

Oskar Østby

# Implementation of power conservation in IoT devices using mesh network

Master's thesis in Cybernetics And Robotics

Supervisor: Geir Mathisen

Co-supervisor: Eivind Tagseth

June 2022





Oskar Østby

# **Implementation of power conservation in IoT devices using mesh network**

Master's thesis in Cybernetics And Robotics  
Supervisor: Geir Mathisen  
Co-supervisor: Eivind Tagseth  
June 2022

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





## Masters Thesis Description

<b>Candidate:</b>	Oskar Østby
<b>Course</b>	TTK4900 - Engineering Cybernetics, Master's Thesis
<b>Thesis title (Norwegian)</b>	Implementasjon av energikonservering i IoT enheter ved bruk av mesh nettverk
<b>Thesis title (English)</b>	Implementation of power conservation in IoT devices using mesh network

### Thesis description:

As IoT devices get smaller and require more features, the demand for power conservation increases. In this thesis an implementation of power conservation, using mesh networking to amortize cost of Low-Power Wide-Area-Network (LPWAN) communication is proposed. The solution is then tested on a realistic scenario, tracking sheep at the pastures.

### The tasks will be:

1. Reviewing existing solutions and previous work, as well as a look into the requirements of sheep tracking.
2. Propose requirements and a test specification for a sheep tracking solution.
3. Propose a design for sheep tracker hardware and software solution.
4. As far as time permits implement the proposed sheep tracking solution and perform the specified tests.

**Start date:** 2022-01-10  
**Due date:** 2022-06-06

**Thesis performed at:** Department of Engineering Cybernetics  
**Supervisor:** Professor Geir Mathisen, Dept. of Eng. Cybernetics  
**Co-supervisor:** Evind Tagseth, Norbit ODM AS

# Summary

Energy conservation is paramount for devices in the Internet of Things (IoT). This thesis uses a sheep tracking scenario to investigate the feasibility of energy conservation, utilizing a low-power mesh network in conjunction with a Low-Power Wide-Area-Network (LPWAN) transmitter.

A literary study is performed, investigating existing sheep tracking solutions and research in the field of distributed systems.

Based on the literary study a system- and test specification are proposed and implemented.

The implemented system is tested, according to the specification, and the test results are examined and discussed.

The test results indicate that, with three or more nodes in a cluster, the system may outperform a baseline solution by up to 30%. The baseline solution is based on how state-of-the-art solutions most likely function.

# Sammendrag

Energisparing er avgjørende for enheter i Tingenes Internett. I denne oppgaven brukes sauesporing som et scenario for å undersøke muligheten for energisparing, ved å bruke et lav-energi mesh-nettverk sammen med en lav-energi WAN sender.

Et litteraturstudie er utført der eksisterende sauesporingsløsninger og forskning om distribuerte systemer undersøkes.

Basert på dette litteraturstudiet blir en system- og testspesifikasjon foreslått, og systemet implementert.

Deretter testes systemet, i tråd med spesifikasjonen, og testresultatene blir undersøkt og diskutert.

Testresultatene indikerer at dersom tre noder er samlet, så vil foreslåtte systemet utkonkurrere en naiv løsning med opptil 30%. Denne naive løsningen er basert på hvordan dagens systemer, mest sannsynlig, fungerer.

# Acknowledgements

I would like to say thank you to Geir Mathisen and Eivind Tagseth for their invaluable guidance during the development of this thesis. Jenny Røste, for helping me get an understanding of sheep farming and gathering. Eivind Yttri for his insight on how to improve sheep tracking and how sheep farming is done today. Ådne Karstad for letting me borrow his Power Profiler Kit, and for being a great sparring partner, throughout the writing of the thesis.

The hardware used in this thesis is provided by Norbit ODM AS. For measurements a Power Profiler Kit 2 is used, provided by Ådne Karstad. All the firmware created is built with the nRF Connect SDK, provided by Nordic Semiconductor.

The rest of the thesis is my own work.

# Table of Contents

Summary . . . . .	i
Sammendrag . . . . .	i
Acknowledgements . . . . .	ii
<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Scope . . . . .	2
1.3 Thesis structure . . . . .	3
<b>2 Previous Work</b>	<b>5</b>
2.1 Tracking solutions . . . . .	5
2.1.1 Findmy . . . . .	6
2.1.2 Telespor . . . . .	6
2.1.3 Nofence . . . . .	7
2.1.4 Smartbjella . . . . .	8
2.2 Hybrid communication systems . . . . .	8
2.3 Device to device communication . . . . .	9
2.4 Leader election . . . . .	9
2.5 Summary of previous work . . . . .	10
<b>3 Theory</b>	<b>11</b>
3.1 Mesh and IoT terminology . . . . .	11
3.2 Mesh networks . . . . .	11
3.3 LTE-M and Power Saving Mode . . . . .	12
3.4 Leader election algorithms . . . . .	13
3.4.1 Spanning Tree Election Algorithms . . . . .	13
3.5 Message Queuing Telemetry Transport protocol . . . . .	14
<b>4 Design requirements and specification</b>	<b>15</b>
4.1 Sheep tracker functionality . . . . .	15
4.2 Stakeholder requirements . . . . .	16
4.3 Hardware specification . . . . .	17
4.4 Software requirements . . . . .	17
4.5 Analysis of requirements . . . . .	18
4.5.1 Traceability matrix . . . . .	21
4.6 Test specification . . . . .	22
4.6.1 Functional tests . . . . .	22
4.6.2 Energy usage tests . . . . .	23
4.6.3 Mechanical tests . . . . .	23
4.6.4 Test coverage matrix . . . . .	24
<b>5 Implementing, and testing, a sheep tracking system</b>	<b>25</b>
5.1 Hardware design . . . . .	26

---

5.2	Sheep tracking hardware . . . . .	27
5.3	Software design . . . . .	30
5.4	Tracking backend software . . . . .	31
5.5	Tracking firmware . . . . .	32
5.5.1	NC2400 driver . . . . .	33
5.5.2	Coordinator module . . . . .	33
5.5.3	Application layer protocol . . . . .	36
5.5.4	Sensor module . . . . .	37
5.5.5	Cloud module . . . . .	37
5.5.6	State machine . . . . .	37
5.6	Firmware image for performance comparison . . . . .	39
5.7	Requirement coverage . . . . .	40
5.8	Testing the implementation . . . . .	42
5.8.1	Functional tests . . . . .	44
5.8.2	Mechanical . . . . .	45
5.8.3	Energy usage . . . . .	46
<b>6</b>	<b>Results</b>	<b>49</b>
6.1	Mechanical test results . . . . .	49
6.2	Functional test results . . . . .	51
6.3	Power usage test results . . . . .	54
6.3.1	10 minute interval - baseline . . . . .	54
6.3.2	10 minute interval - gateway mode . . . . .	56
6.3.3	10 minute interval - node mode . . . . .	57
6.3.4	1 hour interval - baseline . . . . .	59
6.3.5	1 hour interval - gateway mode . . . . .	60
6.3.6	1 hour interval - node mode . . . . .	62
6.3.7	Power usage summary . . . . .	63
6.4	Error sources . . . . .	63
6.4.1	Missing datapoints . . . . .	64
<b>7</b>	<b>Discussion</b>	<b>65</b>
7.1	Test results . . . . .	65
7.1.1	Mechanical tests . . . . .	65
7.1.2	Functional tests . . . . .	66
7.1.3	Energy consumption . . . . .	67
7.1.4	Tracker weight . . . . .	67
7.1.5	Requirement coverage . . . . .	68
7.2	Performance parameters . . . . .	69
7.2.1	Application specific timing parameters . . . . .	69
7.2.2	LTE parameters . . . . .	69
7.2.3	NeoMesh parameters . . . . .	70
7.3	Shortcomings and possible improvement . . . . .	70
7.3.1	GNSS . . . . .	70
7.3.2	Desirability factor . . . . .	71
7.3.3	More tracker hardware variants . . . . .	71
7.4	Notes about mesh networks . . . . .	71
7.5	Beyond sheep tracking . . . . .	72
7.6	System complexity . . . . .	72
<b>8</b>	<b>Conclusion</b>	<b>73</b>
<b>9</b>	<b>Further work</b>	<b>74</b>
	<b>Bibliography</b>	<b>75</b>
	<b>Appendix</b>	<b>77</b>
<b>A</b>	<b>Norbit Tracker Schematics</b>	<b>78</b>

---

---

<b>B</b>	<b>Logfile including several elections</b>	<b>82</b>
<b>C</b>	<b>Database printouts</b>	<b>85</b>



# List of Figures

1.2.1	Illustration of the full sheep tracking system. . . . .	3
2.1.1	Image of the tracker from Findmy on a sheep . . . . .	6
2.1.2	Image of the tracker from Telespor . . . . .	6
2.1.3	Image of the tracker from Nofence . . . . .	7
2.1.4	Image of the tracker from Smartbjella . . . . .	8
3.2.1	Comparison of traditional and mesh network . . . . .	12
3.3.1	PSM timers. . . . .	13
3.5.1	Illustration of the MQTT protocol . . . . .	14
5.1.1	Hardware component diagram . . . . .	26
5.2.1	Simplified overview of hardware design . . . . .	27
5.2.2	NC2400 breakout board . . . . .	28
5.2.3	Image of the Norbit tracker with NC2400 . . . . .	28
5.2.4	Norbit tracker schematic overview . . . . .	29
5.2.5	Norbit tracker power schematics . . . . .	29
5.2.6	Norbit tracker nRF9160 schematic . . . . .	30
5.3.1	Overview of the software design . . . . .	30
5.4.1	Screenshot of database entries . . . . .	32
5.5.1	Overview of the sheep tracker firmware . . . . .	32
5.5.2	NC2400 driver illustration . . . . .	33
5.5.3	State diagram of the coordinator module . . . . .	34
5.5.4	Coordinator protocol header. . . . .	34
5.5.5	Overview of coordinator protocol messages. . . . .	35
5.5.6	Application protocol header . . . . .	36
5.5.7	Overview of the tracker state machine . . . . .	38
5.5.8	State diagram of the Gateway state . . . . .	38
5.5.9	State diagram of the Node state . . . . .	39
5.8.1	Power Profiler application . . . . .	43
5.8.2	Power Profiler configuration menu . . . . .	43
5.8.3	Power Profiler statistics of current window . . . . .	44
5.8.4	Power Profiler statistics of selection . . . . .	44
5.8.5	Overview of estimated current usage. . . . .	47
5.8.6	Image of the field test setup. . . . .	48
6.1.1	Illustration showing measurements of the tracker. . . . .	49
6.1.2	Figures with all the tracker measurements . . . . .	50
6.1.3	Battery weight. The scale reads 10 grams. . . . .	50
6.2.1	Database data where tracker is removed and node desirability falls . . . . .	51
6.2.2	Screenshot of database where tracker is removed and sheep tracker 7 takes over as gateway. . . . .	52
6.2.3	Screenshot of reported tracker neighbours from database. . . . .	52
6.2.4	Screenshot of reported tracker status after node 7 rejoins the cluster. . . . .	53
6.2.5	Screenshot of reported tracker neighbours after node 7 rejoins the cluster. . . . .	53
6.2.6	Screenshot of tracker status after node 7 rejoins the cluster. . . . .	54

---

6.2.7	Screenshot of reported tracker positions from database. . . . .	54
6.3.1	Statistics from baseline firmware during 10 minute interval tests. . . . .	54
6.3.2	Data from baseline firmware, with a single transmit interval selected . . . . .	55
6.3.3	Data from baseline firmware, view of the floor current . . . . .	55
6.3.4	Data set of a tracker in gateway mode with 10 minute transmission intervals . .	56
6.3.5	View of a floor current from tracker running gateway mode with a 10 minute interval . . . . .	57
6.3.6	Data set of a tracker in node mode with 10 minute intervals . . . . .	58
6.3.7	View of a sleep interval from tracker running node mode . . . . .	58
6.3.8	Energy usage statistics of baseline application with 1 hour transmit intervals . .	59
6.3.9	Overview of 1 hour interval, baseline firmware . . . . .	59
6.3.10	Baseline firmware floor current during 1 hour interval setup . . . . .	60
6.3.11	Energy usage stats of gateway with 1 hour transmit intervals . . . . .	60
6.3.12	Overview of 1 hour interval, gateway mode . . . . .	61
6.3.13	Gateway mode floor current during 1 hour interval setup . . . . .	61
6.3.14	Energy usage stats of node with 1 hour transmit intervals . . . . .	62
6.3.15	Overview of 1 hour interval, node mode . . . . .	62
6.3.16	Node mode floor current during 1 hour interval setup . . . . .	63
6.4.1	PPK view showing missing samples in the dataset. . . . .	64

# List of Tables

2.1.1	Summary of parameters from different livestock tracking solutions . . . . .	5
4.5.1	Traceability matrix between stakeholder- and technical requirements. . . . .	22
4.6.1	Proposed test configurations for energy usage . . . . .	23
4.6.2	Test coverage matrix. . . . .	24
5.8.1	Test procedure for network integrity tests. . . . .	45
5.8.2	Test procedure for mechanical tests . . . . .	46
6.3.1	Test results for energy usage tests . . . . .	63
7.1.1	Test results for mechanical tests . . . . .	65
7.1.2	Test results for network integrity tests . . . . .	66
7.1.3	Stakeholder requirement coverage matrix . . . . .	68

# List of Abbreviations

<b>3GPP</b>	3rd Generation Partnership Project
<b>A-GNSS</b>	Assisted GNSS
<b>ADC</b>	Analog Digital Converter
<b>API</b>	Application Programming Interfaces
<b>AWS</b>	Amazon Web Services
<b>BLE</b>	Bluetooth Low Energy
<b>CTS</b>	Clear To Send
<b>D2D</b>	Device to Device
<b>DRX</b>	Discontinuous Reception
<b>DUT</b>	Device Under Test
<b>FSM</b>	Finite State Machine
<b>GLONASS</b>	Global'naya Navigatsionnaya Sputnikovaya Sistema
<b>GNSS</b>	Global Navigation Satellite Systems
<b>GPS</b>	Global Positioning System
<b>IMU</b>	Inertial Measurement Unit
<b>IoT</b>	Internet of Things
<b>IP</b>	Internet Protocol
<b>LoRaWAN</b>	Long Range Wide-Area-Network
<b>LPWAN</b>	Low-Power Wide-Area-Network
<b>LTE</b>	Long Term Evolution
<b>LTE-M</b>	Long Term Evolution - Machine type
<b>MQTT</b>	Message Queuing Telemetry Transport
<b>NB-IoT</b>	Narrow Band - Internet of Things
<b>PCB</b>	Printed Circuit Board
<b>PPK</b>	Power Profiler Kit
<b>PPK2</b>	Power Profiler Kit 2
<b>PSM</b>	Power Saving Mode
<b>pTAU</b>	Periodic Tracking Area Update
<b>QoS</b>	Quality of Service
<b>RTOS</b>	Real Time Operating System
<b>SDK</b>	Software Development Kit
<b>SoC</b>	System on Chip
<b>SPI</b>	Serial Peripheral Interface
<b>SQL</b>	Structured Query Language
<b>TAU</b>	Tracking Area Update
<b>UART</b>	Universal Asynchronous Receive Transmit
<b>UE</b>	User Equipment

# Chapter 1

## Introduction

Internet of Things (IoT) is a field of study that aims to create interconnections between things around us and the internet. The word *thing* is often used to describe a device, usually small and battery-powered, which consists of a sensor and/or an actuator and a connection to the internet. A *thing* in IoT is often very specialised, for example, a small temperature and moisture sensor or a network-attached light switch.

Having an increasing amount of things connected to the internet allows the observation and control of many processes throughout society. Industries may use data and actuation to monitor, control and optimize large scale processes. Private homes can become smart homes, reducing energy usage and streamlining day to day life.

IoT faces a lot of challenges to be practical. Amongst them, the battery life of a thing is of high importance. Longer battery life will lead to lower maintenance cost and higher uptime. Because of this, battery conservation is essential for devices to survive as long as possible.

This thesis will take a look at a technique to conserve energy in an IoT application using a hybrid communication system. The communication system will consist of a low-power, short-range, mesh network and a long-range communication technology for communication with the internet.

The rest of this chapter will present the background, scope and structure of the thesis.

### 1.1 Background

Battery life, being a crucial factor to success in the IoT industry, leads to companies utilizing a broad selection technologies to achieve the lowest possible power consumption in a device. The goal is to be able to transmit data to the internet while using as little power as possible. Various communication technologies are developed to operate at low power and data rate, directly communicating to the internet. Some of these include Long Range Wide-Area-Network (LoRaWAN), Long Term Evolution - Machine type (LTE-M) and Narrow Band - Internet of Things (NB-IoT). These connections are designed for long-range, low energy communication.

For different use cases even lower power mesh-networks can be utilized. Some examples are Bluetooth Low Energy (BLE) Mesh, ZigBee, Thread, NeoMesh, etc. These technologies are designed to operate at much lower ranges, but they require a dedicated gateway to forward data from the local mesh-network to the internet. This gateway is normally connected to the grid and is constantly listening for broadcast messages, resulting in a tendency to use a lot of energy. These gateways are also often fixed in place, which means they are virtually useless in networks where nodes are moving.

This thesis will explore a hybrid solution, utilizing both a long-range direct uplink and a low-range network. The approach will attempt eliminate the need for a grid powered gateway and reduce the

---

number of transmissions utilizing the direct uplink, compared to a network using only long-range uplinks. Earlier work, on the usage of the two different networks in conjunction, suggests that it is possible to conserve power by using a hybrid solution [1]. The work from [1] is mainly theoretical and a practical solution is still desired, to verify that the hybrid solution is feasible under real operating conditions.

A few key questions that needs to be asked is whether such an implementation will lead to similar results, or if the non-ideal world will limit the potential of this sort of power conservation? And will such an implementation be possible to actually use?

Implementing such an energy conservation technique for IoT networks can be beneficial for many use cases. To help set the scope of the thesis, a specific use case will be chosen to show the feasibility of the solution. Following in the footsteps of [1], this thesis will also use sheep tracking as the central use case.

Sheep farming is a large industry and the end product is the sheep themselves. A sheep farmer may own up to 400 ewes and 800 lambs. In the spring the ewes will give birth to one or two lambs, which will stay with their mother until they grow up. To allow the ewes and lambs to gain weight and grow larger they are released into pastures in May or June, depending on the type of pasture, and gathered in the end of September. In Norway sheep farmers usually use pastures in the mountains or in the forest. The forest pastures allow for earlier release of the sheep. The pastures in the forest are usually dense and when gathering the sheep you have a short line of sight, making it difficult to gather sheep that are not moving. Pastures in higher altitudes have less trees and a greater line of sight when gathering sheep. The lambs will have gained between 10 and 15 kg during the grazing season [2].

During grazing an ewe and its two lambs will stick together and are considered a *set*. If lambs are separated from their mothers, the lambs will most likely not survive. In an event, for example a predator attack, where the mother is killed (or in another way separated from her lambs), the lambs may still survive. The sheep farmer can release a new ewe, with no lambs of her own, which may adopt the lambs. For this to work the farmer have to set out the new ewe close enough to the abandoned lambs, so that they have a chance of meeting.

Since the ewes and lambs will stick together, a natural clustering of possible IoT nodes will occur, if trackers are attached to both ewes and lambs. This means that a reasonable assumption is that at least three nodes will be within range of each other, when sheep are at the pastures.

At the end of the grazing season the sheep needs to be gathered. The farmer will often require a lot of help from volunteers or hired help to be able to cover the large pastures. Everyone goes out to the pastures and tries to herd as many sheep as possible back to a truck to be transported home.

From an interview conducted with a local sheep farmer, the issues with existing solutions were found to be mainly unit price and weight. Being able to conserve more energy may help drive the weight, and perhaps price, of trackers down. These considerations and problems will become the basis scenario of the thesis.

## 1.2 Scope

There are countless optimizations and design decisions that can be done to create a working sheep tracking solution. The focus of the thesis is not to create an industrialized solution, but rather to verify that the proposed hybrid IoT system can be used to improve the battery life of IoT nodes.

The scope of the thesis is shown in Figure 1.2.1, where the entire sheep tracking system is also illustrated.

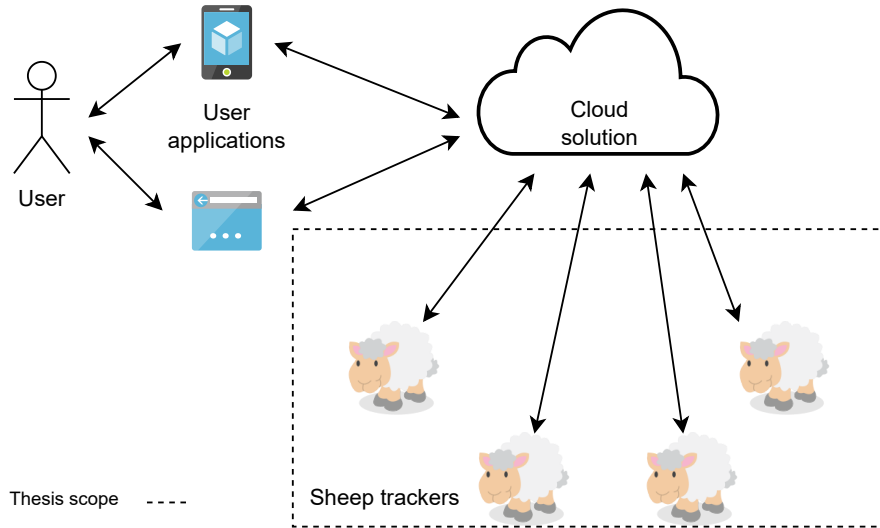


Figure 1.2.1: Illustration of the full sheep tracking system, where the scope of the thesis is within the dashed box.

A simple proposal of a hardware design will be presented so that the work may be verified, but the hardware design and layout will not be thoroughly discussed. The design will be based on an IoT Global Navigation Satellite Systems (GNSS) tracker, provided by Norbit ODM AS, with some modification. In brief the scope of the hardware in this thesis is to outline the components needed to create a verifiable design, any optimizations or improvements to the design is out of scope.

There will be produced firmware for the tracker which will be the main focus of the implementation. The algorithms or solutions used here will try to minimize power usage, but will again be focused towards verifying usability and will not be fully optimized for a production environment.

To receive, store and present data, some sort of cloud solution will be required for a fully functional solution. There are many design considerations to be made before designing a cloud solution, but many solutions already exists and is therefore out of scope. There may be a need for a backend solution to be able to read out data from the cloud, but this will only be briefly explained and not part of the evaluation of the design. To ensure that data is not lost, the choice of communication protocol with the cloud will be a part of the scope, all though it will not be subject of any discussion.

There are some mechanical aspects of the tracking solution as well, like mounting, casing and waterproofing. Since solutions to these problems already exists it is assumed that they will be solvable for the proposed tracker and considered out of scope.

The issue of price and weight of the tracker has been described as an issue. While the weight of the trackers will be within the scope of the thesis, the unit price of a single tracker will not. Pricing of hardware and software is quite complex and dependent on many factors which will be way out of scope for a single thesis. While the casing and mounting, described earlier, will indeed affect the weight of the tracker it will still be in scope to discuss the weight of the resulting sheep tracking hardware.

### 1.3 Thesis structure

This chapter have introduced the thesis, its background and scope. In the following chapter, Chapter 2, existing solution and research on relevant topics will be presented and discussed. To make sure that all theory is clear to the reader, Chapter 3, will introduce and explain some key concepts to make sure the rest of the thesis is understood.

Following these introductory chapters, Chapter 4, will establish the functional requirements of the

---

solution, then divide them into a set of technical requirements. These requirements will form the full specification for the solution. In addition a test specification will be created as a guideline on how to verify the requirements.

In Chapter 5 an implementation, that will attempt to fulfil the proposed specification, will be presented. Chapter 5 will also propose a set of tests that will ensure that the specification is fulfilled. The implementation is then tested, according to the tests proposed in Chapter 5, and the results from these tests are presented in Chapter 6.

The results, any further work, and whether or not the implementation covers the proposed requirements, will be discussed in Chapter 7 and a short conclusion will be given in Chapter 8. Finally a summary of any further work will be presented in Chapter 9.



## Chapter 2

# Previous Work

This chapter will provide an overview of *state-of-the-art* solutions and research topics, relevant to the thesis. Reviewing this work will make sure that the sheep tracking system is on par with existing solutions. There will also be some previous work on power conservation, hybrid communication systems and other algorithms that may provide value to this thesis. This chapter will first compare various existing sheep tracking solutions, then continue over to academic research on subjects that are relevant to creating the tracking solution. Lastly there will be a short summary, noting the most important details from the presented research.

### 2.1 Tracking solutions

In the Norwegian market today, a few companies already exist that provides sheep tracking services. The companies, Findmy [3], Telespor [4], Nofence [5] and Smartbjella [6] will be examined. They are the most prominent, if not largest, companies that provide tracking solutions for livestock in Norway.

In summary Table 2.1.1 contains the most important specifications for each solution. "Importance" in this case will be the parameters that is within the scope of the thesis' solution, i.e. battery life, weight and communication method. The data from the table is extracted from each companies website and a few emails for any missing data [7]–[10].

Company	Battery life	Weight	Communication technology
Findmy	2 - 3 seasons	200g	Satellite
Telespor	86 days to 6 years	104 g	LTE-M and NB-IoT
Nofence	1 week to 3 months	505 g	2G and LTE-M
Smartbjella	up to 10 years	170 g	NB-IoT

Table 2.1.1: Summary of parameters from different livestock tracking solutions

While the parameters in the table is important for the thesis it does not cover pricing, subscription model, user interface, coverage maps, etc. These parameters would be of great interest to a potential user, but are out of scope of the thesis.

In the following sections each tracker will be described in a bit more detail, but as with the table above, some factors like support, user interface, Application Programming Interfaces (API), etc. will not be discussed. The goal of the section is to get an overview of important technical metrics. These metrics will, in turn, be used to create a set of requirements and specification for the solution proposed in Chapter 4.

---

### 2.1.1 Findmy

Findmy is a Norwegian based company that produces asset tracking for livestock, gear and humans. They are the only company, out of the four presented, that uses satellite coverage [11] on their trackers. They use an American system, named Globalstar, which provides commercial satellite solutions [12]. The tracker itself can be seen in Figure 2.1.1.



Figure 2.1.1: Image of the tracker from Findmy on a sheep

The Globalstar chips all include a satellite transmitter and built in GNSS receiver. Globalstar also sell a chip that allows two way communication with the satellite network. Findmys decision to use satellite makes their trackers unaffected by the, possibly poor, cellular coverage in the Norwegian mountains and forests.

Findmy, as per writing the thesis, charges 1980 NOK per tracker plus an additional 229 NOK per tracker, per year in subscription fees. This subscription also includes batteries.

### 2.1.2 Telespor

Telespor is another Norwegian based company that produces tracking solutions for livestock. Telespor features an LTE-M and NB-IoT transceiver, as well as a GNSS receiver for positioning. Their tracker do not include any other specific features, as of writing this thesis, at least not advertised on their website. The tracker is shown in Figure 2.1.2.



Figure 2.1.2: Image of the tracker from Telespor

---

Telespor does have the lightest tracker of the lot. This might be a result of the focus on a smaller set of core features, letting them shave off more weight, but this is guesswork. Whatever the reason might be, they have a tracker that is 56 g lighter than the second lightest.

The unit price of the tracker is currently at about 1000 NOK. This does not include a subscription to their cloud services. So in total the price will be 1000 NOK per tracker and an additional 200 NOK in subscription fees, which also includes a battery.

### 2.1.3 Nofence

Nofence is a Norwegian company that claims to be the worlds first company to develop virtual fencing technology. They have developed solutions for goats, sheep and cows and have started to deliver to customers internationally. The tracker from Nofence is significantly heavier than its competitors, at 505 g. From comparing the different trackers it does seem that their focus is more towards the geofencing solution, even though tracking is also a part of their solution [13]. The Nofence bell can be seen in Figure 2.1.3.



Figure 2.1.3: Image of the tracker from Nofence

The tracker itself also includes a set of solar panels, a 10,000 mAh(10 Ah) battery, accelerometers for motion sensing, 2G and LTE-M transceivers, GNSS receiver and a Bluetooth chip. This large set of features might explain the weight of a tracker. The electronics needed to deliver the geofence shock might also add significant weight to the solution (geofencing often works by delivering a small electric shock to the animal, after a few warning sound queues, as it crosses the configured geofence).

An interesting note about their solution is the Bluetooth chip. While other trackers also include a Bluetooth interface, Nofence's solution also includes something called a shelter beacon. The shelter beacon is mounted in a location where the livestock is known to gather for longer durations. Based on the information available on their website, the trackers then notice when they enter the range of such a shelter beacon. The shelter beacon will then act as a gateway, for the trackers within range, allowing trackers to turn off their GNSS and cellular functionality to save power. An educated guess is that the shelter beacon registers the IDs of trackers in range, and then reports those IDs to the cloud. The position of the beacon is known so the trackers' positions will be based on the position of the beacon.

The pricing of the Nofence is not disclosed on their website as it is quote based.

It is worth to note that even with their large battery pack and solar panels they do not advertise more than 1 week to 3 months of battery life. They do advertise better battery life on their new models, but this information is not available on their website, as of writing this thesis.

---

### 2.1.4 Smartbjella

Smartbjella uses NB-IoT as their communication method, they advertise a battery life up to 10 years and include features like geofence and motion detection. Their is shown in Figure 2.1.4.

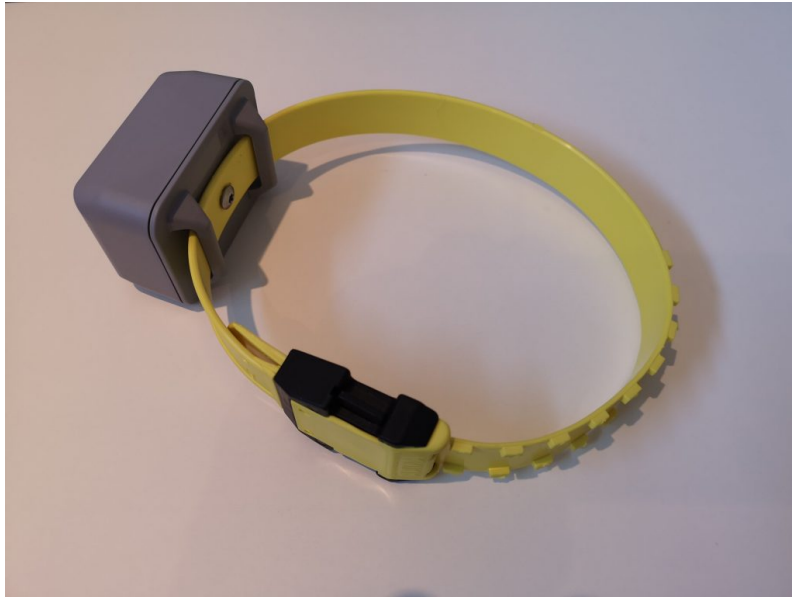


Figure 2.1.4: Image of the tracker from Smartbjella

Smartbjella have the longest advertised battery life of all the trackers at 10 years. This number was taken from their website and is probably based on ideal conditions.

Their pricing is similar to Telespor's at 999 NOK plus a 250 NOK subscription fee.

## 2.2 Hybrid communication systems

Hybrid communication systems where two types of communication technology is used in conjunction to improve performance. The performance increase may be in terms of robustness, connectivity/coverage, energy usage, etc. The term probably comes from telecommunication, combining Internet Protocol (IP) and traditional analog telephone communication.

Wu, Qiu, Wu *et al.* talk about hybrid systems in the IoT world [14]. They outline issues that will come with the rapid expansion of IoT devices that needs to be connected to the internet, primarily the issue of delay. In their work they propose a intermediary IoT node that is fitted with both short and long range transmission capabilities. This node is used to lower the delay when communicating with the cloud. While their solution focuses more on a more powerful router node, that consumes a larger amount of energy, it still showcases the usefulness of combining a set of different communication protocols.

As far as the author could find there exists little work done on the concept of combining different communication technologies in the IoT space. This might be due to existing communication technologies are already fairly tailored towards IoT use cases, where mains power is available and gateways may be placed.

---

## 2.3 Device to device communication

Device to Device (D2D) communication is the concept that devices, or things, communicate between themselves. This communication may be to make decisions, share data, relay information, etc.

Lianghai, Han, Liu *et al.* describes how a large influx of new things into the Internet of Things may overload the legacy cellular networks [15]. In their article, they describe a solution to solve this problem using context-aware D2D communication to reduce strain on the cellular base stations.

Their proposal describes a solution where sensors that are close together may use a single node as a cellular relay node that can relay state for sensors close by. While the problem that Lianghai, Han, Liu *et al.* and this thesis tries to solve is not equivalent, the work from [15] does indicate that the core concept of the solution in this thesis is sound.

To achieve high efficiency the article states that side-link communication, i.e. D2D communication, only should happen between nodes that are close together. They solve this by applying a clustering algorithm which finds nodes that may use D2D communication, forming a cluster. Then when a cluster is formed nodes are assigned a transmission mode, which may be relaying, side-link or direct transmission. This assignment is done by a base station.

The direct transmission mode transmits data directly to the base station. The side-link transmission mode indicates that the node should transmit its data to a relay node which relays the data to the base station. The relay transmission mode accepts data from side-link nodes and relays that data to the base station.

Pawar and Trivedi also write about the importance of D2D communication in the ever-growing Internet of Things [16]. They highlight benefits like real-time data exchange from devices based on proximity, communication between vehicles and ad hoc networks in disaster relief to ensure end-to-end communication as some of the possible benefits of D2D. They also discuss the possibility of reducing traffic load on general IoT infrastructure. While this paper highlights many benefits it also focuses on some of the issues and challenges with this technology. Things like radio interference when more devices are within proximity, security as more and more data is wirelessly exchanged and the lack of route discovery mechanisms to enable high usage of D2D communication.

Bello and Zeadally write about the concept of intelligent D2D communication. They highlight the exchange of information in end devices without the inclusion of any centralized control as a major feature of D2D communication. This exchange may be turned into context-aware sensors and devices on the edge and further showcases the many possibilities of using D2D communication.

Steri, Baldini, Fovino *et al.* proposes a protocol for D2D communication using LTE [18]. Their paper also describes the problems around the aggregate traffic volume that can be created by a large set of IoT nodes. From the looks of the work done, it is similar to the solution tested in [15].

## 2.4 Leader election

In distributed systems, one challenge is to agree on certain decisions. If no leader or governing entity exists in the system a common consensus must be made so that all nodes agree. For example, how can the nodes in a cluster agree on which node should be the elected leader?

This is a problem that has been studied a great deal. Even before IoT, distributed systems were still used and needed to have ways to reach consensus [19], [20]. While [21] shows one of the solutions directed towards specific IoT use-cases.

In [22], a leader election algorithm is proposed that is customized for dynamic systems that may have non-static topologies.

Their algorithm is quite simple. When an election is triggered a spanning tree is built between

---

the nodes, then shrunk to find the leader node. While the algorithm will assure that a leader is eventually elected, it does not guarantee that it happens immediately. This is the trade-off when applying such an algorithm on a non-static network.

## 2.5 Summary of previous work

Today the existing sheep tracking solutions primarily utilize the LTE-M or NB-IoT network for connectivity. They are able to provide sufficient battery life to last a few seasons at the pastures with a weight around 200g. Most tracking solutions include the logging of position data from sheep and sometimes includes extra features like motion detection and geofencing capabilities.

The research highlighted above shows that there exists research on the usage of combining different communication protocols within the system. Even though there are not that many articles targeted towards this area of research, based on the findings of the author.

Research on the usage of Device to Device communication is far more numerous. While their focus is directed towards reducing load on infrastructure, it shows that the usage of D2D communication is of interest.

Leader election algorithms have been researched for a long time and are specific enough that algorithms are optimized for niche use cases, like dynamic IoT environments.

As far as the author could find there were no studies that focused on the usage of the low energy communication channel as the main source of power conservation. Especially where the system has no access to mains power or where the entire network is mobile. This indicates that the work from this thesis should be of interest.

# Chapter 3

## Theory

The rest of the thesis will use a lot of terminology and technologies that may be unfamiliar. This chapter will go through the most important terminology and technologies used in the thesis.

### 3.1 Mesh and IoT terminology

The main topics of the thesis is concerning the IoT or mesh networking. Terms like things, nodes, clusters, gateways and state are used often, and needs to be defined clearly.

Firstly, a *thing*, in the Internet of Things, is a device that is connected to the cloud, either directly or via a gateway. A thing can itself be a gateway or a node reporting to a gateway. The node should be able to somehow report its state to the cloud.

The term *state* is used to describe attributes of a thing, such as its position, battery percentage, temperature, etc. The state may change with time, like the position with a tracker, and is often the main data that is produced by an IoT solution. When it is stated that a gateway "report the state" of a thing, it is meant that the gateway transmits the state of a thing to the cloud in some way.

The term node is used, in large, to refer to a *thing* in a mesh network. Nodes and things may also be used interchangeably in this thesis as things are always used with a mesh network. As with the thing, a node may or may not also be a gateway. If it is not a gateway it must report its state to a gateway, which will in turn report the state to the cloud.

A gateway is a node which is responsible for accepting and transmitting the state of nodes, including itself, to the cloud. It therefore needs some sort of uplink to the cloud, often using IP networking.

### 3.2 Mesh networks

In contrast to a traditional network, where all connected devices transmits data to the router, a mesh network may create routes between other devices to achieve connectivity.

Figure 3.2.1a illustrates a traditional network. In this network the server at the top of the illustration may also be another router, which has its own set of devices. The important observations are that there is always a single route from one device to another and no end-device relays data in the network.

Mesh networks, as shown in Figure 3.2.1b, are just a networks where communication between two nodes may be routed through any set of other nodes, before it reaches the destination node. Meaning that the general network range can be extended or multiple routes between two devices

may exist. These properties provides higher robustness, nodes may be disconnected from the network without interrupting routes. For example, in Figure 3.2.1b disconnecting either Node 2 or Node 3 will still leave a route between Node 6, 4 and 1.

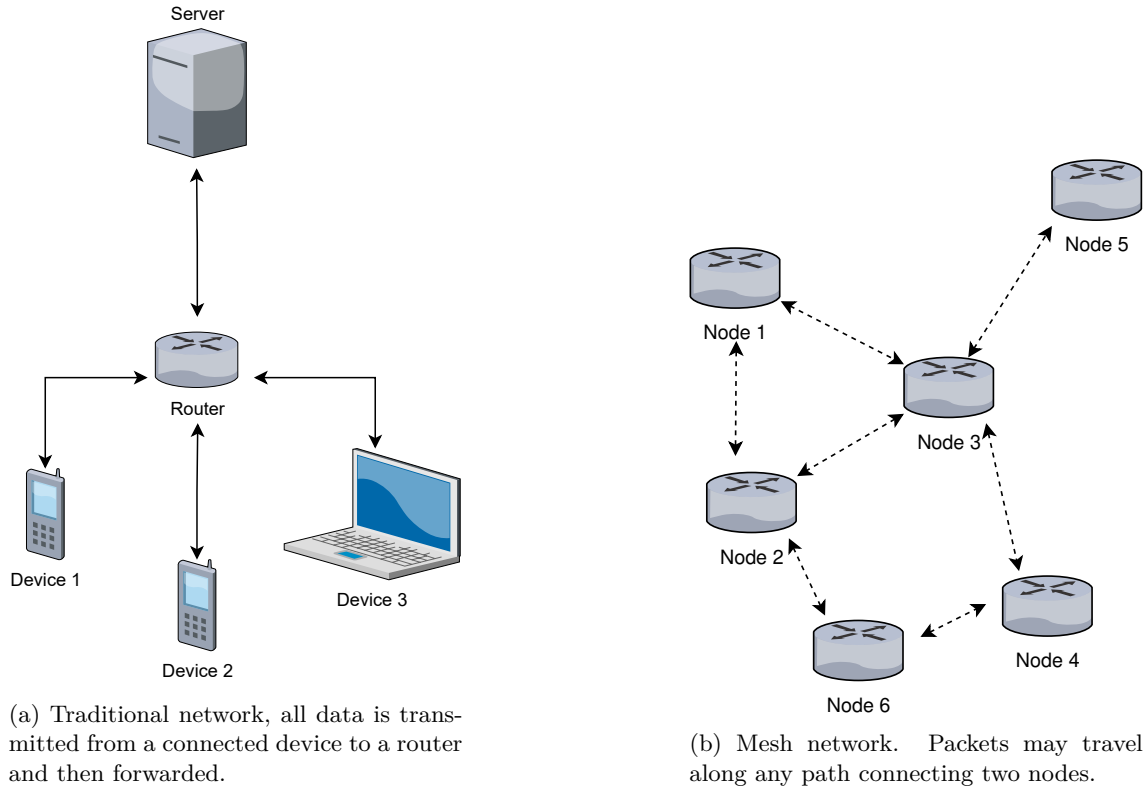


Figure 3.2.1: Comparison of traditional and mesh network

While there are cases where a traditional, non-mesh, network is used in isolation, like small local networks. Most networks are a hybrid of the two. For example if a company have many buildings they may want to connect each site to every other site. If a link between two sites goes down the other route through another site may still be up.

### 3.3 LTE-M and Power Saving Mode

LTE-M is a LPWAN technology designed for IoT, by 3rd Generation Partnership Project (3GPP) [23]. It utilizes existing cellular networks, but with features tailored towards Machine to Machine communication. One of the important features is Power Saving Mode (PSM).

PSM allows the User Equipment (UE) to request an increased time period between when it is required to contact the network and to disable paging outside of certain intervals, illustrated in Figure 3.3.1. The term User Equipment comes from the 3GPP literature like [23] and means devices that connects to the cellular network.

The network requires that a Periodic Tracking Area Update (pTAU) message is transmitted on regular intervals. The Tracking Area Update (TAU) message is used to update the registration status of the UE and in short it is required to stay connected to the network. Paging are intervals where the UE is required to enable its radio to be reachable by the network. It is desirable to avoid to disconnect from the network because the connection event is expensive in terms of energy usage.

When using PSM, the UE may request a TAU interval and a so called Active Time. The TAU interval will be the maximum period between two TAU messages and can range from hours to a



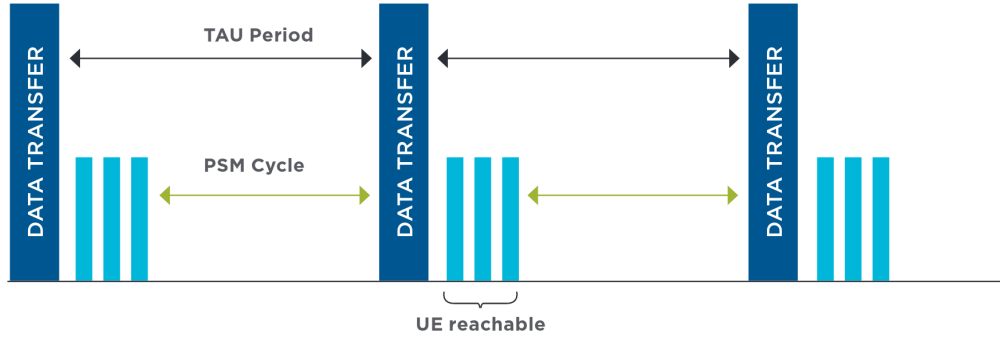


Figure 3.3.1: PSM timers. Source: [23].

year [23].

The Active Time is the time after the pTAU where the UE is reachable by the network after the regular paging interval, called Discontinuous Reception (DRX). The DRX interval is usually about 6 seconds in Norway, using Telenor as a provider [24]. This additional time can be used to receive data from the internet, but is not strictly necessary if the UE does not expect to receive any data.

When the pTAU is transmitted and the DRX and Active Time has passed, the UE may turn its radio into hibernate until the next pTAU is expected. When in hibernate the radio may draw as little as a few  $\mu\text{A}$  of current, allowing applications to conserve a lot of power.

## 3.4 Leader election algorithms

In a hierarchical system making decisions is quite simple. The node in the top of the hierarchy makes the decision and broadcasts it to its subjects. In a distributed system this process is more involved as each node are considered equal.

Different leader election algorithms exists, but they all solve the problem in different environments. A leader election algorithm is often chosen based on the assumptions that can be made about the environment that the nodes exists in. For example in a wired network consisting of routers and switches it is probably a good assumption that the system can send a lot of data and the network is quite stable. Stability in the sense that nodes rarely randomly disconnects from the network. In a wireless mesh network where nodes are dynamic these assumptions will not hold and a different algorithm will be used.

### 3.4.1 Spanning Tree Election Algorithms

One class of leader election algorithms are called spanning tree algorithms. To elect a leader each node helps creating a spanning tree. The node that initiated the election will be the root node and each other node will be leaf nodes.

When a node receives a message that an election is ongoing the node starts to forward the message to it's neighbours. Each neighbour will then do the same until all nodes are included in the spanning tree.

When a node have completed its *computation*, that is all its children in the tree have finished their computations, the node forwards the combined results of its own work and its children's work to its parent.

This means that in essence the spanning tree algorithms creates a spanning tree, reaching all the nodes in the network, then shrinks the spanning tree back to the root node.

---

When the root node receives the completed computations from all of its children it will then conclude the election, based on the results it received, and broadcast the result of the election.

The computation in this setting will often be to compare, in some way, the desirability of the nodes in the network. The result should then be the most desirable node.

### 3.5 Message Queuing Telemetry Transport protocol

Message Queuing Telemetry Transport (MQTT) is a publish-subscribe protocol where data is published to topics and forwarded to subscribers [25]. To allow such communication it, data transmission and forwarding is done via one or more *broker*. The broker accepts data published to topics and is responsible for the transmission of that data to clients subscribed to the topic, illustrated in Figure 3.5.1. The broker is also, often, responsible for authentication and authorization.

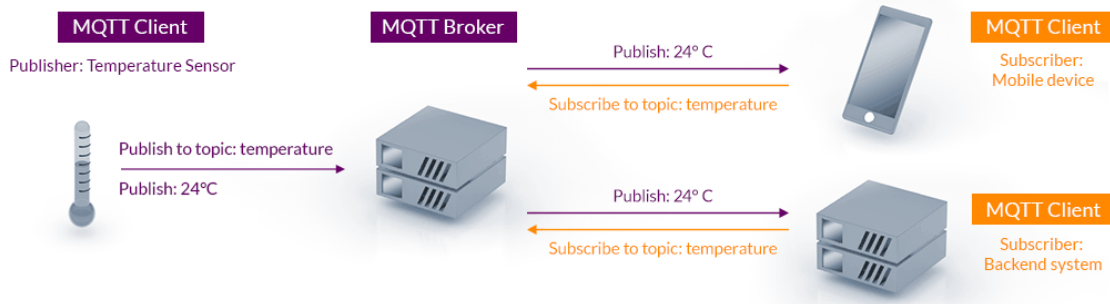


Figure 3.5.1: Illustration of the MQTT protocol, from [26]

MQTT is a good choice for IoT applications as it is light weight and allows data to be organized in topics that have a directory like structure. For example `sheep-tracker/tracker-5/position` for a dedicated topic for a single device or `alarms/water-leak` where the sender might not matter. This gives the user quite a lot of flexibility when it comes to organizing data flow in the application. A client that connects to the MQTT broker may both publish messages and subscribe to topics, which allows IoT devices to receive configuration messages and deliver status updates at the same time.

An important part of MQTT is the Quality of Service (QoS), which indicates how robust the message transfer should be. The MQTT specification specifies three levels of QoS, 0 to 2, where 0 signifies "At most once", 1 signifies "At least once" and 2 signifies "Exactly once". A higher QoS means more control over the messages, but also introduces more network overhead and latency.

## Chapter 4

# Design requirements and specification

This chapter will propose a specification that will be used to later implement the sheep tracking system. Based on the previous work and existing solutions a set of requirements for the sheep tracking system will be derived.

The requirements will be divided into two levels, first a set of *stakeholder* requirements. The stakeholder requirements will be the functionality that a sheep farmer will want from such a system. From the stakeholder requirements a set of technical requirements are derived. The technical requirements are used to have a concrete set of requirements that should be fulfilled by the end solution.

The technical requirements are subdivided into a set of hardware and software requirements. The hardware requirements are further subdivided into a set of requirements for each *role* a node can perform in the system.

The final parts of the chapter will be a test specification. The test specification will describe how the system should be tested to be able to verify that the requirements are covered.

### 4.1 Sheep tracker functionality

In extreme cases farmers can lose up to 20% of their sheep during a single grazing season. As stated in the introduction it is normal to only track the ewes in the herd. With no tracking the lambs of those ewes are also lost. A sheep tracker, as of today, will not fit a lamb as it may be too heavy for the lamb to carry. To be able to provide complete tracking of all animals the trackers will be light enough to be fitted on lambs and ewes alike.

When the sheep is first released onto the pastures they wander a lot, up to a few kilometers per hour. If a sheep tracker runs out of power in the middle of the season the farmer must either leave the sheep with no tracking or travel out onto the pastures to track down the sheep and replace the battery. Given the sizes of pastures, replacing the battery is not very feasible. This means that the battery should last for at least the full season so that no changes of batteries are required.

During the grazing season the sheep mostly take care of themselves. There is no need to continuously check out where the sheep are, but their grazing patterns may be interesting to analyse. In the case where a sheep is in danger, for example by getting stuck or attacked by a predator, it may be practical to be notified. If the sheep was killed by a predator it is often required to document the fact that the sheep was indeed killed by a predator for insurance purposes. Also a sheep that has gotten stuck can be saved if the farmer is notified in time.

On the other hand when the sheep are to be gathered again the sheep may move quite a lot and

---

a fresh position estimate may be required to efficiently collect the sheep. This means that the frequency of position updates cannot be static during the entire season.

Since the sheep move in a lot of unruly terrain the tracker is subject to a lot of stress in the form of the sheep bumping into things. The sheep are also outside all season so rain will be an issue.

The terrain may also be overhanging foliage that in turn may cause reduced coverage for positioning systems or long range transmitters. This is also true if the sheep find some sort of shelter from the rain like old stone huts, overhanging rocks or a cave.

These considerations will be the basis of our stakeholder requirements, where the stakeholder will be the sheep farmer.

## 4.2 Stakeholder requirements

The stakeholder requirements will be divided into two sets. The first is the stakeholder requirements for the farmer, which will be directed towards the requirements required for the farmer to use the solution. The second set is a set of requirements for the thesis itself. These requirements will be directed towards the usage of the mesh networks and power conservation of the network, as per the thesis description.

The problems presented in the previous section allows us to derive the following, concrete stakeholder requirements for the farmer:

**SH-1** The tracking system **must**:

**SH-1.1** present location data about every tracked sheep to the user.

**SH-1.2** allow transmission rate should be adjustable, on demand.

**SH-1.3** be able to handle sheep staying in coverage dead-spots.

**SH-2** The sheep trackers **must**:

**SH-2.1** be able to be fitted to both lambs and sheep.

**SH-2.2** have a battery life of at least one season (4 months).

**SH-2.3** be sturdy enough to survive the entire season.

**SH-2.4** be waterproof.

The following stakeholder requirements are the stakeholder requirements for the thesis itself:

**SH-3** The tracking system **must**:

**SH-3.1** utilize a low power mesh network to enable device to device communication.

**SH-3.2** attempt to conserve energy by utilizing device to device communication.

**SH-4** The sheep trackers **must**:

**SH-4.1** include a debug interface for easier debugging.

The full solution is illustrated in Figure 1.2.1, based on the requirements given above. It is important to note that the full system is not described in the thesis, some out of scope parts of the system are implemented, but not part of any discussion.

---

## 4.3 Hardware specification

The hardware requirements are subdivided into sets of requirements for each *role* a tracker can have in the system, as well as a set of requirements that all trackers must fulfil. While the requirements are divided a single tracker may implement many different roles in the system, e.g. a light tracker may be attached to both lambs and sheep and implement both gateway and tracking functionality.

**HW-1** All trackers **must**:

**HW-1.1** be battery powered.

**HW-1.2** include a debug interface for development.

**HW-1.3** must have enough battery capacity for at least one season(4 months).

**HW-1.4** include an application processor that interfaces to other, on board, systems.

**HW-1.5** include a wireless mesh transceiver for communication with other nodes.

**HW-2** A tracking node **must**:

**HW-2.1** include a GNSS receiver.

**HW-3** A gateway **must**:

**HW-3.1** include be able to communicate with the internet.

**HW-3.2** also function as a general node.

**HW-4** A lamb tracker **must**:

**HW-4.1** weigh less than 100 grams.

## 4.4 Software requirements

In the requirements two software modes are described, the gateway mode and the node mode. When the node is not functioning as a gateway it will be in node mode. In node mode the tracker will be mostly sleeping and trying to conserve power. It will read sensor data and transmit its state to an elected gateway to be transmitted to the cloud.

When a node is elected to work as a gateway the tracker will continue most of its normal node operations. The tracker will also be responsible for accepting state transmissions from other nodes in the cluster and transmitting these to the cloud in a timely matter.

**SW-1** The tracker software **must**:

**SW-1.1** be able to transition between gateway and node functionality.

**SW-1.2** be able to elect a node/nodes to act as a gateway, in a distributed manner.

**SW-1.3** be able to elect a node/nodes to do GNSS searches, in a distributed manner.

**SW-1.4** be able to elect gateways in a way that optimizes power usage.

**SW-1.5** be able to coordinate power saving features of its hardware.

**SW-1.6** be able to initiate GNSS searches and broadcast its position, if the node it is running on includes a GNSS module.

**SW-1.7** be able to recover if the current gateway is lost.

**SW-2** The tracking software, in gateway mode, **must**:

**SW-2.1** be able to report the state of trackers in its cluster to the cloud.

**SW-2.2** be able to confirm that state have been reported successfully.

---

**SW-3** The tracking software, in node mode, **must**:

**SW-3.1** be able to report its state to the elected gateway in its cluster.

**SW-3.2** be able to broadcast its gateway and positioning capabilities.

**SW-3.3** be able to retransmit messages if they are not confirmed to be transmitted to the cloud.

## 4.5 Analysis of requirements

A set of requirements are now proposed, in this section the requirements will be analysed to make sure that they are verifiable. The analysis will also hint towards how to test the requirements.

### HW-1.1

Being battery powered is a core feature of the system, being able to fit the tracker on a sheep is an obvious must. While it is not that important what kind of battery is included it must be able to provide enough current in the spikes that a cloud transmission requires. While it is not important for the results of the thesis, a battery that is simple to switch out is a nice quality of life addition. A standard connector for the battery pack will also be a nice addition to make measuring the energy usage of the tracker easier.

To verify that the device is battery powered it should be possible to start the tracker without any sort of bench power supply.

### HW-1.2

A debug interface should be present on the board, this will make development much simpler and allows for diagnostics and logging when the system is up and running. The debug interface does not add much in terms of functionality, but it will cut down development and debug time significantly.

### HW-1.3

The minimal battery is based on a single season, which is about 4 months. While it is desirable for a system to have much longer battery life, this is a good baseline for the thesis. The only real limiting factor for the battery is its weight. The weight is, in turn, limited by the amount of weight a sheep or lamb can carry. This will mean that to keep the weight as low as possible it is desirable for the tracker to consume as little current as possible.

To make sure the tracker can last a full season the average current consumption must be measured. Then, based on the capacity of the battery that is included on the tracker an estimated battery life can be modelled.

### HW-1.4

An application processor will be chosen and programmed with the software described in the software requirements. The chosen processor should be able to communicate with all the other hardware described in the requirements for the other hardware variants.

This means that the choice of application processor will depend on the choice of the other components in the system. It might be desirable to have some parts of the system build into the application processor to reduce weight or power consumption. The power consumption might be reduced if the use of external communication protocols like Universal Asynchronous Receive Transmit (UART) or Serial Peripheral Interface (SPI) is limited.

---

## HW-1.5

A wireless mesh transceiver must be included, it must support node-to-node communication. It is important that this transceiver consumes less power than the internet uplink. The transceiver should be able to be reachable at almost all times to allow uninterrupted communication between nodes.

## HW-2.1

A key part of the functionality of the tracking system is to have a way to position the tracker on the earth. A GNSS receiver will solve this issue problem. A GNSS receiver can be of any supported type, Global Positioning System (GPS), Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS), Galileo, etc. could be used. The choice of GNSS receiver will depend on where the tracker will be deployed as the coverage of the different systems will heavily depend on where the tracker is located, geographically.

For the GNSS receiver to work a proper antenna and signal conditioning circuitry is also required. A proper antenna and circuitry will be impedance matched with the receiver module to reduce signal losses as much as possible.

## HW-3.1

A gateway must be able to forward data to the cloud, this can be achieved by adding some sort of uplink to the internet. Both cellular or satellite uplinks are viable here, as long as their bandwidth is large enough to be able to forward data for other nodes. A natural choice will be an LPWAN transceiver.

To achieve high efficiency when transmitting it is important that an antenna with good performance is used to reduce signal losses as much as possible.

## HW-3.2

To function as a general node the gateway must also fulfil all requirements for a tracker node as well(HW-2.\*). This is due to the fact that a tracker will alternate on functioning as a gateway or a node to reduce power usage.

If a designated gateway is created it will still require having node capabilities as that includes communication and positioning.

## HW-4.1

A lamb cannot carry as much as a full grown sheep, so the weight must be limited. The weight proposed is based on the lightest of the trackers found in Chapter 2 and will be a good baseline for this thesis. A lower weight will, of course, be preferable.

It is important to note that this tracker weight will not include the weight of the casing. While this weight is probably included in the weights from Chapter 2, it is out of scope of the thesis.

## SW-1.1

To be able to distribute the uplink load between nodes any node that implements the gateway functionality must be able to transition between the node mode and the gateway mode. Any node that implements the gateway functionality may be elected to function as a gateway, therefore every node must be able to freely transition between the two functional modes.

---

## **SW-1.2**

There must exist a way for the nodes to elect one or more nodes to transition to the gateway mode. When a new gateway is elected the old gateway must return to normal node operation.

Electing a leader in a distributed fashion may not be trivial. The cluster may be changing and nodes may disconnect, or restart when electing a gateway. Therefore there are no hard requirements on how quickly a gateway must be elected, but it should be done within a reasonable timespan.

## **SW-1.3**

Since nodes in the cluster will be geographically close together, a position from one of the nodes in the cluster may be used to position the entire cluster.

While the software must be able to elect a GNSS node in some manner, it may be implemented by having an elected gateway choose a node to perform this task. This will imply that the cluster will choose the preferred gateway node based on some power conserving factor and then the gateway will have the authority to delegate a node to do GNSS searches.

While having a single node do GNSS searches for the entire cluster may be the most power efficient, it might be desirable to have more than one GNSS position fixes. More than one fix may allow the cluster to have much more accurate positioning based on the fact that nodes in a cluster are close together.

## **SW-1.4**

To not drain a single set of nodes very quickly the gateway election algorithm must be able to optimize power conservation in the cluster.

The implementer will be free to choose the election algorithm and the way that the nodes are weighted when electing gateways.

## **SW-1.5**

A large amount of power saving will most likely come from built-in power saving features. Ranging from scheduled GNSS searches, PSM for Long Term Evolution (LTE) to switching sleep mode on the on board power supply, the software must be able to use these features to minimize power consumption.

## **SW-1.6**

If the tracker have a GNSS module the tracker software must be able to utilize that module to acquire a position. The position will then have to be made available to the rest of the system so that it can be broadcast to the rest of the system.

## **SW-1.7**

The mesh network is dynamic and nodes will move about in the terrain. If a tracker that is currently acting as the gateway wanders of it will eventually be disconnected from the network. In this case the nodes that remain will have to be able to handle the loss of a gateway without losing data or connectivity.



---

### SW-2.1

The minimal functionality of a gateway node is to receive and forward data from other nodes to the cloud. All gateways must implement this forwarding.

The implementer may chose the cloud communication protocol that best fit their system. The only requirement is that any node may be able to transmit data on behalf of the nodes in the cluster.

### SW-2.2

A node that wants to report its state also cannot assume that the state successfully reaches the cloud when it is transmitted to the gateway. A gateway may run out of power, lost coverage, etc., to handle this the gateway should acknowledge that a report is successfully transmitted to the cloud.

### SW-3.1

The nodes in the cluster must somehow be able to transmit their state to a gateway in the cluster. The node also cannot assume that the state is reported unless it have gotten an acknowledgement.

Since the mesh network may change and links may be broken sporadically it is important that a node await acknowledgement before marking data as transmitted.

### SW-3.2

As nodes may implement different functionalities, have different batteries, etc. the nodes must be able to, somehow, advertise these capabilities to the network so they can be used in the election algorithm.

Things that might be used to calculate the desirability of a node as a gateway may include battery, signal strength, position in the cluster and many more factors.

### SW-3.3

It may happen that a gateway is lost between the time that a node sends a request to transmit data and when the data should have been transmitted to the cloud. If this where to happen and the node assumes that the data is received by the cloud the data would be lost.

## 4.5.1 Traceability matrix

To make sure that all stakeholder requirements are fulfilled by the technical specification the traceability matrix, shown in Table 4.5.1, is created. This matrix have all the stakeholder requirements on the top horizontal row. The leftmost, vertical, column holds all the technical requirements. Each marked cell in the matrix indicates that the stakeholder requirement is covered by the indicated technical requirements.

---

SH HW	1.1	1.2	1.3	2.1	2.2	2.3*	2.4*	3.1	3.2	4.1
					×					
1.1					×					
1.2										×
1.3					×					
1.4										
1.5								×	×	
2.1	×									
3.1	×	×								
3.2	×	×		×						
4.1				×						
<b>SW</b>										
1.1	×			×				×		
1.2								×		
1.3								×		
1.4				×					×	
1.5				×					×	
1.6	×									
1.7								×		
2.1	×							×		
2.2			×							
3.1			×					×		
3.2								×		
3.3			×							

Table 4.5.1: Traceability matrix between stakeholder- and technical requirements. Requirements marked with a \* is considered out of scope of the thesis.

## 4.6 Test specification

To make sure an implementation acts as the specification dictates a set of tests are required. These tests will be used to confirm that the functionality described in the requirements are fulfilled. The tests are divided into network integrity tests, energy usage tests and mechanical tests. The network integrity tests will make sure that the network is robust enough to function. They will be binary tests, i.e. they will either pass or fail. The energy usage tests will focus on measuring the energy usage of the system in different configurations. The mechanical tests will be a mix of binary tests and measurements.

All tests results will be discussed in Chapter 7. These will uncover the viability of the system in practice.

### 4.6.1 Functional tests

To ensure that the system works as expected a set of functional tests are proposed.

The following test cases must be confirmed:

- TC-1.1** Given a network with at least three nodes where one is a leader, after the term duration has passed the leader node should initiate a new election.
- TC-1.2** Given a network with three nodes where one is a leader, if the leader node is removed, the network should properly elect a new leader when the original leader is lost.
- TC-1.3** Given a network with at least two nodes, if the leader node falls in desirability, the network should elect a new, more desired, leader in the next election.

- 
- TC-1.4** Given a network with at least two nodes, if the leader has buffered messages and is demoted, the messages should be transmitted to the cloud.
- TC-1.5** If a node requests a transfer and the gateway is lost the node should retransmit any messages that have not been acknowledged.
- TC-1.6** Given a network with at least two nodes, if a new node comes within range of the existing network it should be included in the network.
- TC-1.7** Given normal operations, nodes should be able to report their states to the cloud.
- TC-1.8** Nodes must be able to report their position.

#### 4.6.2 Energy usage tests

To make sure that the proposed implementation actually conserves energy, the energy consumption of the system must be tested. Since the systems energy usage will depend upon a lot of different parameters it is important to test out the most performance critical parameters. To make sure that the system performs well with regards to the specification it is important that the transmission interval is varied in different tests. Additional tests may be specified, based on the implementation.

While there are a lot of intervals that may be tests a set of intervals that may be interesting for the system is presented in Table 4.6.1.

Transmission Interval	Reasoning
10 minutes	When collecting sheep low update interval is desired.
1 hour	A one hour interval would give quite a lot of data to analyse.
6 hours	A slow interval when few data points are required.
12 hours	Once a day for a very conservative data collection interval.

Table 4.6.1: Proposed test configurations for energy usage

As a general note any interval that is below 1 hour will be considered a quite high update rate in the IoT world. A scale in the order of hours will be more usual, and in that range it will become a question of how much battery one wants to conserve. For this system the sheep will probably report in the 1 - 24 hour interval during grazing and in the 10 - 30 minutes interval when they are gathered.

The following test cases are required:

- TC-2.1** Given a single node, the energy usage should be low enough that a single battery lasts at least one season (4 months).

#### 4.6.3 Mechanical tests

Most of the mechanical tests will be out of scope of the thesis, but should be stated anyhow.

- TC-3.1** All required components must be present on the board.
- TC-3.2** The sheep tracker hardware should be weighed.
- TC-3.3** The sheep tracker hardware should be measured.

#### 4.6.4 Test coverage matrix

It is not guaranteed that all tests will pass, so the test coverage matrix, in Table 4.6.2, will be used to map test results to requirement coverage.

TC HW	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	2.1	3.1	3.2	3.3
	1.1									×		
1.2										×		
1.3									×	×	*	
1.4										×		
1.5										×		
2.1										×		
3.1										×		
3.2										×		
4.1											×	
SW												
1.1	×		×									
1.2		×	×			×						
1.3		×	×									
1.4			×			×			×			
1.5									×			
1.6								×				
1.7		×										
2.1				×								
2.2							×	×				
3.1							×	×				
3.2			×									
3.3					×							

Table 4.6.2: Test coverage matrix, each markings indicates which tests covers which requirements.

\* the requirement is not directly covered by the test, as it will also affect the weight of the final tracker.

## Chapter 5

# Implementing, and testing, a sheep tracking system

This chapter will present the implemented sheep tracking system, based on the specification from the last chapter. First a hardware design is proposed which will be followed by the hardware implementation. Then, based on the hardware design, a software and firmware design is proposed. This will include the tracking backend, the firmware for the tracking hardware and a firmware used to compare the system against.

And the last section of the chapter will go through the concrete test cases that will be performed on the system.

The core of the solution is that it will utilize connections between the sheep trackers to save power. The usage of a LPWAN transmitter, like 4G or satellite, is much more energy costly than transmitting via the short range network. This means that if many sheep are gathered only one of the sheep trackers will actually transmit the data on behalf of all the other trackers, this tracker will be the *gateway* of the others. The other trackers, not the gateway, will not use their long range transmitters and instead send their data to the gateway using the low power network allowing them to save energy. One of the trackers that are within proximity will be the gateway and one, possibly the same tracker, will be responsible for doing a GNSS search for the position of the sheep. GNSS positioning is also a energy costly operation. The accuracy of a GNSS receiver is not better than a few meters, so the position fix of a single sheep in the crowd will be good enough to position all the other sheep as well, possibly saving large amounts of energy.

If the sheep that is currently the gateway moves out of range of the other sheep, the remaining trackers will react and elect a new gateway. The tracker of the sheep that moved out will continue to act as a gateway for itself. This will ensure that the trackers constantly have a connection to the cloud.

The non-gateway trackers may send data to the gateway out of sync with the transmission to the internet. This may mean that a gateway will hold data for another tracker for many hours before they are transmitted. To ensure that no data is lost the gateway transmits a confirmation back to the node when the data have been transmitted to the internet. In the case where the gateway is removed the tracker will attempt to retransmit the data, via the next elected gateway, until it has received a confirmation that the data have been transmitted properly.

To make sure that no single tracker will be used as a gateway all the time and drained of all its energy, the role of gateway will be rotated amongst the trackers. The tracker which will be elected as a gateway will be elected on the basis of some *desirability factor*, dependent on the implementation. In most implementations this will include the current battery capacity of the tracker, but may also include things like 4G or GNSS coverage for the trackers.

In the simplest case, if a tracker has been acting as a gateway for some time it may be demoted

because a different tracker in the network have more energy to spare, meaning that in practice the trackers will be evenly drained during the season. This mechanism may also be used to allow trackers which have faults to not become gateways. An example may be that a sheep has laid down inside a stone structure or under some heavy foliage where there is no coverage. This sheep may then report that it have no coverage and not be elected gateway. Another good example may be a sheep that has managed to somehow break its antenna or have some other fault, if the low range network is still operational it may still report its faults to the internet. This sort of solution may greatly improve system robustness and adds redundancy.

## 5.1 Hardware design

This section will outline the hardware design of the solution based on the specification from Chapter 4.

An application processor is chosen to be the central component of the design. The processor will coordinate all hardware resources and make sure that all other components are using the lowest amount of power.

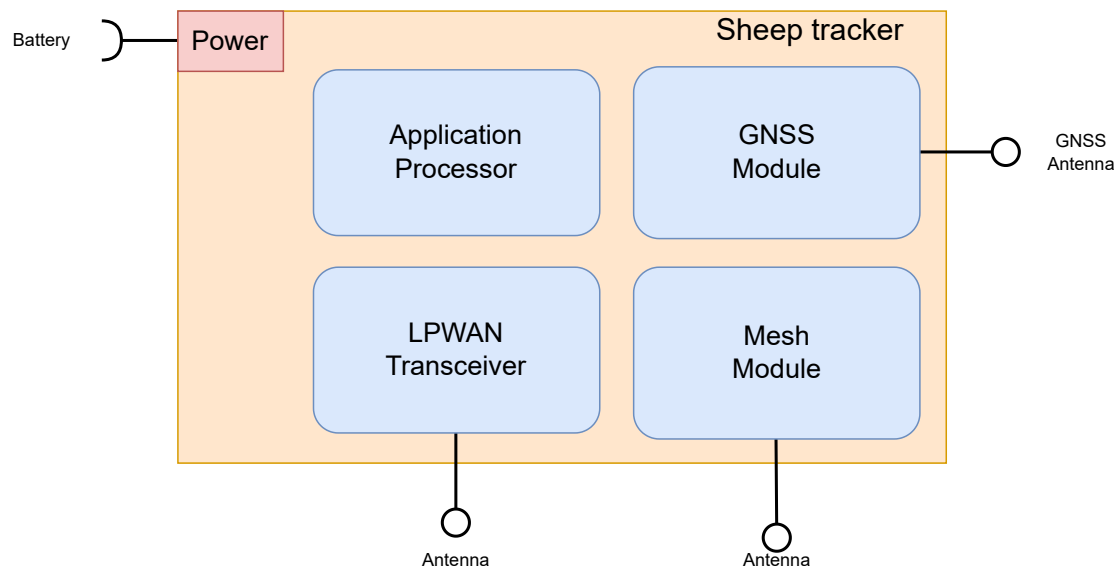


Figure 5.1.1: Hardware component diagram

A mesh unit will be connected to the application processor and should be responsible for node-to-node communication. The mesh unit can be implemented in several ways, for example a standalone unit that can be fed packets by the application processor, a simpler unit that may be used as a modem, which puts more responsible into the software on the application, or even have it built into the application processor itself. The mesh unit will also, most likely, require an antenna.

A GNSS receiver will be connected to the application processor and will also require an antenna. The GNSS receiver may also be built into the application processor or implemented as a separate component, but it should not matter much as long as it supports some way of controlling its GNSS searches and power consumption.

Some sort of long range transceiver unit will also be needed, since its purpose will be to transmit and receive from the cloud the module will be referred to as a cloud transceiver. The cloud transceiver can be built into the application processor or be its own standalone unit like most other units in the design. It needs to expose some sort of interface to transmit packets and to control its power usage to the application processor.

---

Last but not least the design will require a power management module. This module will hold the regulators, battery connectors and other circuitry required to allow low power losses when idle and handle high spikes in current draw when radios are being used. This module only requires to expose some way of determining the current battery level to the application processor.

This leads to a simple design, shown in Figure 5.1.1.

## 5.2 Sheep tracking hardware

In this section the main components of the sheep tracker will be chosen and justified.

As in [1] the main components of the tracker implemented here will be the nRF9160 and the NC-2400.

The hardware design can be represented with the simplified design, shown in Figure 5.2.1, which only includes the most important functional modules and their interfaces.

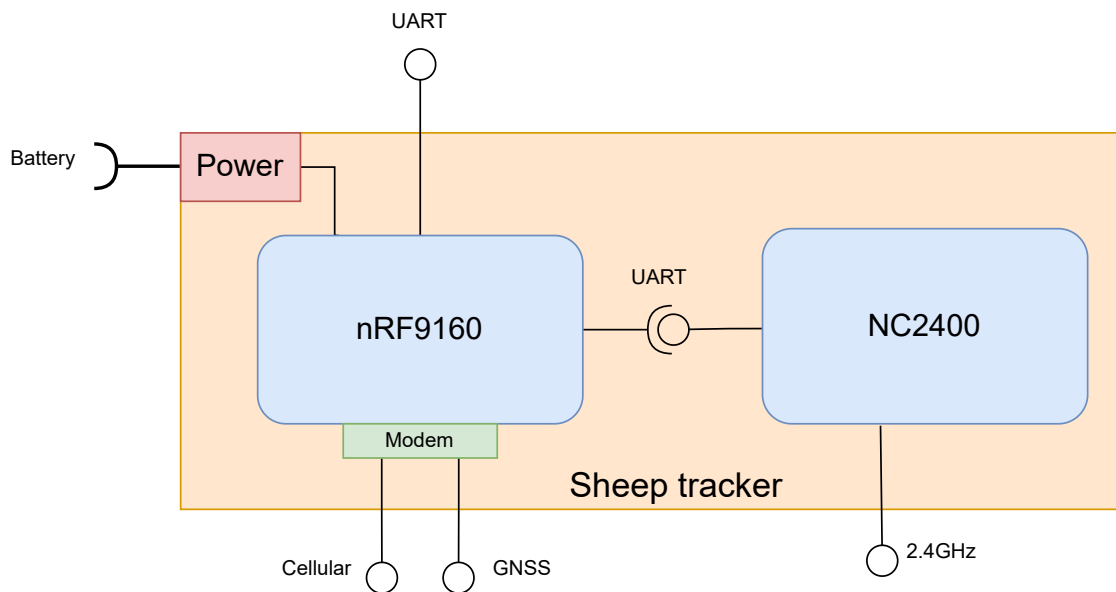


Figure 5.2.1: Simplified overview of hardware design

The nRF9160 is a System On Chip (SoC), produced by Nordic Semiconductor, it features LTE-M, NB-IoT and GNSS features. It is an ultra low power chip and can reach sleep currents in the micro amperes range. In addition to all this it is also a fairly powerful application processor.

The NC2400 is a mesh network module, produced by NeoCortec, which features a 2.4GHz transmitter and ultra low average power consumption, in the micro amperes range. The mesh protocol that comes with the NC2400 is called NeoMesh and is also produced by NeoCortec. The NC2400 also comes on a breakout board, in the FeatherWing format, as shown in Figure 5.2.2. FeatherWing is a Printed Circuit Board (PCB) format developed by Adafruit [27] as an easy way to learn electronics. The NC2400 FeatherWing module will be used as it can easily be solder onto and included in the design.

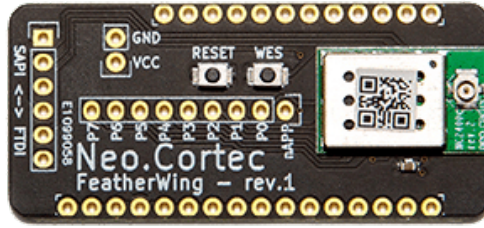


Figure 5.2.2: NC2400 breakout board

These two modules will be the core hardware of the sheep tracker. The nRF9160 is used as the application processor, GNSS module and the cloud communication module. The NC2400 is the node-to-node communication module and will be the way that each tracker communicates together.

In the start of the development process a nRF9160 Development Kit was used to develop the firmware. The development kit was an easy way of getting started with the development, but since the development kit weighs a lot more than what is strictly required, the finished code is ported onto a much smaller PCB, provided by Norbit. The PCB provided by Norbit is not designed to work with the NC2400, but it was retrofitted.

The end result is a board that will be more representative of a final product and with a representable weight. The final tracker hardware, with a NC2400, is depicted in Figure 5.2.3. This figure also shows the patch wires used to connect the NC2400 module to the tracker, as well as a programming header to upload firmware to the tracker.

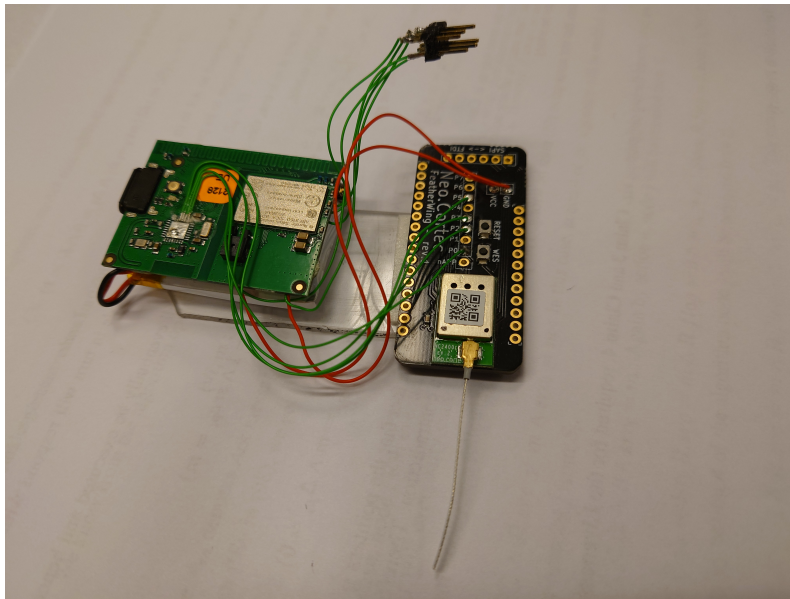


Figure 5.2.3: Image of the Norbit tracker with NC2400

The Norbit tracker PCB is also built to support a rechargeable battery which will make it great for testing battery capacity. Figure 5.2.4 shows an overview of the Norbit tracker schematic. The Bluetooth module, in the upper right corner of the figure, is soldered off and replaced by the NC2400. The Sensor module, bottom left, is an Inertial Measurement Unit (IMU), but it will not be used in this thesis.



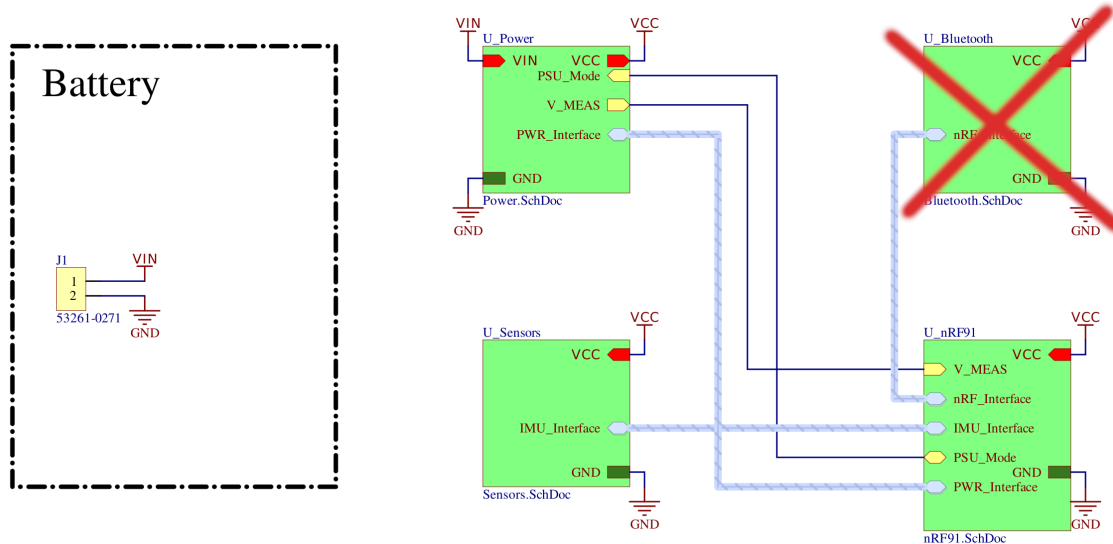


Figure 5.2.4: Norbit tracker schematic overview

The power delivery system is highlighted in Figure 5.2.5, including the battery measurement circuit.

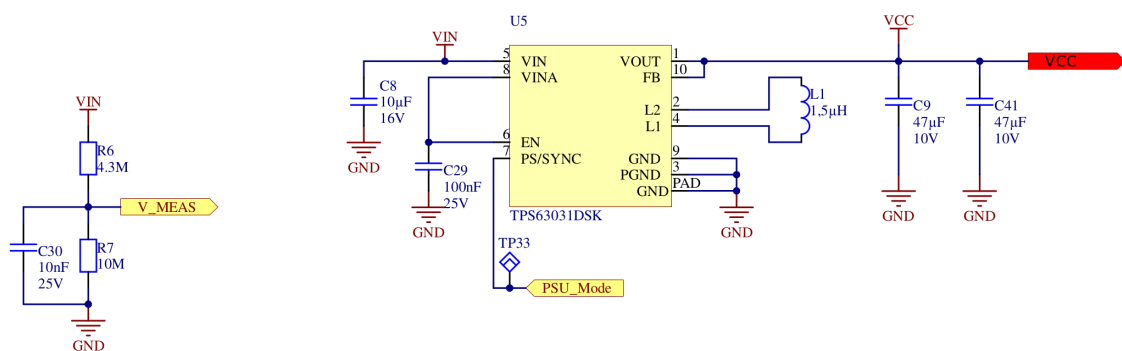


Figure 5.2.5: Norbit tracker power schematics

The schematic of the nRF9160 module is shown in Figure 5.2.6. Most of the schematic is included in Appendix A. In Figure 5.2.6, the `nRF\_Interface` is originally routed to a nRF52 (Bluetooth System on Chip (SoC)) that is included with the board. The nRF52 is soldered off on the trackers used for this project and the `nRF\_Interface` pins are connected to the NC2400 instead.

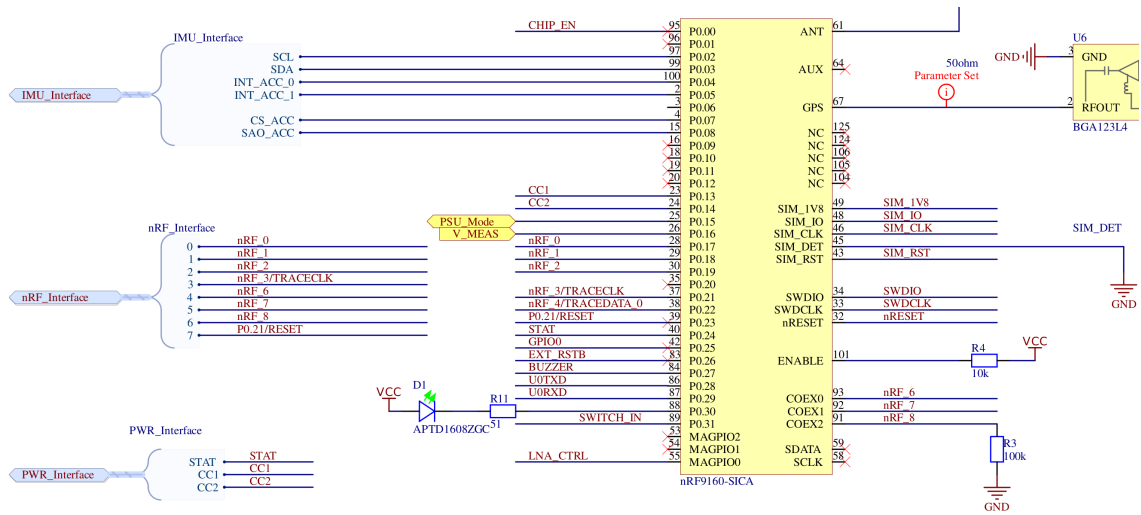


Figure 5.2.6: Norbit tracker nRF9160 schematic

## 5.3 Software design

In this section a software design is proposed on the background of the hardware implementation. The design consists of two components, the backend system and the sheep tracker firmware. Since the thesis is mostly focused on the sheep tracker, the backend system will be mentioned for completeness sake, but will not be the subject of any discussion. The full overview of the system can be seen in Figure 5.3.1.

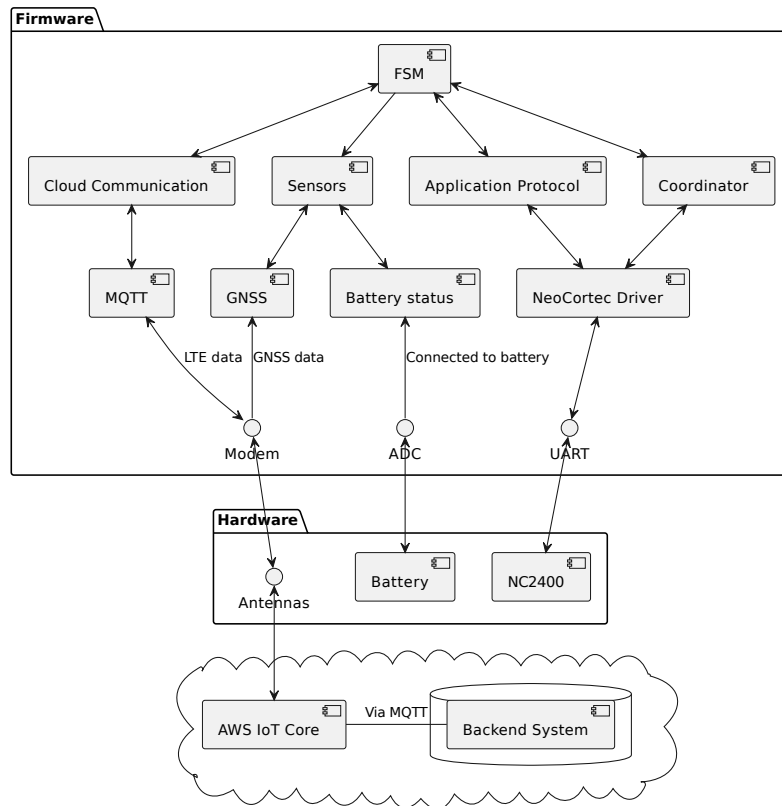


Figure 5.3.1: Overview of the software design

---

This design utilizes the MQTT protocol as a messaging protocol. Amazon Web Services (AWS) is used as the cloud provider, more specifically the AWS IoT Core system. AWS IoT Core is a MQTT broker with some AWS specific system built on top. For this implementation only the basic MQTT broker functionality is used as described in Chapter 3. The design should be simple to port to another cloud provider or an in-house solution as long as it can run a MQTT broker.

The firmware design consists of a few modules that will enables election of gateways, communication between nodes, searching for GNSS fixes and transmitting data to AWS.

To allow gateway elections, a coordinator module is needed to handle elections and keep track of the current gateway. This module will implement a leader election algorithm that will be used to elect gateways, in a distributed manner, using the NeoCortec module. For this design the algorithm from [22] is implemented, with some modifications. The algorithm was designed for unstable, ad hoc, networks like mesh networks, so it fits this use-case.

Extracting sensor data from the node is done via a sensor module. The module expose all available sensors to the rest of the firmware. The sensor data includes battery status and GNSS position, but it can be extended to incorporate other sensors like an IMU or other relevant sensors.

For cloud communication a simple module is implemented to interface with the MQTT broker in AWS. It exposes an interface that allows a node to transmit data to the broker, on behalf of another node.

This section has only covered the high level modules, but some lower level modules is also required to complete the design. A driver for the NeoCortec module, which implements the NeoCortec Application API which is accessed over a UART interface. For the cloud module to function an implementation of the MQTT protocol is used. In addition to these drivers, drivers for the nRF9160 peripherals are also implemented.

## 5.4 Tracking backend software

For the MQTT broker the AWS IoT Core service is used [28]. Using AWS IoT Core a robust and secure MQTT broker can be set up within minutes. Client certificates, authentication and authorization is also handled easily in the web interface.

The backend system is a small python service that is subscribed to all topics where the sheep trackers transmit their data. This is done with a MQTT wildcard subscription topic.

The sheep tracker clients publishes their data to topics on the format:

```
sheep-tracker/<id of a single tracker>/battery_status
sheep-tracker/<id of a single tracker>/location
sheep-tracker/<id of a single tracker>/current_gateway
```

for example, if a tracker with ID 5 reports its battery status:

```
sheep-tracker/0005/battery_status
```

The backend stays subscribed to these topics using the wildcard topic like the following:

```
sheep-tracker/+ /battery_status
sheep-tracker/+ /location
sheep-tracker/+ /current_gateway
```

The `paho-mqtt` python package is used as the MQTT client and the `psycpg2` package for the database interface. As data comes in it is be pushed into a database to keep a history of tracker states.

timestamp	serial_number	tracker_id	gateway	remaining_battery	latitude	longitude	accuracy
2022-05-04 14:38:55+02	31730	5	5				
2022-05-04 14:38:52+02	31730	5		4.018303955078125			
2022-05-04 14:16:20+02	40707	8	8				
2022-05-04 14:16:15+02	40707	8		3.71228369140625			
2022-05-04 13:50:16+02	24223	6	5	3.998283935546875			
2022-05-04 13:47:03+02	52192	7	5	3.955384033203125			
2022-05-04 13:38:52+02	3446	5		4.019734130859375			
2022-05-04 13:16:15.684983+02	41274	8		3.71228369140625			
2022-05-04 13:16:15+02	41274	8	8				
2022-05-04 13:14:34.99956+02	20307	8		3.719433837890625			
2022-05-04 13:14:33+02	20307	8	8				
2022-05-04 13:10:16.249741+02	59467	8		3.719433837890625			
2022-05-04 13:10:15+02	59467	8	8				
2022-05-04 13:09:33.849004+02	56938	8		3.715143798828125			
2022-05-04 13:09:33+02	56938	8	8				
2022-05-04 12:50:16+02	44713	6	5	3.999714111328125			
2022-05-04 12:47:03+02	3759	7	5	3.955384033203125			
2022-05-04 12:38:55+02	10795	5	5				
2022-05-04 12:38:52+02	10795	5		4.0168740234375			
2022-05-04 12:24:51+02	15178	8	8				
2022-05-04 12:24:46+02	15178	8		3.718003662109375			
2022-05-04 11:54:13.107894+02	26318	6		3.999714111328125			
2022-05-04 11:54:07+02	26318	6	6				
2022-05-04 11:51:22.634076+02	27639	7		3.955384033203125			
2022-05-04 11:51:05+02	27639	7	7				
2022-05-04 11:50:25.350222+02	51176	6		3.999714111328125			
2022-05-04 11:50:19+02	51176	6	6				

(27 rows)

Figure 5.4.1: Screenshot of database entries

The data is put into a PostgreSQL database as shown in Figure 5.4.1. PostgreSQL was chosen based on the authors familiarity with it, but any database will suffice.

## 5.5 Tracking firmware

With the hardware implementation and backend system described the only part left is the tracker firmware. This section will present all software modules that is implemented on the tracker. These modules include: A driver for the NC2400, an MQTT client implementation, a GNSS driver, sensor drivers, a coordinator module to handles leader elections, an application layer module and a Finite State Machine (FSM) to coordinate the other modules. An overview of the software is shown in Figure 5.5.1.

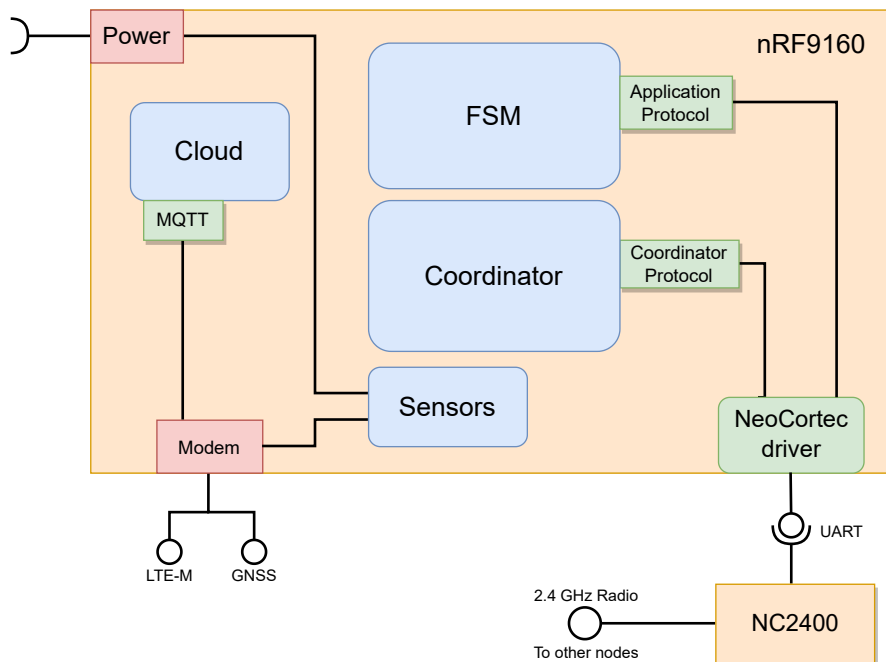


Figure 5.5.1: Overview of the sheep tracker firmware

---

### 5.5.1 NC2400 driver

The NC2400 driver will be a UART based driver that will implement the NeoMesh protocol as described in [29], [30] and [31]. The low level UART driver is provided by the nRF Connect Software Development Kit (SDK), which is Nordic Semiconductor's SDK for the nRF9160. The NC2400 driver is asynchronous and handle UART events in reaction to the Clear To Send (CTS) and Wake Up signals from the NC2400 module. Figure 5.5.2 shows the contents of the driver.

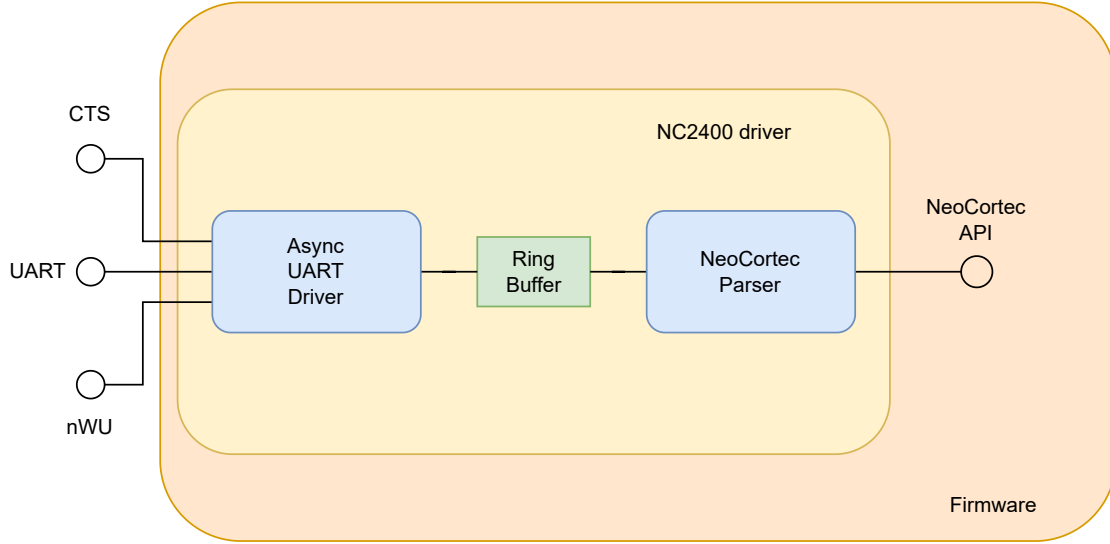


Figure 5.5.2: NC2400 driver illustration

When the nRF9160 is not receiving, nor have data to transfer, the UART peripheral will be powered down to save energy. The NC2400 have two configuration parameters, the *AAPI Wakeup Time*[30, p. 27] and *AAPI CTS Timeout*[30, p. 26], which will make this possible. These parameters give the nRF9160 some time to power on the UART peripheral without missing any UART data sent by the NC2400 and prevents the NC2400 from timing out when it is ready to receive.

The nRF9160 will be awoken by an interrupt when the NC2400 asserts one of its CTS or Wakeup pins. If the CTS pin is asserted the nRF9160 will start to transmit any commands over UART and if the Wakeup pin is asserted a buffer will be prepared for the incoming message. Any received data over this UART interface will be pushed into a ring buffer and parsed in a different thread.

When a valid message is parsed the parsing thread will send an event to a user defined callback allowing the message to be appropriately handled by the application.

The NeoMesh protocol supports ports, so on the application level one port is set to send application messages and one port is used for coordinator messages.

### 5.5.2 Coordinator module

The coordinator module implements the algorithm from [22] with some modifications. It runs a state machine internally that accepts coordinator protocol messages, desirability updates and neighbour list changes. The coordinator then calls a user provided callback if it requires to send messages or if a demotion or promotion event happens.

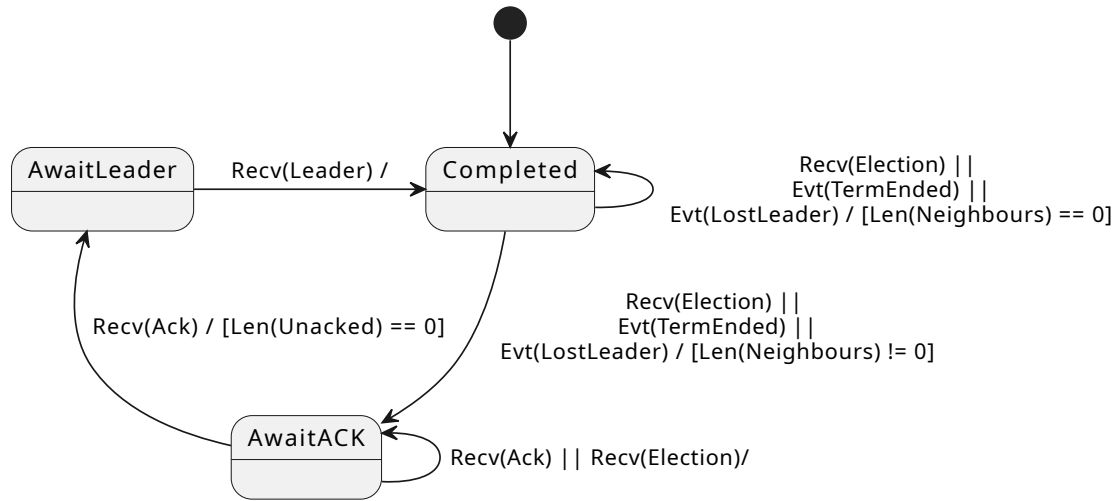


Figure 5.5.3: State diagram of the coordinator module

The state diagram, in Figure 5.5.3, shows the state of the coordinator module. While it does hide some of the nuances of the implementation it gives a nice overview. When a tracker is first booted up it immediately elects itself as a gateway and is taken into the Completed state of the algorithm. If no new neighbours appear the tracker will act as a gateway forever, but on a preconfigured interval the coordinator will still initiate a new election, but will immediately win the election as no neighbours exist.

To communicate with the coordinator module on other nodes messages will be sent over the NeoMesh network. These messages consists of a header, described in Figure 5.5.4 and a payload. The payload is dependent on the type of message sent, the messages that exists are Election, Ack, Leader, Probe and a Reply message. They are all shown in Figure 5.5.5.

The first 6 bytes of the messages in Figure 5.5.5 include an initializer and index field. These fields are the computation index, described in [22], and are used to keep a definite ordering of elections. It consists of an initializer which is the ID of the node that starts the election and an index, simply a counter that is increased for every new election. The ordering is used to keep track of concurrent elections and for the tracker to decide whether or not to join a new election. When a node joins an election it means that it stores the computation index from the Election message as its currently active computation index.

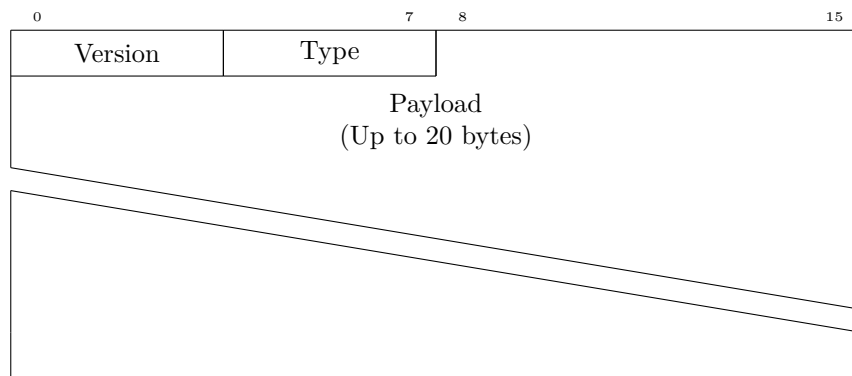


Figure 5.5.4: Coordinator protocol header. The header indicates the version of the protocol and the type of message. The data in the message is then dependent on the message type. The payload is limited to 20 bytes as 21 bytes is the maximum size of a NeoCortec payload

Both the Ack (Figure 5.5.5b) message and the Reply (Figure 5.5.5d) message contains a single byte field. These fields are simply used for flags, there was no need to optimize the protocol size

so a full byte was good enough. The Ack flag indicates whether or not the sender of the message have received all its Acks and has completed its computation. The Flag field of the Reply message is used to both indicate if a final Ack have already been transmitted and if the node is currently in an election. The fields in the Reply message is used to prevent deadlocks if messages are lost in the NeoMesh network.

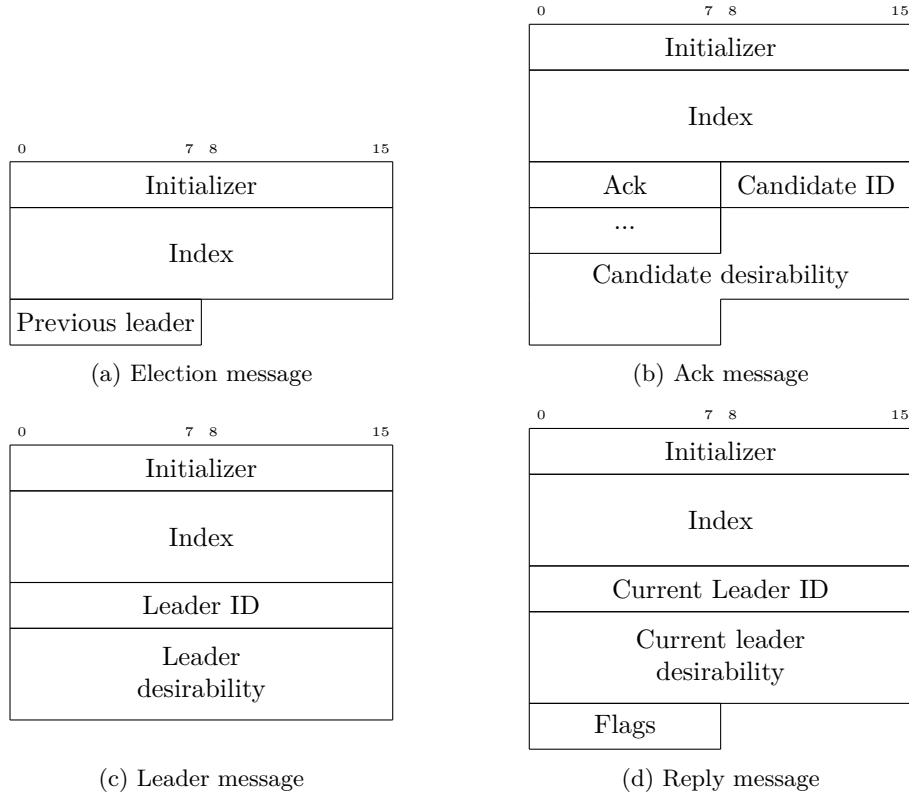


Figure 5.5.5: Overview of coordinator protocol messages. The Probe message has no payload, only a header.

If a new neighbour is introduced the two nodes will exchange Leader messages (Figure 5.5.5c). The Leader message holds information about the last election the node was a part of as well as its current leader. The tracker that have the leader with the lowest desirability adopts the newly introduced leader as its own, and forwards the Leader message to its neighbours, excluding the tracker that sent the Leader message. This also describes the process if two full networks meet and then, subsequently, merges. Leader messages will be exchanged and the most desirable leader will be adopted and spread in the network.

When a network have been stable for some time the current gateway will receive a TermEnded event, as seen in Figure 5.5.3, meaning it is time for a new election. The leader increases its computation index, then transmits an Election message to its neighbours and awaits Ack messages from them before concluding the election. This is how the spanning tree is built, nodes that receive Election messages will forward them to their neighbours. Excluding their *parent* node, i.e. the node that transmitted the original Election message. The nodes further down the tree will in turn await Acks from all their neighbours before doing any further work.

In the case where a node receives an election message, belonging to the election it is currently a part of, the node immediately responds with an Ack. This means that no deadlocks will occur where nodes await each other.

When a node either; has no neighbours other than its parent or has received all its Acks, it will proceed with the election process. The node will compare the leader candidates contained in the received Ack messages, and thereby keep track of the best candidate as Acks are received. When the node no longer awaits any Acks it can forward the best candidate, also having considered itself

---

as a candidate, to its parent node. This process is the *shrinking* of the spanning tree.

These processes will continue on all nodes until every node have reported their most desired leader candidate to their parent. Eventually the node that started the election will receive its final Ack. This node will, as all other nodes, compare all the most desired leader candidates and finally sit with the most desirable candidate in the network. The node that initiated the election will then transmit a Leader message to all its neighbours with the ID and desirability of its leader.

At this point every node in the network should be in the AwaitLeader state and when a Leader message is received they will adopt this new leader and then forward the message to their neighbours, concluding the election.

If at any point the leader or parent of a node is disconnected or travels out of range, the node that detects the event will start a new election, or accept the role of parent (not gateway!). To make sure lost messages do not cause deadlocks, a Probe and Reply system have been implemented. A tracker will periodically send a Probe message to neighbours who have yet to send their Ack message and they will respond with a Reply message. This Reply message holds the nodes status in the election and the probing node may remove the neighbour from their list of unacknowledged neighbours and proceed with the election.

The more difficult edge cases to handle is the one where more than one election is started simultaneously. Every message in the coordinator protocol includes a *computation index*, except the probe message, also described in [22]. This index keeps track of different elections and nodes will either join higher priority elections or discard lower priority messages.

If a node disconnects from the network it will be handled as a loss of the current leader. Since no neighbours are present it immediately wins the election and becomes its own leader.

In the case where a node crashes or restarts the software will initialize the node as its own leader and will quickly join the old network and the joining will be handled as the merging of two networks.

### 5.5.3 Application layer protocol

In addition to the coordinator module, the main application will also need to communicate with other trackers in the network. To allow this data transfer a secondary protocol is created for application data. It is a simple protocol that can serialize and de-serialize GNSS data, sensor data and neighbour lists to be sent via NeoMesh.

Sensor data transmitted over this protocol is also called transmission requests as they are transmitted to the current leader to be forwarded to the cloud.

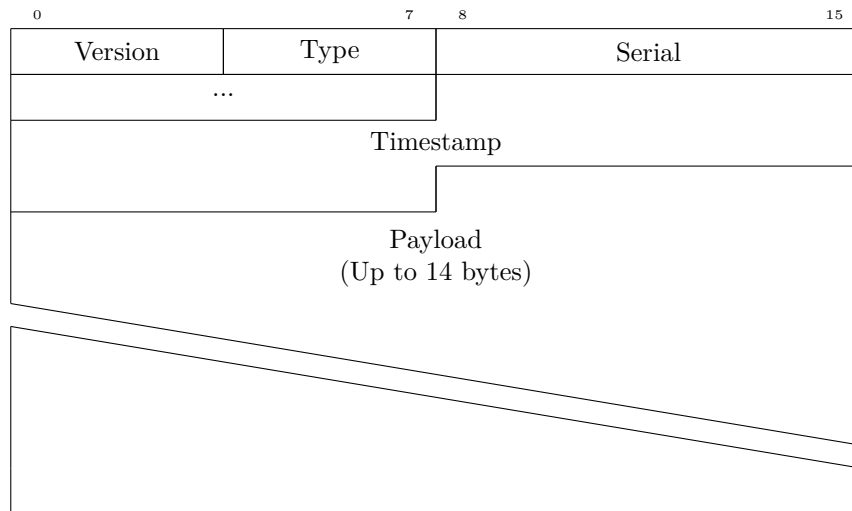


Figure 5.5.6: Application protocol header



---

All messages includes a serial number and a timestamp, as seen in Figure 5.5.6. The serial number is to identify a given message from a given node, the serial is also used by the transmitting node to keep track of which messages are successfully retransmitted or not. The gateway may end up receiving many messages from the same node and the serial number is used to which messages are successfully transmitted to avoid data loss. While the timestamp could be used for this purpose the module may have more than one sensor sampled at the same time, which means they are not the same message. The timestamp is included to be able to correctly timestamp messages by when they were sent or sampled. To reduce the size of the application protocol messages the timestamp is truncated to 32 bits, with one second resolution. This means that there needs to be some sort of additional processing when 64 bit timestamps end up being required.

#### 5.5.4 Sensor module

To gather sensor data, among it data that will be used for desirability, a generic sensor acquisition module will be implemented. It will be a simple module that will gather data from various sources and store them for other modules to use.

It will, in the first place, encapsulate battery and GNSS measurements, but could also incorporate IMU or other sensor measurements that may be relevant.

The module should be polled to start a sensor gathering. It should then power up and sample required sensors, power them down, then return the sampled sensor data.

Since GNSS will be a part of this module it will require some special treatment as GNSS sampling may take quite a long time. This was handled by implementing a polling API, where the user returns an error message saying that the measurement must be taken again.

This module also includes the battery module, which used the Analog Digital Converter (ADC) of the nRF9160 to sample the battery voltage. It also handles transforming the raw sample into a correct voltage.

#### 5.5.5 Cloud module

For MQTT communication a slightly more generic cloud communication module is implemented. The module will encapsulate all functionality regarding publishing data to a cloud solution.

The module will be a "simple" wrapper around the MQTT implementation from Zephyr/nRF Connect SDK, and will expose methods to transmit sensor measurements on behalf of different nodes. To make sure that data is not lost the cloud module will use a QoS level of 1, meaning that all state updates should be delivered "At least once", as described in Chapter 3.

The module contains a buffer of messages that have been received, but not yet sent. If the buffer is close to full any calls to the API will return a message to the user indicating that a transmission is required.

When a transfer is initiated, the module returns a list of all successfully transmitted messages. This list may be used to transmit Ack messages to the nodes that had requested the transfers.

#### 5.5.6 State machine

To bind all the modules together a state machine is implemented to handle transitions between node and gateway mode.

The overview of the state changes and program flow can be seen in Figures 5.5.7 to 5.5.9. This diagram does not, however, include the leader election process. This process will run in the background, continuously, and relevant messages will be forwarded to it for processing, as stated earlier.

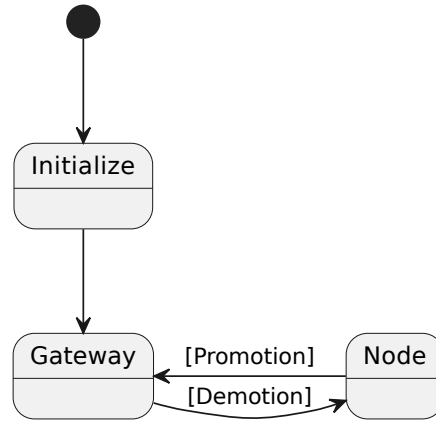


Figure 5.5.7: Overview of the tracker state machine

When the tracker boots up it goes into a initialization phase, where hardware and drivers are initialized. After the tracker have initialized all its module it transitions directly into the gateway mode.

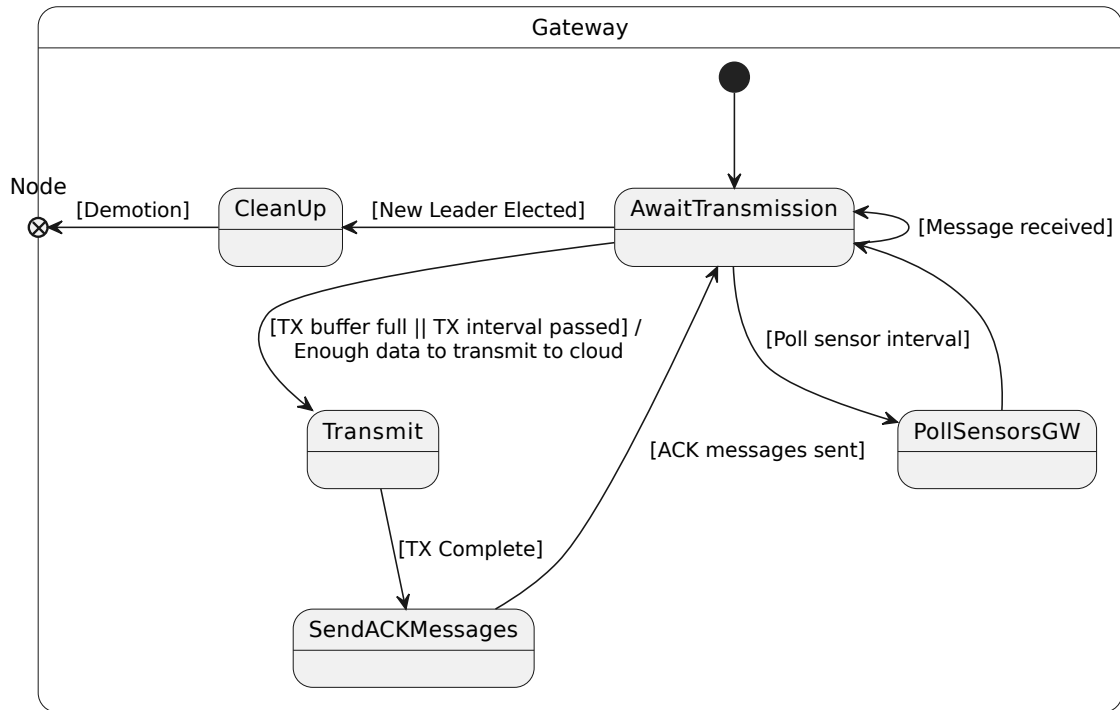


Figure 5.5.8: State diagram of the Gateway state

In gateway mode the tracker accepts messages from other nodes and forwards them to the cloud module. As stated in the cloud module section, if this buffer is close to filling up, a transmission will be started to ensure that no messages are lost because due to buffer overflow.

If a normal amount of messages are received, the node will sleep for a given time and then initiate the transmission regardless of how many messages it has received. When transmitting data on behalf of another node the gateway injects its GNSS data and its own ID in the data transmitted. This is how a gateway transmits GNSS positions for its nodes. The gateway ID is used to be able to track which tracker is acting as a gateway for which nodes.

When the transmission is finished the gateway sends an ACK message via the mesh network to

the node who requested the transmission. That node may discard the message, with the matching serial number, as it has successfully transmitted.

The PollSensorsGW state is equivalent to the one that will be performed in node mode, except that the gateway also searches for a GNSS fix as part of its sensor polling.

When the coordinator module reports that the tracker have been demoted the tracker transmits all its remaining cloud messages, transmits ACKs to the required nodes, then transitions into node mode.

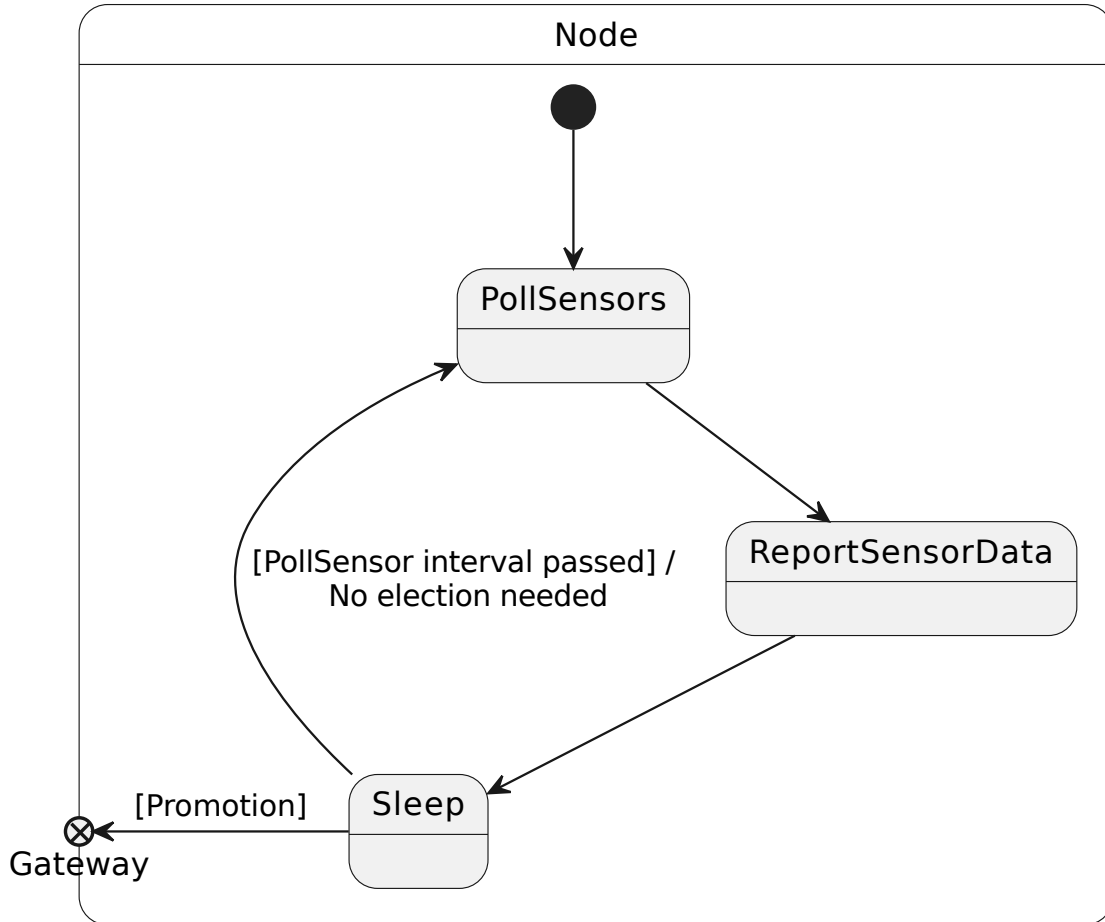


Figure 5.5.9: State diagram of the Node state

The node mode is less involved than gateway mode. The node sleeps for a given interval, unless promoted. When the sensor poll interval passes it polls its sensors and reports that data to the current gateway and go back to sleep. The node will also wake for a brief moment if an ACK message is received from the gateway, this is when the transmitted message should be discarded, but this feature is not implemented due to a lack of time.

## 5.6 Firmware image for performance comparison

Many of the solutions already on the market already use the NB-IoT or LTE-M technologies. To have some sort of comparison between the solution proposed in this thesis and the competition a simple application without any mesh technology attached is developed. This application will run on the Norbit Tracker hardware, but no modifications, except removing a Bluetooth chip from the board, will be done. The software will utilize the sensor module from Section 5.5 to keep the two

---

applications as similar as possible.

This application will boot up, initialize its modules, poll sensors, including GNSS, transmit over MQTT to the cloud. Then it will sleep for the same intervals that the gateways are programmed to sleep in the other examples.

This application will be referred to as the baseline application or software as it represent a baseline for comparison.

## 5.7 Requirement coverage

In this last section the system will be reviewed to make sure that it actually covers the requirements presented in Chapter 4.

### HW-1.1

As seen in the Norbit tracker schematics, Figure 5.2.4, the tracker includes a battery connector. The choice of using the nRF9160 also allows the tracker to use very little power and be fully battery operated.

### HW-1.2

A UART line is available on the tracker for debugging. As seen in Figure 5.2.6, J6 and J7 are pins used for a UART debug interface.

### HW-1.3

To make sure this requirement is fully covered a test must be conducted, measuring the energy usage of a completed tracker and calculating the appropriate size of a battery.

### HW-1.4

The nRF9160 is chosen as the application processor and covers this requirement.

### HW-1.5

The NC2400 module is included in the design. Connected to the nRF9160 via the UART interface.

### HW-2.1

The nRF9160 includes a GNSS receiver and the tracker have a GNSS antenna included in the design, impedance matched to the nRF9160.

### HW-3.1

The nRF9160 comes with both NB-IoT and LTE-M functionalities and the tracker hardware comes with an antenna for these bands.

---

## **HW-3.2**

We only use one node type that fulfils both gateway and general node requirements.

## **HW-4.1**

The tracker weighs 18 grams on its own, including the full FeatherWing NC2400 breakout board and some plexiglass for mounting. The battery used for testing is a 520 mAh battery, weighed at 10 grams. Making the total of the tracker with the battery 28 grams.

## **SW-1.1**

This is handled by the coordinator node and the state machine.

## **SW-1.2**

The coordinator node handles the leader election algorithm and will be able to elect a leader node.

## **SW-1.3**

Since the gateway can act as a leader in the network it will elect a GNSS node, since the cluster agrees on the leader, they will also agree on the GNSS node.

## **SW-1.4**

The gateway is chosen by the battery voltage, which leads to the tracker with the most energy remaining being elected as a gateway. This will balance the energy cost of transmissions between all nodes, optimizing power draw in the sense that energy is evenly drained from the system.

## **SW-1.5**

The Zephyr Real Time Operating System (RTOS), that runs on the nRF9160 platform builds in a lot of hardware power saving features. The firmware will use PSM mode and periodical interleaving GNSS searches to reduce power usage by as much as possible.

The UART driver will also handle powering on and off the UART peripheral to save even more power.

## **SW-1.6**

The GNSS search will be handled by the GPS driver provided by Nordic Semiconductor, as part of their nRF9160 SDK. This position will then be broadcast over the network, using the NC2400 module.

## **SW-1.7**

When a gateway is lost one of its neighbour nodes will detect the node missing from the network and initiate a new election, recovering the network.

---

## **SW-2.1**

The cloud module will use the MQTT library provided by the nRF9160 SDK that will handle transmissions.

## **SW-2.2**

When a transmission have been successful an ACK message will be sent to each node that have submitted data to be transmitted.

## **SW-3.1**

The trackers will use the coordinator module to see if they are currently an elected gateway, if not they will transmit the data via the coordinator. If they are the elected gateway they will submit the message to the cloud module and await an ACK message.

## **SW-3.2**

Each node is responsible for keeping track of its desirability factor. This factor will be utilized in the leader election algorithm. The factor itself will include remaining battery capacity, if the node implements gateway functionalities or not, which will always be true in this thesis.

## **SW-3.3**

This feature was never implemented and the requirement is not properly covered.

# **5.8 Testing the implementation**

To make sure that the proposed implementation follows the specification and fulfils the requirements this section will outline a set of test procedures. The test procedures will be based on the specification from Chapter 4 and should adhere to the tests described in that chapter.

Many of the tests will make use of a Power Profiler Kit 2 (PPK2) from Nordic Semiconductor. The Power Profiler Kit (PPK) can be used as a source meter, sourcing current at a given voltage or a ampere meter. For this thesis the source meter will be used.

To interface with the Power Profiler Kit, Nordic Semiconductor provides an application, the Power Profiler, as part of their nRF Connect Desktop bundle.

In this application the PPK is configured and samples are visualized, shown in Figure 5.8.1.

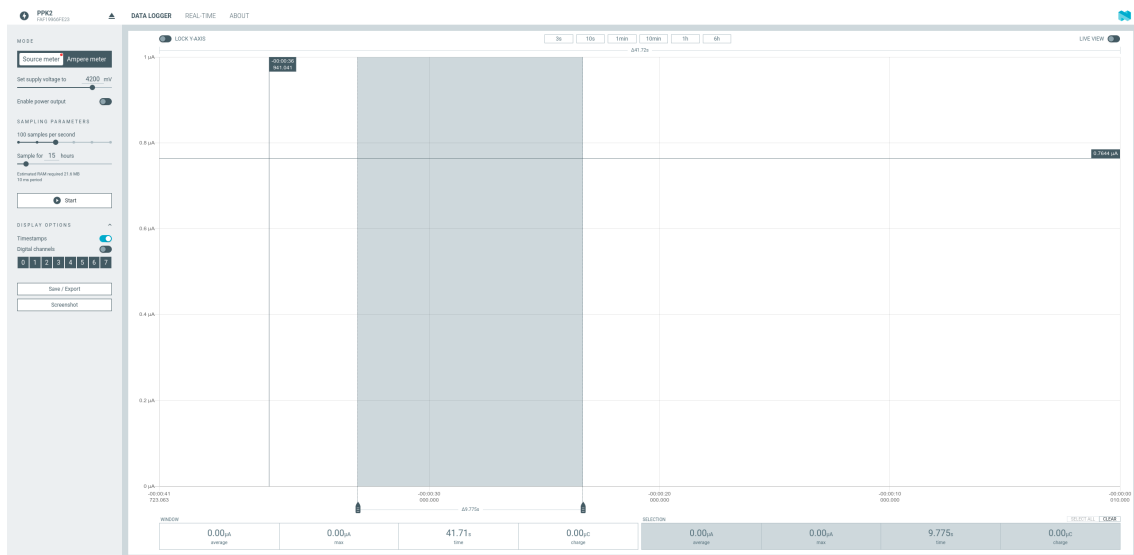


Figure 5.8.1: Power Profiler application

The PPK is configured by setting sample rates, output voltage and sample duration, shown in more detail in Figure 5.8.2.

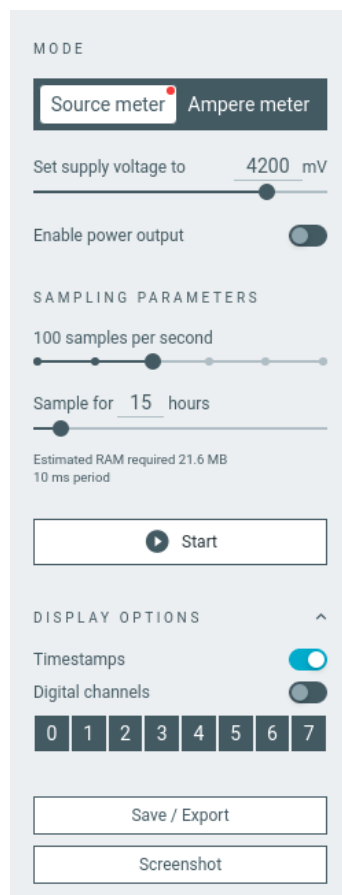


Figure 5.8.2: Power Profiler configuration menu

When the sampling have started the application shows the statistics of the sampled data either by the window in view or a selection made as shown in Figures 5.8.3 and 5.8.4.

---

WINDOW			
0.00 $\mu$ A average	0.00 $\mu$ A max	41.71s time	0.00 $\mu$ C charge

Figure 5.8.3: Power Profiler statistics of current window

SELECTION				SELECT ALL CLEAR	
0.00 $\mu$ A average	0.00 $\mu$ A max	9.775s time	0.00 $\mu$ C charge		

Figure 5.8.4: Power Profiler statistics of selection

### 5.8.1 Functional tests

A set of functional tests are described in Table 5.8.1. The testing of most of the cases are quite simple in a controlled environment, using the PPK to manipulate the "battery" voltage of a tracker. In addition to the tests using the PPK to manipulate the nodes, a field-test will be performed where the tracker is placed in an open area, close together, then moved to verify functionality in a more realistic setting.



---

Test case	Procedure
TC-1.1	<i>Given a network with at least three nodes where one is a leader, after the term duration has passed the leader node should initiate a new election:</i> Enable debugging of the trackers, wait for the term timer to expire, observe that a new election is initiated.
TC-1.2	<i>Given a network with three nodes where one is a leader, if the leader node is removed, the network should properly elect a new leader when the original leader is lost:</i> Have three nodes form a network, use PPK to make a given node the gateway, power off the PPK, observe that another node accepts the gateway role.
TC-1.3	<i>Given a network with at least two nodes, if the leader node falls in desirability, the network should elect a new, more desired, leader in the next election:</i> Have three nodes form a network, use PPK to make a given node the gateway, reduce PPK voltage lower than other nodes, await election and observe that another node accepts gateway role.
TC-1.4	<i>Given a network with at least two nodes, if the leader has buffered messages and is demoted, the messages should be transmitted to the cloud:</i> Have three nodes form a network, with shorter election intervals than transmission intervals, one node with debug enabled, force that node to become leader, observe that it receives a message from another node, reduce its voltage below the other nodes and await election. When an election happens, observe that the received messages are sent to backend.
TC-1.5	<i>If a node requests a transfer and the gateway is lost the node should retransmit any messages that have not been acknowledged:</i> Feature was never implemented, no test was performed.
TC-1.6	<i>Given a network with at least two nodes, if a new node comes within range of the existing network it should be included in the network:</i> Have two nodes form a network and a third node form its own network. Move the single node into the range of the other two and observe that the three nodes creates a new network.
TC-1.7	<i>Given normal operations, nodes should be able to report their states to the cloud:</i> Given three nodes running as expected, allow the transmission and poll interval to pass, observe that data has reached the database.
TC-1.8	<i>Nodes must be able to report their position:</i> Allow three trackers to run for an extended time and observe that position data has reached the database.

Table 5.8.1: Test procedure for network integrity tests. The test specification is given in italics.

## 5.8.2 Mechanical

For mechanical testing the test procedures are described in Table 5.8.2.

---

Test case	Procedure
TC-3.1	Visually inspect board and observe all required components.
TC-3.2	Use a scale that has one 1 gram accuracy. Place tracker on top, including NC2400 breakout board, battery and plexiglass. Read out the weight.
TC-3.3	Use calipers to measure the dimensions of the tracker solution.

Table 5.8.2: Test procedure for mechanical tests

### 5.8.3 Energy usage

The energy usage tests that will be performed will be performed on two tracker configurations. One configuration where the transmission- and polling intervals are 10 minutes and one where they are 1 hour. Three sets of measurements are taken from each configuration to cover all the two modes of the tracker firmware and one taken of the baseline firmware for comparison.

To have some indication of what current usage to expect the Nordic Online Power Profiler can be used to obtain an estimate [24]. In Figure 5.8.5a the Profiler is configured for a 10 minute "transmission" interval, and the expected average current is 1.28 mA. The profiler tool is actually expecting the PSM sleep duration, but it is only used for estimates, so it should not matter. While in Figure 5.8.5b the 1 hour transmission interval estimate is shown, and the expected average current is 214.91  $\mu$ A. It is important to also note that these estimates do not include any other energy usage than the nRF9160 chip itself, meaning that they are probably a bit lower than what can be expected.

<b>General</b>			
Chip	nRF9160 rev2		
Voltage	3.7		
<b>PSM</b>			
Periodic TAU	10.0 min		
Periodic TAU timer element	"10010100"		
Active time timer element	"00000101"		
<b>RRC connected mode</b>			
cDRX average current	5.87 mA		
cDRX charge	28.19 mC		
Connection management charge	24.81 mC		
Total charge	24.81 mC		
<b>RRC idle mode</b>			
Time in iDRX	10.0 s		
iDRX average current	1.5 mA		
Total charge	14.39 mC		
<b>GPS</b>			
GPS average current	1.12 mA		
GPS fix total charge	673.5 mC		
		<b>Current consumption</b>	
		LTE event total charge	90.48 mC
		PSM floor current	2.7 $\mu$ A
		Total average current	1.28 mA

(a) Online Power Profiler configured for 10 minute transmission intervals

<b>General</b>			
Chip	nRF9160 rev2		
Voltage	3.7		
<b>PSM</b>			
Periodic TAU	60.0 min		
Periodic TAU timer element	"00000110"		
Active time timer element	"00000101"		
<b>RRC connected mode</b>			
cDRX average current	5.87 mA		
cDRX charge	28.19 mC		
Connection management charge	24.81 mC		
Total charge	24.81 mC		
<b>RRC idle mode</b>			
Time in iDRX	10.0 s		
iDRX average current	1.5 mA		
Total charge	14.39 mC		
<b>GPS</b>			
GPS average current	187.08 $\mu$ A		
GPS fix total charge	673.5 mC		
		<b>Current consumption</b>	
		LTE event total charge	90.48 mC
		PSM floor current	2.7 $\mu$ A
		Total average current	214.91 $\mu$ A

(b) Online Power Profiler configured for 1 hour transmission intervals

Figure 5.8.5: Overview of estimated current usage for 10 minute and 1 hour transmission intervals from the Online Power Profiler [24]. Total average current is marked in green.

To get accurate measurements of the average current of each of the test cases the tests will be run and sampled for around 15 hours. For the baseline tests the output voltage will be set to 4.2 V to match the voltage when running the gateway mode tests.

In the node and gateway tests there is a single tracker connected to the PPK, which will be the Device Under Test (DUT). The other trackers are powered by 520 mAh Li-Po batteries. The gateways are elected based on their desirability, which is implemented as the highest battery voltage. A 4.2 V output from the PPK will be higher than the measured voltage from a fully charged battery. Therefore 4.2 V will be used as the output of the PPK as it ensures that the DUT will remain a gateway for the duration of the test. For the node tests the output voltage is set to 3.7 V as it is lower than the voltage of the battery when the battery powers off (due to undervoltage protection built into the batteries).

Using these output voltages the tests are guaranteed to run with the tracker constantly in the desired mode.

When the measurements are concluded, the data in the database will be reviewed to make sure



Figure 5.8.6: Image of the field test setup. One box contains two trackers, 5 and 6, while the other contains only one, 7.

that the tracker have not changed state. For a gateway test all reported gateways should be the DUT and for the node tests no reported gateway should be the DUT. The database entries will also show that sensors are polled, and data transmitted in correct intervals. The baseline firmware also uses the same data format and endpoints as the tracking solution, so that verification can be done for the baseline tests transmission intervals.

While there were not enough time to perform practical tests with moving nodes the trackers where put on the roof. Three nodes with IDs 5, 6 and 7 were left on the roof for several days and some nodes where moved around to simulate a sheep straying away from the other sheep. This test setup can be seen in Figure 5.8.6

One of the assumptions made when designing the sheep tracking system is that one ewe and two lambs will graze together. To better be able to compare the baseline solution and the proposed system design this assumption should be taking into account.

Which means: given a baseline mean current  $\bar{i}_{base}$ , a mean gateway current  $\bar{i}_{gw}$  and mean node current  $\bar{i}_{node}$ ; the comparison should be between  $\bar{i}_{base}$  and the mean system current,  $\bar{i}_{sys}$ , given in Equation (5.8.1).

$$\bar{i}_{sys} = \frac{\bar{i}_{gw} + 2\bar{i}_{node}}{3} \quad (5.8.1)$$

# Chapter 6

## Results

In this chapter all findings will be presented and any errors sources will be mentioned. The chapter will go through each set of tests in order. First the mechanical tests, followed by the functional tests, before the energy consumption tests.

### 6.1 Mechanical test results

The mechanical tests were performed as per Table 5.8.2. A visual inspection showed that all expected components were present on the board.

The dimensions of the tracker are illustrated in Figure 6.1.1.

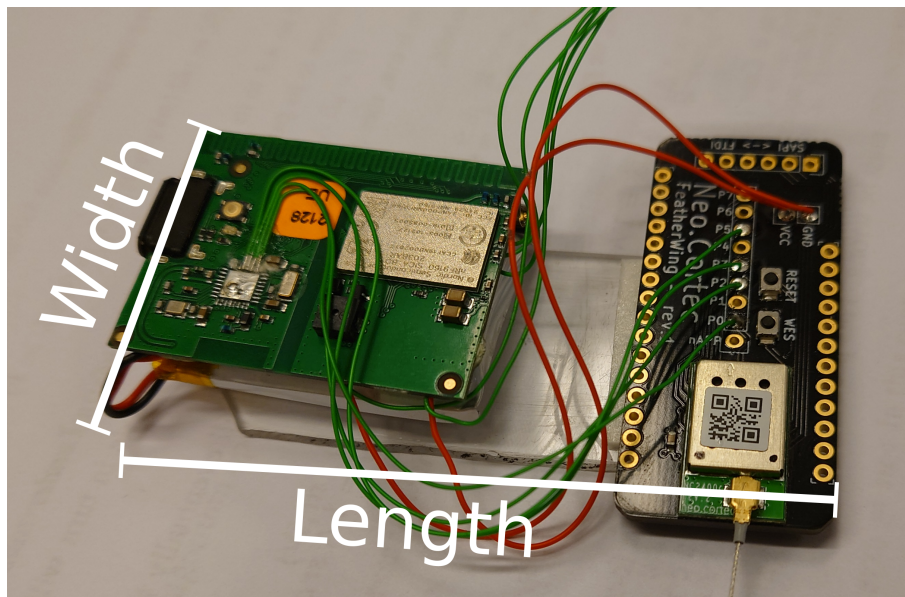
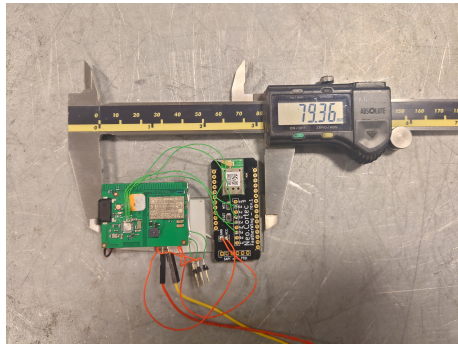


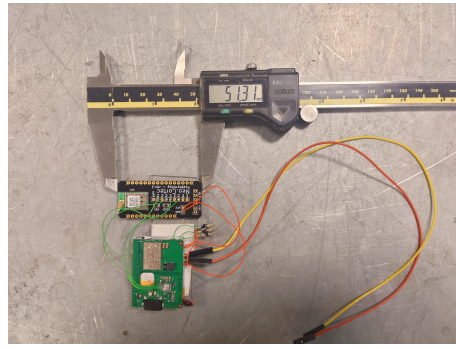
Figure 6.1.1: Illustration showing measurements of the tracker, height is from the table and upwards.

The actual measurements of the tracker is showed in Figures 6.1.2a to 6.1.2c.

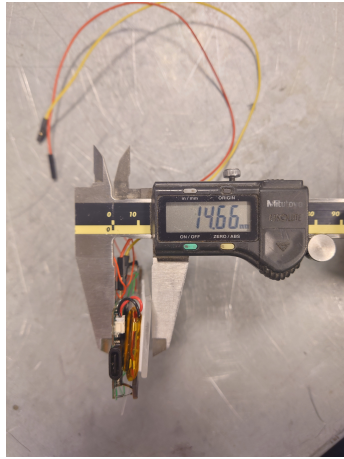




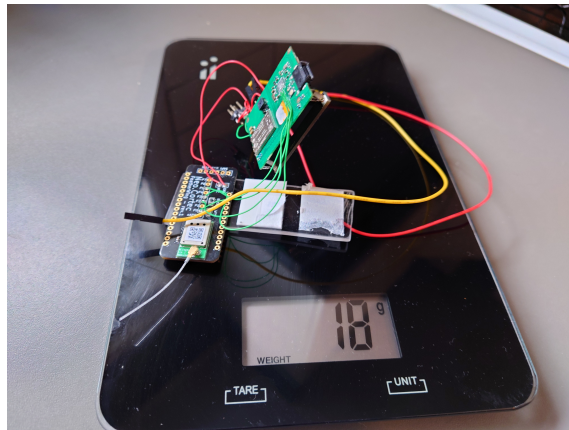
(a) Tracker width. The calipers reads 79.36 mm.



(b) Tracker length. The calipers reads 51.32 mm.



(c) Tracker height. The calipers reads 14.66 mm.



(d) Tracker weight. The scale reads 18 grams.

Figure 6.1.2: Figures with all the tracker measurements

For the last mechanical test a tracker was weighed on a scale, shown in Figure 6.1.2d. The weight was not including the battery, which was weighed separately at 10 grams, shown in Figure 6.1.3.



Figure 6.1.3: Battery weight. The scale reads 10 grams.

---

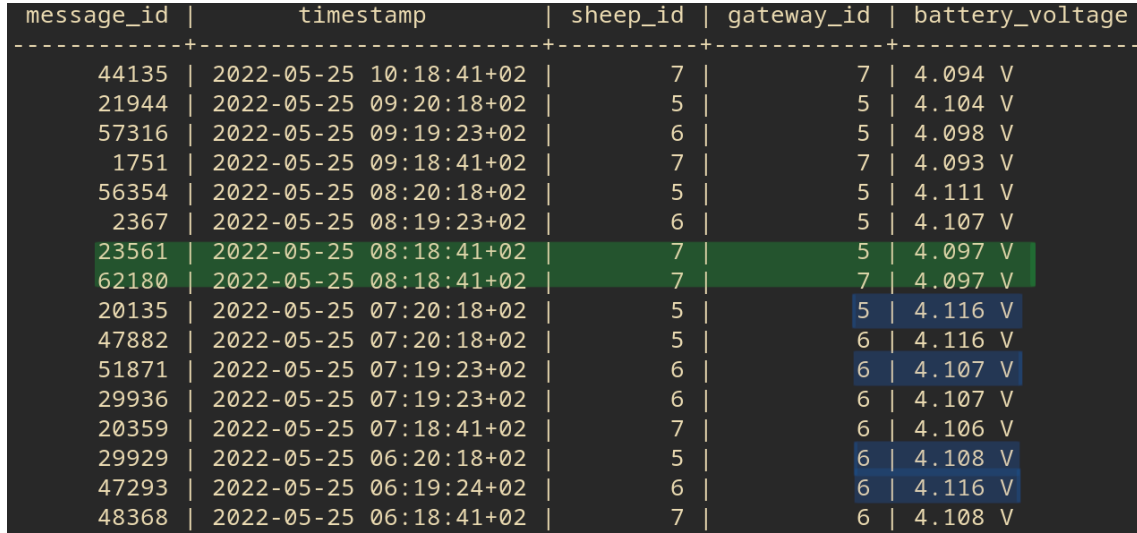
## 6.2 Functional test results

A logfile is included in Appendix B, this logfile shows the startup of a single node. The node is configured with a 180 seconds (3 minutes), term interval. As the tracker boots it immediately elects itself as a gateway, line 10 is where the leader is set, while the promotion can be seen on line 24. The tracker connects to the cloud and transmits a state report(line 36-40), then goes to sleep afterwards. Lines 44 to 52 shows the periodic reading of the neighbour list from the NC2400.

At line 53 of the logfile, after 2 minutes and 37 second, the term is ended and a new election is started. The tracker immediately wins as there are no neighbours, as seen on line 57. The same can be observed on line 70, after 6 minutes.

Figure 6.2.1 shows a screenshot from the database where tracker data is stored. At 08:44 the tracker with ID 7 was removed from the other two nodes. Two samples, marked in green, shows that tracker 7 reports both itself and 5 as gateway, on the same sample. This will be discussed in Chapter 7. After the green marked lines tracker 7 also starts reporting itself as its gateway, instead of 5.

In the same figure, marked in blue, is the battery voltage of tracker 5 and 6. The first two, furthest down on the figure, shows that tracker 6 has a higher voltage than 5. While the upper two shows that 5 have a higher voltage than 6 and that 5 takes over as a gateway.



message_id	timestamp	sheep_id	gateway_id	battery_voltage
44135	2022-05-25 10:18:41+02	7	7	4.094 V
21944	2022-05-25 09:20:18+02	5	5	4.104 V
57316	2022-05-25 09:19:23+02	6	5	4.098 V
1751	2022-05-25 09:18:41+02	7	7	4.093 V
56354	2022-05-25 08:20:18+02	5	5	4.111 V
2367	2022-05-25 08:19:23+02	6	5	4.107 V
23561	2022-05-25 08:18:41+02	7	5	4.097 V
62180	2022-05-25 08:18:41+02	7	7	4.097 V
20135	2022-05-25 07:20:18+02	5	5	4.116 V
47882	2022-05-25 07:20:18+02	5	6	4.116 V
51871	2022-05-25 07:19:23+02	6	6	4.107 V
29936	2022-05-25 07:19:23+02	6	6	4.107 V
20359	2022-05-25 07:18:41+02	7	6	4.106 V
29929	2022-05-25 06:20:18+02	5	6	4.108 V
47293	2022-05-25 06:19:24+02	6	6	4.116 V
48368	2022-05-25 06:18:41+02	7	6	4.108 V

Figure 6.2.1: Database data where tracker is removed and node desirability falls

Figure 6.2.2 shows a screenshot of one of the other tables in the database, which contains each trackers reported gateway. The rows marked in green highlights that node 7, having previously used 5 as a gateway, starts using itself as a gateway at 08:44.

It can also be observed that 6 is used as a gateway for the other two nodes, but 5 takes over after 07:25, seen from the two rows below the highlighted ones.

timestamp	sheep_id	gateway_id
2022-05-25 09:44:15+02	7	7
2022-05-25 09:25:44+02	5	5
2022-05-25 09:19:23+02	6	5
2022-05-25 08:44:15+02	7	7
2022-05-25 08:25:44+02	5	5
2022-05-25 08:19:23+02	6	5
2022-05-25 08:18:41+02	7	5
2022-05-25 07:25:44+02	5	5
2022-05-25 07:25:34+02	6	6
2022-05-25 07:23:35+02	6	6
2022-05-25 07:20:18+02	5	6
2022-05-25 07:18:41+02	7	6
2022-05-25 06:23:35+02	6	6
2022-05-25 06:20:18+02	5	6
2022-05-25 06:18:41+02	7	6

Figure 6.2.2: Screenshot of database where tracker is removed and sheep tracker 7 takes over as gateway.

From Figure 6.2.3, it can be observed that node 7, having previously reported 6 and 7 as neighbours, starts reporting no neighbours at 08:44. The relevant rows are highlighted in green.

The figure also shows that neither 5, nor 6 reports 7 as a neighbour after 08:44 (above the topmost highlighted row).

timestamp	sheep_id	neighbour_id
2022-05-25 09:43:48+02	7	
2022-05-25 09:25:23+02	5	6
2022-05-25 09:19:23+02	6	5
2022-05-25 08:44:14+02	7	
2022-05-25 08:25:24+02	5	6
2022-05-25 08:25:24+02	5	7
2022-05-25 08:19:23+02	6	7
2022-05-25 08:19:23+02	6	5
2022-05-25 08:18:41+02	7	6
2022-05-25 08:18:41+02	7	5
2022-05-25 07:25:29+02	6	7
2022-05-25 07:25:29+02	6	5
2022-05-25 07:25:23+02	5	6
2022-05-25 07:25:23+02	5	7
2022-05-25 07:23:28+02	6	5
2022-05-25 07:23:28+02	6	7
2022-05-25 07:20:18+02	5	7
2022-05-25 07:20:18+02	5	6
2022-05-25 07:18:41+02	7	5
2022-05-25 07:18:41+02	7	6
2022-05-25 06:23:32+02	6	7
2022-05-25 06:23:32+02	6	5
2022-05-25 06:20:18+02	5	7
2022-05-25 06:20:18+02	5	6
2022-05-25 06:18:41+02	7	5
2022-05-25 06:18:41+02	7	6

Figure 6.2.3: Screenshot of reported tracker neighbours from database.

After tracker 7 was left by its own for a few hours it was moved back to the cluster again. This was done at 11:00 on the 25th of May. Tracker 7 did not report any neighbours for a long time, shown in Appendix C. It was assumed that there was a bug in the NeoMesh driver the NC2400 was reset at around 11 on the 27th of May.

As seen in Figure 6.2.4, at 12:18, tracker 7 once again reported tracker 5 as a gateway. The rows are highlighted in green.



message_id	timestamp	sheep_id	gateway_id	battery_voltage
56002	2022-05-27 12:20:13+02	5	5	4.078 V
12892	2022-05-27 12:19:20+02	6	5	4.057 V
6107	2022-05-27 12:18:37+02	7	5	3.991 V
6107	2022-05-27 12:18:37+02	7	5	3.991 V
26878	2022-05-27 11:20:13+02	5	6	4.074 V
17755	2022-05-27 11:20:13+02	5	5	4.074 V
51287	2022-05-27 11:19:20+02	6	6	4.053 V
44526	2022-05-27 11:19:20+02	6	6	4.053 V
17293	2022-05-27 11:18:37+02	7	7	3.995 V
2231	2022-05-27 11:18:37+02	7	7	3.995 V
13695	2022-05-27 10:20:13+02	5	6	4.073 V
40702	2022-05-27 10:19:20+02	6	6	4.061 V
26417	2022-05-27 10:18:37+02	7	7	4.008 V
26417	2022-05-27 10:18:37+02	7	7	4.008 V
16034	2022-05-27 09:20:14+02	5	6	4.078 V
55061	2022-05-27 09:19:20+02	6	6	4.071 V
37419	2022-05-27 09:18:38+02	7	7	4.025 V
22774	2022-05-27 08:20:14+02	5	6	4.080 V
23141	2022-05-27 08:19:20+02	6	6	4.071 V
11832	2022-05-27 08:18:38+02	7	7	4.027 V
11832	2022-05-27 08:18:38+02	7	7	4.027 V
13700	2022-05-27 07:20:14+02	5	6	4.073 V
59458	2022-05-27 07:19:20+02	6	6	4.090 V

Figure 6.2.4: Screenshot of reported tracker status after node 7 rejoins the cluster.

The change in neighbours can also be observed in Figure 6.2.5, where at 11:18, tracker 7 reported no neighbours, hence the missing value, while at 12:18 the tracker reported 5 and 6 as neighbours.

message_id	timestamp	sheep_id	neighbour_id
1104	2022-05-27 13:25:50+02	5	7
1104	2022-05-27 13:25:50+02	5	6
23928	2022-05-27 13:19:20+02	6	7
23928	2022-05-27 13:19:20+02	6	5
2960	2022-05-27 13:18:37+02	7	5
2960	2022-05-27 13:18:37+02	7	6
56002	2022-05-27 12:25:47+02	5	7
56002	2022-05-27 12:25:47+02	5	6
12892	2022-05-27 12:19:20+02	6	7
12892	2022-05-27 12:19:20+02	6	5
6107	2022-05-27 12:18:37+02	7	5
6107	2022-05-27 12:18:37+02	7	6
2231	2022-05-27 11:30:12+02	7	5
51287	2022-05-27 11:25:56+02	6	5
17755	2022-05-27 11:25:49+02	5	6
26878	2022-05-27 11:20:13+02	5	6
44526	2022-05-27 11:19:24+02	6	7
44526	2022-05-27 11:19:24+02	6	5
17293	2022-05-27 11:18:15+02	7	
13695	2022-05-27 10:20:13+02	5	6
40702	2022-05-27 10:18:57+02	6	7
40702	2022-05-27 10:18:57+02	6	5
26417	2022-05-27 10:18:12+02	7	

Figure 6.2.5: Screenshot of reported tracker neighbours after node 7 rejoins the cluster.

Shown in Figure 6.2.4, at 08:18 two messages are transmitted by 7.

message_id	timestamp	sheep_id	battery_voltage
56354	2022-05-25 08:20:18+02	5	4.11125390625
2367	2022-05-25 08:19:23+02	6	4.1069638671875
23561	2022-05-25 08:18:41+02	7	4.0969541015625
62180	2022-05-25 08:18:41+02	7	4.0969541015625

Figure 6.2.6: Screenshot of tracker status after node 7 rejoins the cluster.

Figure 6.2.7 shows a screenshot of the trackers reported position from the database. The position data is highlighted in green.

message_id	timestamp	sheep_id	gateway_id	battery_voltage	latitude	longitude	position_accuracy_percent
14980	2022-05-30 19:20:08+02	5	6	4.031 V	10.428997	63.443147	46.6 %
14980	2022-05-30 19:20:08+02	5	6	4.031 V	10.429753	63.443042	9.3 %
13130	2022-05-30 19:20:08+02	5	5	4.031 V	10.428997	63.443147	46.6 %
13130	2022-05-30 19:20:08+02	5	5	4.031 V	10.429753	63.443042	9.3 %

Figure 6.2.7: Screenshot of reported tracker positions from database.

## 6.3 Power usage test results

This section will present data sampled during the power usage tests. The tests results will be the statistics from the entire test run, a screenshot to verify the transmission period and a selection showing the floor current of each test run.

### 6.3.1 10 minute interval - baseline

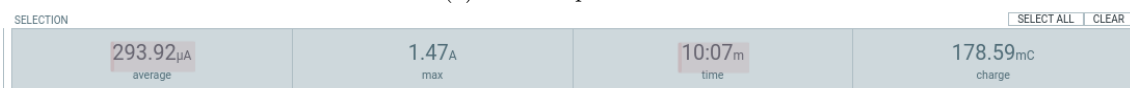
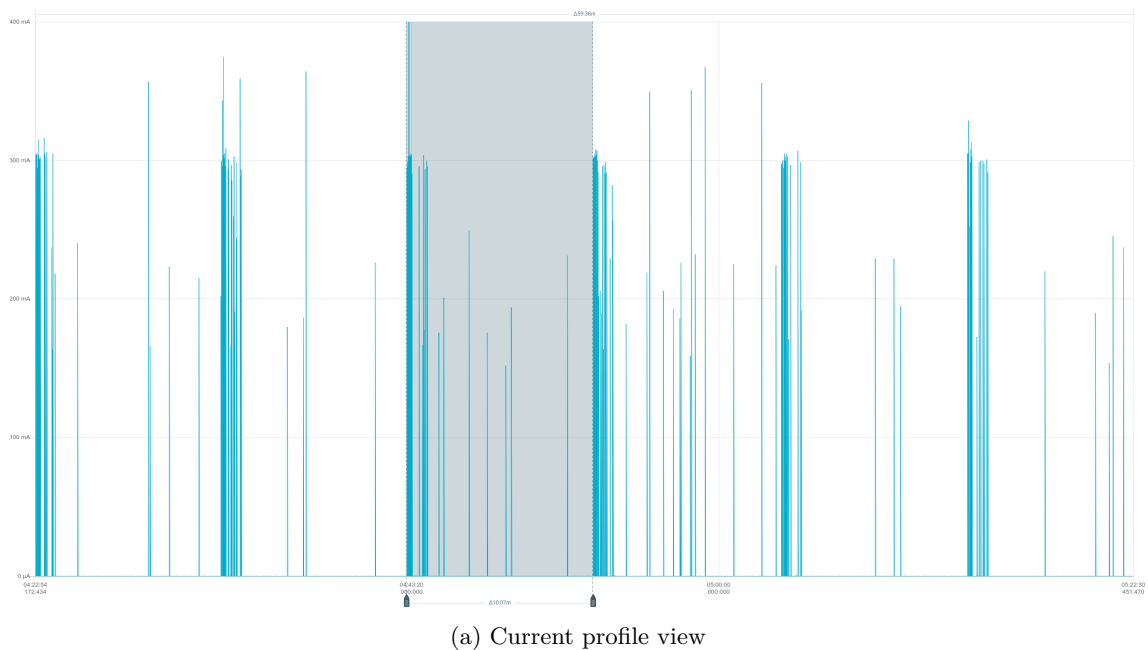
The first tested configuration was the 10 minute interval one.

Figure 6.3.1 shows the results from the baseline application. The Power Profiler have selected the entire interval of 19 hours and the average current is read out to be 336.61  $\mu$ A.

SELECTION			SELECT ALL CLEAR
336.61 $\mu$ A average	2.13A max	19:00:00h time	23.02c charge

Figure 6.3.1: Statistics from baseline firmware during 10 minute interval tests.

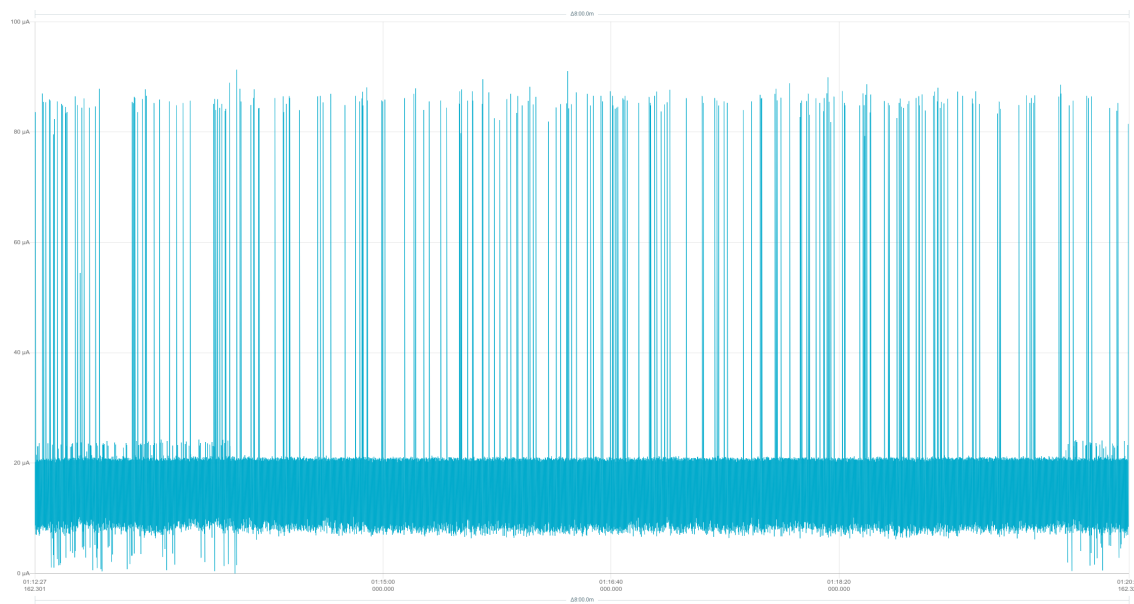
Figure 6.3.2a have selected the period of one transmission and the figure shows that the time between two transmissions is 10 minutes. In Figure 6.3.2b the statistics for that selection is shown, including the duration of the selected interval.



(b) Statistics view

Figure 6.3.2: Data from baseline firmware, with a single transmit interval selected

Figure 6.3.3 shows the floor current of the baseline solution, measured at  $14.35 \mu\text{A}$ .

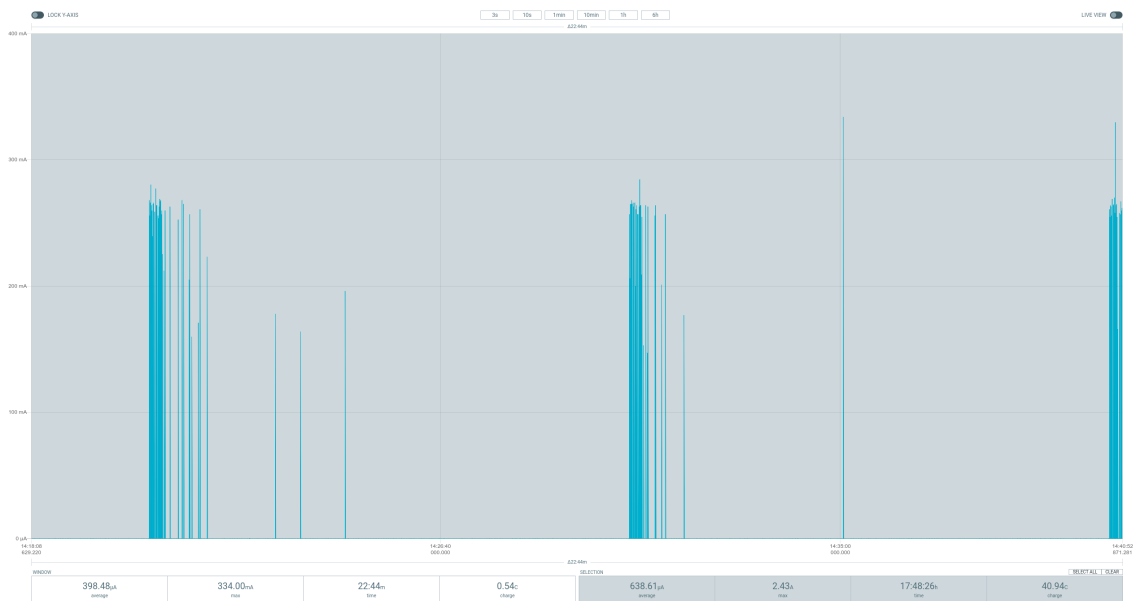


(b) Statistics view

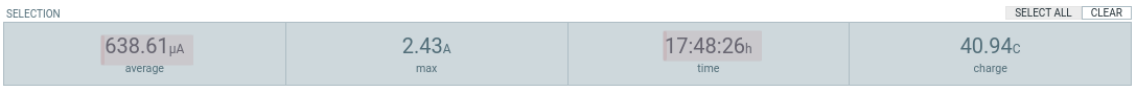
Figure 6.3.3: Data from baseline firmware, view of the floor current

### 6.3.2 10 minute interval - gateway mode

The last test in this configuration is the gateway current usage. It can be seen in Figure 6.3.4. The average current draw of the gateway over 17 hours and 48 minutes is  $638.61\text{ }\mu\text{A}$ , as shown in Figure 6.3.4b.



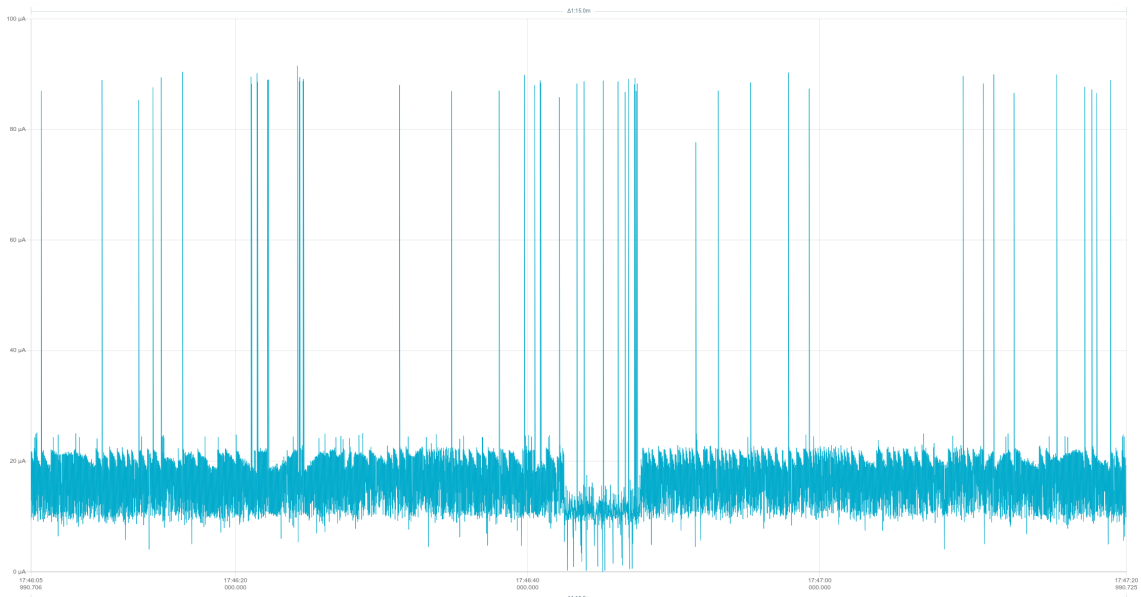
(a) Profile view



(b) Statistics view

Figure 6.3.4: Data set of a tracker in gateway mode with 10 minute transmission intervals

In Figure 6.3.5 the floor current of the gateway is shown. Measured at  $15.95\text{ }\mu\text{A}$ , as shown in Figure 6.3.5b.



(a) Current profile view

WINDOW			
15.95µA average	91.49µA max	1:15.0m time	1.20mC charge

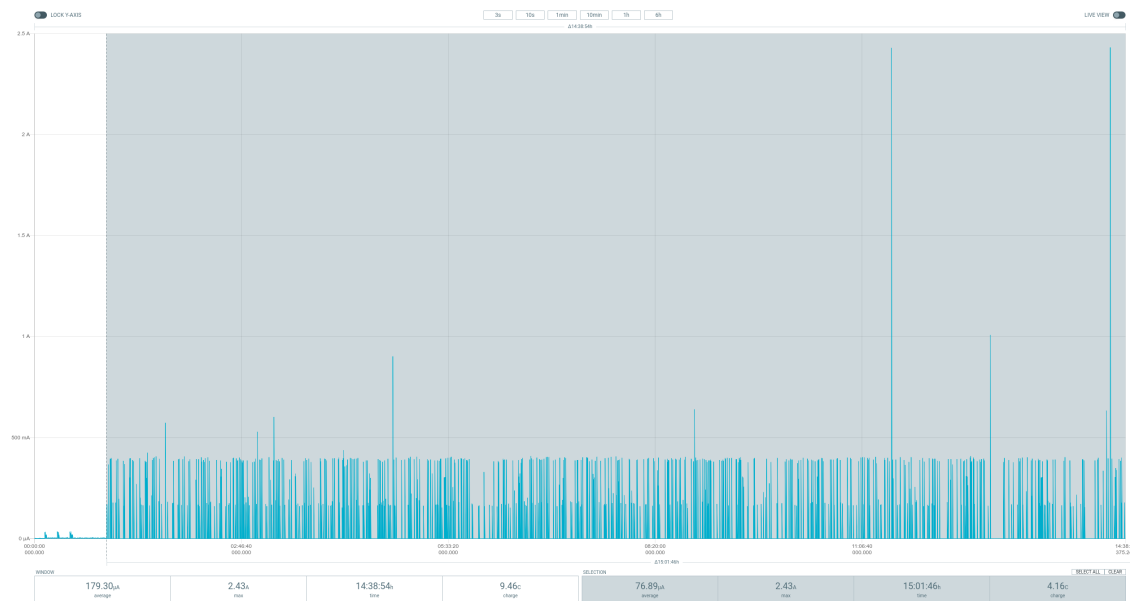
(b) Statistics view

Figure 6.3.5: View of a floor current from tracker running gateway mode with a 10 minute interval

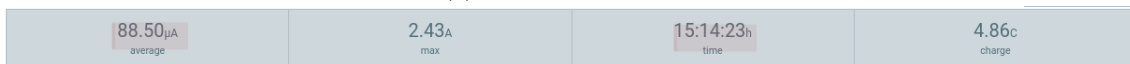
### 6.3.3 10 minute interval - node mode

In Figure 6.3.6 a dataset from a tracker in node mode is shown. The figure also shows that there is a small bit of data excluded from the start of the data. This small section is an LTE connection event, which draws a large amount of current, significantly impacting the measurement. It has been excluded from this measurement, all other measurements are started after the connection event has completed, so it will not impact the comparison between the results.

As shown in Figure 6.3.6b, average current consumption is measured at 88.50  $\mu\text{A}$  over a 15 hour period.



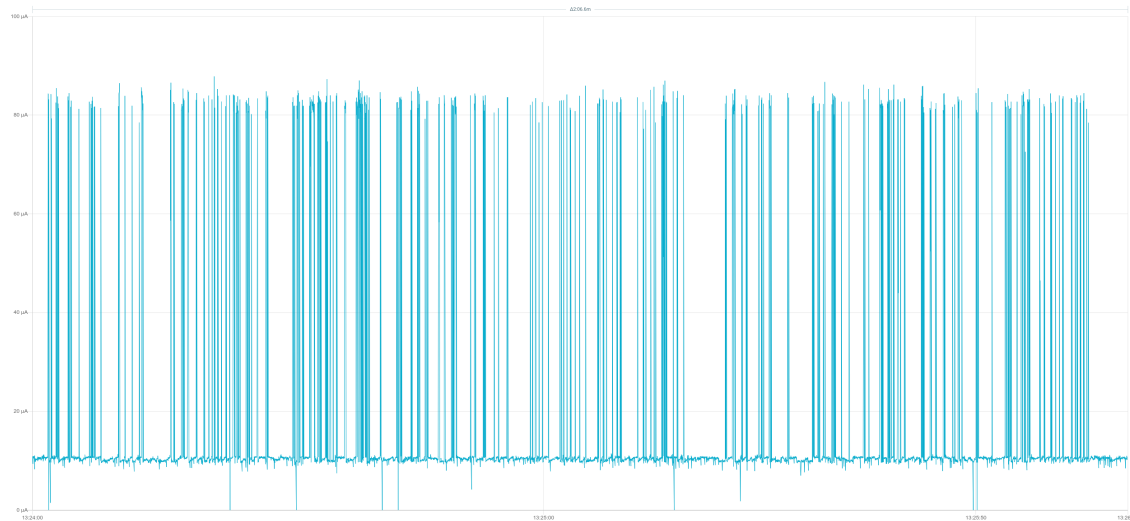
(a) Current profile view



(b) Statistics view

Figure 6.3.6: Data set of a tracker in node mode with 10 minute intervals

Figure 6.3.7 shows the floor current of the tracker in node mode. Measured at 15.94  $\mu$ A, as shown in Figure 6.3.7b.



(a) Current profile view



(b) Statistics view

Figure 6.3.7: View of a sleep interval from tracker running node mode

### 6.3.4 1 hour interval - baseline

Figure 6.3.8 shows the energy usage statistics of the baseline software at 1 hour intervals. The figure shows that the average current usage over 15 hours is  $148.83\ \mu\text{A}$ .

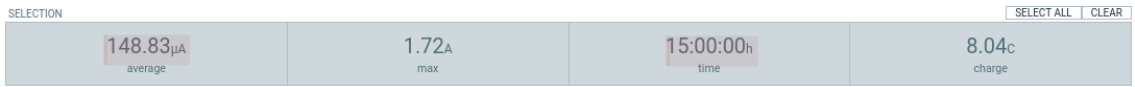
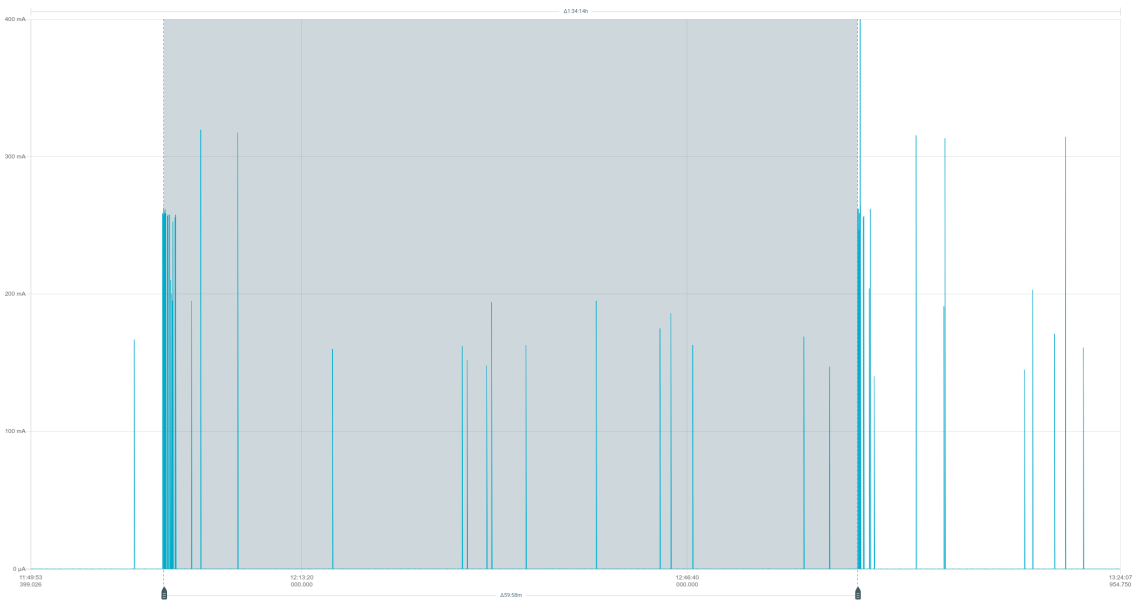
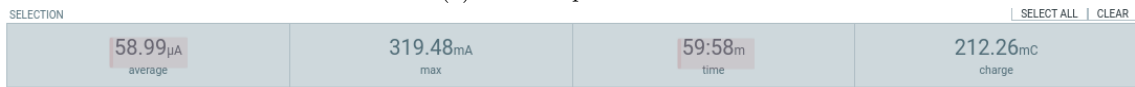


Figure 6.3.8: Energy usage statistics of baseline application with 1 hour transmit intervals

A more detailed view of the current profile is shown in Figure 6.3.9, also verifying the transmission period.



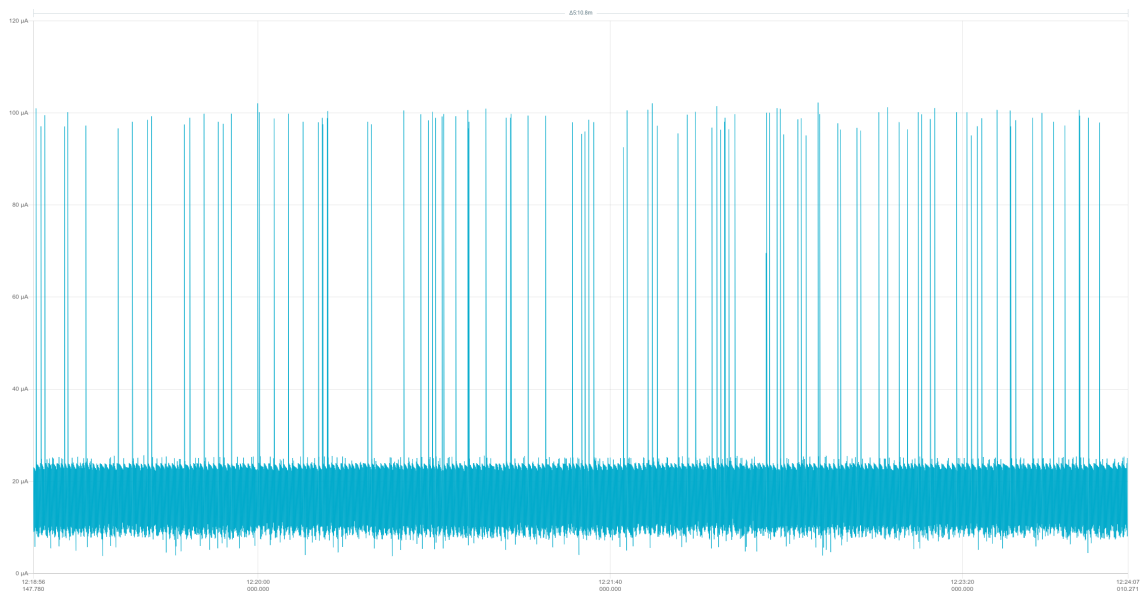
(a) Current profile view



(b) Statistics view

Figure 6.3.9: Overview of 1 hour interval, baseline firmware

Figure 6.3.10 shows a detailed view of the floor current of the baseline solution in the 1 hour configuration. It is measured at  $17.25\ \mu\text{A}$ , as shown in Figure 6.3.10b.



(a) Current profile view

WINDOW			
17.25µA average	102.22µA max	5:10.8m time	5.36mC charge

(b) Statistics view

Figure 6.3.10: Baseline firmware floor current during 1 hour interval setup

### 6.3.5 1 hour interval - gateway mode

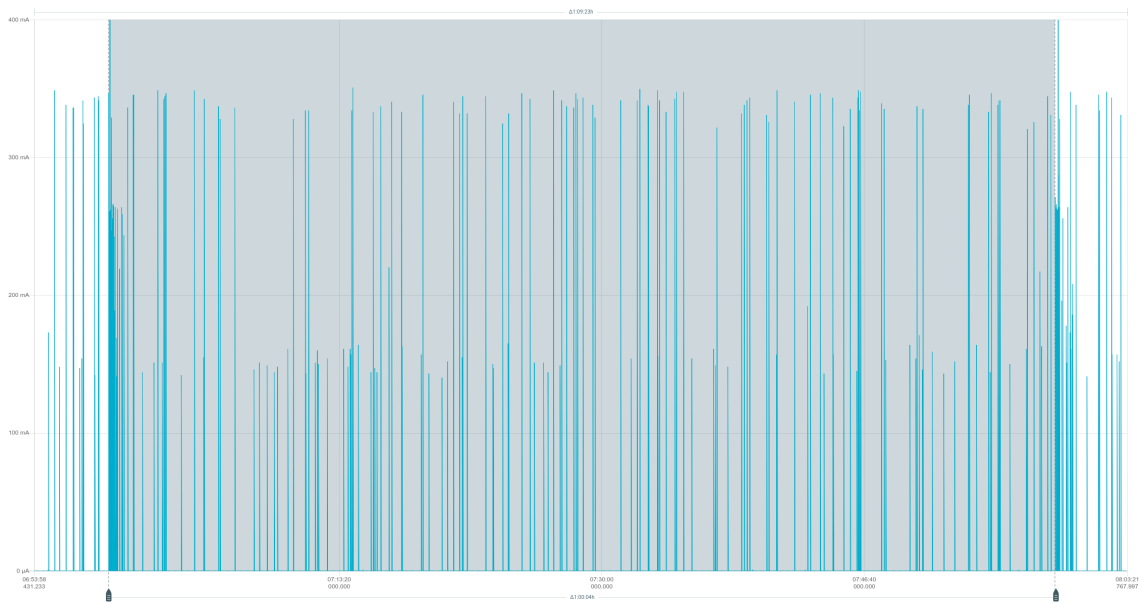
As seen in Figure 6.3.11, the gateway consumed 228.02  $\mu\text{A}$  on average, using the 1 hour interval, also measured over 15 hours.

SELECTION				SELECT ALL CLEAR	
228.02µA average	2.42A max	15:00:00h time	12.31C charge		

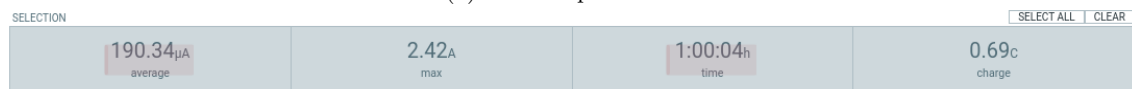
Figure 6.3.11: Energy usage stats of gateway with 1 hour transmit intervals

Figure 6.3.12 shows the interval and the period between the transmissions.





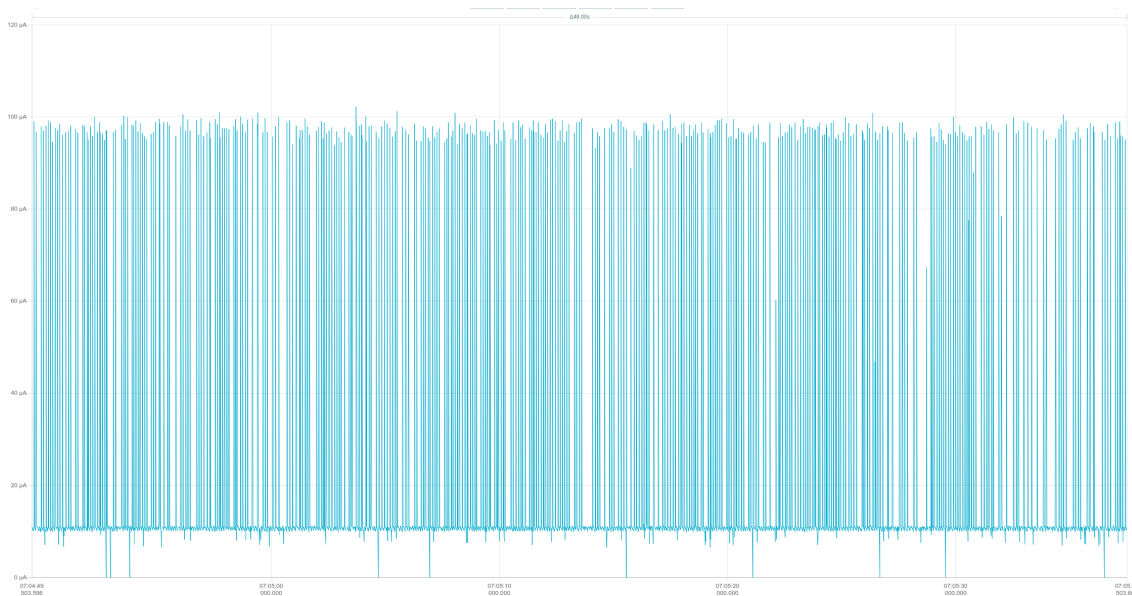
(a) Current profile view



(b) Statistics view

Figure 6.3.12: Overview of 1 hour interval, gateway mode

In Figure 6.3.13 the floor current of the gateway node is shown. It is measured at  $18.58 \mu\text{A}$ , as shown in Figure 6.3.13b.



(a) Current profile view



(b) Statistics view

Figure 6.3.13: Gateway mode floor current during 1 hour interval setup

### 6.3.6 1 hour interval - node mode

Lastly, the node consumed 53.44  $\mu\text{A}$  over the 15 hours, as seen in Figure 6.3.14.

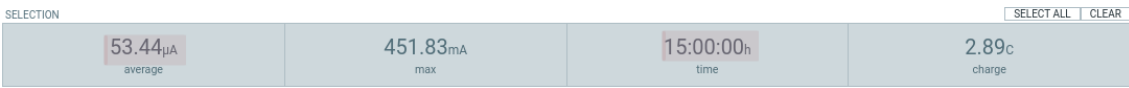
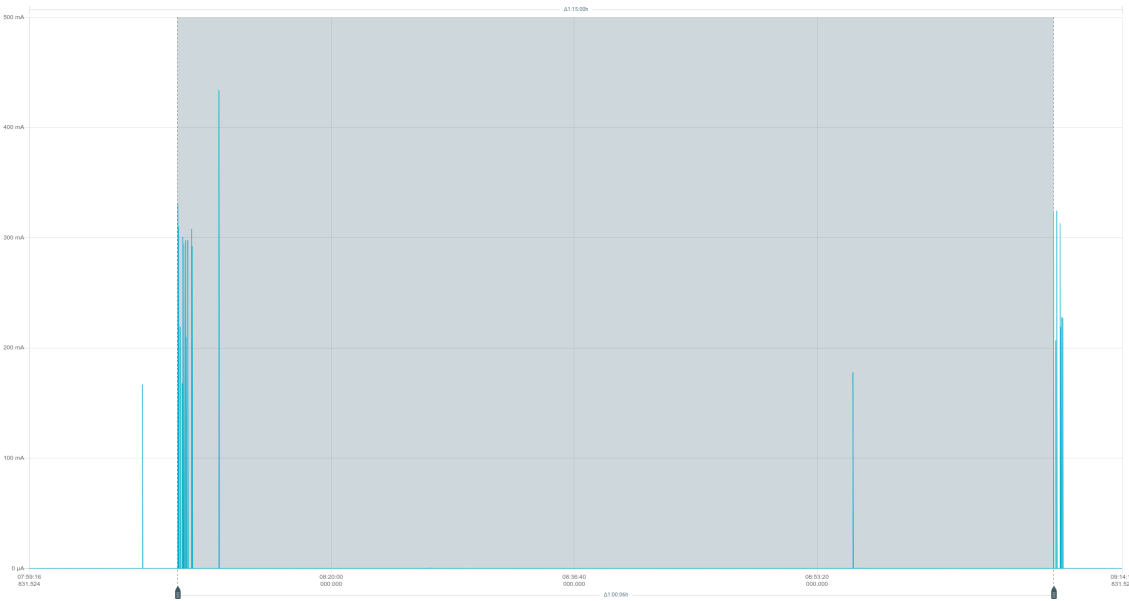
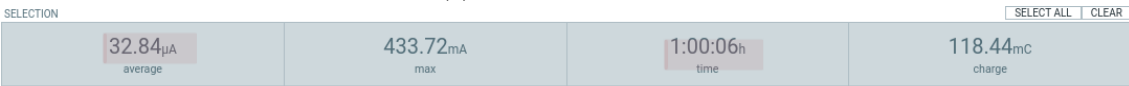


Figure 6.3.14: Energy usage stats of node with 1 hour transmit intervals

In Figure 6.3.15 the 1 hour interval is selected.



(a) Current profile view



(b) Statistics view

Figure 6.3.15: Overview of 1 hour interval, node mode

The floor current of the node is shown in Figure 6.3.16 and was measured at 14.34  $\mu\text{A}$ , shown in Figure 6.3.16b.

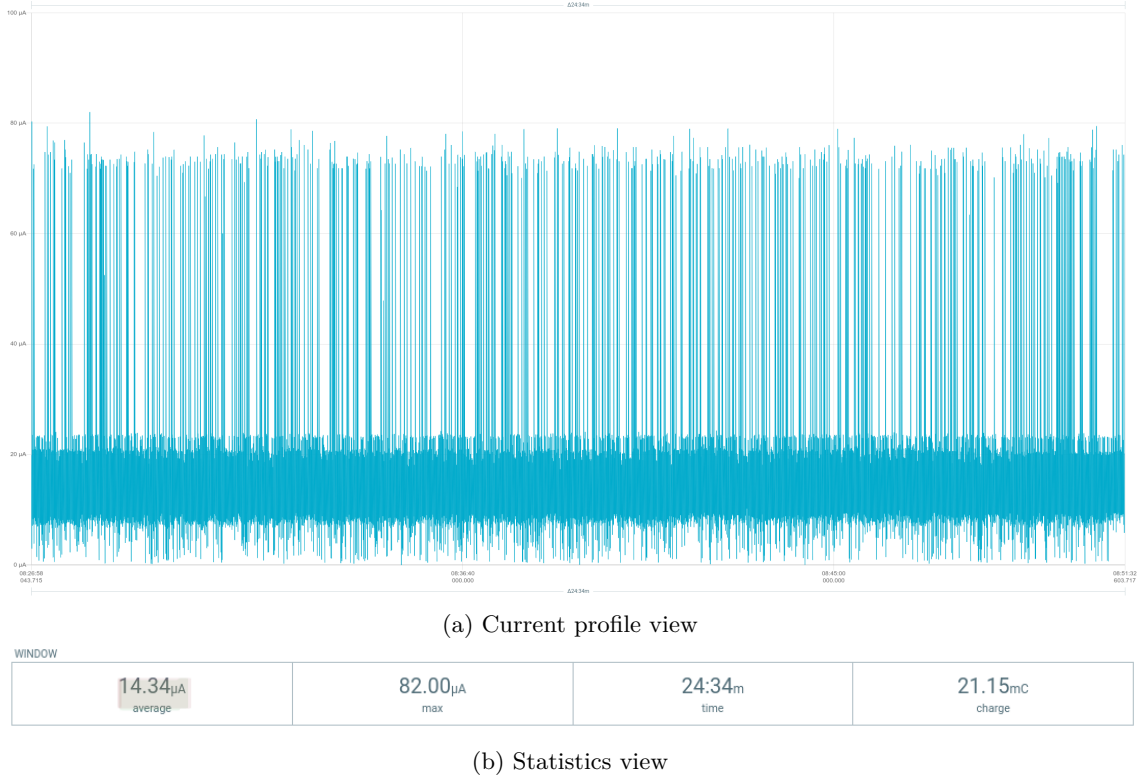


Figure 6.3.16: Node mode floor current during 1 hour interval setup

### 6.3.7 Power usage summary

All the data from the previous sections have been summarized into a single table, Table 6.3.1. The mean system current,  $\bar{i}_{sys}$  has also been calculated and added to the table, using Equation (5.8.1).

	Measurement type	Average current draw	Measurement period
10 minute interval	Baseline ( $\bar{i}_{base}$ )	293.92 $\mu\text{A}$	19 hours
	Gateway ( $\bar{i}_{gw}$ )	638.61 $\mu\text{A}$	17 hours 48 minutes
	Node ( $\bar{i}_{node}$ )	88.50 $\mu\text{A}$	15 hours
	$\bar{i}_{sys}$	271.87 $\mu\text{A}$	
1 hour interval	Baseline ( $\bar{i}_{base}$ )	148.83 $\mu\text{A}$	15 hours
	Gateway ( $\bar{i}_{gw}$ )	228.02 $\mu\text{A}$	15 hours
	Node ( $\bar{i}_{node}$ )	54.44 $\mu\text{A}$	15 hours
	$\bar{i}_{sys}$	112.30 $\mu\text{A}$	

Table 6.3.1: Test results for energy usage tests

## 6.4 Error sources

This section will present the possible error sources in the measurements and test cases.

---

### 6.4.1 Missing datapoints

When using the PPK for a little while an issue arose where a few samples were missed. This can be seen in Figure 6.4.1, where odd gaps in the dataset are clearly missing. The issue happens sporadically, but accounts for less than 0.5% of the samples.

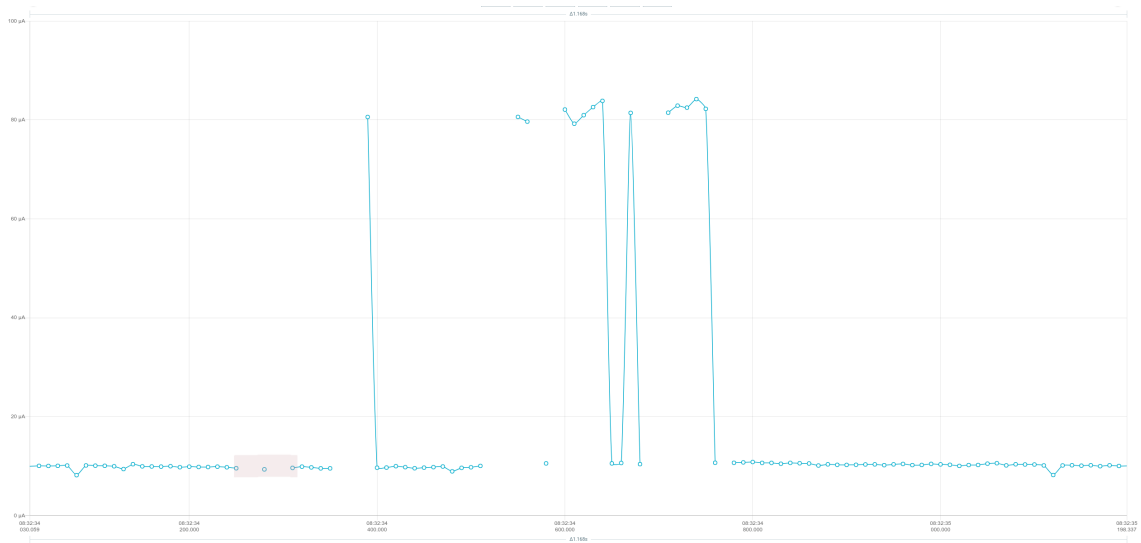


Figure 6.4.1: PPK view showing missing samples in the dataset. An example is highlighted in red.

# Chapter 7

## Discussion

The results from the test cases have been presented and in this chapter the results will be discussed in detail. Initially the results from the tests will be discussed and the final coverage of the stakeholder requirements is presented.

This leads into the discussion of how different parameters affect the overall system performance. Afterwards, shortcomings and possible improvements of the system is presented as well as a look in to how the principles of the thesis can be applied to other scenarios. At the end of the chapter a brief note about the complexity of the system is presented.

### 7.1 Test results

In this section the results from the different tests will be discussed and a summary of the test coverage will be presented.

#### 7.1.1 Mechanical tests

Beginning with the simplest test suite, all mechanical tests passed and the results are summarized in Table 7.1.1, based on the measurements and observations from Figure 6.1.2.

Test case	Result
<b>TC-3.1</b>	Passed. All needed components where present on all trackers.
<b>TC-3.2</b>	Passed. Trackers where weighed at 29 grams.
<b>TC-3.3</b>	Passed. The trackers where measured at 79.39 mm wide, 51.31 mm long, 14.66 mm high.

Table 7.1.1: Test results for mechanical tests

There are some additional remarks regarding the weight of the trackers, but it will be presented in Section 7.1.4

---

### 7.1.2 Functional tests

Based on the observations from Chapter 6 all but one of the functional tests passed. The summary is shown in Table 7.1.2 and will be discussed in more detail below.

Test case	Result
TC-1.1	Passed. Elections are started at regular intervals.
TC-1.2	Passed. When a gateway is removed from the network, a new gateway is elected.
TC-1.3	Passed. If a gateway falls in desirability a new gateway is elected.
TC-1.4	Passed. A gateway will transmit any remaining buffered messages before it is demoted.
TC-1.5	Failed. No tests were performed, test failed.
TC-1.6	Passed, but unstable. As a node is moved close to the network it is eventually merged into the network. There is however a bug in the NeoCortec driver causing the driver to sometimes hang when neighbours change, requiring a reset of the NC2400 for the test to pass.
TC-1.7	Passed. Tracker states can be found in the database.
TC-1.8	Passed. Tracker positions can be found in database, but the positioning subsystem is not robust enough.

Table 7.1.2: Test results for network integrity tests

As presented in Chapter 6, the debug output from the node shows that an election is started on a regular interval. While it is unclear why the first election period is a bit shorter than what was configured, it does not impact the end results at all. In addition to the logfile any change of gateway due to the change of desirability would not have happened. The database entries shows, particularly Figure 6.2.1, that gateways do change when desirability changes. It is fair to say that **TC-1.1** has passed.

When node 7 is removed from the others it adopts itself as a gateway. This is what shown in Figure 6.2.1. The reason that there are two entries, with the same data, except different gateways is the Structured Query Language (SQL) query used to retrieve the data. The query does a join operation on the gateway tables and battery status tables, but it does not take in to account that only a single gateway can be active at the time. A more advanced query could be used, or a non-relational database, but since this was not the focus of the thesis, it was not done. Since 7 does indeed adopts a new gateway when the existing gateway is removed, **TC-1.2** passes.

From Figure 6.2.1, the blue marked areas, show that the battery of tracker 6 drops below that of 5, which subsequently causes 5 to take over as a gateway during the next election. This behaviour confirms **TC-1.3**.

When tracker 7 is reintroduced into the cluster, as shown in Figure 6.2.4, it can be seen, from Figure 6.2.6, that tracker 7 transmits two state reports. This happens because as the tracker is demoted it empties out its internal cloud buffer and transmits the messages. The same measurement is then also sent to its new gateway, resulting in two measurements with the same timestamp, but different message ids. This confirms **TC-1.4**.

No retransmission was implemented for the nodes. The acknowledgements from the gateway was implemented, but the retransmission was left out due to time constraints. Since loss of data is not the most critical issue for the thesis, it was simply left out, but it means that **TC-1.5** fails.

---

Since each tracker forms its own network, consisting of one node, every time it boots, it should suffice to say that **TC-1.6** has passed. But, there is a bug, most likely in the NeoCortec driver, that causes the NC2400 communication to hang. This is what happened in the field test causing tracker 7 not to rejoin the cluster as it was moved back. The fact that tracker 7 reported no neighbours between being moved back and having its NC2400 reset further confirms this hypothesis. Changes in the mesh network is fully handled by the NC2400 and therefore the bug was either a hang in the NC2400 or the driver. Since this bug happened only this once, **TC-1.6** is considered passing.

While necessary, **TC-1.7** and **TC-1.8**, are considered passing as battery voltages are consistently reported and position estimates are sometimes found in the database. It is worth mentioning that the GNSS module did have a few bugs, meaning it could not reliably get a position fix every transmission interval. This will be discussed in a later section.

This concludes the results of the functional tests.

### 7.1.3 Energy consumption

First of all, in the last part of Chapter 6, the problem with missing samples was presented. The issue affects all measurements equally and accounts for so little that it is fair to say that the impact will be negligible.

In a worst case scenario, where a single sheep is isolated and no other trackers are nearby, the proposed system is outperformed by the naive solution. On a 10 minute transmission interval the mesh solution was measured to draw over 110% more current on average, while for the 1 hour interval it was slightly better at a 50% increased current draw. To properly support a 4 month season this indicates that a battery of at least 1850 mAh is required for the tracker to last. This is calculated by using the formula from Equation (7.1.1), where  $c_{min}$  is the minimal capacity needed by the battery.  $\bar{i}$  is the average current draw, multiplied by the amount of hours in a full 4 months.

$$c_{min} = \bar{i} * 24 * 30 * 4 \quad (7.1.1)$$

On the other hand, when the ewe and her lambs stick together the proposed solution shows more promise. Running with a 10 minute interval causes the solution to outperform the baseline solution by nearly 10%. On a 1 hour interval the performance increase is about 30%. In this scenario a battery capacity of only 350 mAh would be necessary to last a full season.

It is important to mention that neither the baseline nor the tracking firmware were as energy-optimized as they could be. Optimizing the firmware should have improved both solutions and may have shifted the results as the tracking firmware may have larger room for optimization. This may be a candidate for further work, to better understand the possibilities of using mesh networks to conserve energy.

After performing the experiments and tests it was uncovered that the IMU attached to the trackers, a WSEN-ITDS IMU, might have been active during the measurements. The module was soldered off and some better performance was observed in all three modes, but there was not enough time to redo all the experiments. Since the IMU should affect all solutions the same, as the same hardware was used for all tests, the impact to the conclusions drawn about the energy usage should be minimal. It suffices to say that a better performance could be achieved in all measurements taken.

Regardless, the measurements and tests show that there is a possibility to utilize a low energy mesh network in conjunction with a LPWAN transceiver to conserve energy.

### 7.1.4 Tracker weight

As mentioned previously, when discussing the mechanical tests, the weight of the tracker is still not set. There are two constraints that must be considered, first the tracker battery must be able

to survive the full season, i.e. 4 months, and the tracker must weigh less than 100 grams, as per the requirements.

To make sure the first constraint is fulfilled the battery attached on the tracker must have sufficient capacity. To add some margins a 2000 mAh battery could be attached, this will ensure that even isolated trackers will survive the season. Assuming that battery weight to capacity is fairly linear the 2000 mAh battery would weigh about 40 grams, using the 520 mAh/10 gram battery as a reference, see Figure 6.1.3. This yields a full tracker weight of 58 grams, which is well below the requirements.

If a future test of the system, on an actual sheep herd, would indicate that at least three sheep stick together during the grazing season, then a smaller battery could be used. To fulfil the battery capacity constraint, with some margin, the 520 mAh battery could be used. This yields a tracker weight of as little as 28 grams. Which is also well below the requirements.

The tracker weight as measured in Chapter 6 is a bit heavier than a more industrialized product would be. When weighing the tracker the tracker itself, the NC2400 breakout board, a lot of patch wires and the plexiglass (used to keep the two modules together) was included. The weight of the plexiglass, wires and the breakout board could be almost completely removed if there was enough time to create a circuit board for the solution. This would have reduced the weight of the trackers by at least a few grams, which would further improve the weight of the tracker.

As stated in Chapter 4, the weight requirements of the tracker is not including the casing, while the weights from Chapter 2 probably is. This means that there should be some margin to add a casing.

The weight of the tracker would be well under the required weight specified in the thesis. There is room to further improve the weight and the tracker, with the casing have potential to be light enough to attach to lambs, enabling tracking of the entire herd.

### 7.1.5 Requirement coverage

In Chapter 4 as set of stakeholder requirements were originally set as the requirements for the system developed in the thesis. To summarize the coverage of each of these requirements Table 7.1.3 shows the coverage of each requirement based on the results of the test cases. The table shows **SH-2.3** and **SH-2.4** as not covered, since they are out of scope. Also **SH-1.3** is not covered as a retransmission feature was never implemented. If a sheep, which is the current gateway, walks out of LTE coverage, then fails to transmit its messages, those messages would be lost.

TC \ SH	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	2.1	3.1	3.2	3.3
1.1	×			×				×		×		
1.2										×		
1.3					×		×	×				
2.1	×		×			×			×	×	×	
2.2	×	×	×									
2.3												
2.4												
3.1	×	×	×	×		×	×	×		×		
3.2			×			×			×	×		
4.1										×		

Table 7.1.3: Stakeholder requirement coverage matrix. ■ indicates the requirement is covered, while ■ indicates that the requirement is not covered, ■ indicates that the requirement is covered, but some test results are unstable.

While a full coverage would be preferred, it is also worth noting that the thesis never intended to



---

implement a fully industrialized solution. The solution presented in this thesis should be considered a proof of concept and a lot of additional work is required to make it viable as a product.

## 7.2 Performance parameters

The solution shows promise in terms of energy conservation, but it is valuable to discuss the performance parameters of this solution. Performance parameters are all tunable parameter that may affect the performance of the system. Either in terms of energy usage or robustness.

### 7.2.1 Application specific timing parameters

The first important parameter is the timing parameters from the application. Including the sensor poll interval, transmission interval and election interval.

The most energy spent by any of the trackers will be spent on searching for GNSS fixes or communication with the internet. The mean current when using the GNSS antenna is about 30 mA which is quite a lot [1]. Due to how hot and warm starts work, it might be energy conserving to search for GNSS fixes more frequently. This will keep the GNSS receiver in a hot state, meaning that the total energy consumption may be reduced because of shorter fix times. The GNSS interval may be tuned without changing the reporting interval and it may be interesting to see how much the tuning of this parameter affects the system performance.

The reporting interval decides how often the gateway will connect to the internet and transmit its data. LTE is the second most energy consuming operation. Since measurements are timestamped, nodes may report their state as seldom or as often as they like, and the gateway may store those data points for quite a long time before transmitting. This means that the transmission interval will affect how often new data is available to the cloud, while the polling interval will decide how fine grained the available datapoints are. Mapping out how much the reporting interval affects the solution may also be an interesting topic for further studies.

The election interval will most likely not affect the energy usage of the application significantly. All election messages are transmitted via NeoMesh. The energy usage of NeoMesh is not heavily reliant on the frequency of data transfers [1]. NeoMesh does have limitations regarding how much data can pass through the network. This means that if elections happens too frequently, compared to the scheduled transmission intervals of the NeoMesh network, the network may become congested. The NeoMesh configuration is discussed a bit more on its own in a later section.

### 7.2.2 LTE parameters

The LTE performance is heavily affected by the PSM parameters that are chosen for the application. For all firmware images in this thesis a pTAU interval of 1 hour was requested as well as an Active Time of 10 seconds. Since the solution does not expect any data from the cloud the active time could be set to 0 seconds, resulting in a lot less time spent with the LTE radio powered on. This would most likely affect the power usage in all solutions. Choosing an Active Time that fits the application is important for high energy efficiency.

The pTAU timer chosen was based on the transmission interval expected by the system. All trackers, including the nodes that are not responsible for any cloud communication, are required to resend TAU messages to stay connected to the cellular network. This means that even though trackers in node mode does not transmit any data they still send the TAU message every hour. If a much longer pTAU interval is chosen<sup>1</sup>, compared to the expected transmission interval, this may lead to some additional energy conservation from the trackers in node mode. Since, under the

---

<sup>1</sup>pTAU intervals are ultimately decided by the operator of the network where the device is connected. Meaning that not all pTAU intervals are available on all networks.

---

assumptions of three sheep sticking together, nodes outweigh gateways at least 2 to 1, this may lead to a significant reduction in energy usage of the system as a whole.

### 7.2.3 NeoMesh parameters

While the rest of the system does its work the NeoMesh module stays on and tends to the mesh network. The configuration of the NeoMesh module will also affect the system performance, possibly in a dramatic way. As seen from [1], the configuration of the NeoMesh module will vary the constant current draw from as little as a few micro amperes to many hundreds. Given that the mesh module will be powered on at all times this configuration will affect all trackers in the entire network. No real work was put into mapping out how the increase or decrease of the transmission interval of the mesh protocol affects the stability or reliability of the system. This would be a very interesting topic to explore as it will create a better understanding of how slow the node-to-node communication may be, maybe in relation with how often the nodes are expected to report their state.

NeoMesh includes support for hibernation on its mesh modules with remote wake ups. This was not explored at all in this solution, but it might have been useful to further reduce the energy usage of trackers in node mode.

## 7.3 Shortcomings and possible improvement

While the scope of the thesis is quite tight, there are a few features that did not get implemented, which may have lead to a performance increase.

### 7.3.1 GNSS

When a node is promoted to a gateway it will become its responsibility to report the position of the cluster. If the node have not been a gateway before it means that the GNSS search will start cold. This will lead to a long time spent on getting the first fix for every newly elected gateways. If the previous gateway is still in the network a handover routine could be implemented to reduce the impact of cold starts. The previous gateway could send its up to date GNSS data over the mesh network to the new gateway. The issue can also be solved by the gateway broadcasting the GNSS position over the network when ever a fix is achieved. This will result in a bit more overhead on the mesh network, but will also shorten the time spent on first fix for any node that might be promoted in the future. Another remedy would have been to add Assisted GNSS (A-GNSS) to the system, which collects up-to-date satellite parameters from the internet, based on things like the LTE cell that the device is in. A-GNSS would have increased GNSS performance overall at the cost of some more cellular activity.

The system has had a generally poor GNSS performance. There are samples where the GNSS search was completed, but most of the measurements in the database are without position data. It is difficult to pinpoint why there were so few GNSS fixes and there was not enough time to fully debug and fix the issues. GNSS is a complex system, affected by many parameters as well as the conditions of the signals received. For this solution it may be caused by a too low timeout of the GNSS search, so fixes could never be found. Since a GNSS search is performed, this issue did not affect the energy usage tests, only the performance of the solution.

The system specification says that a GNSS node must be elected, in a distributed manner. For this system the trivial solution of having the gateway also handle GNSS was chosen. This does not have to be the case and it would be a useful feature to decouple the gateway and GNSS functionality. Decoupling GNSS and gateway could be done by holding a secondary election where a different desirability factor could be chosen for the different elections.

---

### 7.3.2 Desirability factor

The implemented solution used the current battery voltage as the desirability factor when an election occurred. A very interesting topic of further study could be to see if other parameters could be incorporated into the desirability factor. Parameters like uplink quality, number of neighbours(may reduce the network congestion), number of GNSS satellites in view, etc. This may lead to an increase in energy conservation as the gateway may end up spending less energy on amplification of its LTE signals or may get a GNSS fix way faster.

### 7.3.3 More tracker hardware variants

The system specification indicated that there could have been more than one type of tracker hardware. While the current tracker may be light enough to attach to lambs, there could have been developed two sets of trackers. One larger tracker, with more battery capacity, designed to always be a gateway and one smaller tracker designed for lambs. The lamb tracker could have a much smaller battery and be designed to only activate if the lamb got separated from its mother. Such a solution may have been a good way to track the entire herd, but takes into account the fact that an ewe can carry a lot more than its lambs.

## 7.4 Notes about mesh networks

Since the mesh network is a central component of the system it is important to shed some light on how it performed. The NC2400 UART interface was quite easy to work with, but it may be possible to further increase the energy conservation of the system if the mesh module was tailored towards this sort of solution.

For example: A large part of the communication with the NC2400 was the polling the module for an updated neighbour list. The coordinator module is dependent on being notified on changes to the neighbour list so it must be polled constantly. The UART peripheral is quite energy hungry, so the constant polling leads to a waste of energy if no changes happen to the network topology. To combat this a way to be notified about the changes to the neighbour list would probably really help the performance of the system as a whole, for example by a programmable interrupt line.

As the research for this thesis started NeoMesh was the only time scheduled mesh network solution the author could find, which has sufficiently low energy usage. Another possibility for further energy conservation would be the creation of a more tailored mesh protocol or SoC. Such a system may have functionality like programmable interrupts for neighbour list updates and other central events. It could even include an application processor, like the nRF9160, but with the mesh network built in instead of the 4G and GNSS modem. Seeing that the module spends more time interacting with the mesh network than with the 4G modem that might lead to considerable improvements to solutions utilizing the concept proposed in this thesis. This leads to a very interesting topic for further studies; an application processor with tighter coupling with a low-power time scheduled mesh network.

Another issue that was encountered in this thesis was the issue of data transmission over NeoMesh. Using NeoMesh the maximum payload size of a single transmission is 21 bytes. Since the latency of the network being directly related to the energy consumption of the node, packing as much data into a single transmission was important. While this was not a large issue it certainly posed a problem when it came to the transmission of GNSS data. From the nRF9160 GNSS data is represented as 2 doubles(8 bytes each), and 4 floats(4 bytes each) for latitude, longitude, altitude and two accuracy parameters. Having to split up GNSS data is undesirable, because it introduces the issue of having to get two packets for complete information and it will further congest the network. For this system it was solved by truncating latitude and longitude and removing altitude all together. In a sheep tracking system this may not be a large issue, but for other scenarios it may heavily impact system performance.

---

## 7.5 Beyond sheep tracking

While the design in this thesis is centred around sheep tracking, the principle outlined in the thesis is applicable to other use-cases. The most general application of energy conservation using mesh network would be in general asset tracking, sheep tracking is in some sense just asset tracking as well. The only condition for the principle to work would be that nodes are naturally clustered. Given a scenario where nodes are not clustered the mesh network would become useless and the energy cost of the mesh network would not be justifiable.

There are also scenarios where clustering of nodes do happen, but the solution is not optimal, for example in warehouses or factories. If access to mains power is easy and stationary gateways can be placed, this will most likely outperform the sort of system described in this thesis.

In scenarios like shipping, where pallets are loaded and transferred around the world, the tracking solution could be applied. If shipping providers adopt this solution pallets or packets could be fitted with trackers that would detect the presence of other nodes and form dynamic mesh networks to conserve power. Even trucks could be fitted with a gateway node, connected to the trucks battery allowing all pallets in the truck to stay in node mode.

Asset tracking of tools and vehicles may also be a viable scenario. If tools or vehicles are stolen they have the capabilities to report their positions, but when they are at the site where they are supposed to be they could use a central, yet mobile gateway to conserve power.

There are possibly more scenarios where the concept could be utilized, trying to attempt to generalize the concept could be a very interesting topic for further study.

## 7.6 System complexity

While this solution is more a proof of concept than a fully fledged solution to sheep tracking it is worth noting the system complexity.

For comparison, the baseline application consists of about 700 lines of code, not including the code from the nRF Connect SDK. The full tracking system solution consists of about 4000 lines of code. While there is a large room for improvement in the code this should still be a fair comparison.

There are many moving parts in the system and since many of those parts are distributed the system as a whole is quite complex. This is important to note as the cost of developing, maintaining and debugging a complex system quickly becomes expensive if one considers engineering hours spent.

## Chapter 8

# Conclusion

Previous work has indicated that the usage of low-powered mesh networks in conjunction with a Low-Power Wide-Area-Network (LPWAN) transmitter may be utilized to conserve energy. Research on node-to-node communication, hybrid communication systems, and distributed leader election algorithms was inspected to get insight into how distributed systems operate. Existing sheep tracking solutions were also examined to get an overview of state-of-the art solutions on the market today.

A specification consisting of stakeholder- and technical requirements, as well as a test specification for a sheep tracking system, was proposed. Afterwards, the tracker hardware and software system was designed. The design was used to implement a functional sheep tracking system, composed of a set of battery-powered trackers, that could function as both gateways and low power nodes. The system was subsequently tested to see if any energy conservation could be achieved.

The test results indicate that, with three or more nodes in a cluster, the system may outperform a baseline solution by up to 30%. The baseline solution is based on how state-of-the-art solutions most likely function.

The resulting system shows promising results when using device-to-device communication to conserve energy. The principles outlined in this thesis can also be applied to other use cases in addition to sheep tracking.

## Chapter 9

# Further work

In Chapter 7 many improvements, extensions and interesting additional work was identified. This chapter will highlight the most important of these.

- Further improvements to the robustness of the implemented solution.
- Investigate if more advanced desirability factors can be used for better performance.
- Attempt to test out the system in practice, on real sheep.
- Investigate if the mesh network can be used to either improve GNSS accuracy or improve coverage.
- Attempt to create an implementation for other systems than sheep tracking.
- Attempt to build a more specialized mesh network protocol to improve performance of the proposed concept.
- Investigate ways to generalize the system for other asset tracking applications.

# Bibliography

- [1] O. Østby, ‘Energy cost of data uplink, GPS and mesh network communication in IoT networks’, 2021. [Online]. Available: <https://gitlab.com/n2521/ttk4550-specialization-project>.
- [2] Y. Rekdal, M. Angeloff, E. Skurdal, F. Avdem, V. Tømmerberg and T. Skeidsvoll Tollersrud, *Pamphlet - Utmarksbeite til sau*, Nortura, 2019. [Online]. Available: [https://www.geno.no/contentassets/68e6876772f147fe8bbecf28c473044f/utmarksbeite\\_sau\\_web.pdf](https://www.geno.no/contentassets/68e6876772f147fe8bbecf28c473044f/utmarksbeite_sau_web.pdf).
- [3] FindMy. ‘Find My - Home’. (2021), [Online]. Available: <https://www.findmy.no/>. (Accessed: 2022-01-24).
- [4] Telespor, *Telespor - Electronic Surveillance of Livestock*, 2021. [Online]. Available: <https://telespor.no>.
- [5] Nofence, *Nofence - World’s first virtual fence for livestock*, 2022. [Online]. Available: <https://www.nofence.no>.
- [6] Smartbjella. ‘Smartbjella - Home’. (2021), [Online]. Available: <https://smartbjella.no/>. (Accessed: 2022-01-24).
- [7] FindMy. ‘Find My - Model 2 Product Overview’. (2021), [Online]. Available: <https://www.findmy.no/nb/model2>. (Accessed: 2022-01-24).
- [8] Telespor, *Battery calculator*, 2021. [Online]. Available: <https://tintin.telespor.org/batt-4712/>.
- [9] Smartbjella, *Product Statistics*, 2021. [Online]. Available: <https://smartbjella.no/produkt/>.
- [10] Nofence, *Nofence - Virtual fence for goat and sheep*, 2022. [Online]. Available: <https://www.nofence.no/produkter/sm%C3%A5fe>.
- [11] FindMy. ‘Find My - How the Satellites Work’. (2021), [Online]. Available: <https://www.findmy.no/>. (Accessed: 2022-01-24).
- [12] Globalstar, *Globalstar - Home*, 2022. [Online]. Available: <https://www.globalstar.com/en-us/>, (Accessed: 2022-01-22).
- [13] Nofence, *Nofence - What is Nofence?*, 2022. [Online]. Available: <https://www.nofence.no/en/what-is-nofence>.
- [14] F. Wu, C. Qiu, T. Wu and M. R. Yuce, ‘Edge-Based Hybrid System Implementation for Long-Range Safety and Healthcare IoT Applications’, *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9970–9980, 2021. DOI: 10.1109/JIOT.2021.3050445.
- [15] J. Lianghai, B. Han, M. Liu and H. D. Schotten, ‘Applying Device-to-Device Communication to Enhance IoT Services’, *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 85–91, 2017. DOI: 10.1109/MCOMSTD.2017.1700031.
- [16] P. Pawar and A. Trivedi, ‘Device-to-Device Communication Based IoT System: Benefits and Challenges’, *IETE Technical Review*, vol. 36, no. 4, pp. 362–374, 2019. DOI: 10.1080/02564602.2018.1476191. eprint: <https://doi.org/10.1080/02564602.2018.1476191>. [Online]. Available: <https://doi.org/10.1080/02564602.2018.1476191>.
- [17] O. Bello and S. Zeadally, ‘Intelligent Device-to-Device Communication in the Internet of Things’, *IEEE Systems Journal*, vol. 10, no. 3, pp. 1172–1182, 2016. DOI: 10.1109/JSYST.2014.2298837.
- [18] G. Steri, G. Baldini, I. N. Fovino, R. Neisse and L. Goratti, ‘A novel multi-hop secure LTE-D2D communication protocol for IoT scenarios’, in *2016 23rd International Conference on Telecommunications (ICT)*, 2016, pp. 1–6. DOI: 10.1109/ICT.2016.7500356.

- 
- [19] B. Awerbuch, ‘Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election, and Related Problems’, in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC ’87, New York, New York, USA: Association for Computing Machinery, 1987, pp. 230–240, ISBN: 0897912217. DOI: 10.1145/28395.28421. [Online]. Available: <https://doi.org/10.1145/28395.28421>.
  - [20] A. Bounceur, M. Bezoui, M. Lounis, R. Euler and C. Teodorov, ‘A new dominating tree routing algorithm for efficient leader election in IoT networks’, in *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2018, pp. 1–2. DOI: 10.1109/CCNC.2018.8319292.
  - [21] N. Kadjouh, A. Bounceur, M. Bezoui *et al.*, ‘A dominating tree based leader election algorithm for smart cities IoT infrastructure’, *Mobile Networks and Applications*, pp. 1–14, 2020.
  - [22] S. Vasudevan, J. Kurose and D. Towsley, ‘Design and analysis of a leader election algorithm for mobile ad hoc networks’, in *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004.*, 2004, pp. 350–360. DOI: 10.1109/ICNP.2004.1348124.
  - [23] ‘LTE-M Deployment Guide v3’, Tech. Rep. [Online]. Available: <https://www.gsma.com/iot/wp-content/uploads/2019/08/201906-GSMA-LTE-M-Deployment-Guide-v3.pdf>, (Accessed: 2022-05-31).
  - [24] NordicSemiconductor, *Online Power Profiler for LTE*, 2021. [Online]. Available: <https://devzone.nordicsemi.com/power/w/opp/3/online-power-profiler-for-lte>.
  - [25] *MQTT Version 5.0*, v5.0, OASIS, Mar. 2019. [Online]. Available: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>.
  - [26] *MQTT: The Standard for IoT Messaging*, OASIS, 2021. [Online]. Available: <https://mqtt.org>.
  - [27] *Adafruit - Home*, Adafruit, 2021. [Online]. Available: <https://www.adafruit.com/>.
  - [28] *AWS IoT Core*, Amazon Web Services, 2022. [Online]. Available: <https://aws.amazon.com/iot-core/?nc=sn&loc=2&dn=3>.
  - [29] NeoCortec, *Datasheet - NeoCortec-NC2400 Wireless Mesh Network Module Series*, English, version 1.2, NeoCortec, 8th Nov. 2021, 8 pp.
  - [30] NeoCortec, *User Guide*, English, version 2.4, NeoCortec, 31st Jul. 2019, 42 pp., Release.
  - [31] NeoCortec, *Integration Manual for NCxxxx series Modules*, English, version 3.1, NeoCortec, 31st Jul. 2019, 23 pp., Release.

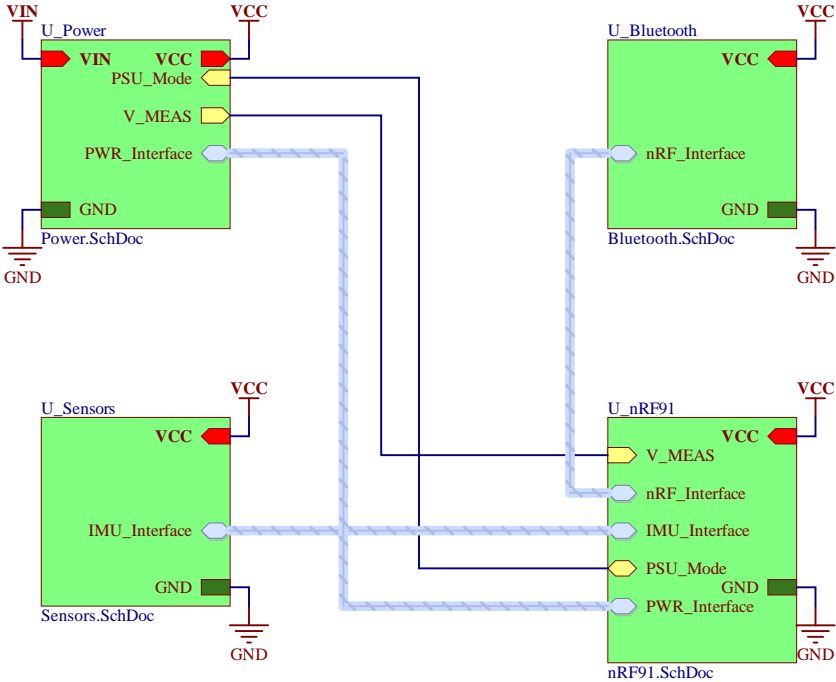
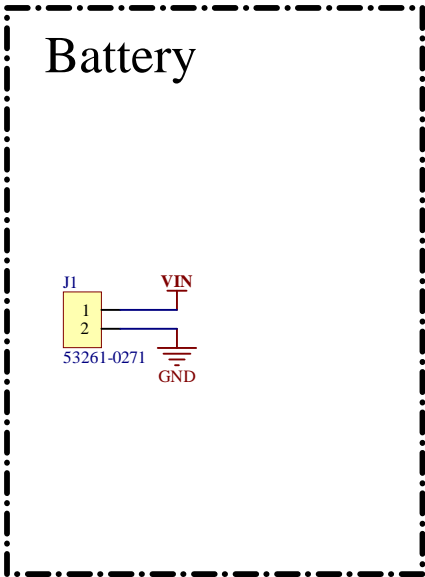


# Appendix

## Appendix A

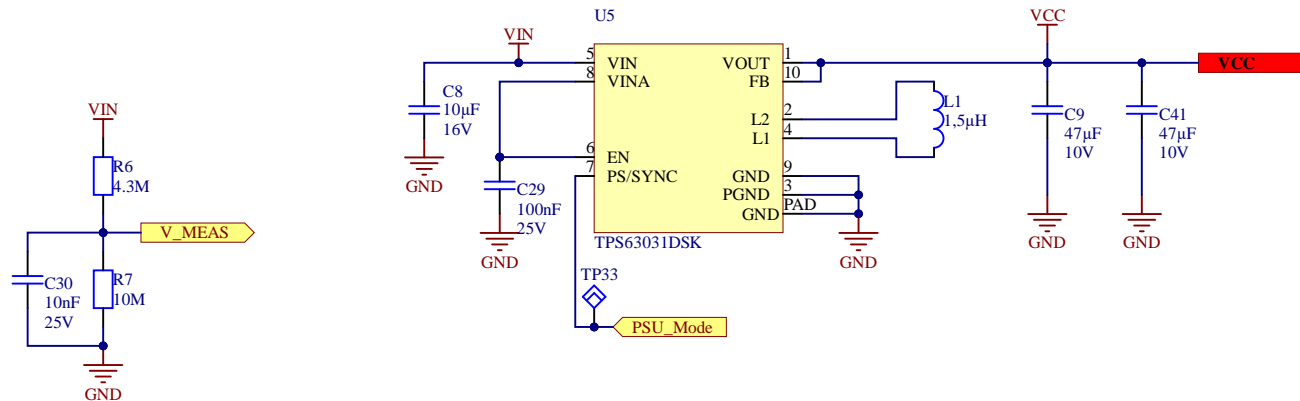
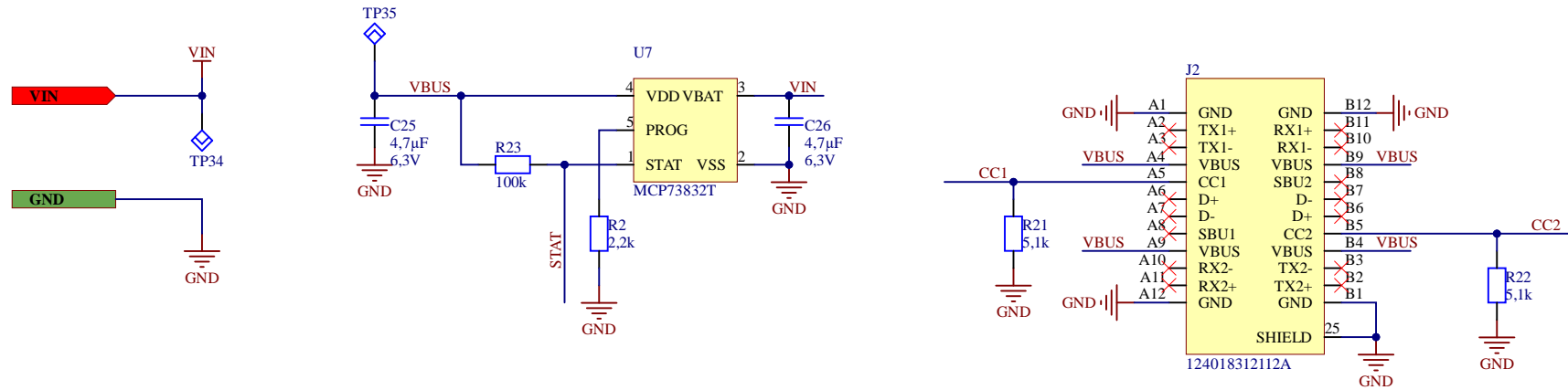
# Norbit Tracker Schematics

All pages of the schematic is not included, the included pages of the schematic should be enough to recreate a functional design.



<b>NORBIT</b>		Title	Rechargeable IoT GNSS Tracker		
		Part #	31503		
Section	Main				
Variant	Default				
Filename	Main.SchDoc				
Revision	4-1	Engineer	Adrian Opheim		Sheet 2 of 6
Date	05.07.2021				





NORBIT	Title	Rechargeable IoT GNSS Tracker		
	Part #	31503		
Section	Power			
Variant	Default			
Filename	Power.SchDoc			
Revision	4-1	Engineer	Adrian Opheim	Sheet
Date	05.07.2021			6 of 6

## Appendix B

# Logfile including several elections

```
1  *** Booting Zephyr OS build v2.7.99-ncs1 ***
2  [00:00:00.476,745] <dbg> main.main: FSM: FSM_STATE_INIT < FSM_EVT_NO_EVT
3  [00:00:00.483,856] <inf> main: AWS ID: sheep-tracker-03, size: 17
4  [00:00:00.490,386] <inf> main: Node ID: 0007
5  [00:00:00.495,086] <dbg> main.fsm_state_init: Initializing sensors
6  [00:00:00.511,169] <wrn> date_time: Valid time not currently available
7  [00:00:00.518,371] <err> main: Time is invalid, err: -61
8  [00:00:00.524,414] <dbg> main.fsm_state_init: Initializing NeoCortec module
9  [00:00:00.531,890] <dbg> main.fsm_state_init: Initializing Coordinator
10 [00:00:00.538,818] <dbg> coordinator.set_new_leader: Setting new leader: ffff
    -> 0007
11 [00:00:00.547,210] <dbg> main.fsm_state_init: Initializing LTE Link Controller
12 [00:00:00.568,786] <wrn> lte_lc: Enabling TAU pre-warning notifications failed,
    error: 65536
13 [00:00:00.577,880] <wrn> lte_lc: TAU pre-warning notifications require nRF9160
    modem >= v1.3.0
14 [00:00:00.593,994] <wrn> lte_lc: Enabling modem sleep notifications failed,
    error: 65536
15 [00:00:00.602,722] <wrn> lte_lc: Modem sleep notifications require nRF9160
    modem >= v1.3.0
16 [00:00:00.634,338] <wrn> lte_lc: Enabling TAU pre-warning notifications failed,
    error: 65536
17 [00:00:00.643,432] <wrn> lte_lc: TAU pre-warning notifications require nRF9160
    modem >= v1.3.0
18 [00:00:00.659,576] <wrn> lte_lc: Enabling modem sleep notifications failed,
    error: 65536
19 [00:00:00.668,334] <wrn> lte_lc: Modem sleep notifications require nRF9160
    modem >= v1.3.0
20 [00:00:00.718,292] <dbg> main.fsm_state_init: Awaiting LTE connection
21 [00:00:03.207,366] <dbg> main.lte_callback: Got PSM active time: 0, TAU:
    1800
22 [00:03.206,817] <dbg> main.fsm_state_init: Initializing Cloud module
23 [00:00:03.221,954] <dbg> main.set_state: FSM:[FSM_STATE_INIT -> FSM_STATE_NODE]
24 [00:00:03.229,644] <dbg> main.main: FSM: FSM_STATE_NODE < FSM_EVT_PROMOTION
25 [00:00:03.237,030] <dbg> main.set_state: FSM:[FSM_STATE_NODE ->
    FSM_STATE_GATEWAY]
26 [00:00:03.245,056] <dbg> cloud.cloud_worker_fn: Awaiting TX start
27 [00:00:04.245,117] <dbg> main.main: FSM: FSM_STATE_GATEWAY < FSM_EVT_LTE_ACTIVE
28 [00:00:04.253,356] <dbg> main.do_cloud_transmission: Transmission completed,
    sending ACKs
29 [00:00:04.261,962] <dbg> cloud.cloud_worker_fn: Connecting to AWS
30 [00:00:04.268,524] <dbg> cloud.cloud_worker_fn: Awaiting CONNACK from AWS
31 [00:00:04.275,787] <dbg> cloud.aws_event_handler: AWS EVENT: 1
32 [00:00:06.691,040] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
```

---

```
33 [00:00:06.698,394] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
34 [00:00:07.945,190] <dbg> cloud.aws_event_handler: AWS EVENT: 2
35 [00:00:07.951,416] <dbg> cloud.aws_event_handler: AWS EVENT: 3
36 [00:00:07.957,702] <dbg> cloud.cloud_worker_fn: Transmitting cloud buffer
37 [00:00:07.964,935] <dbg> cloud.transmit_cloud_buffer: CLOUD TX: <0007, 17694>
38 [00:00:07.973,632] <dbg> cloud.transmit_cloud_buffer: CLOUD TX: <0007, 17694>
39 [00:00:07.982,910] <dbg> cloud.transmit_cloud_buffer: CLOUD TX: <0007, 17694>
40 [00:00:07.991,241] <dbg> cloud.cloud_worker_fn: Disconnecting from AWS
41 [00:00:08.000,701] <dbg> cloud.aws_event_handler: AWS EVENT: 4
42 [00:00:08.006,958] <dbg> cloud.cloud_worker_fn: Awaiting TX start
43 [00:00:36.841,735] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
44 [00:00:36.849,090] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
45 [00:01:06.992,401] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
46 [00:01:06.999,786] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
47 [00:01:37.143,096] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
48 [00:01:37.150,482] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
49 [00:02:07.293,823] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
50 [00:02:07.301,208] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
51 [00:02:37.444,519] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
52 [00:02:37.451,873] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
53 [00:03:00.547,241] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_TERM_IS_OVER>
54 [00:03:00.558,776] <dbg> coordinator.set_parent_node: New parent: 0xffff -> 0
    x0007
55 [00:03:00.566,833] <dbg> coordinator.update_current_candidate: Updating
    candidate: <0007, 3702273 -> <0007, 0>
56 [00:03:00.577,362] <dbg> coordinator.update_current_candidate: Updating
    candidate: <0007, 0 -> <0007, 3702273>
57 [00:03:00.587,829] <dbg> coordinator.set_new_leader: Setting new leader: 0007
    -> 0007
58 [00:03:07.595,214] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
59 [00:03:07.602,569] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
60 [00:03:37.745,880] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
61 [00:03:37.753,265] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
62 [00:04:07.896,606] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
63 [00:04:07.903,991] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
64 [00:04:38.047,302] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
65 [00:04:38.054,656] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
66 [00:05:08.197,998] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
67 [00:05:08.205,383] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
68 [00:05:38.348,724] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
69 [00:05:38.356,109] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_UPDATED_NEIGHBOUR_LIST>
70 [00:06:00.596,252] <dbg> coordinator.coordinator_event_handler: Event handler:
    <ELECTION_STATE_COMPLETED, EVT_TERM_IS_OVER>
71 [00:06:00.607,788] <dbg> coordinator.update_current_candidate: Updating
    candidate: <0007, 3702273 -> <0007, 0>
72 [00:06:00.618,286] <dbg> coordinator.update_current_candidate: Updating
    candidate: <0007, 0 -> <0007, 3702273>
73 [00:06:00.628,753] <dbg> coordinator.set_new_leader: Setting new leader: 0007
    -> 0007
74 [00:06:08.499,389] <dbg> neocortec.uart_data_callback: UART_RX_BUF_RELEASED
```

---

---

75 [00:06:08.506,744] <dbg> coordinator.coordinator\_event\_handler: Event handler:  
<ELECTION\_STATE\_COMPLETED, EVT\_UPDATED\_NEIGHBOUR\_LIST>



# Appendix C

## Database printouts

### Reported neighbours of 7 from removal to reintroduction

This query is querying for all neighbours reported by tracker 7 between when it was removed from the cluster and when the NC2400 was reset, as described in Chapter 6.

message_id	timestamp	sheep_id	neighbour_id
6107	2022-05-27 12:18:37+02	7	6
6107	2022-05-27 12:18:37+02	7	5
2231	2022-05-27 11:30:12+02	7	5
17293	2022-05-27 11:18:15+02	7	
26417	2022-05-27 10:18:12+02	7	
37419	2022-05-27 09:18:14+02	7	
11832	2022-05-27 08:18:16+02	7	
105	2022-05-27 07:18:12+02	7	
52599	2022-05-27 06:18:13+02	7	
62961	2022-05-27 05:18:15+02	7	
48642	2022-05-27 04:18:15+02	7	
12358	2022-05-27 03:18:13+02	7	
6510	2022-05-27 02:18:14+02	7	
38687	2022-05-27 01:18:16+02	7	
17980	2022-05-27 00:18:18+02	7	
4154	2022-05-26 23:18:14+02	7	
15971	2022-05-26 22:18:15+02	7	
13658	2022-05-26 21:18:15+02	7	
10679	2022-05-26 20:18:13+02	7	
13322	2022-05-26 19:18:15+02	7	
15500	2022-05-26 18:18:15+02	7	
6073	2022-05-26 17:18:18+02	7	
33577	2022-05-26 16:18:15+02	7	
8414	2022-05-26 15:18:15+02	7	
46052	2022-05-26 14:18:18+02	7	
30148	2022-05-26 13:18:48.059439+02	7	
49147	2022-05-26 12:43:43+02	7	
55765	2022-05-26 11:43:44+02	7	
56221	2022-05-26 10:43:46+02	7	
62371	2022-05-26 09:43:47+02	7	
44365	2022-05-26 08:43:44+02	7	
17803	2022-05-26 07:43:44+02	7	
46353	2022-05-26 06:43:47+02	7	
55447	2022-05-26 05:43:43+02	7	

---

35825		2022-05-26	04:43:44+02		7		
50236		2022-05-26	03:43:46+02		7		
35080		2022-05-26	02:43:48+02		7		
63021		2022-05-26	01:43:44+02		7		
1874		2022-05-26	00:43:47+02		7		
13774		2022-05-25	23:43:48+02		7		
43249		2022-05-25	22:43:44+02		7		
55115		2022-05-25	21:43:44+02		7		
5421		2022-05-25	20:43:47+02		7		
13077		2022-05-25	19:43:48+02		7		
45775		2022-05-25	18:43:44+02		7		
48790		2022-05-25	17:43:46+02		7		
40500		2022-05-25	16:43:48+02		7		
50907		2022-05-25	15:43:50+02		7		
47271		2022-05-25	14:43:46+02		7		
16041		2022-05-25	13:43:48+02		7		
23176		2022-05-25	12:44:14+02		7		
10881		2022-05-25	11:43:44+02		7		
44135		2022-05-25	10:43:46+02		7		
1751		2022-05-25	09:43:48+02		7		
62180		2022-05-25	08:44:14+02		7		
23561		2022-05-25	08:18:41+02		7		6
23561		2022-05-25	08:18:41+02		7		5
(57 rows)							

