkADRAns where the network server request the end device to change data rate, TX power, redundancy, or channel mask. In these command the network server sends the request and the end device acknowledges with an answer. The sensor node handling of these MAC commands in the sensor node can be seen in Listing 2.4 that shows how the end device are waiting for a LinkADRReq, updating the settings and respond with a LinkADRAns to either acknowledge the changes or decline the changes from the network server [27].

2.9.3 Adaptive data rate

The LoRaWAN ADR (Adaptive Data Rate) algorithm is a tool for changing the data rate and transmitting power to optimize airtime and energy consumption in the network. The algorithm runs on the network server and will indicate changes to the end device using MAC commands, specifically the ADRRequest command [27].

In LoRa, the ADR algorithm controls the transmit power and the data rate that can be mapped to different spreading factors and bandwidths. The data rates in use are regulated by regional

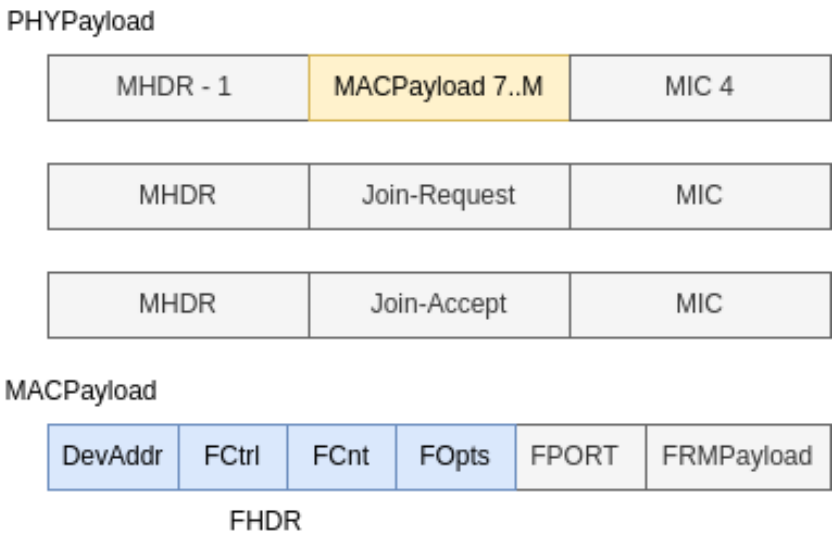


Figure 2.16. The payload structure for a LoRaWAN transmission showing the PHYPayload

parameters and the ones for this project is specified in EU868 and the data rates and their mappings are shown in table 2.3 below [31, 32].

DR0	SF12	BW125
DR1	SF11	BW125
DR2	SF10	BW125
DR3	SF9	BW125
DR4	SF8	BW125
DR6	SF7	BW125

Table 2.3. Mapping between the data rates and the spreading factor and bandwidth

The network server can use the MAC command ADRRequest to indicate to the update the data rate, TX power, redundancy, or channel mask used in the end device. There is no definitive ADR algorithm, but the essence is to calculate the link budget for the received packet, figure out if a lower setting is viable and change to it.

In the project a single ADR device is emulated on the sensor node to see the behavior of the default ADR algorithm in the selected LoRaWAN network server Chirpstack. In the network server it is also possible to add custom ADR algorithms, combining this with device emulation can make it possible to easily to test different ADR algorithms, this is possible with the implemented sensor node and infrastructure of this project but are not investigated further.

2.9.4 LoRaWAN network overview

The LoRaWAN network setup for this project include three main parts, the Sensor node as the end device, the gateway to receive and relay messages between the network server and the end node, and the LoRaWAN network server.

The sensor node sends packets that are received at the gateway that then forwards these packets to the network server, and the other way around when the network server transmit downlink packets. The message sequence diagram in figure 2.17 shows the flow of these LoRaWAN transmission from the sensor to the network server and with the optional downlink from the network server that can send commands and messages to the sensor.

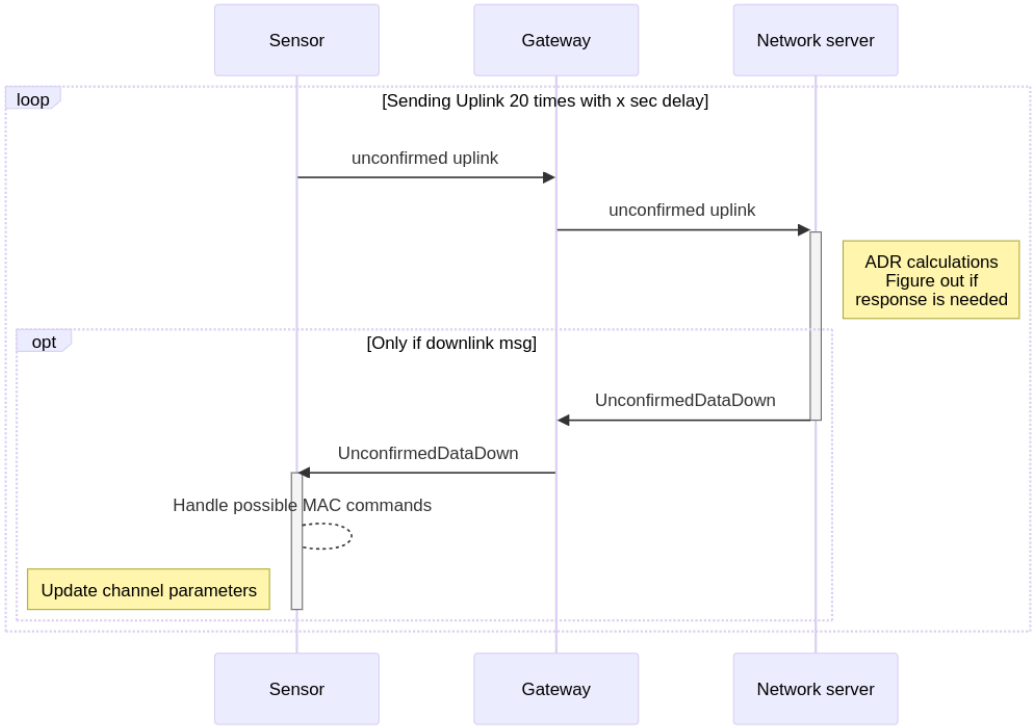


Figure 2.17. Showing the message sequence between the sensor, gateway and network server when a transmission is made

2.9.5 Chirpstack

ChirpStack is an open-source LoRaWAN network server that can be self-hosted. The network server handles the network and facilitates communication with end devices and gateways. In Chirpstack it is possible to setup multiple gateways and devices that can communicate with the network server.

ChirpStack is the LoRaWAN network server that stores and responds to a transmission that arrive in from the device if the transmission needs a response or the network server has a downlink message for the device. The message for the device could be for a Join request answer or telling the device about settings to use when ADR is enabled.

ChirpStack comes with a web interface that enables gateways and device management. Another management level is added by using device profiles that determine the version of LoRaWAN to be used; in this project, version 1.03 is used.

Deployment

In the project, the main infrastructure runs on a Claaudia instance [33]. The main part of the infrastructure is the Chripstack network server, which is open-source and provides setup instructions for running it using Docker. In the Claaudia instance, the only prerequisite

then needed is Docker, and Docker Compose as the Chirpstack server is running multiple containers.

The docker environment makes it easy to setup on a new server, by only providing the necessary configurations files. Before starting the server, the configuration files for the network server are modified. The modifications are made to enable the use of custom payload decoders, custom ADR algorithms, and update the channel bands for the regional.

In development the deployment were local without the use of Claaudia, and in the change from using a local machine to using a server on claaudia provides its difficulties. These changes in the deployment from testing environment and the deployment for the measuring campaign is presented in Figure 2.18.

Transitioning from a local machine to a Claaudia server presented several challenges. During the deployment of the Claaudia instance firewall rules for connections were set, this setup allow connection on specific ports, and the ports for SSH, HTTP and the network server were all added to give access. The gateway is using UDP on port 1700 to communicate with the network server, and while the gateway were connected to the internet from the university using AAU network it was not able to send UDP packets to the network server as they were filtered. This filtration was verified, after hours of investigating with the gateway configuration, by sending UDP test packets from alternative sources. The command used to send UDP packet was netcat, specifically (nc -u [networkserver ip] 1700) in combination with (tcpdump udp and port 1700) on the network server to monitor any traffic on the UDP port.

As a long-term solution, reassigning the subnet of the gateway could prevent UDP packet filtering. However since the gateway installation is temporary for the project, it was decided to use a tunnel network. With a tunnel network the gateway was able to connect to the network server, with the additional benefit of having a static IP for the tunnel so when moving the gateway no additional changes were required.

Custom payload decoder

Chirpstack makes it possible to develop a custom payload decoder in Javascript, turning the payload into a readable object directly inside the web interface. The custom codec has an input

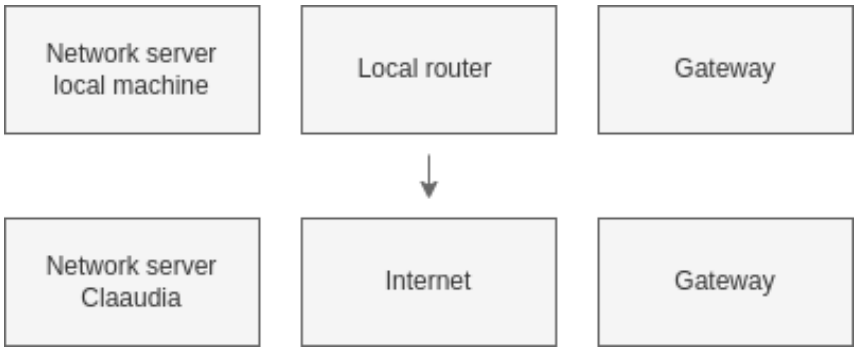


Figure 2.18. Change in the deployment between development and measuring campaign environment

object consisting of the uplink payload as a byte array and the fPort and device-configured variables. The custom decoder added to this project is used on the packet device when new information is gathered from that device. The decoder takes the payload and reads values from the byte array using a Javascript data view object, as can be seen in the decoder code in listing 2.2 the data view object makes it easy to extract values as float and integers, also stating the endianness of the value as this project sends the data with little-endian encoding.

Adding this decoder to Chirpstack and assigning it in the device profile for the packet device makes this final object available inside the event tab on the web ui when looking at incoming packets, making it easy to quickly see the values from new packets.

2.9.6 Dragino gateway

Dragino LPS8v2 is the gateway used in this project. The model is an indoor gateway and is preinstalled with Dragino's software, which exposes a web interface where settings for the network can be inserted. The gateway needs to know where the Chirpstack network server is to enable communication with it using UDP over port 1700. On the other end, it needs to set up its radios for the correct region, in this case, EU868. The device also supports running a local LoRaWAN server but is not in use as the Chirpstack is deployed at a separate Claaudia instance.

Telenor modem

As the gateway will be placed close to Limfjordstunnelen during the testing phase, and there is no network for this project that the gateway can join, an alternative method for getting network

```
function decodeUplink(input) {
    let buffer = new Uint8Array(input.bytes).buffer;
    let dataView = new DataView(buffer);
    let unixTimeStamp = dataView.getUint32(0, true)
    let datetime = new Date(unixTimeStamp * 1000)
    return {
      data: {
        unix: unixTimeStamp,
        date: datetime.toLocaleString(),
        v_bat: dataView.getFloat32(8, true),
        v_hrv: dataView.getFloat32(12, true),
        v_cap: dataView.getFloat32(16, true),
        current: dataView.getFloat32(20, true),
        drain: dataView.getInt8(24),
                 lux: dataView.getFloat32(25, true),
      }
    };
}
```

Listing 2.2. Custom decoder for packets received in Chirpstack

access is needed. In the project, a Telenor modem is installed and added to the Dragino gateway to add network connectivity. The network modem is from Huawei and needs to be set up to connect to the Telenor network. The commands in the following document how the modem is set up to be registered as a modem/network card and not a storage unit. The usb modeswitch command is used to change the state of a "multi-state" usb device, in this case as the windows firmware is included as a seperate storage device, the modem needs to be switched into the modem state.

```
$ apt install usb-modeswitch
$ usb_modeswitch -v 12d1 -p 14fe -M \
  '555342431234567800000000000001106200000100000000000000000000'
```

After it is detected as a modem, the tool ModemManager is used to set up the configurations for the Telenor network, to initially test the setup of the modem, the following command can be used to make a manual connection from the modem to the network and give the gateway an IP address

```
$ mmcli -m 0 --simple-connect="apn=internet" && dhclient wwan0
```

To verify the setup was successful, the current IP configuration status was checked, and with the modem working, an additional configuration was needed to make the modem connect to the Telenor network automatically on boot. This is achieved by using the tool NetworkManager, it was ued to add a "GSM" connection and to set it to auto-connect, this adds a profile that tells the network manager service how to initialize connections after booting. The commands required to do this are:

```
# nmcli connection add type gsm ifname "*" con-name "portal" apn "internet"
# nmcli radio wwan on
# nmcli connection modify portal connection.autoconnect yes
# nmcli connection up portal
```

Placement of gateway

Deciding the placing of a gateway for LoRaWAN can be challenging due to the wide range of devices that can connect to it. The devices may be spread across a large area, making it difficult to determine the optimal location for a single gateway or if multiple gateways are necessary. Additionally, the placement of the gateway and the sensors may require updating the parameters for each device to ensure proper connectivity and performance [34].

In this project, as there is only one physical device, the gateway should be placed within that device range but also at a location that allows it to be seen if the impact device emulation and

how moving traffic through the tunnel changes what packages can be received. Due to the limitations to the placement possibilities at the tunnel, the gateway is placed in the workshop right above the entrance to the tunnel marked with the red point, as seen in Figure 2.19, which also shows the placement of the sensor node marked with the blue point. The illustration also shows the distance between the sensor node and the gateway to be around 67 meter, measured by the line on the plot.

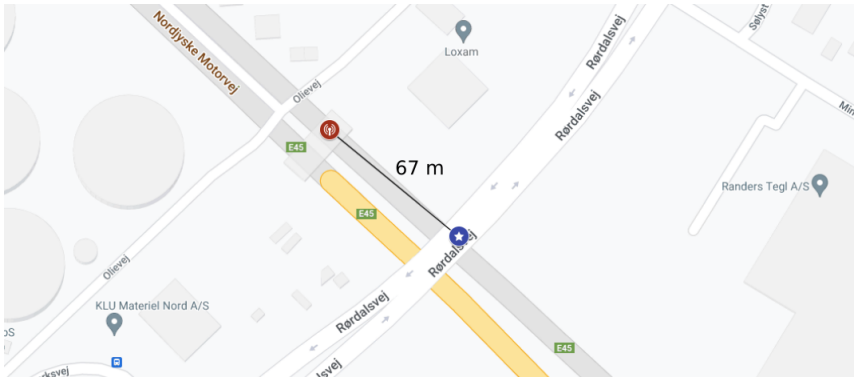


Figure 2.19. Illustrating the gateway's position with a red radio tower icon and highlighting the distance to the sensor node indicated with a blue star icon [35]

2.9.7 Device Emulation

Device emulation makes the sensor appear as multiple devices for the LoRaWAN network. Emulation allows the network to see the physical device as various different devices. This also allows for custom setup for each setting in the Chirpstack server.

In Chirpstack, every device is linked to a device profile that outlines its capabilities. The device profile can be modified to adjust the device's settings and capabilities, including the MAC version, custom decoder for payload, ADR algorithm, join method, etc.

Ten devices are emulated on the sensor node, and the devices' settings are chosen to match the best option for increasing power considering energy consumption, as described in our P8 Project. The device's settings follow the curve for power settings, as shown in Figure 1 from the P8 project. It shows that increasing the TX power first on the lowest SF and then increasing the SF when reaching the highest TX power. The ten devices spread out on the curve result in the settings for the emulated devices, as seen in Table 2.4.

2.9.8 Sensor node implementation

When integrating LoRaWAN support for the sensor node the implementation utilizes drivers developed for the SX126x transceiver alongside a specialized driver for LoRa communication developed for the Tinygo project. However, these drivers lacked necessary features for this project, dur to this the drivers are forked and extended to add the needs of this project. The enhancements of the drivers include the support for receiving data from the network server, and also device emulation that extend beyond the standard LoRaWAN specification.

Device id	Data rate	Transmit power
D1	DR2	20
D2	DR3	20
D3	DR4	20
D4	DR5	20
D5	DR5	17
D6	DR5	14
D7	DR5	11
D8	DR5	8
D9	DR5	5
D10	DR5	2

Table 2.4. Mapping of the devices to LoRa settings of data rate and transmit power

The most considerable extension to the driver code is the added code to enable receiving after transmission in the receive windows specified in LoRaWan. The code added to receive in R1 or R2 can be seen in code block 1. Here the function takes in if it should receive using R1 or R2, as the second receive window uses another radio settings as seen applied on line 13 in Listing 2.3.

This code can receive commands from the network server and also makes it possible to enable the adaptive data rate if the correct MAC commands are handled correctly. The MAC commands from the network server can be sent to the end device in two ways, either in a packet of its own in the FRMPayload or piggyback on another packet already queued for the device.

Thus the implementation needs to look for MAC commands in both locations and then act

```
1  // Listen downlink, window 1 for RX1 window config (uplink), everything else is RX2 window
   ↪ config
2  func ListenDownlink(window int) ([]uint8, error) {
3          if regionSettings == nil {
4                  return nil, ErrUndefinedRegionSettings
5          }
6          if window == 1 {
7                  ActiveRadio.SetIqMode(lora.IQInverted)
8                  resp, err := ActiveRadio.Rx(LORA_RX_TIMEOUT)
9                  if err == nil && resp != nil {
10                         return resp, nil
11                 }
12         } else {
13                 applyChannelConfig(regionSettings.Rx2Channel())
14                 ActiveRadio.SetIqMode(lora.IQInverted)
15                 resp, err := ActiveRadio.Rx(LORA_RX_TIMEOUT)
16                 if err == nil && resp != nil {
17                         return resp, nil
18                 }
19         }
20         return nil, nil
21 }
```

Listing 2.3. Golang code from the sensor that enable receiving packets from the network server

according to the specification and the information inside the packet. The most interesting command is the ADRRequestCommand that requests the device to update its current parameters for sending, and answer with an acknowledgment in form of a ADRResponse command. To get the command from the received packet and onto a format in the sensor node the LoRaWAN package from brocaar [36] that provides the stucture from the specification.

The handling of the received LoRAWAN packets can be seen in code block 2.4. The code receive the payload from the receive window. The code shown only support handling a LinkADRReq to update the radio settings with the given, and respond with an acknowledgment.

Another vital change for enabling device emulation in the sensor node involves updating the radio settings for every transmission. The settings are applied every time a new packet is sent, as these need to be parsed for each emulated device instead of keeping a single state. The way the changes update the radio can be seen in Listing 2.4 line 27-32, where the active radio settings are updated; the same method of updating the radio is applied whenever a device is sending a packet. This is possible as each device holds the radio settings needed to update the radio.

2.10 Rørdal Bridge

As seen in the placement of the gateway in section 2.9.6 and on Figure 2.19 the sensor is placed under the Rørdal Bridge. The sensor node is mounting under the bridge that is just before the tunnel, and this section details how the sensor node was enclosed and mounted at the bridge.

2.10.1 Mounting / Enclosure

To mount the sensor node at the bridge the sensor node is placed inside a cabinet, the cabinet can be seen on Figure 2.3 where the strip board is being test fittet. This enclosure is modified with the addition of a 3D printed ring to extend the height of it to add additional needed room for the second version of the sensor node. Outside on the sensor node the Solar cell mounted on top of the cabinet using a mount made by laser cutting out a acrylic sheet. The mounting of both the solar cell and the strip board version near the camera on the underside of the bridge can be seen in Figure 2.4.

The second version of the sensor node is installed using a two by four placed over the driving lanes under the bridge. The installation of the sensor can be seen in Figure 2.20. The device is mounted to have the solar cell pointing towards the cars to optimize potential energy from the vehicle even though the expected energy is low.

```
    func HandleMACCommands(payloads []ll.Payload, device LoraDevice) {
        for _, payload := range payloads {
                mc, ok := payload.(*ll.MACCommand)
                if !ok {
                        fmt.Println("*MACCommand expected")
                }
                if mc.CID == ll.LinkADRReq {
                        adrReq, ok := mc.Payload.(*ll.LinkADRReqPayload)
                        if !ok {
                                fmt.Println("*LinkADRReqPayload expected")
                        }
                        // Update to new settings.
                        acceptDatarate := 12-adrReq.DataRate >= 7
                        if acceptDatarate {
                              // lorawan.ActiveRadio.SetSpreadingFactor(12 - adrReq.DataRate)
                                // lorawan.ActiveRadio.SetTxPower(int8(adrReq.TXPower))
                                device.DeviceChannelSettings.SpreadingFactor = 12 -
                                ↪  adrReq.DataRate
                                device.DeviceChannelSettings.TxPower = int8(adrReq.TXPower)
                        }
                        // Send reponse to Network server
                        adrans := ll.LinkADRAnsPayload{
                                ChannelMaskACK: true,
                                DataRateACK:    acceptDatarate,
                                PowerACK:       true,
                        }
                        by, _ := adrans.MarshalBinary()
                        data := append([]uint8{0x03}, by...)
                        if device.DeviceChannelSettings.SpreadingFactor >= 7 &&
                        ↪  device.DeviceChannelSettings.SpreadingFactor <= 12 {
                                lorawan.ActiveRadio.SetSpreadingFactor(
                                    device.DeviceChannelSettings.SpreadingFactor
                                )
                        }
                        if device.DeviceChannelSettings.TxPower >= -2 &&
                        ↪  device.DeviceChannelSettings.SpreadingFactor <= 22 {
                         lorawan.ActiveRadio.SetTxPower(device.DeviceChannelSettings.TxPower)
                        }
                        err := lorawan.SendUplinkMac(data, device.Session)
                        if err != nil {
                                println("Failed to send uplink ack mac")
                        }
                }
        }
}
```

Listing 2.4. Golang code from the sensor that handle the MAC commands received from the network server

Figure 2.20. The sensor node mounted using a two by four placed over the driving lanes under the bridge

Energy Harvesting Results 3

This chapter shows the results from the energy harvesting module of the sensor node. It shows the collection of data from the solar cell and how this is utilized to find the cumulative energy harvested over the measurement campaign.

3.1 Data Collection

The data was collected using the LoRaWAN connection on the gateway, at the same interval of 5 minutes. However the data uplink was also sent without retry on reception failures, and as such a small amount of measurement data is missing. The intervals of missing data is insignificant compared to the rate of change of the collected data. Finally the LoRaWAN data collection includes a checksum, that was verified for all the collected data carrying packets, which means the integrity of the data was verified.

The presented data spans 2 measurement intervals, the periods of which are outlined in table 3.1. A secondary source of data exists on the SD card[1]. However the analysis is based on the data that was collected via LoRaWAN as it was available for analysis immediately after installation.

Label	Period-Start	Period-End
"First"	2024-04-25	2024-05-15
"Second"	2024-05-18	2024-05-29

Table 3.1. Labels and time periods for each data collection series. An update of the sensor node was performed between May 15. and May 17.

The sensor node included redundant measurements to evaluate different measurement methods, and to better validate the resulting data. The sources and their tradeoffs are listed.

- Solar cell voltage and current samples, can be sampled periodically, in order to get a measure of the power input for the MPPT, the sample frequency is low, and as such it does not capture high frequency variations well.
- Super capacitor voltage, can be used to estimate the energy collected, as the voltage has a direct relationship with the energy stored in the capacitor. It is also a time integrating component, which means that it does not require frequent sampling, to capture short

[1] The SD card was not missing samples in random intervals, but as the sensor was able to run for a longer time period than the SD card could be supplied its time range is not as extensive

intermittent periods of energy. However it also includes all the inefficiencies in the MPPT, charging and self discharge.

- Lux measurements can be used to estimate the power output of the cell, given that the cell is well characterized.

3.2 First Measurement Campaign

During the first measurement campaign the lux meter was found to be faulty, as all the readings were 0, later identified as an I2C bus error, this failure was likely common among all the I2C devices on the board, including the ammeter.

With the Ammeter and Lux meter measurements missing for the first measurement period the focus will instead be on the capacitor and cell voltage readings. The capacitor voltage is shown in figure 3.1, which shows that the capacitor drain was higher than the charge rate until it had discharged to around 600mV, at which point the drain rate change, and the capacitor voltage was stabilized. At the end of the measurement campaign the capacitor voltage began increasing. However with the apparent unpredictable behaviour of the charge and discharge sources, the data would not function as a basis for determining the energy harvesting potential.

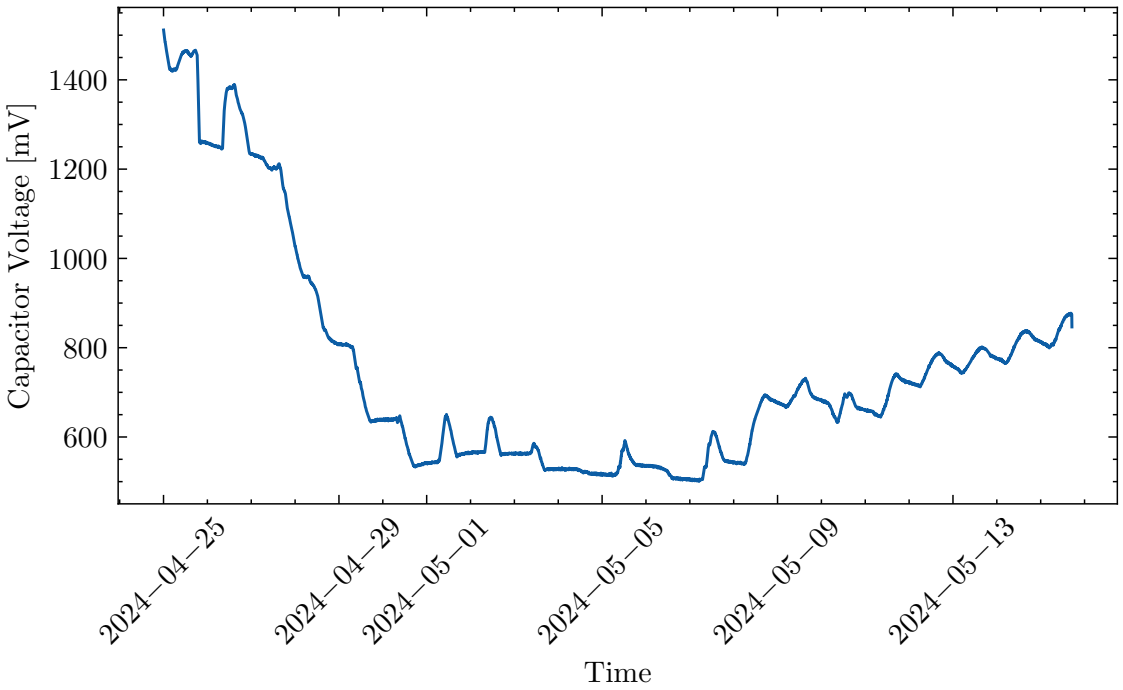


Figure 3.1. First: Super Capacitor voltage

The solar cell voltage was the second data source available during the first measurement campaign. The solar cell voltage measurements can be seen in figure 3.2, the measurements represent the voltage of the solar cell at the maximum power point, as determined by the MPPT in the EM8500. The solar cell voltage could not trivially be converted to a power reading without the current measurement.

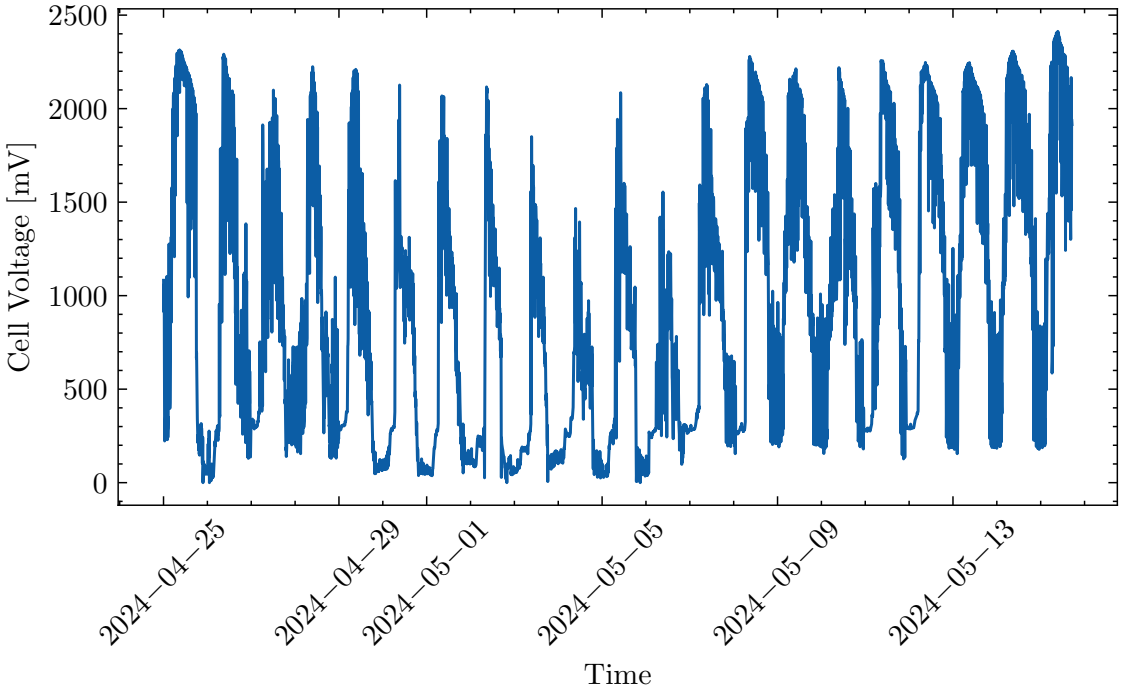


Figure 3.2. First: Solar cell voltage readings, the readings are taken during MPPT operation

As a recourse for the lacking data, an alternative solution was decided upon, which involves estimating the current of the cell from the voltage reading, and the assumption that it is operating at max power. However performing this mapping requires good estimations of the ideality factor, reverse saturation current and temperature of the solar cell. This lead to the decision of ending the measurement campaign. With the sensor node retrieved, a second iteration of the sensor node was made, with the goal of remedying the issues with the I2C bus and capacitor drain, and along with it the solar cell was characterized, the findings of which can be located in Journal in Appendix A on page 80.

3.3 Mapping of voltage data

Estimates of the current were found using the assumption that the voltages at maximum power, at a particular temperature were unique, and as such could be found through the current voltage (IV) curve. The IV curve was calculated using the reference insolation, the measured reverse saturation current and ideality factor, and the dry air temperature, as reported by DMI at the nearest station. The temperature measurements are shown in figure 3.3.

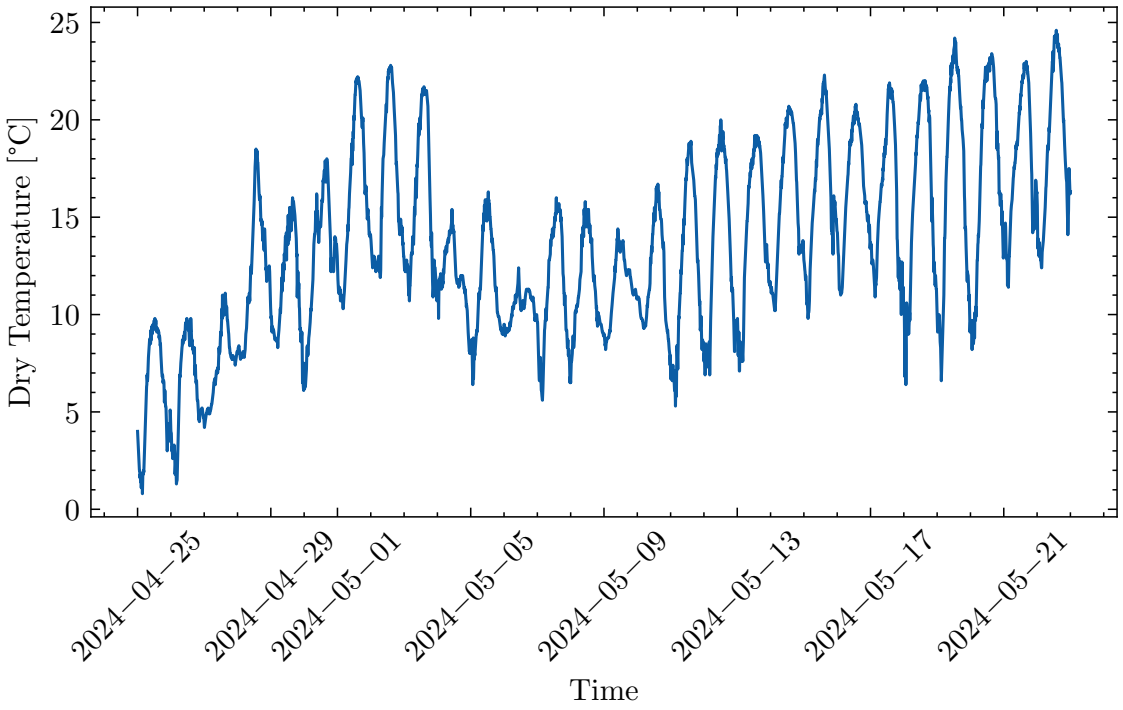


Figure 3.3. The dry air temperature (DBT) reported by the Danish Meteorological Institute at Aalborg Airport, the closest station, during the measurement campaign

Data structures used

- voltage bins. A table of the bins representing the range of digitized voltages.
- temperature bins. A table of all the unique temperature observations during the measurement campaign
- IV MAP, hash based lookup for pairs of digitized voltage at max power (V_{mp}) and temperature to current at max power (I_{mp})

In order to map the voltages to a particular IV curve a lookup table was constructed, spanning a a valid range of insolation and the range of temperatures recorded during the measurement period. for each pair of insolation and temperature, the cell characteristics were calculated, the cell characteristics, temperature and insolation values were then used to create function for the IV curve. The IV curve maximum power was then determined using sampling, the current at max power is then mapped and keyed by the voltage and temperature, in order to perform the 2D point mapping correctly a second index of the temperature to voltage mapping is made for each temperature . The process is repeated for all pairs of insolation and temperature pairs.

The exact floating point match is unlikely to be found with a reasonable amount of computation time, and the voltage measurements are therefore discretized to the natural numbers range in units of mV. In order to use the map to find the power output, the temperature data is aligned and merged with the voltage readings, to get a timeseries of voltage and temperature pairs. The pairs are then mapped using the IV map and the temperature lookup table, to finally retrieve the estimated power output of the cell.

The finally transformation from power to energy is performed by integrating over the timedelta

between packets, this could also be a source of error, depending on the ratio of power between transient sources and stable sources. The sunlight is expected to vary slowly, however it could start to vary quickly in patchy cloud cover, and finally the power from vehicle lighting will be hard to estimate with the low sample rate.

The relationship between power output and voltage is shown in figure 3.4, which shows that the power is not a linear scaling of the voltage, but that a mapping exists.

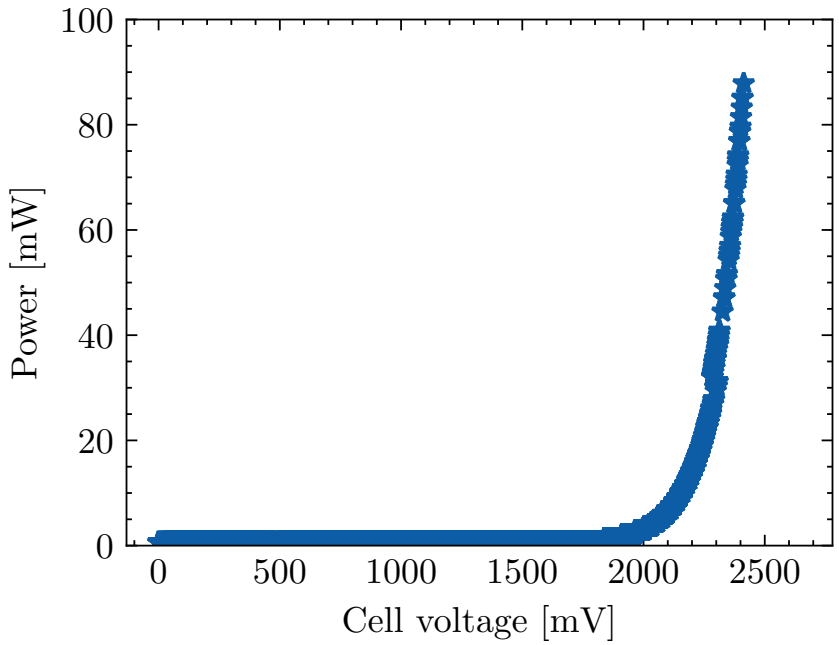


Figure 3.4. The relationship between voltage at max power, and the power produced by the cell. The figures shows the non linear relationship.

3.4 Second Measurement Campaign

The second measurement campaign was performed in order to get validating sources of data for both the LoRa communication and the energy harvesting. The I2C bus error was fixed, and the lux meter and current sensor were accessible. However the range of current sensor was found to be insufficient as data clipping occurred, causing it to only report valid values during peak hours of operation. The clipping of the current sensor is shown in figure. 3.6 where the minimal value recorded is 6.5mA.

During the sensor node refit a set of reference measurements were made using laboratory equipment for validation. The test leads and the fit are shown in figure 3.5. The equipment readouts were close to the values obtained from the sensor node, indicating that the current estimations were successful.

After a period of 5-6 days, on the 22. of May the voltage readings of the sensor node failed, as is shown in Fig. 3.7, and the issue was not identified at the time of the data analysis. The values beyond the 22. of May are considered invalid, when it comes to further analysis in this chapter

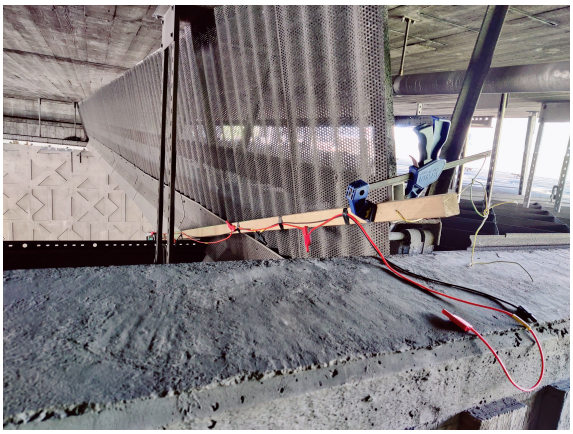


Figure 3.5. The sensor node was temporarily attached with test leads, to allow for current measurement using laboratoy equipment in the real world scenario

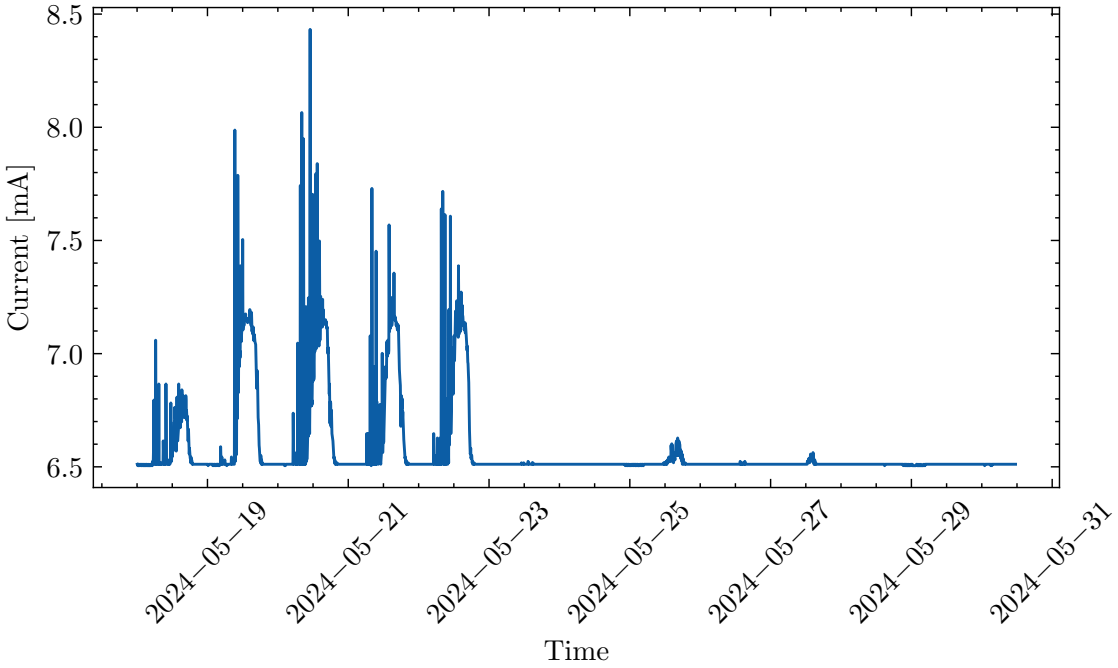


Figure 3.6. Second: measurements from the current sensor, where clipping of the data is seen at 6.5mA

and the following network calculus chapter.

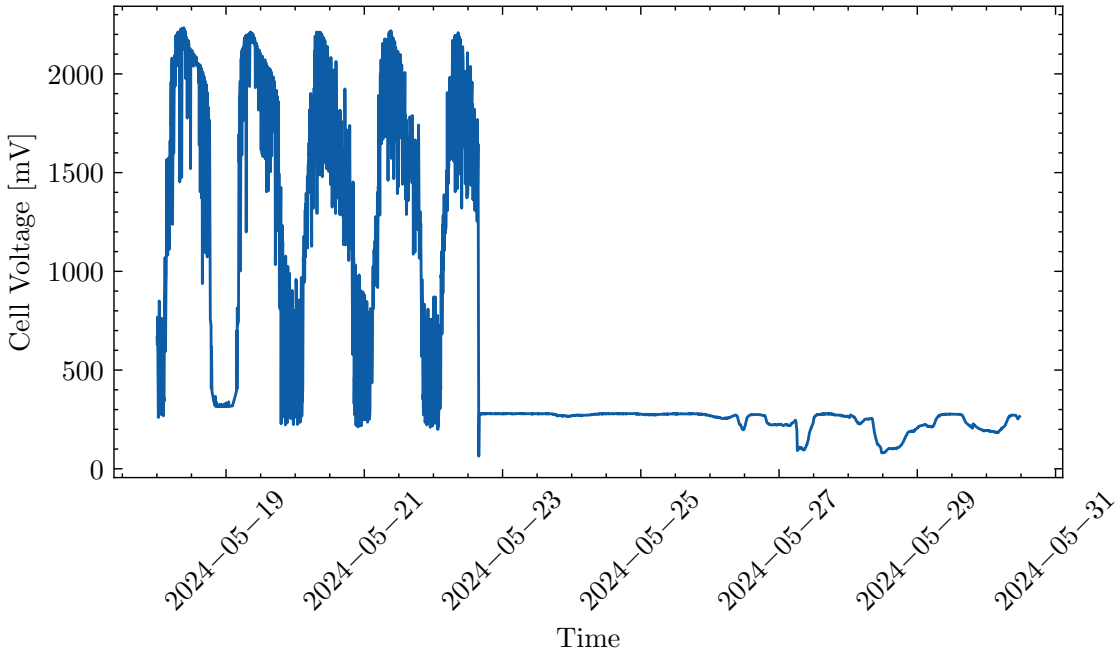


Figure 3.7. Second: the solar cell voltage during MPPT operation. Initially the cell voltage follows the pattern of daylight. After may 22. the voltage readings stay relatively constant, and stop following the daylight variations. This is consider invalid

The data collected from the lux meter is shown in figure 3.8, where the illumination of the lux meter, which has the same orientation as the solar cell, is observed to measure daylight variations. The initial measurements show asymmetric peaks, indicating that the sunlight is obstructed more in the afternoon than in the morning. The lux meter data also shows that the daylight intensity can vary between days, and this is likely due to variations in cloud cover. Finally the night time measurements give an indication of the intensity of the artificial illumination, the measured value was on average 2 lux, with only small variations.

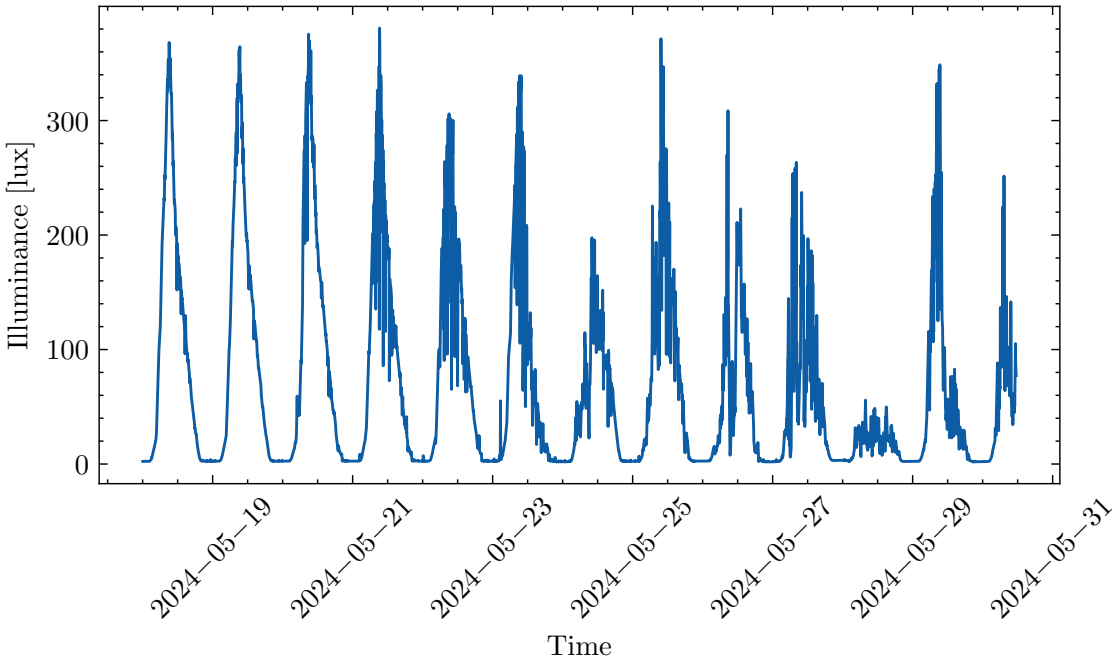


Figure 3.8. Second: Measurements from the Lux meter. The lux varies with daylight at the peaks. The night time lux was measured at 2 lux on average

The capacitor voltage displayed similar anomalous behaviour after ensuring the MOS-FET gated voltage divider and drain functionality were properly deactivated during microcontroler sleep.

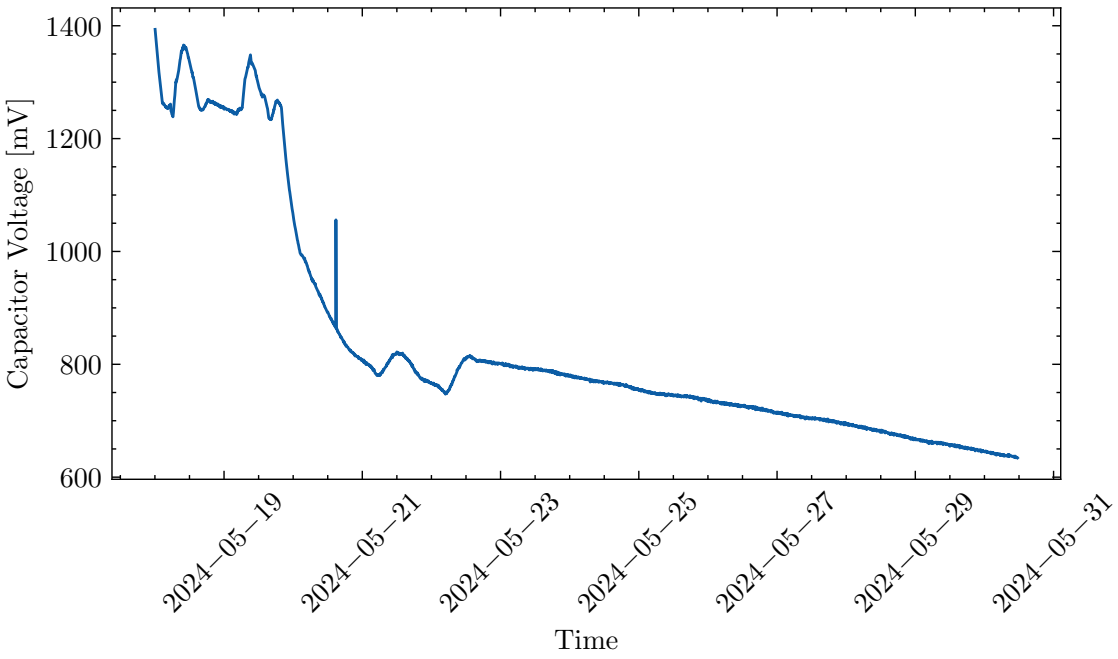


Figure 3.9. Second: the capacitor voltage displays the same erratic behaviour, no observations can be with regards to changes around 600mV, as the measurement period is too short.

3.5 Cumulative Energy Arrival

The cumulative energy arrival is calculated as a basis for analysis in the NC chapter. The energy arrivals on days, with low sun exposure are seen to rely more heavily on the low and long lasting current provided by the artificial lighting.

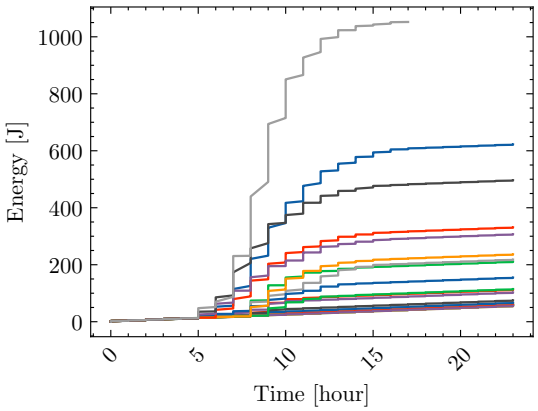


Figure 3.10. First measurement period

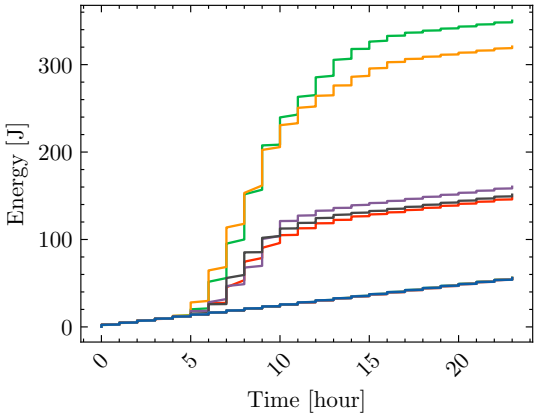


Figure 3.11. Second measurement period

Figure 3.12. Cumulative Energy Arrivals pr hour of day. Each series represents the sum of energy collected in a day. The majority of the collected energy is within the hours of 06 to 16

LoRa performance results 4

This chapter begins with a description of the data received in the LoRa experiments. Followed by the results of the testing completed at the Limfjords tunnel. The reliability of each emulated device will be evaluated with and without repetitions to find the most efficient transmit pattern when taking energy into consideration.

4.1 Device metrics

The LoRa transceiver on the sensor node is transmitting and the network server is collecting data. For each device the following metrics are transmitted and captured:

1. Received
2. RSSI (Received signal strength indicator)
3. SNR (Signal-to-noise ratio)
4. Timestamp

Then there are known parameters known for each configuration, these include:

1. Data rate
2. Energy consumption. (P8 Project)
3. Payload length

4.2 Received threshold

The threshold for the lowest link budget device received in every period is shown in Figure 4.1. The y axis is the devices emulated with different configurations, where 10 is the lowest link budget configuration with SF of 7 and an Tx power of 2, then going up in Tx power and SF as going down in numbers, this follows the configurations as seen in Table Table 2.4 on page 40. In the X axis is the time, there are 5 minutes between every point in the plot.

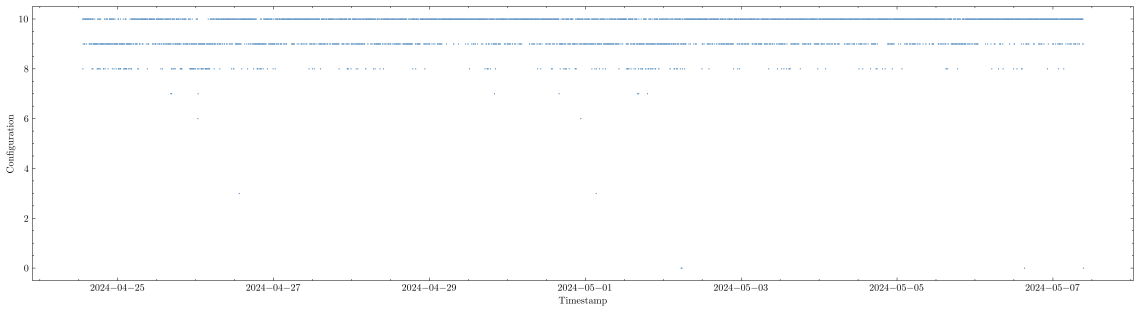


Figure 4.1. Showing the lowest link budget device to receive at every data collection point, over the whole measurement campaign

Figure 4.2 is the same structure as Figure 4.1, but the X-axis is limited to a single day.

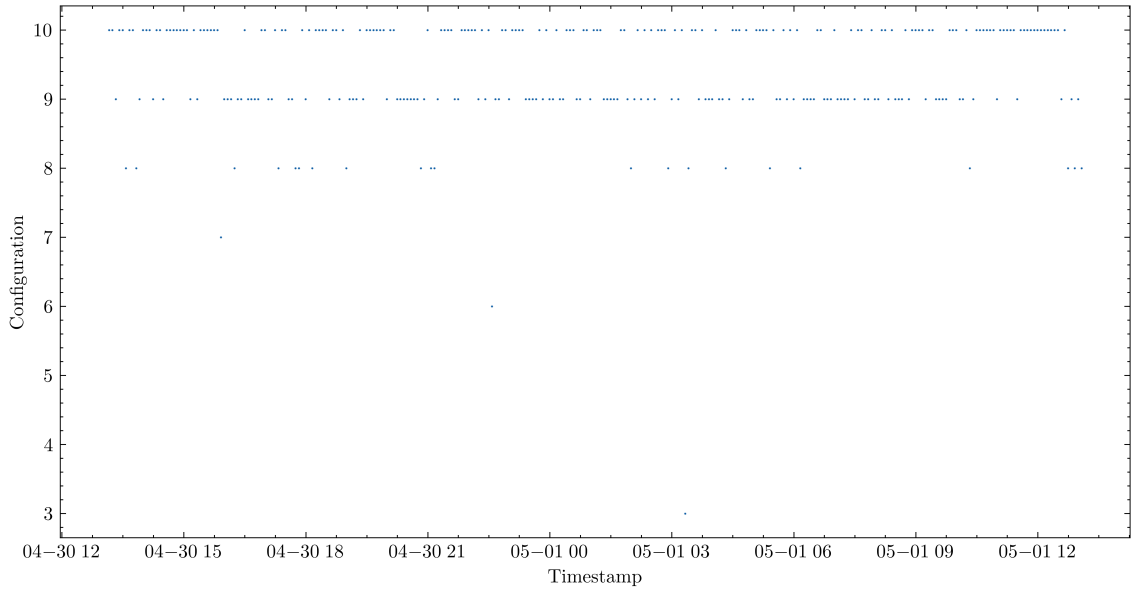


Figure 4.2. Shows the lowest link budget device to receive at every data collection point, over a representative day

Figures 4.1 and 4.2 shows that the threshold for receiving a packet from the sensor node at every timestamp are mostly at device 10 or 9 and sometimes device 8. These results indicates that the gateway and sensor node is to closely located to see the full range of devices that can not reach and devices that are reliable.

By looking at the received SNR at at each device as seen in figure 4.3 that shows the SNR recorded over the day. In the figure every new line represent a new day and the the lines are plotted for configuration D3. And when looking at the devices from D1 to D10 in this testing period the SNR recorded is between $8dB$ and $-8dB$.

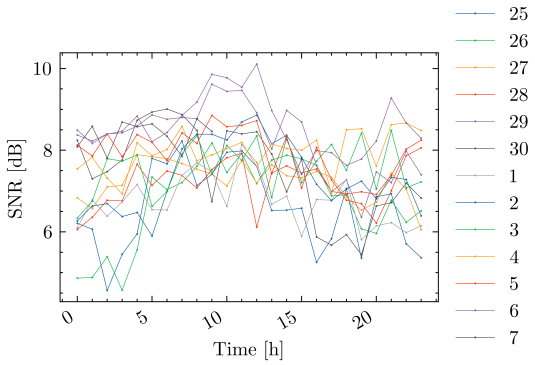


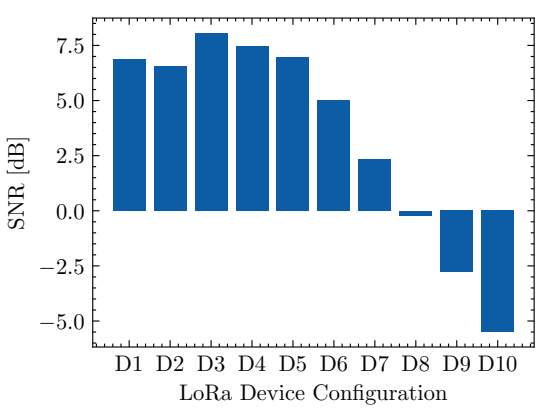
Figure 4.3. Displays the SNR over each day

By adding 6 dB attenuation to the second phase of testing with LoRa the aim is to move the threshold of reliable reception down a few configurations to avoid clipping of the cutoff point on the configurations. With this change the SNR is expected to decrease by 6dB for all configurations. Comparing the mean SNR from the day immediately before and after the attenuator was added, the drop in SNR was close to 6 dB for configurations D1 to D8, The SNR difference for a subset of those configurations are shown in Table 4.1.

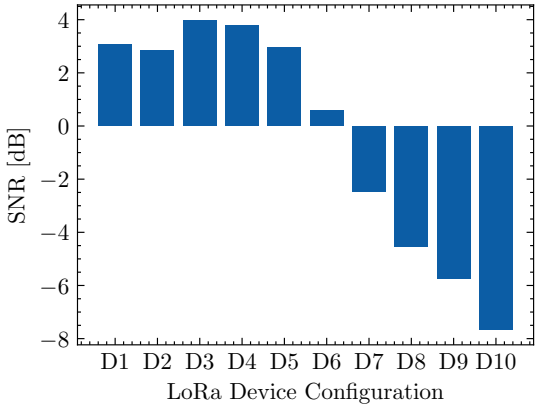
Device	SNR Before	SNR After	Difference
D1	7.3	1.6	5.7
D3	8.52	2.2	6.32
D6	5.63	-1.1	6.73
D8	0.67	-5.73	6.4

Table 4.1. SNR before and after adding 6 db attenuation

Figure 4.4 shows the results of the added attenuation for the 2 datasets, where the threshold for the lowest link budget device that reliable receive is moved as seen to be focused around D7 and D8 instead of D9 and D10. The SNR difference in Figure 4.4 is less than 6dB, which is caused by a long term increase in the SNR in the 2 weeks after reinstalling the sensor node.



Period: 2024/4/27-2024/5/15 without attenuation



Period: 2024/5/17-2024/5/27 +6dB attenuation

Figure 4.4. Mean SNR per emulated device for all packets received in the experiment

4.3 Device packet delivery ratio

Figures 4.5 and 4.6 displays the PDR (Packet Delivery Ratio) achieved for each device in the LoRa testing. Figure 4.5 shows the reliability from the first testing phase before adding 6dB attenuator to the antenna, while Figure 4.6 shows the reliability for each configuration afterwards.

In both plots it can be observed that the reliability is higher with the devices with the higher link budget. Figure 4.5 shows that the reliability only starts falling of in device D9 and D10 where the D1-D8 is all around 95% PDR. It is also worth noticing that D1 and D2 are slightly lower than D3-D7 even though the devices have higher link budget, this might be caused by the settings in these are modifying the spreading factor resulting in packets that have a longer Time on Air and thus also more susceptible for interference with other packets.

By adding the 6dB attenuator to the antenna before getting new data changes where the falloff in PDR begins. In Figure 4.6 it can be seen that device D10 is nearly never received, and then also the full impact of the device range tested, as the figure now include a device that is not received that often.

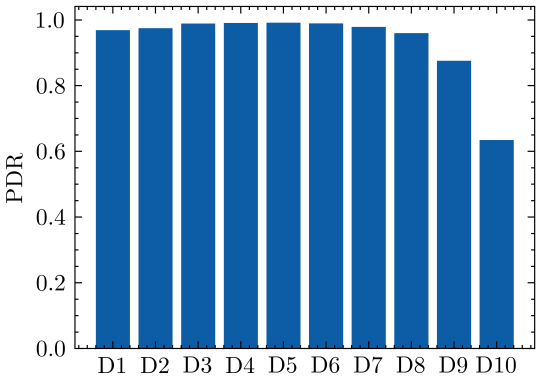


Figure 4.5. Packet delivery rate for different tested configurations, without attenuation

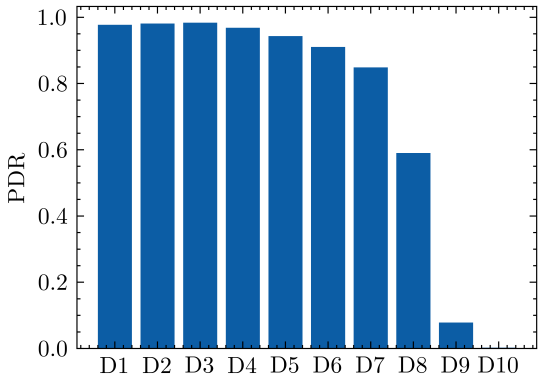


Figure 4.6. Packet delivery rate for different tested configurations, with 6 dB attenuation

Each device is using a certain amount of energy per transmission, determined energy consumption that is depended on the device settings and the Time on Air of the packet. Our 8th semester project made a comprehensive test of the energy consumption used by LoRa with different settings, utilizing the results from that project makes it possible to fit a model for the energy consumption based on the device setting and the Time on Air. The method for finding the models for each setting used in this project can be seen in Appendix B on page 83 .

To calculate the Time on Air for a packet and for all the emulated the devices the the equations below are used, starting by finding the time for a symbol as seen in Equation (4.1). This is used to multiple to the preamble length and the payload length to get $T_{preabmle}$ and $T_{payload}$ respectively as seen in equation (4.2) and (4.4) [37].

$$T_{sym} = \frac{1}{Rs} \tag{4.1}$$

$$T_{preamble} = (n_{preamble} + 4,25) * T_{sym} \tag{4.2}$$

To get the number of symbols in the payload is shown in Equation (4.3), that is depending on the radio setting.

$$n_{payload} = 8 + \max(\lceil \frac{(8PL - 4SF + 28 + 16CRC - 20IH)}{4(SF - 2DE)} \rceil (CR + 4), 0) \tag{4.3}$$

Where:

PL	Payload bytes (1 to 255)
SF	Spreading factor (6 to 12)
IH	Header enabled (0 or 1)
DE	Low data rate optimize (0 or 1)
CR	Coding rate (1 corresponding to 4/5, 4 to 4/8)

$$T_{payload} = n_{payload} * T_{sym} \tag{4.4}$$

After finding the time on air for the preamble and the payload the total packet time is given by equation (4.5), where it is the sum of the two.

$$T_{packet} = T_{preamble} + T_{payload} \tag{4.5}$$

As this depends on the radio setting and the payload size, this can be calculated for every emulated device. In the calculation of time on air for each device the payload bytes is fixed, and only the settings difference over the devices take effect.

To identify the trade off between the PDR and the energy consumption used to send these with these devices, the PDR is normalized with the energy consumption. The equation to normalize the PDR can be seen in (4.6).

$$\frac{PDR}{EnergyConsumption} \tag{4.6}$$

This normalization of PDR is shown in the Figures 4.7 and 4.8. In Figure 4.7 it can be seen that even though the PDR of D10 is lower than D9 it still gives the highest value after being normalized and this is because the energy consumption compensates for this. In the second phase where the PDR of D10 were lower the best result after the normalization is D7 as can be seen in Figure 4.8 with a score just above 200.

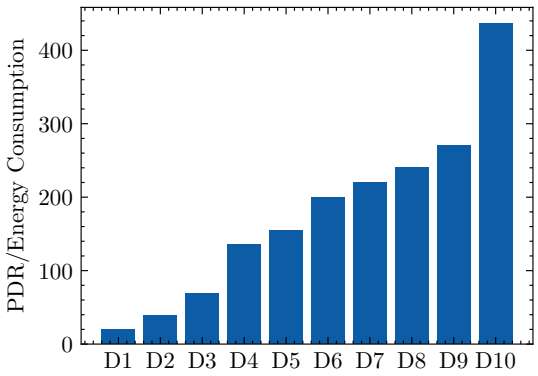


Figure 4.7. Packet delivery rate normalized with energy consumption, without attenuation for each device

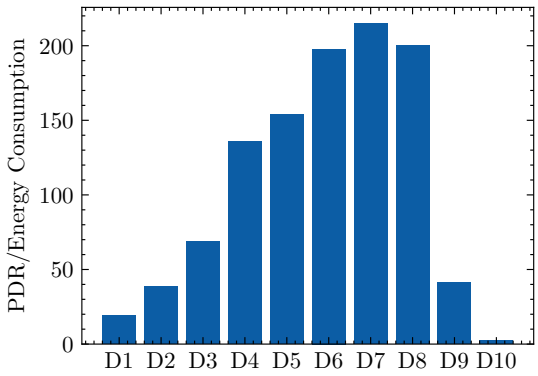


Figure 4.8. Packet delivery rate normalized with energy consumption, with attenuation for each device

These figures use the observed reliability and as it is desired to have a packet delivery rate greater than a certain value, and one method to improve the reliability of the device is to add repetitions of the transmission. When using repetitions the new reliability for a device R_{new} with 2 repetitions can be calculated as:

$$R_{new} = 1 - ((1 - R) * (1 - R))$$

With a desired packet delivery rate of 95% or greater, the number of repetitions required can be calculated, and Table 4.2 shows the amount of repetitions needed for each device to get achieve a PDR to be greater than the 95%. The following is using the data from after adding 6 dB attenuation.

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
Reliability	0.971	0.982	0.980	0.964	0.997	0.993	0.977	0.974	0.994	0.986
Repetitions	1	1	1	1	2	2	2	3	7	12

Table 4.2. Repetitions needed for probability of greater than 95%

Figure 4.9 compares the devices with the added repetitions by adding the cost to the energy consumption before the normalization. The results in that the device D4 is the best option when looking for a PDR of 95% and higher. This corresponds to the last device only needing to send one time as seen in table 4.2. Figure 4.9 also illustrate that if two devices end up using same number of repetitions the most energy efficient device is best as expected.

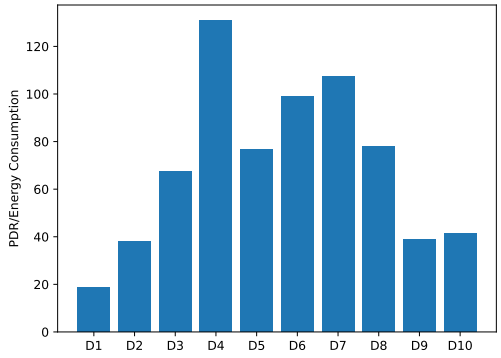


Figure 4.9. Packet delivery rate normalized with energy consumption, with attenuation and repetitions for each device

4.4 Reliability over time periods

Every result shown so far have been using the reliability calculated using the entire dataset collected, but as the testing environment at the tunnel and bridge changes over time, this change could be the traffic or the weather. It might also be interesting to see how the PDR behaves over different time of the day.

The packet delivery rate can be calculated based of the time of the day, by grouping the data by the hour. This can be seen in Figure 4.10 for the devices D6-D10, the other devices not shown as they are similar to D6. Here it is shown that there are a difference in the reliability and when the transmission occur doing the day. The times 12, 16 and 20 are marked with red vertical lines. It can be seen that at 12 there is a small drop over all devices, and between 16 and 20 the a drop especially for D9 and D10, this corresponds to when there are a lot of traffic in under the sensor .

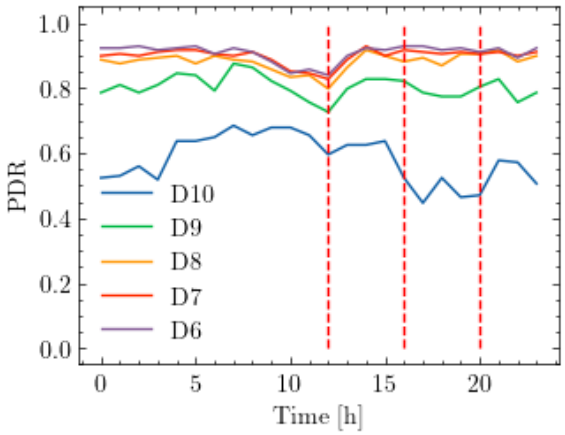


Figure 4.10. Reliability grouped over the hours of a day for different devices

Another interesting metric in the time domain is to see if there is a day of the week for the collected data that have a higher reliability. The PDR over each weekday can be seen in Figure 4.11 and with the data collected it seems that transmitting in the weekend is preferred and that

Wednesday might be a bad day for transmitting, but with the limited testing completed in the measurement campaign, it is hard to conclude if this is specific to an event or is common to the day of the week. More investigation and a longer measurement campaign should be conducted to really see the if the week days have any impact on the transmission reliability.

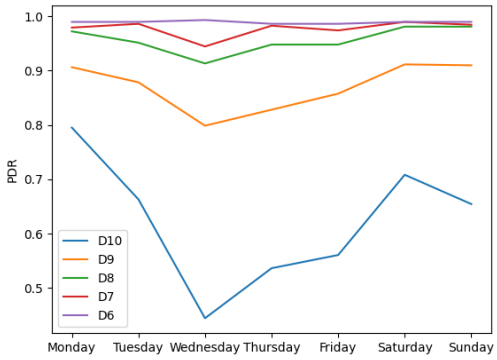


Figure 4.11. Reliability grouped over the weekdays for different devices

4.5 Sub conclusion

This chapter presented the data from the measurement campaign conducted at the bridge focusing on the LoRa transmissions, their reliability and the cost effectiveness across different configurations. By utilizing data from prior project with the transmissions the packet delivery rate normalized with the energy consumption showed the energy cost of transmitting compared to the delivery rate.

The measurement campaign were split into two phases with and without addition of attenuation to mimic the device being further away from the gateway. Without the attenuation the best device to transmit with were D10, with the attenuation the best device were D7. These results were based on the observed reliability and the energy cost, adding a fixed reliability of 95% and reaching this with the use of repetitions the preferred device to transmit changes to D4 as seen in Figure 4.9. The settings from this device is used in the next chapter network calculus to find the consumption curves for transmitting.

Additional analytics were made over different time periods to investigate the impact of the environment over time, here it was shown that time does impact the reliability and that devices as D10 with the weakest link budget struggles more in certain periods. Also an interesting discovery of events or weekdays that might be better or worse, but without more data the confidence on these are low, so a longer campaign is needed.

Network calculus 5

This chapter introduces network calculus for the uninitiated, and the definitions introduced serve as a basis for discusions on how the theory and the method is applied and how it is adopted to analyze the data collected from the sensor node. The first section covers the network calculus theory and the results it can provide when analyzing a network system, including some additional information for the min-plus algebra concept. Following this is an explanation for how network calculus can be applied to the energy harvesting data, by varying the choice of arrival, output and service functions.

5.1 Introduction to Network calculus

Network calculus is a branch of system theory that focuses on communication networks. It provides a mathematical framework designed to analyze the performance of a queuing system, within these networks. Network calculus provides a method to predict and guarantee metrics like delay and backlog by focusing on the network flows in the system under consideration, it has input flows and output flows as indicated on a simple network system as seen on Figure 5.1.

In the figure two flows are shown $R(t)$ and $R^*(t)$ where $R(t)$ is the input flow and $R^*(t)$ is the output flow from the system. The system is shown with β the service curve, that describes how the system processes the input flow, described later with it's see definition 3.

The input and output flows can be seen as bits over time, and are shown, as an example, with a staircase function on Figure 5.2. $R(t)$ is also later referred to as the arrival function, and $R^*(t)$ is the output function also referred to as the departure function.

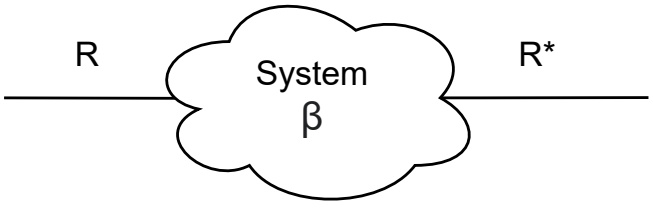


Figure 5.1. Illustration of the mapping between arrival R, departure R^* and service β functions to the basic system model.

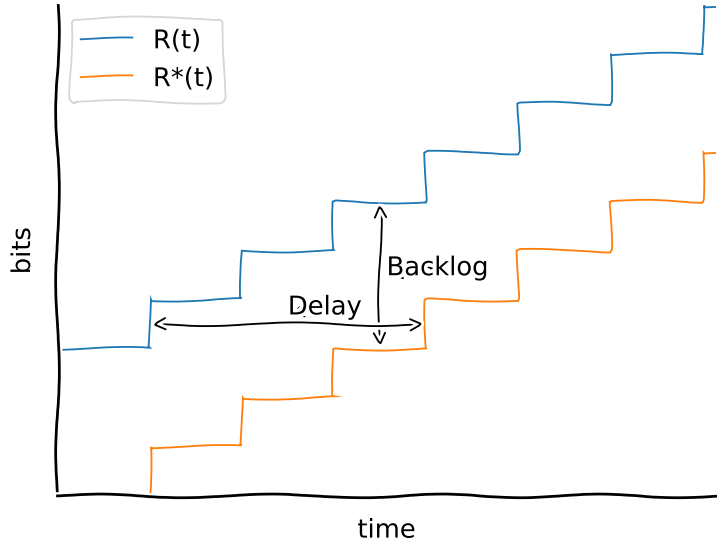


Figure 5.2. Illustration of the delay and backlog in network calculus.

As indicated in Figure 5.2 the backlog and the virtual delay [1] can be calculated from these two flows [38]. The backlog for any time t can be calculated with:

$$Backlog(t) = R(t) - R^*(t)$$

Where:

R	Arrival function
R^*	Departure function

The backlog at time t is the amount of bits that are held in the system for that time, when the system under consideration is a single queue it indicates the queue length. But if the system is a more complex it is the sum of bits in all queues [38].

The virtual delay is the time it takes between bits arriving and the same bits, then leaving the system. And if the system preserves the order in the flows the delay can be found as indicated in Figure 5.2 and defined by the following equation:

$$d(t) = \inf\{\tau \geq 0 : R(t) \leq R^*(t+\tau)\}$$

Where:

R	Arrival function
R^*	Departure function
d	Delay
τ	Time shift

[1] Virtual delay is used to emphasize the theoretical and analytical nature of the delay results from network calculus

5.1.1 Arrival curve

To make any guarantees for these parameters, there is a need to limit the flows. Limiting the input flow is done with the Arrival curve that has the following definition:

Definition 1 *(Arrival Curve). Given a wide-sense increasing function α defined for $t \geq 0$ we say that a flow R is constrained by α if and only if for all $s \leq t$:*

$$R(t) - R(s) \leq \alpha(t-s) \tag{5.1}$$

We say that R has α as an arrival curve, or also that R is α-smooth [38].

As seen in the definition the input for the arrival curve $\alpha(t-s)$ is a time interval between t and s and is the upper bound for the bits received in that interval as seen by $R(t)-R(s)$. This results in overlapping intervals that defines the arrival curve.

One can use an affine arrival curve given by $\alpha(t) = r\,t + b$, this is also known as a peak rate limit. The added b is called the burst tolerance, while the r is the rate. b is then the amount of data that can be sent at once. If the equation is simplified with $r = 0$, then $\alpha(t) = b$ and the system could only ever send a total of b bits. An example system using the rate could be when the system is limited by a physical link with a bit rate limit of r b/s only allowing a certain flow.

Another possible arrival curve can be represented using a staircase, a good use case is when the system receives packet of fixed size k and are spaced at least with T time units. Then the staircase arrival function is defined by the step size k as the height and the spaced time T as the width of the edge.

The arrival curve limits the flow, but it is possible to find a "good" flow that is the greatest lowest bound for the flow. A flow is said to be "good" by definition 2.

Definition 2 *Consider a function α in F. We say that α is a "good" function if any one of the following equivalent properties is satisfied*

1. *α is sub-additive and $\alpha(0) = 0$*
2. *$\alpha = \alpha \otimes \alpha$*
3. *$\alpha \oslash \alpha = \alpha$*
4. *$\alpha = \bar{\alpha}$ (sub-additive closure of α).*

5.1.2 Service curve

The system is said to offer the arrival function a minimal service curve β that guarantees the service rate of the processing in the system. The service curve indicates how quickly the network system can process the incoming data.

To better understand an example can be a system consisting of a leaky bucket, it has an arrival curve that represent the amount of traffic arriving into the bucket. The service curve then represents the amount of traffic the leaky bucket leaks, e.g. a constant rate. This system is also known as a constant rate server with rate c and it has the service curve: $\beta(t) = ct$

The definition for the service curve is stated in 3.

Definition 3 *(Service Curve). Consider a system S and a flow through S with input and output function R and R^* . We say that S offers to the flow a service curve β if and only if β is wide sense increasing, $\beta(0) = 0$ and $R^* \geq R \otimes \beta$ [38]*

The definition states that the output function R^* is greater then the convolution between the input function R and the service curve β . This states that R^* must be above the curve $R \otimes \beta$. This can also be stated using the infimum as seen in the following equation where β is a wide sense increasing function and for all $t \geq 0$:

$$R^*(t) \geq \inf(R(s) + \beta(t-s))$$

This gives the greatest lower bound for the output function of the system with the given input constraint and service.

Using these curves in network calculus gives the results also known as bounds for the network system. There are 3 bounds from network calculus, and the first bound is the backlog that is bounded by the vertical deviation between the arrival and service curves. The second bound is on the delay that is bounded by the horizontal deviation between the arrival and service curve. This can be seen in Figure 5.3, notice that this is similar to the example shown for the input and output functions, but here the time is in the time interval domain, and here the backlog and delay are upper bounds.

The third bound is on the output function, where the output is constrained by an arrival curve. This is when the system is not smooth compared to the input, and thus there might become an increase in delays and or the burst on the output from the system. The output flow is constrained by the curve with $\alpha^* = \alpha \oslash \beta$.

For the backlog and delay the bounds are tight if the arrival curve α is a "good" function and the service curve β is a wide sense increasing function and $\beta(0) = 0$. But they are only as tight as the arrival and service curves are [39, 40].

5.1.3 Numerical Implementation

In order to perform analysis on collected data, a number of numerical implementations of methods in network calculus, can be used. most network calculus tools perform efficient computation using line segments, or are limited to parametric models. In order to perform min-plus convolution of capture data traces, a python adaption was made of the convolution

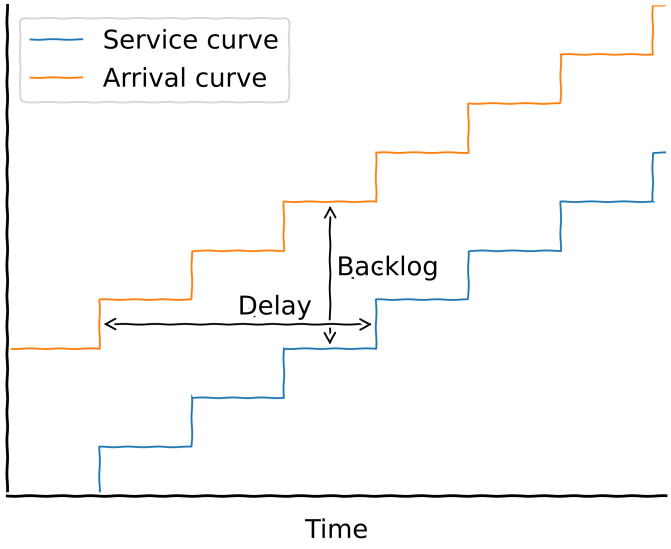


Figure 5.3. Illustrating an arrival curve over a service curve, including indicators for delay and backlog

function derived in [41], for CUDA. As the data analysis in this project is not concerned with high rate or high resolution internet traffic, a cpu implementation should suffice, and it comes with the benefit of greater portability, as no special hardware is required. The adapted code can be found in Listing below

```
def convolution(A, B):
    lenG = len(G)
    lenF = len(F)
    lenFG = min(lenG, lenF)
    FG = np.empty(lenFG)
    for t in range(lenFG):
        res = np.inf
        r = min(t + 1, lenG)
        l = max(0, t - lenF + 1)
        s = l
        while s < r:
            res = min(res, F[s] + G[t - s])
            s += 1
        FG[t] = res
    return FG
```

The functions for delay and backlog can be defined as element wise operations, to allow for inspection of the variations in delay and backlog over time, and using the result series, the bounds can be found. In these computations it is not possible to achieve a negative backlog and delay.

```
def backlog(a, b):
    """ backlog computes the backlog at every interval

    Example computation for Max backlog
    np.max(backlog(a,b))
    """
    return np.maximum(a - b, 0)

def delay(a, b):
    """ delay computes the pairwise delay at every interval

    Example computation for Max delay
    np.max(delay(a,b))
    """
    assert len(a) == len(b)
    c = np.empty_like(a)
    for ia, va in enumerate(a):
        for ib, vb in enumerate(b):
            if va == vb:
                c[ia] = (ib - ia)
                break
    return np.maximum(0, c)
```

5.2 Network calculus applied for energy system

In this project the arrival curve for energy consumption is estimated and the service curve based on the transmission cost is found based on the data presented in chapter 3 and 4. The energy harvested from the solar cell is defined as the input for the system. In the context of network calculus the system and queue is based on the energy storage, the capacitor and in its simplest form it is similar to a single buffer queue. The output is defined to be the energy consumption used by the system, where the service curve is the amount of energy that the system can use for transmissions.

The next sections explore how the data from Chapter 3 and 4 is fitted to the arrival and service curves respectively in network calculus.

The service curve is based on the energy consumption used to send LoRa transmissions. By changing the parameters for LoRa transmissions the energy consumption vary, there are many possible ways to configure LoRa. In this section only device D4 and the configurations settings and the power consumption associated with it is considered.

The results from the network calculus by doing it with these curves presents the amount of

transmissions that are possible with the constraint that no energy deficit is allowed. In other words the amount of energy used by the service curve must never exceed the energy harvested. This can be expressed as: $B(t) >= 0$ and the backlog, or energy surplus/deficit is calculated using $B(t) = \alpha(t) - \beta(t) >= 0$ for all t.

5.3 Energy harvesting models

The energy harvesting represents the arrival curve, and in this model of network calculus, finding the greatest lower bound on the energy harvesting is desired. The least upper bound on energy input, is primarily useful in determining the maximum energy storage size, if no loss is desired. The collected data could be used in a greedy approximation, where a shifted version of the data serves as an upper or lower bound, yet it fails to provide the benefit of the modeling, as it does not make analysis easier. The trade off in this modeling is the complexity of the function chosen. A desired set of functions exist in DNC, which allows for fast computation using analytical results, these functions could be affine, or staircase functions.

A comparison is thus made to determine the greatest lower bound using the affine and staircase functions. The affine curve is defined using the minimum slope of the accumulated energy, with a "burst" capacity of an arbitrary fully charged super capacitor. The curve is shown in relation to the input function in Fig. 5.4.

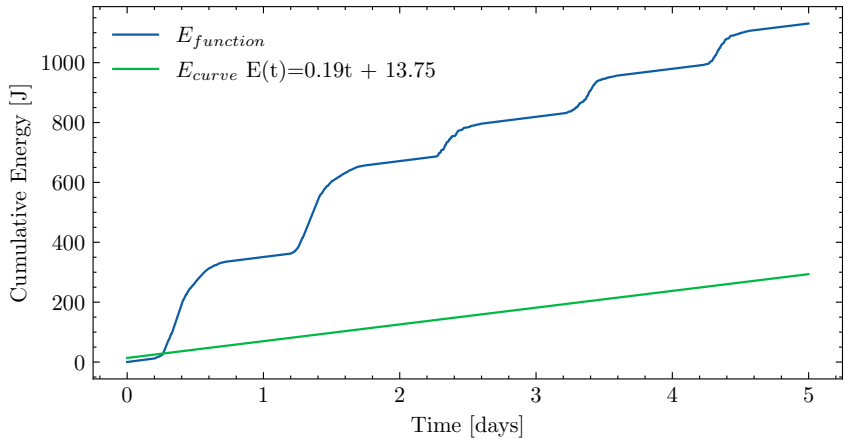


Figure 5.4. Energy arrival function (blue) and affine Energy curve (green). The affine curve fails to model "latency" of the function, and as such suffers with a lower long term rate.

The second function considered is the staircase function, which was determined by the curve that minimizes backlog and delay with respect to the arrival function, without at any point exceeding the arrival function. The function is shown in fig 5.5, where an additional burst capacity equal to the step size is added, for later analysis.

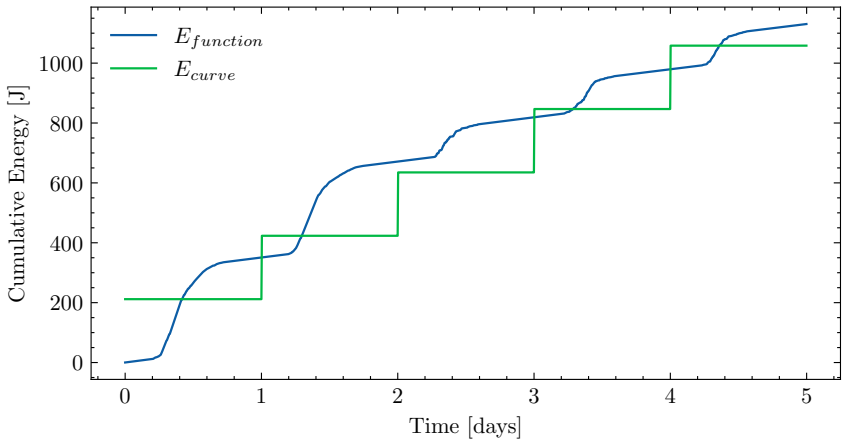


Figure 5.5. Energy arrival function (blue) and stair case curve (green). The staircase curve more accurately model the daily arrivals of energy, and as such show an improved greatest lower bound on the rate.

5.4 Energy consumption models

This sections presents how the data from chapter 4 and the energy consumption's models from Appendix B on page 83 is used to model a service curve.

To get the power consumption from the appendix there are two parameters, the configuration and the time on air for the transmission, with these the calculation of the energy consumption for a single transmission is completed. Adding these transmission costs together as a staircase function, where the step is the time between transmissions, and the energy consumption is the step height results in the service curve.

To find the service curve presented in Figure 5.6 the configuration is chosen, in this case SF 10 and TX power 20. The Time of air is calculated based on this and a payload size of 200 bytes. These are used to calculate the step height, and the step width is the variable that is used to find the best fit. The width is given in the seconds between transmissions. In Figure 5.6 the Y-axis show the energy consumption and the X-axis is the time given in steps, each representing a single transmission.

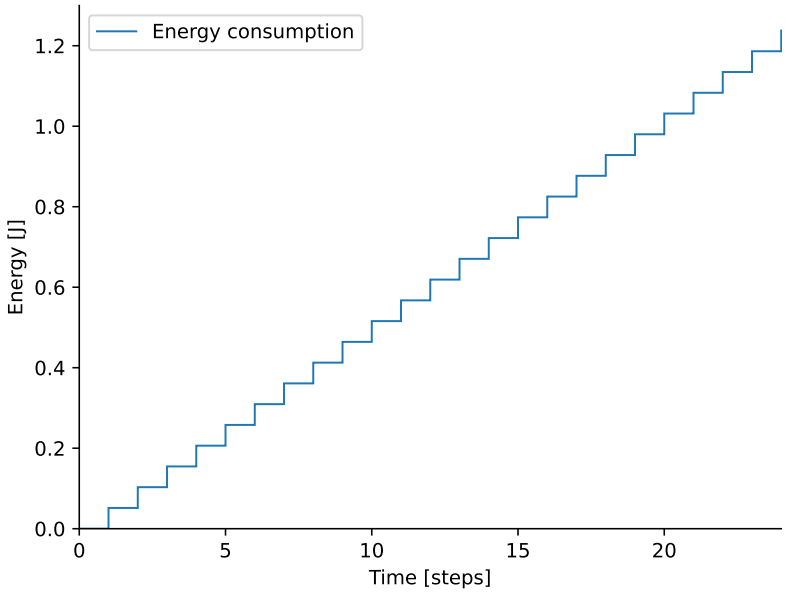


Figure 5.6. Energy consumption model for configuration SF 10 and TX 20 with the time axis being based on a variable step

5.5 Network calculus results

This section combines the models from 5.3 and 5.4 to find the best possible service curve that still holds the constraint that the energy storage can not be in deficit. This is the expression $B(t) = \alpha(t) - \beta(t) >= 0$ in the figures this means that the service curve always needs to be lower than the arrival curve.

The first model uses the worst case of energy increase observed over a 5 min period as the rate for a affine arrival curve $E_{Arrival}$. Using the energy consumption model with the configuration for D4 as selected in chapter 4 and adjusting the rate to be dependent on the energy consumption and finding the interval between transmissions.

To find the service curve for the affine arrival curve the calculations in equation (5.2) were used to find the packets possible for the interval. Using this equation the resulting interval of time between transmitting a LoRa packet to be 26.4s.

$$ppi = rate/ppe \tag{5.2}$$

Where:

ppi	Packets per interval	
$rate$	Arrival rate per interval	0.19
ppe	Power per packet	0.00736 J

Also the second function is considered where the arrival curve is given as a staircase function.

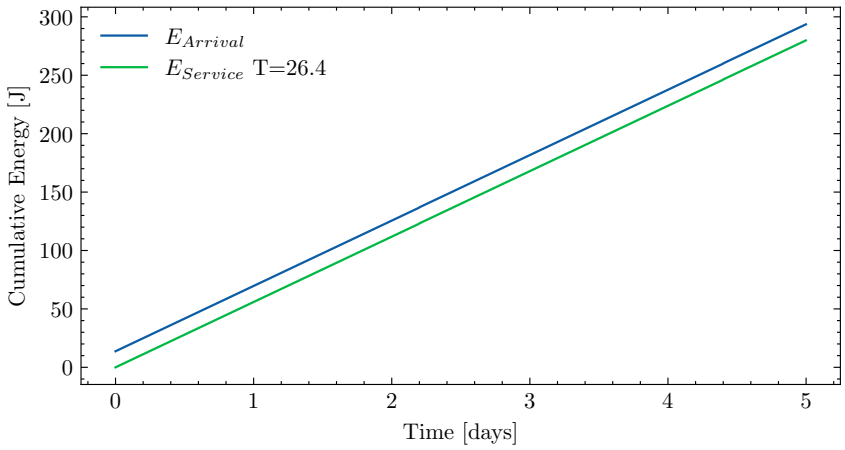


Figure 5.7. Affine arrival curve over the service curve that best utilize the energy available

The same idea is applied here and finding the best service curve that still holds the constraint. This finds the time interval between possible LoRa transmissions.

There is an additional step when finding the time interval between packets. As the arrival curve is a stair case the rate of this stair case is found before doing the same calculations for this arrival curve. The rate is found by taking the step height over the step interval.

By using this second staircase function that is an improved greatest lower bound the interval rate for transmitting the rate becomes 3.01s.

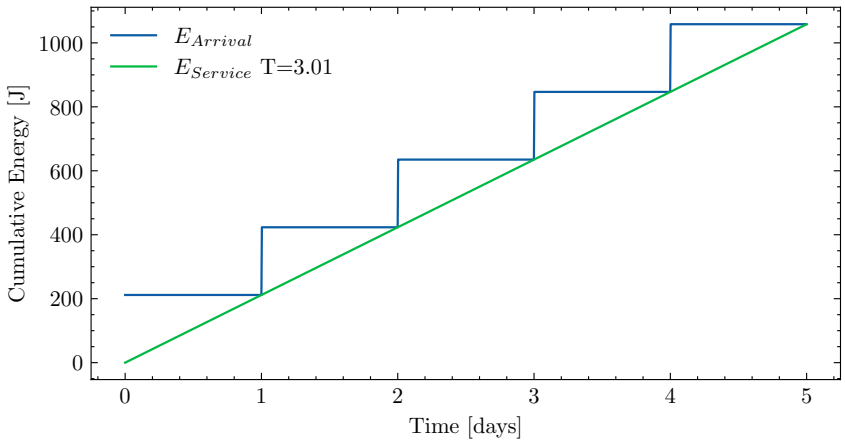


Figure 5.8. Staircase arrival curve over a service curve that best utilize the energy available

6 Discussion

The discussion covers the sources of errors in the measured data, and a discussion is provided on the assumption made during the project, and degree of satisfaction for the problem statement

One the most significant challenges faced during the project was missing data points for each phase of measuring. The missing data points prevented quick and easy access to the energy harvesting values and additional computations were needed, this utilized the light intensity, temperature and characteristics of the solar cell to provide an approximation of the energy harvested. In the best case the data points for both voltage and current is available for every data collection to calculate the actual energy. Because of the approximation of the energy the uncertainty raises as assumptions of the solar cell characteristics and minor changes in temperature could influence the actual value. The method to approximate however has not seen much use in the range of extremely low insolation, that were observed by the sensor in this project.

The energy consumption in the project is calculated using the reliability observed and then using the ToA (Time on Air) for a desired packet length. But as observed with higher spreading factors and longer transmission time is more prune to collisions, and the reliability measured utilizes a very small packet to save energy doing the measuring campaign. Using the correct packet length for scenarios in structural health monitoring might result in different reliability observations.

The deterministic network calculus models were applied to the energy harvesting, and while it was possible to fit the NC models to the energy harvesting data, it was limited by the duration of the measurement campaign, as only days to weeks of data were available. This does not account for variations encountered by seasonal changes, and this should be considered for a valid result in network calculus.

However the energy modeling could be extended based on similar larger datasets, to achieve the correct shape, and a parametric model could be constructed to better match the general intensity of light received. Finally the network calculus modeling could be used to show the relationship between daily energy harvesting and the achievable throughput.

Conclusion 7

In this project an energy harvesting sensor node was developed, to support a measurement campaign of 1) tracking solar energy harvesting under an overpass and 2) measure the reliability of LoRaWAN in an environment with large varations in vehicle traffic. The sensor node was designed iteratively to integrate experiences from field deployments over 2 iterations. A method rapid prototyping with later refinement was applied, to quickly develop a complex sensor node. Additionally infrastructure to control and receive LoRa communication were deployed.

The measurement campaigns performed by the sensor node served as a foundation for analyzing the efficiency of repetition reliability and noise variations for LoRaWAN communication, and for modeling of energy harvesting, based on multiple validating sensor sources, i.e. lux meter, ammeter and capacitor charge with discharge sources. The energy modeling and the results for LoRaWAN reliability enabled the use of Network Calculus as a tool to, in a basic example, determine the maximum throughput supported by an energy harvesting source, where the atypical application of arrival bounds was reconciled using the greatest lower bound for energy arrival, using known NC functions to better capture the periodic daily energy arrivals.

Likewise for wireless communication, a method allowing for deterministic analysis of LoRa communication was found through packet repetition, to ensure a common reliability, when comparing the energy efficiency of LoRa parameter settings. The findings suggest that the reliability of LoRa communication does not rely solely on the energy per packet, but also on the transmission time, which was prolonged for high values of spreading factor. The optimal transmission in the tested environment was found to be a trade off between increasing energy using transmission power and spreading factor.

The maximum throughput discovered by using network calculus in the rapport were found based on the arrival curve to be a transmission every $26.4s$ or with a greater lower bound curve to be $3.01s$ interval. These interval are far greater then the requirements for most structural health monitoring scenarios.

To answer the final problem statement for the project

How can real-world data on energy harvesting and wireless transmissions be modeled using network calculus to characterize the time-dependent relationship between the harvested energy and consumption required for wireless transmissions in structural health monitoring scenarios?

Real world sensor data on energy harvesting and wireless transmissions were collected from an overpass that was a part of the Limfjord tunnel entrance, providing an atypical propagation environment. The energy harvesting and LoRa wireless transmission data was modeled using Deterministic network calculus, and the time dependent relationship was analyzed in order to provide guarantees for the greatest lower bound on the data throughput of the sensor node. The tools presented can be used to integrate end user requirements when developing structural health monitoring solutions.

Future work 8

This chapter introduces some possible future works that can be considered to investigated with results presented in this project. The chapter will future work in all aspect of the project, from the sensor node implementation and deployment to the usage of the network calculus models presented.

If any further work should extend this project directly the sensor node should be refactored with the knowledge gained, add the modifications completed as patches in this project as indicated in Section 2.3 on page 11. This could preferably be done by ordering the PCB board, instead of manufacturing the board at the university. With a refactored sensor node additional measurements campaigns should be conducted to investigate transmissions reliability in the environment even further. This could allow to investigate the time periods mentioned in Section 4.4 on page 59.

While testing the transmission the ADR algorithms in LoRaWAN could also be investigated. It could be interesting to monitor different known ADR algorithms, find the energy consumption and compare the results to the results found in the rapport. Additionally customs ADR algorithms for structural health monitoring scenarios could be developed and tested to see if an ADR algorithm could be optimized for the requirements for these scenarios.

The project also only consider the single location for the gateway, and a single sensor node as seen in Fig. 2.19 on page 39, and it could be interesting to see the impact of adding multiple sensors for the same gateway, does the reliability change as there becomes more interference from the additional sensors.

Future work could also investigate the optimal places to have gateways if sensors should be placed throughout the whole tunnel or if a completely different network typology is better. If the topology introduces relaying of messages the network calculus system should be updated accordingly. Utilization of using convolution of the flows in network calculus would make it possible to investigate the worst case point of the network.

Future work could also consider flipping the network calculus system model to have energy consumption as the service curve, and then utilizing expected or wanted packets as the arrival and the departure function as the transmission sent. Both the arrival and departure should be using a specific LoRa configuration and related to the energy consumption to match the service curve. This method would be able to utilize the bounds of network calculus to see if the transmission immediately, enough power in the battery or there is times without service. This

removes the constraint used in the network calculus method investigated in this project.

The energy harvested from solar cell in this project allows the sensor node to transmit quite frequently, and future work could investigate alternative energy harvesting methods, or a type of solar cell that is meant to be used for this purpose.

Bibliography

[1] Randall Munroe. XKCD Font. https://github.com/ipython/xkcd-font, 2013. Visited 27-05-2024.

[2] EnABLES. Research Infrastructure to Power the Internet of Things. https://www.enables-project.eu/wp-content/uploads/2021/02/EnABLES_ResearchInfrastructure_PositionPaper.pdf, 2021. Visited 27-05-2024.

[3] Cordis. Up to 78 million batteries will be discarded daily by 2025, researchers warn. https://cordis.europa.eu/article/id/430457-up-to-78-million-batteries-will-be-discarded-daily-by-2025-researchers-warn?WT.mc_id=exp, 2021. Visited 27-05-2024.

[4] S RAUPACH, M. and R Sensortec GmbH. Corrosion monitoring using a new sensor system for installation into existing structures. Durability of Building Materials and Components, 8, 06 1999.

[5] K. Kumar, S. Muralidharan, T. Manjula, Karthikeyan Sakthivel, and N. Palaniswamy. Sensor systems for corrosion monitoring in concrete structures. Sensors and Transducers, 67:553–560, 05 2006.

[6] Saraswathy Velu. Corrosion monitoring of reinforced concrete structures - a review. International Journal of Electrochemical Science, 01 2007.

[7] Paul Darlington Ceng Fiet Firse. Energy harvesting and 'fit and forget' wireless sensors. Rail Engineer News, https://www.railengineer.co.uk/energy-harvesting-fit-and-forget-wireless-sensors/, 2023. Visited 28-04-2024.

[8] COWI and Vejdirektoratet. Permeable Belægninger. https://www.klimatilpasning.dk/media/1147166/vd-2015-0079.pdf, 2015. Visited 27-05-2024.

[9] Vibeke Lyngklib Svansø. Droner og Kunstig Intelligens fordobler Storebæltsbroens levetid fra 100 til 200 år. Berlingske, https://www.berlingske.dk/business/droner-og-kunstig-intelligens-fordobler-storebaeltsbroens-levetid-fra-100, 2020. Visited 27-05-2024.

[10] Brian Lohse. Intelligent overvågning af tilstanden af broer og tunneller. Force Technology, https://forcetechnology.com/da/om-force-technology/nyheder/intelligent-overvaagning-broer-tunneller-corrosense, 2023. Visited 27-05-2024.

[11] Tobias Wasner, Max Helm, and Dominik Scholz. What is deterministic network calculus. IITM Network Architectures and Service, 10 2019.

[12] Kai Wang, Florin Ciucu, Chuang Lin, and Steven H. Low. A stochastic power network calculus for integrating renewable energy sources into the power grid. IEEE Journal on Selected Areas in Communications, 30(6):1037–1048, July 2012. ISSN 1558-0008. doi: 10.1109/JSAC.2012.120703.

[13] Stefano Basagni, M. Yousof Naderi, Chiara Petrioli, and Dora Spenza. Wireless Sensor Networks with Energy Harvesting, chapter 20, pages 701–736. John Wiley & Sons, Ltd, 2013. ISBN 9781118511305. doi: https://doi.org/10.1002/9781118511305.ch20. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118511305.ch20.

[14] LoRa E5 Mini. Seeed Studio, https://www.seeedstudio.com/Wio-E5-LE-mini-Dev-Board-STM32WLE5JC-p-5764.html, 2023. Visited 28-04-2024.

[15] LoRa E5 Mini Board file. Seeed Studio, https://files.seeedstudio.com/wiki/Loramini/LoRa-E5%20mini%20v1.0.brd, 2023. Visited 28-04-2024.

[16] Jonathan Valdez and Jared Becker. Understanding the I2C Bus. Texas Instruments, https://www.ti.com/lit/an/slva704/slva704.pdf, 2015. Visited 28-05-2024.

[17] Patricio Whittingslow. fatfs. https://github.com/soypat/fatfs, 2024. Visited 28-05-2024.

[18] Mikroe. Click boards. https://www.mikroe.com/click-boards, 2024. Visited 28-05-2024.

[19] Mikroe. Solar Energy 2 Click. https://www.mikroe.com/solar-energy-2-click, 2024. Visited 28-05-2024.

[20] EM Microelectorinic. EM8500 datasheet. https://www.emmicroelectronic.com/sites/default/files/products/datasheets/8500-ds.pdf, 2017. Visited 28-05-2024.

[21] TAOS. EM8500 datasheet. https://download.mikroe.com/documents/datasheets/TSL2583%20Datasheet.pdf, 2012. Visited 28-05-2024.

[22] Mikroe. Illuminance click. https://www.mikroe.com/illuminance-click, 2024. Visited 28-05-2024.

[23] Lionel Clasing, Simon Schaaf, Ulf Blieske, Nicholas Riedel-Lyngskær, Adrián A. Santamaria Lancia, and Nils Reiners. Calculation of the short-circuit current of colored bipv modules under field conditions by application of spectrally and angle resolved measurement data. In Proceedings of the EU PVSEC 2021, pages 803–807, 2021. 38th European Photovoltaic Solar Energy Conference and Exhibition, EU PVSEC 2021 ; Conference date: 06-09-2021 Through 11-09-2021.

[24] Edson L. Meyer. Extraction of saturation current and ideality factor from measuring voc and isc of photovoltaic modules. International Journal of Photoenergy, vol. 2017, page 9, 2017.

[25] Sutirtha Chakraborty. The init function in Golang. GODocs, https://golangdocs.com/init-function-in-golang, 2020. Visited 28-04-2024.

[26] Understanding init in Go. Digital Ocean, https://www.digitalocean.com/community/tutorials/understanding-init-in-go, 2019. Visited 28-04-2024.

[27] LoRaWAN Alliance Technical Committee. LoRaWAN® L2 1.0.4 Specification. LoRaWAN Alliance, https://resources.lora-alliance.org/technical-specifications/ts001-1-0-4-lorawan-l2-1-0-4-specification, 2020. Visited 28-05-2024.

[28] Nordic Semiconductor. Power Profiler Kit II - User Guide. https://infocenter.nordicsemi.com/pdf/PPK2_User_Guide_v1.0.1.pdf, 2023. Visited 24-05-2023.

[29] Nordic Semiconductor. Power Profiler Kit 2. https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2, 2023. Visited 22-05-2023.

[30] ETSI EN 300 220-2 V3.2.1 (2018-06). European Telecommunications Standards Institute, https://www.etsi.org/deliver/etsi_en/300200_300299/30022002/03.02.01_60/en_30022002v030201p.pdf, 2018. Visited 28-05-2024.

[31] LoRaWAN Alliance Technical Committee. LoRaWAN® 1.0.4 Regional parameters. LoRaWAN Alliance, https://resources.lora-alliance.org/technical-specifications/rp002-1-0-4-regional-parameters, 2020. Visited 28-05-2024.

[32] The Things Network. Adaptive Data Rate. https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate/, 2021. Visited 24-05-2023.

[33] Aalborg universitet. CLAAUDIA - RESEARCH DATA SERVICES. https://www.en.its.aau.dk/claaudia, 2024. Visited 28-05-2024.

[34] Behnam Ousat and Majid Ghaderi. Lora network planning: Gateway placement and device configuration. 2019 IEEE International Congress on Internet of Things (ICIOT), pages 25–32, 2019.

[35] Google. Google My Maps. Google, https://mymaps.google.com/, 2024. Visited 28-04-2024.

[36] brocaar. LoRaWAN go. https://github.com/brocaar/lorawan, 2012. Visited 28-05-2024.

[37] SEMTECH. SX1276/77/78/79 Datasheet. https://www.semtech.com/products/wireless-rf/lora-connect/sx1276#documentation, 2020. Visited 29-05-2024.

[38] Jean-Yves Le Boudec and Patrick Thiran. NETWORK CALCULUS A Theory of Deterministic Queuing Systems for the Internet. https://leboudec.github.io/netcal/latex/netCalBook.pdf, 2022.

[39] Jean-Yves Le Boudec. An Introduction to Network Calculus. https://www.youtube.com/watch?v=ABQ327BTc_o, 2019.

[40] Jean-Yves Le Boudec. An Introduction to Network Calculus - Slides. https://leboudec.github.io/netcal/resources/tutorial-nc-dagstuhl-EPFL-2019-LEB.pdf, 2019.

[41] Natchanon Luangsomboon, Robert Hesse, and Jörg Liebeherr. Fast min-plus convolution and deconvolution on gpus. In Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2017, page 126–131, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450363464. doi: 10.1145/3150928.3150958. URL https://doi.org/10.1145/3150928.3150958.

[42] Thomas Gundgaard Mulvad, Rasmus Sibbern Frederiksen, and Khadar Abdi Guled Mohamed Ali. Feasibility analysis of concrete embedded lora: Optimizing energy consumption with a heuristic explorative approach to adaptive data rate. Aalborg University Student Projects, 6 2023.

Journal: Light solar cell test A

journal is to show the setup and results of characterisations done on the Azure space solar cell assembly. This test was performed in order to get good estimates of the reverse saturation current and the ideality factor, without access to expensive solar simulators or diode testing equipment, the test was conducted at the sound lab at Aalborg University.

The test is composed of multiple small tests with the same base setup, with minor changes for each test. The tests conducted here are:

- Lux range of lamp.
- VOC-ISC Method. Measuring the short circuit current, open circuit voltage over a range of illumination.

A.1 Hardware & Equipment

Table A.1 shows the hardware used for the tests, and table A.2 shows the equipment used to set up the experiments.

Hardware
TJ GaAs Solar Cell Assembly 3g30A
ILLUMINANCE CLICK
Wio-E5 mini

Table A.1. Hardware used to perform the experiments conducted on the solar cell.

Equipment
Nordic RF Power Profiler Kit II
Tripod
Amaran HR672W Light
STK33502 Ambient Light Sensor (Integrated Phone lux meter)
Multimeter

Table A.2. Equipment used for the test setup

A.2 Test setup

The test is conducted with the test setup as seen on Figure A.1 and A.2. The test setup consist of the Amaran HR672W Light mounted on to the tripod and is pointing down towards the ground. On the ground is than the hardware tested, either a Lux meter or the solar cell. All of this is placed inside a room where there is almost no natural light, so the amount of light can be controlled in the testing.

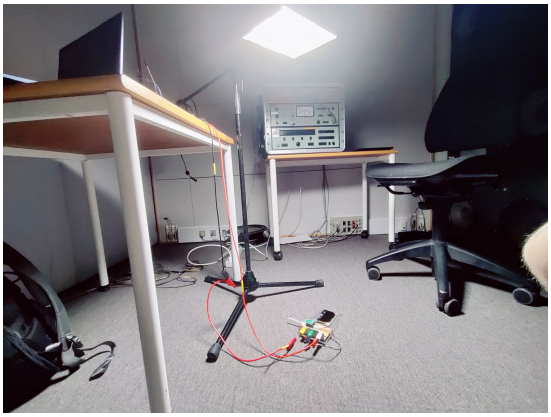


Figure A.1. Picture of the test setup

Figure A.2. Picture of the test setup

A.3 Procedure

Test procedures for each test is described below.

Getting the lux values from the light over its settings from 0% (off) and 10% to 100% by repeating the following steps.

1. Select the percentage on the Amaran HR672W Light
2. Read lux value from lux meter
3. Save value

To test the current from the solar cell the PPK2 is connected and the values is extracted from the software power profile app for the PPK2 by following these steps:

1. Select the percentage on the Amaran HR672W Light
2. Start measurements inside the nRF Connect Desktop power profile app
3. Turn of screen to limit other light sources
4. After waiting turning on the screen
5. Select the window the screen was off and save the average value for the current.

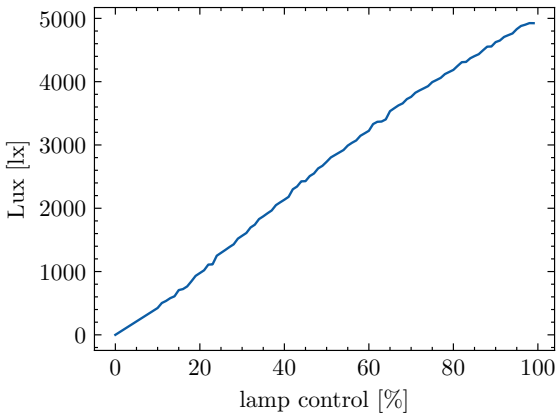


Figure A.3. Calibration of the lamp illuminance to gain ratio at a fixed distance

A.4 Results

In order to create the cell characteristic over a wider range, the cell was moved closer to the lamp. However the specific illumination value is not important for the VOC-ISC method, only the range of measurements.

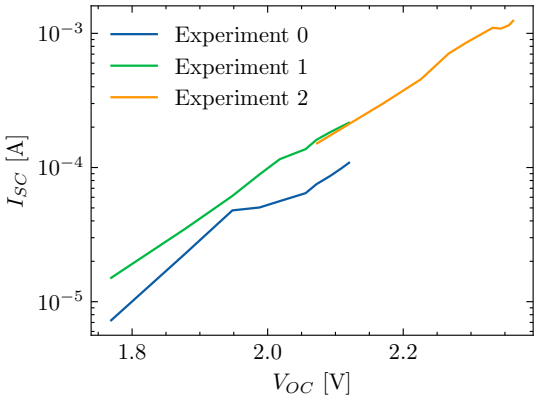


Figure A.4. Measurements of open circuit voltage and short circuit current, for 2 ranges of irradiance

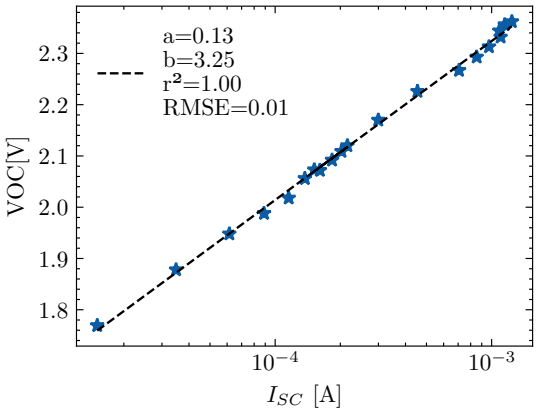


Figure A.5. Results of the log linear fit.

Journal: Energy consumption models

B

This journal's purpose is to show how the energy consumption models for each LoRa transmission configuration used in the testing are calculated. The calculations utilize the energy measurements completed for another project we conducted in the 8th semester.

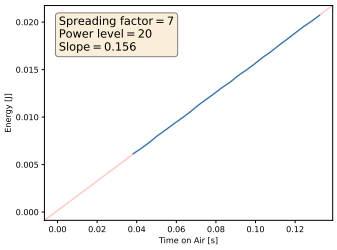
The 8th-semester project article is filed in the Aalborg University student project database by the title: Feasibility Analysis of Concrete Embedded LoRa: Optimizing Energy Consumption with a Heuristic Explorative Approach to Adaptive Data Rate[42]

B.1 Utilizing data for models

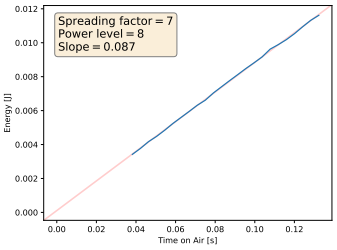
This section shows the data used to get energy models for each configuration. The data from the 8th semester holds information on energy consumption and the configuration that is defined using the bandwidth, spreading factor, coding rate, and power level. Each configuration is tested with different preamble lengths, which gives it a different time on air.

The goal is to get a mapping for each configuration between the time on air and the energy consumption. This goal is achieved by filtering the data on the configuration, calculating the time on air and plotting the energy consumption over the ToA. The plots show that there is a linear correlation between the ToA and the energy consumption, so finding the best linear fit results in an equation for each configuration that can calculate the energy consumption based on the time on air.

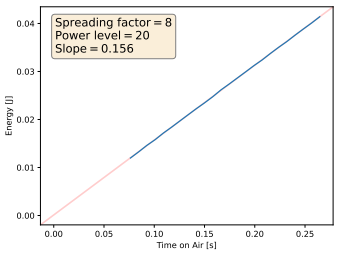
Figure B.1 shows the individual fit for each configuration on the data, and Table B.1 shows each fitted equation.



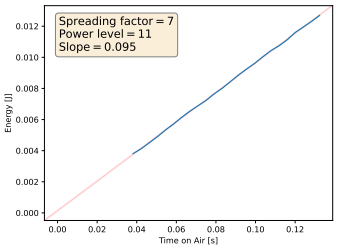
(a) SF7_PL20_lin_energy_toa



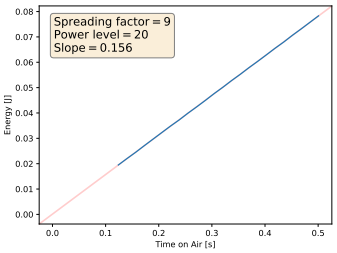
(b) SF7_PL8_lin_energy_toa



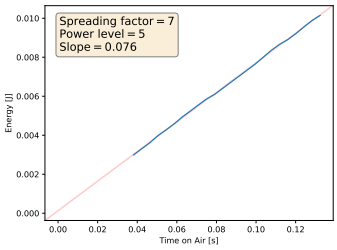
(c) SF8_PL20_lin_energy_toa



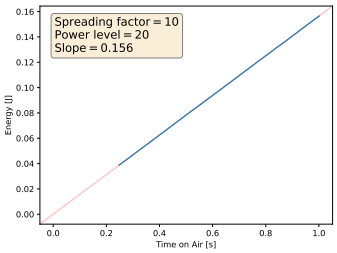
(d) SF7_PL11_lin_energy_toa



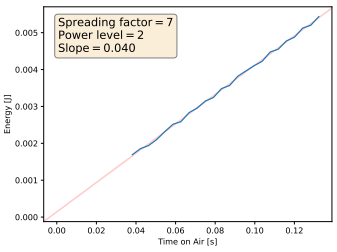
(e) SF9_PL20_lin_energy_toa



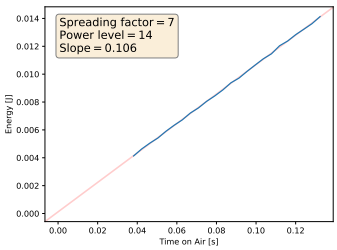
(f) SF7_PL5_lin_energy_toa



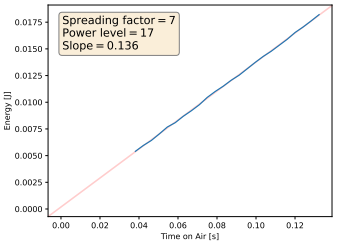
(g) SF10_PL20_lin_energy_toa



(h) SF7_PL2_lin_energy_toa



(i) SF7_PL14_lin_energy_toa



(j) SF7_PL17_lin_energy_toa

Figure B.1. Grid of 10 plots.

Spreading Factor	Power Level	Linear Regression
10	20	$0.1564 \times \text{ToA} + 0.0000$
9	20	$0.1561 \times \text{ToA} + 0.0001$
8	20	$0.1560 \times \text{ToA} + 0.0001$
7	20	$0.1555 \times \text{ToA} + 0.0002$
7	17	$0.1358 \times \text{ToA} + 0.0002$
7	14	$0.1057 \times \text{ToA} + 0.0001$
7	11	$0.0951 \times \text{ToA} + 0.0001$
7	8	$0.0873 \times \text{ToA} + 0.0001$
7	5	$0.0759 \times \text{ToA} + 0.0001$
7	2	$0.0398 \times \text{ToA} + 0.0001$

Table B.1. Spreading Factor, Power Level, and Linear Regression Equations