NTNU

Norwegian University of Science and Technology

Faculty of Information Technology and Electrical engineering

Department of Engineering Cybernetics

Specialization Project

Candidate:	Oskar Østby
Course	TTK4550 - TTK4551 Specialization Project
Thesis title (Norwegian)	Energikostnad av data oppkobling, GPS og mesh kommunikasjon i IoT nettverk
Thesis title (English)	Energy cost of data uplink, GPS and mesh network communication in IoT networks

Thesis description:

When farmers let their sheep and lambs onto the pasture, during grazing season, they sometimes fit the sheep with a battery powered tracking device. When the season ends the sheep are gathered, but a handful of sheep and lambs are lost every year.

This thesis will explore the possibility of using an ultra low power mesh network to reduce energy consumption in a set of sheep trackers. By amortizing the cost of transmission and positioning over many nodes it would allow for trackers small enough for lambs, higher reporting intervals or longer battery life.

The tasks will be:

1. Perform initial research on existing solutions and technologies used for sheep tracking and state of the art IoT power conservation techniques.
2. Propose an abstract tracker and test setup to allow measuring energy usage of gateway transmissions, GNSS search and mesh communication.
3. As far as time permits, implement and perform the proposed tests.

Start date:	2021-08-20
Due date:	2021-12-20

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

NTNU
Kunnskap for en bedre verden

Department of Cybernetics and Robotics

TTK4550 - Specialization Project

Energy cost of data uplink, GPS and mesh network communication in IoT networks

Author:
Oskar Østby

December, 2021

Table of Contents

List of Figures iii

List of Tables iv

1 Introduction 1

1.1 Background . . . 1

1.2 Motivation . . . 1

1.3 Scope . . . 2

1.4 Thesis structure . . . 2

2 Research 3

2.1 Mesh technologies . . . 3

2.2 Power saving techniques in IoT . . . 3

2.3 Existing solutions . . . 4

3 Theory 5

3.1 Mesh networks . . . 5

3.2 NeoMesh . . . 6

3.3 Quick comparison of LTE-M, LoRa and NB-IoT . . . 7

3.4 LTE power saving . . . 8

3.5 Satellite IoT . . . 9

3.6 Global Navigation Satellite Systems . . . 9

4 Proposed design 10

5 Test specification 11

5.1 Gateway energy usage . . . 11

5.2 Measuring IoT mesh network energy usage . . . 11

5.3 GNSS positioning system . . . 12

6 Test setup, equipment and instruments 13

6.1 Instruments . . . 13

6.2 NeoCortec NC2400 . . . 15

6.3 Thread . . . 16

6.4 Gateway - 4G . . . 18

6.5 GPS . . . 19

7 Results **20**

7.1 NeoMesh 20

7.2 LTE-M 24

7.3 GPS 26

7.4 Thread 27

7.5 Error sources 28

8 Discussion **30**

8.1 Viability of Thread 30

8.2 Energy usage in NeoMesh 30

8.3 LTE-M performance 30

8.4 Cost of Positioning 30

8.5 Mesh communication vs. LTE-M 31

9 Conclusion **33**

10 Further work **34**

Bibliography **35**

List of Figures

3.1 Illustration of a full mesh topology 5
3.2 Illustration of a partial mesh topology 5
3.3 Illustration of a resilient mesh topology with many paths 6
3.4 Illustration of beacon scanning in NeoMesh, from [19] 6
3.5 Transmission synchronization in NeoMesh, from [19] 6
3.6 Packet flow in a NeoMesh network, from [19] 7
3.7 LTE Attach Sequence, User Equipment(UE) is the LTE modem 8
4.1 Proposed sheep tracker components 10
6.1 Overview of N6705B power analyser display 13
6.2 Image of the Nordic Semiconductor Power Profiler Kit 2 14
6.3 PPK test setup 14
6.4 Screenshot of the Power Profiler application 15
6.5 A NeoMesh node ”station” 16
6.6 A NeoMesh node with makeshift attenuation 16
6.7 The test setup when testing Thread 17
6.8 The two development kits used when testing Thread 18
6.9 nRF9160 development kit used for testing LTE 19
7.1 A single NeoMesh beacon and scheduled transmit event 20
7.2 An overview of all NeoMesh radio events 21
7.3 Transmitting a message with maximum payload size 21
7.4 NeoMesh sleep current, no radio events 22
7.5 Overview of NeoMesh performance given percentage of transmissions with payload 24
7.6 Current profile of intial LTE connection 25
7.7 A 210 bytes CoAP transmission from PSM 25
7.8 A 21 bytes CoAP transmission from PSM 26
7.9 Time spent to get fix 27
7.10 Current profile when searching for GPS fix 27
7.11 Full Thread Device current profile 28
7.12 Minimal Thread Device current profile 28

List of Tables

6.1 Different configurations used when evaluating NeoMesh 15
7.1 Overview of average energy usage of NeoMesh radio events 22
7.2 Performance of different NeoMesh configurations 24
7.3 Overview of energy usage of LTE radio events 26

List of Abbreviations

Abbreviation	Description
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
A-GNSS/GPS	Assisted GNSS/GPS
IoT	Internet of Things
NB-IoT	Narrow Band - Internet of Things
LTE	Long Term Evolution
LTE-M	Long Term Evolution - Machine type
SoC	System on Chip
4G	4th Generation (broadband cellular network technology)
BLE	Bluetooth Low Energy
LoRa	Long Range (proprietary radio technology)
LoRaWAN	Long Range Wide Area Network
IEEE	Institute of Electrical and Electronics Engineers
IPv6	Internet Protocol version 6
MTD	Minimal Thread Device
FTD	Full Thread Device
3GPP	3rd Generation Partnership Project
TAU	Tracking Area Update
DRX	Discontinous Reception
eDRX	Extended Discontinous Reception
PSM	Power Saving Mode
IC	Integrated circuit
PMIC	Power Management IC
MCU	Micro controller unit
CSV	Comma Separated Values (data format)
PPK	Power Profiler Kit
DUT	Device Under Test
SDK	Software Development Kit
CoAP	Constrained Application Protocol
UART	Universal Asynchronous Receive Transmit
CTS	Clear To Send
UE	User Equipment
GPRS	General Packet Radio Service
SIM	Subscriber Identification Module
ULP	Ultra Low Power

1 Introduction

This section will describe the background, motivation and the scope of this thesis. In the end of the section, the structure of the thesis will be presented.

1.1 Background

When a sheep farmer releases their sheep into the wild during grazing season, the sheep usually stray far from where they were originally released. As the end of grazing season comes, the farmer needs to collect all their sheep and bring them back to the farm. The process of gathering the sheep is a large ordeal, as the sheep can be spread over an extremely large area, and often a team of people is employed to make this as efficient as possible.

A seasoned farmer may have a good idea where to find their sheep and they will most likely have dogs to help them, but gathering the sheep is still a challenge. Especially if no tracking devices are used, since there is no way to pinpoint their location. Employing a large team of people will be helpful, but it is both expensive and the process still time consuming.

To streamline the sheep gathering process one can attach a tracker to each sheep in the herd and use a mobile device to track their location when collecting the sheep. This scenario will be the baseline of this thesis.

A few immediate design requirements are apparent: the tracking unit will be required to have a battery life that is longer than at least the average grazing season. The tracking unit will need some way of communicating with a cloud solution and tracking unit will have to be light enough so that the sheep can carry them, without being tired out.

The issue with such a tracker is that communicating via f.ex. a 4G radio or searching for GNSS position is quite expensive in terms of energy usage and having a full herd of sheep transmitting 4G or searching for GNSS fixes will require a lot of energy in total.

1.2 Motivation

In the industry today sheep tracking is somewhat used, but to make sure the tracker has enough power it requires a large battery. The battery is heavy, so it will only be able to be attached to a sheep, not a lamb [1], [2], [3].

This is an issue because a lamb can get away from it's mother, f.ex. in a predator attack, and thus we will lose track of the lamb since no tracker is attached to it. If a sheep is lost or dies the lambs are mostly considered lost as well because the lambs cannot survive without their mother. Farmers often just refer to a mother and its two lambs as a *set*, since the lambs are so dependent on their mother. Losing a few sheep or lambs each season is normal, but with a tracker, that is small enough to be attached to a lamb, we could potentially reduce these losses.

Now lets work under the assumption that a set of sheep, i.e. one mother and two lambs, stay close to each other during their stay at the pastures. If we deploy an ultra low power, short range mesh network to enable communication between nodes(trackers are considered nodes when in a mesh network), in addition to the existing data uplink to the cloud, we may be able to achieve very low energy usage. This allows us to reduce the size of a trackers included battery, so that we may attach them to lambs as well.

This is the main motivation of this thesis; is it possible to utilize a low power mesh network, in conjunction with a gateway protocol and positioning system, to reduce energy usage of a sheep tracker?

The sheep tracking scenario is a nice practical example to help set the scope when exploring this sort of energy conservation. In section 8 a few points about generalization will be presented.

1.3 Scope

The world of IoT is very complex and there are heaps of considerations and design decisions to make. To be able to finish the work on this thesis a few limitations to the scope must be made.

A simpler subset of cases in the scenario must be chosen to focus on, as developing and testing all cases in the sheep tracking scenario is not feasible in a single thesis. First let us assume we have a single set of sheep, that are each fitted with the proposed tracker with mesh, GNSS and gateway functionality. And assume that this set sticks together, like they will under normal conditions. How much does it cost for each sheep to collect GNSS data and send it to the cloud, compared to how much it costs if one tracker collects GNSS for the set, and one dedicated gateway receives this information, via mesh, and then transmits it to the cloud? How much energy does it cost to keep the mesh network active? What happens to the integrity of the mesh network if a lamb wanders off for a moment? How does this energy cost vary between different mesh protocol?

If this case yields reasonably good results, we could potentially increase the amount of energy saved. This may be done by using a larger amount of nodes, if additional sets of sheep are within mesh network range. With a larger amount of nodes, an increase in network traffic will follow, to maintain the mesh network and routes. More sets of sheep will also mean that our mesh network may be constantly changing, which will be much more complex to model. This will be a possible area for more research.

To further specify the scope of the thesis we make a few assumptions that will help to reduce the amount of time spent on technicalities and deployment specific issues.

First of all, we assume that the trackers' gateways always have coverage. The effects of a loss of connection to the cloud could be mitigated by storing data in a non-volatile storage, but we will not delve any deeper into this issue.

We assume that a mounting solution is available to attach trackers to the sheep and lambs. A lamb will grow quite a lot during grazing season [4] and it might not be trivial to find a way to attach the tracker to f.ex. their necks, but this is definitely outside the scope of this thesis.

We assume that security and provisioning of nodes, i.e. handing out network keys or certificates to nodes, is not an issue. This will be a highly deployment specific problem, it will depend on many choices of technology, but in the thesis we will assume that every node is pre-provisioned and that all communication is secure.

1.4 Thesis structure

In section 2, Research, an overview of existing solutions and literature about the subject will be presented. Then, in section 3, a brief introduction to theory that will be used in the rest of the thesis.

Following, in sections 4 and 5, Proposed design and Test specification, a proposal of a design for a sheep tracker together with a specification of how to test it.

In section 6, Test setup, equipment and instruments, the proposed design will be implemented and tested. Section 7, Results, will present our findings. These findings will be discussed in section 8 before a conclusion is presented in section 9. Lastly a brief listing of further work is presented in section 10.

2 Research

It is important to get an overview of existing solutions and available technologies, before we start to propose a design for a sheep tracker. In this section an overview of research, solutions and techniques, from the area of mesh networking and sheep tracking, is presented. This will later be used as a foundation for the design of a sheep tracker.

2.1 Mesh technologies

From the start of the research it became clear that there is no abundance of research on the topic of homogeneous low power mesh networks. As far as the author could find there are no papers directly researching the possibility of a completely homogeneous mesh network.

In a survey on relevant IoT technologies, ZigBee, Bluetooth Low Energy(BLE) Mesh, Thread, Lightweigth Mesh(LW Mesh) and LoRa Mesh, are mentioned as the large relevant technologies in the wireless mesh networking scene [5]. The survey also briefly mentions the basic working principle of the different technologies. It quickly becomes clear that the industry standard is based upon the IEEE 802.15.4 standard [6]. The IEEE 802.15.4 is a standard that defines a general solution to wireless mesh networking by defining a set of node responsibilities and roles.

One widely used technology is the BLE Mesh. The BLE Mesh is an extension of the very popular BLE technology [7]. BLE is also tried and tested, over many years, which makes extensions of this protocol easier to adopt. BLE Mesh consists of nodes that have different capabilities and responsibilities within the mesh network. Some nodes are responsible for relaying messages, while others are ultra low power nodes that only report state, actuates devices and mostly spends time in a sleep state. A lot of studies have been conducted on the BLE Mesh technology. Among them a study that takes a deep dive into the power usage of the protocol [8]. This paper presents a model of the power usage of a BLE Mesh low power node. The paper predicts that a low power node can have up to a 15.6 month long battery life, with a 235 mAh battery. The same study also concludes that the BLE Mesh is not suited for networks were relay nodes do not have access to grid power. This makes BLE Mesh of little value to this thesis.

Another candidate is the Thread protocol. Thread is a low-power mesh protocol designed for IoT, it uses IPv6 for addressing and uses the same physical interface as BLE [9]. The specification is developed by the Thread Group, but the actual implementation is developed by Google. Thread networks are based on two fundamental node types, the Minimal Thread Device(MTD) and the Full Thread Device(FTD). These fundamental nodes will in turn take on a role, like a router, leader, sleepy end device or border router.

A last candidate is the NeoMesh protocol. Developed by a company named NeoCortec. The NeoMesh protocol is a hierarchy free protocol where all the nodes are equal. It works by timing transmission and reception periods within the network as a part of the networks configuration. Each node sends out beacon transmissions at a known interval and when a beacon is received it synchronizes the nodes. Using this technique the nodes may sleep for the majority of their lifetime, only awakened by a timer on scheduled intervals, for transmission and reception.

2.2 Power saving techniques in IoT

From the research presented above, it can observed that many IoT solutions rely on a node that can act as a gateway, either within the network or from the network to the cloud. Since this solution is so widely used, research has been done to try to reduce energy usage of the gateway [10].

The most common way that the mesh protocols save energy, is by relying on the gateway being powered. Almost every mesh technology mentioned above, uses some sort of gateway system that is used to enable end devices to sleep for the majority of their life time. This is a clever way to

enable power saving, but it creates the requirement for a heterogeneous mesh deployment. One of the positive benefits, are that the sleeping node may transmit at any time with very low latency.

In general, it seems that power saving in IoT mesh networks are primarily gained by using a power efficient protocol. There is rarely a need for further power saving.

2.3 Existing solutions

There exists some solutions today that focus on tracking sheep, or more generally livestock. In Norway we have at least three companies that supplies this solution, Smartbjella [1], Findmy [2] and Telespor [3]. Another, U.S. based, company called Digitanimal [11] have also been reviewed.

Smartbjella, Telespor and Digitanimal, have all adopted LTE-M and Narrow Band - IoT and allows for up- and downloads. Smartbjella and Telespor advertise many years of battery life with one transmission every 24 hours, and almost 3 months battery life with transmissions every 5 minutes [12], [13]. Digitanimal reports a maximum of 6 months battery life [14].

Findmy uses satellite to report data, where the normal data rate is one report per day [15]. They also advertise 2-3 seasons of battery life, before a battery replacement is required [16].

To the extent of the authors knowledge none of these solutions actually uses any mesh technology to save power, but relies on large batteries and power saving features of NB-IoT or LTE-M. The closest thing that could be found is that Findmy have added Bluetooth functionality, used to locate sheep when gathering them at the end of the season.

There is also some solutions used in academia [17], but it often seem like the data is logged and collected by hand and not via a cloud solution. An interesting find is that almost 50% of studies that are done with sheep uses GNSS sensors [18].

3 Theory

To make sure that the reader have sufficient knowledge of the terms and technologies described in this paper this section will explain terms and technologies that are central to the paper.

3.1 Mesh networks

A mesh network is a network topology where every node in the network *may* connect to any other node. There are two types of mesh topologies, the partial mesh- and the full mesh topology. In the full mesh topology all nodes can communicate directly to any other node in the network. This is illustrated in figure 3.1.

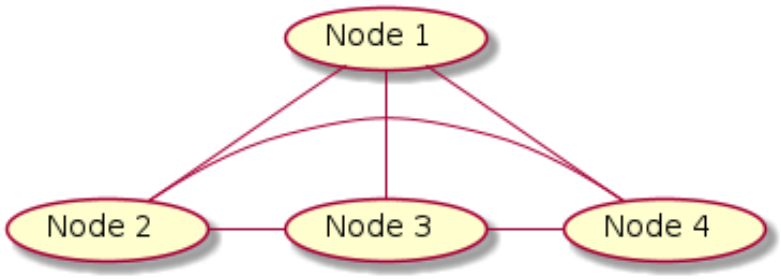

Figure 3.1: Illustration of a full mesh topology

The more common of the two is the partial mesh topology which is similar to the full mesh topology, but the nodes are not fully connected with every other node. In this topology every node is still reachable by every other node, but packages sent between them might need to be relayed by one or more intermediary nodes. This is illustrated in figure 3.2 When a package is relayed by an intermediary node it is often called a *hop*. So direct communication is one hop, while each intermediary node will add one hop to the packet path.

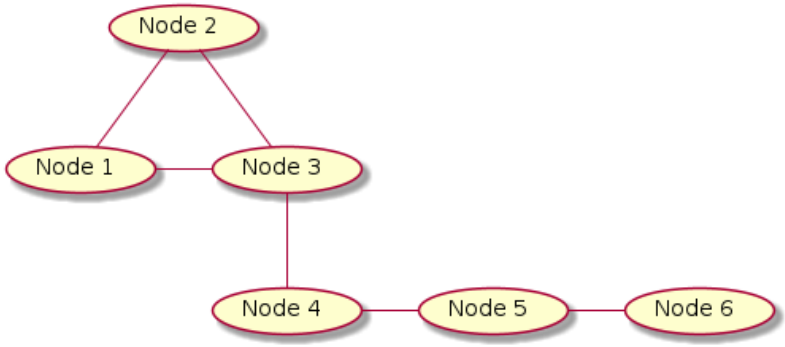

Figure 3.2: Illustration of a partial mesh topology

In a mesh network there may be many paths from one specific node to another. This in turn means that many nodes may be disconnected before a path between two nodes disappears. This makes mesh network resilient to broken paths and nodes disconnecting from the network like shown in figure 3.3.

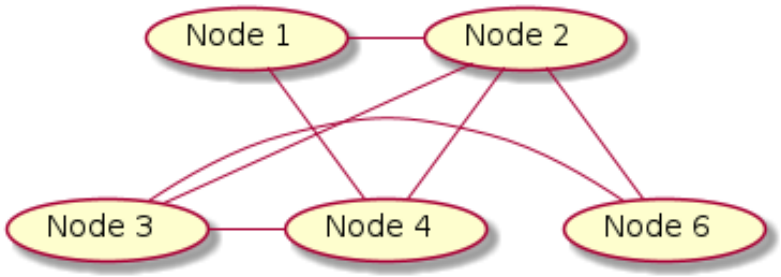

Figure 3.3: Illustration of a resilient mesh topology with many paths

3.2 NeoMesh

NeoMesh is a protocol that will be used a lot in this thesis, as stated in section 2, it is a hierarchy free mesh network [19].

The working principle of the NeoMesh protocol is to use time synchronization to allow nodes in the network to sleep most of the time. To make sure all nodes are synchronized a requirement is that every node in the network have the same configuration. The configuration includes the rate of beacon and transmissions in the network, which is needed for every node to calculate how often it should sleep.

A beacon transmission is sent at an interval between 1 and 30 seconds. When a node is powered on and not included in any network it will start to transmit beacons and scanning for beacons from other nodes. After powering up a node will perform a series of full beacon scan. When a full beacon scan is performed, the node will keep its radio on for the entire period of the beacon transmission interval. This means that any beacons transmitted within range should be received. When a beacon is received the node will synchronize itself with the node that transmitted the beacon and communication can start. These beacons and a full scan is shown in figure 3.4, here the two nodes are synchronized after the full beacon scan.

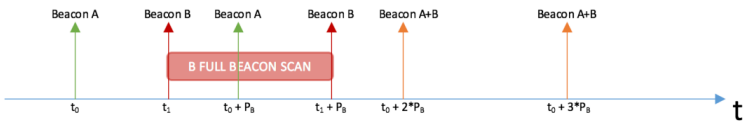

Figure 3.4: Illustration of beacon scanning in NeoMesh, from [19]

Within the beacon transmission each node adds information about the next time it plans on transmitting data. So when a node receives a beacon it will start a timer so that it is awoken during the other nodes next transmission. It is illustrated in figure 3.5, after the second beacon is transmitted the two nodes are synchronized and can transmit data between each other.

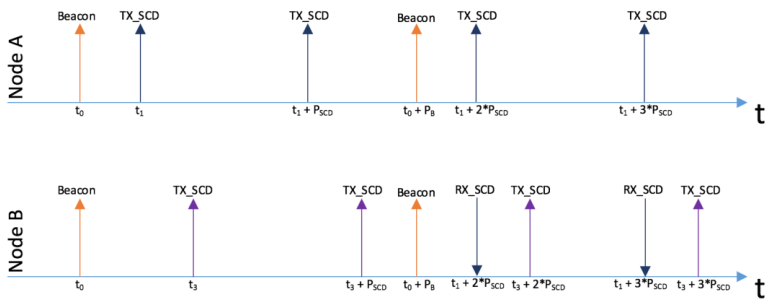

Figure 3.5: Transmission synchronization in NeoMesh, from [19]

When nodes are synchronized to each others scheduled transmits they are considered neighbours. Now we may transmit packages across the network. As stated in [20] the maximum payload of a package is 21 bytes, which will be used later as a baseline for data transmissions. When a node transmits a package to a different node it just transmits the package during its scheduled transmit, knowing that its neighbours will be listening. When a packet is transmitted to a node that is note ones immediate neighbour, it is routed through the network based on a routing table that each node holds. An illustration of how packets travel in the network is shown in figure 3.6.

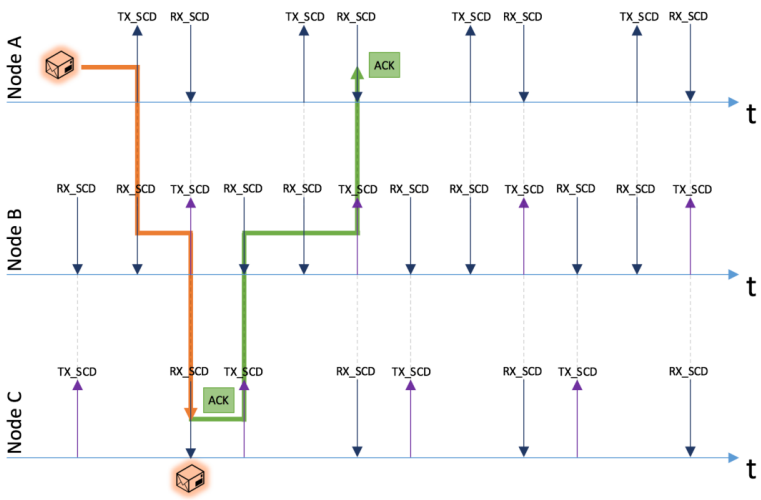

Figure 3.6: Packet flow in a NeoMesh network, from [19]

Network maintenance in NeoMesh is baked into the scheduled transmission events. Even though beacon transmissions is necessary for normal operation, after the network have been formed, it is still transmitted. This is to handle the cases where nodes move around and the topology changes. The beacons will then be used to update neighbours throughout the networks lifetime.

If a node does not receive any data from its neighbour on, a configurable, number of scheduled transmissions the neighbour is considered lost.

NeoMesh exposes an application layer to the user which will be used to control the nodes. The application layer supports different message types, different means to control nodes and other useful features. In this thesis the acknowledged packet type will be used for transmitting data.

3.3 Quick comparison of LTE-M, LoRa and NB-IoT

Both LTE-M and NB-IoT are specified by 3rd Generation Partnership Project(3GPP). LTE-M was first introduced in 3GPP Release 12 and NB-IoT in Release 13. In this paper LTE will be used as a group term for both technologies, even though LTE consists of much more then LTE-M and NB-IoT. LTE is useful because it uses existing telecommunication infrastructure, more specifically the 4G infrastructure [21].

The LTE-M standard is optimized for larger data rates and is suitable for non-stationary nodes. This includes f.ex. transportation or supply chain tracking [21]. The NB-IoT technology is a single 200 kHz narrow band(NB) network, it is more suitable for stationary nodes that have long battery life and low cost.

LoRa, short for Long Range, is a low-power, long range wide-area network. LoRa offers even lower data rates and due to no collision detection it is best suited for low cost, low-power, and low density applications. LoRa uses its own proprietary gateway system which runs LoRaWAN (Long Range Wide Area Network). Gateways can be purchased and set out by consumers or companies, so the network coverage has quite good [22].

So in summary, LTE-M supports the highest data rates, NB-IoT is in the middle and LoRa supports the lowest data rates. LTE-M also has support for moving nodes, while NB-IoT is optimized for stationary nodes, no literature have been found to indicate that LoRa does not support moving nodes.

3.4 LTE power saving

To be able to have meaningful discussions about the use of LTE in IoT a proper introduction to power saving features of the standard is required. When a device with an LTE modem wants to reach the internet it first needs to *attach* to the network. This attachment procedure is a set of complex steps, illustrated in figure 3.7, involving many systems and is out of scope for this paper.

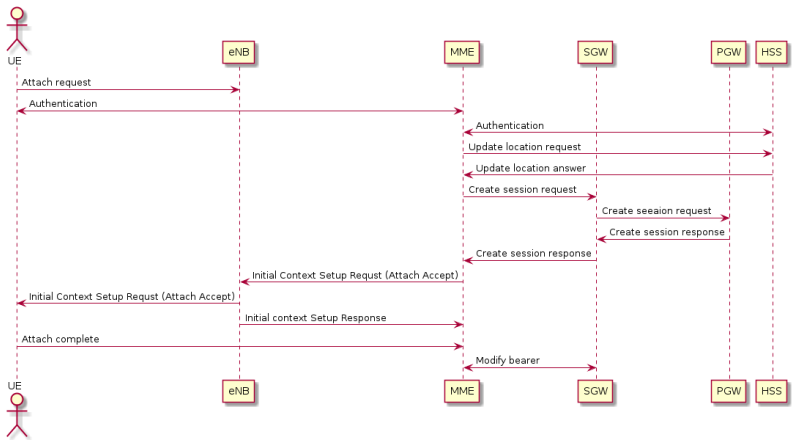

Figure 3.7: LTE Attach Sequence, User Equipment(UE) is the LTE modem

It is sufficient to say that the process requires a lot of transmitting back and fouth between the modem and the LTE base station. When the device is attached it should have received an IP address and may reach the internet over LTE. While attached to the LTE network and to be able to receive data the modem must keep the radio powered on at regular *paging intervals*[23]. These paging intervals are used be the base station to send information about data for the device. In addition to this periodic *Tracking Area Update*(TAU) messages must be sent to the base station to stay attached.

In a ULP setting sending these messages and keeping the modem radio on during paging intervals may consume and order of magnitude more current on average than what is acceptable.

To be able to use these LTE technologies in IoT we need to remove or sufficiently reduce these operations to achieve adequate power consumption. This can be achieved by using one of the different power saving features that LTE-M and NB-IoT supports. Fist we have Power Saving Mode(PSM), which negotiates an increased pTAU interval and an *active time* with the base station. The pTAU interal can be everything from a few minutes to a few hours which greatly reduces average power usage. The active time is a period after the device has sent a pTAU where it must be ready to receive paging instances so it is reachable by the base station, the active period can be from a few seconds to a few minutes. An important point here is that it is the base station that decides which pTAU and active time the device gets. A device may *request* a pTAU and active time, but the base station might not grant it.

Another power saving feature is the Extended Discontinous Reception(eDRX) which is an extention of the Discontinous Reception(DRX) used by many smartphones today to save battery power. eDRX may be used in conjuction with PSM to achieve greater power saving. In short it greatly increases the interval where the device is not listening to the network allowing the modem to switch of and save power. Like PSM the interval of eDRX must be negotiated with the base station and the base station decide wether the interval is granted or not.

3.5 Satellite IoT

There are some companies that have utilized satellite to communicate with the cloud. This possibility was discovered too late to be considered in this paper, but it might be a possible replacement for LTE. Satellite IoT comes with a few benefits. It will have a much larger coverage as it is not dependent on any base station on earth. It is also reliant as it often builds on a very fault tolerant L1-Band satellite service [24], [25]. Satellite IoT will work in mostly the same way as LTE, you acquire a SIM card for authentication and a modem that supports communication with the satellites and off you go.

3.6 Global Navigation Satellite Systems

Global Navigation Satellite Systems or more often GNSS is a system that is used in many different contexts to acquire position and velocity data for a piece of equipment. A quick intro to GNSS is required to discuss the power usage of the GNSS in section 6, but it will be kept brief as it will quickly fall out of scope for this paper.

GNSS systems are implemented by using satellites. A set of 20 to 30 satellites are in orbit for each system which broadcasts periodic signals. The signals themselves contains orbital information about the satellites, as well as data about the satellites on board atomic clock [26]. When a receiver can observe the signals sent from a satellite it is said that the satellite is *in view*. A receiver will have to listen to these signals, extract the orbital and clock correction data, and then use this data to estimate the distance to each satellite and any clock drift (caused by the receivers comparably terrible clock). When the clock error and relative distance to each satellite is acquired, the receiver can calculate its position on the earth, by using the satellites orbital data [26]. This method is based on the relative distance to each satellite in view and the clock error, this means that a minimum of four satellites are required to be in view for the entire duration of the estimation process. In short, to get a GPS fix we require four satellite in view.

The clock correction and orbital data transmitted by the satellites are transmitted at a rate of 50bps. An entire navigation message (almanac, ephemeris, clock data and some control data) consists of 25 frames of 1500 bits. This means that downloading all the data takes 12.5 minutes. It is also required that the entire package is downloaded at once, so if a device loses satellite coverage it must start over. The almanac, containing coarse orbital parameters, is valid for 6 days, the ephemeris, precise orbital parameters, is valid for 2 hours and the clock data needs to be continuously up to date(but is also transmitted more often).

Having no valid orbital or clock data and having to collect all this data before getting a fixed is often referred to as a *cold* start. If orbital data is valid, but our initial position guess is no longer we refer to this as a *warm* start. When orbital data, clock correction and our initial guess is valid we call this a *hot* start. Which often takes just a few seconds.

Another important factor in this process is that GNSS estimation requires an initial ”guess” at the receivers position. This is often solved by placing the initial guess at the center of the earth, but this will lead to a longer time spent to get a fix.

To reduce the time to first fix(TTFF) a support system called Assisted GNSS (A-GNSS or A-GPS where GPS is used) is implemented. This system allows a GPS receiver, often one that can communicate with GPRS base stations, to download navigational messages and initial position guesses from the internet. The data is based on the GPRS mast it is communicating with. The mast have a known location and can continuously hold updated navigational messages. Using A-GPS will let our equipment go straight to a warm or hot start without having to download any navigational data.

4 Proposed design

In this section we will explore an imagined sheep tracker's components and functionality.

The main components of such a tracker would be a microcontroller, a GNSS module, a gateway module like 4G or LoRa, a mesh module, a battery with a power management IC(PMIC) and some sort of mounting solution. A component diagram showing this proposed design is shown in figure 4.1.

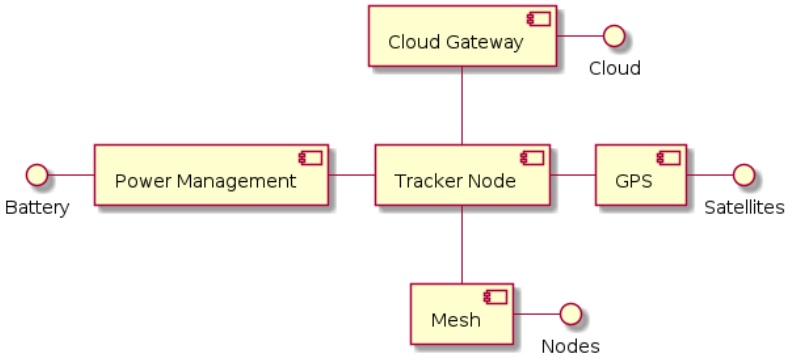

Figure 4.1: Proposed sheep tracker components

In most cases an MCU would also include a gateway module or mesh module, but for the sake of generality these are two separate modules in the design.

This proposed design of a tracker is not very specific. With enough time and engineering effort it is possible to build a tracker by the modules that we have presented in this section. While it would be interesting to see how the tests would run on an actual implemented tracker, this is out of scope for this thesis.

The microcontroller will be responsible for coordination of each of the peripheral units. Its job is to communicate with the cloud, via the gateway module, extract GNSS data from the GNSS module, coordinate sleep and communicate with the other mesh nodes. The complete software solution will not be discussed in more detail in this thesis.

The GNSS module will require an antenna that can receive the GNSS signals it should support sleeping and/or a configurable duty cycle to save power.

The gateway module will be responsible for handling connections and data transmissions to the cloud. It should controllable by the MCU.

The mesh module will be responsible for maintaining the mesh network and to transmit data to other nodes in the network.

The power management system should be designed with ultra low power in mind with a support for the voltage range of the batteries used.

5 Test specification

In this section a set of test scenarios will be described that will be used to investigate the comparative power usage of mesh protocols and network communication. First a presentation of the test setup of the gateway technology will be presented, then the mesh network setup.

Before we go into the details of how we want to test the two different communication methods it is important to discuss their differences.

5.1 Gateway energy usage

To have sufficient data to compare mesh and gateway energy consumption we need a few important metrics. Network maintenance will be measured, it will include the cost of connecting to the network and cloud, but also how much energy it takes to keep that connection alive.

Next a measurement of the cost of sending data to the endpoint will be required. To measure this we will look at the total amount of data sent compared to how much energy is spent. This is done to account for the differences in application layer protocols that can be used for the gateway protocol. Then we can analyse the findings and see how well we can amortize the cost of sending data over the cost of network maintenance.

Then we will combine these two measurements into an average energy usage. This will be done by averaging the cost of data transmissions over the time where the module is idle. This should give us an average cost of using the gateway, given the frequency and size of payloads transmitted.

The practical test setup will be simple. Program some sort of gateway unit with a program that sends a known sized package to the endpoint. Then identify the network maintenance part and the data transmission part of the measurement. When the two parts of the transmission are identified one can calculate the energy usage per byte from the transmission part of the data.

5.2 Measuring IoT mesh network energy usage

When measuring the power consumption spent on mesh communication there are a few parameters that needs to be kept in mind.

The topology of the mesh network will affect the energy performance of the network. For example a full mesh topology may require more energy to maintain the network than a line topology, where all nodes only have two neighbours, or vice versa. This might be protocol dependent. To make sure that we account for this measurements in a small variety of topologies should be done. Interesting topologies might include the full mesh, where every node can communicate in one hop. The single line topology, where all the nodes are in a single line, meaning that end-to-end communication will reach every single node in the network. The star topology where a single node will be the choke point of every message in the entire network. Which topologies that should be measured will depend on the protocol used and how that protocol handles network maintenance and routing.

The main ways that the mesh network will spend its energy is by a) sending messages, b) receiving messages, c) relaying messages and d) doing network maintenance.

To be able to create a sufficiently detailed model of the energy usage all four of these points should be analysed. The reason that relaying messages has its own point is that, depending on the protocol, relaying messages might require more or less computations energy than just sending and receiving a message. The network maintenance will also be heavily dependent on the protocol that is used as most protocols use their own routing algorithms.

5.3 GNSS positioning system

GNSS is a complex system and have many complex external parameters that decide its performance. To get ideal data about GNSS performance testing the receiver in as realistic conditions as possible is important. Since it is not always feasible to deploy a set of realistic test a heuristic approach could be taken. Try to find or create a data set that is as representable as possible and use that as an estimate. Since we have a very clearly defined scenario to work against we can use this to decide how such a data set should look.

We need a dataset where the GNSS coverage might be varying quite a bit. Sheep may graze in an open field or in a dense forest. This will have a large impact on GNSS fix times as dense foliage will block GNSS signals. We also need some data on how much current is spent when the GNSS is actively searching for a fix. This should be as simple as starting a GNSS search on the GNSS module and measuring the current usage over a time interval. Since the GNSS current usage is not very dynamic this should be a decent estimate.

From these data sets a mean time spent to get fixes can be used to then calculate the average energy consumption on each fix.

6 Test setup, equipment and instruments

In this section we will discuss the implementation of each module of the sheep tracker, how the software was designed, and present the methods use to test each module.

6.1 Instruments

A Keysight N6705B DC Power Analyzer will be used to measure the current and voltage output in some experiments. This instrument features up to 24.80 micro seconds sampling rate and 0.025% + 8 nA accuracy. To make measurements the device under test will be connected to a channel on the instrument and an output voltage will be configured. When a measurement is started the instrument will sample the current output over time and the data can be extracted from the instrument as a CSV file.

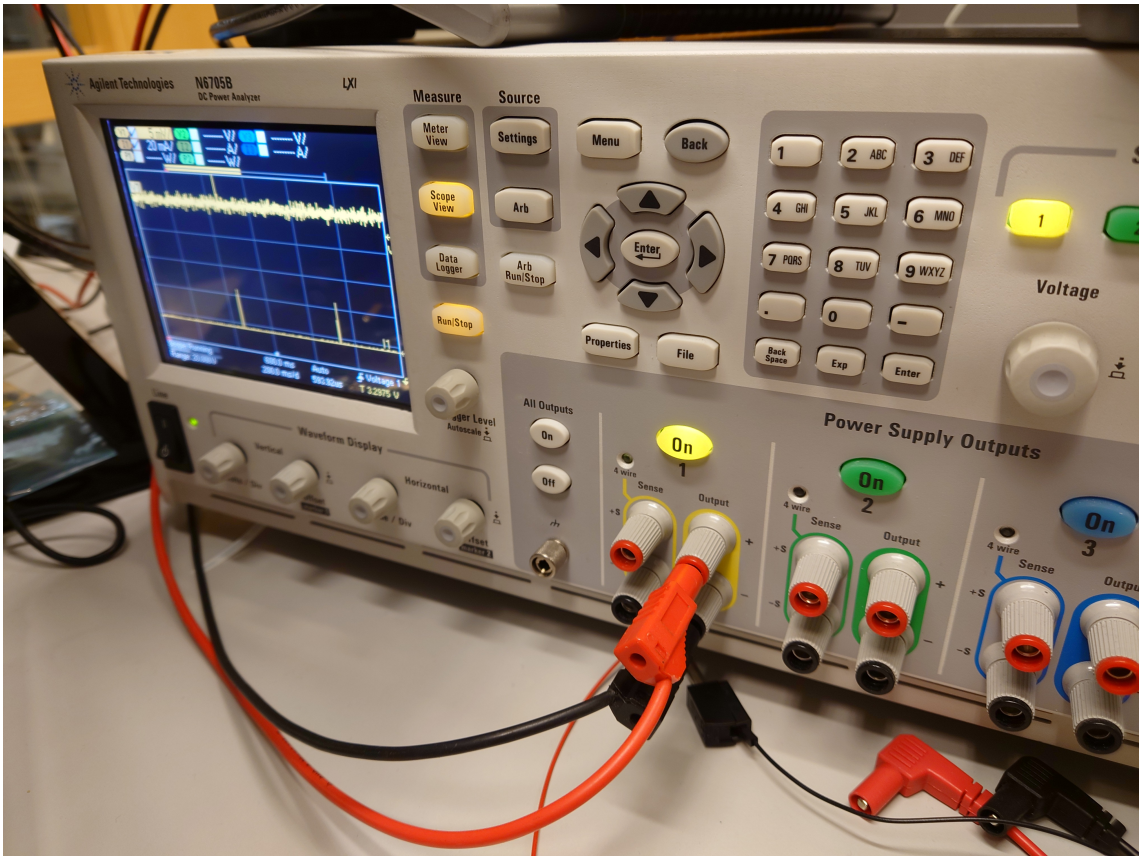

Figure 6.1: Overview of N6705B power analyser display

The N6705B is a great instrument to measure the active current profile, but with the IoT nodes larges jumps in current it does not have a large enough dynamic range to be able to measure sleep currents accurately enough. To supplement the measurements the Nordic Semiconductor Power Profiler Kit(PPK) 2, shown in figure 6.2, will be used to do additional measurements on sleep currents. The PPK features accuracy all the way down to 200 nA and a large dynamic range. This makes it possible to measure accurate current profiles of IoT devices.

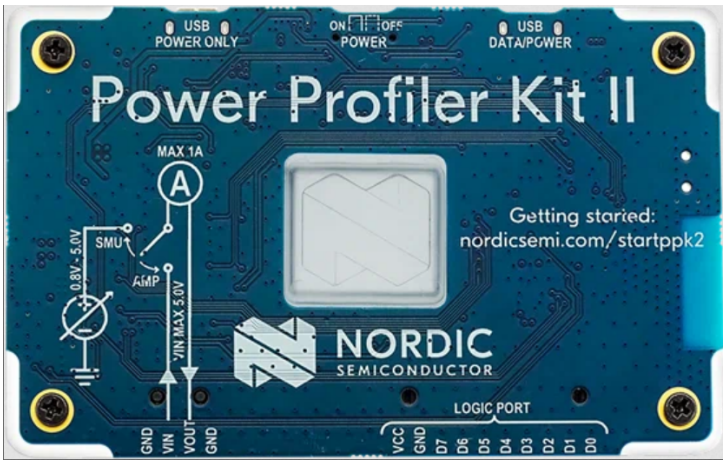

Figure 6.2: Image of the Nordic Semiconductor Power Profiler Kit 2

The PPK supports two modes, an ammeter mode and a supplied meter mode. In ammeter mode it simply samples the current flowing through the kit, while in supplied meter mode the PPK supplies a variable current between 5 and 1.8 volts and measures the current draw. A sampling rate of 100kHz gives us a very nice view of the radio events that we measure.

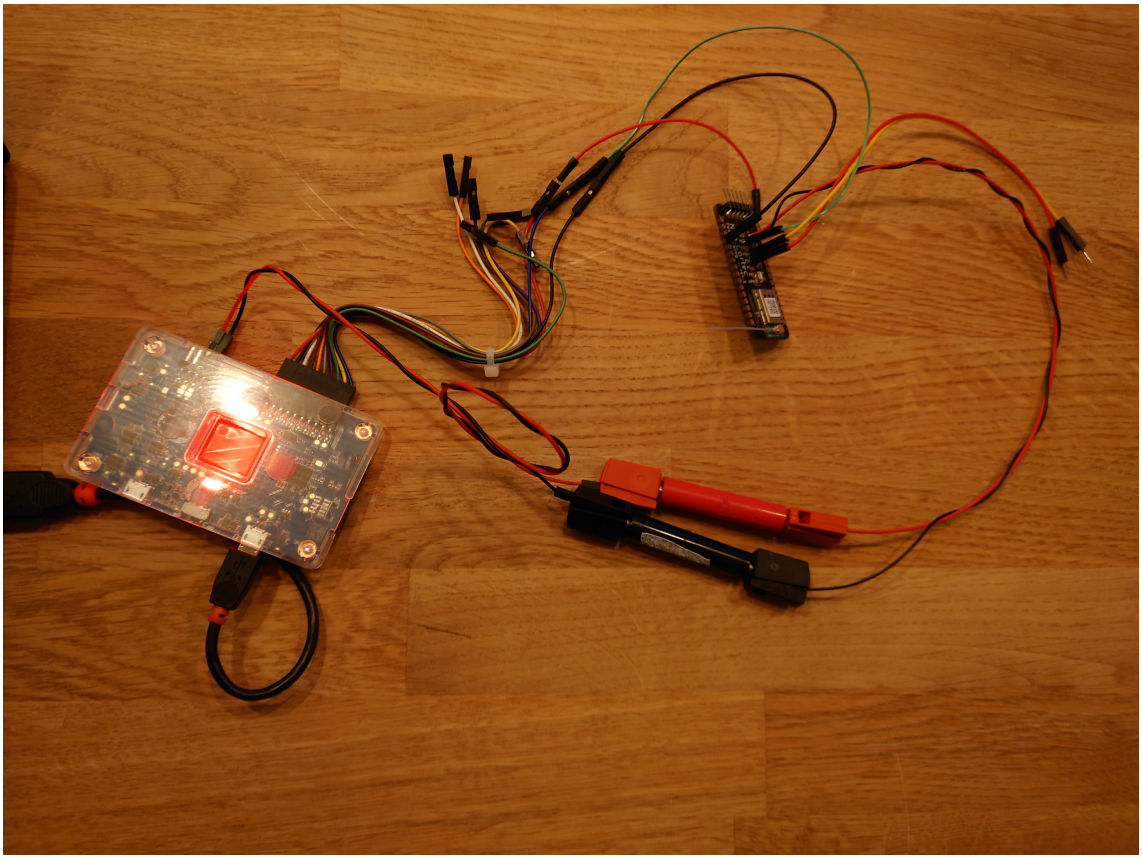

Figure 6.3: PPK test setup

As shown in figure 6.3 we connect the DUT with twisted power wires and our desired ”marker” signals are connected to the IO header.

After a measurement is completed the data can be viewed and exported from a graphical interface. Shown in figure 6.4

Config N	TX Power	Scheduled Data Rate	Beacon Rate	Beacon Full Scan Rate
1	1dBm	1s	1s	500
2	1dBm	15s	15s	500
3	1dBm	15s	15s	1000
4	1dBm	30s	30s	500
5	1dBm	30s	30s	1000
6	-30dBm	1s	1s	500
7	-30dBm	30s	30s	1000
8	1dBm	5s	30s	1000

Table 6.1: Different configurations used when evaluating NeoMesh

Figure 6.4: Screenshot of the Power Profiler application

6.2 NeoCortec NC2400

For the mesh protocol to compare against LTE-M we will utilize the NeoCortex NC2400 module[27]. The NC2400 provides a fully functional mesh network stack and can easily be configured using the included configuration software. The module provides a mesh network called NeMesh, explained in section 3. NeoMesh is suited for this thesis because it does not rely on a coordinator node to create it's network and is developed with ultra low power in mind. These two features will make it an ideal mesh network to test our hypothesis as all nodes should be homogeneous. In this paper the FWNC2400C will be used as it is a simple breakout board for the NC-2400 module. The module uses a 2.4GHz radio and comes with the *NeoMesh Protocol* stack. The included NeoMesh Protocol is a mesh protocol designed for flexibility and reliability.

A module is set up by simply soldering on a RS232 header and a set of power pins. Then we connect 3.3V and we configure it using the NeoCortec Configuration tool. To make sure we have reliable data and make sure the module is as isolated as possible a few key configurations are done.

Since the NeoMesh performance is very dependent on the configuration of timing parameters like the scheduled transmit rate and beacon rate we want a set of configurations that we can use in evaluation. In section 6.2 a set of configurations are presented that will give a broad overview of the performance of NeoMesh in different configurations.

To be able to test out different mesh topologies the modules were placed with sufficient distance between them so that the required topology was formed. The node under test was connected to

the power analyser and the other nodes were moved or had their antenna attenuated to form the required topologies. A node test ”station” was made as shown in figure 6.5 and placed around the office space. figure 6.6 shows an attempt to attenuate the signal to reduce the radiated signal, reducing the signals power, but not affecting the node’s transmit power. This method proved unsuccessful and the transmit power of nodes that are *not* under test will be reduced in stead.

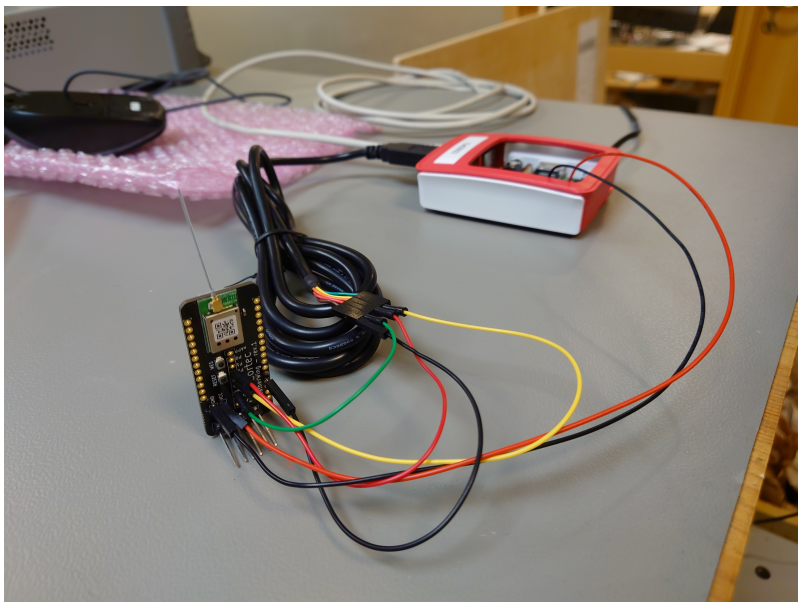

Figure 6.5: A NeoMesh node ”station”

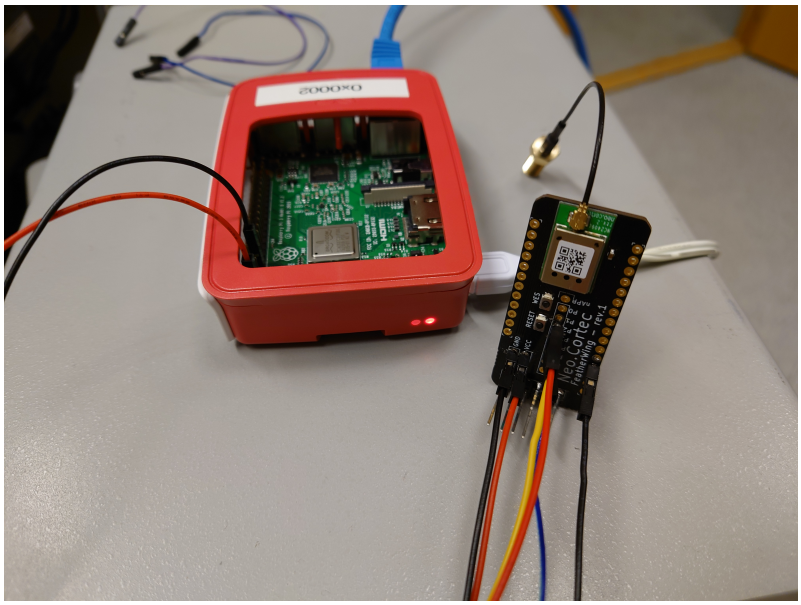

Figure 6.6: A NeoMesh node with makeshift attenuation

6.3 Thread

We will also analyse the Thread protocol. To set up a simple thread network a few nRF52840 development kits will be used. The nRF52840 is built for BLE, but also supports OpenThread. The code for the project will be a modified example from the nRF Connect SDK. This will mostly be an attempt to see if it is at all possible to use a IEEE 802.15.4 based protocol for this purpose.

In the nRF Connect SDK a OpenThread sample can be found that implements a CoAP server [28] and client [29] over OpenThread. This will be the example used for testing the OpenThread network. The CoAP server exposes two endpoints, one ”provision” endpoint for pairing a client with a server, and one ”light” endpoint which is used to set the state of an LED on the server node. The server will always be configured as a Full Thread Device, mostly because the sample code is not configure to also work as a Sleepy End Device, but it should be possible to implement the server as a SED as well. The CoAP client is configured to include Minimal Thread Device capabilities. The client can therefore be a SED in the Thread network. The client sends a provision request to a server to pair with it, then it can send ”/light” requests to the server to toggle LEDs.

Since the sample is meant to be used as an example on how to start development using the OpenThread stack on the nRF52, some modifications was made to achieve comparable deep sleep power usage. On both the server and the client node the shell and UART was disabled, this peripheral draws large amounts of current and should not be active at all times. The system power management and tick-less idle options from the Zephyr kernel was also enabled to further reduce the power usage. Then the development kits were flashed with software. When testing the kits the nRF52 was disconnected from the interface MCU that is used to flash the kits, and the N6705B was used as an external power supply to measure both voltage and current usage. This resulted in 5 μA of average sleep current on the SED. It should be possible to optimize the nRF52 for even less power usage, but this would be sufficient as a baseline for these tests.

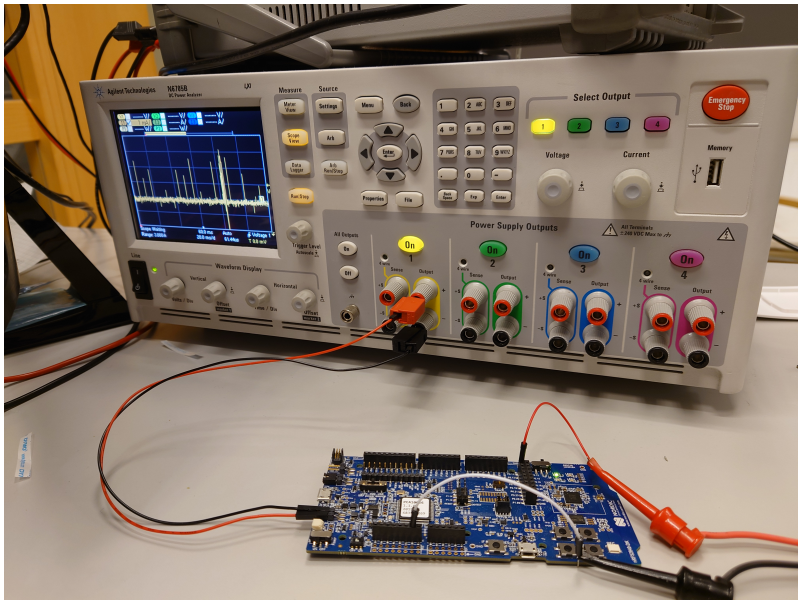

Figure 6.7: The test setup when testing Thread

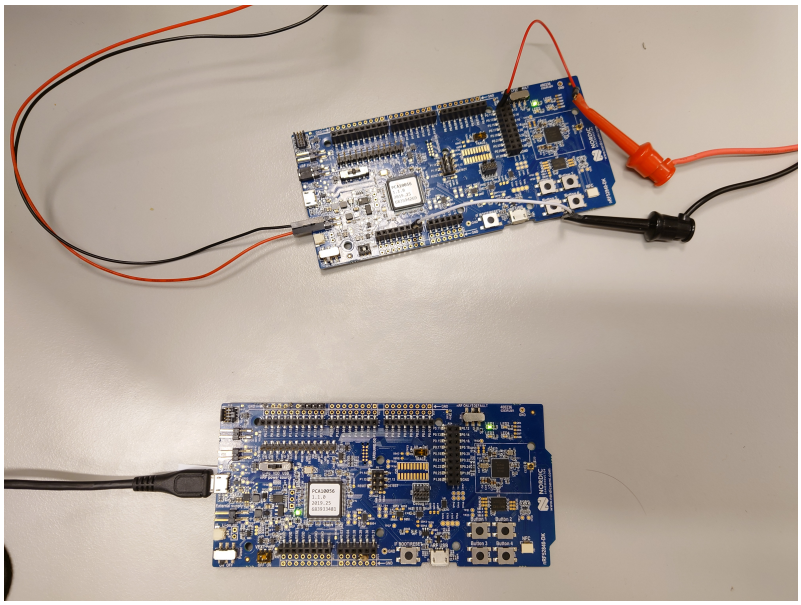

Figure 6.8: The two development kits used when testing Thread

6.4 Gateway - 4G

The nRF9160 is another SiP that provide an application processor with an integrated LTE-M/NB-IoT modem and a GPS receiver. The nRF9160 development kit will be used for testing 4G connectivity and GPS in this paper. To be able to connect to the 4G network we need a SIM card, for this paper a SIM card from 1NCE is used. It may roam in Norway and it supports PSM mode.

A basic CoAP client sample(from the nRF Connect SDK) [30] was used as a starting point for our measurements. It is a simple example that turns on the modem, connects to LTE-M and then requests a timestamp from a CoAP server. To get a broad set of tests a lot of changes was made to test different uses of the 4G modem. Firstly the UART peripheral was disabled to save power and the system power management and tickless idle features where enabled in Zephyr. This brought the average sleep current down to about 5 μA which is a sufficient baseline, though further optimizations could be done here as well. For the first test a CoAP transmission was done every 20 seconds, leaving the modem on between each transmission while the CPU was sleeping. In the next test the modem was disabled and enabled between each transmission and in the last test PSM was enabled. PSM enables the modem and the base station to agree on an active and inactive period which the station should be ready for packets from the modem.

When the modem is turned off and on between transmissions the modem needs to connect to the radio tower for each cycle. When PSM is enabled the modem does not have to constantly reconnect to the tower, but may still almost completely turn off. This saves a lot of power and is the main feature that makes 4G LTE-M viable for IoT, considering it allows for a floor current of 4 μA during PSM [31].

For reference measurements when PSM mode is enabled were acquired. Currently, in Norway, using the 1NCE SIM card the nRF9160 was handed out a 1800 seconds pTAU interval and a 60 seconds active time. This means that for all our measurements there will be a 60 second interval where the modem turns on its receiver for paging indicators. The PPK kit was used to measure this current as the N6705B only allows for a total of 30 seconds of sampling, with a sufficient sample interval.

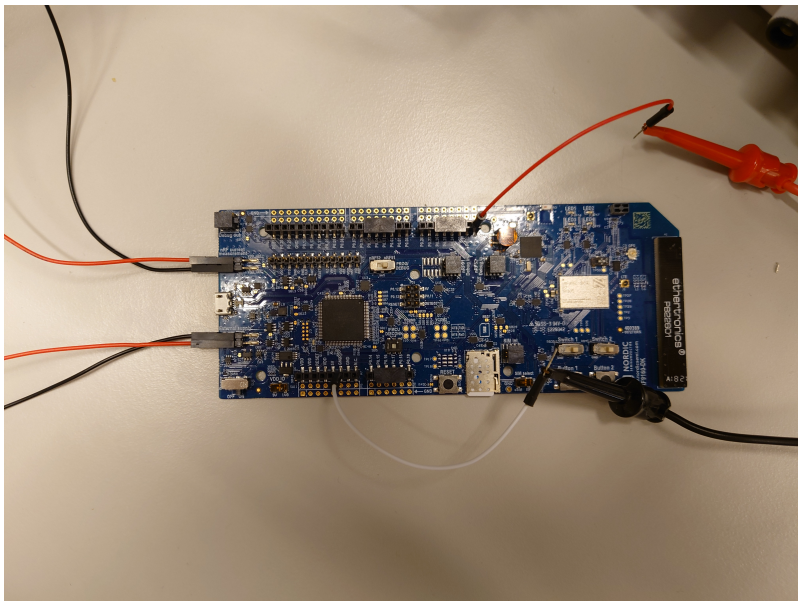

Figure 6.9: nRF9160 development kit used for testing LTE

As shown in figure 6.9 the development kit is connected to an external power supply to reduce noise. In this example we use the Agilent E3640A DC Power Supply. The power supply is connected to the P28 header. The power analyzer is set to current measurement mode and connected to the P24 header which is used to measure the current of the chip. This setup was used to measure the active times of the LTE modem.

6.5 GPS

The nRF9160 DevKit also includes a GPS receiver and antenna. This was used to test out the GPS energy usage. The goal is to measure current draw while searching for a GPS fix. The test setup is the same as in figure 6.9.

Most of the software from the LTE testing was reused. This software was already configured for optimal sleep currents. The changes that were made was that in stead of connecting to the LTE network the nRF9160 started a GPS search in stead. Then a measurement was done to acquire the average current drawn when searching for a GPS fix.

A study from the UK [17] who conducted a GPS tracking study on sheep had a GPS fix success rate at above 95% on a 30 minute interval. With a properly configured GPS receiver it is assumed that a success rate close to this should be achievable. Based on this an assumption that each tracker manages to get a successful fix each time it searches is made.

7 Results

In this section the results from the data acquisition will be presented and error sources will be discussed.

7.1 NeoMesh

Since the NeoMesh documentation, at the time of writing, does not contain any detailed power profile some analysis was required. As described above we expect to see three kinds of radio events from the NeoMesh nodes. A beacon, a scheduled transmit and N scheduled receives, one for each neighbour. To identify the different events we first run a single node without any neighbours. This means that after the initial full beacon scans the node should only transmit beacons on a known interval and transmit on another known interval. To make it even easier the CTS interleave setting was set on the node, which means that the CTS line would only be available when a TX event happens. So using the IO ports on the PPK a dataset with the current usage and the value of the CTS pin was acquired and it is quite trivial to then identify the two events. The beacon and scheduled TX events are shown in figure 7.1.

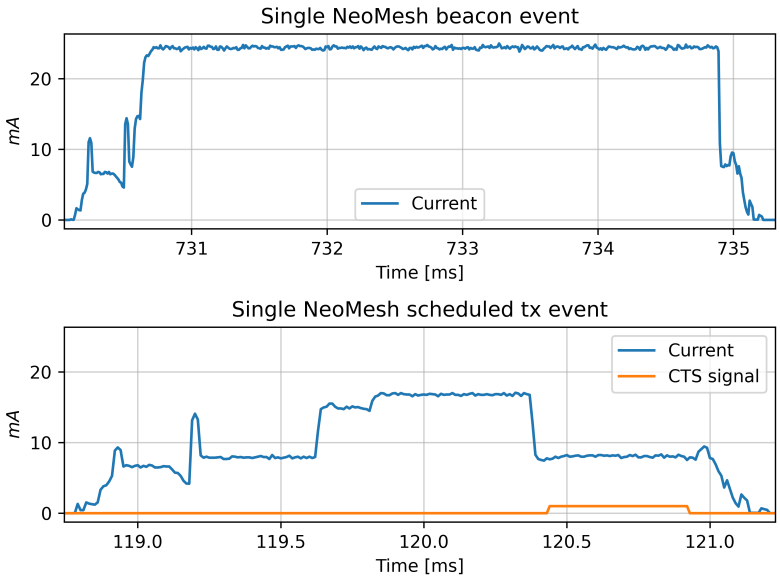

Figure 7.1: A single NeoMesh beacon and scheduled transmit event

Now to identify the last signal needed to do a full analysis, the scheduled receive event. To identify this event a second node was introduced into the system and became the neighbour node of our original node. Then we repeated the process mentioned above except the third event was identified. Then to end up with the topology that we want to run further tests on we included a third node, being neighbour only to our original node.

With this setup and the knowledge about the events we can now measure and identify all signals in this topology, presented in figure 7.2.

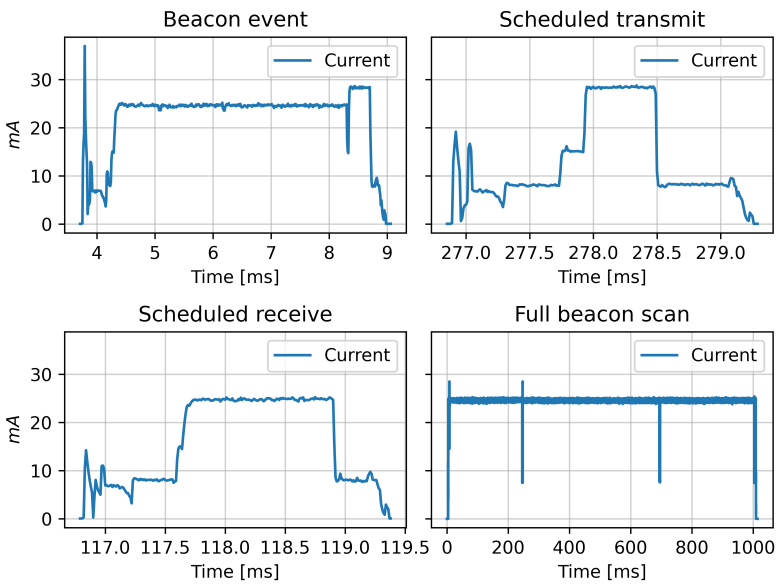

Figure 7.2: An overview of all NeoMesh radio events

After reviewing over 75 different events the mean current during each event is given in table 7.1.

After measuring idle energy consumption we move on to the measurements when sending data. One outer node is constantly transmitting a full sized payload through another outer node, this means that the middle node will receive, then relay this message. Using the same technique as above we find the CTS signal when the nodes are transmitting and we find a transmit event with data, shown in figure 7.3.

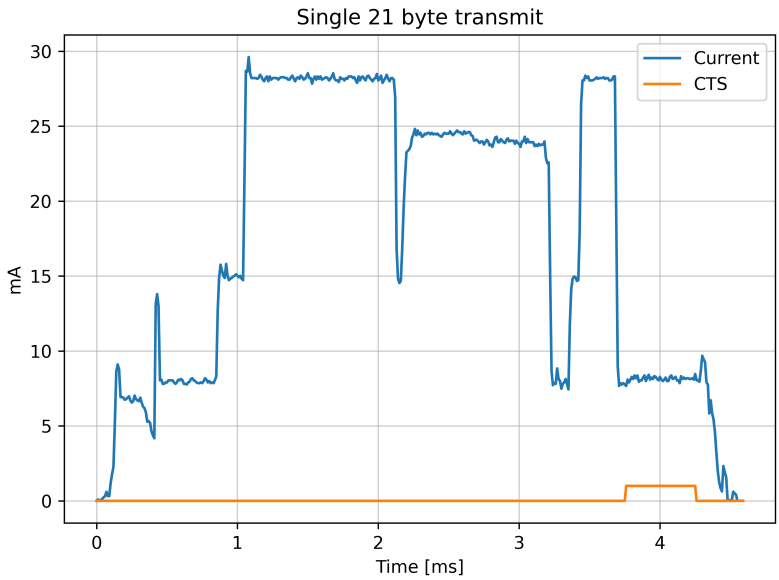

Figure 7.3: Transmitting a message with maximum payload size

Lastly we want to figure out the sleep current of the mesh node. A overview of the sleep current is shown in figure 7.4. To obtain these results we have filtered out any data points above $0.8mA$ and then taken the mean of the remaining data. This means that we get a slightly higher mean that the exact sleep current, but with a data set of 120 seconds it should account for very little. The data seems noisy because we get the start and end of the radio events included in our measurements.

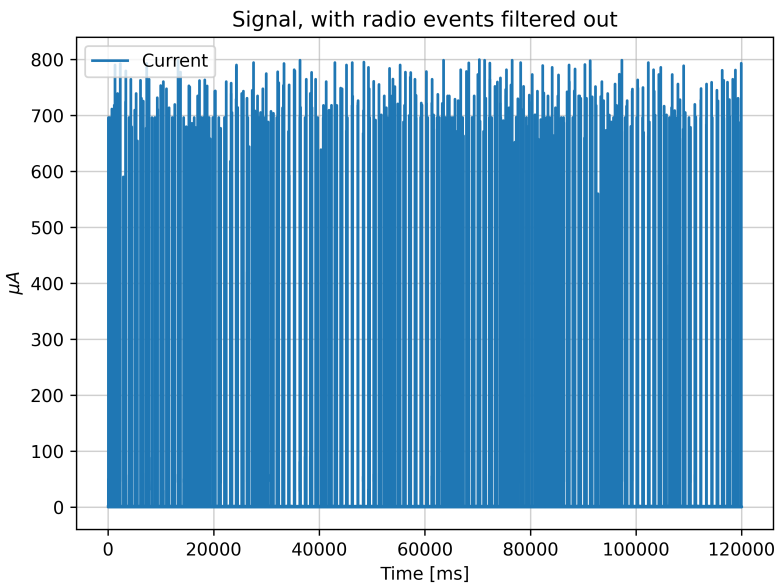

Figure 7.4: NeoMesh sleep current, no radio events

When events where identified a number of them were sampled to create a better average overview of the cost of each event. For the beacons, receive and transmit events 25 events were used to create an average. For the full beacon scan only 7 events, but the variance of the data was so low that it should not matter. A summary of these measurements are shown in table 7.1.

Event	**Duration**	**Mean current**	**Charge usage**
Beacon	$5.162ms$	$21.622mA$	$111.60\mu C$
Scheduled transmit(1dBm)	$2.394ms$	$12.838mA$	$30.724\mu C$
Scheduled transmit(-30dBm)	$2.451ms$	$10.27mA$	$25.181\mu C$
Scheduled receive	$2.585ms$	$15.953mA$	$41.244\mu C$
Full payload transmit	$4.577ms$	$17.59mA$	$80.511\mu C$
Full beacon scan	$1000ms$ (T_{beacon})	$24.521mA$	$24.664mC$
Sleeping	N/A	$0.895uA$	$t_{sleep} \cdot 0.06048\mu C$

Table 7.1: Overview of average energy usage of NeoMesh radio events

Based on the results in table 7.1 we get the following model for average current usage, based on $N_{neighbour}$, T_{beacon}, $T_{fullscan}$ and T_{tx}:

First we make find the percentage of time spent on each event(equation (7.1)).

$$p_{beacon} = \frac{t_{beacon}}{T_{beacon}} = \frac{5.35 \cdot 10^{-3}}{T_{beacon}} \tag{7.1a}$$

$$p_{tx} = \frac{t_{tx}}{T_{tx}} = \frac{2.44 \cdot 10^{-3}}{T_{tx}} \tag{7.1b}$$

$$p_{rx} = N_{neighbour} \cdot \frac{t_{tx}}{T_{tx}} = N_{neighbour} \cdot \frac{2.61 \cdot 10^{-3}}{T_{tx}} \tag{7.1c}$$

$$p_{fullscan} = \frac{T_{beacon}}{T_{beacon} T_{fullscan}} = \frac{1}{T_{fullscan}} \tag{7.1d}$$

$$p_{sleep} = 1 - (p_{beacon} + p_{tx} + p_{rx} + p_{fullscan}) \tag{7.1e}$$

$$\tag{7.1f}$$

And we now calculate the mean current of each event over time:

$$i_{beacon} = \bar{i}_{beacon} \cdot p_{beacon} = 21.622 \cdot p_{beacon} \tag{7.2a}$$
$$i_{tx} = \bar{i}_{tx} \cdot p_{tx} = 12.72 \cdot p_{tx} \tag{7.2b}$$
$$i_{rx} = \bar{i}_{rx} \cdot p_{rx} = 15.71 \cdot p_{rx} \tag{7.2c}$$
$$i_{fullscan} = \bar{i}_{fullscan} \cdot p_{fullscan} = 14.521 \cdot p_{fullscan} \tag{7.2d}$$
$$i_{sleep} = \bar{i}_{sleep} \cdot p_{sleep} = 0.000895 \cdot p_{sleep} \tag{7.2e}$$
$$\tag{7.2f}$$

Then we can express the average idle current draw as in equation (7.3).

$$I_{avg} = i_{sleep} + i_{tx} + i_{rx} + i_{beacon} + i_{fullscan} \tag{7.3}$$

We can also make a simple model that includes transmitting packages based on the percentage of times we expect to transmit a package ($p_{payload}$). We add equation (7.4):

$$p_{tx,payload} = \frac{t_{tx,payload}}{T_{tx}} = \frac{4.577 \cdot 10^{-3}}{T_{tx}} \tag{7.4a}$$
$$i_{tx,payload} = 17.59 \cdot p_{tx,payload} \tag{7.4b}$$
$$\tilde{i}_{tx}(p_{payload}) = p_{payload} \cdot i_{tx,payload} + (1 - p_{payload}) \cdot i_{tx} \tag{7.4c}$$

And modify equation (7.3) to end up with equation (7.5):

$$I_{avg,transmit}(p_{payload}) = i_{sleep} + \tilde{i}_{tx}(p_{payload}) + i_{rx} + i_{beacon} + i_{fullscan} \tag{7.5}$$

It is important to note that this model is not exact. It does not take into account a few complex scenarios.

Because of the slight imperfect period the transmit or receive events may overlap a beacon transmission event, this is not accounted for by the model. It happens fairly rarely, so the effect will be minimal, which is why it has been neglected. When nodes are in a mesh with more than a few neighbours nodes may also skip random beacon transmissions to avoid collisions, this is not accounted for.

In the proposed models the fact that $N_{neighbour}$ is assumed constant may also lead to errors in the model. The neighbour count may, especially in our example be changing dynamically and the nodes will have to increase or decrease the amount of receive events. For example if a set of sheep moves in and out of a complete mesh into a string like topology the neighbour count will switch between one and two neighbours. A way to model for this would be to change the weighting of the receive event in our model to use the expected average neighbour count in stead of the static neighbour count.

To test out the viability of the model it was applied to three samples. Two samples with two neighbours, 1 second scheduled transmit and 1 second beacon rate and one with no neighbours, but the same data rates.

In the first sample the node was not transmitting and in the second the node was constantly relaying info between two nodes. In these samples no full beacon was scanned,so that term has been set to zero to get a more accurate read. The first sample we calculate $I_{avg} = 225.757\mu A$ while the data shows $\bar{I} = 196.14\mu A$. In the second sample we calculate $I_{avg,tx}(1) = 275.520\mu A$ while the data set shows $\bar{I} = 244.64\mu A$. The last example with no neighbours we get $I_{avg} = 143.265\mu A$ and the data gives us $\bar{I} = 151.73\mu A$.

Now we can apply this model to the different configuration from section 6.2. This yields table 7.2 and in figure 7.5 we can observe how the configurations perform given a percentage of the scheduled transmits sends data.

Config N	Average current consumption
1	$234.79\mu A$
2	$24.91\mu A$
3	$20.39\mu A$
4	$17.41\mu A$
5	$12.89\mu A$
6	$228.64\mu A$
7	$12.69\mu A$
8	$31.77\mu A$

Table 7.2: Performance of different NeoMesh configurations

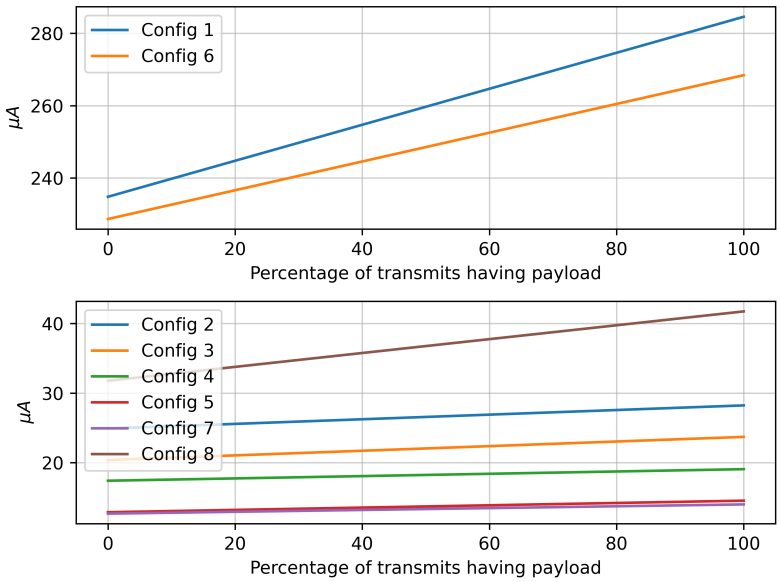

Figure 7.5: Overview of NeoMesh performance given percentage of transmissions with payload

7.2 LTE-M

In figure 7.6 we can se an LTE registration with the base station from a ”cold” state, meaning there has been no previous communication between the LTE modem and the base station. It takes about 15 seconds, depending on the coverage in the area where the LTE modem is located.

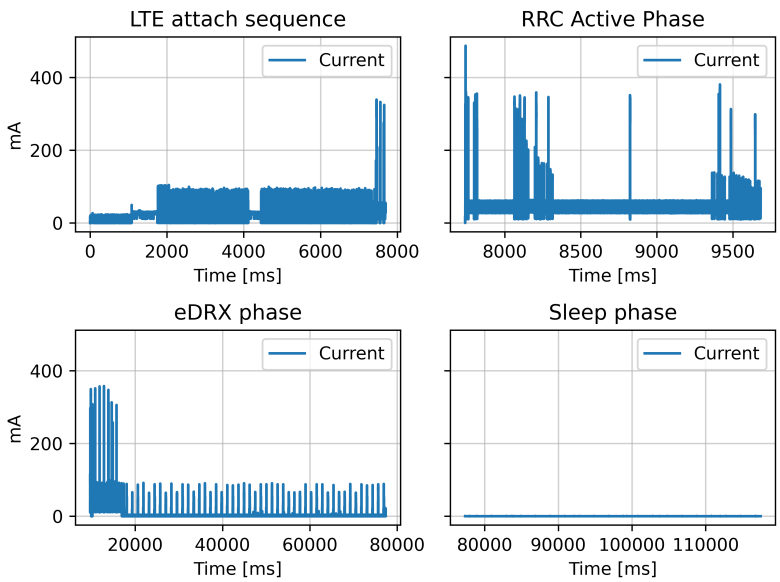

Figure 7.6: Current profile of intial LTE connection

The full first time connect can be divided into 4 stages, the first attach sequence, an RRC active phase, a DRX phase and then finally the sleep phase. They are all shown in figure 7.6.

After the device has registered with the base station it requires less energy to maintain the connection as described in section 3. In Trondheim, using a iBasis SIM card the nRF9160 was given a 1800 seconds Tracking Area Update timer and a 60 second active time. Which means that the base station expects the device to transmit data each 1800 seconds and the device must be reachable for 60 seconds after that.

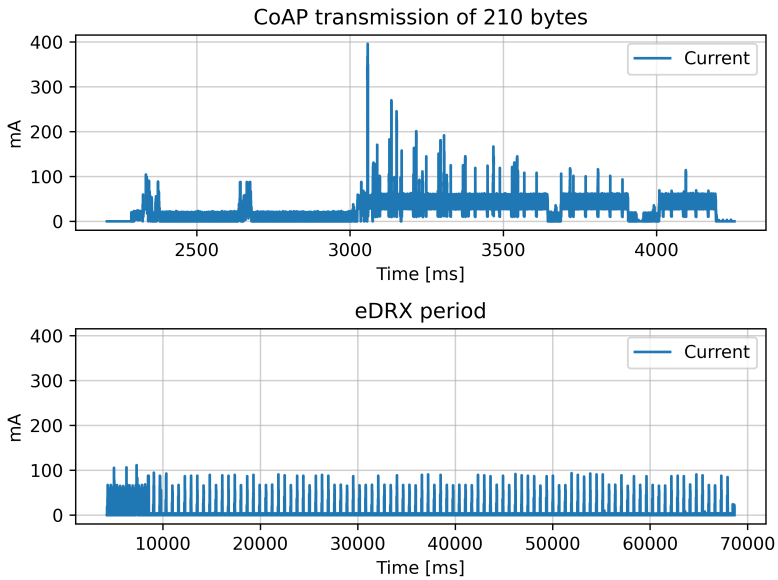

Figure 7.7: A 210 bytes CoAP transmission from PSM

In figures 7.7 and 7.8 we can observe the current profile when the device wakes up from a PSM sleep interval to transmit data.

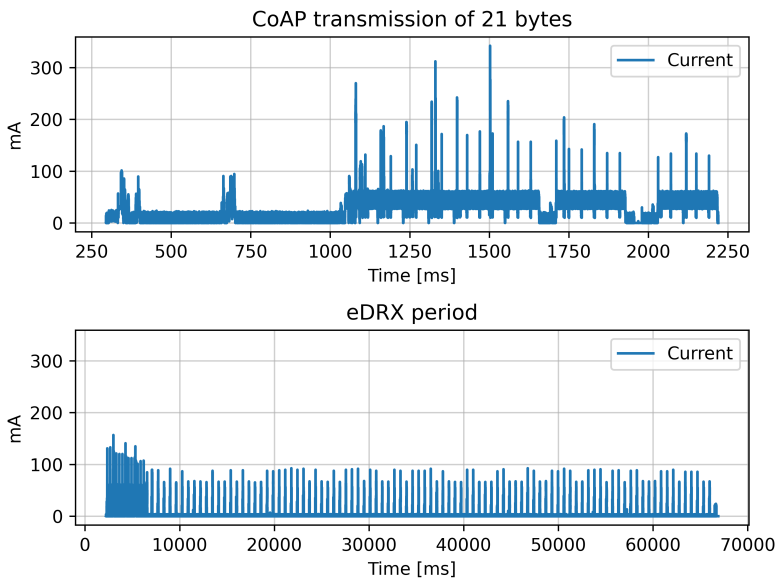

Figure 7.8: A 21 bytes CoAP transmission from PSM

In figure 7.7 a transmission where the payload has a size of 210 bytes was done, which corresponds to transmitting 10 regular mesh payloads in one transmission, while in figure 7.8 we transmit a single 21 byte payload.

Event	**Active time**	**Mean current**	**Charge usage**
Initial connect	$77.36s$	$21.622mA$	$590.690mC$
Transmit 21 bytes	$66.59s$	$2.1906mA$	$145.898mC$
Transmit 210 bytes	$66.45s$	$2.2175mA$	$147.378mC$
Sleeping	N/A	$4.0814uA$	$t_{sleep} \cdot 4.0814\mu C$

Table 7.3: Overview of energy usage of LTE radio events

These events are summarized in table 7.3. We can use this to calculate the mean current used to transmit the two payloads.

$$I_{mean} = \frac{i_{sleep} \cdot t_{tau} + i_{tx} \cdot t_{active}}{t_{tau} + t_{active}} \tag{7.6a}$$

$$I_{mean,21B} = \frac{4.0815 \cdot 1800 + 2190.6 \cdot 60}{1860} = 74.6142\mu A \tag{7.6b}$$

$$I_{mean,210B} = \frac{4.0815 \cdot 1800 + 2217.5 \cdot 60}{1860} = 75.482\mu A \tag{7.6c}$$

7.3 GPS

To reduce the amount of work it takes to acquire GPS data an existing data set, from an internal GPS tracker project at Norbit, that contains data about the time spent to get a fix will be used in conjunction with a mean current usage when searching for GPS fix.

In figure 7.9 we can see an overview over the time spent to get fixes in a deployed GPS tracker. In average the time spent to get a fix is 32.54 seconds.

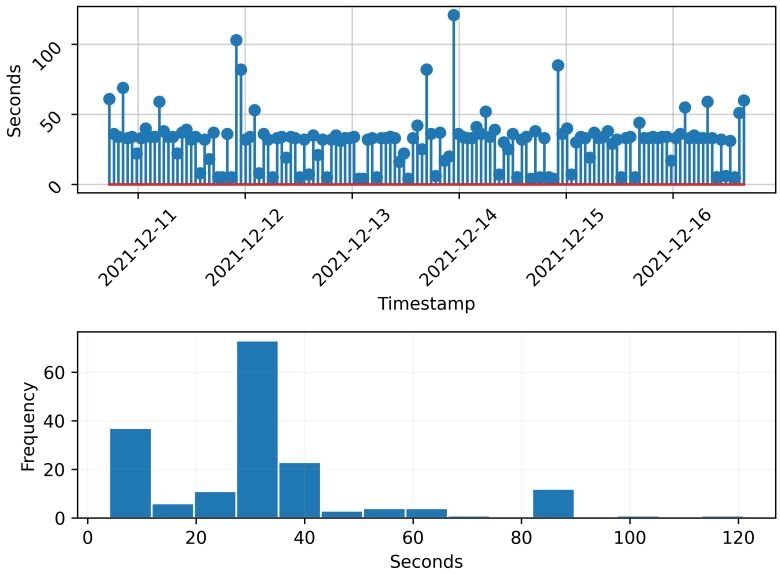

Figure 7.9: Time spent to get fix

The values in figure 7.9 is extracted from a tracker that is attached to a car. The car is mostly driving around in Trondheim city, which means it moves further and less constantly around than sheep.

A current profile for the GPS search is also acquired, shown in figure 7.10, which lets us find the mean current consumption of a GPS search. From this data set of 30 seconds we find that the current usage have a mean of $35.194mA$.

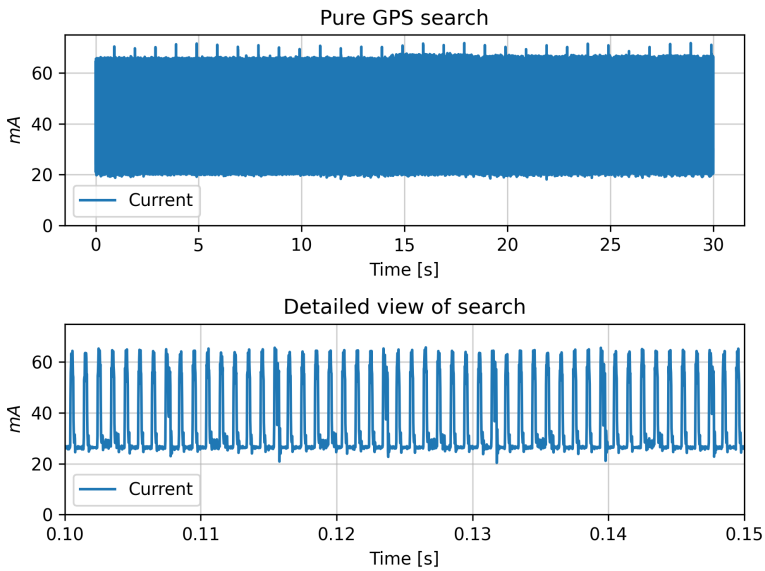

Figure 7.10: Current profile when searching for GPS fix

7.4 Thread

For the Thread we have measured two different devices, one FTD or full thread device that acts as a leader in the network and one minimal thread device that is a mesh node and runs on ultra low power.

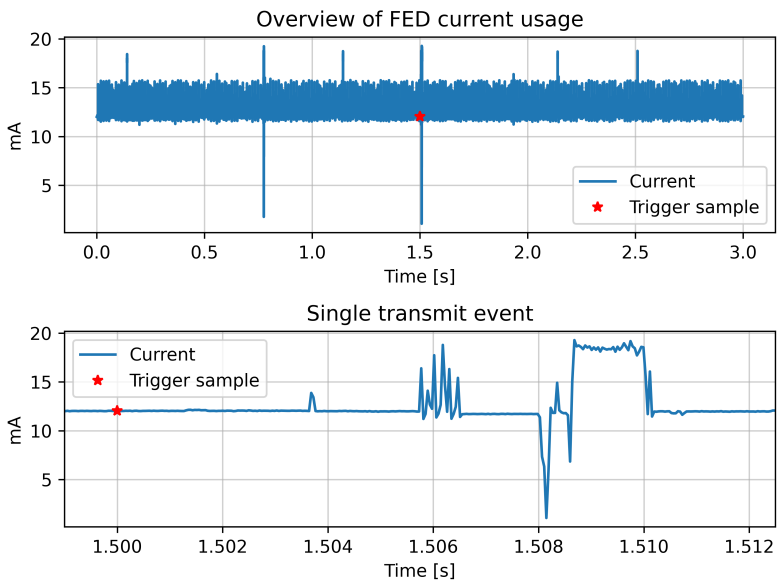

Figure 7.11: Full Thread Device current profile

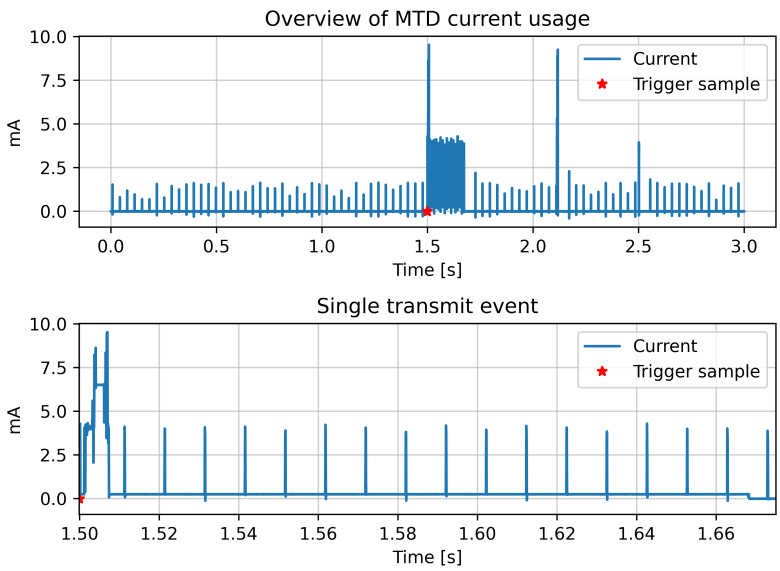

Figure 7.12: Minimal Thread Device current profile

7.5 Error sources

In ultra low power designs(ULP) it is important to choose a highly efficient PMIC with a very low quiescent current. A ULP device will spend most of its time in a sleep or deep sleep state which makes the quiescent current that leaks through the PMIC non negligable in proper designs. In the experiments performed in this paper we ignore the losses of the PMIC. All Nordic development kits are developed with current measurement capabilites in mind while the NeoMesh module is a simple breakout board which means it should work very closely to a production setting.

A good few months into the measurement work a flaw was found in the N6705B. When measuring the fast dynamics of the current profiles the instrument could not switch between current ranges quick enough. This means that most of the readings done by the N6705B have an unknown bias and low accuracy during sleep, i.e. currents in the μA ranges. Some of these measurements are

still used in the analysis, but to reduce the impact of the inaccurate low current measurements only the section with radio activity is used and new samples of sleep events where done with the PPK2.

The NeoMesh average current model is only based on empirical data, so it may not be used as an exact model. To create a more accurate model better instruments should be used to capture more of the dynamics in the model. A assumption that is made is also that the different radio events are perfectly periodic, which is not the case. When autocorrelating the datasets we find that the events are a bit off. There might be a technical reason for the period offset, but no references to this was found in the documentation [27],[19],[20].

8 Discussion

In this section we will discuss the results presented in the last section.

8.1 Viability of Thread

First we will discuss the viability of the Thread protocol in the context of homogeneous mesh networks. Since the network is assumed to be homogeneous it means that all nodes will have access to the same amount of energy after each recharge. From the analysis of the OpenThread devices, more specifically the Full Thread Device in figure 7.11, we can see that the current consumption of such a device is at $\tilde{1}2mA$. This of course comes from the fact that a FTD is required to keep its radio on at all times. Radio is one of the most expensive activities our nodes can perform, so this is not ideal.

While the Minimal Thread Device supports a very small sleep current, down in the nano amperes, the $12mA$ mean current consumption of the FTD makes the Thread protocol unfit for this specific use case. Thread FTDs should have a larger source of power than a small battery.

8.2 Energy usage in NeoMesh

The main parameter that decides the performance of the NeoMesh network in terms of energy usage is also the parameter that decides the networks performance in terms of network throughput and how long it takes networks to form and change. That means that when deploying a NeoMesh network it will in large to figure out the tradeoff between network performance and energy performance. For some networks a high throughput may be desired, but this will result in lower battery life and vice versa.

For a sheep tracker we can make a few assumptions about how the nodes will move and change over time. As mentioned earlier sheep tend to travel in sets, one sheep and two lambs. This means that a network of three nodes is very common and they will most likely stay within range of each other most of the time. Since sheep are not the quickest of creatures a fairly slow scheduled transmit period could be chosen and an even slower beacon rate. This means that something like configuration 8 from section 6.2 may be quite ideal for this sort of usage.

One thing this thesis have not touched is the network performance in such a mesh network. It may be a problem that is worth further studies. The proposed model in this paper is also built on a limited set of observation, so a more detailed model may be worth pursuing.

8.3 LTE-M performance

It is quite clear from the results that LTE-M is not ideal to communicate with the cloud at a high rate. Transmitting data over LTE-M is quite expensive, but in turn it is also quite efficient. The difference between sending 21 and 210 bytes to the cloud is almost negligible which makes it quite good for transmitting larger amounts of data to the cloud at slower intervals.

As of writing this thesis the LTE-M performance is also very much based on the PSM parameters that are granted to the receiver from the LTE network. This will likely improve in the future when IoT infrastructure is improved and more specific PSM parameters can be requested for different use cases.

8.4 Cost of Positioning

Acquiring GPS fixes are quite expensive in terms of energy, which is clear from the $34.04mA$ mean current usage and the average 30 seconds spent to get GPS fixes. If every sheep reports their

position, lets say once and hour this adds another $34.04 \cdot 30/3600 = 283.36\mu A$ of average current usage for each node. Given that even the most energy hungry NeoMesh configuration only draws $234\mu A$ on average, this is a large cost.

An interesting solution to this problem may be to only acquire a position fix on one node in the mesh network and then have that node distribute the position fix to its neighbours. Since mesh networks have quite limited range and GPS positioning may have a considerable error using the GPS fix across many neighbouring nodes may be accurate enough for many systems.

In sheep tracking we have the additional information that a lamb will follow its mother during the entire grazing season. This means that, as long as no predator scares the set and drives them apart, to find a set of sheep you can use any of their positions. Now assume that we are looking at a single set, i.e. a sheep and its lambs, if they are not scattered by predators the cost of GPS positioning will be amortized over the three nodes over time. Leaving us with a cost of $234\mu A/3 = 79\mu A$ which is a large energy saving.

Also, since our nodes are homogeneous, if the set is indeed scattered by a predator a farmer may be notified(by the sets mesh network being split up) and they may travel out and rescue the separated lambs. Normally if a sheep is killed by a predator the lambs are considered lost because they cannot survive on their own. The tracked lambs may be rescued by releasing a sheep with no lambs close to the lost lambs and they will follow the new mother for the rest of the season.

As discussed in section 3 to reduce the time to get a first fix we can use a A-GNSS system. Now since all the nodes very likely is using the same base station for LTE communication, they will all get the same A-GNSS data from the mobile network. So another optimization may be to have a single node fetch the A-GNSS data and then broadcast it to other nodes using the mesh network.

It would be interesting to conduct further studies on if, and potential how, one can optimize GPS positioning in a mesh network like the one proposed in this thesis. Since we have a lot of knowledge about the positioning of nodes, relative to each other it should be possible to f.ex. gather more than one GPS fix in the network and then use these data points and the network topology to further increase accuracy of the position of each node. This might be a computationally heavy task so it might be more suited for being solved in the cloud.

8.5 Mesh communication vs. LTE-M

In the sheep tracker setup the NeoMesh and LTE-M serves two different purposes. Mesh is used for local, node-to-node, communication while the LTE-M component is used to report state and update parameters of the nodes. Now the big question is if sharing state via mesh and then doing a shared LTE-M transmission is more efficient than every node transmitting its own state via LTE-M.

In our scenario we have three trackers, one set of sheep, that grazes close together and their trackers have formed a mesh network. Given that we utilize configuration 8, from section 6.2, which would give us a nice trade-off between network latency and power consumption. Also lets assume that the sheep is configured in a full mesh topology most of the time. How will the energy usage compare to the scenario where each tracker gets a GPS fix by itself and then reports it at 1800 second intervals?

In the first case we assume that we search for a GPS fix at each LTE-M active time. In this case the energy usage will be the mean current of the nRF9160 while transmitting 21 bytes of payload(in this case, 4 bytes longitude, 4 bytes latitude, 4 bytes altitude with a few bytes for other control data/state) including the current usage of a GPS search each half hour.

This yields us a mean current of:

$$74.6142\mu A + 35.194 \cdot 10^3 \mu A \cdot \frac{32.54}{1800} = 656.587\mu A$$

In the second case we have the nodes connected through mesh. They share state continuously and

only a single node transmits LTE and a single node searches for GPS. This means that our mean current will be the sum of the average current of NeoMesh configuration 8, the LTE transmission and GPS searches, but the two latter are divided by 3. Since we are sharing state we cannot ignore the fact that the nodes will be transmitting via mesh. They will be doing this constantly, regardless of LTE transmissions or GPS search, so to keep state updated on a 30 minute bases about 40% of scheduled transmits having payloads should be more than enough.

All of this gives us

$$35.753\mu A + \frac{74.6142\mu A + 35.194 \cdot 10^3 \mu A \cdot \frac{32.54}{1800}}{3} = 254.615\mu A$$

Now these results are heavily reliant on the fact that it is possible to more or less trivially share state between nodes. This might not be the case and the synchronization of transmissions, GPS searches and general state is a possibility of further studies. There are also quite a few assumptions made about the signal levels and LTE coverage in this thesis. It would be interesting to do a more specific study on how different topologies would affect this sort of solution and how well it would perform on actual sheep.

9 Conclusion

In this paper we have explored a sheep tracker with mesh networking, LTE-M connectivity and GPS positioning. Using a non-hierarchical mesh network and a homogeneous set of nodes it is possible to amortize the cost of energy costly radio events over a set of nodes and therefore conserve energy. Using nodes without mesh the energy usage would be $656.587\mu A$ on average, while utilizing three nodes in a mesh network reduced that to $254.615\mu A$ on average.

It has also been found that Thread is not a viable solution for this use case as mesh networks with coordinators or leader nodes draw too much current to be of use.

10 Further work

In this paper the focus has mainly been on comparing two different mesh protocols and 4G. It would of course be very interesting to see a broader study that compares more gateway technologies and more mesh protocols to see if an even more efficient solution can be found.

If we have such a flat hierarchical system that shares the responsibility of being the gateway an algorithm for choosing the optimal node would be an interesting topic for further studies.

There is also the subject of data integrity. If we have a flat hierarchy and a rotating gateway, how can a node be sure that its data have been successfully transmitted to the cloud? This is also an interesting topic that can build upon this paper.

Since we are looking at nodes with positioning capabilities and we know their topology a method for increasing positioning accuracy given GPS fixes and network parameters would be an interesting topic.

Bibliography

[1] Smartbjella, *Smartbjella - Home*, 2021. [Online]. Available: https://smartbjella.no/.

[2] FindMy, *Find My - Home*, 2021. [Online]. Available: https://www.findmy.no/.

[3] Telespor, *Telespor - Electronic Surveillance of Livestock*, 2021. [Online]. Available: https://telespor,no.

[4] 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.

[5] A. Cilfone, L. Davoli, L. Belli and G. Ferrari, 'Wireless Mesh Networking: An IoT-Oriented Perspective Survey on Relevant Technologies', *Future Internet*, vol. 11, no. 4, 2019, ISSN: 1999-5903. DOI: 10.3390/fi11040099. [Online]. Available: https://www.mdpi.com/1999-5903/11/4/99.

[6] 'IEEE Standard for Low-Rate Wireless Networks', *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, pp. 1–800, 2020. DOI: 10.1109/IEEESTD.2020.9144691.

[7] *Learn about bluetooth*, Accessed: 2021-11-19. [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/recent-enhancements/mesh/.

[8] á. Hernández-Solana, D. Pérez-Díaz-De-Cerio, M. García-Lozano, A. V. Bardají and J.-L. Valenzuela, 'Bluetooth Mesh Analysis, Issues, and Challenges', *IEEE Access*, vol. 8, pp. 53 784–53 800, 2020. DOI: 10.1109/ACCESS.2020.2980795. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9035389.

[9] R. Alexander, S. Aliciuc, S. Ashton *et al.*, *Thread Specification*, English, version 1.1.1, ThreadGroup, 13th Feb. 2017, 348 pp., Release.

[10] J. Chandroth, N. Saxena, A. Roy and E. Rastogi, 'A New Design and Analysis of Power Saving for IoT Gateway', *IETE Technical Review*, vol. 0, no. 0, pp. 1–9, 2021. DOI: 10.1080/02564602.2021.1880343. eprint: https://doi.org/10.1080/02564602.2021.1880343. [Online]. Available: https://doi.org/10.1080/02564602.2021.1880343.

[11] digitanimal, *digitanimal - Home*, 2021. [Online]. Available: https://digitanimal.com/.

[12] Smartbjella, *Product Statistics*, 2021. [Online]. Available: https://smartbjella.no/produkt/.

[13] Telespor, *Battery calculator*, 2021. [Online]. Available: https://tintin.telespor.org/batt-4712/.

[14] digitanimal, *Frequent questions*, 2021. [Online]. Available: https://digitanimal.com/extensive-farming/?lang=en.

[15] FindMy, *Find My - How the Satellites Work*, 2021. [Online]. Available: https://www.findmy.no/.

[16] FindMy, *Find My - Model 2 Product Overview*, 2021. [Online]. Available: https://www.findmy.no/nb/model2.

[17] S. Rutter, N. Beresford and G. Roberts, 'Use of gps to identify the grazing areas of hill sheep', *Computers and Electronics in Agriculture*, vol. 17, no. 2, pp. 177–188, 1997, Livestock Monitoring, ISSN: 0168-1699. DOI: https://doi.org/10.1016/S0168-1699(96)01303-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169996013038.

[18] E. S. Fogarty, D. L. Swain, G. Cronin and M. Trotter, 'Autonomous on-animal sensors in sheep research: A systematic review', *Computers and Electronics in Agriculture*, vol. 150, pp. 245–256, 2018, ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2018.04.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169918300450.

[19] NeoCortec, *User Guide*, English, version 2.4, NeoCortec, 31st Jul. 2019, 42 pp., Release.

[20] NeoCortec, *Integration Manual for NCxxxx series Modules*, English, version 3.1, NeoCortec, 31st Jul. 2019, 23 pp., Release.

[21] T. Connexion, *LTE-M vs NB-IoT – a guide exploring the differences between LTE-M and NB-IoT*, 2021. [Online]. Available: https://www.telenorconnexion.com/iot-insights/lte-m-vs-nb-iot-guide-differences/.

[22] L. Alliance, *LoRaWAN Coverage*, 2021. [Online]. Available: https://lora-alliance.org/lorawan-coverage/.

[23] Tech. Rep.

[24] Vodaphone, *Satellite IoT*, 2021. [Online]. Available: https://www.vodafone.com/business/iot/managed-iot-connectivity/iot-satellite.

[25] C. A. Ogaja, *Applied GPS for Engineers and Project Managers*. ASCE Press, 2011, ISBN: 978-0-7844-1150-6. DOI: doi:10.1061/9780784411506.ap02.

[26] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance*, 2nd. Ganga-Jamuna Press, 2006.

[27] NeoCortec, *Datasheet - NeoCortec-NC2400 Wireless Mesh Network Module Series*, English, version 1.2, NeoCortec, 8th Nov. 2021, 8 pp.

[28] NordicSemiconductor, *nRF OpenThread Server Sample*, 2021. [Online]. Available: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/1.8.0/nrf/samples/openthread/coap_server/README.html.

[29] NordicSemiconductor, *nRF OpenThread Client Sample*, 2021. [Online]. Available: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/1.8.0/nrf/samples/openthread/coap_client/README.html.

[30] NordicSemiconductor, *nRF9160 CoAP Client Sample*, 2021. [Online]. Available: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/1.8.0/nrf/samples/nrf9160/coap_client/README.html.

[31] NordicSemiconductor, *Online Power Profiler for LTE*, 2021. [Online]. Available: https://devzone.nordicsemi.com/power/w/opp/3/online-power-profiler-for-lte.