Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Øystein Molvik

Network of sensors for measurement of potential for delivery of electrical energy from solar panels

Master's thesis in Cybernetics and Robotics

Supervisor: Geir Mathisen

June 2020

NTNU
Norwegian University of Science and Technology

Øystein Molvik

Network of sensors for measurement of potential for delivery of electrical energy from solar panels

Master's thesis in Cybernetics and Robotics
Supervisor: Geir Mathisen
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

MASTER THESIS DESCRIPTION

Candidate:	**Øystein Molvik**
Course:	**TTK4900 Engineering Cybernetics**
Thesis title (Norwegian)	**Sensornettverk for registrering av mulig produksjon av elektrisk energy fra solcellepanel**
Thesis title (English):	**Network of sensors for measurement of potential for delivery of electrical energy from solar panels**

Thesis desctiption: The delivery of energy (DoE) from a solar panel can momentarely decrease to 1/7 as a cloud slide in and shadow an earlier direcly radiated panel. For solar panels being in the line where the cload moves, this decrease will happen to all the panels with a time-lag given of the distance between the panels and the speed of the cload. By monitoring the momentarly DoE from all the solar panels, it might be possible to predict the speed of the cload and to forsee when the cload will shadow a solar panel further in the line of the cload's moving direction (and analogous for disappearance of cloads). This again will help predict momentarely DoE in the near future.

In lack of being able to monitor DoE from a number of solar panels we want to develop a sensor network demonstrating the possibility to acquire information about potential for delivery of electrical energy from solar panels in the neighbourhood. Each sensor should be wireless and selfcontained with respect to internal energy consumption

The tasks will be:

1. Conduct a litterary study of relevant / related technologies for sensor networks as described. systems used for
2. Propose a system for real-time gathering of acquision of momentarely potensial for delivery of electrical energy from solar panels
3. As far as time permits, implement the suggested system from previous point.

Start date: 11. January, 2020
Due date: 30. June, 2020

Thesis performed at: Department of Engineering Cybernetics
Supervisor: Professor Geir Mathisen, Dept. of Eng. Cybernetics

Summary

In the industry, distributed measurements systems have existed and been used for many years with great results and success. In recent years, new technologies within the topic of Internet of Things (IoT) have emerged and completely changed the game. Long Range Wide Area Network (LoRaWAN) and Narrowband-IoT (NB-IoT) are two of those technologies who took the market by storm, and specialize in low-power IoT devices. With a climate focus increasing by the year, a slight shift in focus towards solar energy arrived. More and more devices and buildings being powered by solar energy new difficulties surfaced. Is there a way to at some point be able to predict the delivery of energy (DoE) from solar panels, based on others around it?

In this thesis an embedded power measurement sensor was designed, tested and finally implemented using LoRa and NB-IoT as communication. The sensor will measure voltage and current produced from a solar panel and transmit said data to a back-end application. A small network of 4 nodes were originally deployed in order to simulate a distributed measurement system (DMS) using both LoRaWAN and NB-IoT as communication protocol. The thesis investigates self-sufficiency, time synchronization, power consumption and necessary data to identify a time-lag between nodes. The deployed system did to some extent work as intended, but due to several discussed factors, not provide sufficient and good enough data.

A full system in order to actually predict cloud movement were never deployed. However, a small DMS with the intention of illustrating proof-of-concept were deployed but did not manage to procure sufficient data for a conclusive result.

Sammendrag

I dagens industri har distribuerte målesystemer blitt brukt i flere år og gitt gode resultater. I nyere år har ny teknologi dukket opp innen temaet IoT og fullstended endret spillereglene. LoRaWAN og NB-IoT er to av disse teknologiene som tok markedet med storm og spesialiserte seg innen ”low power IoT”. Med et stigende klimafokus blant befolkningene har et lite fokusskifte innen strømforsyning skjedd mot energien fra solen. Apparater og bygninger blir mer og mer forsynt via solcellepaneler, og dette og ledet til at nye vanskeligheter har blitt eksponert. Kan man predikere hvor mye et solcellepanel vil produsere ved en gitt tid basert på information fra andre paneler i nærheten?

En trådløs strømsensor som målet strøm og spenning og kommuniserer over LoRaWAN og NB-IoT ble designet og implementert. Sensoren målet strøm og spenning fra et solcellepanel og sender data til en server. Et mindre sensornettverkt på 4 nodes ble originalt satt i drift for å simulere et DMS vha. LoRaWAN og NB-IoT som kommunikasjonsprotokoll. Oppgaven undersøker egenskaper som selvforsyning av strøm, tidssynkronisering mellom noder, strømforbruk og tidsforsinkelser mellom produksjonen i nodene. Systemet virker som det skulle til en viss grad, men pga. ufortsette problemer og faktorer diskutert i oppgaven, ble det ikke produsert nok god reproduserbar data til å trekke konklusjoner.

Et komplett system med evnen til å predikere skybeveglser ble aldri implementert og testet. Istedenfor ble et liste sensornettverkt på 4 nodes utplassert med intensjon om ”proof-of-concept”, men evnet ikke å produsere nok reproduserbar data til å trekke en konklusjon.

Preface

This thesis concludes my five year master's degree in Cybernetics and Robotics at the Department of Engineering Cybernetics, Norwegian University of Science and Technology. The work done in the master thesis draws some inspiration from the work done in the TTK4450-project (specialization project).

The embedded system developed in this thesis is to a small extent inspired by the system I developed in the specialization project. Chapter three consists of relevant theory taken from the specialization project, and was intended to provide a basic introduction to the relevant aspects of LoRaWAN to proide a better understanding. Some software were reused from the specialization project and some taken from online sources. This is better explained in Chapter 5. The remained work presented in this thesis is done by me between the start and end date of the thesis.

Working with this thesis has provided a significant learning experience and been very challanging. A lot of uncommon challenges emerged with the COVID-19, which resulted in the school shutting down. This prevented me from using the necessary labaroty for finishing the original hardware. School shutting down and massive delays in order from China led to shift in implementation of the thesis.

I would finally like to thank my supervision Geir Mathisen for all his help and guidance during this project. A special thanks also goes to Åsmund Stavdahl down at the laboraty for his help and surviving all my nagging, and Are Viberg for his support and being a great conversation partner throughout this semester.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Internet of things (IoT), also defined as the interconnection between the internet and embedded devices, is the new up and coming concept within low power sensor technology. Its potential is far from reached as there are new applications and possibilities discovered every year. Billions of devices are already connected to the internet and the number keeps increasing. IoT is in simple terms helping us make the world simpler, smarter, and more responsive by merging the physical and digital universe.

Another important keyword is ”smart”. A simple light intensity sensor could easily measure a value, transmit it to a back-end server, and be called an IoT device. To be smart, the sensor must contain a chip capable of processing the data and perform predefined functions with it. For example, the sensor could also determine whether a cloud is above it or not, if it's night or day, etc. and also provide the back-end with this information. Expressions such as ”smart home”, ”smart city” and ”smart industry”, have all undeniably been adopted by most countries around the world, contributing to a massive advancement to the world as we knew it.

Further, as the climate problem steadily increases, a general focus for a lot of people has shifted towards new energy sources, such as solar power. This technology is also improving every year and can produce an increasing amount of power per cm^2. This has resulted in people purchasing panels to provide energy to their homes, industries, and even cities. These solutions usually come with big batteries for storage. For households or industries not capable of using a battery, or just desire efficient usage, it would be interesting to see if the production from panels could be predicted. Using that information, parts of the load connected to a panel could, for example, be alleviated, and used more efficiently.

Certain technologies for this exist already but involve weather stations and big home projects. These installments are usually too large and inconvenient and based on short-range communication. This thesis initially planned to look at the development of a flexible low power sensor, where multiple of those would be deployed in a grid and used to predict

the movement of clouds.

1.2 Motivation

To work on a subject so popular and new as the Internet of Things is incredible. New ways of combining different sensors for new solutions occur almost every day, so to be a part of the same flow and spear this development is highly motivating. Most importantly, to engage in the realization of creating something from scratch, combining all aspects of embedded systems with up and coming communication protocols creates a sense of ownership.

1.3 Limitations

At the beginning of the thesis, it was already decided to use nRF9160 DK as a modem with NB-IoT. This was due to close relations with Nordic Semiconductor, and to alleviate some of the work that would be required to integrate the nrf9160 system-in-package (SiP) with the rest of the solution. Deploying sensors around the city provided a greater challenge than expected. Not enough people have locations suited for solar panels and a sensor (which is not the roof).

The biggest limitation of this thesis was the event of the pandemic, Covid-19. This provided several problems which delayed the work considerably. Also, due to these delays, a forced alteration in the thesis scope occurred. Shifting the focus away from cloud movement prediction. Nevertheless, the literature study conducted based on cloud movement remains.

Chapter 2

Literature Review

The scope of this paper is to investigate the possibilities of using a grid of self-powered smart sensors to predict to some extent the movement of clouds. To realize this task, certain challenges need to be explored. The main challenges in this project involve communication, being self-sufficient on power, and synchronization between nodes. This chapter explores existing technologies and ideas within these fields which could be applicable for this project.

2.1 Low-power communication

In general, smart sensors function as a wireless sensor powered by a sort of battery to expand the range of where the sensor can be placed. To do so, the sensor must use a minimum amount of power to extend its maximum lifetime. To achieve this effect, there are multiple aspects to be considered. The most important one is choosing an efficient communication protocol with a focus on coverage, price, and power consumption. For this thesis, only Low Power Wide Area Network (LPWAN) technologies were considered, as they suited the scope of this thesis best, and short-range technologies were disregarded. Currently, there are primarily two different connectivity tracks for the many IoT applications that require a wide-area coverage, namely ”Unlicensed LPWA” and ”Cellular Technology”.

2.1.1 Unlicensed LPWA

Unlicensed LPWA is new proprietary radio technology. These WANs have been solely developed for machine-type communication applications and address the important ultra low-end sensor part of the market. The two main providers of this technology within this category are Long Range (LoRa) and Sigfox [10]. To evaluate these protocols for this thesis, K. Mekki, E. Bajic, F. Chaxel, and F. Meyer do a great job highlighting their differences in their study [11]. Those differences are shown in table 2.1.

Table 2.1: Differences between LoRa and Sigfox [11]

	Sigfox	LoRaWAN
Bandwidth	100 Hz	250kHz and 125kHz
Max. data rate	100 bps	50 kbps
Max. messages/day	140 (Uplink [UL]), 4 (Downlink [DL])	Unlimited
Power consumption	about 10 years	about 10 years
Max. payload	12 bytes (UL), 8 bytes (DL)	243 bytes
Range	10 km (urban), 40 km (rural)	5 km (urban), 20 km (rural)

2.1.2 Cellular Technology

These technologies operate on the licensed spectrum and have throughout history primarily targeted high-quality mobile data and voice services. Now, on the other hand, new functionality has evolved and two new main access technologies have emerged, narrowband IoT (NB-IoT) and Long-Term Evolution Category M1 (LTE-M) [12]. While these two are highly complementary to each other, they address different types of use cases based on their strengths. In a comparative study between these two technologies, done by B. E. Benhiba, A. A. Madi, and A. Addaim, they discovered certain differences between NB-IoT and LTE-M [13]. These are shown in table 2.2. The paper concludes that NB-IoT has a certain advantage over LTE-M. This is due to LTE-M being capable of hand-offs between cellular towers but at the price of more synchronization and slightly higher power consumption. This makes NB-IoT a very good choice for static devices, like sensors for simple measurements.

Table 2.2: Differences between LTE-M and NB-IoT [13]

	Cat-M	NB-IoT
Max. system bandwidth	1.4MHz	200kHz
Downlink peak rate	1 Mbit/s	66.7 kbps
Uplink peak rate	1 Mbit/s	32.4 kbps
Power consumption	about 10 years	about 10 years
Module cost	Moins de 5$	Moins de 10 $

2.1.3 Cellular vs. Unlicensed in applications

When considering which protocol most suited for an application, different factors are favored. The most common ones in IoT, are listed below:

1. Quality of Service (QoS)
2. Battery life & Latency
3. Network coverage & range
4. Deployment Model

5. Cost

6. Coverage

From R. S. Sinha, Y. Wei and S. Hwang's study [14], NB-IoT, and LoRa are compared in terms of the aforementioned IoT factors. Quality of service is a feature included in using a licensed spectrum technology. NB-IoT, for example, uses a time-slotted synchronous protocol which is optimal for QoS, while LoRa and Sigfox utilize an asynchronous protocol. This advantage is at the expense of cost, as the licensed band spectrum is typically above 500 million dollars per MHz [14]. This trade-off indicates that applications that require QoS, prefer NB-IoT. NB-IoT has, in general, better coverage, but the cost of building new stations is immense compared to setting up a new LoRa gateway. This is illustrated in table 2.3 and 2.4.

Table 2.3: Current consumption and latency between LoRa and NB-IoT [14]

	Peak current	Sleep current	Latency
LoRa	32 mA	$1\mu A$	Insensitive to latency
NB-IoT	120/130 mA	$5\mu A$	$< 10s$

Table 2.4: Difference of costs between LoRa and NB-IoT [14]

	Spectrum cost	Network & Deployment cost
LoRa	Free	\$ 100-\$/gateway
NB-IoT	$>$ \$500 million/MHz	\$15000/base station

2.2 Communication Coverage

For this project, coverage is the most essential aspect to consider when choosing a good protocol. With regards to LoRa, the public coverage around the country is not that widespread, and for a LoRa node to function, it needs a LoRaWAN gateway that receives its messages. So far there are only a small amount of them placed around the country. Considering the county of Trøndelag, only Trondheim has any coverage at all. The coverage provided is illustrated in figure 2.2, taken from TTN's website [2]. Zooming into Trøndelag, there is a community in Trondheim currently providing 9 gateways for general coverage [1], and can also be visualized in figure 2.1. A particularly good property of LoRa is the adaptability and mobility of the nodes by just setting up a new gateway where coverage is required [14]. Sigfox currently provides no coverage in Norway, as shown in figure 2.3.

Comparing the coverage of cellular bands and LoRa, two companies in Norway have approximate the same coverage and provide the best option of LTE-M and Nb-IoT, Telia and Telenor. Their coverage of the relevant technologies are shown in figure 2.4 and 2.5. From the figures about coverage, the areas lacking are mainly rural ones. A significant advantage of the LoRaWAN is its flexibility. The only requirement to expand its coverage is installing a new gateway around the desired area. From table 3 in R. Balani's "Energy

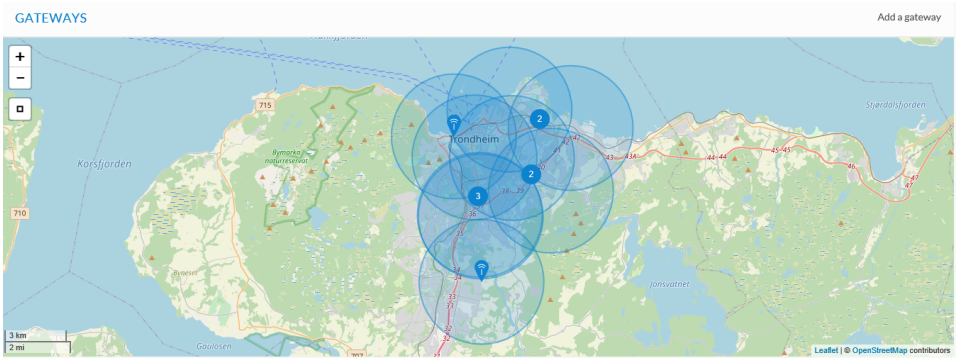


Figure 2.1: LoRa coverage in Trondheim [1]

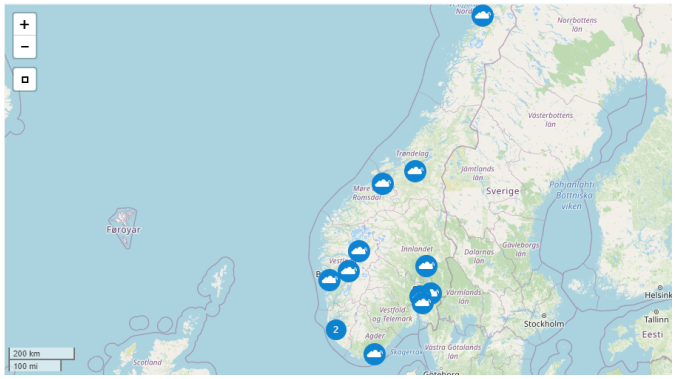


Figure 2.2: LoRa coverage in Norway [2]

Consumption Analysis for Bluetooth, WiFi and Cellular Networks" [6] paper, shown in figure 2.6, the operating range for LoRa is between 5-15 kilometers. This shows that not many gateways are required to cover a big area.

2.3 Prediction of sky movement and solar irradiance

This section is focused on existing technologies within the ability to observe and predict cloud movement or estimate it. Several methods exists for this purpose, but the two most common ones are All-sky images and weather classification and support vector machine (SVM).

2.3.1 All-sky images

This method uses a machine with camera to picture a large portion of the sky. Its pictures cover a greater area than any normal camera would be capable of doing. This is often done by a special camera pointing downward onto a spherical mirror, illustrated in figure 2.7.

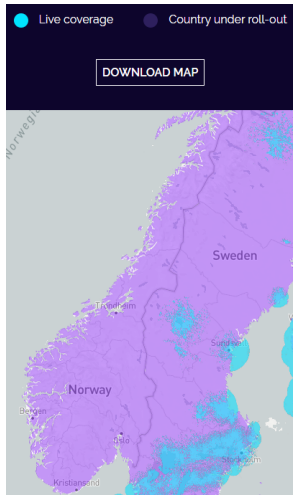
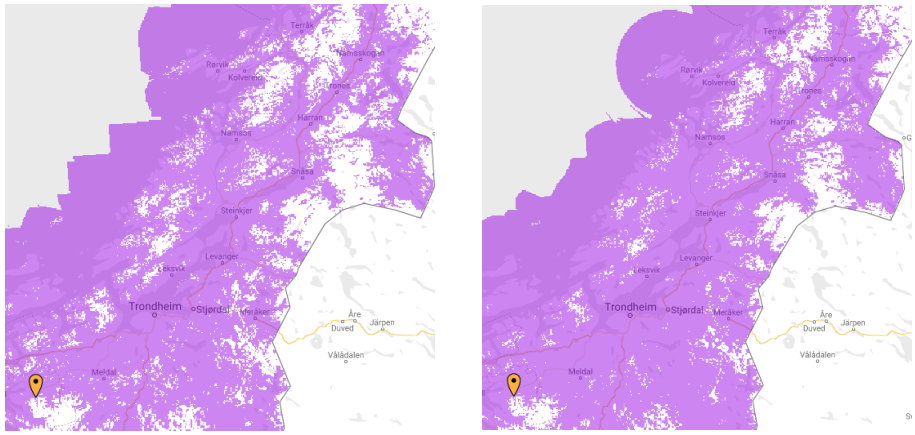


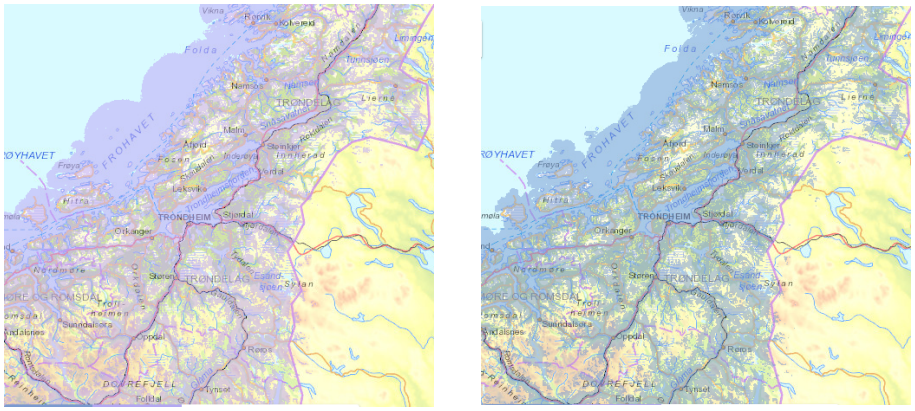
Figure 2.3: Sigfox coverage in Norway [3]



(a) LTE-M **(b)** NB-IoT

Figure 2.4: Coverage from Telia in Trøndelag [4]

The cloud detection technique used with this all-sky imager is based on the concept that clouds scatter the visible wavelengths more evenly than clear skies. In order to predict the movement, cloud velocity and direction of motion is determined using a cross-correlation method (CCM) applied to two consecutive sky images. The method predicts the movement within an hour [7].



(a) LTE-M **(b)** NB-IoT

Figure 2.5: Coverage from Telenor in Trøndelag [5]

Table 3. Particular comparison for different LoRaWAN modules.

Company	Module	IEEE Protocol	Designed for Network Protocols	V_{DD} (Volt)	I_{TX} (mA)	I_{Rx} (mA)	I_{sleep} (µA)	Bit Rate (Kb/S)	Operation Range Km
Microchip [25]	RN2483	Close Alignment with IEEE 802.15.4	LoRaWAN™ Protocol Stack	3.3	38.9* 32.9**	14.2	9.9	5.468 - 300	5 - 15 Km
Multiteh [26]	MTDOT-868-X1-SMA	Close Alignment with IEEE 802.15.4	LoRaWAN™ Protocol Stack	3.3	26 - 41	12	30.9	5.47 - 21.9	Up to 8 km
Nemeus [27]	Nemeus-MM002	Close Alignment with IEEE 802.15.4	LoRaWAN™ Protocol Stack	3.3	20 - 39.5	11.7	<2	0.3 - 40	12 Km line of sight

*Maximum transmitted power and 868MHz band. **This value for Maximum transmitted power and 433MHz band.

Figure 2.6: Table illustrating power consumption with different LoRaWAN modules [6]

2.3.2 Weather Classification and Support Vector Machines

This technique is meant to forecast the power output of large photovoltaic (PV) systems. J. Shi, Wei-Jen Lee proposes algorithms for this type of forecasting [15]. In their process, they divide the weather conditions into four: a rainy day, foggy day, cloudy day, and clear sky. Weather forecasting data, historical power output data are used with the principle of support vector machine (SVM) learning to predict the power output of the PV system. Compared to the all-sky images, they provide the possibility of one-day-ahead forecasting, instead of within the hour.

2.4 Self-powered IoT Solutions

In a network of many sensors it's highly inconvenient and at some points expensive, to frequently be required to maintain and charge the different sensors. The node should, therefore, be able to recharge itself and maintain itself to a great extent. Tore Apeland investigates this possibility in his master thesis [16]. From his thesis, two different designs

Figure 2.7: TSI 440A Total sky imager [7].

were proposed, one utilizing NB-IoT and an extra development kit (nRF9160), and the other utilizing a LoRa modem with an ATtiny817 as a central unit. His thesis does not provide a solution to a working implementation but shows the theory and a suggestion of how it can be done.

2.5 Node network synchronization

To fulfill the purpose of using data to see cloud movement, synchronization is an essential aspect of the solution. To use the data, the nodes must, therefore, transmit in sync. L. Tessaro, C. Raffaldi, M. Rossi, and D. Brunelli propose a lightweight algorithm with self-calibration for synchronizing industrial LoRa sensor networks [8]. This algorithm also compensates for clock skew. Hardware functionalities are utilized to compensate for drift over time and clock skew. The algorithm is illustrated by figure 2.8, and they were able to manage an average synchronization error of $4.54 \pm 1.28ms$. This algorithm is executed in advance before the nodes are deployed in the field. It does not provide an implementation for over-the-air synchronization after deployment.

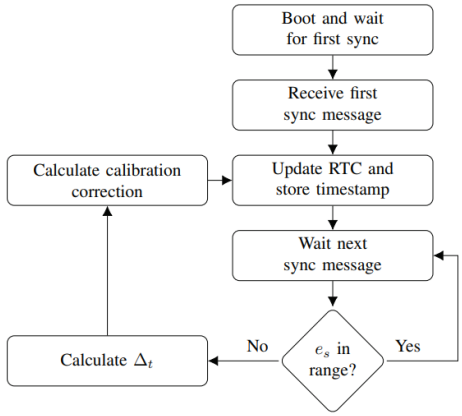


Figure 2.8: Synchronization algorithm for LoRa sensor network [8].

Chapter 3

Theory

This chapter is meant to provide the basic theory relevant to the thesis. The most important aspects to have a basic understanding of is LoRaWAN and the network server The Things Network (TTN) used to receive LoRa messages. Most of the theory below have been retrieved from a previous project completed by the author [17].

3.1 LoRaWAN

LoRaWAN is abbreviated from ”Long Range Wide Area Network” and is often mistaken for LoRa. The difference is that LoRa, compared to the OSI model, is only the physical layer, while LoRaWAN adds the data-link layer and network layer. LoRaWAN is a cloud-based MAC-layer but acts as a network layer to manage communication between the LoRaWAN gateways and end-nodes. The general architecture can be seen in Figure 3.1.

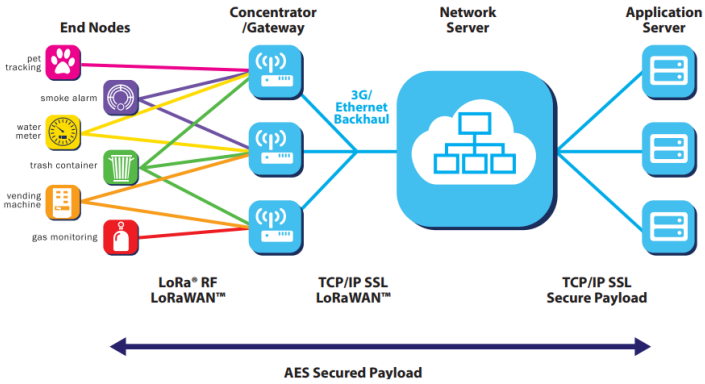


Figure 3.1: LoRaWAN architecture [9]

3.1.1 Classes

End-devices all serve different purposes and tasks. To optimize a variety of application profiles, LoRaWAN utilizes different device classes, namely classes A, B, and C.

Class A

This class is meant for all devices where battery life is of utmost importance, while not needing a full-duplex communication. This class only allows downlink messages when an uplink message has been sent, meaning the sensor can only receive a message after itself has sent one. There are specific receive windows for this, RX1 and RX2, shown in figure 3.2. The windows are static and determined by the network. For The Things Network (TTN) (see section 3.2, the first window is 1 second after the transmission, TX+1s, while the other on is TX+2s. It's important to note that the device cannot be reached by the back-end if a message has not been sent, and will instead be queued and transmitted on the next received message.

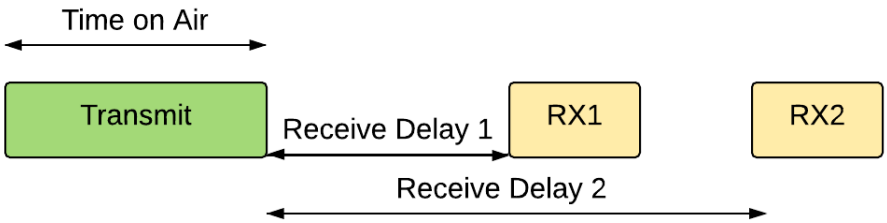


Figure 3.2: Simple illustration of LoRaWAN receive windows.

DataRate	Configuration	Indicative physical bit rate [bit/s]
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 125 kHz	11 000
7	LoRa: FSK	50 000

Table 3.1: Available data rates for LoRa end-devices, stated in LoRaWAN Regional Parameters [19].

Class B

Class B acts as a beacon with the same functionality as with class A but adding the possibility of scheduled downlink windows. This makes the device reachable at specific scheduled times as well as after each transmission. This class is suited more for less battery hungry devices which needs some more communication with the back-end.

Class C

Class C is continuous and is suited for devices that require the least amount of latency for received messages. This class has an always-open downlink window, meaning the device is always listening for downlink transmissions.

3.1.2 Messages

There are two types of messages used in LoRa communication, downlinks and uplinks. Uplinks are messages from the end-device, while downlinks are messages to the end-device. Each message either contains a payload or a join request/response. The payload size is based on which data rate used for the current device. From table 3.1, data rate 0-6 has a payload maximum size of 255 bytes, while data rate 7 has a maximum payload of 64 bytes (see RN2483 Command Reference [18]). When transmitting messages, there are two ways of doing this, either ”confirmed” or ”unconfirmed”. Transmitting confirmed messages forces the end-node to reply with an acknowledgement. With unconfirmed messages, the sender ”does not care” whether message is received or not.

3.1.3 Security

Encryption

As the security part of LoRaWAN is not too relevant for the project, it will not be explained in this project. It is worth mentioning that all messages used in LoRa transfers are encrypted with AES 128-bit encryption.

Join procedures

To secure radio transmissions, the LoRaWAN protocol relies on symmetric cryptography using two different session keys. These keys are:

- Unique 128-bit Network Session Key (nwkSKey) share between the network server and end-device.
- Unique 128-bit application session key (appSKey) shared end-to-end at the application level.

When a device is requesting to join a network server via LoRaWAN, there are two methods to achieve this, either ”Activation by personalization (ABP)” or ”Over the air activation (OTAA)”.

To join the network using ABP, the keys nwkSKey and appSKey are generated in advance and hard-coded into the devices. This is a less secure way to connect to a network compared to OTAA. Using OTAA, two 8 byte EUI's and an application key (appKey) are generated in advance and hard-coded into the device. The procedure for joining over OTAA is shown in figure 3.3. The important part is that the device and server negotiate the session keys which may result in a device being denied. Opposed to ABP where the device is automatically joined and no procedure is necessary.

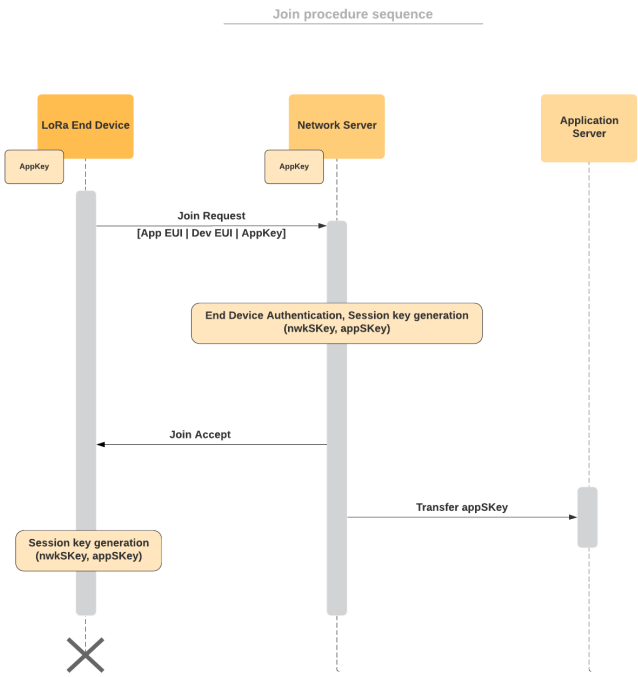


Figure 3.3: Join procedure sequence diagram

3.2 The Things network

The Things Network (TTN) is a public global network for IoT devices and gateways, using LoRaWAN only. Due to this being a public open network, anyone anywhere in the world can contribute to expanding the network.

3.2.1 TTN Console

The TTN-console is a tool for developers or anyone working with LoRa IoT devices can manage their devices and applications. Essentially the console is an online GUI. As mentioned earlier in section 3.1.3, the keys require in OTAA joining are auto-generated when adding a device to an application in this console.

3.2.2 Gateways

A gateway is considered a ”bridge” between the network and end-node. Nevertheless, the gateway and devices don’t ”know each other”, in a way that the device just broadcasts a message to any gateway that can receive it (based on distance). The gateway will receive the message and route it to the correct network. The gateway will also listen to the TTN-handler in case it has enqueued a downlink message for a specific device. This queue is very important considering some end-devices cannot be reached at any time (see section 3.1.1).

3.2.3 Application

Application is the back-end software that can be utilized to receive the information the end-node sends. TTN refers to this as an application server and provides two interfaces the application can use to interact with TTN, either HTTP or MQTT.

Chapter 4

Specification and Design

This chapter is divided into 5 parts: Functional specification, Technical specification, Acceptance criteria, Design of original solution, and Design of an alternative solution. Requirements for each part are derived in their respective sections.

4.1 General overview

This section focuses on the entire system as a whole, seen from the perspective of a single sensor. The solution intends to support both NB-IoT and LoRaWAN, but with only one protocol active at a time. The solution then consists of an end-device, external LTE modem (if LTE used for that node), a server to receive and forward LoRaWAN messages (if LoRa used for node), and a back-end server to log data and synchronize nodes.

The end node samples the current and voltage from a solar panel every second, and then transmit said data. In the case of LoRa, the message goes via TTN's server and then forwarded to a handler/back-end server for logging. When using NB-IoT, the message travels from the end-node to nRF9160-DK, which forwards it via an MQTT broker directly to the back-end server/handler for logging. When receiving a message, the server is capable of generating a callback which can transmit a message back to the end-node. Figure 4.1 illustrates the solution from one node's perspective with both protocols available.

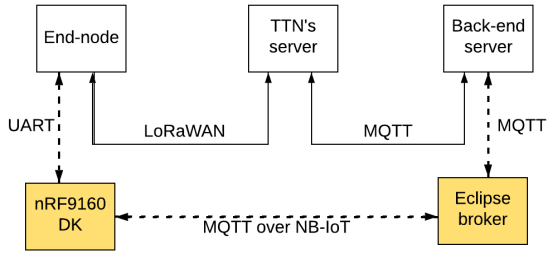


Figure 4.1: Overall system single node communication.

4.2 Functional specification

This section lists the specifications required for creating a functioning end-node network:

1. Each node is portable and operates wirelessly.
2. Each node has recharge capabilities
3. Each node should be self-sufficient on power.
4. Each node should sample an appropriate level of potential power from the solar panel when measuring.
5. Each node supports wireless 2-way communication for remote configuration of device.
6. Each node are capable of being reset.
7. Transmitted data from each node must be stored for analysis by a server.
8. A back-end application capable of synchronizing each node on time to provide comparable data.
9. Information about location and orientation should be automatically determined and transmitted.
10. Units must have option for debugging.
11. Units must to some extent be waterproof.
12. Wide coverage for easy deployment of each node.

4.3 Technical Specifications

This is the list of technical specifications corresponding to the functional ones in section 4.2, and split between the end-node and back-end server.

4.3.1 Single End-node

Hardware

1. Unit is battery-driven (pt. 1 in section 4.2).
2. Unit can be recharged from connected solar panel, or through micro-usb connector (pt. 2, 3 in section 4.2).
3. Unit must be powered during charging (pt. 3, 4 in section 4.2).
4. Charge controller capable of accepting the current and voltage range produced by solar panel and micro-usb (pt. 2, 3 in section 4.2).
5. Load circuit for excess produced power to travel (pt. 4 in section 4.2).
6. Simple battery diagnostics (provide battery voltage). (pt. 3 in section 4.2).
7. Provide Voltage and current measurement from connected solar panel to MCU (pt. 4 in section 4.2).
8. Low power consumption (pt. 3 in section 4.2).
9. Support the usage of either a LoRa modem, or nRF9160-DK for LTE modem (pt. 5, 12 in section 4.2).
10. Unit must have a GNSS and 9-axis accelerometer (IMU) for determining location and orientation (pt. 9 in section 4.2).
11. Unit must reset locally (button) (pt. 6 in section 4.2).
12. Unit must be contained in a waterproof box. (pt. 11 in section 4.2).
13. Option for JTAG interfacing to debug and program unit (pt. 10 in section 4.2).
14. Multiple test points on PCB for debugging (pt. 10 in section 4.2).

Software

1. High duty cycle in sleep mode for save energy.
2. Message interface for uplinks and downlinks (both LoRa and NB-IoT) (pt. 5, 6 in section 4.2).
3. At desired interval, detect and store wanted measurement data (pt. 4, 6 in section 4.2).
4. Able to transmit stored data to the back-end server using NB-IoT or LoRaWAN (pt. 5 in section 4.2).
5. Capable of executing commands sent from the server (pt. 6, 5 in section 4.2).
6. Be able to Synchronize with server (pt. 8 in section 4.2).
7. Routine for retrieving and determining its angle, orientation and location on startup. (pt. 9 in section 4.2).

4.3.2 Server/Handler

Technical specifications for the back-end server corresponding to relevant functional specifications.

1. Must be able to access a time-server for self-synchronization (pt. 8 in section 4.2).

2. Must be able to have a callback for fast response to all nodes (pt. 5 in section 4.2).

3. Should be able to run a script to synchronize all nodes to a specific time (pt. 8 in section 4.2).

4. Must support receiving messages from LTE-devices and LoRaWAN-devices (pt. 5, 7, 12 in section 4.2).

5. Must have timer functionality.

6. Must be able to log received data (pt. 7 in section 4.2).

4.4 Acceptance Criteria

Based on the technical requirements in section 4.3, a list of acceptance criteria have been derived and are shown in table 4.1. These criteria are the are required to be passed for a system to function as intended.

Label	Description
AC1	End-node transmits measurement data and device diagnostics over LoRaWAN or NB-IoT
AC2	Server stores received data with correct timestamp in correct file.
AC3	End-node are self-sufficient and battery should not lose any % over the course of three weeks.
AC4	End-node measures considerably higher current that charge-circuit is capable for is sunny weather when measuring.
AC5	End-node can be deployed anywhere with LoRaWAN or NB-IoT coverage.
AC5.5	End-node is considered having proper coverage when a total message loss is under 10%.
AC6	Server is capable of synchronizing all active nodes to specific timestamp.
AC7	End-node must reset, change transmission interval and timestamp when receiving such commands from server.
AC9	End-node enters sleep-mode when idle
AC10	End-node supports interface with peripherals over I2C, UART and SPI.
AC11	End-node can survive outside in rain indefinitely
AC12	End-node must be able to be recharged using micro-usb and by produced power from a solar panel.

Table 4.1: Acceptance criteria for entire system

4.5 Design

4.5.1 Original Solution end-node

This section covers the design of the original idea. It will focus on the single node and server, and not consider the node network as a whole. This is due to there not being any visual aspects to it, as well as it only matters on the back-end. The context diagram in figure 4.2 illustrates the mechanics and which parts the printed circuit board (PCB) interfaces with. The pink box in the figure is only added in the case of NB-IoT, otherwise, LoRaWAN is utilized and the PCB will communicate directly with the server without an external board. In the case of NB-IoT, the nRF9160-DK is required to provide an LTE modem and will act as the central processing unit.

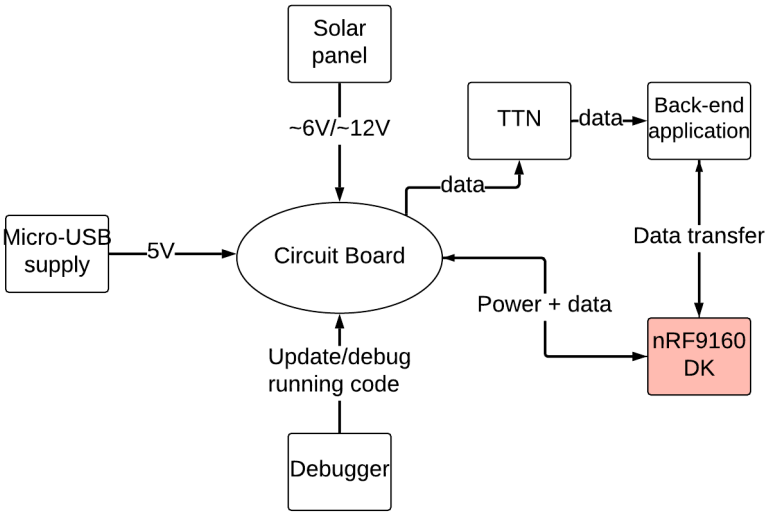


Figure 4.2: Context diagram for original solution

Simplistic PCB overview

The complete PCB design is explained in section 5.1, and will not be gone into detail here. This section merely covers a simplistic overview to illustrate the idea behind it based on the technical specifications in section 4.3. The design is illustrated in figure 4.3. As the figure shows, the board can be charged by two methods, solar panel, and micro-USB. The board is entirely battery-driven and uses two different voltage regulators to power the rest of the circuitry. This includes the external nRF9160 in case NB-IoT is being used. It's planned to use three buttons, 2 of which are of dummy behavior (whatever desired), and one RESET button, used to reset it locally. To visualize different statuses, like functioning properly, for example, three LEDs will be added. Lastly, the remaining available GPIO pins will be mapped to headers, providing extra functionality for other peripherals or testing.

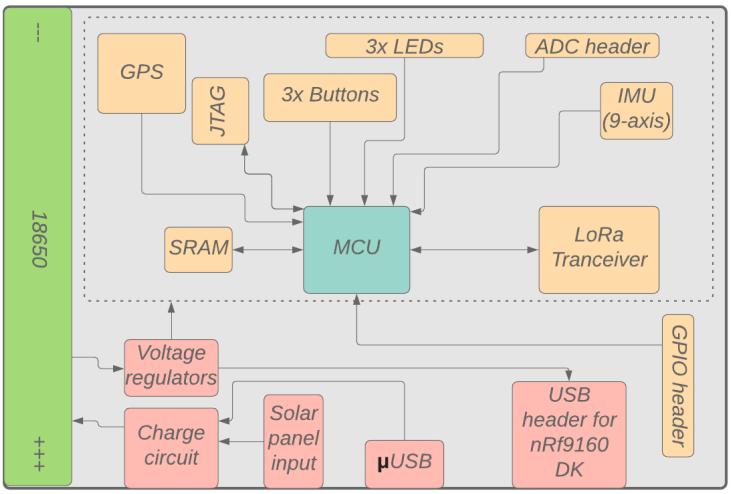


Figure 4.3: Simplistic PCB design original solution.

4.5.2 Alternative Solution end-node

This section covers the alternative solution used for most of the implementation in this thesis due to reasons mentioned in the introduction. Therefore this solution does not cover a PCB design, as another existing board has been utilized. T. U. Rasmussen uses in his study [20], a quite similar board, which made the transition to a new solution more comprehensible.

Compared to the context diagram in the original solution [4.4], certain design choices were made. The dotted rectangle illustrates the hardware components required to merge his solution with mine. The main components and features his solution is missing are:

1. GNSS
2. IMU with magnetometer for orientation
3. Charging capability from solar panel.
4. Measurement of potential energy from solar panels.

T. U. Rasmussen's board only accepts 5V for charging the battery. It is therefore necessary to determine an approach for converting the given voltage range from a panel to 5V.

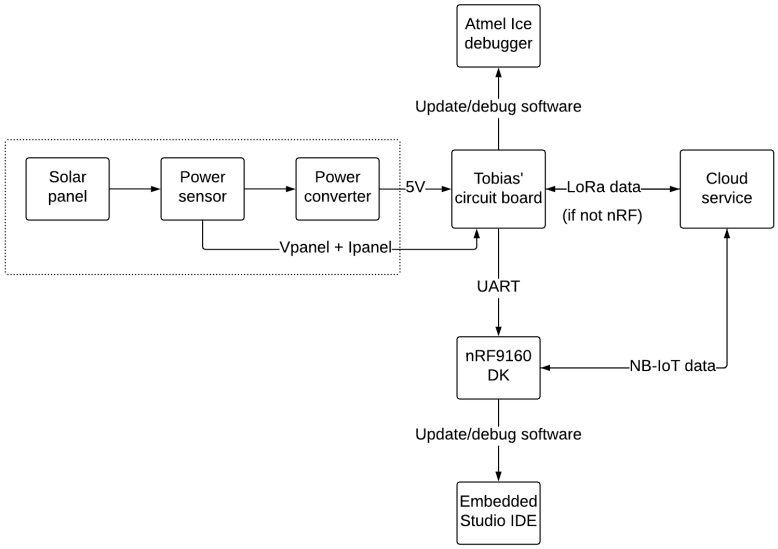


Figure 4.4: Context diagram for alternative solution

4.5.3 Back-end application

The back-end server for this thesis needed to provide an option to communicate with end-nodes over different communication protocols, LoRaWAN, and NB-IoT. Figure 4.1 shows The Things Network's server, a server required to receive and forward LoRa messages. Normally, this could be achieved by routing your own LoRaWAN gateway straight to the back-end application, but this limits the coverage to that specific gateway. Using TTN's handler/server, allows this application to use a preexisting network governed by TTN. This coverage is shown in figure 2.2. TTN's server has the option of forwarding their messages via MQTT. They also provide such API developers can adopt to implement their application.

Another limitation of this thesis is the use of nRF9160 DK to add LTE functionality. This modem can use MQTT to send data over NB-IoT. In order to achieve this type of communication, an MQTT broker needs to be configured. For simplicity and time-saving reasons, the public MQTT broker "mqtt.eclipse.org" was utilized.

This back-end server does therefore only require MQTT support to communicate with any devices used in this application.

4.5.4 Message format

In order to communicate seamlessly, a simple message format was designed. This format is illustrated in figure 4.5

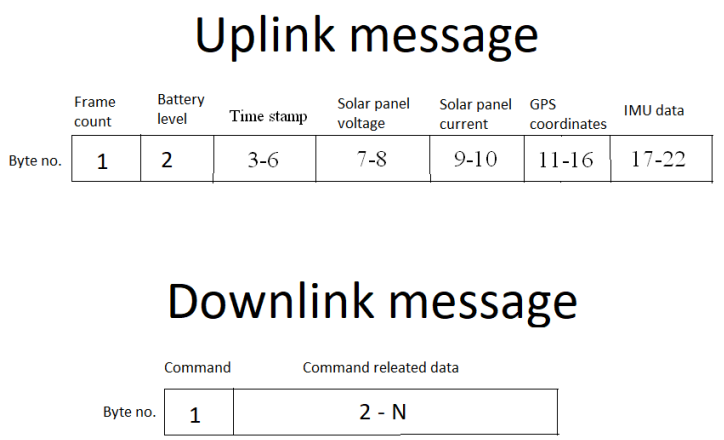


Figure 4.5: Message format used in communication between nodes and server.

Chapter 5

Implementation

This chapter provides a detailed explanation of how the design in the previous chapter was realized. The chapter will focus on the single node, both hardware, and software, as well as the back-end application. The single-node part was again split into two sections, original solution and alternative solution. Explaining each solution, the hardware is split into first explaining the choice of a component and then its circuitry. As all software was written for the alternative solution, the software section will only cover this. As the components of both solutions are very similar, it was not deemed necessary to provide a new section explaining the software for a potential original solution. Instead Appendix A shows how to port software from the alternative solution to the original one.

Consistent with both solutions was the choice of solar panels. It was ordered two different types, with different maximum power production. Physical differences are shown in figure 5.1 while technical differences are listed in the following table: Where $V_{open\_circuit}$ for both panels were detected by measuring with a multi-meter with no load connected. Similarly $I_{short\_circuit}$ were detected by connecting the multi-meter in series, both measurements done on a day with clear skies and sunny weather.

Through the course of developing the PCB and schematics for the original solution, *Altium Designer* was the main tool. There, most components were of type Surface Mount Device (SMD), meaning they were soldered with solder paste directly to the PCB, without the use of holes. The software was also used to visualize general circuitry for the alternative solution.

	$P_{optimal}$	$V_{optimal}$	$I_{optimal}$	$V_{open\_circuit}$	$I_{short\_circuit}$
Type 1 (big)	4.2W	12V	350mA	16V	400mA
Type 2 (small)	2.0W	6V	333mA	7.5V	380mA

Table 5.1: Solar panels used for this thesis (Type 1: [21], Type2: [22]

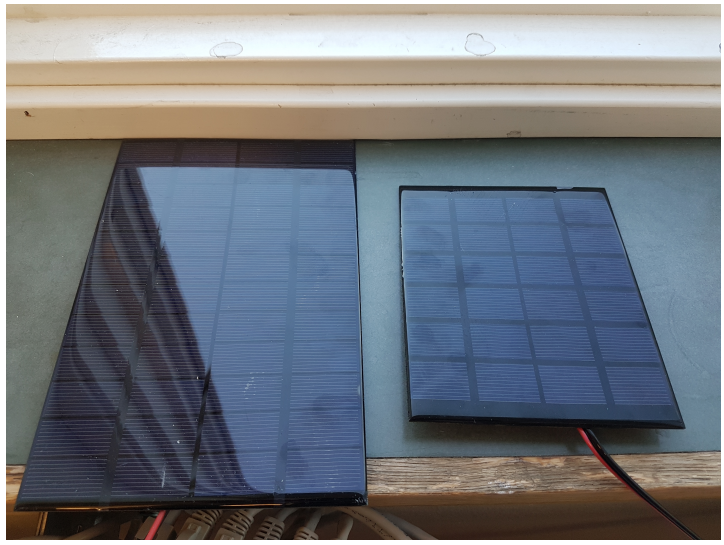


Figure 5.1: The two ordered solar panel types. Type 1 to the left, and type 2 to the right.

5.1 Single Node Hardware with Original Solution

The full schematic for the end-node are located in Appendix B1, with corresponding parts list in Appendix B3. This section goes into detail of which components were chosen for a generic embedded end-node in order to realize the relevant acceptance criteria in section 4.4. Even though this solution is not the main focus for results and testing, due to circumstances mentioned in the introduction/preface, a PCB for this solution was at some point created and to some extent tested. The scope is to use LoRa and NB-IoT in a distributed embedded system and transmit useful data about the potential energy of connected solar panels for analysis, in other words: a self-sufficient battery-driven smart sensor that supports multiple communication protocols.

5.1.1 Components

This section explains which main components were used to complete the PCB and why they were chosen. Some of the minor components in the immediate vicinity of the major ones are further explained in later sections, as well as calculation required for them to operate ideally. The components are also seen in Appendix B3.

MCU: ATmega324pb

The main reason for choosing this micro-controller was due to it being a very familiar system for the developer. Microcontrollers in the AVR family, especially ATmega series were used frequently in different courses in earlier years of university due to them being well established in the world of microcontrollers and are simple to implement. In addition, ATmega324pb was also in the earlier specialization project conducted in spring 2019 [17].

The ATmega324PB micro-controller utilizes the AVR-based RISK architecture and is an 8-bit high-performance processor [23]. It contains 1 kB EEPROM storage with 2kB SRAM. More significantly, it features two separate I2C buses, three UART, one SPI, and multiple available GPIO pins for analog and digital purposes. In addition, the MCU supports the external clock source for the processor. This solution does not utilize the possibility for external clock source to reduce power consumption but instead uses an internal clock source division which reduces the default internal clock of 8MHz to 1MHz. The MCU features multiple timers, both 8-bit and 16-bit. "TIMER2" is here of biggest relevance, because it supports the option of the external clock source. This makes "TIMER2" ideal for creating a real-time clock (RTC) by connecting a 32.678 kHz crystal. The circuitry around the MCU is shown in figure 5.2

Circuitry The MCU uses a passive low pass filter between AVCC and VCC to GND because VCC is supplying the ADC as well as contributing as a reference voltage for all conversions. It takes a total of three analog voltage inputs, which are voltage and current measurements from the solar panel (V_{panel} and V_{curr}), and the battery voltage (V_{bat}. Since V_{panel} and V_{bat} outrange the reference voltage of the ADC voltage division for both inputs were introduced and calculated using standard voltage division formula:

$$V_{out} = \frac{R_B}{R_A + R_B} * V_{in} \tag{5.1}$$

where R_B represent the resistor connected to ground.

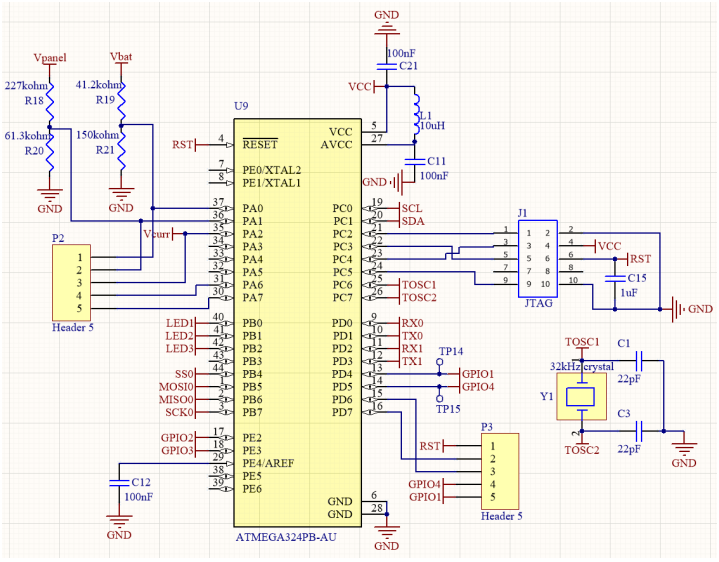


Figure 5.2: Schematic of immediate circuitry around ATmega324PB.

LoRa modem: RN2483A

RN2483A is a low-power LoRa transceiver developed by Microchip and based on the LoRaWAN Class A protocol. The modem is familiar due to being used in the previous project [17]. By integrating a command API, it's suitable for interfacing with an external host MCU. This communication is done over UART, it features an implemented protocol stack for class A [18]. Voltage requirements lie between 2.1-3.6 V. It can also enter sleep mode where it only consumes $2 - 26\mu A$ [24], depending on the input voltage. The schematic in Figure 5.3 shows the implementation of RF transmitter RN2483A.

Circuitry As seen in Figure 5.3, the modem is interfaced with UART0 on the MCU, which also shares the global reset (RST) line. C19 and C20 are the two required decoupling capacitors between voltage input and ground. RFH and RFL, the two radio outlets, are connected to a micro coaxial RF-connector to which the antennas can attach. They correspond to the frequencies of the modem which are European standard frequencies 868MHz (RFH) and 433MHz (RFL). The end-nodes will in this application be in Norway, thus only requiring the 868MHz band.

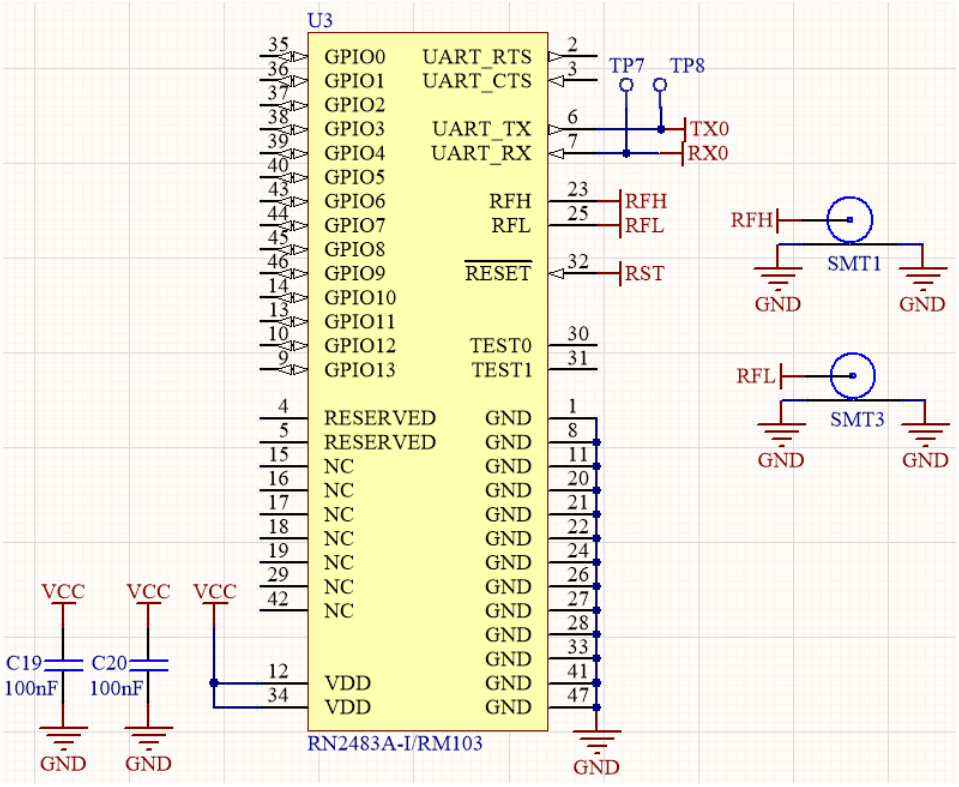


Figure 5.3: Schematic of circuitry near and including RN2483.

NB-IoT modem: nRF9160-DK

As mentioned in section 1.3, the solution was from the start going to utilize the nRF9160-DK as the NB-IoT modem. The development kit is a highly functional PCB with a lot of functionalities. It features an internal GPS that will be utilized in end-nodes meant to use NB-IoT instead of LoRaWAN. From its datasheet [25] its designed for low-power consumption with "IDLE" power consumption of $1.8\mu A$ (modem off).

The only circuitry relevant for the NB-IoT modem is its connections to the main board. Those are illustrated in figure 5.4, and consist of a power supply of 5V, angle, and orientation sent over SPI and the analog measurements from the solar panel sent as analog input signals.

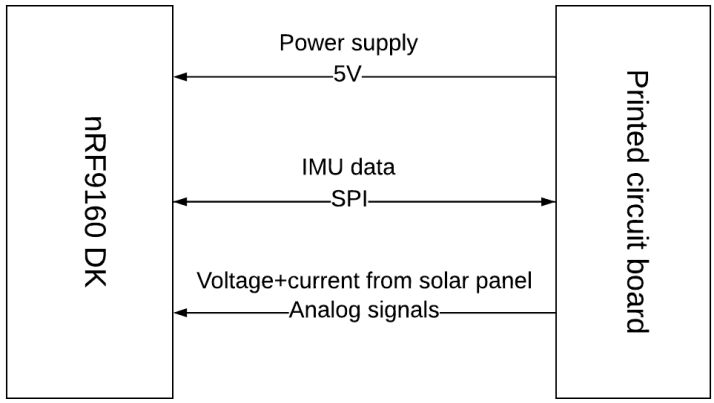


Figure 5.4: Connections between the nRF9160-DK and original PCB.

Current sense amplifier: MAX44284

In order to measure the current produced by the solar panel, a current sense amplifier was deemed necessary. As it can operate between 1.7-5.5V and a supply current of $21\mu A$, it a good fit for battery-driven applications [26]. It offers high precision and accuracy as well as an input common-mode voltage range between 0-36V. It presents four different gains to provide accurate output:

1. $G_1 = 50V/V$ - MAX44284F
2. $G_2 = 100V/V$ - MAX44284H
3. $G_3 = 200V/V$ - MAX44284W
4. $G_4 = 500V/V$ - MAX44284E

Circuitry for how its setup are illustrated in figure 5.5.

Circuitry In order to calculate the value of R_2, the highest available voltage drop over it needs to be calculated. From its MAX44284's datasheet [26], the voltage drop (V_{sense_range}) is calculated by the following formula:

$$V_{sense\_range} = \frac{V_{dd}}{G_i} \tag{5.2}$$

V_{dd} was in this case the voltage supply and 3.3V. A gain of 50 was chosen, which resulted in a V_{sense_range} of 66mV. R_2 was then calculated using Ohm's law and procedure is shown in eq. 5.1.1

$$V_{shunt\_max} = V_{sense\_range} = I_{short\_circuit} * R_2 \tag{5.3}$$

$$\Rightarrow R_2 = \frac{V_{sense\_range}}{I_{short\_circuit}} \tag{5.4}$$

$$Substitutinginusingeq.5.1.1, \tag{5.5}$$

$$\Rightarrow R_2 = \frac{V_{dd}}{I_{short\_circuit} * G_i} \tag{5.6}$$

With this formula, V_{dd} as 3.3V, $G_i = 50$ and $I_{short\_circuit}$ retrieved from table 5.1 (333mA and 350mA), gives $R_{2\_type1} = 0.189\Omega$ and $R_{2\_type2} = 0.198\Omega$. To be flexible in attaching a random solar panel to an end-node, the lowest value for R_2 is used, and it's floored to the nearest available resistor. In this case, that value was 0.18Ω.

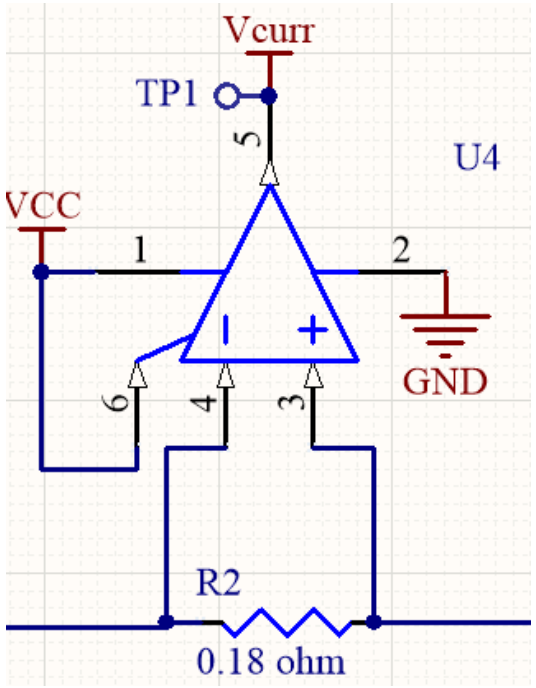


Figure 5.5: Schematic showing the circuit for MAX44284.

GNSS: Teseo-LIV3F

Determining a good and effective GNSS was a more challenging task. The main focus being low power consumption and a straightforward interface to communicate over. It was also important the necessary antenna circuitry not became to complex. The Teseo-LIV3F, developed by STMicroelectronics, became a natural choice befitting desired specifications in section 4.3. It operates at 2.1-4.3 V input and supports both UART and I2c, making it fairly easy to transmit data to external host MCU. If necessary, it has an RTC which could be utilized. Based on the input voltage, it has a default standby current of $17\mu W$ consumption [27]. With a tracking sensitivity of -163 dBm and 1.5m circular error probability (CEP), the component proves well-suited for this application. The circuitry required for the component to be operational are shown in figure 5.6 and are based on suggestion schematic from its user manual [28].

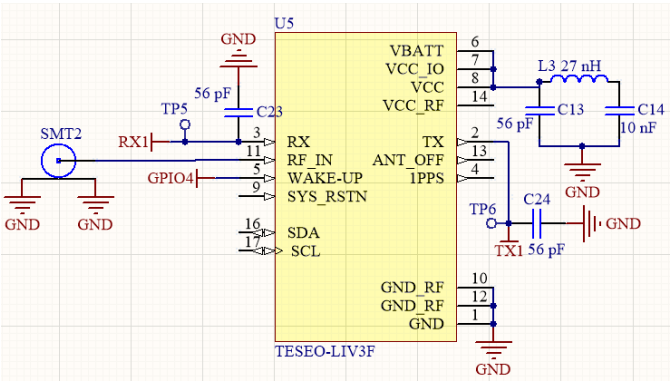


Figure 5.6: Schematic showing the circuit for the GNSS (TESEO-LIV3F).

IMU: ICM-20948

The ICM-20948 IMU was developed by TDK InvenSense and consists of a 3-axis gyroscope, 3-axis accelerometer, and 3-axis compass/magnetometer. It features the option for SPI or I2C interface to an external MCU. The operating voltage range is between 1.71-3.6V, providing a good range. The chip can enter sleep-mode having a current consumption of $8\mu A$ [29], or operating accelerometer and magnetometer only with a consumption of $158.9\mu A$. The accelerometer features four levels of sensitivity: $\pm 2g$, $\pm 4g$, $\pm 8g$, and $\pm 16g$, and the compass has a wide range of $\pm 4900\mu T$. Each data sample is stored in 2 8-bit registers and read like a 16-bit integer. The circuit is shown in figure 5.7.

Circuitry Most of the circuitry is based on suggestion schematic from its datasheet [29]. The main difference being the voltage divider added (R11, R12), to supply the VDDIO pin with the correct voltage. As VDD range from 1.71-3.6V, the VDDIO range from 1.71-1.95V, and thus require a voltage divider to reduce the voltage. Using eq. 5.1, R11, and R12 succeed in providing correct voltage while minimizing to some extent drawn current.

As the IMU also supports auxiliary devices as a slave over I2C, but not required, those signals were pulled low.

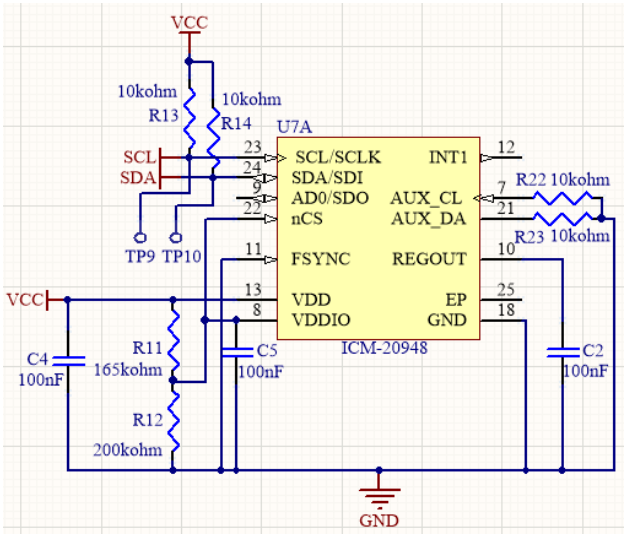


Figure 5.7: Schematic showing the circuit for the IMU (ICM-20948).

Flash memory: SST25VF080B

The SST25VF080B holds up to 1MB of data and has a standard endurance of 100,000 write cycles. For instance, giving the likelihood of storing minute-measurements of 17 bytes, it can hold data up to 40 days. The device uses an SPI interface, making it friendly to an external host MCU. It's also very battery friendly as it in standby mode only consumes up to $5\mu A$, and operates at 2.7-3.6 V.

Charge controller: BQ24210

As each node is supposed to be operational at all times during the day, the controller has to support a wide range of voltage input, ranging between 0V at night, to 16V during optimal conditions. This controller features a battery tracking mode to maximize the charge rate from solar panels. It can support a current up to 800 mA with a maximum rating of voltage input between 0 and 20V [30].

Voltage converter: TPS61201DRCT

After comparing the different voltage operating ranges of the other components, an output voltage (VCC) was deemed to 3.3V. The TPS61201DRCT provides a selectable output voltage of 3V3 or 5V0. It is rated for a 300 mA output current at 3V3, which should be quite above the required supply for the rest of the board. With a low quiescent current, at $55\mu A$ it suits this application well [31].

Voltage converter: MCP1253T

The MCP1253T provides a buck/boost DC/DC converter with a low power consumption rated at $80\mu A$. It achieves up to 120 mA output current with a voltage input range between 2.0-5.5V. It's capable of being in shutdown mode reaching a reduced power consumption down to $0.1\mu A$ [32].

5.1.2 Power circuit

The part of the schematic that is considered "power circuit" covers all from connector P1 (solar panel), through the charge controller (U6), pass the battery, and until converted into both 3V3 and 5V0, and fully illustrated in Appendix B4. U4 (MAX44284) have already been explained in section 5.1.1. Visualized in the figure, the power is drawn from two different connectors, either a micro-USB type (J2) or solar panel (P1). J2 is connected to the controller via a Schottky diode, to prevent backward current through the USB, and provide 5V to recharge the device.

Charge controller circuit - BQ24210DQCT

Circuitry for the charge controller is primarily based on the suggestion in datasheet [30]. The thermistor (RTH) is added to provide the controller with temperature sensing. $\overline{PG}$ is connected directly $\overline{EN}$ due to the controller being used in battery tracking mode instead of load mode [30]. R_7 is attached to set a maximum fast-charge current. Using the formula from the datasheet,

$$R_{ISET} = R_7 = \frac{K_{ISET}}{I_{ISET}} \tag{5.7}$$

where K_{ISET} is 472, and I_{ISET} were based on strongest possible current, 400 mA resulting, resulting in $R_7 = 1.18k\Omega$. $\overline{PG}$ and $\overline{CHG}$ are input signals used in series with diodes (D1 and D2) to providing feedback on weather device is charging and if power applied is good enough. Their truthtable can be seen in the datasheet [30].

Load circuit

The strongest solar panel is capable of producing a power up to 4.2W, therefore a load capacity of that power was required. R5 and R6 are therefore two power resistors, with a power dissipation up to 3W each.

5.1.3 Headers and peripherals

From schematic in Appendix B1, D4, D5, and D6 were three diodes added to provide the status of the device while operating. Due to wrong order, three red LED's were ordered and soldered. The buttons SW1 and SW2 were added to trigger an external interrupt, to provide extra tools when testing and programming. The RST button causes a hard-reset on both the MCU and LoRa transceiver, by pulling the reset pin to GND. For testing purposes and the possibility of new functionality in the future, all remaining free pins from the MCU were connected to corresponding headers. In addition to remaining free pins, three of the

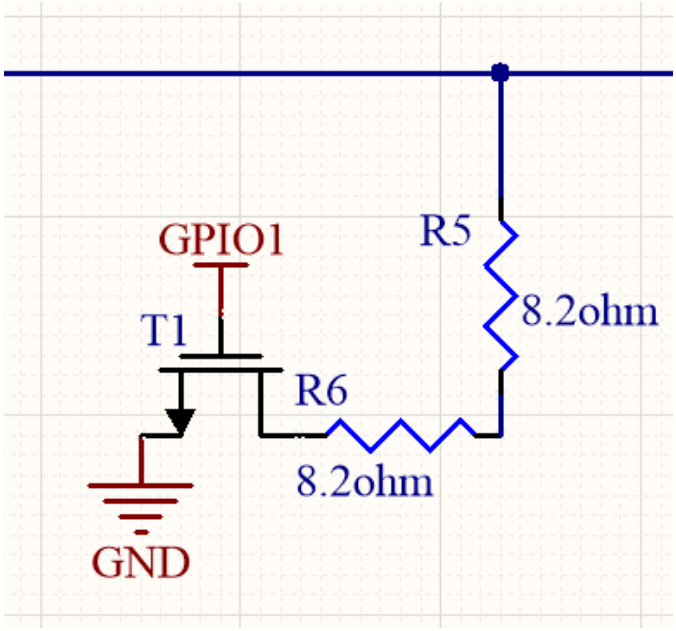


Figure 5.8: Schematic load circuit.

measured signals (V_{panel}, V_{bat} and V_{curr} were also connected to a header (P2). This was because, in case of using nRF9160 as a central processor unit, it required those signals to take measurements).

5.2 PCB results

The assembled PCB is shown in figure 5.9 with its PCB design shown in Appendix B2. The PCB was created and designed in *Altium Designer*, and prototype ordered from *JLCPCB*[1]. Upon receiving ordered prototype, it was soldered at NTNU and partly tested.

5.3 Single Node Hardware with Alternative Solution

The alternative solution consists of using T. U. Rasmussen's PCB, with the addition of a few breakout boards to achieve the desired functionality. The schematic for the implemented hardware (not including the PCB), can be found in Appendix C1, with corresponding parts and components list in Appendix C2. This solution prioritized energy measurements and did not provide any support for the angle, orientation, and location data. The central processor unit, ATmega324PB was used for all nodes, even when using NB-IoT. In the case of NB-IoT, Tobias' PCB was used for all measurements, with Atmega324PB

[1] www.jlcpcb.com

Figure 5.9: Assembled end-node without battery for original solution.

as the main processor unit. Data was then transferred to the nRF9160-DK over UART and forwarded to the server by MQTT over NB-IoT.

5.3.1 Components

This section covers the components used in the alternative solution.

T. U. Rasmussen's Circuit Board

The PCB used in T. U. Rasmussen's thesis [20], consists of most of the functionality desired for this task. It provides the same LoRa modem and central processor unit. The main shortages of the PCB, are its lack of support for variable voltage charging. His charge controller required a constant 5V input. It does not provide a solution to measure current and voltage from the panel. Nevertheless, it provides available analog inputs as well as interfaces like I2C and UART available of communication with potential external sensors.

NB-IoT/CAT-M1 modem: nRF9160 DK

As mentioned in section 1.3, the solution was from the start going to utilize the nRF9160 DK as the NB-IoT modem. The development kit is a highly functional PCB with a lot of functionalities. From its datasheet [25] its designed for low-power consumption with "IDLE" power consumption of $1.8\mu A$ (modem off). Nevertheless, low power consumption for the NB-IoT modem was disregarded in the alternative solution and therefore powered by a power-bank/wall outlet.

The only circuitry relevant for the NB-IoT modem is its connections to the main board.

For the alternative solution, UART1 was utilized from T. U. Rasmmussen's PCB to communicate with the modem.

INA219: High Side DC Current Sensor

The essence of measuring potential energy lies with the current produced from the panel. INA219 provides both current and the voltage produced from it. It uses the same principle in the original solution, measuring the voltage across a shunt resistor to calculate the current. The sensor supports a voltage bus range from 0-26V, making it available to measure both panels [33].

COM-15208: Buck-boost converter

The core component of the COM-15208 breakout-board is the TPS63070 buck-boost converter. This converter supports an input voltage range between 2.0-16V and an output voltage range between 2.5-9V. With a fixed output voltage at 5V, its power consumption in standby mode is typical $54\mu A$, which can be reduced to typically $2\mu A$ in shutdown mode [34].

Load circuit

In order to enable and disable the load circuit, a n-channel MOSFET transistor (T1: [35]) were connected between the power resistors and ground (GND) (see fig. 5.8). This transistor was chosen due to its simple characteristic of acting as a switch. It was capable of drawing 500 mA, with a drain-source voltage (V_{DS}) up to 60V.

Based on results and testing (explained further in chapter 7), another implementation was also used. Replacing T1 with a bipolar NPN transistor (2N6178: [36]) could lead to better results. To use this transistor, a resistor between GPIO1 and the base pin was required, this resistor was implemented with both 220Ω and 470Ω.

5.3.2 Result

The resulting product is shown in fig. 5.10. The hardware was soldered onto a perfboard in order to provide better robustness as opposed to using a breadboard.

5.4 Sensor node firmware

This section covers the software for the end-node. These drivers were mainly developed from scratch, except for the I2C driver. This driver was still modified to suit this solution. In addition, some software was also imported from an earlier completed project [17], but greatly modified.

All development for the end-node software was completed with *Atmel Studio 7.0* as the main tool. Atmel Studio provides a lot of advantageous functionality, like device-programming (setting fuses) and a well known functional debug tool. All software for the end-node was written in C as this was deemed more practical when programming

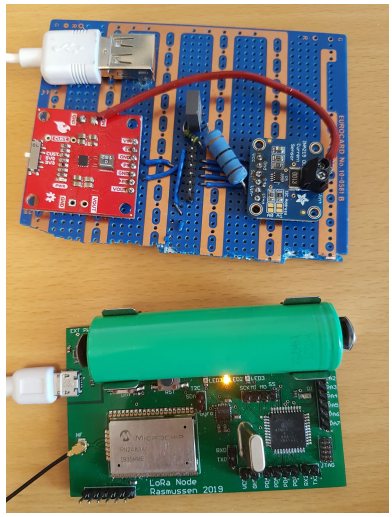


Figure 5.10: Assembled and connected end-node with battery for alternative solution.

embedded hardware, using the AVR-GCC compiler. The software has been thoroughly tested, but the main flow of the program still possess a few minor bugs.

5.4.1 RTC

From acceptance criteria 6 in section 4.4, timestamp functionality were required. In order to provide this, an 8-bit RT-timer was implemented on the ATmega324PB. This was implemented using *timer 2* with an external 32.768 kHz crystal. This frequency is perfectly divisible with 2 (2^{15}), making it able to count seconds. This was completed by using a prescalar of 128, reducing the number of ticks to 256 and overflowing every second. The end-node was implemented with an internal counter to keep track of a timestamp and transmit said timestamp as a part of the message. This functionality was later simplified and disregarded, further explained in section 5.5.3. Nevertheless, each node uses the real-time counter as an interrupt to do a measurement as well as keeping track of when to send a message and waking up for sleep. The real-time counter with interrupt is shown in fig. 5.11.

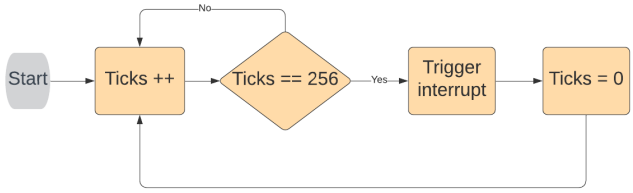


Figure 5.11: Flow chart of real-time timer.

5.4.2 Drivers

UART

There are three available UART modules on ATmega324PB, two of which are used for the alternative solution, namely RN2483A or nRF9160 DK interfacing and debug/printing.

When using LoRa, the transceiver is interfaced with the ATmega over UART0 to receive and send commands. The drivers provide generic functions to transmit single or an array of characters, as well as receiving single or an array of characters. The MCU utilizes these functions in order to communicate with the transceiver. The second module, UART2, is then used for debugging with *printf* functionality to a terminal.

When using NB-IoT with nRF9160 DK, UART0 is not utilized. Instead, only UART2 is used to interface with the modem. This is because the modem requires a header to attach cables with, and UART2 was available on the circuit board [20]. NB-IoT is not as restrictive as LoRa in class A and can receive commands at any time. To allow the nRF9160 to transmit received commands to the ATmega324PB, UART2 was configured with the possibility of waking the ATmega from sleep by transmitting a start condition and then receiving the command.

All printf statements were coated with a ”#ifdef DEBUG_M” statement. In order to then have the end-node live printing to a terminal, the code line ”#define DEBUG_M” in ”config.h”, has to be uncommented. This saved time and power in order to debug with printing or not.

RN2483 - Tranceiver driver

This driver inherits and utilizes the functionality from the UART driver. It features functions for OTAA join procedures, setting data-rates and channel duty cycles, and changing the baud rate. Upon transmitting a correct command to the modem, a response is always transmitted back. It was therefore created a small list of available responses (based on a list from the command reference document [18]), in the header file which were used to provide error information when required.

ADC

The ATmega324PB provides a 10-bit resolution ADC with 7 channels. A simple driver was developed to read different analog inputs, like battery level for the alternative solution, and also voltage and current produced for the original solution. The functions featured initialize the ADC, reads a specific channel, and changes the channel.

INA219 current sensor

This driver was developed to communicate with the high-side current sensor. The sensor interfaces over I2C. Each register returns a 16-bit value of which needs to be converted to provide useful information. This information is mainly converted at the back-end application but using the formulas explained here. The sensor provides a wide range of options that can be configured for optimal operability. It uses a 12-bit ADC used for all conversions. By configuring the sensor using software, appropriate scales for the ADC were

applied. For instance, the bus voltage can be measured from 0-16V or 0-32V[2]. In this driver there are multiple procedures for configuring the sensor, reading desired registers, and triggering a conversion.

The configuration register is a 16-bit register where all configurations are written to. The three most important configurations were the bus voltage range, shunt voltage drop range, and when to do a conversion. Conversion could happen either continuously or when triggered. To save power, each conversion was triggered by software.

The bus voltage scale was set to 0-16V for more accurate measurements and within the solar panels range. The measured bus voltage is then stored in 12 of the 16 bits in its register (bit 3-14) and is processed with simple bit-shifting operations after retrieving the 16-bit value

To measure the current, the sensor calculates the voltage drop over the 0.1Ω shunt resistor (R_{shunt}). For a most accurate conversion, it could be configured for different scale ranges:

- 0-40 mV
- 0-80 mV
- 0-160 mV
- 0-320 mV

As the absolute maximum current possible through is 400 mA, maximum voltage drop (V_{shunt}), is:

$$V_{shunt} = I_{short\_circuit} * R_{shunt} = 40mV \tag{5.8}$$

and the correct scale is 40 mV.

The last important configuration is the calibration register. The sensor is not capable of detecting R_{shunt}, and therefore uses the *calibration register* to provide required information. This register is 16-bits, but because it also supports bidirectional current, MSB indicates the sign. The register was calculated using the formula from INA219's datasheet [33]:

$$\text{Calibration Register} = trunc(\frac{0.04096}{LSB_{current} * R_{shunt}}) \tag{5.9}$$

where $LSB_{current}$ is a desired step-size for the current register where measured current is stored, and follows the formula:

$$\frac{I_{max\_expected}}{2^{12}} < LSB_{current} < \frac{I_{max\_expected}}{2^{15}} \tag{5.10}$$

$LSB_{current}$ is then chosen as high as possible. Using $I_{max\_expected} = 400mA$, $LSB_current$ was chosen to be $15\mu A$. Calibration register was then calculated to be 273066.

[2]From datasheet [33], only the scale changes, the sensor can never exceed 26V on the bus

Board driver

This driver was implemented to modulate and gather certain functions to make the code more readable. It features functions for GPIO management like initializing needed GPIO and pin interrupts to buttons, getting battery percentage (converting read ADC value to percentage) and in general setting up the board.

Utility functions

This driver contains functions for providing simplicity for the other drivers if needed. The main functionality is to encode and decode the messages sent and received. It also has the option to encode floats into hexadecimal, but this was not used in the final version.

General configuration

The entire software solution was designed using the logic provided by the configuration file ”config.h”. The purpose of this file was to provide simple matters of enabling or disabling features or even entire modules by commenting singe lines. A snippet of the code is shown as an example in figure 5.12.

```
#ifndef F_CPU
#define F_CPU 8000000UL
#endif

/*******        TIMER FUNCTIONALITY         *******/
//#define TIMER1
#define TIMER2


/*******        LORA FUNCTIONALITY          *******/
//#define LORA_NODE
#ifdef LORA_NODE
    #define LORA_NODE1
    #define OTAA
    #define LORA_DR 1
    #ifdef LORA_DR
        #define LORA_ADR 0
    #endif
    #define LORA_BAND 868
    #define LORA_PWRIDX 1
#endif
```

Figure 5.12: Snippet of configuration file.

Command byte [hex]	Command data [hex]	Purpose	Example
11	xx xx xx xx (4 bytes)	Update timestamp, 4x bytes are the new timestamp.	”1122334455 (sets new UNIX time to: 573785173)
22	xx xx (2 bytes)	Change the transmission interval	”22003C” (set interval to every 3C=60 sec)
99	Doesn’t matter	Reset the end-node	”99”

Table 5.2: Downlink configurations available to all nodes.

Message format

After the scope change midways in this thesis, the message format designed in section 4.5.4 was simplified. As the resulting focus went into getting good measurements, aspects like automatic GPS location, angle, orientation, and timestamps were no longer prioritized. This resulted in a new uplink message format shown in figure 5.13. The downlink format remained the same.

Per the acceptance criteria, three configurations were implemented, the ability to reset, change transmission interval, and update the timestamp of the end-node. Following the design in figure 4.5, these commands (with example) were implemented as general downlink commands:

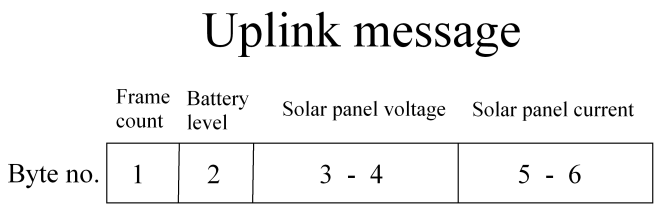


Figure 5.13: Implemented message format for uplink messages.

5.4.3 Main program

In standard operating mode, the node acts as illustrated in the state diagram in Figure 5.14, while Appendix E1 show relevant functions and files utilized by the diagram. The red arrows represent when using NB-IoT, while the black arrows represent LoRaWAN state transitions. The software is programmed to not trigger on external events (except the ”reset button”). Each state will follow its flowchart explained below and in accordance with the state diagram. Timer 2 interrupt is the counter mentioned in section 5.4.1. The default settings are to transmit minute measurements by averaging 60 samples. As mentioned earlier, this interval can be changed by command, but each sample will always be sampled every second. The states are: INIT, ACTIVE, SLEEP and NOT_JOINED, and are explained in the following sections. Each state, except for INIT, has a flowchart to visualize its flow.

The figures show the flow for both options, NB-IoT node, and LoRaWAN node by using different color coding.

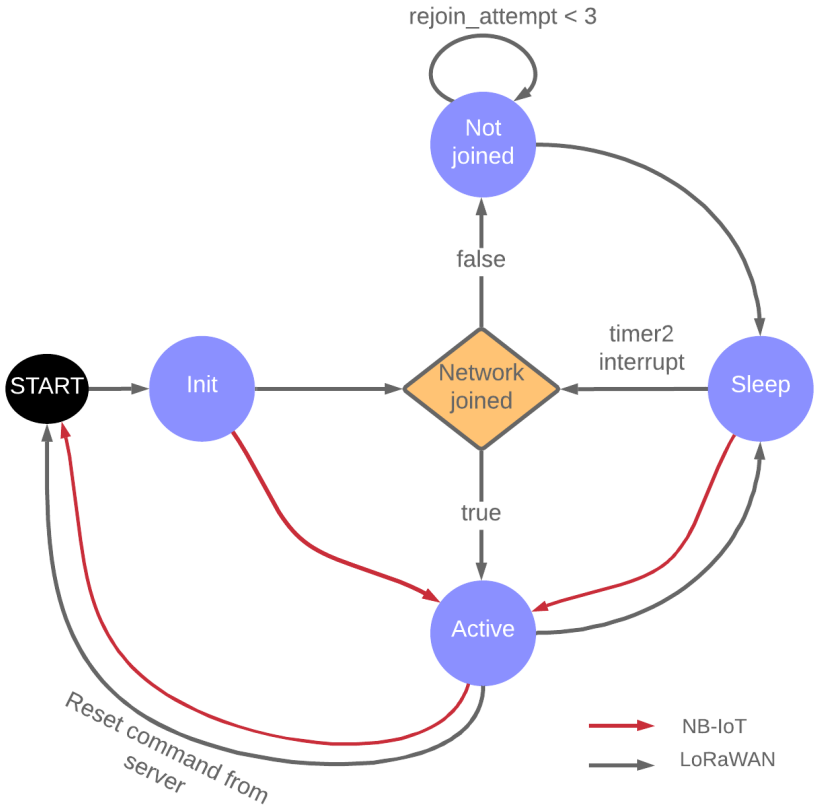


Figure 5.14: State diagram including transitions for both NB-IoT and LoRaWAN.

INIT state

The INIT state is only entered once at startup, or after reset. It initializes all the other drivers, sets up the board, and attempts to connect to the LoRaWAN (if LoRa is used).

SLEEP state

The ATmega324PB provides multiple power management modes. The one chosen for this solution is ”power-save mode”, as it provides the lowest power consumption and relevant wake-up sources [23]. This state also acts like an IDLE state in the sense that it chooses which state it enters after woken up. This logic was chosen as it’s desirable for the MCU

to be sleeping as long as possible, and because the MCU can be woken up by UART and timer interrupts. Power-save mode uses less power due to not enabling the watchdog timer, which is only enabled just before reset. The flow of the state are shown in figure 5.15, while Appendix E1 show relevant functions and files utilized by the chart. The figure shows the flow when using NB-IoT illustrated with red arrows, and black arrows for LoRaWAN. Unique with the NB-IoT flow, is that in order to transmit commands to it from the server, the modem must be able to wake it up if it's sleeping. As mentioned before, this is achieved by receiving a byte over UART. It's visualized in the figure with the use of green boxes.

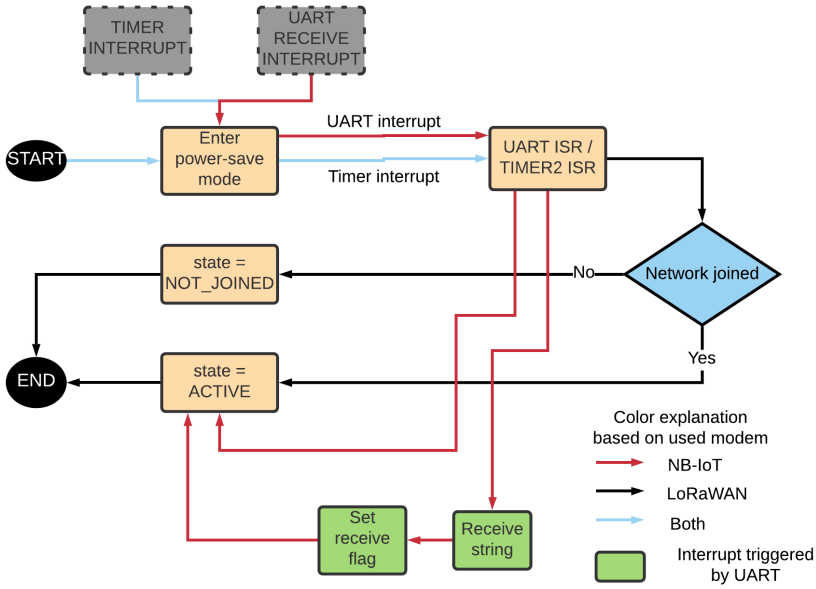


Figure 5.15: Flowchart illustrating the flow in SLEEP state.

NOT_JOINED state

This state is only used when end-node uses LoRaWAN. The reason being when NB-IoT is used, an external modem is used which follows its code flow and keeps the connection alive. The flow of the state is visualized in figure 5.16, while Appendix E1 show relevant functions and files utilized by the chart. The node is allowed to attempt to join the network a certain number of times without using all of its available air time. For this solution, the maximum amount was set to three times, and after it must wait at least 3 minutes before retrying. These values are not calculated thoroughly and just based on testing and failing with the chosen duty cycle.

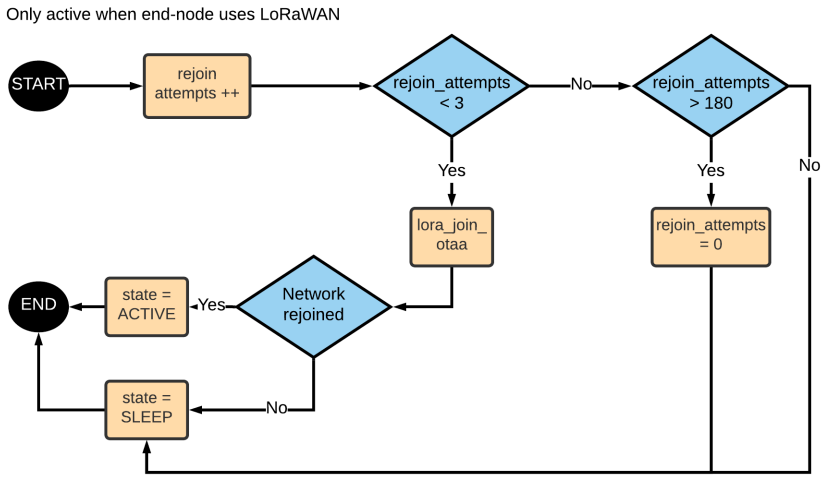


Figure 5.16: Flowchart illustrating the flow in NOT_JOINED state.

ACTIVE state

In this state is where most of the data retrieving and processing occurs. The entire desired flow is illustrated in a simplified version in figure 5.17, while Appendix E1 show relevant functions and files utilized by the chart. The deployed nodes follow this flowchart with the exception of enabling and disabling the load circuit. In their software version, the load circuit is enabled at all times due to a logic error realized too late. The new software has only been implemented in a test node, which does not provide continuous data, but used for testing and achieves desired functionality. The main idea remains the same. The node reads the voltage and current registers from INA219 and adds them to the previous samples. It then polls if the real-time counter is divisible with the chosen transmit interval, and if true it will read the battery level, and together with the frame it will encode a hex-string ready to be transmitted. Based on whether the node is using LoRa modem or NB-IoT, it will either transmit the message and wait for a potential response (LoRa), or transmit and go straight to sleep (NB-IoT).

5.4.4 nRF9160 DK software

The nRF9160 development kit uses its MCU and therefore needed its software. The code implemented was made as easy as possible due to insufficient time to comprehend the massive SDK environment used with Nordic Semiconductor devices. The script is based on their sample code ”mqtt_simple” [37], and heavily modified. Because the power consumption was disregarded, the modem has only one state, ACTIVE. The flow of the state is shown in figure 5.18, while Appendix E1 show relevant functions and files utilized by the chart. It will go in a loop maintaining the connection with MQTT broker and poll if the message has arrived. The modem sleeps between the keep-alive interval, and therefore a

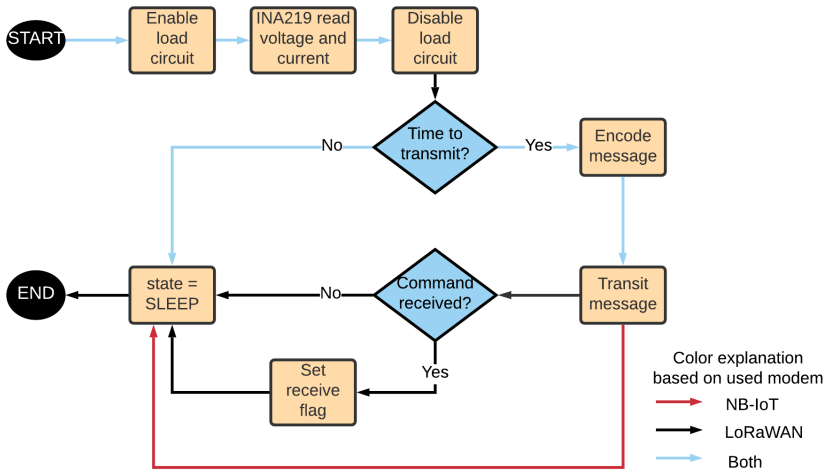


Figure 5.17: Flowchart illustrating the flow in ACTIVE state.

callback was implemented to be able to receive messages at any time. This thesis will not go into detail on how to create the needed SDK environment for programming nRF9160 DK. For unknown reasons, the modem (most likely due to software), often disconnected from the MQTT-broker, and while always trying to reconnect had a massive power consumption which depleted several power banks. Even though the node was deployed, it did not provide consistent data like the others.

5.5 Server firmware

The main tool for programming the back-end application was Visual Studio Code. This was simply a familiar and easy-to-use code editor. It was created five different scripts, the main script that runs continuously and four which can be run whenever needed. The main script is the actual server which is running continuously. The other four are scripts for synchronizing the nodes, two for creating graphs (voltage, current and power, and complete battery life), and one to calculate the packet loss-rate in each node. Only two scripts will be explained further below, the main script and the one to synchronize all nodes.

5.5.1 MQTT brokers

The two brokers used for this solution were TTN's console, and the public broker "mqtt.eclipse.org".

TTN Console

The console was briefly explained in section 3.2.1, and is used with LoRa devices. A big advantage of TTN's console, is that TTN provides a library for easy integration be-

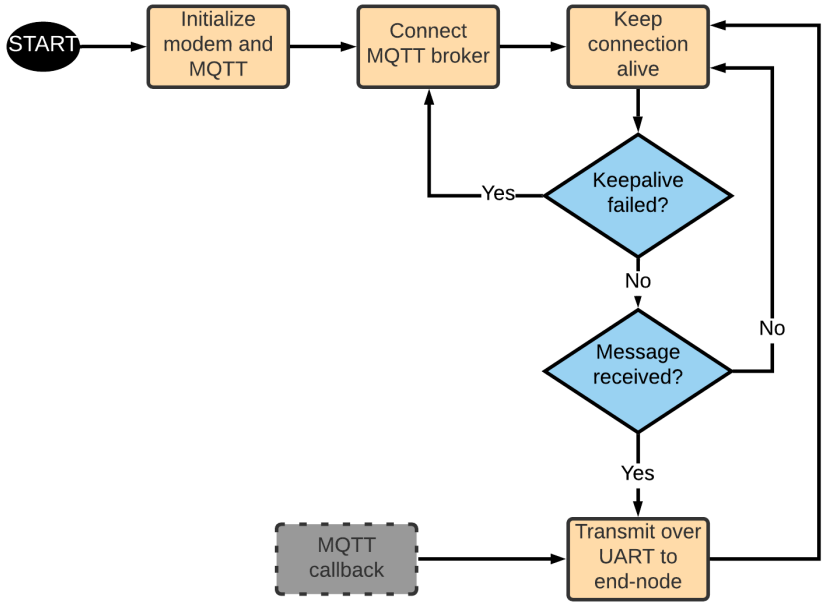


Figure 5.18: Flowchart illustrating the flow in nRF9160 NB-IoT modem.

tween server and LoRa nodes [38]. Connection with the broker is achieved by creating a ”HandlerClient” by providing the standard TTN discovery address and the access key for the application. In addition, it is also required to provide the exact name for the application, ”application ID”. The Console also acts as a GUI for configuring application and device information. Whenever a generic device is added, all EUI’s and keys are generated automatically. Snapshots of said GUI is illustrated in Appendix D.

The client then subscribes to the application server and whenever available, received uplink messages from the corresponding end-nodes. Furthermore, both the console and the client can enqueue downlinks to the end-nodes. This feature is used to configure the end-nodes in regards to transmit interval, timestamp, and resetting, and utilized in the synchronization script.

Eclipse broker

This broker is a simple public broker where the developer decides two channels in which the end-node and server subscribe to one each and publishes in the other one. Eclipse provides a library for developers to use when using standard MQTT communication, ”paho-mqtt” [39].

5.5.2 Main script

This script is continuously running on a computer and stores all messages received by the end-nodes. In order to communicate with both the LoRaWAN devices (via TTN) and NB-IoT devices (via Eclipse) concurrently, two extra threads were deemed necessary. The main program, therefore, consists of three threads, two clients over MQTT and the main thread. The flow of each thread are shown in figure 5.19, while Appendix E2 show relevant functions and files utilized by the chart. The main thread simply keeps track of the current date. The two client threads, after connecting to their broker, waits for a message from an end-node with the usage of callback functions. After receiving a message, the message is processed and data extracted. Messages are logged in files with the current date reception as the name. These files are stored in a folder named after the device ID. Each message in the files are stored with the following parameters:

- Frame
- Battery level
- Voltage [V]
- Current [mA]
- Power [mW]

where the power parameter is simply calculated by multiplying current with voltage.

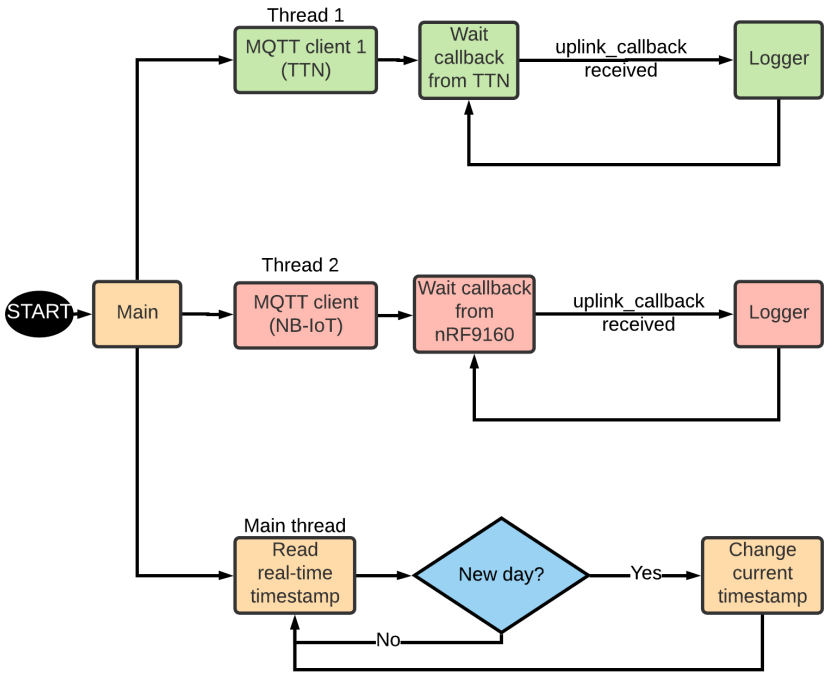


Figure 5.19: Flowchart: Main script back-end application.

5.5.3 OTA synchronization

This is a stand-alone script used to synchronize all nodes that are out of sync. The script is designed for this solution and therefore only syncs the four registered and active nodes, but is easily scalable. The general flow is illustrated in figure 5.20, while Appendix E2 show relevant and utilized functions and utilized by the chart. All nodes can be synchronized toward any desired node, but the flowchart and testing were completed with node 2's timestamp as the target. It's worth noting that this reference to node 2's timestamp, actually refers to the timestamp used by the server when storing the message. As the node doesn't send its internal timestamp, the server simply reads the current time whenever receiving a message. For easier understanding, the flow is explained in the following list and uses an error margin of up to 3 seconds:

1. Connect to TTN's MQTT broker (if LoRa nodes are active).
2. Connect to Eclipse's MQTT broker (if NB-IoT nodes are active).
3. Read the last message sent and stored by node 2.
4. Extract the timestamp and only keep the seconds part while disregarding what hour it was and how many minutes on it (Ex: 14:24:36 $\Rightarrow$ 00:00:36 $\Rightarrow$ 36).
5. Same as in 4, but with node 1.

6. Find the absolute difference between node 1 and node 2 (absolute value of (node2.seconds - node1.seconds)).

7. Check if difference is greater than 3 seconds.

8. If 7 is true, create a corrected timestamp and transmit it to node 1 (further explained later).

9. Repeat 5-8 for all remaining nodes.

10. Disconnect the MQTT brokers.

The logic behind calculating a corrected timestamp is based making all the nodes send at a specific time. As all nodes transmit a message every minute it can be assumed that within that interval (0-60 seconds), each node will have tried to transmit a message. Therefore, by synchronizing each node to the same number of seconds into each minute they transmit it is possible to have them transmit around the same time. For example, Node 2 transmitted its last message at 14:24:36, and node 1: 14:24:08, then node 1 needs to shift its internal counter by 28 seconds forward. In the opposite case, where node 2 would synchronize on node 1, node 2 would have to shift its internal counter backward by 28 seconds. This is based on the assumption that the nodes' next transmission would be at 14:25:36 and 14:25:08, respectively, and that the server stores the messages with exact timestamp upon arrival. The solution is in no way ideal, but compromising and short-term realized. It accepts an error margin as desired (default is 3 seconds), and does in no way count for general drift in LoRa transmissions.

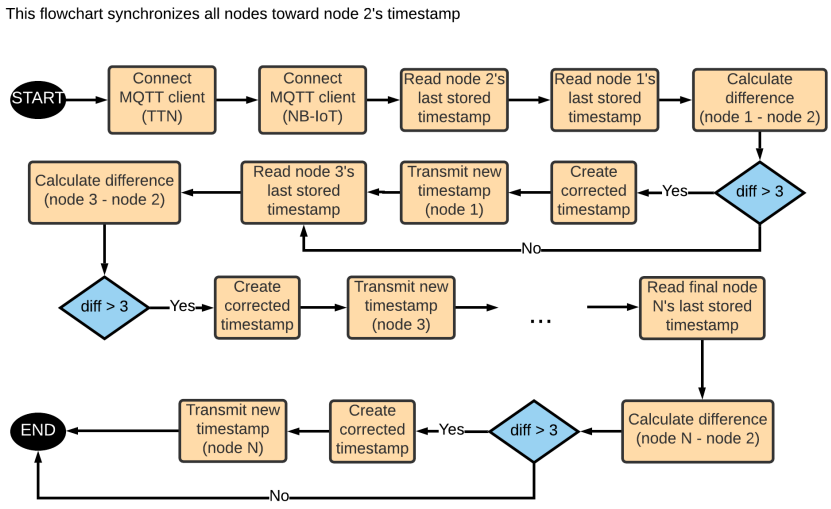


Figure 5.20: Flowchart: Synchronization script.

5.6 Placement and coverage

As 4 solar panels were ordered, 4 nodes were prepared and deployed around the city. Their placement are shown in figure 5.21. Locations were chosen based on available places with a big distance between them as possible.

All nodes except LoRa-node 1 were deployed and prepared for outdoor weather. The nodes are in no way IP67 waterproof, but with a semi-planned way of placing the node, the chance of water damage is highly reduced. An example of an outdoor installment is shown in figure 5.23. Node 1, shown in figure 5.22, are located inside in an office at the university. The boxing solution simply consists of a waterproof plastic container, with a drilled hole able to fit the solar panel cables.

Figure 5.21: Location of deployed sensors.

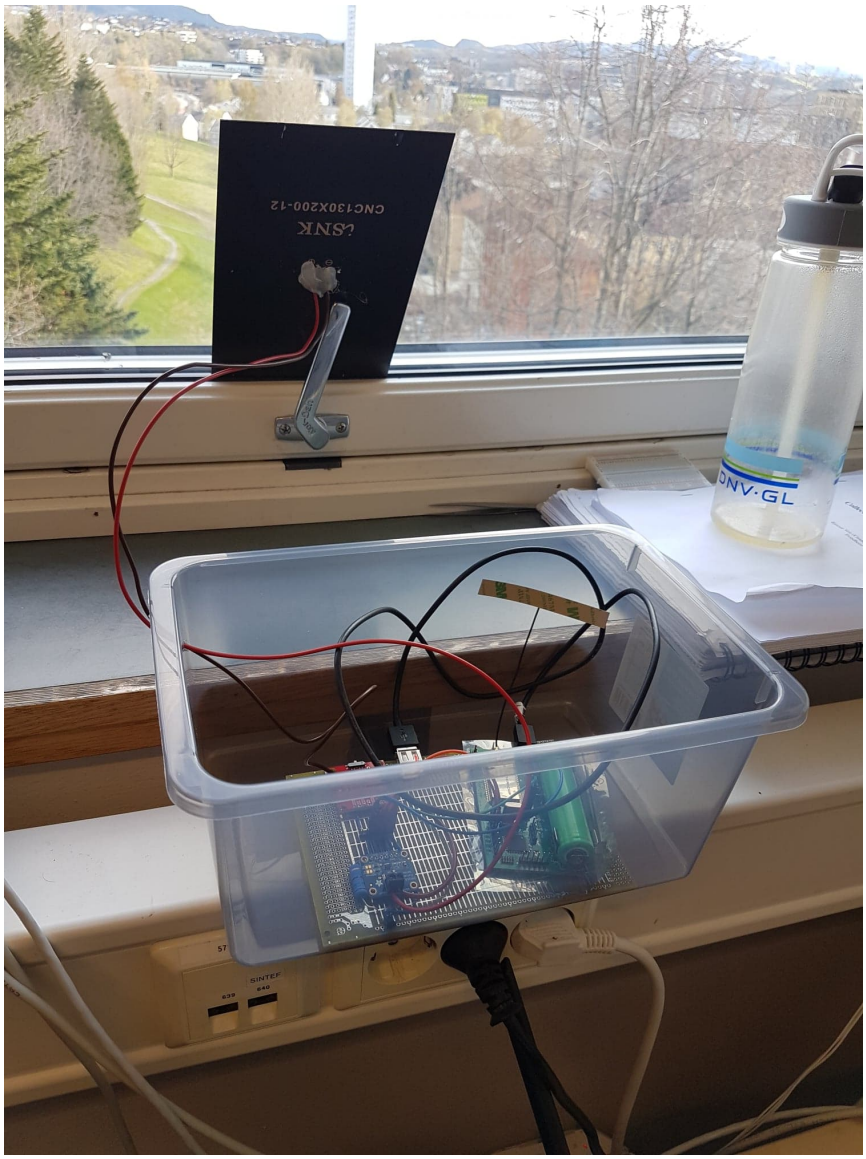


Figure 5.22: Deployed LoRa-node 1.

Figure 5.23: Deployed LoRa-node 3. Outside with no roof.

Chapter 6

Testing & Results

This chapter covers the various testing and results of the system proved and verified against the acceptance criteria covered in section 4.4. The chapter will be split into to main sections, covering the original solution and the alternative solution. Due to minimal testing on the original solution, the alternative one will be the main focus, as these devices are the ones currently deployed and measuring data. It's also worth noting that except for the hardware testing and tests for basic functionality like transmitting, receiving, etc. LTE-node 1 did not provide enough data and is therefore not present in most tests.

6.1 Original solution

The original solution, shown in figure 5.9, was in no way thoroughly tested, but instead only focused on the basic functionalities like certain hardware, debugging UART communication, and LoRa functionality.

The only components tested, were the MCU, LoRa modem, charge controller, and voltage converters. Using a multi-meter, the board is capable of converting the battery voltage into both a 3.3V out for most of the circuitry, and to 5.0V out to the USB header (J3). Providing power from a power supply into the solar panel header (P1), the two LED indicators lit up to indicate the battery was recharging and power is good. The next day, observing the battery voltage, it had increased. When trying to recharge with different voltages, the LED indicators only seem to stay lit in a specific voltage range between 4.8-5.2V. At that range, the power supply would provide a current up to 50 mA, as opposed to less than 10 mA.

Atmel Studio was able to flash code to the MCU via Atmel Ice normally.

Before testing the LoRa modem, UART communication had to be established properly. It was noticed that there was an error in the PCB design where for both active UARTs, RX was connected to RX and TX to TX. Also, no UART was connected to a header for simple debugging. A short-term solution was therefore implemented and a small wire soldered on to UART2s TX pin. From there the MCU was able to print to a terminal. Having verified the UART functionality, a slightly adapted main program used in the alternative solution

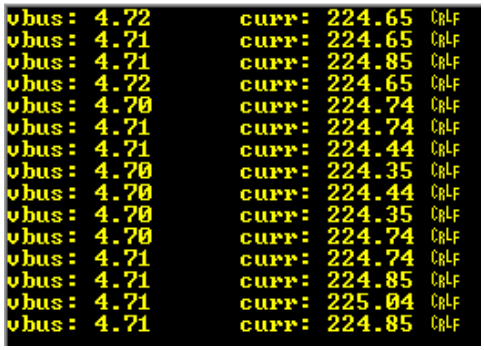


Figure 6.1: Voltage and current measured by INA219.

was executed, and the board starting transmitting normally. This led to the server logging new files. Further testing was not completed due to focusing on the alternative solution. From the design it was also noticed that the header for I2C was missing, meaning the nRF9160-DK modem had no way of communicating with the PCB.

6.2 Alternative Solution

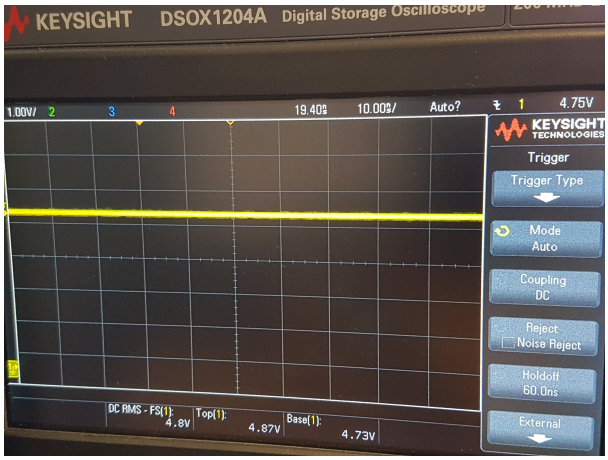
6.2.1 Hardware testing

The main PCB for this solution was a pretested board created by T. U. Rasmussen for his thesis [20], and there thoroughly tested. The hardware testing consists of confirming the breakout boards implemented in section 5.3, and the load circuit. To test the INA219 and its measurement ability a circuit based on schematic in Appendix C1 was used, but with buck-boost converter disconnected. Then a power supply was applied directly to the INA219 input with an oscilloscope connected in parallel with the power supply socket to confirm voltage. Also, a multi-meter was connected in series with the power supply to confirm current. To verify, the PCB's MCU would then read the values received from INA219 over I2C, and print values to a computer. All values are illustrated in figures 6.2 and 6.1.

Next, the buck-boost converter (U2) was connected to test charging. As soon as voltage went above 2V the circuit starting charging, this was confirmed by observing the LEDs implemented on in the charge circuit on T. U. Rasmussen's PCB. Both "CHG" LED and "PWR" LED were lit while above 2V. In addition it was verified by connecting an oscilloscope to the output of the converter, shown in figure 6.3.

6.3 Potential energy from solar panel

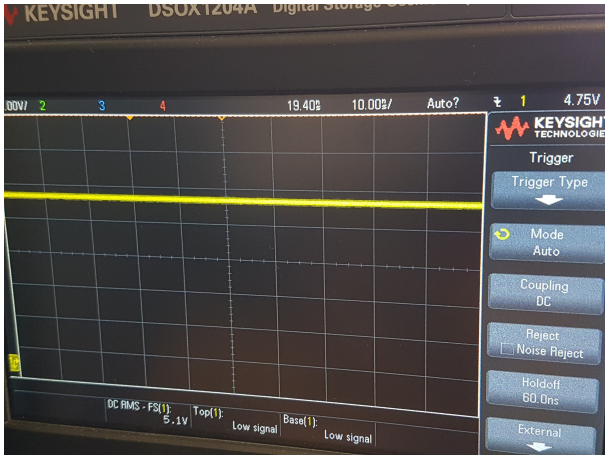
After several days deployed, the different nodes were compared by visualizing the stored data from each node. These nodes are all deployed with the circuitry shown in Appendix C1, and using the 2N7000 MOSFET transistor. The current firmware in all deployed nodes



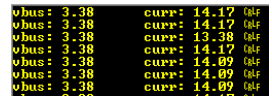
(a) Oscilloscope: Voltage [V]

(b) Multi-meter: Current [A]

Figure 6.2: Measured voltage and current from oscilloscope and multi-meter respectively.



(a) Output voltage

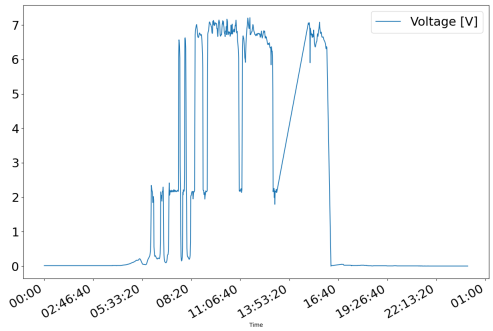


(b) Input voltage and current.

Figure 6.3: Comparing voltage input and voltage output of the buck-boost converter OCM-15208.

70	12:49:29	94	6,34	14,685	93,1029
71	12:50:29	94	6,356	14,64	93,05184
72	12:51:24	94	5,844	13,425	78,4557
73	12:52:24	94	6,368	14,76	93,99168
74	12:53:24	94	6,304	14,775	93,1416
75	12:54:24	94	6,232	14,82	92,35824
76	12:55:24	94	6,236	14,805	92,32398
77	12:56:24	94	6,112	14,805	90,48816
78	12:57:24	94	5,068	14,985	75,94398
79	12:58:24	94	2,16	13,41	28,9656
80	12:59:24	94	2,248	11,025	24,7842
81	13:00:24	94	2,212	10,11	22,36332
82	13:01:24	94	2,256	10,515	23,72184
83	13:02:24	94	1,992	12,075	24,0534
84	13:03:24	94	2,188	14,055	30,75234
85	13:04:14	94	1,796	11,58	20,79768
86	13:05:14	94	2,172	14,145	30,72294
87	13:06:14	94	2,148	14,67	31,51116
88	13:07:14	94	2,168	14,82	32,12976
89	13:08:14	94	2,192	14,37	31,49904
90	13:09:15	94	2,232	14,385	32,10732
91	13:10:15	94	2,136	13,29	28,38744
92	13:11:15	94	2,164	15,66	33,88824
199	14:58:11	94	7,06	14,475	102,1935
200	14:59:11	94	7,024	14,475	101,6724
201	15:00:11	94	7,008	14,505	101,651
202	15:01:11	94	7,064	14,505	102,4633

(a) Snippet from a LoRa-node 1 log, columns explained left to right: Frame, time, battery, voltage, current, power.



(b) Graph showing voltage.

Figure 6.4: Irregular behavior illustrated with a snippet from a log file and graph from the 13th of May 2020.

provide a ”always-on state” for the transistors. The V_{Gate_Source} provided by the PCB to switch on the transistor was out of spec. According to its datasheet, the transistor requires a voltage up to 5V in order to completely switch on the transistor with a current up to 350 mA. This led to multiple nodes transmitting weird results and measurements. A simple illustration of the affected result can be seen in figure 6.4. Figure 6.4a show a voltage changing from 6V to 2V and then up to 7V (fourth column) instantly, instead of gradually. The fifth column represents current which barely changes in the same timespan. This behavior was not consistent with all nodes on all days but happened frequently.

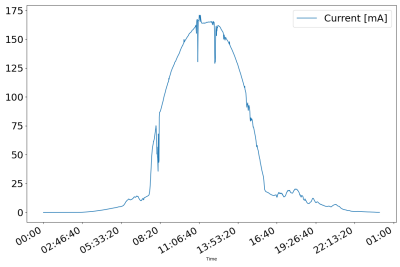
Figure 6.5 illustrates a sunny day without any clouds. There, the voltage does not vary much after 08:00, while the current rises and decreases rapidly. The Node loses the sun at around 17:00.

Figure 6.6 show four different dates where it was cloudy for the biggest part of the day.

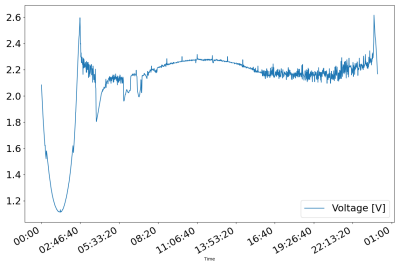
One of the 4 nodes was after a while retrieved back to the office for testing with different transistors and logic. A new test was based on only turning the transistor on at the moment of measurement. For accurate results, finding the time delay between turning the transistor on and the current stabilizing was important. The test consists of replacing the BS170 MOSFET transistor with a 2N6178 bipolar NPN transistor and measuring the time delay for the voltage to stabilize. Results are shown in figure 6.7.

Afterward, the same test was completed with the 220-ohm resistor instead, this achieved the result illustrated in figure 6.8. Conclusion from both tests are shown in table 6.1.

This is an extremely low time delay, and simple to implement in a new software driver. To determine the new resistance across that path, different light intensities were applied by concealing parts of the solar panel, producing a current and voltage corresponding to given light intensity. The results are shown in table 6.2. Using the voltage and current

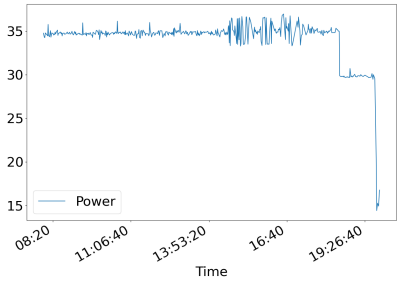


(a) Current

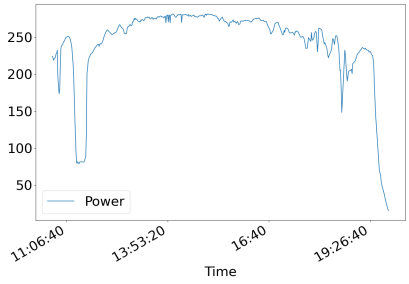


(b) Voltage

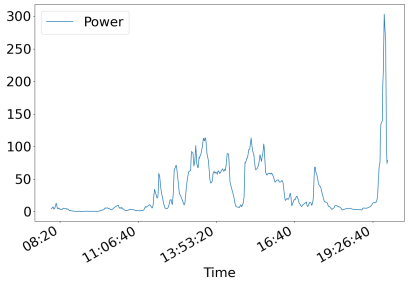
Figure 6.5: Measurements from LoRa-node 1 on 14th of June during a sunny day without skies of LoRa-node 1.



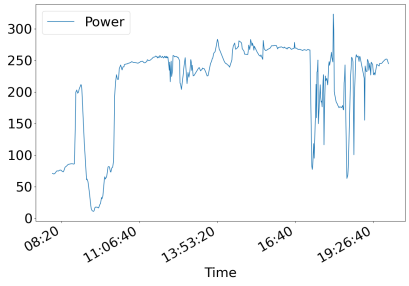
(a) 26.05.20



(b) 02.06.20



(c) 07.06.20



(d) 10.06.20

Figure 6.6: Power [mW] production in different scales of cloudiness from LoRa-node 2

	Delay	**Voltage drop across transistor when switch ON**
With 470Ω resistor	$12\mu s$	1.62V
With 220Ω resistor	80 ns	990 mV

Table 6.1: Time delay test results for 2N6178 transistor.

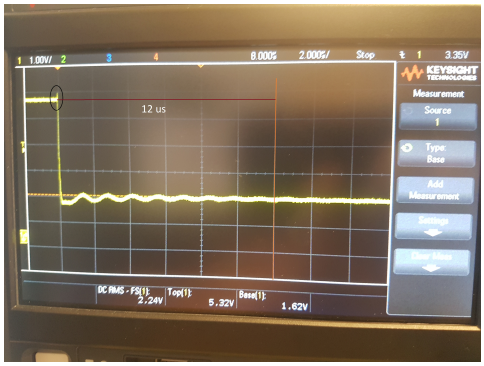


Figure 6.7: Time delay after switching on 2n6178 NPN transistor with a 470Ω resitor.

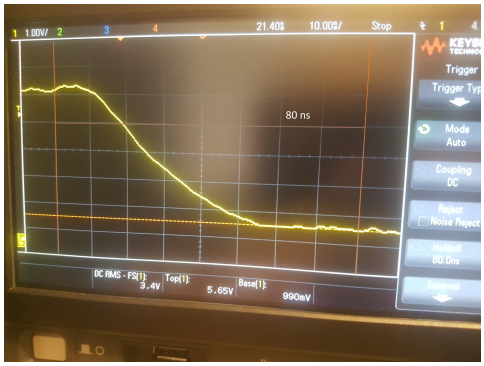


Figure 6.8: Time delay after switching on 2n6178 NPN transistor with a 220Ω resistor.

measurements from INA219, which are already proved to be precise, total resistance was calculated using Ohm's law. Weather conditions when testing were clear skies.

6.4 Node synchronization

From the implementation mentioned in section 5.5.3, a simple algorithm was developed to synchronize the nodes with over-the-air messages. Running said script led to a message sequence shown in figure 6.9. The test was completed on three different nodes, where it was decided to synchronize against node 2's timestamp. The figure is partly edited, where the sequence of actually receiving the messages where removed. The result shows three nodes sending a message three different times, 18:29.39, 18:29:42 and 18:29:44, node 1, my_board, and node 2 respectively. Timestamp to synchronize on was node 2's 18:29:44. The next message sent after receiving synchronizing command were: 18:31:45, 18:31:46, and 18:31:44. Running this script provided a lot of variable results. Figure 6.9 is a result where the synchronization worked. The script was run by transmitting both confirmed and unconfirmed commands (explained in section 3.1.2) over LoRaWAN. Using unconfirmed

Voltage	Current [mA]	Total resistance [Ω]
0.48	25.2	19.05
1.32	74.4	17.74
2.5	138	18.12
2.65	154	17.21
4.4	214	20.56
6.66	284	23.45
7.33	304.8	24.05

Table 6.2: Total resistance in circuitry with different sunlight exposure.

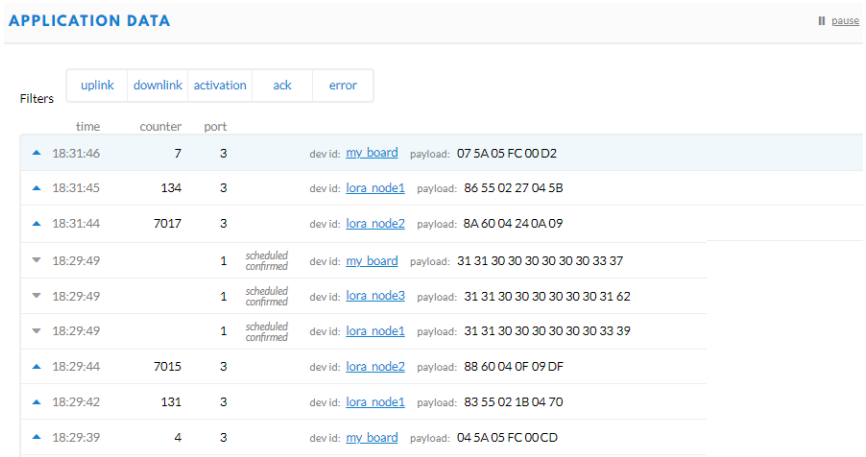


Figure 6.9: Message sequence synchronizing nodes.

messages resulted in multiple occasions where the command was simply not received, and with no quality of service (QoS) in LoRa, it was not automatically resent. Transmitting with confirmed messages resulted in multiple retries even though the messages usually were received.

It is also worth noting that the nodes drift with their transmission timestamp. Even though the node, in theory, should transmit every 60 seconds (default), due to internal factors of the ATmega324PB and external crystal, it will drift over time. Figure 6.10 illustrates this. The figure shows a very truncated message pool to only provide the necessary information. Over 42 minutes, the node has drifted 2 seconds.

6.5 Self-sufficiency and low power consumption

Testing self-sufficiency and low power consumption were deemed difficult due to the circumstances. With the lack of time and available equipment, standard measurements of the power consumption of the board in different states were not completed. Instead, the main

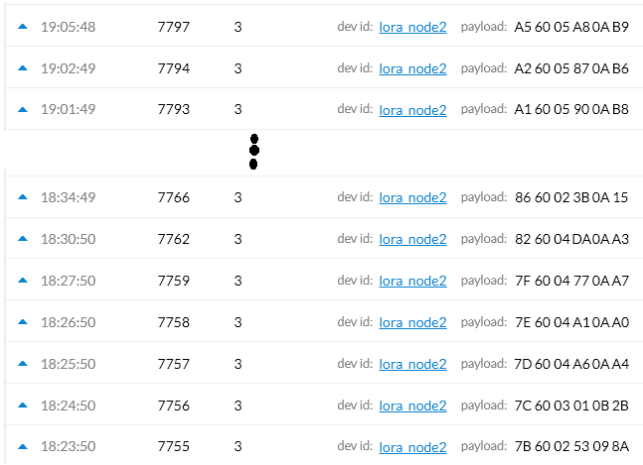


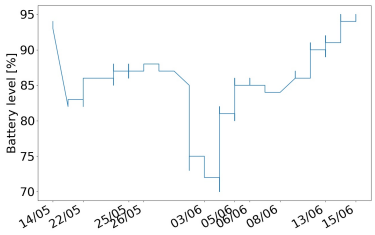
Figure 6.10: Illustrates the internal time drift in a node.

test for self-sufficiency was done in accordance with acceptance criteria 3 in section 4.4. In order to look at the behavior of the nodes, a graph was created which plots the battery level based on data collected between 14.05.2020 and 15.06.2020 for the three active LoRa based nodes. LoRa-node 3 stopped sending data on 10.06.2020 because it was recalled due to the thesis reaching its end. The placement of the remaining two nodes prevents the need for being recalled. The result can be seen in figure 6.11. Figure 6.12 illustrate examples of graphs during a couple of nights (00:00-06:00). The biggest battery level drop during a night was 2 %.

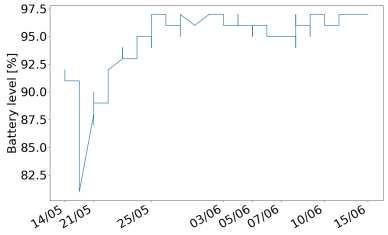
Further, to provide an indication whether a node would be capable of surviving longer periods of time with less to no sun, a test to understand the current required for a almost depleted battery and a semi-fully charged battery were conducted. First, a 18650 battery were depleted until around 3.0V and the PCB then connected to a power supply which could provide however much power the PCB needed. Afterwards, the same test was conducted with a 3.62V semi-fully charged 18650 battery. Results are shown in table 6.3.

6.6 Coverage and placement

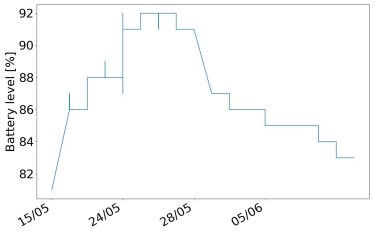
An important acceptance criteria was the criteria about placement and coverage. In order to test this, and retrieve measurements for analysis, 4 nodes were initially deployed in four different places, shown in figure 5.21. Over four weeks, the nodes remained outside (except LoRa-node 1), during which it on some days rained. The nodes took no damage and LoRa-node 3 was recalled on the 10th of June while approaching the end of thesis testing. In regards to coverage and acceptance criteria 5.5 in section 4.4, a simple script was programmed to calculate lost packages and at what percentage it was able to receive. The total average percentage for each node are listed in table 6.4.



(a) LoRa node 1

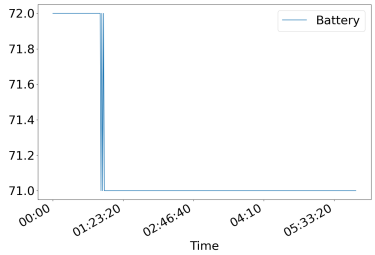


(b) Lora node 2 (until 10th of June)

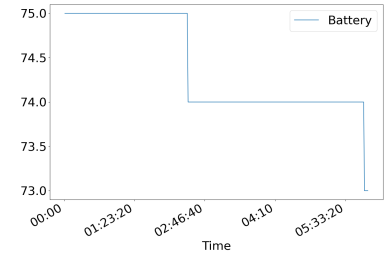


(c) Lora node 3

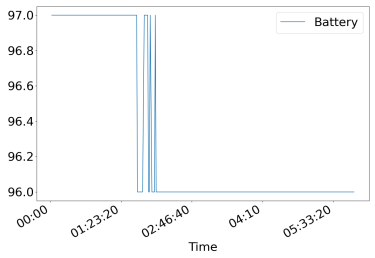
Figure 6.11: Battery level changes over a time span of 1 month from 14th of May until 15th of June.



(a) LoRa node 1 - 06.04.2020



(b) Lora node 1 - 06.03.2020



(c) Lora node 2 - 13.05.2020

Figure 6.12: Battery graphs over a time span between 00:00 - 06:00 on different dates.

	$V_{bat} = 3.0V$		$V_{bat} = 3.62V$	
V_{supply}	**Min. current drawn [mA]**	**Power[mW]**	**Min. current drawn [mA]**	**Power [mW]**
2.63	16.9	44.45	16.7	43.92
3.68	10.7	39.38	10.98	40.41
4.50	9.90	44.55	9.3	41.85
5.46	7.18	39.20	7.1	38.77
7.52	6.1	45.87	5.8	43.62
8.95	4.8	42.96	4.9	43.86
10.62	4.1	43.54	4.3	45.67
12.77	3.6	45.97	3.7	47.25
13.96	3.0	41.91	2.9	40.51

Table 6.3: Current drawn to charge circuit with almost depleted and semi-fully charged 18650 battery.

LoRa-node 1	LoRa-node 2	LoRa-node 3	LTE-node
97.2 %	71.7 %	50.9 %	74.2 %

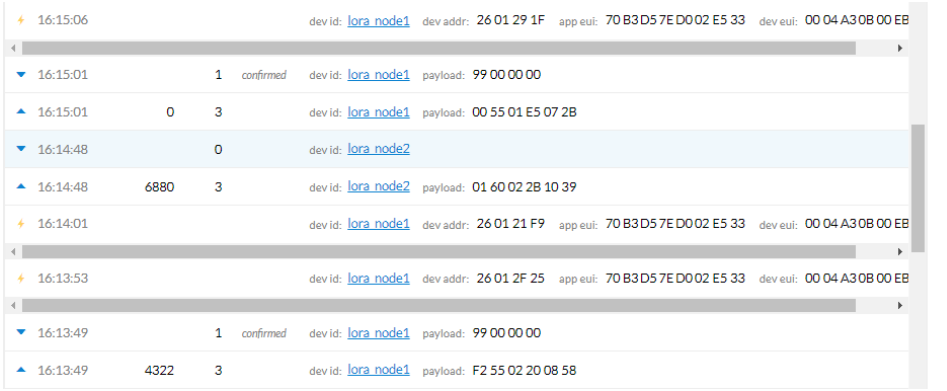
Table 6.4: Average percentage of packets received.

6.7 Remote configuration of end-nodes

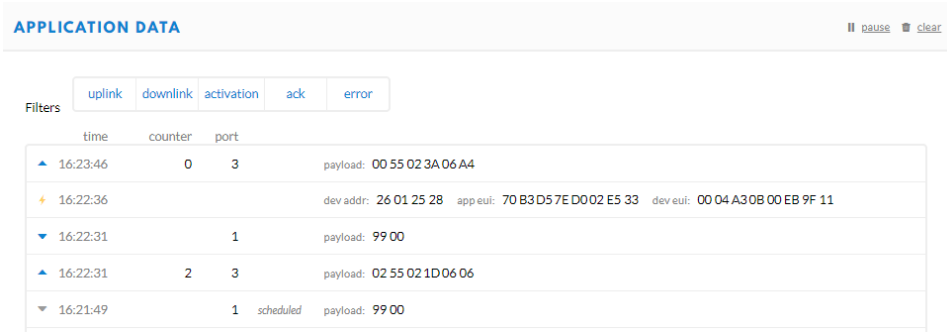
From the acceptance criteria, three types of configuration were required. These were: reset end-node, change transmission interval, and update internal timestamp. Updating internal timestamp was already confirmed by section 6.4, which showed the synchronizing. The remaining two configurations were simply tested by transmitting the appropriate command (tab. 5.2) from TTN's console and observing the response. It should also be mentioned that similar behavior as with node synchronization when using unconfirmed and confirmed occurred.

The general behavior is shown in figure 6.13a, where a confirmed reset command was sent two times and the device reset both times. The figure shows LoRa-node 1 sending and uplink and then immediately receiving a downlink "99 00 00 00". 4 seconds later the console received join requests from the node. 1 minute after joining the node transmits a new uplink, and then received a new reset command and rejoins. This behavior happened on multiple occasions and was only solved using unconfirmed messages, shown in figure 6.13b. The downside to using unconfirmed messages is the QoS, where it cannot be guaranteed the message is delivered. Disregarding the fact that not necessarily all commands are received, the end-node executed the reset command every time it arrived. This was verified by connecting the node to a computer over UART for print functionality and reading the terminal.

To test the ability to change transmission interval, a command requesting a 10-second interval was sent first, and then after a short while, a new request for the original 60-second interval was transmitted. This is shown in figure 6.14. The figure shows that after sending the downlink "22000A", it starts transmitting rapidly, every 10 (0A) seconds. Then, after



(a) Confirmed reset command



(b) Unconfirmed reset command

Figure 6.13: Snippet from TTN's Console after a reset command to LoRa-node 1.

receiving the second change request "22003C", the node starts transmitting every 60 (3C) seconds.

6.7.1 Time-lag between nodes' delivery of energy

As mentioned in the thesis description as well as the introduction, an important aspect of this thesis was the prediction of Delivery of Energy (DoE). Even though the main focus had been shifted to achieving a functional distributed sensor network, measurements were gathered, and in this section compared. To achieve a big as possible indication of time-lag, LoRa-node 1 and 3 were compared with LoRa-node 2, as their distance is far greater than between node 1 and 3. For each sample, most of the graph will be shown with a time span from 11-17, and then an enhanced version for each graph, illustrating the potential time-lag. Results are shown below:

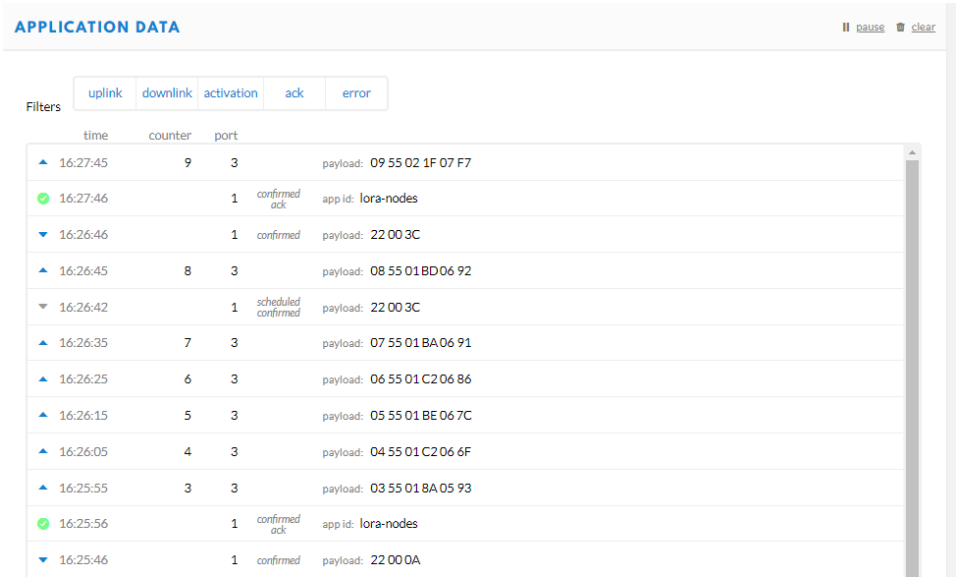
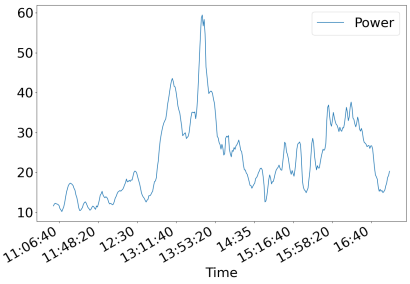
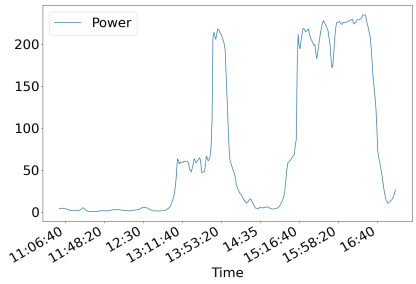


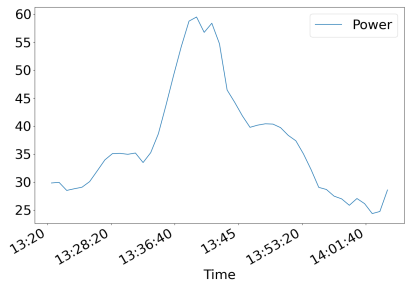
Figure 6.14: Snippet from TTN's Console after command to change transmission interval.



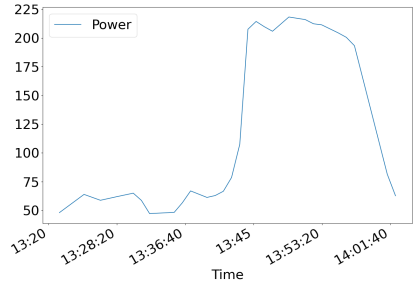
(a) LoRa-node 1: 1100 - 1700



(b) LoRa-node 2: 1100 - 1700

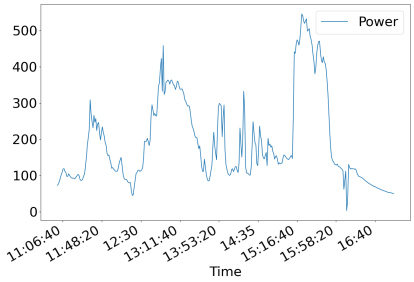


(c) LoRa-node 1: 1300 - 1400

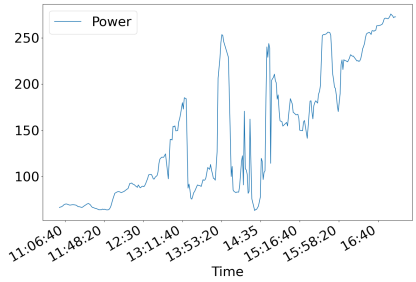


(d) LoRa-node 2: 1300 - 1400

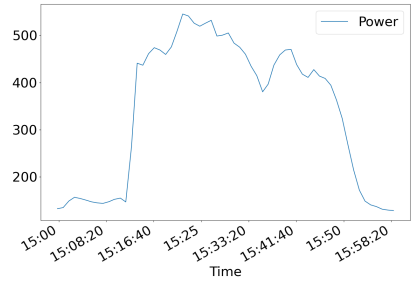
Figure 6.15: Illustration of potential time-lag between Lora node 1 and 2 on 03.06.2020 using graphs from produced power [mW].



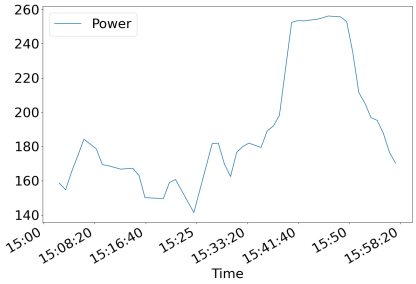
(a) LoRa-node 1: 1100 - 1700



(b) LoRa-node 2: 1100 - 1700

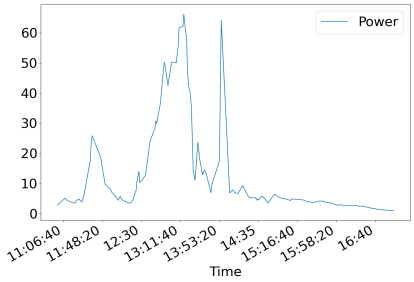


(c) LoRa-node 1: 1400 - 1500

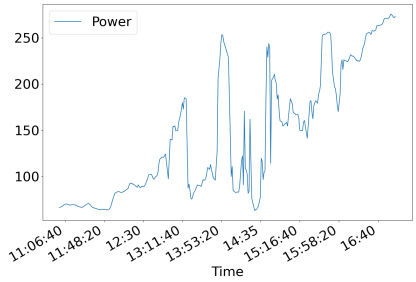


(d) LoRa-node 2: 1400 - 1500

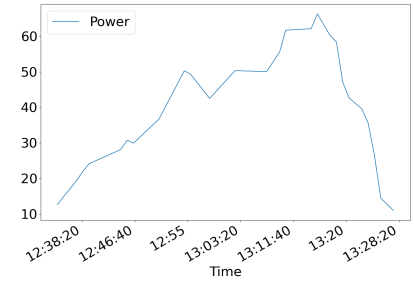
Figure 6.16: Illustration of potential time-lag between Lora node 1 and 2 on 04.06.2020 using graphs from produced power [mW].



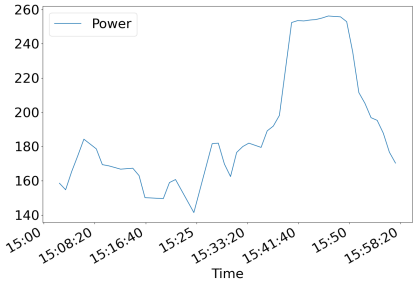
(a) LoRa-node 3: 1100 - 1700



(b) LoRa-node 2: 1100 - 1700

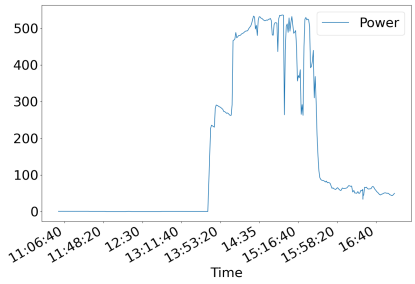


(c) LoRa-node 3: 1200 - 1300

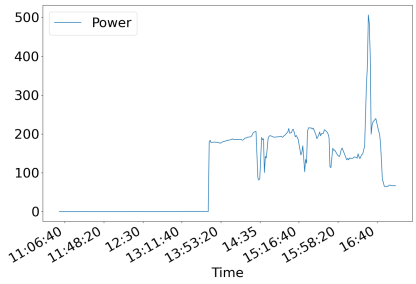


(d) LoRa-node 2: 1200 - 1300

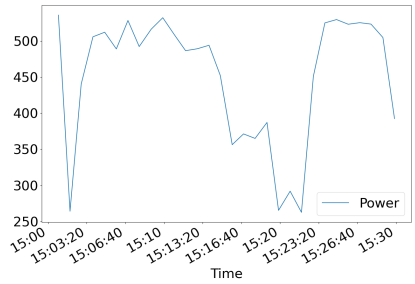
Figure 6.17: Illustration of potential time-lag between Lora node 3 and 2 on 04.06.2020 using graphs from produced power [mW].



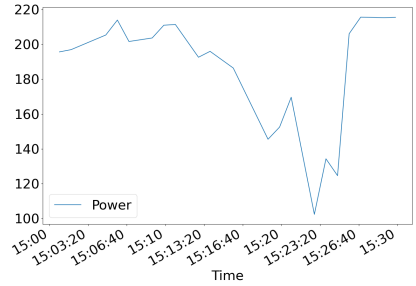
(a) LoRa-node 1: 1100 - 1700



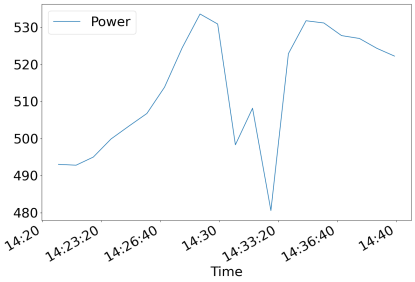
(b) LoRa-node 2: 1100 - 1700



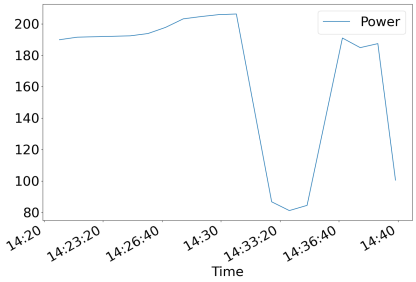
(c) LoRa-node 1: 1500 - 1530



(d) LoRa-node 2: 1500 - 1530

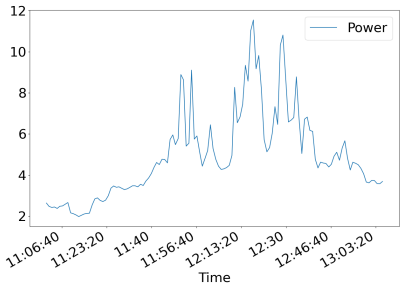


(e) LoRa-node 1: 1420 - 1440

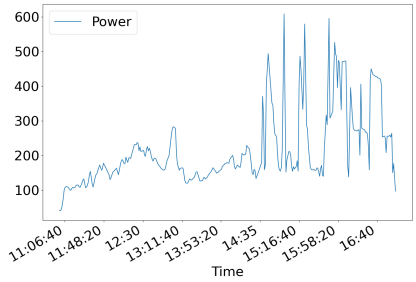


(f) LoRa-node 2: 1420 - 1440

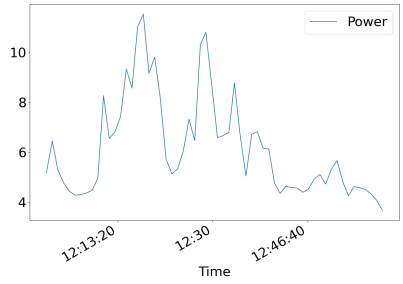
Figure 6.18: Illustration of potential time-lag between Lora node 1 and 2 on 05.06.2020 using graphs from produced power [mW].



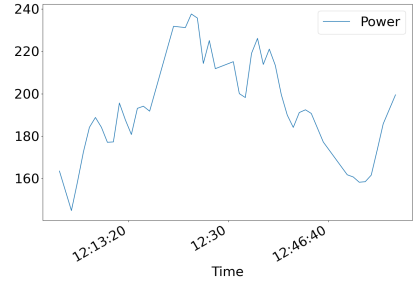
(a) LoRa-node 1: 1100 - 1700



(b) LoRa-node 2: 1100 - 1700

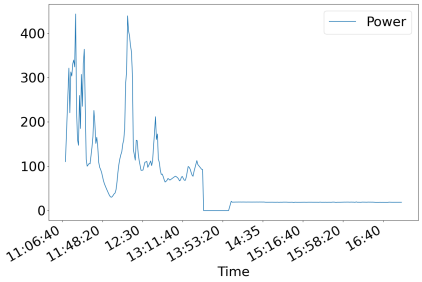


(c) LoRa-node 1: 1200 - 1300

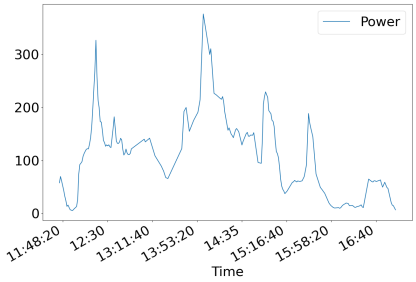


(d) LoRa-node 2: 1200 - 1300

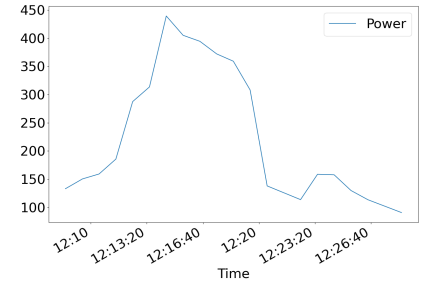
Figure 6.19: Illustration of potential time-lag between Lora node 1 and 2 on 08.06.2020 using graphs from produced power [mW].



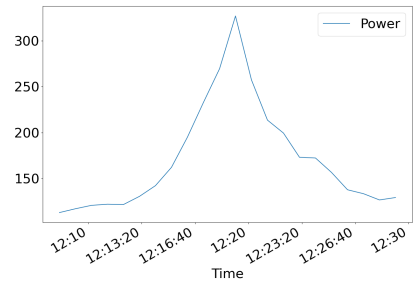
(a) LoRa-node 1: 1100 - 1700



(b) LoRa-node 2: 1100 - 1700



(c) LoRa-node 1: 1400 - 1500



(d) LoRa-node 2: 1400 - 1500

Figure 6.20: Illustration of potential time-lag between Lora node 1 and 2 on 28.05.2020 using graphs from produced power [mW].

Chapter 7

Discussion

This chapter provides a discussion meant to cover the overall system results in accordance with the acceptance criteria from section 4.4. Furthermore, the results regarding coverage and placement, node synchronization, and self-sufficiency are more deeply analyzed in their respective sections.

7.1 Single end-node system results for original solution

Based on the very short amount of testing completed in section 6.1, only AC9, and AC2 were passed. The device is capable of being recharged by the solar panel, but only at a specific range of voltages. This suggests there might be a wrong resistor connected to one of the pins of the controller. From the implementation, multiple resistors were calculated based on different desired behavior, which could explain the problem. To solve this, more thorough testing is required. The server received and stored information from the PCB, but this information is just dummy data, and not measured in any way.

To deploy the PCB similarly to the alternative solution, certain functionalities need to be tested. Features such as the MAX4484 to measure current was not tested, preventing the board from passing AC1. In addition, for a fully functional solution as intended, drivers for three components are required. Those are for the flash memory, GNSS and IMU. These features have proven to be desired based on testing and discussion done on the alternative solution, explained further below.

In conclusion, the PCB works to some extent, and need more thorough testing before any components and parts of the circuitry can be deemed faulty (except for UART connections).

7.2 Single end-node system results for alternative solution

Looking at figures 6.4, 6.5, and 6.7 [FIND REF], the devices are capable of measuring current and voltage from the solar panel and doing basic board diagnostics by measuring

the battery level. Criteria AC1 and AC2 are therefore met. From AC3, the node should be self-sufficient and the battery must not lose overall percentage over three weeks. Verdict for this criteria are further discussed in section 7.5. Comparing table 6.3 and figure 6.5, the load circuit works properly as a significantly higher current is measured as opposed to what the PCB itself can draw to charge. Nevertheless, the logic used for the deployed boards turns the transistor on at all times and not only during measurements. This was fixed and tested with a new transistor, shown in figures 6.7 and6.8, where the voltage drops when turning it on, proving current goes through the transistor. Using the 470 ohm resistor, a greater voltage drop occurs opposed to the 220 ohm resistor. Also, the time required to stabilize the current were measured as low as 80 ns. This delay is easily implemented in the next firmware update. Table 6.2 show the total resistance in the circuit when U2 (buck-boost converter) is disconnected, and all curent flow through the load circuit. In all different scale most likely to occur on the outside, the transistor shows a reliable voltage drop, proving that it's more suited than the BS170 MOSFET transistor. The new transistor was never tested over time in the sunlight, but all facts indicate that AC4 is met.

AC7 involves the remote configuring of single or multiple nodes. Each node is capable of executing a received command, explained in section 6.7. Still, each node does not receive all packets, and neither can it be guaranteed, which suggests that AC7 was not passed. This is also connected with the placement and coverage, which is discussed more in section 7.3.

From section 6.6 everything indicates that the nodes are capable of surviving the outdoors in rain. The boxing solution should be improved, but for this purpose and available data, AC11 is passed.

AC12 and Ac12, saying a node must be able to be recharged using micro-USB and support I2C, UART, and SPI, has already been proven by T. U. Rasmussen in his thesis [20].

7.3 Coverage and placement

7.3.1 Placement

AC3, AC4, AC5, AC5.5, AC7, and AC6 are all criteria that fall under the category "coverage and placement", meaning that both coverage and placement directly factor into the results affecting these criteria. While AC3 and AC4 are better covered in section 6.5, it's worth noting that how nodes are oriented have a direct effect on the criteria. Mentioned in section 4.5.1, the original plan was to automatically detect and calculate the angle and orientation of each node. This was scrapped in the alternative solution due to the circumstances and the prioritizing necessary to finish the thesis. Nevertheless, as the angle and orientation differ from the deployed end-nodes, so does the results. For instance, LoRa-node 1 is placed on the inside behind a window, which limits the amount of sunlight its able to receive to the size of the windows. The windows add a limiting layer between the panel and sun, also reducing the amount of light able to hit it. LoRa-node 3 is placed outside but attached to the corner of two walls vertically. An optimal angle is for the solar rays to hit the panel perpendicularly, which never occurs in this node. Also, the walls limit the time-window of receiving light. Optimal angle and orientation would probably provide

more accurate results and graphs, especially looking at the correlation between nodes in regards to time-delay between clouds.

7.3.2 Coverage

Looking at the logged files and graphs, figure 6.4 for instance, and especially table 6.4, it is clear that LoRa-node 2 and 3 do not have well enough coverage to provide good accurate data. The criteria set in this thesis was to have a packet loss of maximum 10%, and was only met by LoRa-node 1. This simply suggests that LoRa-node 1 has considerably better coverage than the other two have. Having to depend on other people's gateways could lead to varying results and increased packet loss. The frame counter for each message only goes up to 255, meaning that if a node would go offline for 4 hours and 15 minutes (255 minutes), the script calculating the packet loss % would not notice. This all suggests that better and more consistent data are necessary to pass AC5.5, and perhaps self-owned LoRaWAN gateways could provide better and more consistent coverage. For LTE-node1, the coverage was not the main issue, but the software used in the nRF9160DK. Updating the software to a more robust and better code would most likely solve all NB-IoT coverage issues. Nevertheless, each node was able to transmit measurement data and diagnostics through their respective protocol, which indicates that AC5 is passed.

7.3.3 Remote configurations

Sending a command to an end-node, either from TTN's console or from a simple script, was proven to be quite easy. However, with the lack of QoS, not being able to guarantee its arrival can seem to be a problem. In the implementation and testing, it was deemed most natural to use "unconfirmed" messages, but this completely removes the little QoS and that was available in LoRawan, but saves power and does not produce and weird behavior. Sending confirmed message led to on occasion multiple commands sent, one for each new message received by the server. The importance of the commands is not grave, such that if a device was reset a couple of minutes after initially sent, the consequences are not major. In addition, it can also be argued that if the device is reset multiple times, the main difference would be the frame of the next message not changing. Change the sample time of a node, on the other hand, could lead to more noticeable differences in graphs between graphs with different sampling intervals. At the same time, sending a command to change interval multiple times does not change anything, as the sample interval variable will just be set to the same as it already is. Based on how the time synchronizing script works, repeated messages to update the internal timestamp would work the opposite as intended. If a node received a command to update its timestamp for transmitting 15 seconds later than usual, if the same command were sent 1 minute later as well, the node would be desynced by 15 seconds.

Based on these results, the following table illustrates which commands and whether to transmit confirmed or unconfirmed:

Command	Confirmed	Unconfirmed
Reset	x	x
Update timestamp		x
Change transmit interval	x	

Table 7.1: Which type each command should use.

7.4 Node synchronization

The success rate for synchronizing nodes depend on multiple factors:

- Airtime
- Internal clock drift
- Coverage

Based on the explanation of the script used to synchronize the nodes and the results it provided, there are several pitfalls for the script to fail. This was to some extent known in advance but not taken into consideration as an okay solution in time was preferred over a not complete one. For starters, the nodes do not provide an internal timestamp in their message to provide any information on what time their messages were sent. Instead, the server reads the time from a global server as soon as a new message arrives. Using only this information disregards the air-time of any message, which varies. In addition, the internal drift for each node further explained in T. U. Rasmussen's thesis [20], affects the arrival time of the message. This factor was not taken into consideration as it was estimated that all nodes would have a very similar internal drift, due to factors explained by Microchip [40]. Time was therefore not spent on software and testing for compensating this error.

Based on results in sections 6.4 and 6.7, table 7.1 was constructed and suggests that "update timestamp command" only should be used as unconfirmed. Assuming a loss rate of 51%, like LoRa node 3 in table 6.4, the script would be required to run twice as many times as necessary. This does not pose as much of a problem and can simply be automated within the main server script. The result is that on a single execution, the script would perhaps only synchronize some of the nodes, and the rest on the time the script was executed. In regards to AC6, although the quantity of data is not optimal and the algorithm is flawed, with a decent error margin and option to be executed multiple times, it is passed.

7.5 Self-sufficiency

In order to understand life expectancy and self-sufficiency, a test to measure drawn current over time and in different situations is optimal. Since this type of equipment wasn't available, different estimation based on other data and results will be used to indicate whether a node is self-sufficient or not. It's important to know, that a logic error was found calculating the percentage of the battery. The calculations assume a voltage range for the battery to be 0-4.2V, instead of the actual range of 2.5-4.2V. Due to this fact, the variations in the battery-related graphs will be significantly lower than real, and therefore provide less

	Sunrise	Sunset	Daylight
01.01.20	09:59	14:43	4:44
01.06.20	03:24	23:09	19:44

Table 7.2: Sunset and sunrise times for Trondheim [41]

information than desired. With regards to the battery graphs, a battery will be considered depleted at 60%.

For starters, all data was gathered during summer, and because winter is a completely different environment for a solar panel, other methods were required to estimate winter behavior. Starting with summer, looking at figure 6.11, it's quite apparent that the three LoRa nodes passed AC3. The big battery level drop on the 3rd of June form LoRa-node 1 was due to a battery change. Now, in order to estimate winter behavior, it's necessary to know how much power a node requires for optimal charging and how much it's capable of producing on a cloudy day. Data on how much current is required to optimally charge the node are shown in table 6.3. From this data, it appears that the charge circuit is capable of charging the battery with a power of right under 50 mW. Figure 6.6 show the power production of four different days with consistent low current production. From all of the data, the lowest average power produced is fig. 6.6a. This graph shows an average of 35 mW and represents the most cloudy day available. Comparing 6.6a with the other 3 graphs, it would seem plausible for a generic winters day to produce enough power to charge a battery based on power alone. It's also worth noting that all measurements are achieved with an always-on load circuit, which is able to produce a higher power than what the charge circuit could. Therefore, the nodes must change the measuring logic to only activate the load circuit that millisecond a measurement is taken. The other problem with a generic winter's day is that it's several hours shorter than in the summer, considering sunrise to sunset. A website [41], have estimated the sunrise and sunset for each day throughout the year. Choosing the 1st of January and 1st of June as a winter and summer day, their sunset and sunrise can be seen in table 7.2. Based on this date, the winter day daylight windows are up to 15 hours shorter than summer, which could prove to be a problem for the nodes.

Overall, it's not enough consistent data by multiple nodes to make a good estimation on whether a node would survive winter. Most importantly, the logic to only activate load-circuit upon measuring would greatly improve the correlation between battery behavior and power produced. Per now, it can be assumed a lot of the power is directed through the load circuit instead of to the battery, preventing the PCB for charging optimally. Based on all these results, it does not seem plausible that a node would be able to survive winter, ergo AC3 is not passed. Improvements for better performance could be to create a better transmission strategy of when to measure, preventing any transmissions during the night for instance. Also, it's highly recommended to do a proper measurement of actual current consumption.

7.6 Time-lag between nodes' delivery of energy

Section 6.7.1 explains the testing/comparing of data from different nodes in order to detect a possible time-lag between them. As explained in earlier sections, the data pool gathered between the nodes are not ideal, and consist of several problems. Therefore the most optimistic days were chosen to attempt to illustrate the time-lag. For most of the figures in section 6.7.1, the strategy to detect time-lag is not based on when a cloud appears, but the opposite, when the node gains direct sunlight. This was due to the data received as a few of them seem to be clear-skied. The exception is figure 6.18. Analyzing the data, it was noticed a lack of data for actual weather for each day. This could provide a better understanding of received data and should be implemented in future work.

The two top graphs in figure 6.15 show similarities in behavior over the same period. Enhancing them shows a potential time-lag of up to 7 minutes. As LoRa-node 1 in the figure has a considerably lower power output, it's hard to be too conclusive. This supports the idea that the pool of data is poor. Figure 6.16 and 6.17 show a comparing between Lora-node 1 and 2, and Lora-node 3 and 2 on different times of the day. Looking at the top graphs, it's hard to imagine a pattern in the cloud movement. Figure 6.19 and 6.20 show some similarities between the top graphs, but also quite a bit of "noise". This noise could be due to the transistors acting odd, complex clouds or just two completely different parts of the sky as the direction of the clouds are unknown. Enhancing the images could show some indication of time-lag, but with the mentioned uncertainties, it's not conclusive.

A more plausible result can be seen in figure 6.18, where both nodes are most of the day in complete sunlight. The difference in produced power could be due to the angle towards the sun, orientation, and time of day. Nevertheless, enhancing graphs (a) and (b) to between 1500-1530, a time-lag of up to 4 minutes appears between (c) and (d). Looking at the other big production drop, shown in (e) and (f), a similar time-lag occurs. This time, it's up to 3 minutes.

Based on these results it's clear that more and more consistent data are necessary to achieve more conclusive results.

Chapter 8

Conclusion

In this thesis, a system for real-time gathering of acquisition of momentarily potential for delivery of electrical energy from solar panels was implemented to some extent. Even though the final solution was not in complete accordance with the original design, a solution was implemented. Each node is intended to operate indefinitely being self-sufficient from solar energy and smart power management. This introduced several challenges related to power management, instrumentation, and wireless connectivity.

Two different embedded designs were proposed, where one was based on the original idea, and the other a simplified alternative idea. The original idea was never implemented further than simple hardware testing, while the alternative design was realized by ordering multiple breakout boards and integrating them with a preexisting board similar to the original design. Four different nodes were deployed and overtime three of them collected power measurements from solar panels. During the deployment, features like synchronization properties, self-sufficiency, and proper measurements were investigated. The main idea behind the system has proven to work, but several problems and design flaws occurred which resulted in a sub-optimal data pool. More nodes on different locations are required for a better correlation between end-nodes in regards to the delivery of energy prediction.

Most of the acceptance criteria were satisfied and most of the ones that were not passed are mainly due to not enough and poor data. The end-node was never fully capable of handling unexpected errors. Also, bugs of unknown cause and origin occurred which both led to a more fragile system prone to weird and undefined behavior.

The full solution was deployed, and most proofs-of-concept were tested proven to work, except for the main concept of predicting energy production in solar panels.

Chapter 9

Future Work

Based on discussion in chapter 7 a number of points for improvements have been identified.

- Original Solution

1. More thourough testing of the BQ24210 and MAX4484.
2. Correct the UART connections in schematics and PCB.
3. Add I2C header for communication between nRF9160-DK and IMU.
4. Test ATmega324PB's ADC.
5. Implement drivers for SPI, the GNSS (TESEO-LIV3F) and IMU (ICM-20948) and Flash Memory (SST25VF080B) and test components.

- Alternative Solution

1. Update and finalize firmware for nodes with new transistor and battery logic.
2. Create a more robust waterproof packaging for each node.
3. Adopt angle and orientation feature for better measurements. Or at least measure those values when deploying a node.
4. Strive for locations with good coverage, or place new gateways.
5. Update and finalize firmware for nRF9160DK.
6. Add timestamp to uplink messages to account for air-time and drift when synchronizing.
7. Investigate optimal transmission scheme for when to transmit data to save power.
8. Do a proper current measurement of a single node while active.

Bibliography

[1] Trondheim - The Things Network Community. https://www.thethingsnetwork.org/community/trondheim/. Accessed: 2020-06-01.

[2] Norway. https://www.thethingsnetwork.org/country/norway/. Accessed: 2020-06-01.

[3] Coverage — Sigfox. https://www.sigfox.com/en/coverage. Accessed: 2020-06-01.

[4] Dekningskart — Telia. https://www.telia.no/dekning/. Accessed: 2020-06-01.

[5] Dekning for IoT på 4G. https://www.telenor.no/bedrift/iot/dekning/. Accessed: 2020-06-01.

[6] Rahul Balani. Energy consumption analysis for bluetooth, wifi and cellular networks. 01 2007.

[7] Chi Wai Chow, Bryan Urquhart, Matthew Lave, Anthony Dominguez, Jan Kleissl, Janet Shields, and Byron Washom. Intra-hour forecasting with a total sky imager at the UC San Diego solar energy testbed. *Solar Energy*, 85(11):2881–2893, nov 2011.

[8] L. Tessaro, C. Raffaldi, M. Rossi, and D. Brunelli. Lightweight synchronization algorithm with self-calibration for industrial lora sensor networks. In *2018 Workshop on Metrology for Industry 4.0 and IoT*, pages 259–263, 2018.

[9] 3GLTEinfo. LoRa Architecture - LoRaWAN Tutorial. http://www.3glteinfo.com/lora/lora-architecture/. Accessed: 2020-02-13.

[10] J. Peña Queralta, T. N. Gia, Z. Zou, H. Tenhunen, and T. Westerlund. Comparative study of LPWAN technologies on unlicensed bands for M2M communication in the IoT: Beyond Lora and Lorawan. In *Procedia Computer Science*, volume 155, pages 343–350. Elsevier B.V., jan 2019.

[11] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT.

In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2018*, pages 197–202. Institute of Electrical and Electronics Engineers Inc., oct 2018.

[12] Cellular networks for Massive IoT—Whitepaper - Ericsson. https://www.ericsson.com/en/reports-and-papers/white-papers/cellular-networks-for-massive-iot–enabling-low-power-wide-area-applications. [Accessed 2020-05-23].

[13] Badr Eddine Benhiba, Abdessalam Ait Madi, and Adnane Addaim. Comparative study of the various new cellular iot technologies. In *2018 International Conference on Electronics, Control, Optimization and Computer Science, ICECOCS 2018*. Institute of Electrical and Electronics Engineers Inc., jan 2019.

[14] Rashmi Sharan Sinha, Yiqiao Wei, and Seung Hoon Hwang. A survey on LPWA technology: LoRa and NB-IoT, mar 2017.

[15] Jie Shi, Wei Jen Lee, Yongqian Liu, Yongping Yang, and Peng Wang. Forecasting power output of photovoltaic systems based on weather classification and support vector machines. In *IEEE Transactions on Industry Applications*, volume 48, pages 1064–1069, 2012.

[16] Tore Mattias Apeland. Design of solar measurement iot node using lpwan. Master's thesis, 2019.

[17] Øystein Molvik. Embedded system for current measurement from solar panels. 2019.

[18] Microchip Technology Inc. RN2483 command reference. page 50, 2015.

[19] LoRa Alliance. LoRaWAN® Regional Parameters RP002-1.0.0 — LoRa Alliance®. https://lora-alliance.org/resource-hub/lorawanr-regional-parameters-rp002-100. Accessed: 2020-05-10.

[20] Tobias Ulfsnes Rasmussen. Investigation of Connectivity, Energy Consumption and Real-time Properties in a LoRa-Network, using Vibration Sensors as case. Master's thesis, NTNU, 2020.

[21] Solar Cell 12V DC Mini Solar Panel kit DIY For Battery Cell Phone Chargers Portable 12 Volt 1.5W 1.8W 1.92W 2W 2.5W 3W 4.2W Watt—Solar Cells— - AliExpress. https://www.aliexpress.com/item/33009882680.html?spm=a2g0s.9042311.0.0.59024c4db1CxL5. Accessed: 2020-06-04.

[22] 6V Solar Panel with 30/100/200cm wire Mini Solar System DIY For Battery Cell Phone Charger 0.6W 1W 1.1W 2W 3W 3.5W 4.5W Solar—Solar Cells— - AliExpress. https://www.aliexpress.com/item/32877897718.html?spm=a2g0s.9042311.0.0.59024c4db1CxL5. Accessed: 2020-06-04.

[23] Microchip Technology Inc. ATmega324PB Datasheet. 2017.

[24] Microchip Technology Inc. RN2483 datasheet. pages 1–22, 2019.

[25] nRF9160-SiP datasheet. *Building Research & Information*, 21(1), 1993.

[26] Maxim Integrated. MAX44284 datasheet.

[27] ST Microelectronics. Teseo-LIV3F datasheet. (September), 2019.

[28] ST Microelectronics. Teseo-LIV3F user manual. 3304(January):1–148, 2012.

[29] InvenSense. ICM-20948 datasheet. 2017.

[30] Texas Instruments. bq24210 datasheet. 2015.

[31] TPS6120 datasheet. 2008.

[32] Microchip Technology Inc. Mcp1252/3 datasheet. 2014.

[33] Texas Instruments. INA219 datasheet. (September), 2011.

[34] Texas Instruments. TPS63070 datasheet. Technical report, 2016.

[35] ON Semiconductor. BS170 datasheet. https://www.onsemi.com/pub/Collateral/BS170-D.PDF.

[36] RCA. 2N6178 Datasheet. https://datasheetspdf.com/datasheet/2N6178.html. Accessed: 2020-06-01.

[37] Nordic Semiconductor. nRF9160: Simple MQTT. https://github.com/nrfconnect/sdk-nrf/tree/master/samples/nrf9160/mqtt_simple. Accessed: 2020-04-10.

[38] API Reference — The Things Network. https://www.thethingsnetwork.org/docs/applications/python/ap reference.html. Accessed: 2020-05-11.

[39] paho-mqtt · PyPI. https://pypi.org/project/paho-mqtt/. Accessed: 2020-05-14.

[40] AN2711 Real-Time Clock Calibration and Compensation on AVR ® Microcontrollers Features. Technical report, 2018.

[41] Sunrise and sunset times in Trondheim. https://www.timeanddate.com/sun/norway/trondheim. Accessed: 2020-06-13.

Appendix A: Porting firmware to original solution

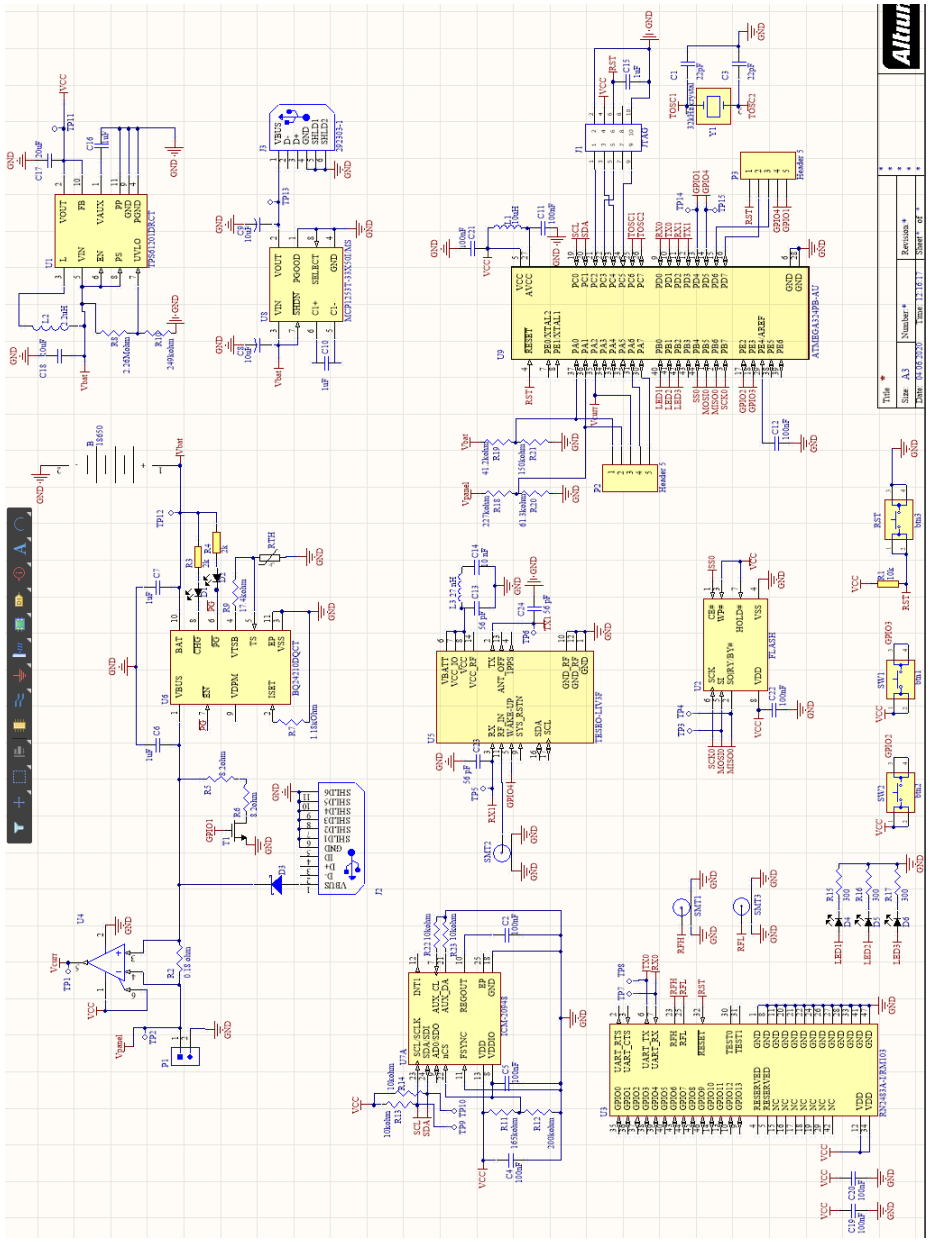
Most of the hardware are identical between the nodes and operate in a similar matter, but some of drivers is not necessary. The following list show what steps needs to be taken in order to port the firmware to the original PCB.

1. Port the following drivers: ADC, BOARD, RN2483A, USART, TIMERS, UTIL FUNCTIONS and I2C (for the IMU).
2. Port the following single files: ”config.h”, ”main.h”, ”twi.h”,
3. Update the ported drivers’ and files’ remove unnecessary dependencies (”#include ¡...h¿.
4. Add functions in ”board.c” to read ADC values for solar panel voltage and current.
5. Change the struct in ”board.h” to no longer use INA219, and instead add voltage and current from solar panel as variables.
6. Update the code lines in ”FSM.C” where solar panel voltage and current is calculated to use functions from pt. 4 instead of reading I2C via INA219.
7. Add desired features and desired ”define statements” in ”config.h” not available with the alternative solution.

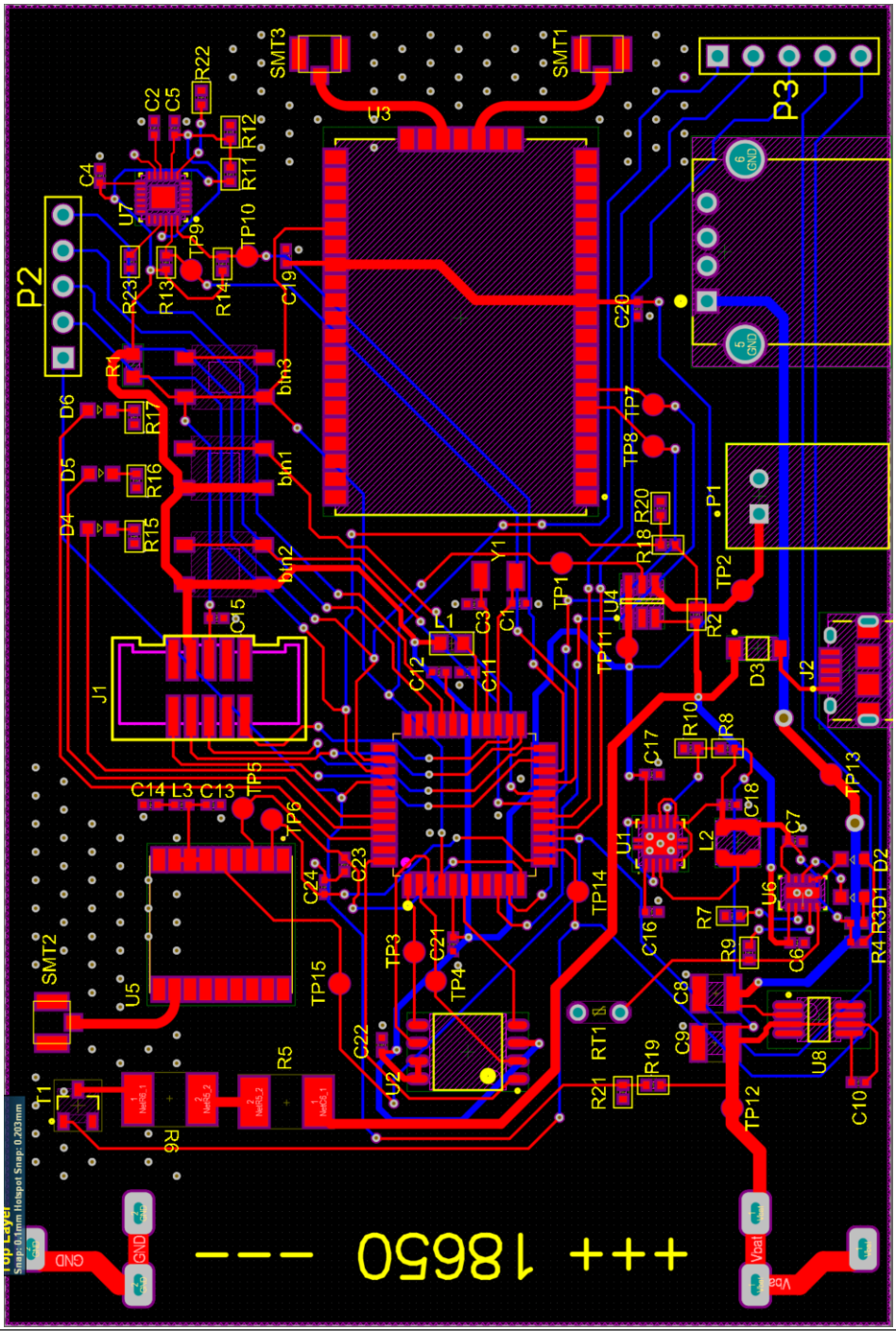
These are the main step required in order to use most of the code from the alternative solution with the original PCB. New drivers are also necessary to utilize all features available on the PCB (GNSS, IMU, Flash memory).

Appendix B: HW End-Node Original Solution

B1: Schematic



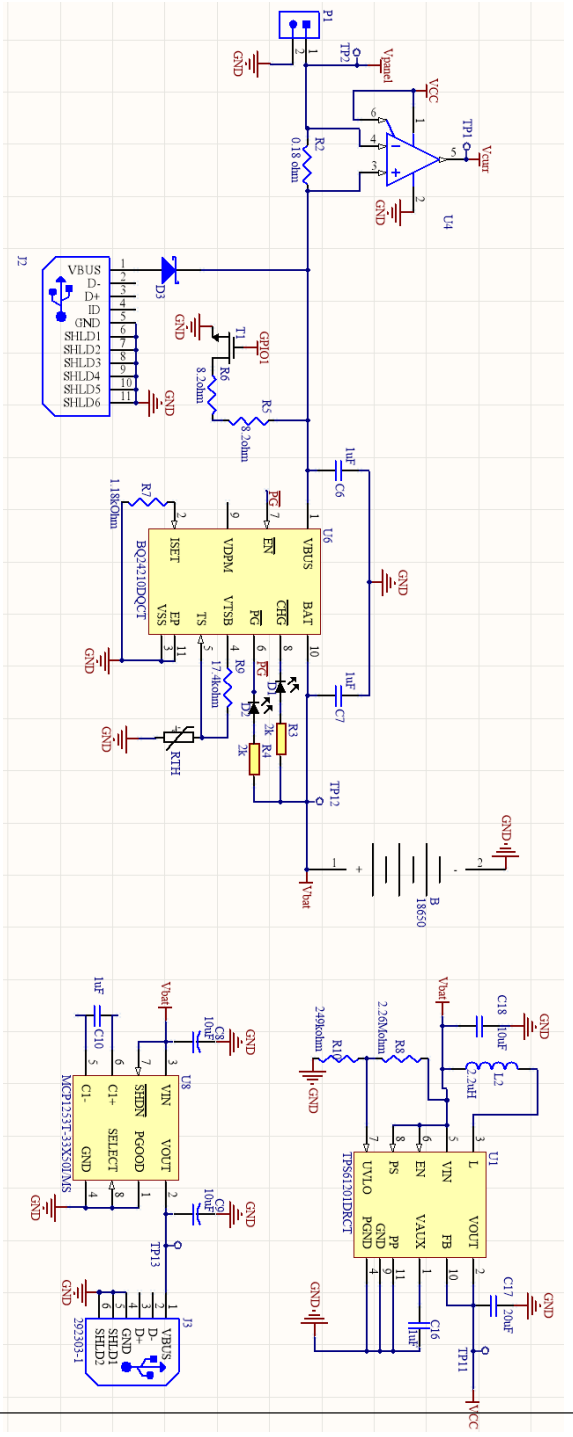
B2: PCB Design



B3: Parts list

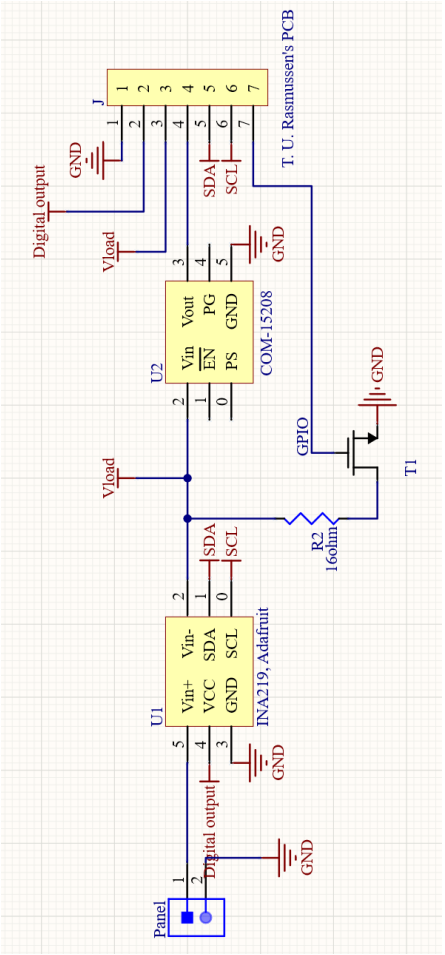
Designator	Description	Value	Package (inch [mm])
R5, R6	Power resistor	8.2 ohm	SMD (2512 [6432])
R2	Shunt resistor	0.18	SMD (0402 [1005])
R3, R4	Resistor	2 kohm	SMD (0402 [1005])
R9	RT1 (BQ24210)	17.4 kohm	SMD (0402 [1005])
R8	R3(TPS61201)	2.26 Mohm	SMD (0402 [1005])
R10	R4(TPS61201)	249 kohm	SMD (0402 [1005])
R1, R13, R14, R22, R23	Resistor	10 kohm	SMD (0402 [1005])
R11	Volt divider (IMU)	165 kohm	SMD (0402 [1005])
R12	Volt divider (IMU)	200 kohm	SMD (0402 [1005])
R15 , R16, R17	For LEDs	300 ohm	SMD (0402 [1005])
R19	Volt divider (MCU)	41.2 kohm	SMD (0402 [1005])
R21	Volt divider (MCU)	150 kohm	SMD (0402 [1005])
R7	R_ISET (BQ24210)	1.18 kohm	SMD (0402 [1005])
L3	Filter GPS	27 nH	SMD (0402 [1005])
L2	Inductor	2.2 uH	SMD(x [3030])
L1	Filter MCU	10 uH	SMD(0603 [x])
C1, C3	Crystal cap	22 pF	SMD (0402 [1005])
C13, C23, C24	Signal cap	56 pF	SMD (0402 [1005])
C14	Filter GPS	10 nF	SMD (0402 [1005])
C2, C4, C5, C11, C12, C19, C20, C21, C22	Decoupling cap	100 nF	SMD (0402 [1005])
C6, C7, C10, C15, C16	Decoupling cap	1 uF	SMD (0402 [1005])
C8, C9	Low esr cap	10uF	SMD(1210 [3528])
C18, C17	Decoupling cap	10 uF	SMD (0402 [1005])
D1, D2, D4, D5, D6	LEDs	Green	SMD (0603 [x])
D3	Micro usb diode	-	SOD-123
Y1	32.768 kHz crystal		SMD
SMT1, SMT2, SMT3	Antenna socket	-	SMD
J1	JTAG header	-	SMD
J2	Micro USB socket	-	SMD+TH
J3	USB-A socket	-	TH
P1	Solar panel socket	-	TH
P2	Board-to-board connector	-	TH
P3	Board-to-board connector	-	TH
btn1, btn2, btn3	Buttons	-	SMD
B	Battery clip	-	TH
U1	DC-DC converter	TPS61201DRCT	SMD
U2	Flash memory	SST25VF080B	SMD
U3	LoRa modem	RN2483A	SMD
U4	Current amplifier	MAX44284FAUT+T	SMD
U5	GPS	TESEO-LIV3F	SMD
U6	Battery charger	BQ24210DQCT	SMD
U7	9-axis IMU	ICM-20948	SMD
U8	DC-DC converter	MCP1253T-33X50I/MS	SMD
U9	Microcontroller	ATMEGA324PB	SMD

B4: Power module



Appendix C: HW End-Node Alternative Solution

C1: Schematic

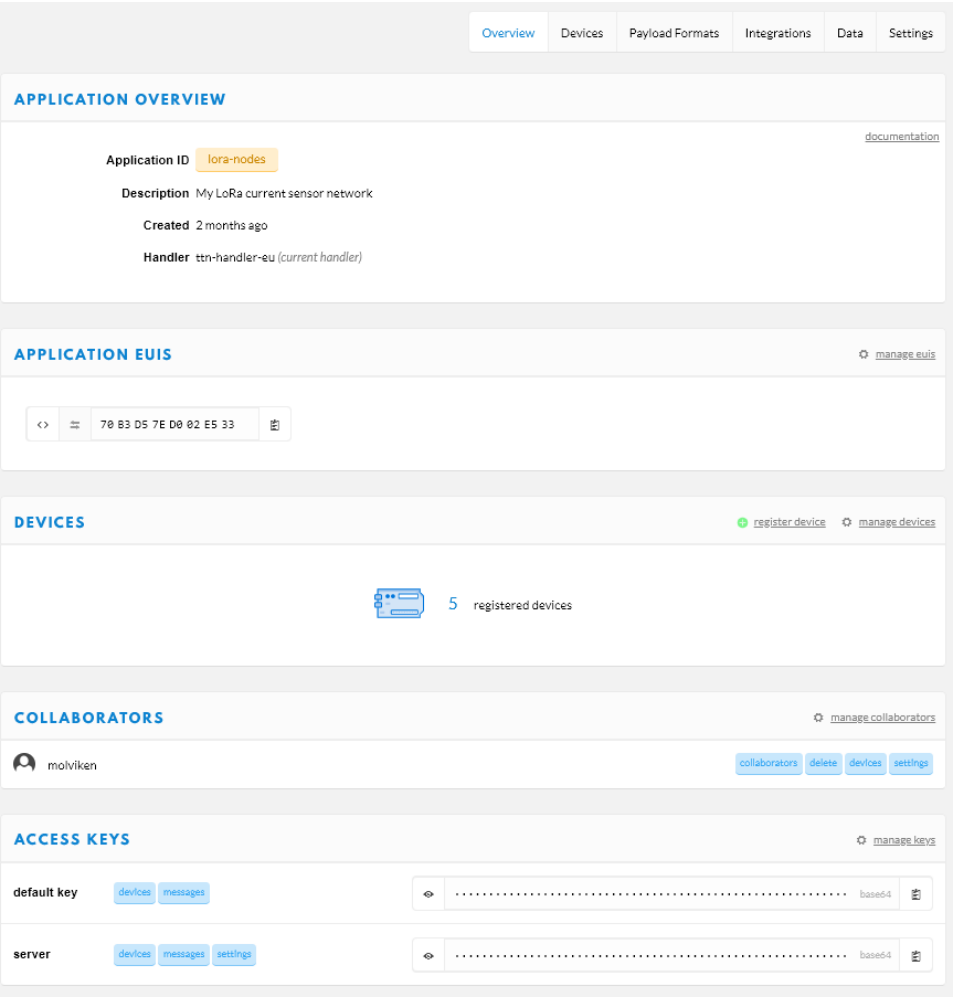


C2: PCB Design

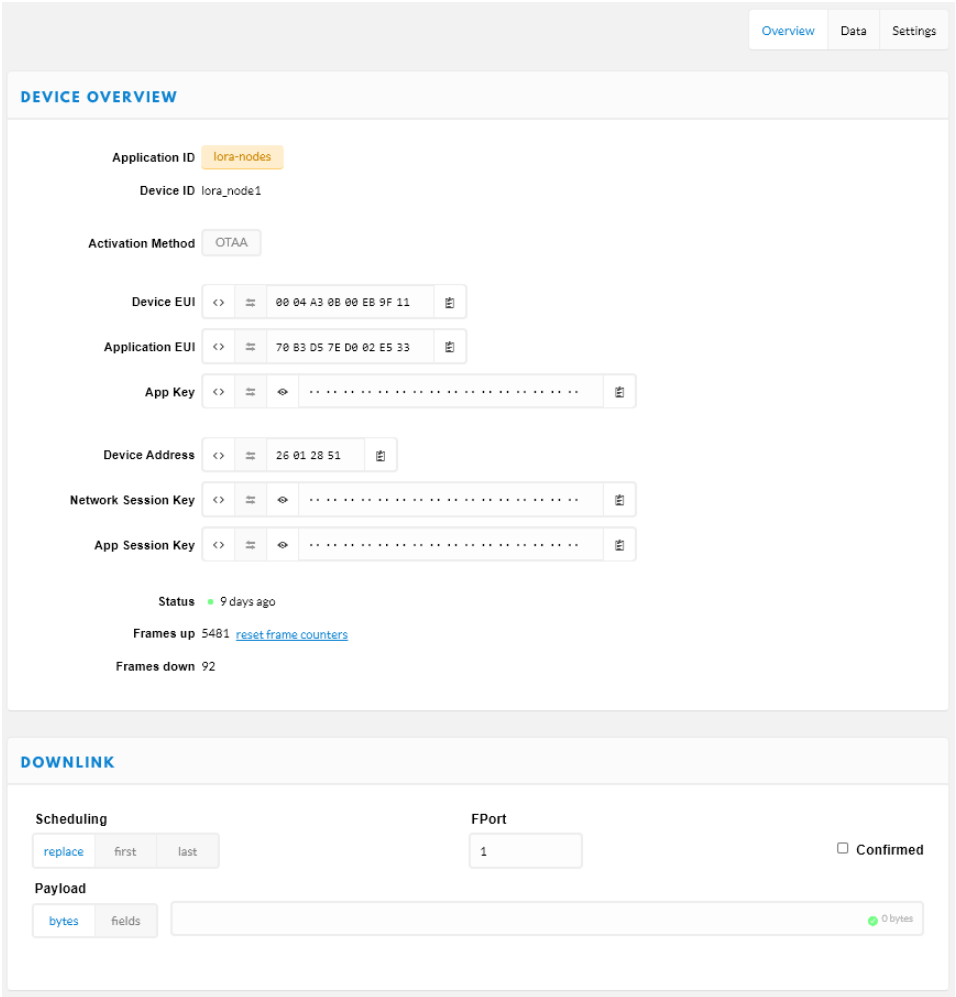
Designator	Description	Verdi	Packaging
T1	Through hole transistor	BS170, 2N6178	Through Hole
R2	Power resistor	3W, 16 ohm	Through Hole
Panel	Header for solar panel	-	Through Hole
J	Wire-to-board connector	-	Through Hole
U1	Current and voltage sensor	INA219	Through Hole
U2	Buck-boost converter	COM-15208	Through Hole

Appendix D: The Things Network

D1: Application server GUI



D2: Device control panel



Appendix E: Flowchart mapping to files

The appendix maps blocks and certain arrows in the flowchart to their respective code files. This is only if the script/software uses functions from multiple files.

E1: End-node

From state diagram (Figure 5.14:

1. START = main() in ”main.c”.
2. Remaining transitions and states are located in function FSM_RUN() in ”FSM.c” and run sequentially.

Flowchart illustrating SLEEP state (Figure 5.15:

1. Enter power-save mode: Function in ”timers.c”.
2. UART ISR / TIMER2 ISR: Interrupt service routines in ”FSM.c”.
3. Receive string: Function ran by the ISR and located in ”UART.c”.
4. remaining blocks occur sequentially in ”FSM.c” within the state.

Flowchart illustrating NOT_JOINED state (Figure 5.16:

1. lora_join_otaa: Function in ”RN2483A.c”.
2. Remaining blocks and arrows occur sequentially in ”FSM.c” within the state.

Flowchart illustrating ACTIVE state (Figure 5.17:

1. Enable and disable load circuit: Function in ”config.h”.
2. INA219 read voltage and current: Functions in ”INA219.h”.
3. Encode message: Function in ”util_functions.c”.
4. Transmit message: Running lora_transmit if LoRaWAN, located in ”RN2483A.c”. Running puts(message) if NB-IoT, located in ”FSM.c”.
5. Set receive flag: Set by lora_transmit in ”RN2483A.c”.
6. Remaining blocks and arrows occur sequentially in ”FSM.c” within the state.

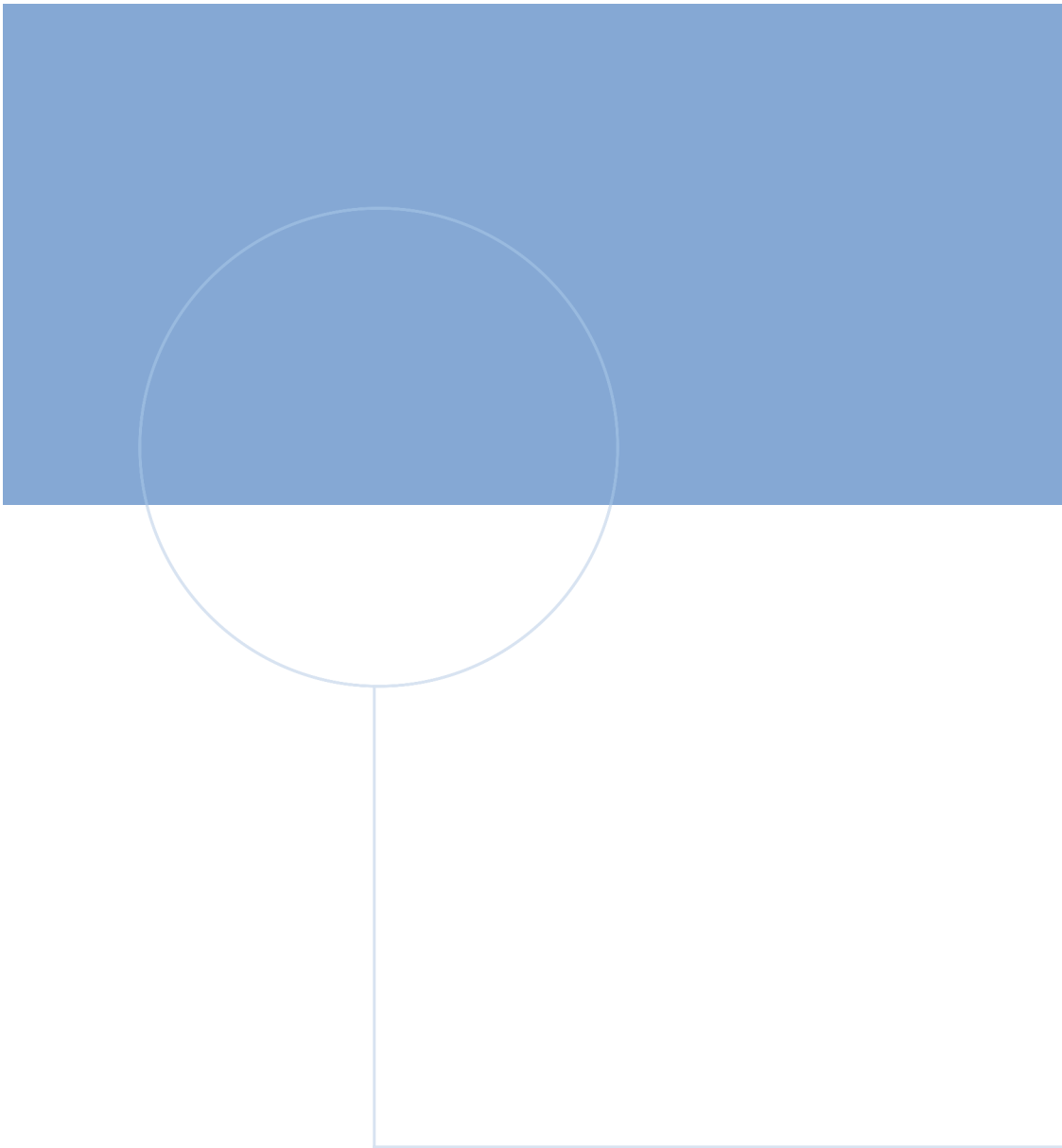
Flowchart for nRF9160 is ran by one file, ”main.c”.

E2: Back-end application

Flowchart illustrating main server script (Figure 5.19:

1. Logger: Function located in ”logger.py”.
2. Wait callback from ...: Behind the scenes infinite loop keeping connection alive and polls for callback.
3. Uplink callback received: When triggered, uplink_callback function is called. Located in main script ”TTN-server.py”.
4. Remaining blocks are located in original file ”TTN-server.py” and operate sequentially within their thread.

Flowchart illustrating node synchronization script (Figure 5.20) all happen sequentially in the file ”sync_nodes.py”.



NTNU

Norwegian University of
Science and Technology