UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Beneath the snow – Developing a wireless sensor node for remote locations in the Arctic

Øystein Tveito

Master's thesis in Computer Science – INF-3990 – May 2020

"If people never did silly things,
nothing intelligent would ever get done."
–Ludwig Wittgenstein

"The difference between theory and practice is that in theory,
there is no difference between theory and practice."
–Richard Moore

Abstract

With climate change as an ever-present concern, ecologists from COAT are hypothesizing that with a warmer climate, more rain-on-snow events will occur. This creates ice layers in the snow which can inhibit the natural gas exchange in rodent tunnels on the Arctic tundra, leading to a build-up of CO_2. COAT wants to explore this, but currently does not have any means of collecting data on CO_2 concentrations under the snow.

In this thesis we describe how we designed, built, deployed, and improved upon a robust hardware- and software solution, tailor-made to this scientific question. During the course of this project, we created three distinct versions and we have conducted two deployments of the sensor nodes in the Arctic tundra. The node is able to measure CO_2, temperature, and humidity, in addition to monitoring an already existing COAT experiment. As the energy budget is a crucial factor for the success of our project, we have conducted experiments to optimize the power efficiency of the node.

The sensor nodes communicate over the LTE CAT M1 network, are waterproof, and are capable of operating in temperatures as low as $-25°$C. Through the use of software optimization, low-power components, and efficient duty-cycling, our solution is capable of operating for several years on battery power.

This novel sensor node solution will help the ecologists monitor and predict the impact of climate change on life beneath the snow on the Arctic tundra. The approach described will be applicable to a diverse set of scientific questions, spanning many branches of data-driven research.

Acknowledgements

I would like to thank:

- My collaborator, Michael J. Murphy, for allowing me to participate in this project, and sharing his knowledge on this topic. This thesis would not have been what it is without help.
- My main advisor, Professor John Markus Bjørndalen, for guidance and input throughout the the work on this thesis. Especially for his willingness to read through the poor English of the early drafts.
- My co-advisor, Professor Otto Anshus, for his excellent guidance throughout this project, and for hammering in the important separation between accuracy and precision.
- My other co-advisor, Professor Anders Andersen, for his help and support, even with the most densely packed calendar I have ever encountered in Outlook.
- Our student advisor, Jan Fuglesteg, for his guidance through the administrative jungle throughout my time at UiT.
- COAT, for providing such an interesting problem for us to solve.
- COAT Tools (Thematic project, funded by UiT - The Arctic University of Norway) and The Distributed Arctic Observatory (DAO, funded by the Research Council of Norway, IKTPLUSS program, grant number 27062), for funding the project.
- Ken-Arne Jensen, for sharing his experience in engineering, and for forcing me to defend my choices, sometimes unsuccessfully.
- Telenor Research, and specifically Dr. Arne Munch-Ellingsen, for donating SIM-cards for our project, and for the help in debugging once we encountered some problems with the signal.
- My brother, Torbjørn Tveito, for his excellent feedback and help with my writing, his free physics lessons when we had problems with the antenna, and lastly for his relentless badgering about that *power consumption* is the wrong word for *energy consumption*.
- My mother, Aase Tveito, for her support, her services with proofreading, and her babysitting, allowing me to works.
- My wife, Linda Tveito, and my children, Ada and Thomas, for their patience with me throughout this period.

Contents

List of Figures

1

Introduction

Climate change will have a significant impact on the planet in different ways. We are already seeing warmer temperatures, more extreme weather events and local ecological changes. It is important to try to mitigate some of the effects, but to do that we need to know what the consequences of climate change are.

In order to explore these consequences, we need to monitor a range of parameters over time to identify trend lines and separate natural variations from the effects of climate change. Often, these parameters should be measured in locations far from people, and thus with little or no infrastructure supporting the measurement setup. As a consequence of this isolation, robust and durable systems are required to ensure stable data collection over time. One such project is COAT (Climate-ecological Observatory for Arctic Tundra)[7], ran by *UiT - The Arctic University of Norway* (UiT) in Varanger and Svalbard.

Figure 1.1: Picture of one of the camera traps hidden beneath rocks in the arctic tundra (Photo: Michael J. Murphy)

Figure 1.2: A picture taken by the camera trap (Photo: COAT)

1.1 COAT research project

Amongst other parameters, COAT is currently gathering data on rodent populations underneath the snow cover during the winter months using camera traps [37]. They are conducting long term research on how the population is affected by different conditions such as climate, weather, and other external and internal factors. The existing setup consists of a tunnel with a wildlife camera mounted in it. By photographing rodents going through the tunnel they can estimate population size and when reproduction and growth is happening.

The picture data is correlated with weather data from both the national weather service and COAT's own weather stations in the area. In addition to this COAT are gathering data using other methods, such as catch-and-release, kill-traps. This is all part of the data set being used to model which factors influence the ecosystem in the arctic tundra.

1.2 Complementary data

Because of warmer winters in the Arctic, the upper snow layers can melt and create ice layers[40]. These layers might inhibit the natural gas exchange through the snow. Without this exchange, the chemical makeup of air in the tunnels may change. COAT's hypothesis is that CO_2 may build up in the tunnels.

The research group DAO was asked to create a sensor node that could detect CO2 levels inside their pre-existing camera trap tunnels. Heightened CO_2 levels may change rodents reproductive behavior. In addition, sufficiently high CO_2 concentrations can be fatal.

In addition to CO2 readings they wanted a health check on their camera to make sure that the camera was recording events, as the camera is not connected to any network. We were not allowed to modify the camera in any way, as altering their solution might skew the results gathered from the camera. Long-term data collection is what enables them to draw conclusions on the data gathered. Changing parameters in data collection will make the data harder to correlate with data gathered from earlier years in addition to the risk of our solution not working as well as their proven solution.

The camera records time of day and temperature, but data on the accuracy of the temperature readings are not listed in their product specification. In addition, the temperature sensor in the camera is directly attached to the camera PCB. When the camera operates, the power consumed will be transformed to heat in the camera case. This adds a bias to the temperature readings, shifting them to a higher value. The camera case acts as an insulator, delays outside influence on the temperature measured by the sensor. These effects are likely much less significant than measurement error inherent in the sensor itself. COAT wanted a second temperature reading to either collaborate with or replace the readings from the sensor in the camera.

The camera takes a set of images each time it registers an event in addition to one picture every 12 hours. Having the temperature sensor on the sensor node also allows for regular temperature readings more often than the camera does. The period between control pictures can be shortened, but that would reduce battery life.

When the camera traps are set, there is only a small window of time where it can be revisited for maintenance or battery replacement. When the snow comes the traps cannot be touched. This is because we cannot disturb the habitat without risking changing parameters. Therefore, the sensor node should stay active through the year.

1.3 CO_2

The knowledge gap we are attempting to fill is whether the CO_2 levels in rodent tunnels are significantly different from the above atmosphere. Small variations in the concentration of CO_2 may have significant ramifications for the planet

as a whole, but it hardly registers on an individual level. Previous experiments have shown that the CO_2 concentrations underneath snow can be as high as 70, 000 parts per million (ppm) [21]. Rodents can be affected by as little as 50, 000ppm [19] or by long term exposure to 3, 000ppm[35]. It is our objective to investigate whether the conditions in the rodent tunnels are approaching these levels, or if they are closer to a normal atmosphere at 400ppm[8]. We are therefore not overly concerned with minimizing errors in our measurements. If we have a variance of as high as 50% of the value, it doesn't necessarily reduce the usefulness of our dataset. The most interesting point to know is if we are close to fresh air values, or in the thousands.

1.4 Computer science in ecology

We propose a wireless sensor node that will be able to gather the information requested and synchronizing it with our campus servers. The research and technology advantages done in later years within wireless sensory solutions have provided many exciting new possibilities. The 4G standard even introduces dedicated networks for sensor solutions with LTE CAT 1[6]. The problems described are interesting from an IT point of view as they introduce some real life challenges which must be overcome.

The proposed sensor node will have both dimensional and functional restrictions. Top amongst the challenges are how to have a node survive for a year, whilst synchronizing both data and software updates from a remote location with limited infrastructure. Both the hardware and the software must be robust. Once deployed it is required to handle any problem in an adequate manner without human intervention.

Creating IoT solutions like this require the software and hardware to work together. The consolidation of software and hardware must be considered at all stages in the project. This adds an additional dimension to the project compared to a pure software project.

1.5 Overview of contributions

This project has been realized largely by two people: Øystein Tveito, the master student and author of this thesis, and Michael J. Murphy, a PhD student at UiT.

The project described is a part of the PhD-project of Murphy. Throughout this

project, we have cooperated closely. In order to give a coherent description of the project, it is necessary also to include the contributions of Murphy. This does not include all the work of Murphy, only the parts relevant for this thesis. In the following, I shall describe the main division of work between us. In all phases of the project, help has been given and offered, but the division of responsibility and the main efforts are distributed as follows:

1.5.1 Michael J. Murphy

Murphy was in the beginning responsible for the software of the sensor node. The parts of this project described in this thesis that the author does not claim any credit for are:

- Software for version 1.0
- Software for version 1.1
- The first physical deployment

1.5.2 Øystein Tveito

The parts of the thesis for which the author claims full credit are:

- Hardware for version 1.0, 1.1, and 2.0
- Design and manufacturing of parts
- Assembly of units
- Software for version 2.0
- Testing and experiments done on version 2.0 of the sensor node

1.5.3 Remaining tasks

The remainder of the work have been done on together, and both Tveito and Murphy should have credit for it.

2 Requirements

This project will cohabit the pre-existing research sites that COAT currently operates with cameras in them on the arctic tundra. This results in a number of requirements, restrictions, and issues that will have an impact on the software, hardware, and packaging of the prototype node. Some requirements are absolute, while others give "nice to have" value to the project.

2.1 Non-interference

The research project we are participating in is based upon time series of continuous and comparable data, spanning years and decades. It is therefore important that our contribution in no way interferes with or modifies the existing experiment, namely the camera trap. This project should also not be visible from inside the the main compartment of the camera trap, as this could potentially interfere with the rodents behavior. furthermore, no wires should be exposed anywhere in this compartment, as rodents tend to chew on wires in the experience of the COAT researchers. A severed connection can interfere with our data collection, and dead rodents could interfere with the original experiment. The box can also not be altered in any major way to ensure continuity with the data already being collected.

2.2 Physical dimensions

The box where the camera is placed has two smaller spaces in it with the dimensions $100mm \times 140mm \times 230mm$. Both compartments are empty and we are allowed to use both if needed.

Figure 2.1: A dimensional accurate camera trap prototype we have in our office. Color and lid mechanism has changed in deployed camera traps. (Photo: Øystein Tveito)

2.3 Measurement types

As mentioned in the introduction, the primary objective of the project is to measure CO_2 concentrations. In addition, the ecologists want to get temperature readings and the status of the camera they have deployed. Each of the camera traps already onsite has a temperature sensor and record the current temperature each time it takes a picture. The problem with this is that if no rodents go through the trap, only one picture will be taken every 12 hours. And so, the camera trap only records one measurement of temperature every 12 hours. It is requested that we take a temperature measurement on a more regular basis.

Sometimes when a field operator deploys a camera, it is done incorrectly. They might, for example, have forgotten to turn it on or it may be incorrectly configured. It is also possible that the camera is malfunctioning. Physically going out to the experiment after deployment is expensive, difficult, and time consuming. If a camera is not operating correctly there is no way of knowing until it is retrieved the following summer. Because of this, the ecologists want a way to remotely check if a camera is working. If not they might consider going out in the field to fix that specific unit.

2.4 Data quality

For this project, the main data point to gather is CO_2 concentrations in the rodent tunnels. COAT does not have a clear image of what concentrations they expect to find. Therefore, the first round of experiments are meant to set a baseline for future experiments.

As this is a first-order exploratory experiment we are not overly concerned with minimizing measurement error. The sensor needs to be able to measure the entire span of feasible concentrations. It also needs to be able to determine the value within half an order of magnitude. In other words, a relative error up to 20% is acceptable for our purposes.

The CO_2 concentration is expected to be anywhere from 400ppm as found in normal atmosphere[8] to 70.000ppm as found in previous research[21]. The conditions where the camera traps are placed may have significant differences from the environments investigated earlier. It is therefore possible that the CO_2 reaches even higher concentrations. The node therefore needs to be able to detect a large range of CO_2 concentrations, at least up to 70.000ppm. Our data is intended to show if and how much CO_2 concentrations in rodent snow tunnels rise over the course of winter. While not crucial, minimizing sensor error is part of the "nice to haves".

When it comes to temperature measurements it is important to minimize both bias and error, while maximizing temperature resolution. We assume that the temperature inside the tunnels will consistently be within a few degrees of the freezing point of water. In this temperature range even small differences can be significant. This is why the requirements for the thermometer are so strict. Thermometers are widely avaliable and reasonably cheap. As such, the high requirements for this sensor do not increase the cost or complexity of the sensor node significantly.

The camera health check only needs to provide the information of whether

or not the camera functions. It is important to note that the non-interference requirement limits the avaliable methods to indirect observation only.

2.5 Climate

The climate of the arctic tundra can prove a harsh environment for the sensor node. In order for the project to succeed, the node will need to withstand the elements for at least a year. The camera traps are covered in rocks, which helps keep it stable, upright, and hidden (see Figure 1.1). They are not waterproof and are occasionally flooded during snow melts. The sensor nodes must therefore be able to survive being submerged in water. Before being covered in snow, the camera traps can experience extreme temperatures. Due to this, the electronics and batteries must remain functional after being exposed to temperatures as low as $-25°C$.

2.6 Data storage

The sensor node must record its measurements and log it's status. This requires a storage medium able to contain a year's worth of measurements and logs. Data corruption is an ever-present concern, and steps must be taken to minimize the risk of data loss. Furthermore, the camera health check must be remotely accessible. This requires the sensor node to have network communication capabilities. When the node is connected to a network, it is beneficial if it is able to send the other data as well.

2.7 Time resolution

The intended use of the CO_2 data is to investigate gas buildup in rodent tunnels over the course of winter. Short term fluctuations are not expected to occur. As a result, only a few CO_2 measurements a day are required. The data of interest are only gathered while the camera trap is covered in snow. The snow acts as a strong insulator, and the ground is a large reservoir of heat. Combined, these effects act to dampen temperature fluctuations in the tunnels. Therefore the temperature only needs to be measured on an hourly basis.

2.8 Camouflage

Some of the deployment areas for this project are within national parks. According to the Natural Diversity Management law regarding national parks: "*Forskriften skal [...] sikre en uforstyrret opplevelse av naturen*", (Naturmangfoldloven, §35, 2009) . Our translation of it's intent is that the national parks must appear to be unspoiled nature. While COAT has received dispensation to deploy the camera traps, they must be camouflaged to be as unnoticeable as reasonably possible. The sensor node cannot make the camera trap significantly easier to notice. In addition to the legal requirement of camouflage, curious hikers who find the camera traps might interfere with the experiment. It is therefore necessary to camouflage all boxes, not only those in national parks.

2.9 Power

Each deployment cycle is one year. It is therefore a strong preference that the sensor node can stay active for at least that time. As stated previously, however, the data of interest is when the camera trap is covered in snow. As such, it is not strictly necessary that the node remains active after the snow melts. Due to the camouflage restriction, solar panels and other common energy harvesting techniques are not viable.

2.10 Practical concerns

Deployment involves carrying the sensor node several kilometers on foot. This means that it should not be excessively heavy. It must also be durable enough for some rough handling during transport. Effort must be made to ensure that deployment is reasonably simple. It is preferable that the sensor node can use the same type of battery as the camera. This would make deployment and maintenance less complicated.

Price point is also an important feature of this project. While there is no fixed budget, price should be considered an important factor for component choices.

3

Related works

3.1 Commercial solutions

Scientific research often employs costly generic commercial solutions with high precision and accuracy. Commercial products like the iButton from Maxims Integrated or Hobo from Onset Computer are widely used[11] due to their ease of use. One of the problems one might encounter is their relatively high price[26].

In addition to price, neither provide a solution with CO_2 sensing capabilities. LI-COR and Lascar[11] both have CO_2 sensor solutions with data loggers that would meet our requirements to precision and accuracy. Again price would be an issue, but not as big as the physical dimensions. The sensor solutions comes as multiple briefcases made for fixed installations. Their other option is handheld data loggers made for short term, manual data logging.

Tinytag from Garmin[11] have a couple of CO_2 sensors that conforms to our needs for precision and accuracy, in addition to spatial dimensions. Here, the problem is that they are made for indoor environments, and would not survive the climate we are exposing our sensor node to.

Testo have a few compact, energy efficient solutions that can measure most of the data we need for this project. This product is also expensive, and is not made for outdoor use. Aside from this, the sensors are limited to only WiFi connectivity and a closed source cloud solution that will not work for our

use-case.

Summarizing many of the units on the market, most of the available solutions have limitations that will hinder us from doing what we are planning. From the list in the study on widely used sensor solutions, all of them suffer from at least one of these problems:

- Is too large for our use-case
- Price is too high for a large scale deployment
- Not suitable for outdoor use
- Not made for prolonged deployment without re-calibration
- Limited communication alternatives
- Proprietary cloud solutions without control over our data
- Limited selection of sensors
- Lacking support for third party sensor solutions to allow us to gather of the needed data

3.2 Prior research in custom solutions

With the increasing availability of easy-to-use portable computing systems, like the Raspberry pi and Arduino, several studies have employed custom, DIY solutions in ecology [26, 29, 25]. While the Raspberry pi[29, 25] will give us access to more computing power, the energy consumption of such a machine would make it problematic for our use-case. Arduino[26] compatible controllers or similar provide enough power for data collection, with a significantly lower power consumption.

Prior state of the art in wireless sensor nodes (WSN) have been applied to many research fields, including both local and global environmental monitoring[2, 14, 20, 34, 41]. WSNs typically employ two different types of networks. One type is a local network, where nodes exchange data and synchronizes time and other configuration. This can be implemented by several different network technologies, including WiFi, LoRa, and ZigBee. The other typical network type is a remote network. This is a network that connects the nodes to the internet or a remote location. This is typically implemented through public infrastructure

like phone networks or satellites. When deploying close to infrastructure WiFi is also commonly used for this purpose.

For our purposes, having a local network could have been necessary, and still can in future work. The local network allows nodes in network shadows to communicate though other nodes. The implementation of such a network can be done in a few different ways. There can be dedicated gateway nodes. A gateway node then acts as the endpoint for the deployed sensor nodes. One gateway node usually serves multiple sensor nodes. An advantage of this scheme is that the sensor nodes do not have to have a remote network. A remote network is usually more expensive to communicate through, power-wise, than local direct connections. The power supply of the gateway node can then be scaled for the added communication demands. One issue that is introduced by having a dedicated gateway is that the network now has a single point of failure. This can be remedied by having redundancy or by having the gateway node easily accessible for repair.

Another approach is enabling all nodes to communicate with both the remote and local networks. In this solution all nodes can be equal and all nodes can act both as a normal node and a gateway. This is preferable in some cases as there then is no need to develop two different hardware- and software solutions. The WSN could also be able to react dynamically to changing network conditions.

Having bridging or multi-hop networks does, however, complicate the design. As most battery powered solutions rely on duty cycling, having synchronized time is important. Load distribution is also an interesting topic. Both of these concerns have been addressed in previous research[3, 28, 5]. Solutions with direct connection to permanent infrastructure do not have these concerns.

Sensor nodes have previously been deployed in harsh environments, including rivers[4], a desert[42], and a volcano[41]. Environments can be challenging from a practical perspective by having to protect the unit from harm. Contamination of measurements is also a concern, including humidity, dust and temperature. One important lesson learned in previous research is that the node will need to be built ruggedly enough to withstand the elements. Maintenance of the node also needs to be considered. Some sensors need periodic re-calibration to function as specified. If this is not possible, this needs to be taken into account.

4 Architecture

This chapter describes the overall hardware and software architecture of the sensor node. The architecture is the same for all versions of the node.

4.1 Hardware architecture

The hardware architecture is described in this section.

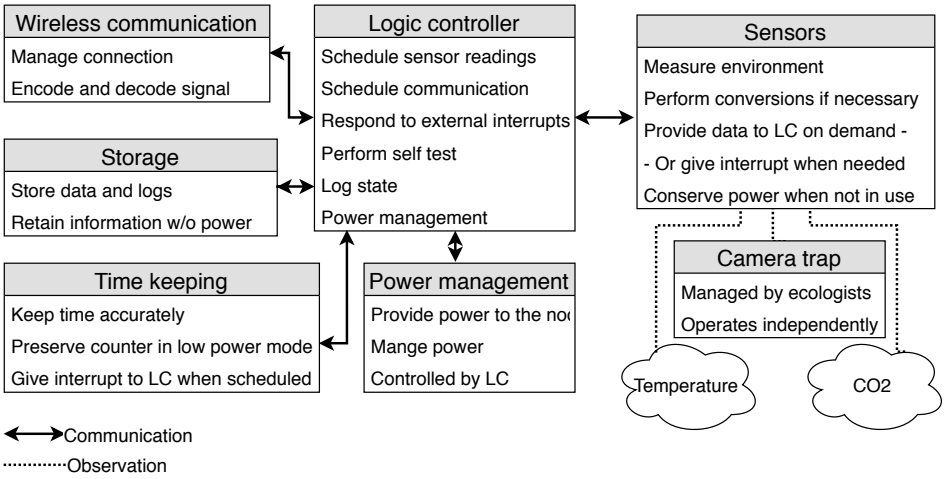

Figure 4.1: Diagram illustrating the overall architecture of the sensor node

4.1.1 Wireless communication

The sensor node must be able to communicate wirelessly with a backhaul network. Limited infrastructure on site sets restrictions on technology choices. Environmental factors may also at times impede signals, which may make communication difficult.

4.1.2 Time keeping

The sensor node needs the ability to keep track of elapsed time in order to manage measurement intervals. The gathered data must also be correctly timestamped. This indicates that there should be a time keeping device in the sensor node solution.

4.1.3 Logic controller

All processing and control in the sensor node should be delegated to a central logic controller. The logic controller will initiate all measurements and be in control of the node's network activity. It therefore must be able to communicate with all the components of the sensor node.

4.1.4 Data storage

All data collected by the sensors must be stored by the sensor node. The storage medium needs to have sufficient capacity for both data and logs that are created during a deployment cycle. The chance of data corruption has to be considered when choosing storage technology. Due to the possibility of power failure, the data must be stored on a non-volatile medium.

4.1.5 Power management

The power management system must be able to supply all components with power simultaneously. In addition, the system must store enough energy for the node to remain active for at least one deployment cycle.

4.1.6 Sensors

The sensor array must contain sensors that are able to measure the temperature and the concentration of CO_2. It also needs to be able to investigate the camera

status.

4.2 Software architecture

In this section, the software architecture is presented.

4.2.1 Operating modes

When the node is not doing other tasks it must activate a low-power mode. It is critical that the node is able to engage the active mode from this state. Reactivation must be possible by both scheduled and unscheduled events.

4.2.2 Periodic sensor readings

Some measurements must be done at regular, configurable intervals. The sensor node is therefore required to be able to access time and create scheduled events. At scheduled activations, the node must prepare the sensors and record the data to the storage medium.

4.2.3 Event-driven sensor readings

It is assumed that the camera health check must be event-driven. This means that the sensor node must be able to handle unscheduled, external interrupts. The node must then record the event and return to a low-power state. These activations must not affect the periodic reactivation schedule.

4.2.4 Communication

The sensor node must be able to communicate with the backend servers at regular intervals. When communicating, the sensor node must send all data and logs which have not been confirmed to be received by the servers previously. It must also check for and apply avaliable software updates. Network communication can be used to synchronize time.

4.2.5 Self test

The sensor node must have a built-in self-test. This must investigate all the essential functionalities of the sensor node. The results of the self-test must be presented in an intuitive manner. The operator deploying the sensor node should not be assumed to have intimate knowledge of the project.

5

Design

In this chapter we present the sensor node design. There is one main hardware design and three software designs, one for each version of the node. The software design is presented as a base design, with differences between versions in subsequent sections.

5.1 Hardware design

The hardware design in Figure 5.1 is presented in this section. This design is mostly common for all three versions. The design adjustments are presented at the end of the section.

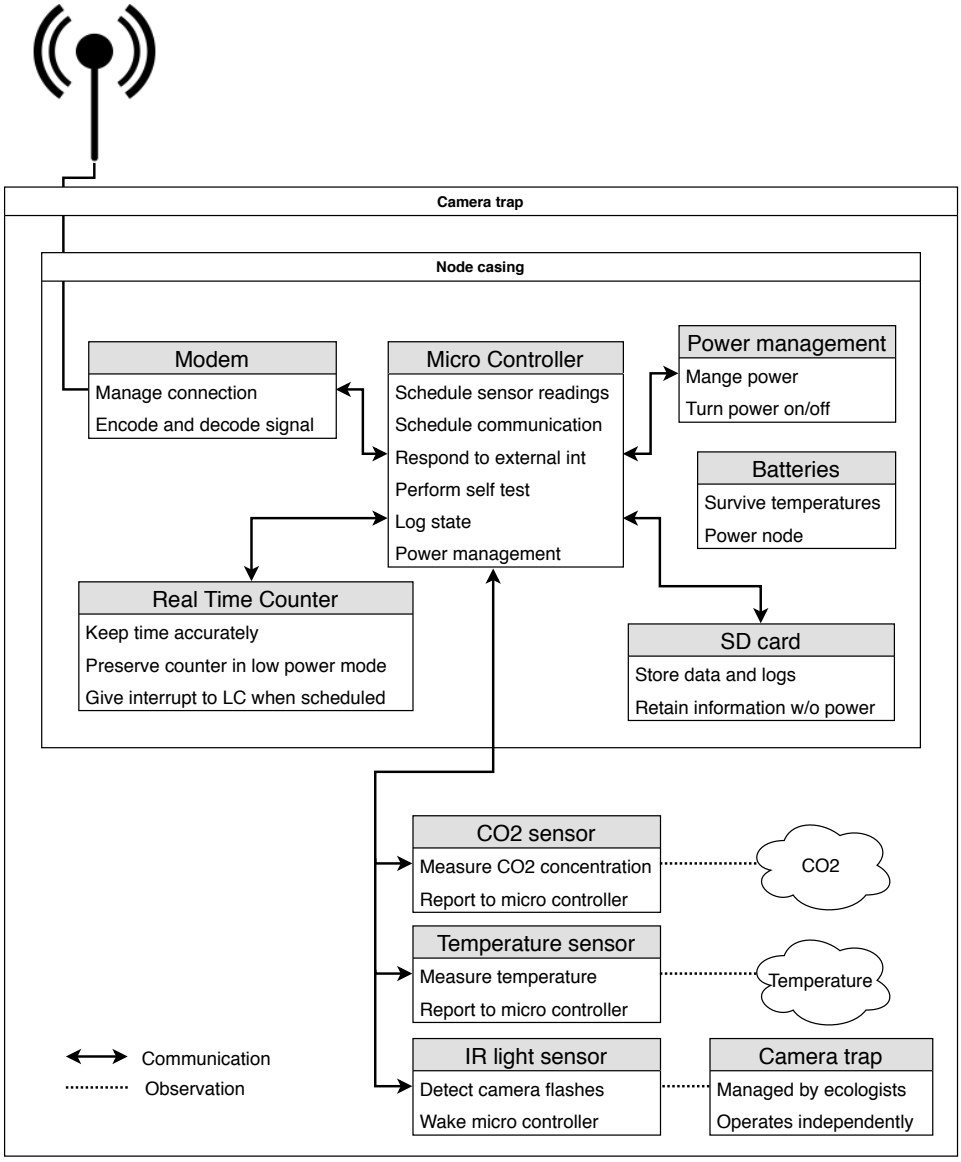

Figure 5.1: Diagram showing the hardware design (Diagram: Øystein Tveito)

5.1.1 Logic controller

The logic of the sensor node will be handled by a micro controller. It would be preferable if the ratio of time where the micro controller is in low-power mode is as high as reasonably possible. As such, it is important that the micro controller is powerful enough to complete it's assigned tasks in a speedy manner. In addition, the power consumption in low-power mode should be considered of high importance. The controller must have limited internal storage accessible

from the user program. The storage is intended for state variables for the program, thus the size only needs to be a few bytes in size.

5.1.2 Casing

The main board and the batteries of the sensor node will be placed in a waterproof case. The sensor node needs to be able to function while completely submerged in water. This is because the camera traps are not waterproof, and are prone to flooding. The box must be able to fit in one of the camera trap compartments.

5.1.3 CO_2

The CO_2 sensor must not have an error in excess of $\pm(300 + 15\%)$ppm. It must be able to provide measurements within a few seconds of powering up. Due to the limited power budget, power efficiency is an important concern for sensor selection. The CO_2 sensor must be mounted on the outside of the sensor node casing. As such, the sensor must either be waterproof or protected by other means.

The sensor technology chosen must measure CO_2 directly. Some "CO_2" sensors measure other components in the air and returns an "equivalent CO_2" (eCO_2) value. CO_2 is a stable gas and hence it is hard to measure it with normal chemical or resistive sensors. The eCO_2 is based on known ratios of volatile components in common CO_2 sources. Those can be effective in estimating CO_2 indoor or from cars. In our case, however, it is not certain that the CO_2 sources will produce the other components in the same ratio.

5.1.4 Temperature sensor

The temperature sensor will be mounted on the outside of the sensor node casing. It must not have a measurement error in excess of $\pm 0.5°C$ when within our expected temperature range.

5.1.5 IR light sensor

The wild-life camera uses an infra-red (IR) flash to illuminate rodents in the camera trap. Therefore the camera health can be determined by using an IR light sensor to detect these flashes. In order to do this, it is important that the IR sensor has an unobstructed view of the main camera trap compartment. As

this sensor must be active continuously, it is especially important to consider power efficiency.

5.1.6 Storage

The node must have a SD-card connected with a storage capacity of at least 2GB. The micro controller must have an internal, non-volatile storage of at least 4MB.

5.1.7 Real Time Counter

An RTC will be responsible for timekeeping. Time drift should be limited to no more than a few seconds per day.

5.1.8 Batteries

Batteries needs to sustain the node for one year of operation. The batteries chosen are the same AA lithium batteries used in the camera.

5.1.9 Visual feedback device

The sensor node must have the ability to give visual feedback to the operator during deployment. A RGB LED will be used for that purpose. This light must be visible form the outside of the unit with the lid closed.

5.1.10 Communication

Communication will be accomplished with LTE CAT M1. LTE communication will limit the amount of infrastructure this project need to be deploy.

5.2 Software design V1.0

In this section, the software design of version 1.0 of the sensor node is presented. The design is laid out in Figure 5.2.

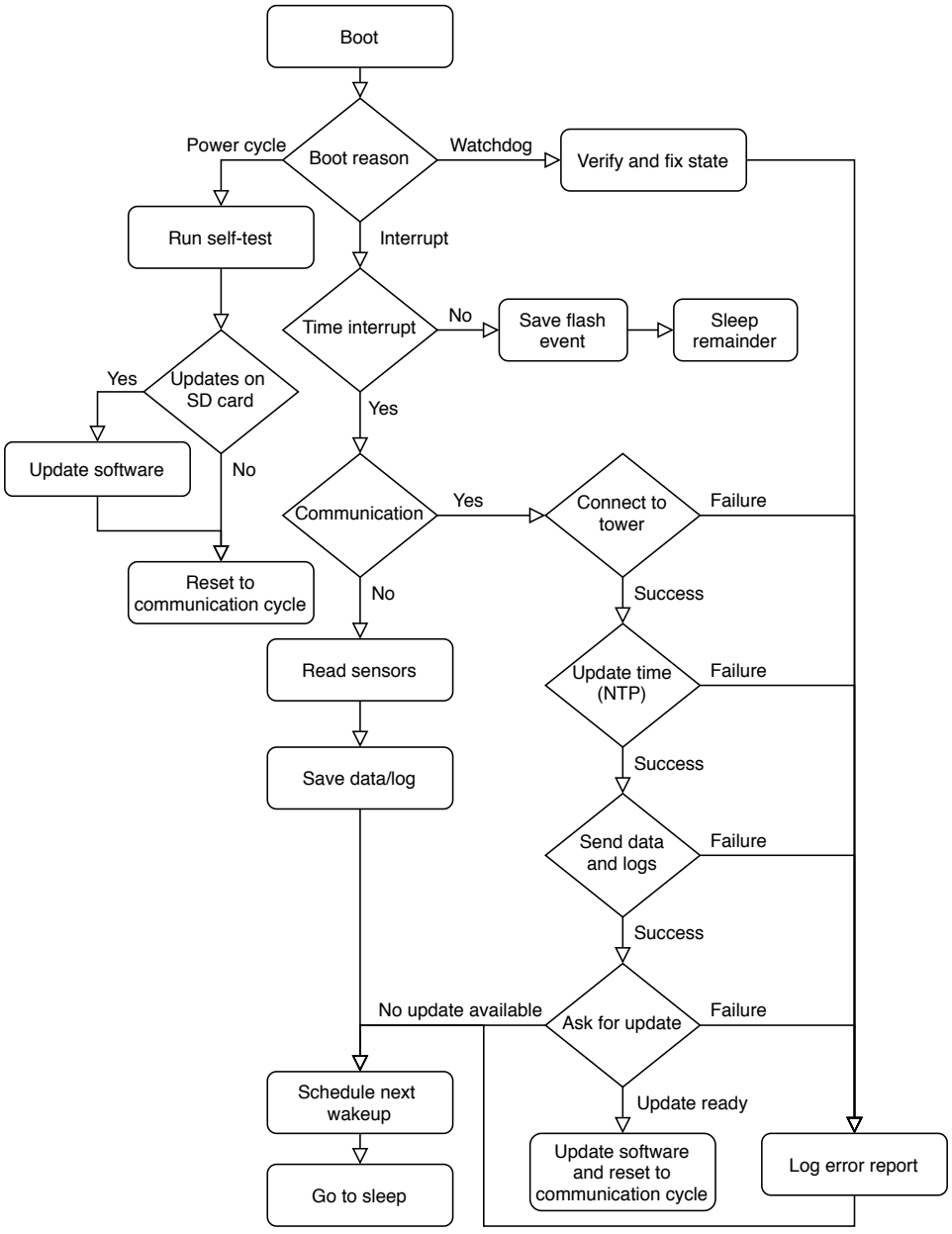

Figure 5.2: Diagram showing the software design of software version 1.0 (Diagram: Øystein Tveito)

A schematic representation of the software design is presented in Figure 5.2.

5.2.1 Operating system

The program running on the device will have almost total control over the hardware as the underlying operating system is a minimal real-time operating system. The program therefore needs to handle all scheduling and communication explicitly.

5.2.2 Boot procedure

On boot the program will check what caused the activation. There are three alternatives: power on, watchdog timer reset, or an interrupt. Each activation type is handled differently, as described below.

5.2.3 Power on

When the node detects that it was powered on it should run through a self-test procedure. The results of the self test must be presented visually to the operator. It should then check the internal storage medium if there are any updates ready to install. If so, the update should be installed. It should then schedule an immediate communication cycle and reset.

5.2.4 Watchdog reset

The watchdog mechanism is a timer which restarts the system when it reaches 0. In order to avoid triggering this mechanism, the system must regularly reset (or "feed") the watchdog timer. If the watchdog reset has been activated, the program has gotten stuck at an unknown point during execution.

If a watchdog reset was triggered while doing a self test, the self test runs again until it is completed successfully. In any other state, the program will not try to resume its previous activity. It will instead schedule the next wake up time, and go to low-power mode.

5.2.5 Interrupt

There are two types of interrupts. One is a scheduled interrupt for a communication or measurement cycle. The other is due to an external event detected by the camera sensor. The sensor node checks which type of interrupt occurred.

In the case of an external interrupt, the node records the event to the internal storage of the micro controller. The node determines how much time is remaining until it's next scheduled wake-up, and activates low-power mode for this duration. For measurement cycles, the node activates and reads all sensors. It then records this information to the external storage.

5.2.6 Communication cycle

In a communication cycle, the first task is to establish a connection to a cell tower. If this fails, a back-off timer is increased by one day. At each failed attempt, the back-off timer is incremented by one day, up to a maximum delay of 7 days between communication cycles.

In the case of a successful connection, the first task is to synchronize time. The back-off counter is also reset. The node is to connect to "pool.ntp.org", using DNS to resolve the host name. Then the node must connect to the data server at UiT. The data is to be sent first, then logs. The node must then check for and apply any available updates.

5.2.7 Scheduling

After all tasks are completed, the node will schedule the next task and enter a low-power state. Sensor cycles are scheduled with a fixed period of 30 minutes. Communication cycles are normally scheduled daily, where communication occurs at a random time between 03:10 and 03:20 GMT. This is done to reduce interference between nodes, which may happen if all the nodes tried to communicate at the same time. The number of days between communication cycles is dependant upon the back-off counter.

5.3 Software design V1.1

The software design of version 1.1 is almost identical to version 1.0. There are small, but significant differences in the design. These differences were introduced due to experiences from field-tests of version 1.0. The rationale of these changes are described in Chapter 8. The design is laid out in Figure 5.3.

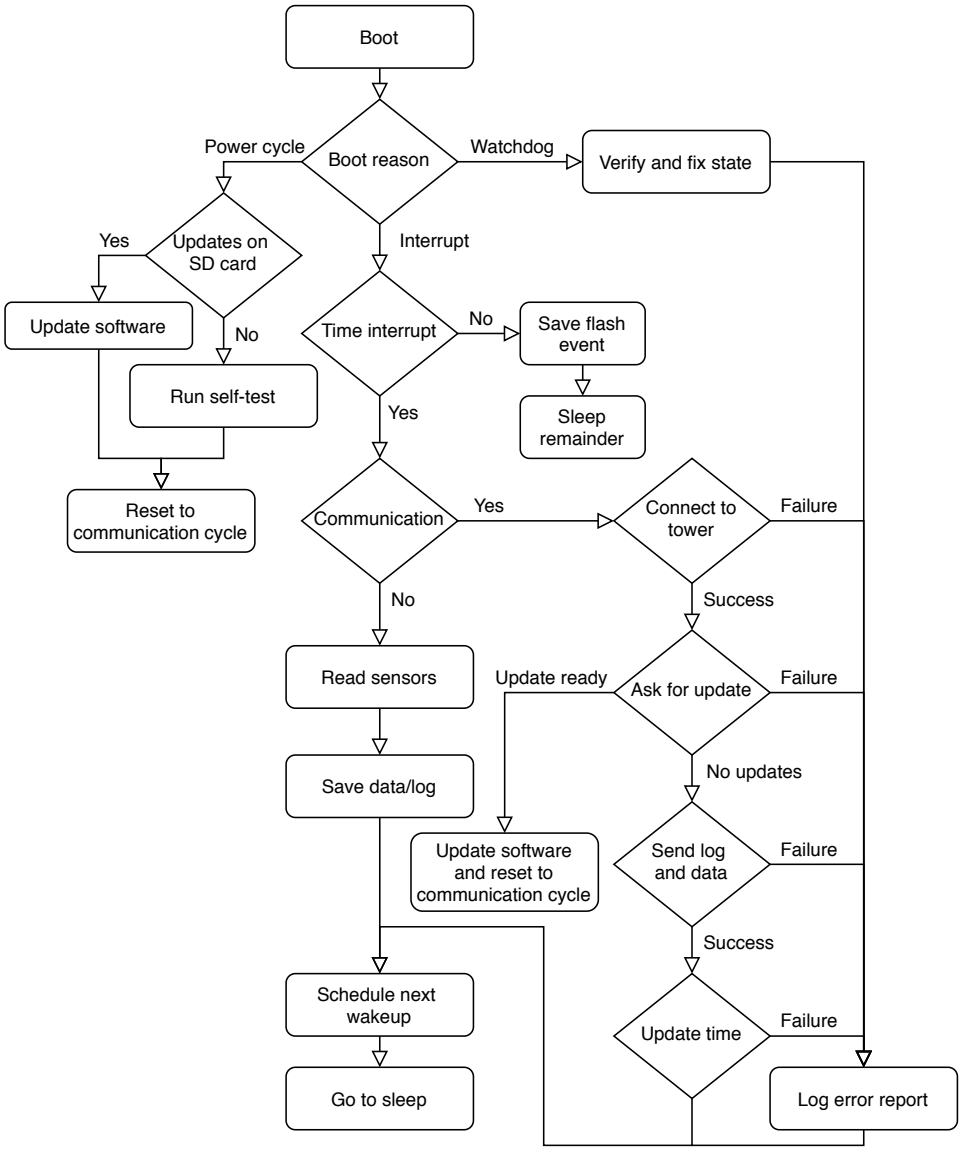

Figure 5.3: Diagram showing the software design of software version 1.1 (Diagram: Øystein Tveito)

A schematic representation of this software design is presented in Figure 5.3.

5.3.1 Self-test

When the node boots and detects a power cycle, the self test is administered as before. In this version, if a watchdog timer is triggered during the self-test, the

self-test is not restarted. This eliminates the possibility of an infinite self-test loop.

5.3.2 Communication cycle

This design no longer relies on DNS look-ups. The node configuration has hard coded IP addresses for each of the servers it is supposed to reach. The NTP server is changed from "pool.ntp.org" to an NTP server hosted at UiT. This sacrifices redundancy to allow for a hard coded IP address.

The order of communication in version 1.1 is different to that in version 1.0. The node will first check for and apply available updates. After this, the node will send recorded operation logs and then gathered data. Lastly, it will contact the NTP server and update the RTC.

5.4 Software design V2.0

While the software design of version 2.0 is drawing on the experiences from the previous versions, it is mostly redone from scratch. There are several differences in this version compared to the previous designs. The design is laid out in Figure 5.4.

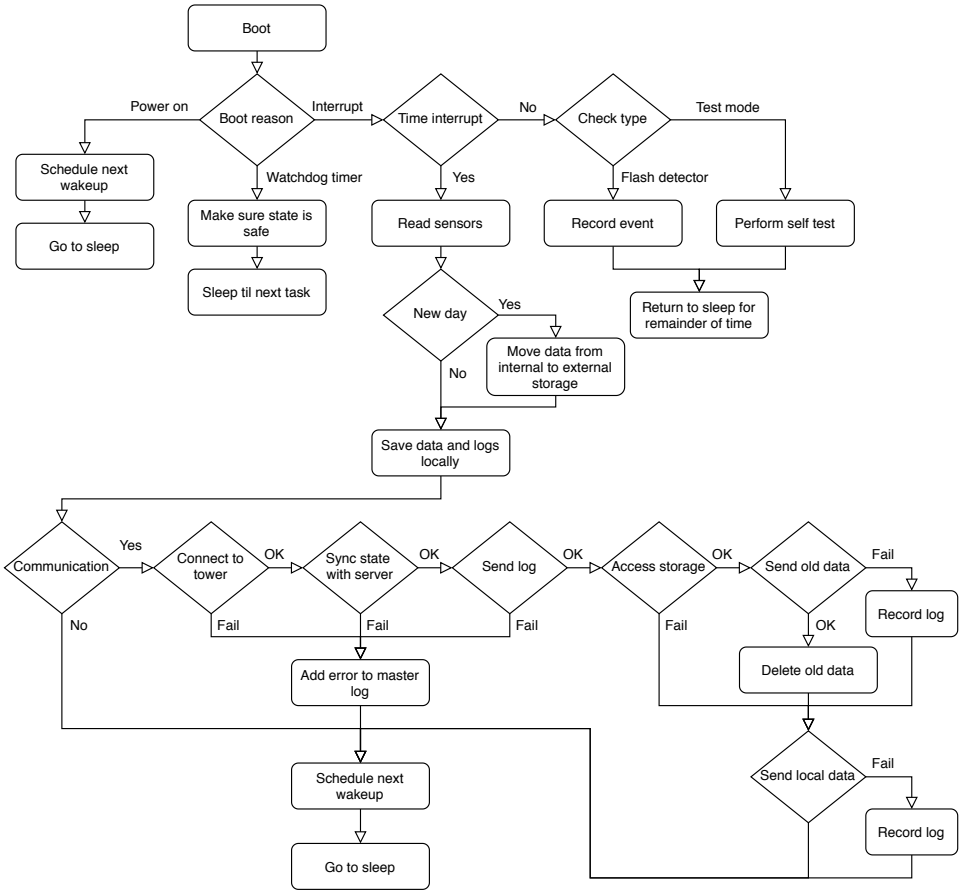

Figure 5.4: Diagram showing the software design of software version 2.0 (Diagram: Øystein Tveito)

5.4.1 Boot procedure

On boot, the cause for the activation needs to be determined. In this version of the design, when a power on is detected, it will calculate the next scheduled measurement time immediately, and immediately enter low-power mode. A self test is not performed automatically.

5.4.2 Watchdog reset

Watchdog timer resets are handled in the same manner as version 1.1.

5.4.3 Events

Interrupts can, as in previous versions, either be timer based or event driven. In this version we introduce a new event. This event is an external interrupt, triggered manually by the operator. There are therefore two different types of external interrupts and the node will need to distinguish between them. The new event is used to initialize the self-test functionality. The procedure for the camera sensor event has not changed.

5.4.4 Scheduled interrupts

In this design, timer based interrupts are only used to initiate measurements. At every scheduled event, the sensor readings are recorded. As opposed to previous designs, the internal storage of the micro controller is used to record both measurements and logs. The data is moved to the external storage once a day. Limiting the number of times the program writes to the SD-card will drastically reduce the risk of corruption. The logs are always kept in internal storage to allow for logging in the event of corruption of the SD-card.

5.4.5 Communication

The node will make an attempt to communicate once a day.

This is done right after a normal sensor cycle. The node will first try to connect to a cell tower. If it succeeds, it will try to synchronize its state with the data server. This includes time, software version and configuration. It will then send the master log, always saved in internal storage to the server. If any of these steps fails, the error will be logged and the node will schedule the next task and go to sleep.

If communication is established, the node will first try to access the external storage. If successful, it will send all data on the external storage to the server and then delete it. If a failure occurs it will try to send the locally stored data and after that delete it. After these steps have been completed, the next task will be scheduled and the node will go back to sleep.

/6 Implementation

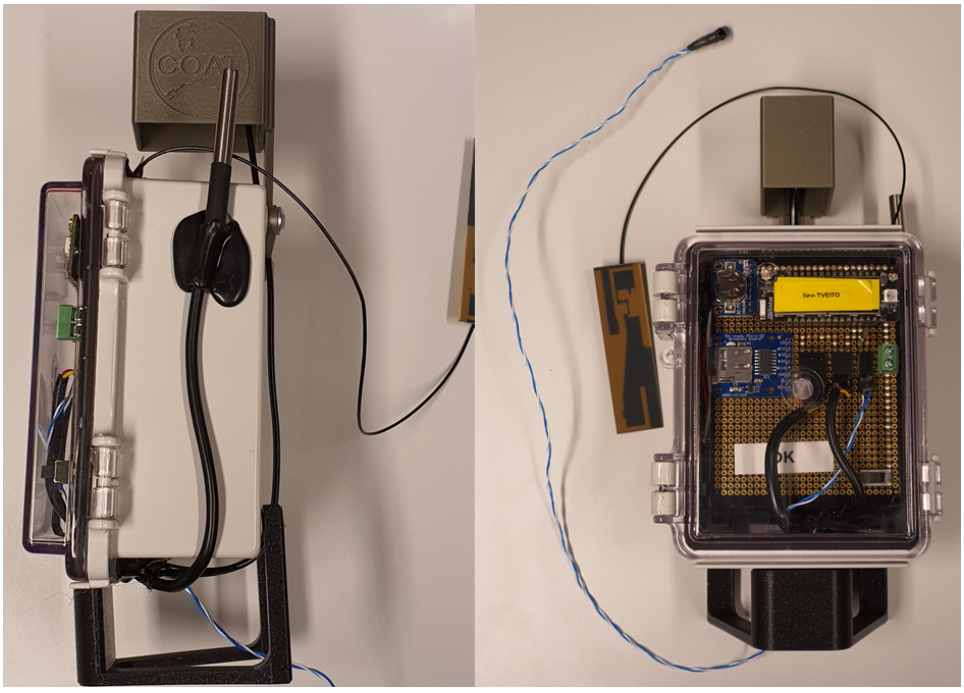

Figure 6.1: A picture of a finished version 1.0 of the CO_2 sensor node from the side (left) and front (right) (Photo: Øystein Tveito)

This chapter will describe the implementation of the hardware, software, and the physical construction of the sensor node. We present some arguments and reasoning for the different choices made in later chapters.

6.1 Hardware version 1.0 and 1.1

There are no differences in the implementation of the hardware between version 1.0 and 1.1. The below description is valid for both.

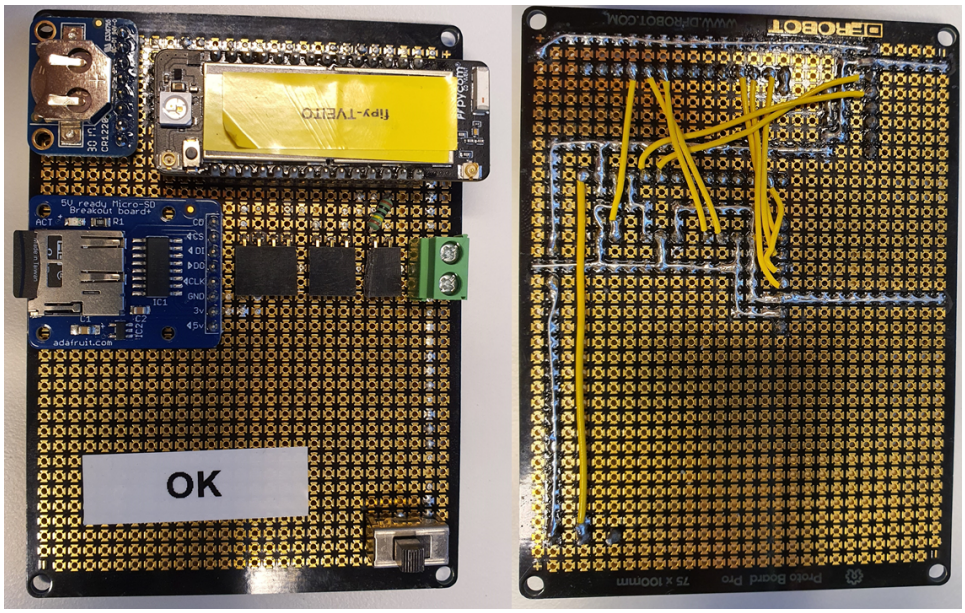

Figure 6.2: Picture of the front (left) and back (right) of the version 1.0 main board of the co2 sensor. Dimensions are 75mm·100mm. (Photo: Øystein Tveito)

6.1.1 Parts

The parts for the node have been selected based on the design and requirement constraints given in previous chapters.

Micro controller

The micro controller chosen for the project is a FiPy[32] from Pycom. This is a ESP32[10] based micro controller that runs MicroPython(µPython)[27]. The FiPy also integrates LoRa, SigFox and LTE-M modems, the latter including NB

IoT[6] and CAT M1[16].

CO_2 sensor

The carbon dioxide sensor chosen is the ExplorIR-w-20[15] from CozIR. The sensor is based on NDIR technology and can measure from 0% to 20% CO_2 concentrations.

Temperature sensor

For measuring temperature, the DS18b20[23] temperature sensor in a water-tight package is chosen. This allows the sensor to be directly exposed to the elements without risking component failure.

RTC

We chose a breakout of the DS3231[22] as the RTC. This is a low cost, but accurate RTC chip with internal temperature compensation.

SD-card

The SD-card was chosen to be a Panasonic RP-SMLF08DA1. This is a cheap, 8GB SD-card that can handle temperatures down to $-25°C$. The SD-card is slotted into an SD-card breakout board from Adafruit with a 2.54mm pitch. The SD-card breakout board simplifies access to the pins of the SD-card.

Camera monitor

The camera monitor is an unknown general purpose IR NPN transistor left over from a previous project. We do not have a part number for this, but most IR NPN transistors with the correct wave length should work for this purpose.

Power management

The power to the SD-card, the CO_2 sensor, and the temperature sensor is cut while in low power mode by a N-channel MOSFET[18]. The MOSFET is wired between the ground connection of the parts and the ground of the node.

Batteries

The battery type chosen is the Energizer Ultimate Lithium[9]. The same batteries are used by the ecologists in the cameras this project is co-located with. Packs of three batteries batteries in series gives $1.5V * 3 = 4.5V$ nominal voltage. The node have four of these battery packs connected in parallel resulting in $3500mAh * 4 = 14Ah$ of capacity.

6.1.2 Budget

The cost per sensor node is now as follows:

Computing parts		
Microcontroller	Pycom FiPy	$60
MicroSD breakout	Adafruit 254	$8
Real-time clock (RTC)	Adafruit DS3231	$14
Sensors		
CO_2 sensor	Gas Sensing Solutions ExplorIR-W-20	$160
Temperature sensor	DFRobot DFR0198	$7
IR photo transistor	Unknown NPN	<$1
Communication		
LTE Antenna	Pycom LTE antenna kit	$17
SIM card	Telenor SIM card with data subscription	donated
Electronics		
Connector board	DFRobot FIT0203 78 mm x 58 mm	$2
Power switch	NKK MS12ANA03	$5
Pin header kit	Adafruit 4174	$2
MosFET	MOSFET N-CH 100V 17A TO-220AB	$1
Resistors	1M and 2 x 1K 1/4W	<$1
Enclosure		
Enclosure	Polycase WH-02-03 Enclosure	$14
Battery holders	4x Memory Protection Devices SBH331A	$10
Hood for CO_2 sensor	custom 3D printed	<$1
Feet	custom 3D printed	<$1
Accessories		
MicroSD card	Panasonic RP-SMLF08DA1	$11
Batteries	12x Energizer L-91 1.5 V Lithium	$17
RTC battery	Clas Ohlson CR1220	$2
Silica packet	1/6U	<$1
Total		$330

6.1.3 Assembly

The main board of the CO_2 sensor is created on a perforated board with 2.54*mm* spacing. All parts are soldered by hand to this, and traces are either manually created by solder between holes or by wires going from point to point (see Figure 6.2).

6.2 Hardware version 2.0

The new version of the hardware has a few improvement in parts and manufacturing process.

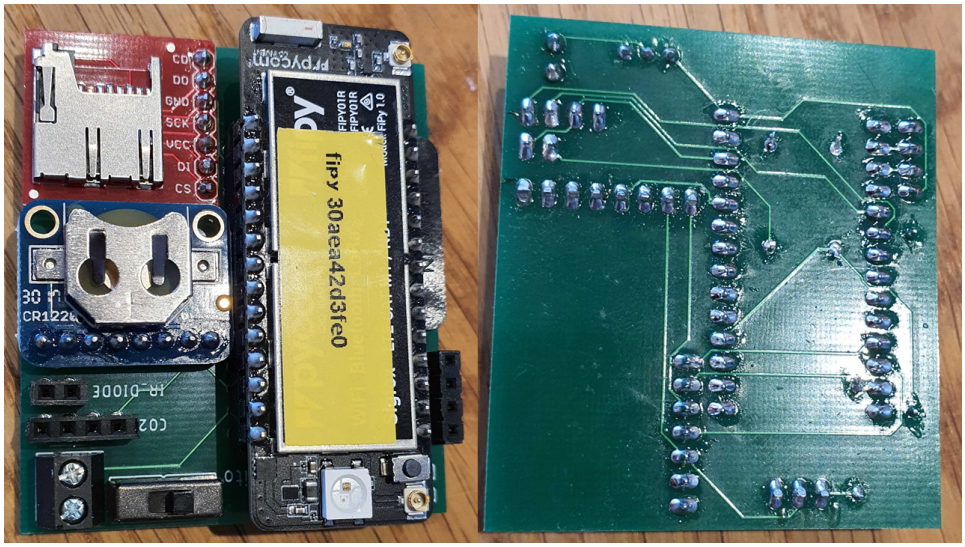

Figure 6.3: Picture of the front (left) and back (right) of the version 2.0 main board of the co2 sensor (Photo: Øystein Tveito)

6.2.1 Parts

There have been two substitutions, one added component, and one component is removed. The battery capacity has also been changed.

CO_2

The ExplorIR-W-20 CO_2 sensor has been replaced by the ExplorIR-W-EH-20. The added "E" means that the temperature range is extended. The new sensor can operate from $-25°C$ to $55°C$. The "H" means it has a built in temperature and humidity sensor that is used to give more accurate results.

Temperature

The dedicated temperature sensor is removed from this implementation. Temperature readings are now given by the temperature and humidity sensor on the CO_2 sensor.

SD-card

The SD-card is replaced by an industrial grade SD-card[1] with 2GB storage. The new card is based on single-level-cell technology. This should be more robust than the previously chosen SD-card.

Reed switch

A reed switch have been added to allow waking up the unit with a strong magnetic field. This is used to invoke the self-test procedure.

Batteries

The new main board of the sensor node is significantly smaller than that of the earlier versions. This allows for more batteries to be fitted inside the sensor node casing. Two additional sets of batteries have been fitted, for a total of six sets of three. This gives the newest version a total battery capacity of 21, 000mAh.

6.2.2 Assembly

The assembly process have changed from soldering by hand on a perforated board to a dedicated Printed Circuit Board (PCB). The PCB have holes for all the components, and all the traces between components are within the PCB.

6.3 Physical implementation

The physical implementation is mainly restricted by the hardware implementation and the requirements. The node needs to fit inside the designated compartment of the camera trap. The sensors need to reach the environment they are monitoring. Finally, the signal coverage must be taken into account as the camera trap is made of metal.

All versions have a similar implementations, with a few alterations. Version 1.0 will be described first. For the following versions, only the modifications will be described.

Figure 6.4: Picture showing the a unit deployed in a camera trap before the antenna and camera sensor is secured (Photo: Michael J. Murphy)

6.3.1 Version 1.0

The casing for the unit is a NEMA 4X[30] box from Polycase[31]. It has usable internal dimensions of $110mmx80mmx55mm$. The external dimensions are $130mmx100mmx70mm$.

The battery packs are in the bottom of the case and the main board is placed on top of them. This allows the operator to see the main controller and the RGB light during the self test.

The CO_2 sensor is encased in a 3D printed hood. The hood protects the sensor from water from the top and sides. It has no bottom, allowing the sensor to be in direct contact with the air in the camera trap.

The CO_2, temperature and camera sensor cables are routed out of the bottom of the sensor node. The hole is sealed with ethylene-vinyl acetate (hot glue) that allows for a watertight seal, even with temperature variations.

The antenna must be placed on top of the lid of the camera trap. The antenna wire is too short for it to be routed out of the bottom of the node. Thus, an additional hole had to be made at the top of the sensor node.

The camera sensor is routed over the separation wall between the side compartment and the main camera trap compartment, directed towards the camera.

The antenna is routed out of the camera trap and adhered to the top of the lid with a layer of painters tape, then a layer of packaging tape.

6.3.2 Version 1.1

This version adds a standoff that is slotted into the side for the sensor node box. This protects the SD-card that is protruding from the main board.

It also adds a standoff for the antenna of 10*mm*. The standoff is 3D printed with minimal amounts of plastic to minimize electromagnetic impedance. A 3d printed cover is added to protect the antenna from the environment

6.3.3 Version 2.0

As the new PCB is designed with the SD-card in mind, the stand offs are no longer needed. A 3d printed fitting is installed in the box instead to make everything fit together and hold everything in place.

6.4 Software implementation

All versions of the software is implemented in µPython on version 1.18.1.R4. Version 1.0 and 1.1 are implemented by Michael J. Murphy, the PhD student. Version 2.0 is implemented from scratch by Øystein Tveito, the author of this thesis. A quick overview of some of the changes will be given here:

Feature	V1.0	V1.1	V2.0
Watchdog timer	10 seconds	60 seconds	180 seconds
Log storage	External only	External Only	Internal storage Pushed to external storage daily
Data storage	External only	External Only	Internal storage Pushed to external storage daily
Can operate w/o SD-card	No	No	Yes
Communication interval	Daily with back-off	Daily	Daily
Communication schedule	Separate wake-up	Separate wake-up	Combined with measurement cycle
OTA software update	Yes	Yes	Yes
OTA firmware update	No	No	Yes
Checksum on updates	No	No	Yes

Even though the newest version of the source code contains much of the same functionality, the code is written from scratch. The newer code is based on the experiences gathered from the testing of the first two versions.

The watchdog timer was decided to be set to 180 seconds instead of the 10 and 60 seconds as in previous versions. This was done to keep its intended functionality of only interfering with a running program if the program is stuck or otherwise can not function as intended. The use of the *feed* function is also heavily reduced as a consequence of this. This function is only called once in measurement, once when transferring files from internal to external storage, and once for each attempt at communication with the server. Previous versions of the code had to feed the timer more frequent as the timer was shorter. This makes the code more complex.

As described in the design of the version 2.0 software, the data is now stored primarily in internal storage, and only pushed to the SD-card once each day. To simplify scheduling of this, the first time a reading is recorded in a day all old data is pushed to SD-card. As data and logs are stored in files named as the current date, detecting a new date can be done by checking if a file containing

the current date is present.

With data stored in internal storage, the node can theoretically operate as normal without an SD-card. No planned scenario involves deploying nodes without SD-cards, but this is useful in the event of a corrupted SD-card. Older versions of the software would not function after the corruption of the SD-card. In the last version we would still be able to get both data and logs as long as the node is in regular communication with the network. We would also be able to determine that a node has a corrupt SD-card from the logs. This makes remote debugging easier.

Communication is scheduled as frequent as in previous versions. The only difference is that it is combined with a normal measurement cycle. This saves the node from one wake-up event. As there is some overhead with waking up, this saves energy. The random scheduling scheme done in version 1.0 and 1.1 was done to eliminate interference from neighboring nodes. The nodes use a cellular network designed for several hundred simultaneous users. Further, the nodes are located in a remote location with minimal competing users. This feature was therefore deemed unnecessary. If, in the future, the node would be based on different network communication technologies, this feature could reintroduced.

The update functionality is in the older versions based on a custom function. Files are downloaded from the backend server, and the files in the file system are replaced. This function does no checksum functionality. Not checking the checksum of files being downloaded introduces the risk of corrupted update files. There is also no support for firmware updates. The newest versions rely on the built-in Over The Air (OTA) function implemented in µPython. This function uses checksums to ensure that files are not corrupted. Firmware updates are also supported by this function.

7

Version 1.0 design and implementation discussion

This chapter will explain some of the choices made in the architecture, design and implementation of version 1.0 in greater detail.

7.1 Characteristics

Selecting the appropriate parts is an important step in the process. When selecting sensors there were predominately five distinct characteristics that needed to be weighted against each other: accuracy, reliability, power consumption, price, and ease of use.

7.1.1 Accuracy

This is the ability to get accurate readings from the sensor. It is often one of the first issues that must be determined. The accuracy required varies tremendously depending on the application. The accuracy needed in a magnetometer in a phone intended for use in navigation will be several orders of magnitude lower than the accuracy needed when monitoring the shift in the earths magnetic fields. Similarly, we have to decide how accurate the readings from the different

sensors we chose needs to be. Accuracy will likely come at the expense of either cost or power consumption, and in most cases both.

7.1.2 Reliability

The short explanation of reliability is how much we can depend on the sensor, but in our case, there is mainly one concern that needs to be addressed. The hardware needs to survive the harsh climate in the arctic tundra. This will in most cases be a Boolean and absolute concern, either it is good enough to survive, or it is not. If a part is made to survive the climate we expect the units to be exposed to, it will be good enough for us.

7.1.3 Power consumption

Since our device will need to survive for at least a year on batteries this is one of the most important factors. Two main factors determine if the part is suitable for our purpose: The average consumption of power, and the warm-up time. Many sensors have a specified warm-up time, meaning it needs anywhere from a few milliseconds to a few minutes before they can deliver reliable data. This becomes significant if the entire system needs to be powered on for longer while waiting for the sensor to deliver data. Then it is no longer the power consumption of the sensor times the time needed, but the consumption of the entire system times the additional wait time.

7.1.4 Cost

Part of the goal is to gather data from as many locations as possible to get a holistic view of how the sub snow conditions are effected by today's and tomorrow's climate. Subsequently, the cheaper we can get each of these units, the more units we can produce. We would rather get data of slightly lesser quality from hundreds of locations, than high quality data from a few locations. Therefore the cost needs to be taken into account when choosing the parts for the project.

7.1.5 Ease of use

This might be one of the most important factors to consider in this project. From the project got conceived to our defined deployment day, not much time was available for producing and getting the different parts to work together. Having parts that are easy to interface with, preferably with a software library

already existing for our logic controller, is a crucial factor for finishing the project on time. For this project, we therefore decided early on that we would use breakout boards for every part where one were available.

A breakout board is a separate PCB with all the necessary electronics, and sometimes even a separate logic controller, for the sensor to work with minimal effort. With a breakout of a sensor, it is usually just required to wire up power and ground, and the required signal lines. They also normally come with a pin-out consisting of pins with 2.54mm spacing, the de facto standard of hobby electronics. This allowed us to use a perf board with the same spacing, and we could solder all the component to a preexisting PCB and wire up all the components ourselves. Thus there was no need to have a custom PCB created at the beginning of the project.

7.2 Communication technology

Communication is one of the main features of this sensor node. We wanted to get the data back to the scientists analyzing it as soon as possible. To accomplish that we needed a communication technology that works in remote locations and that does not deplete the batteries too quickly. Four technologies have been discussed for us to use: satellite, NB-IoT, CAT-M1 and LoRa.

7.2.1 Satellite communication

The major upside to satellite communication is that there is near-global coverage. With a satellite transceiver we can establish two-way communication from anywhere and not rely on the infrastructure where we are deploying. The downside to satellites is delays and power consumption. When looking at transceivers from iridium, one of the few commercial satellite data providers, their transceiver requires up to 1.3A peak current and 145mA average during transmit and 156mA peak during receive. This would limit our communication budget by draining the batteries quite quickly.

Iridium is using the 1.6GHz band to communicate to and from the satellites. This frequency does not have the same penetration power as the other communication technologies we considered which are operating in the 800 and 900MHz bands. The basic principle is the lower the frequency, the better the penetration.

7.2.2 NB-IoT

NB-IoT, formally known as LTE CAT M2, is one of the newly established LTE protocols. It is made specifically for IoT devices and is a network provided by telecom companies. It is using Direct-Sequence Spread Spectrum (DSSS) modulation which is specifically designed to reduce interference with other signals. The range of NB-IoT is also significantly greater than traditional LTE (4G) networks.

One of the drawbacks of NB-IoT is the reduced data rate you get. The downlink is capped to 27.2Kbit/s and uplink to 62.5Kbit/s. This makes it hard to transfer firmware updates or other files to and from the units in the field when desirable. The network standard also allows delays of up to 10 seconds for a packet to get through, severely limiting the usefulness of TCP or other higher-level protocols for guaranteed delivery of data.

Telenor and Telia are the two telecom companies in Norway offering this network. At the time of writing, both Telenor's and Telia's coverage maps for NB-IoT indicates coverage at the planned test site[38, 39], but further east for several of the planned deployment sites Telenor states they have no coverage at all (Figure 7.1), and Telia's coverage is spotty at best (Figure 7.2).

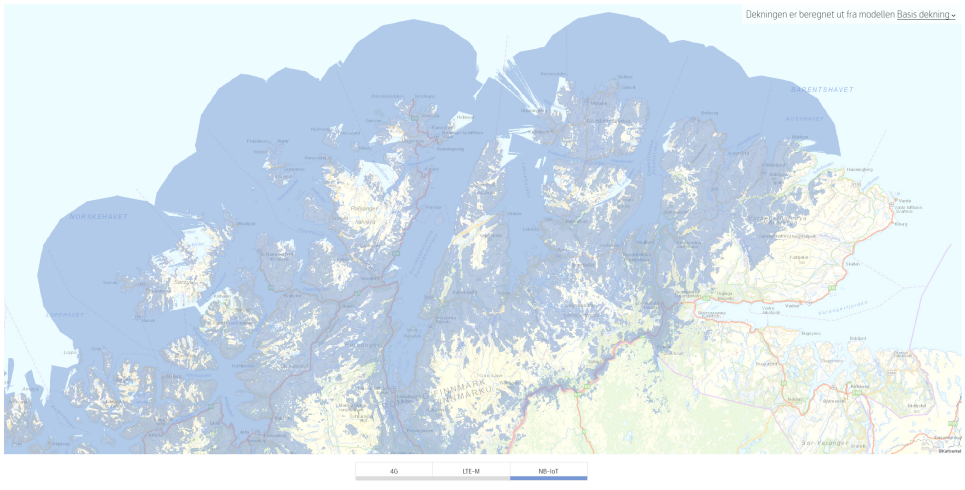

Figure 7.1: Screenshot from Telenor's coverage map of NB-IoT (Source: telenor.no/bedrift/iot/dekning/)

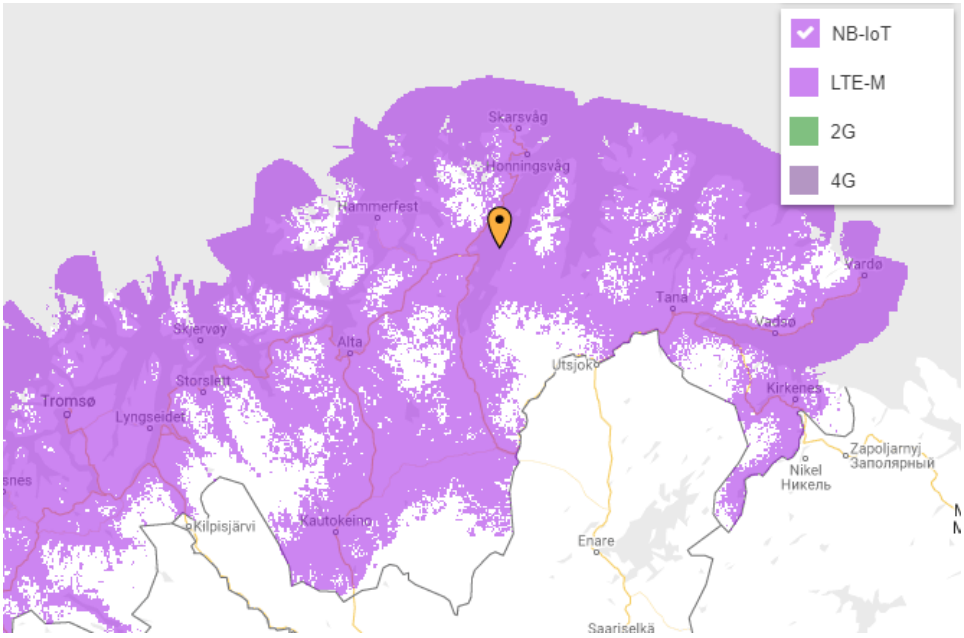

Figure 7.2: Screenshot from Telia's coverage map of NB-IoT (Source: telia.no/dekning/)

7.2.3 CAT-M1

This technology is also defined together with NB-IoT, but this network has a significantly higher bit rate, namely 300Kbit/s up and 375Kbit/s down. Both CAT-M1 and NB-IoT have a higher bit-rate than what is stated here according to the protocol, but that requires multiband modems, which are both more costly and use more power and is therefore not taken into consideration.

Cat M1 is offered through telecom companies. At the time of deployment, Telenor had coverage comparable to NB-IoT, and also covering the eastern part of Finnmark (Figure 7.3). Telia, on the other hand, had only a few test networks in the southern part of Norway. They now have a nation-wide coverage.

The network delay is also much lower than NB-IoT, with a guarantee of <1s. The downside to CAT-M1 is that when compared to NB-IoT it uses more power to transmit. This is however mitigated in practice by the fact that the bit-rate is much higher on CAT-M1. The time needed to send is much lower and therefore comparable amounts of power should be spent regardless of witch of the two aforementioned LTE-M technologies are used.

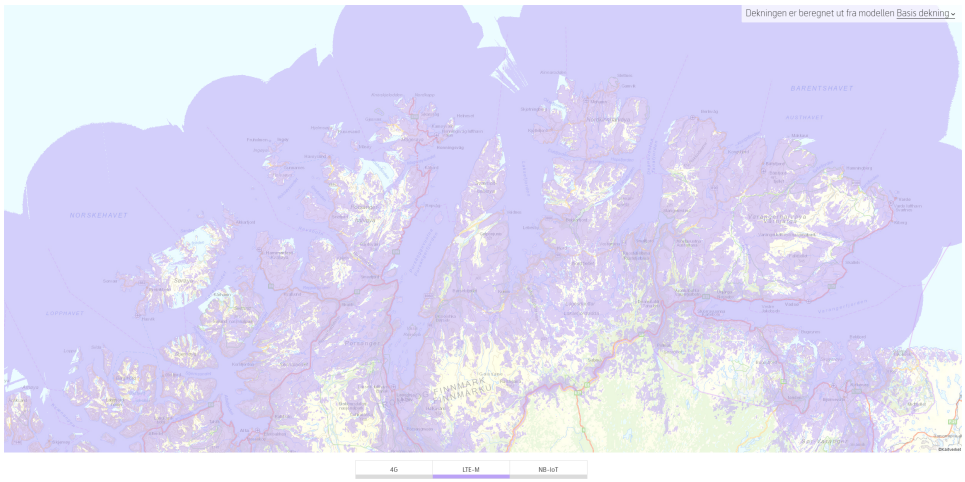

Figure 7.3: Screenshot from Telenor's coverage map of CAT-M1 (Source: telenor.no/bedrift/iot/dekning/)

7.2.4 LoRa

The last option we considered was LoRa. LoRa has many of the same properties as the two LTE technologies, but with the added benefit of being on an open radio band. This means that we could set up our own gateway for all the devices to connect to and place the gateway on a location with line of sight both to the nodes and to a radio tower offering 4G or other type of network connection. This would allow for more flexibility in deployment, allowing us to deploy in areas without coverage, for example in valleys and network shadows.

This would come at a cost of more complexity of both software and hardware. Firstly, we would need another unit to deploy in a different location than the rest of the units, making deployment more complex. It would also introduce a single-point-of-failure: if the gateway goes down, no communication can be done. This could be compensated for by having all the units act as gateways and allowing units without LTE network to send their data to units with coverage. This would introduce a greater need for time synchronization and added power needs. Time synchronization is not an easy feat in low power, wireless devices. The lack of it would either deem communication a random event or introduce the need for huge time buffers. Devices would need to stay online to listen for other devices that want to communicate when it otherwise could go back to sleep.

Finally, this will in most cases result in a skewed power consumption by requiring the units with coverage to transmit both their own data and the data from everyone else. To make sure this scheme would work, the battery capacity of

all units would need to be increased compared to the scheme where all devices run independently of each other.

LoRa therefore seems like a good solution if we need to deploy in locations we suspect to lack coverage, but with deployments in areas with coverage the other methods might be preferable.

7.2.5 Technology choice

We landed on LTE CAT-M1 for the first prototype as we feel this covers most of our needs without any major drawbacks. The test area is according to Telenor well within coverage from their towers, as is NB-IoT, but the added transfer rate from CAT-M1 is desirable. Satellite communication is too hard on the batteries and LoRa gateway and LoRa meshing drives up the complexity too much with our limited budget and time restraint for the first deployment. Telenor donated the SIM-cards to the project, but as Telia did not have coverage in the area this was not defining for the choice of operator.

7.3 Part selection

The characteristics discussed in Section 7.1 were weighed up against each other to find the ideal parts for our project. For each part we defined the minimum specs, and then tried to find the ideal part to fill that role.

CO_2 sensor

This is the most important reading we are collecting and the most expensive sensor. The first consideration is range of measurement. As previously discussed, prior work has shown cases where the concentration can reach as high as 7%. As we had no way of determining whether the conditions there are similar to ours, we decided to make sure that we could measure up to 20%. This gives us room for higher measurements than previously discovered.

Power consumption is the next crucial point of the sensor. We need to run the node for a year on batteries and to allow this we need the sensors to consume as little power as possible. There are here two factors: power consumption and warm-up time. As the node is in low power mode for most of the time, the warm-up time would be the most important part of this as the entire node has to wait for the sensor to take a measurement before it can go back to sleep.

The choice fell on ExplorIR®-W 20% CO2 sensor[15]. This sensor has less than ten seconds warm-up time and a power draw of just $3.5mW$. The sensor also has an accuracy of $\pm 70ppm \pm 5\%$ which is sufficient for the first stage of this research project. 20% CO_2, or $200.000ppm$, allows it to have a wide range it can measure as we are still not certain what levels we can expect to find.

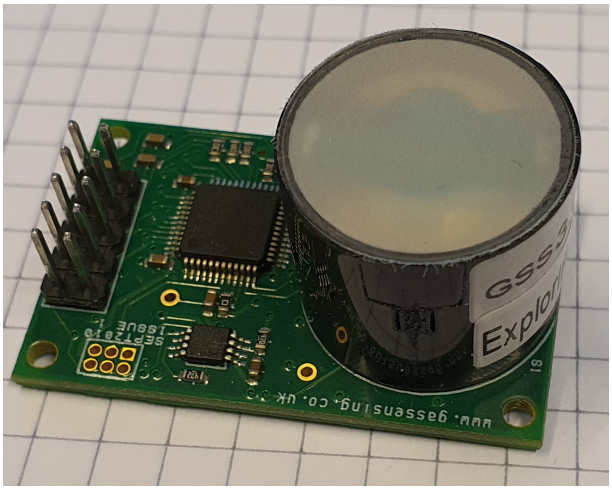

Figure 7.4: ExplorIR®-W CO_2 sensor (Photo: Øystein Tveito)

Temperature

Temperature is probably the easiest sensor to source. There is a vast amount of temperature sensors that are accurate and cheap, and they are avaliable with about any protocol or configuration needed. To get an accurate reading the ideal placement of the sensor should be in open air, outside the box and out of direct sunlight. As sunlight won't be an issue in the winter part of the year we can ignore this point. To place it outside the box, we would want it to be waterproof.

The choice fell on a waterproofed version of the DS18B20[23]. This sensor is easy to work with and have an accuracy of $\pm$ 0.5 °C in the temperature span we are expecting, and $\pm$ 1 °C outside of that. Combined with low power consumption and an insignificant start up time this will fit this project.

Figure 7.5: DS18B20 temperature sensor with waterproof housing (Photo: Øystein Tveito)

Another sensor that was considered, but ultimately rejected was the SHT10[36], a combined temperature and humidity sensor. This sensor have comparable specifications on the temperature sensing, but has the added benefit of a humidity sensor. At the time of design, humidity was not considered an important parameter to measure. Most humidity sensors, including this one, have the added problem that it needs direct contact with air to measure. It was therefore decided that, since the data was not required, the added benefit was not worth the added complexity.

Health check

In order to know that the camera is operating as expected we needed to find a way to monitor its health without interfering with it. As stated before, we are not allowed to monitor it directly, so we had to figure out a way of doing it indirectly. Every time the camera goes off it will give off an infra-red flash to illuminate the event it is taking a picture of. After some testing we determined that that an IR transistor would work to detect this event.

When the flash goes off, the IR transistor will close a circuit to raise the voltage on one of the pins of the micro controller, triggering an interrupt. The interrupt takes the controller out of low power mode, which then records the camera flash to the sensor log.

Another approach that where tried, without reliable results, was to monitor the electromagnetic field emitted by the camera. When the camera activates, the electricity will flow through its circuits and create magnetic fields. The only way we were able to catch this in our experiments was by putting a magnetometer directly next to the camera. This also needed constant monitoring by the micro controller. This was not practical in our use-case, but could be an option in a scenario similar to ours, if the monitored device does not have a flash.

The camera operates by having a motion sensor. Once it detects movement it will take two pictures, a few seconds apart, and then go back to sleep. It was considered adding a motion sensor to our node as well. A motion detector would give us data on how much movement there is inside the camera trap. Comparing this data to the detected flashes from the camera would give us information about whether the camera is operating correctly. The cameras are configured to take one picture every 12th hour even without detecting motion. Hence, we didn't need to detect movement as well, as a lack of events would tell us that the camera is most likely not operating correctly.

Time keeping

A difficult hurdle to overcome with remote devices that are recording events is timekeeping. The way a controller will keep time, even when it is turned of or in power saving mode, is by using a Real Time Counter (RTC). The RTC normally consists of a crystal for counting time, a battery for when power goes off and a small controller for handling communication or events. The power consumption of these devices is negligible. The controller we are using does have a built in RTC, but without an option for a dedicated RTC battery, meaning every time power is cycled, the RTC will lose its time. Furthermore, the internal RTC is not precise enough for our purposes, with a time drift that can reach minutes a day.

Having accurate time keeping could be an important data point for the ecologists. The cameras have a built in RTC that will keep time, but as the camera is without any form of communication, the only way to set the RTC is manually by putting the timestamp on an SD-card. The lack of communication leads to a large time drift over the time of deployment. Correct time is needed to investigate how time of day impacts behaviour beneath the snow. By correlating the camera health sensor with the images output by the camera, a much more precise time could be calculated for each of the images. Thus an external RTC is needed.

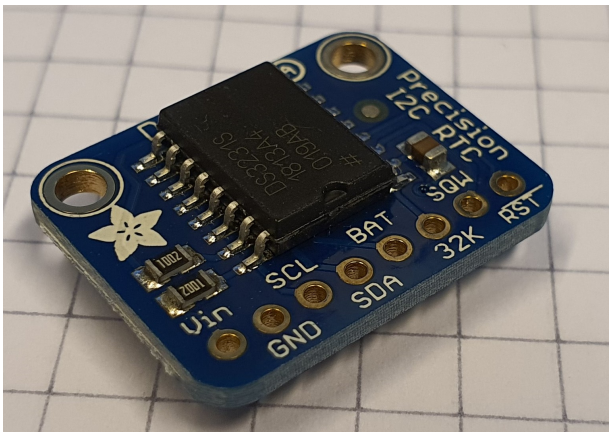

Figure 7.6: A picture of the DS3231 RTC used (Photo: Øystein Tveito)

When selecting the external RTC, price and accuracy were considered the major factors. The accuracy of an RTC is how well it is able to keep time over a time period without correction. The crystal inside is consistent in a stable environment but will be affected by temperature. If the exact operating temperature is known, the drift can be easily calculated outside the RTC, as the crystal have a fixed temperature correlation. In this case, the exact temperature the node would be subject to was not known in advance. The temperature measurements taken by the node is also too infrequent to reliably calculate this drift. The temperature compensation should therefore be done in real time. This can either be done on the node controller or the RTC controller. As this calculation would need to run in real time, the controller would need to be operational all the time. This would severely impact the power consumption. We opted for the DS3231[**?**] with integrated temperature compensation to make sure we could keep time even with long periods between communication. This is a reasonably priced RTC that fulfills our requirements.

7.3.1 Logic controller

The logic controller is the brain of the sensor node and is in charge of scheduling all the readings and communication. When selecting the logic controller we had to consider interface options, power consumption, ease of use and cost.

All the sensors are attached to the controller. It is important that the controller have enough interface options to allow this. The latter years, the selection of micro controllers available have exploded, allowing selection from a huge range of different architectures, operating systems, programming languages and characteristics.

One of the main criteria for the controller is power consumption. As the project is battery powered, the power consumption needs to be kept low. The time resolution for our readings does not need to be that high, meaning that most of the time the entire node can be in low power mode. The logic controller therefore needs to be able to go into a low power state in between readings and communication. This makes some of the more powerful controllers, like the popular Raspberry pi[33], unsuitable as it does not have a low power mode. In addition, controllers with a complete operating system also suffers from prolonged boot times compared to controllers without a full operating system to load.

We also had to consider ease of use as the time frame for the development was quite short. The disadvantage of not having a operating system is that a lot of tools that rely on an operating system will not work on a bare bone system. This means that we will have to develop a lot more of the software to have the unit working.

A few years ago, almost all micro controllers needed to be programmed directly in assembly language, C/C++ or another compilable language. There have been other approaches to this, also in the past, with controllers that supported the interpreted BASIC and FORTH languages[17]. Today even extremely high level languages like JavaScipt is available for micro controllers with runtime interpretation[24]. Something in between this and the classic C/C++ is µPython[27]. Python is a higher level programming language that allows us to do faster development as we have more abstractions available to use in a simpler syntax.

The controller that was selected was the Pycom FiPy[32]. This controller is based on a ESP32[10], a dual core 240MHz micro controller with ample flash and memory that runs µPython. It also has the ability to go into deepsleep, turning off all non-essential components internally, including CPU, main RAM and flash. This drastically reduces the power draw of the unit. The operation is then taken over by the Real Time Counter (RTC) and the Ultra-Low-Power Co-processor (ULP)[10, p. 24]. We can still keep time with the internal RTC, wake up on time events or on external interrupts. The current draw in this mode can theoretically get as low as 10µA but in our experience with he FiPy we can consistently get between 14µ and 18µA, which is orders of magnitude lower than the running consumption of around 60mA.

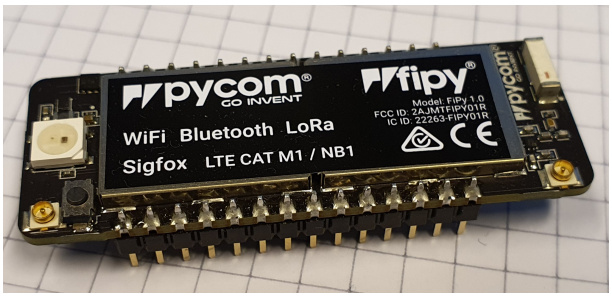

Figure 7.7: A picture of the FiPy micro controller from Pycom (Photo: Øystein Tveito)

One of the advantages with the FiPy is its multi-radio capabilities. It supports WiFi, Bluetooth, sigfox, LoRa, LTE NB-IoT, and LTE CAT-M1. This allows us to use the same controller for a multitude of projects using different communication technologies. This also allows us to change communication technology from software, if the need arises.

7.3.2 Power

The advancements in mobile technology during the last two decades has improved battery technology significantly. This has given us rechargeable batteries that are more dense in power and with a low self-discharge rate. LiPo and Li-IoN batteries is the standard in most battery powered technology today. This allows for long lifetime and recharging instead of throwing away the batteries once they are depleted. However, rechargeable batteries does still have a problem with self-discharge. A rechargeable battery left alone will eventually loose some of its charge. This is a lesser problem with non-rechargeable batteries.

The cameras will function for a little over a year on the batteries installed in them. Hence, someone will need to go to each camera trap once a year to change the batteries. Because of this, the logistics of getting the batteries in our node changed is drastically reduced. We opted to use the same batteries as the cameras use to simplify logistics of the project. The added time used to change the batteries in both the camera and node compared to only the camera is insignificant.

The number of batteries was set by spatial restrictions. We were able to fit four sets of three batteries in the selected enclosure. Based on our estimates and preliminary energy measurements, this would be adequate for its operation. We will in Chapter 12 dive deeper into energy usage and power budget of this and subsequent sensor node versions.

8

Lessons learned from the first deployment

The test bed for the first deployment was decided in conjunction with the ecologists and Telenor's LTE Cat-M1 coverage map. An area with good coverage was chosen and preparations were done. The units were transported by car to Vadsø, where a Murphy would meet up with some field researchers from COAT. From Vadsø there was an one hour drive to Nyborgmoen where the equipment was loaded over on ATVs. From there there was a three hour drive along ATV trails close to Reinhaugen where the units were going to be deployed.

Because of regulations in the area, no motorized transportation can happen outside the designated trails, and therefore the equipment needed to be transported on foot the remainder of the deployment. The test bed is approximately a 20 minutes hike from the trails, and the individual nodes are between 300 and 800 meters apart, spanning a couple of kilometers. The deployment itself takes approximately 10 hours.

This illustrates why it was crucial to the project that the sensor node was robust and well tested before deployment. Maintenance of a unit is a major undertaking, and we should have taken every step possible to ensure that the units would work once deployed.

8.1 Deployment procedure

The twelve camera traps all consist of a metal box with a wildlife camera mounted in the lid and the two compartments we are allowed to use. The main compartment where the camera is installed also have a hole cut into two of the walls, allowing it to be integrated into the rodents tunnels. The unit is placed in one of the compartments with the feet at the bottom and the CO_2 sensor at the top. The photo transistor is taped on the edge of the compartment, facing the camera (Figure 8.1). The LTE-antenna is routed outside the box and taped on top of the lid (Figure 8.2).

Figure 8.1: This picture show how the IR transistor is secured in the camera trap (Photo: Michael J. Murphy)

Figure 8.2: This picture show how the antenna is positioned on top of the camera trap lid (Photo: Michael J. Murphy)

Before deploying the nodes, the researcher initializes the self-test procedure of the units. This is to make sure that all the sensors were working correctly and that the unit can communicate. With an all clear, the box is placed as described above.

8.2 Deployment problems

During the deployment, three units had their SD-card dislodged and wedged between the lid and the box. Two were broken, whereas the third could be put back in place. This was not an anticipated problem and we had not issued any extra SD-cards. Because of this, one unit got deployed with the SD-card of the test unit, and one unit could not be deployed. The test unit was programmed to communicate every 10 minutes instead of once each day.

All other units completed the hardware self-test, but only three units completed the communication tests successfully. Five where able to attach to the network, but where not able to complete an NTP request, resulting in an infinite self-test loop. Two units could not attach to the network at all, and one unit was not tested in fear of the self-test loop. We had never seen the looping problem in our pre-deployment testing.

The researcher tested different solutions to break the self-test loop, amongst

others placing the antenna inside the camera trap. He was hoping that the metal box would act as a Faraday cage to prevent the units from attaching to the network. This did not work. Disconnecting the antenna was not an option as this would risk destroying the LTE-M modem[32]. The solution ended up being dislodging the SIM-card from the FiPy while it was in the test cycle and carefully re-seat it after it gave up on communication. This made the self-test fail instead of looping.

8.3 Communication problems

After deployment we expected some units not to communicate, as the self-test had shown that the coverage in the area was worse than the coverage map suggested. What we did not expect was that we got absolutely no communication at all. At August 26th, we got partial communication from one of the units, but this communication was interrupted by a timeout on our server. After this we received nothing from any of the units.

Norwegian telecom regulation is quite strict on getting meta data from the telecom companies, but we managed to get some logs from Telenor of the last contact each of the SIM-cards had had with a tower. According the the logs, all units where seen by the towers during deployment. After that only two units had been able to attach to the tower at all, one of which is the one that we could see in our servers log. The other had not been able to contact us.

8.4 Hypotheses

The communication problems had to be solved. In this section we will present the different hypotheses we had for the cause of the problem and discuss the perceived likelihood of each.

8.4.1 Communication back off

Built into the code there is a back-off algorithm designed to limit the communication drain on the batteries when the units are covered with snow and not able to communicate. We expected the units to, at least occasionally, fall out of communication as snow depths can reach several meters and we did not think the LTE signal would be able to penetrate this amount of snow. Water is known for its radio dampening properties, and meters of snow seemed to us unlikely to help the signal.

Because of this, for each failed attempt to reach a tower, the communication cycle will be extended by one day, up to a maximum of seven days between each communication attempt. This will assure that a minimum amount of battery capacity is wasted when not able to communicate, while giving us a reasonably short time until we regain communication once the snow has melted. Every successful communication attempt will reset the timer back to one day between each attempt.

Before the deployment, the units had been transported by car by one of the ecology researchers. During the car trip, the units spent a couple of days in a town without any Cat-M1 coverage. this would have triggered the back-off algorithm, but we should not reach the seven day cap in this stay alone. Because of a design flaw in the original program, the communication self-test did not reset the back-off counter.

With the back off at its maximum of seven days between communication attempts, we should still have been able to get some communication attempts from the units. The deployment tests showed that almost all units were able to attach to a tower, even if they were not able to connect to our servers. One explanation for this would be if the back-off algorithm somehow had malfunctioned and sent the units into an eternal back-off state.

We conducted an experiment where the communication delay was set to 2 minutes. The unit was also connected by USB and the output of the unit was monitored. Further, the SIM-card was removed, making any communication impossible. The device showed the expected behaviour of increasing the communication delay to 35 minutes. When the SIM-card was reintroduced, the back-off algorithm returned to 2 minutes, as intended. At that time, we considered this hypothesis implausible.

8.4.2 Weather conditions

We know that wireless communication does not fare well through water[12]. If the units were to be covered in snow, we expected this to severely impact the signal. We expected the units to lose communication at some point though the winter. We hoped that they would be able to get a signal through now and then. According to the weather forecast, no snow coverage was present in the area of deployment. Later we have learned that snow does not impede wireless signals nearly as much as we thought as water in a frozen state is not dielectric.

8.4.3 Software or firmware problems

Aside from the back-off algorithm, other software problems could be the cause of our lack of communication. Multiple researchers in our research group went through the code, and even though some minor potential problems where found, and corrected, none of them would give us the issues we experienced in our deployment.

The way the firmware works on the micro controller we selected, any uncaught exception would put the controller into REPL mode. In this mode the developer have an interactive python shell allowing debugging the device immediately. This is handy while developing, but this could potentially cause a severe problem in the field. It is therefore important to be sure to catch all exceptions.

The way this is normally done in these controllers is that there is one global exception handler that will handle any exceptions generated by the user code, and all system calls done by the user code. If there is any particular risky code that needs to be run, or code where the exception needs to be handled in a specific way, this code is wrapped in a local exception handler. Everything else will be handled by a simple handler that will just set all parameters to a known state and ether restart the device, or put it to sleep.

Our code does have this, and if any exception that was not specifically handled occurred, the device would go into recovery mode and go to sleep until the next measurement cycle.

We then discussed whether the problem could be caused by an exception triggered by the boot code in the firmware. This would not have been caught by our exception manager. This could be caused by a corrupted or wrong register or fuse setting. As there are no way of setting the fuse settings of the ESP32 directly from µPython, this would have to be caused by a bug in the firmware.

Any uncaught exceptions would put the device into REPL mode, with a power consumption of around 80mA. The device would in this mode run out of battery within hours instead of months. Furthermore it would not run any of the user code, and thereby not trying to communicate with us. This was still one of the hypotheses we kept until the real problem was eventually found.

8.4.4 Hardware problems

Since none of the nodes had communicated since deployment, we considered potential hardware failure. In particular, the possibility of excessive power consumption was discussed.

The hardware was carefully designed to use as little power as possible while sleeping, turning of all components except the micro controller. As all the circuit boards were handmade, some failures are to be expected. If the deployment had consisted of only a few units, hardware problems would be on the top of the list of likely causes of failures. This is one of the reasons we planned on deploying 12 unites at the same time, mitigating that risk. It is unlikely that mistakes would make all the units malfunction if there is not a problem with the design itself.

Any fault that would make all the units malfunction should therefore be replicable in the lab using the same hardware design. All our testing in the lab showed the expected power consumption; both with a unit with accelerated cycles of 5 minutes, and a stock unit. Hardware problems where therefore deemed unlikely as the cause of this specific problem.

8.4.5 Signal problems

In our correspondence with Telenor, we also asked about any irregular coverage problems on their end. We were assured by Telenor that they had not experienced any major and lasting problems with the towers in the area reachable from our deployment site.

We conducted some experiments with signal strength on our units in the lab. We found that the positioning of the antenna on top of the box severely impeded the signal. Signal strength when the antenna was in open air, away from the camera trap, averaged around −53dBm. With the antenna taped on top of the metal lid of the camera trap, the signal was approximately −75dBm. Thus we were losing 99% of the signal. We still had no problems connecting and sending data from our office. −75dBm is considered a reasonably good signal for a Cat-M1 device. All pre-deployment testing had been done in our office, which has a very good network coverage. Few places close to our lab have a signal strength as low as we expect it to be on the test site. Thus, testing in close to equal conditions is hard to achieve.

When deploying the units, they were all running the self-test routine before they were placed in the camera trap, and before the antenna was taped to the lid. This explains why nearly all nodes could establish a link to the tower in

the self-test, but not after they were deployed.

8.5 Investigation

Antenna issues appeared to be the most likely cause of the connectivity problems. In this section, we will attempt to explain the cause of the reduced signal and find a means to combat it.

8.5.1 Capacitor effect

After some research, we realized that the antenna configuration we used could be approximated as two metal plates separated by some space. This is the basic principle of a capacitor. Sufficiently large capacitors coupled with the correct resistor can be considered a short circuit for alternating current signals (AC). There is always a resistance element in a signal cable. The internal resistance from the modem to the antenna combined with the capacitor we have made with the antenna results in an RC circuit. This is also called a low-pass filter.

A low-pass filter is used in signal processing to filter out signals above a certain frequency. To calculate the range of the filter, we first need to know the capacitance made by our capacitor and the resistance from the modem wave generator.

We did not have the necessary equipment to precisely measure the capacitance, but this can be calculated:

$$C = \frac{k \cdot \varepsilon_0 \cdot A}{d}$$

Where C is capacitance, k is the permittivity of the material between the plates, A is the area of the metal plate in m^2 and d is the distance between the two metal plates in m. The metal plate in the antenna has an irregular shape (see Figure 8.3). This makes it hard to precisely measure the area. We estimated it to be about half of the antenna plate.

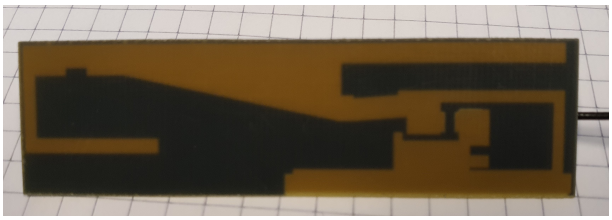

Figure 8.3: Picture of the underside of the antenna exposes where the antenna is covered with conductive material (brown) and where it is left bare (black) (Photo: Øystein Tveito)

$$A = \frac{75mm \cdot 21mm}{2} = 0.0007875m^2$$

.

The distance between the antenna and the lid equals the thickness of the substrate of the antenna. This was measured to $0.3mm = 0.0003m$. We assume that the antenna substrate is made from woven fiberglass. According to the FR-4 NEMA standard[13], woven fiberglass shall have a permittivity of $k = 4.4$. In the following, we assume that the material adheres to the FR-4 NEMA standard.

In spite the uncertainties of these parameters, the calculation will provide an acceptable "order-of-magnitude" estimate of the capacitance of the system.

This gives us:

$$C = \frac{4.4 \cdot \varepsilon_0 \cdot 0.0007875m^2}{0.0003m} = 0.00010226F = 102.26pF$$

We were not able to measure the resistance between the wave generator and the antenna. The wave generator is inside the modem, and thus inaccessible. The antenna has a long, relatively thin cable and a small connector, both introducing some resistance. For this exercise we will assume that the resistance is 1Ω.

These values where entered into Altium Designer™that calculates the cutoff frequency and transient response automatically. The cutoff frequency, when the magnitude is halved, was calculated to be 1.55GHz. There are different frequency bands available for LTE Cat M1. In the upper end of the spectrum, band 41, has a frequency of 2.593GHz.

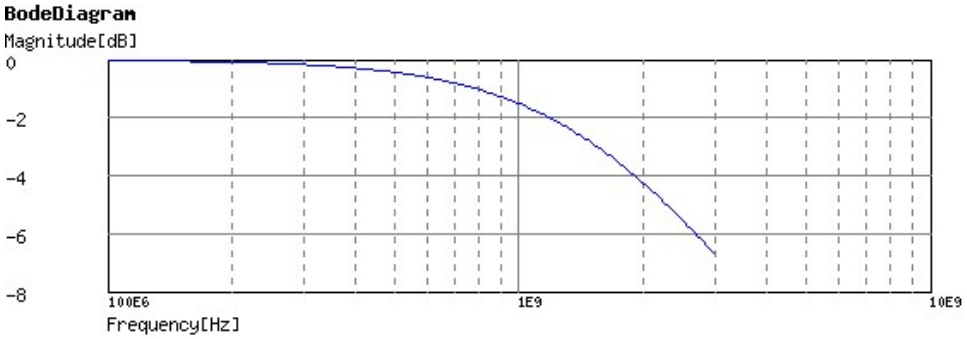

Figure 8.4: Frequency response of RC filter (Graph: Øystein Tveito)

It is plausible that this band is used for the stations we are connecting to. With this frequency, we would get approximately −6dB signal strengthFigure 8.4, reducing our signal with 75%.

8.5.2 Non-linear-electromagnetic effect

When the frequency is high enough relative to the antenna size, the problem becomes more complicated. With high frequencies there will be a charge difference between the near- and far-side of the antenna. Thus, the calculations above are not as straight forward as they are with low frequencies.

In RC filter calculations, we assume that the entire metal plate of the capacitor holds the same charge. As a non-linear electromagnetic problem, this assumption no longer holds true. The calculations of how much this would influence the signal is more complex. It must take into account the exact shape of the antenna, which is a multi band antenna, and therefore quite complex. This would be out of scope of this thesis, and more suiting for an antenna theory study.

8.5.3 Antenna attenuation

The third factor that could affect the signal quality and strength in our case would be attenuation problems with the antenna. As described in the two prior subsections, the linear and non-linear capacitor effect will weaken the signal, but they will also attune the antenna.

In a simple antenna the length of the exposed conduction directly affects the harmonic frequencies the antenna will respond to. This antenna is a multi band antenna, meaning it has multiple harmonic frequencies, but the basic principle

still applies. The shape of the antenna is precisely calculated to allow certain frequencies to resonate in the antenna.

The capacitor effect will, as previously mentioned, to some degree work as short circuit for higher frequencies, and as a resistor for lower. In any case, the fine tuned impedance of the antenna will be changed. This could drastically impede the signal.

8.6 Countermeasures

The low-pass filter effect, antenna attenuation, and the non-linear electromagnetic effect combined can explain our problem in the first deployment. The next step was to figure out how to fix it.

As we can not change the resistance of the antenna, the only parameter causing the low-pass filter effect that can be easily changed is the capacitor effect. Reducing the capacitance of the antenna will reduce the unwanted non-linear-electromagnetic effect. It will also solve the attenuation problem.

8.6.1 Countermeasure testing

As the conclusion was that the proximity to the lid was the cause of the problems, the solution seems obvious; move the antenna away from the lid. The next thing to determine is by how much, or if we could find another solution that would give the same result.

To find the ideal antenna position, several tests where done with different antenna orientations and distances. The fist was to test distance. We tested distances of 1mm to 10mm in 1mm increments, and 5mm to 100mm in 5mm increments.

For the $1 - 10$mm tests we made shims of a specific thickness that would fit underneath the antenna when taped down. We took a few readings of each distance before replacing the shim with the next one. As the changing between the different thicknesses was somewhat elaborate, 3 successive measurements were done on each thickness before moving on. This could potentially make temporal variations in signal strength influence our results, but reduced the time required to complete the tests. This compromise was shown to be acceptable as no temporal variations in signal strength were found in subsequent experiments.

For the 10 – 100mm tests we created a tower with slots of 5mm increments. This allowed us to quickly transition between elevations. The tests on this scale where done with one measurement for each elevation. The entire test was repeated several times, to reduce potential effects from temporal variations in the signal. The results showed no significant temporal variations.

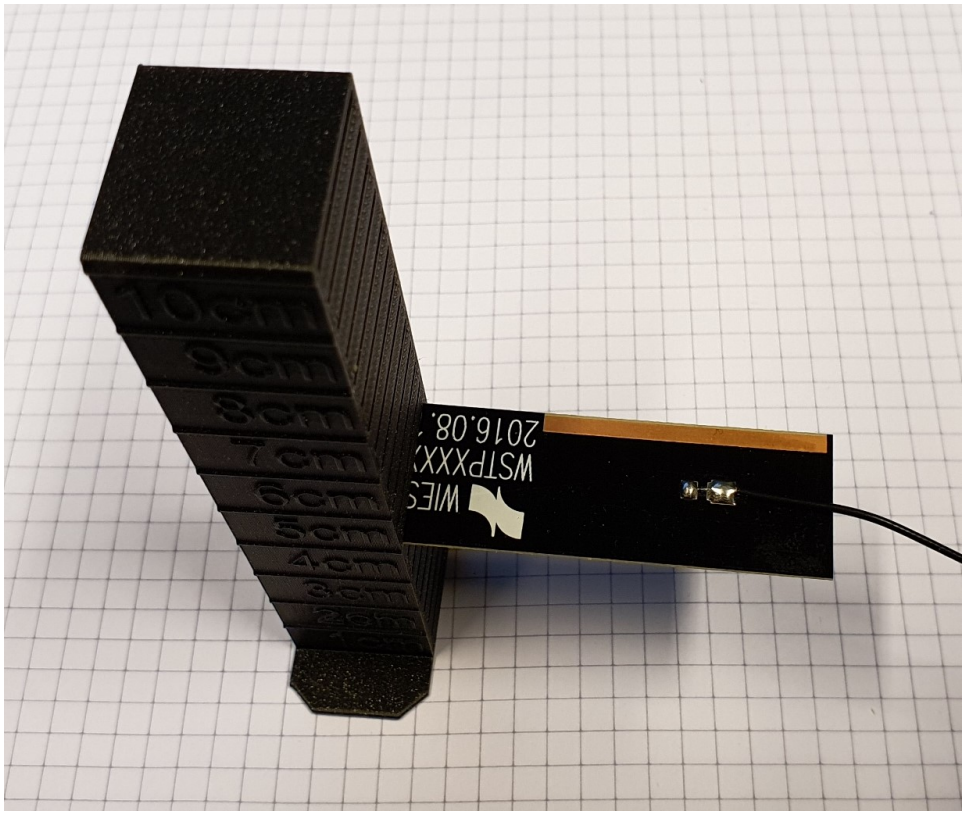

Figure 8.5: The tower used to test 10 – 100mm (Photo: Øystein Tveito)

In addition we tested positioning the antenna in a non-planar position on top of the camera trap; at 30, 60, and 90 degrees. Tests done with the antenna in this fixture was done in a few different horizontal directions as well to see if the facing direction would influence the signal. From the testing we wanted to find the ideal position of the antenna and the minimum viable signal strength.

We did the tests described above in three different locations. The first test took place in our office As previously mentioned our office has excellent coverage.

To find a location with coverage closer to that of the deployment site we

measured the signal strength at several locations in our building. The lowest we could find was in a culvert connecting the building to other buildings on campus.

After these tests where completed we repeated the tests at a location in Futrikelv, at the outskirts of Tromsø. One of the supervisors, professor Otto Anshus, had personally experienced that the coverage from Telenor was lacking in that area. This would give us even better opportunity to test in conditions similar to that of the deployment site.

8.6.2 Countermeasure results

From the testing we found that moving from 1mm to 10mm we had a huge increase in signal strength, and were now comparable to the tests done without the camera trap there. From 10mm to 100mm there were no measurable improvements. Tilting the antenna had no significant impact on the signal strength. The results are shown in Figure 8.6 and Figure 8.7.

From the results we have also investigated what signal strength is the required for a successful communication cycle. This is shown in Figure 8.8. Here we can see that a signal strength between −80dBm and −90dBm is necessary for communication to be successful, with one outlier of −95dBm.

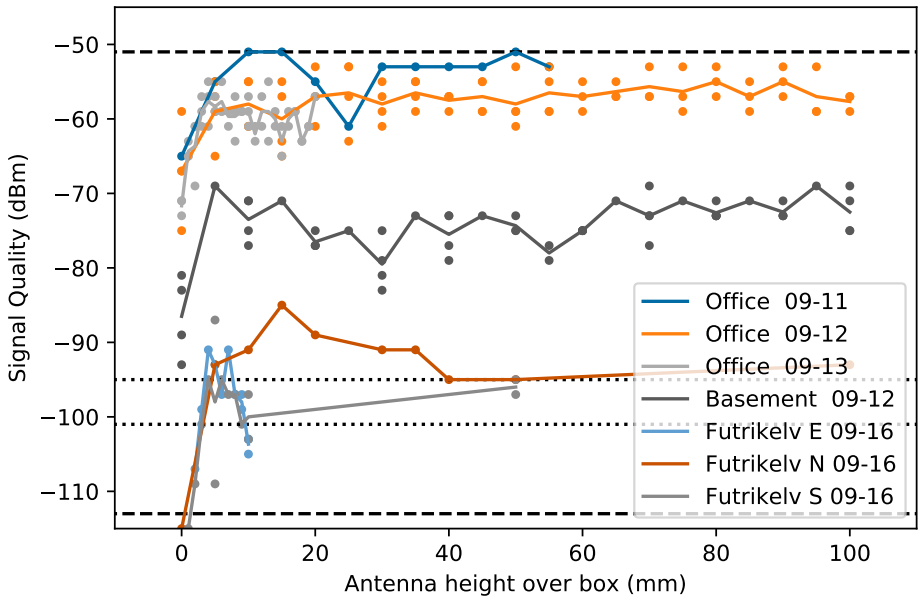

Figure 8.6: Graph of signal strength from multiple testing runs in multiple locations from 1 – 100mm (Graph: Michael J. Murphy)

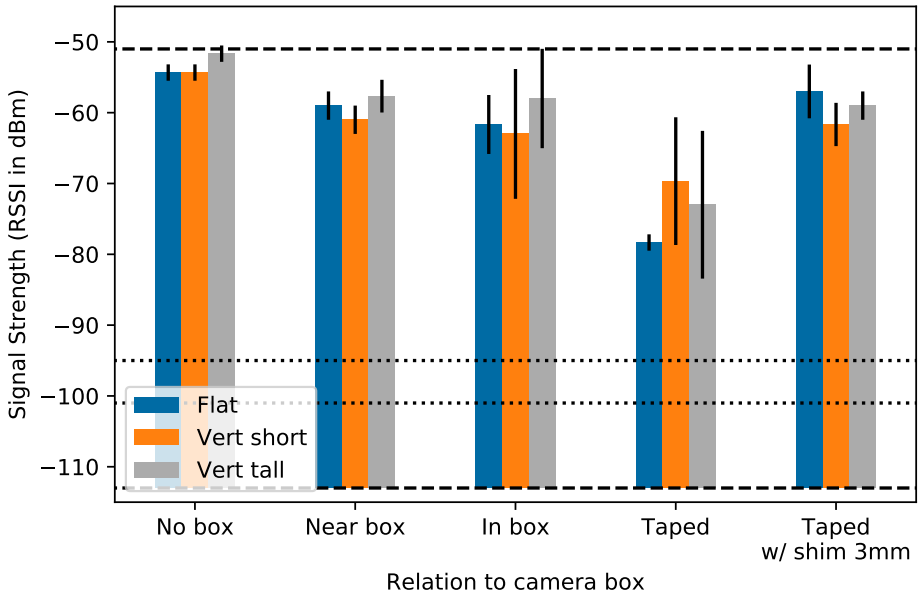

Figure 8.7: Bar graph showing the signal strength of different antenna setups (Graph: Michael J. Murphy)

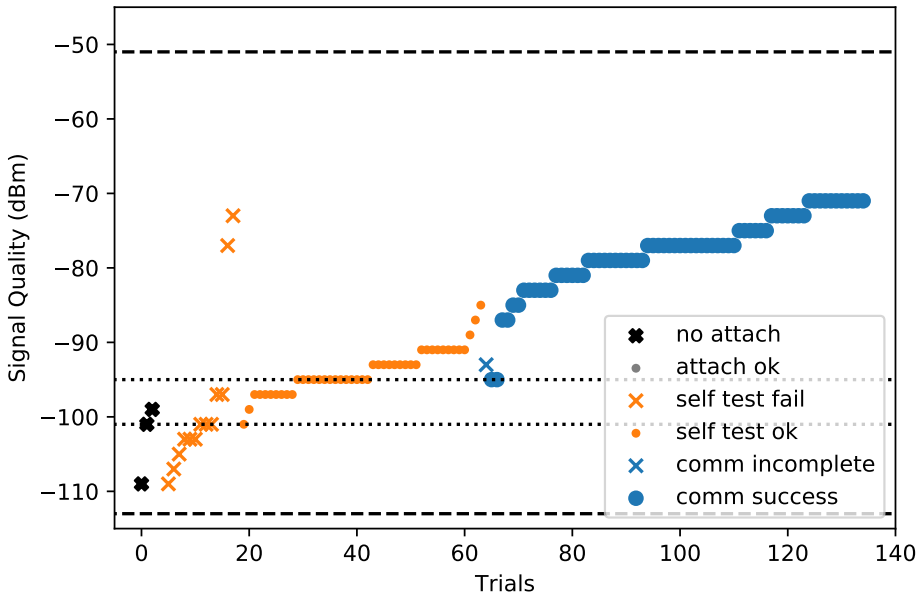

Figure 8.8: Graph showing signal strength and communication status (Graph: Michael J. Murphy)

9

Second deployment - Modifications and experiences

As the snow had not yet covered the deployment site, we had the opportunity to do a second deployment where we could add the countermeasure to the antenna. At the same time we applied some minor software changes.

9.1 Changes to the device

We took the opportunity to add a few changes, both to the physical implementation and the software running on the device.

9.1.1 Software changes

During the investigation following the first deployment a few minor problems where found with the code. As mentioned earlier, the self-test was not able to run on most of the units without resulting in an infinite loop. This was corrected by introducing two changes to the self-test code. Firstly, the node uses a fixed IP for the NTP server. Secondly, if no connection could be achieved

on the first try, the test would terminate.

Another problem pointed out in the first code review was that the watch dog timer of 10 seconds was a bit short. The watch dog timer is not meant to trigger in normal conditions. It is there to avoid an infinite waiting state that may occur if the code encounters a blocking call that does not return. We found that some UDP requests could take up to 10 seconds before the socket would time out. The timer was thus increased to 60 seconds.

Measurements of the signal strength was added to the communication cycle and reported along with the sensor measurements. If we had this from the beginning, we assume that the antenna problem would have been discovered much earlier.

9.1.2 Physical changes

We had previously had problems with the SD-cards breaking. This was caused by the SD-card partially sticking out of the main board of the sensor node, which was not securely fastened in the box. The SD-card could then be pressed in between the lid and the box, and it would break off by a small jolt to the box. To solve this, some spacers where designed and added to the side of the box ensuring that the main board could no longer move. This solved the problem.

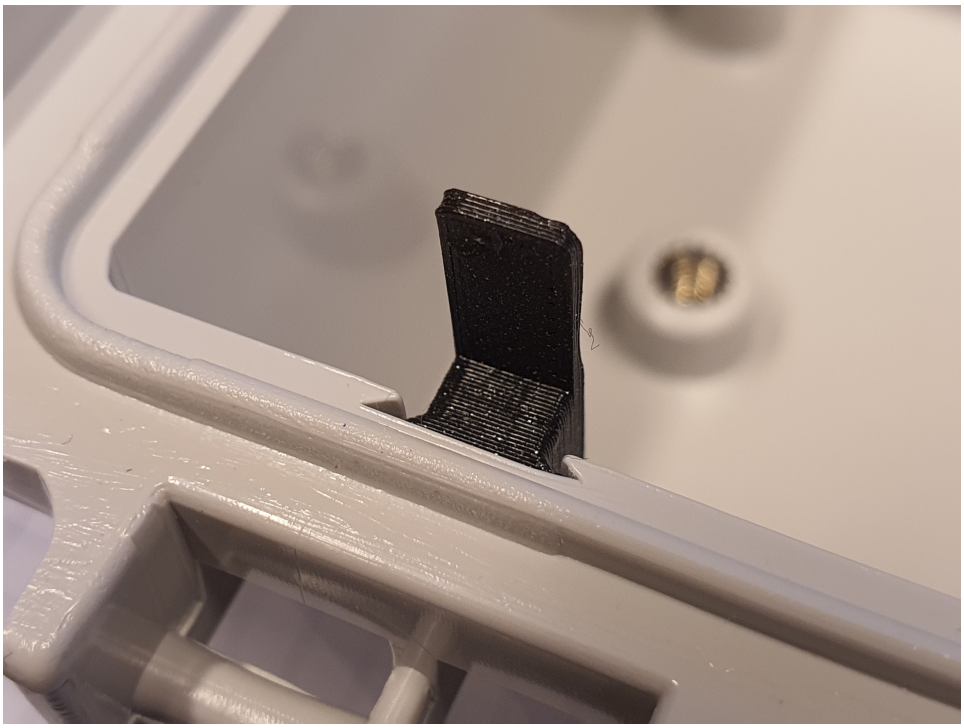

Figure 9.1: Photo of the spacer slotted into the sensor node enclosure (Photo: Øystein Tveito)

9.1.3 Antenna changes

With the knowledge acquired in the testing of the antenna, we decided that a shim of 10mm would mitigate the problem. We also wanted to give the antenna some added protection. Previously we relied on a thin coat of lacquer on top of the antenna and the tape to protect it from the elements. We designed the shim for the antenna as a box with a lid that would provide the added protection.

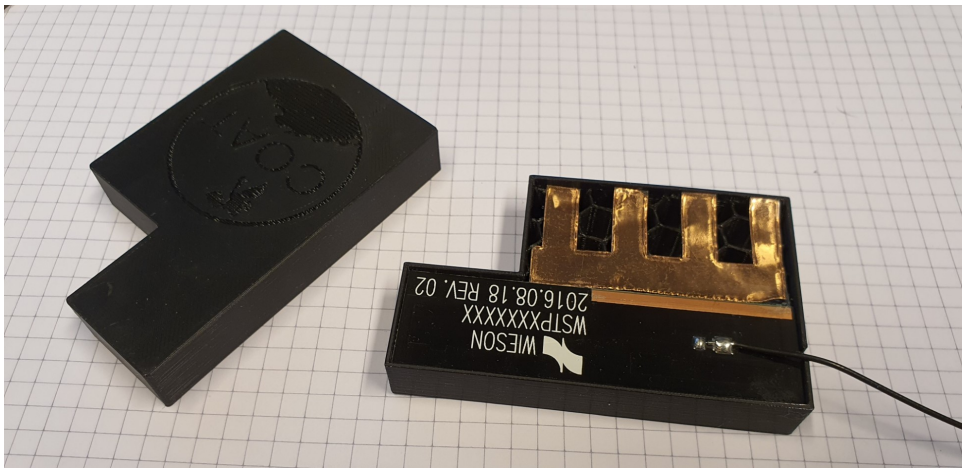

Figure 9.2: Photo of the antenna in the protected shim (Photo: Øystein Tveito)

9.2 Second deployment procedure

Both the physical and software changes had to be applied to the already deployed units in the field. We analyzed the risks and benefits of three different approaches to this.

- Load new software from an SD-card
- Reprogram the nodes in the field
- Replace the micro controllers with new, pre-programmed controllers

The first software version has a self-update procedure built in. This allows us to load the new software to an SD-card and the controller loads it from there. The problem with this approach is that this procedure is started right after the self test. As we learned from the first deployment, described in Section 8.2, this could be complicated. We could have gone through the same procedure, as we did in the first deployment, by removing the SIM card and reinserting it at the correct time. This, however, is time consuming and would force us to spend time with the nodes exposed to the weather.

Alternatively, the nodes could be reprogrammed in the field. As the micro controller is programmed through a serial interface, it is possible to create a program that would allow us to quickly reprogram the controllers. A prototype of this was created but later abandoned. Programming in the field would, as with the SD-card solution, force us to expose the electronics of the node to the

elements for longer than we preferred.

We decided that the best solution was to bring new micro controllers already programmed with the revised software. This allows us to quickly update the node from version 1.0 to 1.1 without exposing the electronics unnecessarily to the weather.

The deployment procedure now consists of:

- Open the box
- Remove the micro controller and SD-card
- Insert the new micro controller and SD-card
- Add internal spacers
- Close box and position it in the camera trap
- Close lid and apply shim to the antenna

This procedure was expected to take less than five minutes in the field.

9.3 Second deployment

The second trip to the deployment site was without the ecologists. As the task was only to change the nodes we had deployed, a two man team from DAO went to deploy the changes. A liaison from COAT followed us to the deployment site as he had to install some upgrades to a weather station they have in the area.

Right from the first node we had some unexpected problems. Some of the nodes were already out of batteries. We had brought with us four spare units and a twelve extra batteries. We expected the node with communication intervals of 10 minutes to be out of batteries, but we did not expect the other nodes to be dead.

During the deployment, seven nodes where out of batteries, four still had power. Thus we had to strategically replace batteries and nodes. We decided that we wanted the prioritize the nodes in the higher ground, as those were the once with the best chance of communicating. Therefore, two camera traps were left without nodes, and two nodes were given half the amount of batteries they

were originally planned to have.

We ended up with ten populated camera traps, where two of them were expected to die during the winter.

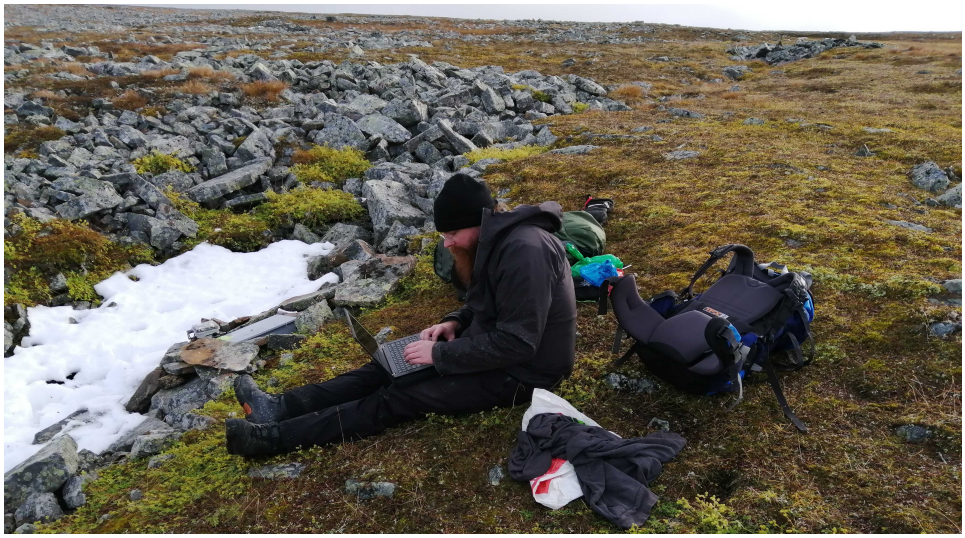

Figure 9.3: Michael J. Murphy inspecting the logs of an inoperable sensor node at the deployment site (Photo: Øystein Tveito)

9.4 Results

From a total of ten working units, we have since heard from six. We had long since given up the hope of getting report from all of them, as the coverage in the area is worse than we expected. From our experience with the first deployment, having heard from six of them was a relief. Some of the nodes have been heard from nearly every day, some only reports occasionally (see Figure 9.4).

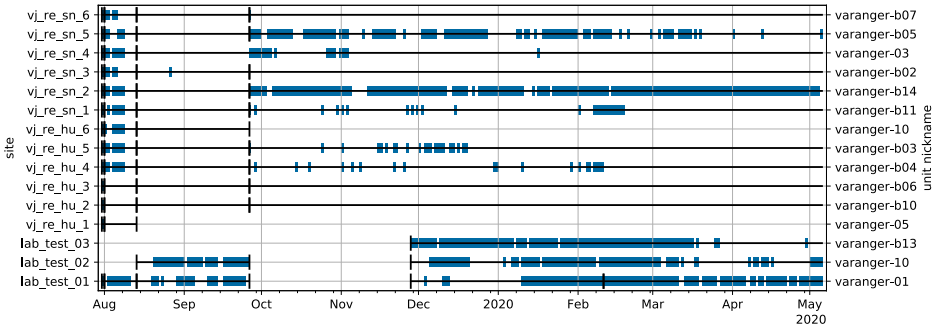

Figure 9.4: Graph showing communication over time. First vertical line is first deployment. Second vertical line is second deployment. (Graph: Michael J. Murphy)

Only hearing from some of the nodes is of course not ideal, but on the other hand is arguably more interesting from a research point of view as we will discuss later in this thesis. The fact that we at this point had working nodes that at varying degrees reported back regularly gives us the opportunity to start analyzing our experiences.

10

Project analysis

The final analysis will need to be done after the deployment, but we can still analyze what we have learned so far. This project is a large project for two people, executed in a short time. Multiple mistakes have been made, and multiple lessons have been learned from them. Here we will try to analyze the key elements of what has worked and what has not.

Analyzing what we have learned from our existing experiment will later be broken off into three major components:

- Experience on how to create the hardware that can withstand the harsh climate of the arctic tundra.
- Experience on how to create software that will work and not crash, even if some of the hardware breaks.
- Experience working on a project from beginning to end in order to ensure that the end-product works as intended.

10.1 Status this far

From the ten sensors we currently have in the field, we have been in communication with six. Two are regularly sending data, and the rest are only able to

contact us infrequently. While this might not be the result we were hoping for, it might be more interesting from a scientific point of view. This way the units are tested on how they handle temporary communication loss.

10.1.1 CO_2

The data we are receiving from the nodes is unstable. The CO_2 values are jumping between zero and several thousands, and in one case several hundred thousand ppm. We can therefore confidently say that these measurements are incorrect. The interesting thing we can see from the reading we are getting is that at least on some charts we can see an inverse correlation between the CO_2 value and temperature. The second interesting thing we can see is that on almost all the nodes, where we are getting values from the CO_2 sensor, we see a sudden stop in the temperature fluctuation. This is most likely the point where the camera trap became completely covered in snow and therefore the temperature becomes stable and so do the CO_2 values.

It is therefore reasonable to believe that the sensor is temperature dependent to a greater extent than we thought it would be from reading the data sheet. Another reasonable assumption is that the sensor is sensitive to humidity. This was not accounted for in our hardware design, and therefore we did not add humidity sensing capabilities to our node. We know that when the temperature goes down, the ability of air to absorb water is reduced, and therefore the relative humidity increases. When we see the inverse correlation between temperature and CO_2 value, it could be just as likely that what we actually see is a correlation between humidity and CO_2 readings. It is also a possibility that condensation or deposition of water from the air affects the measurements.

When reading the data sheet[15] it appeared that the sensor we chose, the explorir-w-20, was the wrong one for the job. This sensor is not calibrated below $0°C$. The sensor we should have chosen was the explorir-we-20 which is in essence the exact same sensor, but have been calibrated all the way down to $-25°C$. The calibration can be done by the manufacturer if we send the sensors back to them, and they should then work in the extended temperature range. The reason they are not all calibrated for the extended temperature range is that the calibration itself is a time consuming and expensive process. This caveat was not clear to us when we read the data sheet before deciding on this specific sensor for our design.

On two of the sensor nodes, the CO_2 value is constantly 0. This is an indication that the sensor itself is malfunctioning. This is probably because of ice depositions on the sensor itself, or it could be a randomly failing sensor. We will not

get a definite answer until the nodes are recovered from the tundra.

10.1.2 Temperature

The temperature readings we are receiving seem reasonable, and when compared to the historic weather data from weather stations in the area, they are not far off. However, we have decided to replace this sensor, as we have realized that we also needed to record humidity.

10.1.3 Camera flash detection

The camera flash detection seems to be working as intended. All nodes from which we have received data have indicated that the cameras are in working order.

10.1.4 Logs

The node logs that we have received show that the nodes are mostly operating as intended. The watchdog timer rarely triggers and not many other errors are recorded. In hindsight we should have logged more extensively. This would have helped us in determining the cause of failure for the retrieved nodes. It could also possibly help us understand what causes some nodes to record obviously erroneous sensor data.

10.1.5 Unknowns

There are still a lot we do not know about how the nodes are operating. From the nodes we have in the field, only six have been in communication since deployment. We expect this to be because of the network coverage in the area, but it could also be that the nodes have failed somehow. This will not be answered until we either achieve communication or until we recover the nodes next autumn. Our hope is that they are operating, but not able to communicate. All the sensor data and logs should then have been recorded to the SD-card so that we can analyze it.

10.2 Hardware

The inaccessibility, harsh climate, and the lack of network coverage of the arctic tundra present challenges normally not faced in IoT development. The temperatures routinely gets below 20 degrees Celsius. The devices could be buried under meters of snow. If something breaks, the devices are inaccessible half of the year, both because of the snow and that we can not disturb the active experiment that the ecologists are conducting. The network coverage is limited. Due to the low signal strength, both weather conditions and local geography can severely affect communication capabilities.

From what we have experienced so far, the changing temperatures and humidity seems to be one of our biggest hurdles. Before the snow covers the camera traps, they are exposed to fresh air. The temperature difference between day and night during the autumn can be large, and we are hypothesizing that some of the CO_2 sensors have been damaged by having ice depositions on the sensor head.

When the snow covers the camera trap we see that the temperature becomes stable. Before this, the rapid change in temperature after the sun goes down can cause the humidity in the air to condense or deposit on exposed surfaces. This could cause problems for the sensors, and may be the cause of some of the erroneous sensor data we have received.

During testing in our lab in parallel with the deployment, we have also found that the SD-cards we are using are prone to corruption. This was a common occurrence during testing in a freezer we have in the lab which holds $-35°C$. This is outside the stated working range of the SD-cards, and we have not seen these temperatures in the field. Nevertheless, we suspect that the rapidly changing temperatures might influence the cards, but further testing will be needed to determine the severity of this effect.

Until the existing nodes are brought back to the lab we can not determine if the enclosures of the devices have provided sufficient protection from the climate of the tundra. The only issue we have encountered during our testing of the sensor node housing was that the SD-card sometimes got wedged in the lid. Aside from this, it seemed to provide sufficient protection from our testing conditions.

One shortcoming of this hardware platform is the lack of humidity sensing. This has proven to be a major problem when trying to analyze the incoming CO_2 data. The addition of a humidity sensor could possibly have improved the data from this deployment. Due to the possibility of a humidity dependence in the CO_2 sensor, we are unable to determine the validity of our measurements.

Another shortcoming of the platform is that the CO_2 sensor chosen is not calibrated for our conditions. The calibration is stored as a table of calibrated values on the internal storage of the sensor. Intermediate values are interpolated from this table. The sensor will still attempt interpolation when measuring values outside it's calibrated range. However, the calibration table is filled with junk values prior to calibration, leading to meaningless results. When the temperature is on or above freezing we have higher hopes for the data as this is within the calibrated range.

The sensor nodes do not appear to consume much more power than anticipated. None of the units we have for testing in the lab have been shown to use more than expected power. The nodes in the field have been running for several months and some of the nodes are still in active communication. The power supply therefore seems to be sufficient.

We did encounter some units that had flat batteries while doing the second deployment round. The working theory of this is that the SD-card have been corrupted at some point, possibly by a watchdog timer, while trying to write to the SD-card. In this scenario, the unit has no routine to handle this. A corrupted SD-card will result in an exception, which in turn will be written to the log, located in the SD-card, generating another exception. This loop will not be broken until the unit is out of battery.

All the prototypes were hand built from multiple components. All the connections between them were soldered by hand. This introduces many potential flaws in the design by simple human error. Even with a relatively simple design without too many components, a lot of soldering was needed. Each soldered connection adds the potential for mistakes. Even though each node was tested initially, weak soldering joints can cause issues later.

10.3 Software

Creating a robust observation unit requires both the hardware and software to work together and should continue to function even if non-essential components malfunction. If a single sensors stops working, the observation unit should be able to continue operation without it. Interacting with prototype hardware presents challenges that software normally does not have to deal with. In a normal program, once the proper tests have been done, one can be reasonably confident that the program will work. Bugs in the code are always present, but a software developer usually does not have to account for bugs in hardware. It is also usually assumed that hardware components do not suddenly break or start behaving erratically. When working with prototype hardware, these

challenges become prominent.

10.3.1 Watchdog timer

On micro controllers, watchdog timers are a widely adopted way of ensuring that the program running never gets stuck in unbreakable loops or blocking calls that do not return. The watchdog timer is implemented on a very low level, sometimes in hardware, and will overrule all user code running on the device. On the ESP32 the watchdog timer is implemented in the FreeRTOS operating system, and based on a hardware timer and dedicated registers. The length of the watchdog timer is important, as the watchdog should only trigger when the code is actually stuck and not when doing tasks that could take a long time to complete.

The first revision of the program had the watchdog timer set to 10 seconds. This proved to be too short for our purpose, as several calls related to network communication could easily take more than that. Having to feed the timer so frequently also made the code more complex. In the second revision of the code it was set to 60 seconds. As the watchdog timer is only meant to be a last resort, this is a more appropriate value. Furthermore, the self-test no longer goes into a loop if the communication fails between attachment and the completion of a successful communication cycle.

10.3.2 CO_2 readings

Another issue is whether there could have been done anything in software to remedy the erroneous CO_2 values. The current program is only extracting the "calibrated" values from the sensor, even outside the calibrated range. The raw values are the output of the light sensor internal in the senor head. In retrospect, we should have modified the code in V1.1 to also store the raw sensor values. By having these, we could potentially give an estimate of the CO_2 values also outside of the calibrated range. It could also have been valuable if we could get the sensors calibrated after the retrieval and use the new calibration to interpret the raw values.

We assume that humidity have a huge impact on the CO_2 values from the sensor. The lack of humidity readings is therefore a problem. The inverse correlation between relative humidity and temperature is well known. It could potentially be modeled what the humidity, both relative and absolute, in the camera trap was at each reading based on historic and current temperature readings.

10.3.3 Operation

The operation of the unit is quite straight forward. It is supposed to wake up at regular intervals, take readings and go back to sleep. In addition to this, it should wake up at another interval and communicate the findings over the network. The resulting software ended up being quite complex and in the end made debugging harder than it needed to be. A slimmer and simpler version of the program would have made debugging, both before and in between the deployments easier.

10.3.4 Conclusion

Overall, the preliminary results from the changes to the 1.1 version of the code seems promising. We will again not get a final answer until we retrieve the units in the field.

11

Final version

The final version of the sensor node is built based on the experience we had with the previous deployments. It is built on the same architecture as the previous versions, but with some refinements where needed. The design and implementation of 2.0 version of the node are described in Chapter 5 and Chapter 6. This chapter will explain some design choices made in the final version of the CO_2 sensor node.

11.1 Hardware

As discussed previously, the hardware is mostly the same for this version of the node as with earlier versions. The results reported back from the nodes in the field are looking promising, but some aspects of the previous versions needed to be improved.

11.1.1 Main board

The biggest hurdle when creating the original node hardware was the assembly time. Each unit required about two hours of soldering. This time does not include the time to remedy any eventual mistakes. The most cumbersome task when creating a main board was to connect all the components together. This was done by hand-soldering all the signal and power paths, either by solder

trace or by wire.

Version 2.0 has a PCB that connects all the parts together. This reduces the time spent manufacturing the main board from hours to minutes. This also reduces the risk of human error as there is less manual work involved.

Problems

There has been one production run of the PCBs. From testing, we have found that the board works as intended except for two small flaws.

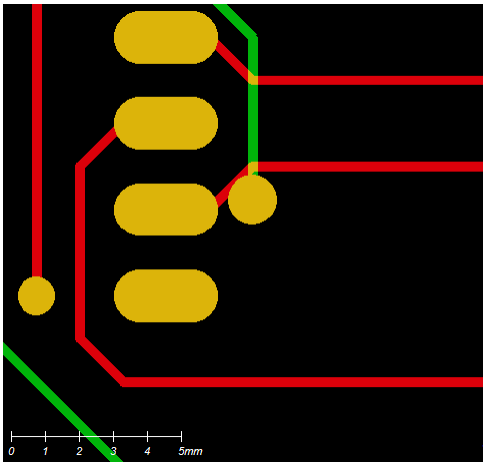

Figure 11.1: Illustration showing a portion of the PCB where a through hole connection unintentionally shorts two signals. Red = bottom layer. Green = top layer. Yellow = through holes, and copper on both layers. (Illustration: Øystein Tveito)

The clock line for the SPI bus (red) is touching the through hole connection (yellow) of the resistor going to the gate of the MOSFET (see Figure 11.1). These signals are then connected, and electricity can flow between them. The consequence of this is that the SPI bus is not working properly as the driver for the power pin outperforms the driver for the clock pin. This problem is remedied by cutting the trace going to the resistor and soldering the resistor directly to the pin on the backside. This is shown in Figure 11.2 and Figure 11.3.

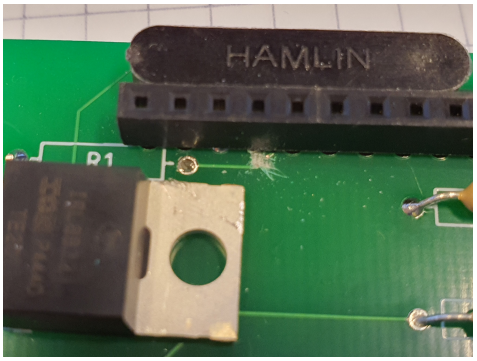

Figure 11.2: Front side of the PCB shows where the trace is cut (Photo: Øystein Tveito)

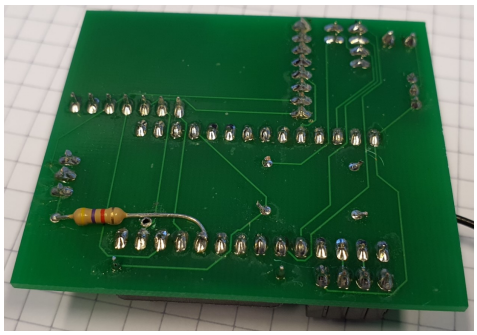

Figure 11.3: Back side of the PCB shows how the resistor is installed to bypass the intended hole (Photo: Øystein Tveito)

The second problem is that the board is configured for a Normally Closed (NC) reed switch, while the ones at hand are Normally Open (NO). The result of this is that the voltage level of the function pin is normally high, switching to a low state when magnetized. This voltage is meant to wake the sensor node when the operator wants it to perform a self-test. The node can still be programmed to be woken on a falling edge event. The problem with this is that the FiPy can only have one type of event for waking up from a low power mode. Since we already have the flash detector go high when an event is detected, we need the node to wake on rising edges.

This problem can be remedied in a few different ways. The first and simplest solution would not involve any modification to the nodes hardware or software. By attaching a magnet to the outside of the case the signal would go low. The operator can then remove the magnet to perform the self test routine.

Another fairly simple fix is to replace the reed switch with an NC variant. The board would then behave as intended. The NPN IR transistor used in the flash detector could also be switched out for a PNP type IR transistor. This would essentially reverse its functionality, making the sensor pin voltage go high when not detecting any event, and low when an event is detected.

The last solution would be to switch the order of the reed switch and resistor. The way we measure the position of the reed switch is by measuring the voltage between the switch and a resistor. In the current PCB, a voltage is applied to a resistor and the reed switch is connected to ground. The signal is measured between these. By applying the voltage to the reed switch instead, the signal would conform to the same pattern as the flash detector. Each of these fixes would successfully solve the problem.

Solution

As we will be ordering a new batch of PCBs, the best solution in our mind is to fix this problem on the PCB. The last version of the PCB have both the aforementioned problems solved as shown in Figure 11.4.

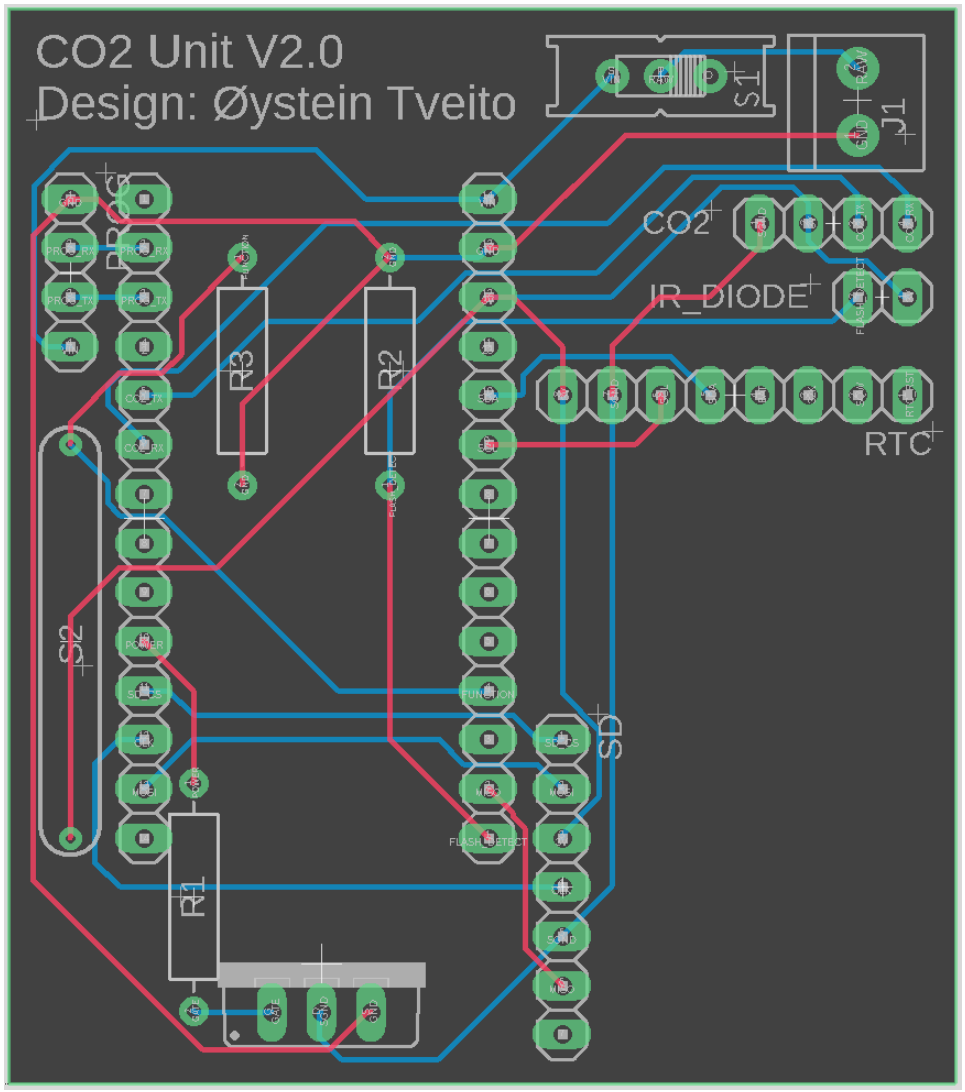

Figure 11.4: PCB layout showing the copper deposits on the PCB. red = bottom layer, blue = top layer, green = both layers, white = silkscreen (Layout: Øystein Tveito)

11.1.2 CO_2 and temperature sensor

The CO_2 sensor module has been replaced by one which is calibrated to work for a larger temperature range. The new module is a version with the same sensor, that has been calibrated to work from $-25°C$ to $55°C$. We have also opted for a version with a temperature and humidity sensor added to the module.

This simplifies the design slightly as we no longer need the dedicated temperature sensor. As we have mentioned earlier, humidity might play a role in our CO_2 measurements. The added humidity sensor will give us this data, allowing us to get more reliable CO_2 estimates.

Unfortunately, due to the the COVID-19 pandemic, these sensors have not been ordered. Fortunately, as the sensor employs the exact same sensor controller and firmware, testing of the new solution can be done with the version we already have. The protocol for communicating with the sensor is the same for both versions, and there are only a few parameters that must to be changed when switching over to the new sensor.

11.1.3 SD-card

We have replaced the SD-cards from that of the previous designs. The SD-cards of multiple units have been corrupted, both in the lab and from the recovered nodes after the first deployment. The original SD-card used conformed to the required specifications, and we are unsure exactly why they failed. We decided to go for industrial grade SD-cards instead of the commercial grade ones. The new cards are made with an older, more robust technology. The price is higher, but we believe that these cards should be less likely to fail.

11.1.4 Power supply

The smaller size of the main board of the node also allows for more batteries. Version 2.0 of the sensor node has six sets of three batteries, compared to four sets of three in the previous versions. Combined with the reduced energy consumption, the 50% increase in available power will give a bigger energy buffer for the node. It is also conceivable that the node could be deployed for longer periods. In the next chapter we will take a closer look at the power consumption of the different versions of the node.

11.2 Software

The new software addresses some of the issues we currently have on the deployed nodes.

11.2.1 Storage

As described in the hardware section and previously in this thesis, storage corruption has been a problem. This can be caused by faults in the physical card or by corruption of the file system. One of the improvements in the new software is that all data and logs are stored internally on the micro controller. The data and logs are then pushed to the external SD-card once a day. The internal storage is large enough for several days worth of data and logs. Every time a write operation is performed, there is a chance that the SD-card will be corrupted. If the controller is suddenly powered off while doing a write operation, there is a significant risk of corruption. By only writing the data to the SD-card once a day we reduce this risk.

Another benefit of using the internal storage is that accessing it is much faster. Reducing the time to record measurements will slightly improve the power consumption for every measurement cycle.

SD-cards can only be accessed in blocks. When writing a single byte to an SD-card, the entire block needs to be read into a buffer. The byte that is to be changed is then changed in the buffer. Finally, the entire block is written back to the SD-card. The block size of our SD-cards is 512 bytes. Because of this mechanic, writing 1 or 512 bytes will take exactly the same amount of time and requires just as many operations. This solution therefore reduces the stress on the SD-card by bulking the write operations together and writing more data at a time.

Another significant difference with having logs and data stored internally is that the node can operate without an SD-card. Compared to the SD-card, the internal storage of a few MB is limited. The node will not be able to store an entire years worth of data and logs. If the node is in semi-frequent contact with the network, the logs and data will be uploaded to the servers and then deleted from the node. In this way, as long as communication is satisfactory, the node will be able to operate normally, even if the SD-card is broken (see Figure 5.4.

11.2.2 Power consumption

A flaw in the previous software versions have led to slightly higher power consumption than necessary. The way the node powers off the sensors when it is in low power mode is by breaking the sensors' connection to ground. All sensors therefore have the voltage input high, but nowhere for the electricity to go to ground.

The problem with the previous software versions is that even though the ground connections are severed, the data pins are not addressed. When a voltage is applied to the sensor, the entire sensor will eventually reach this voltage. When the micro controller is put in low power mode, the signal pins of the controller will go to low voltage, or ground. The electricity going into the sensors can then leak through the signal pins. This is accomplished by forcing all the signal pins to remain high during low power mode. With the signal pins at the same voltage as the sensor, there is no path for the electricity from the sensor to ground.

Another decision that influences the power consumption of the sensor node is how the communication cycle is scheduled. In version 1.0 and 1.1 the communication cycle is scheduled as a separate event, with its own wake up. Every time the system powers up from low power mode, some time is spent on setting up the underlying systems like FreeRTOS and the µPython interpreter. Version 2.0 combines the communication cycle with a normal measurement cycle, essentially reducing the number of wake ups by one. This might not influence the overall power consumption by much, but small savings build up over a year-long deployment.

Version 1.0 implemented a back-off algorithm that reduces the communication attempts for nodes with poor coverage. If a node is totally without coverage, the node will, after some time, only try to communicate every seven days. This will reduce the power consumption of the node as communication is one of the more costly operations done by the node.

In version 1.1 and 2.0 we chose not to implement such an algorithm. A failed attempt at connecting to the network will use less power than what is assigned to daily communication in the energy budget. In the case where a node has no coverage at all, the node should therefore still be able to function throughout the deployment period. In nodes that have sparse coverage, we would prioritize getting the data back to our servers for this deployment, rather than conserving power. This may be subject to change in future versions if we experience that nodes use more power than anticipated on failed communication cycles.

12

Power consumption - Experiment and results

In this chapter we will take a closer look at the power consumption of the different node versions. As this was one of the major concerns when designing the sensor node, we conducted an experiment to look at the difference in power consumption between the three versions of the node. As the hardware configuration is mostly the same, this will indicate the impact of the software design on the efficiency of the node. Finally, we will discuss the potential impact of the power consumption on deployment period and measurement frequency.

12.1 Methodology

It is more important to find an upper limit for power consumption than to quantify it exactly. As such, if we think that a variable might be hard to measure, we will rather aim to over-estimate it. This means that we can guarantee that the power consumption is less than the value we find here. All current measurements are done with a Fluke 37 multimeter. The power source is an Aim TTi EX355R power supply set at 4.5V.

The first measurement is the low-power mode, also referred to as deep sleep,

power consumption. The sensor nodes spend most of their time in this mode. Optimization of low-power mode consumption is therefore of utmost importance. The way the measurements were planned was to take 100 measurements of the low-power mode consumption and average them.

The power consumption while the node is active was harder to measure than in low-power mode. When operating the power consumption fluctuates significantly. The measurements are therefore not as precise as with the low-power mode. While it would be possible to measure this to a high accuracy, a ballpark estimate is sufficient for our purposes, and can be done with avaliable equipment. The power consumption was estimated by visual inspection of continuous measurements of the current.

The time spent in active mode is also an important variable. This is time spent from waking up from low-power mode, taking measurements, saving it, and going back to low-power mode. The time was measured manually with a stop-watch, and therefore a degree of human error must be expected.

The time spent in the communication cycle is harder to measure. Here variables like connection quality, amount of data to be transmitted, and available updates have a huge impact.

Due to the inherent variability of this value, measuring it reliably isn't practical. In order to preserve the over-estimation ideal, we therefore set the duration of the daily communication cycle to 3 minutes. This value should be significantly higher than the average communication time.

The yearly consumption for each of these three stages is calculated based on the time spent doing a task multiplied by the power consumption.

We have not considered energy usage during the self-test or the camera flash detection. The reason the self-test is excluded is that it is only done once for each deployment. The camera health sensor energy usage is difficult to estimate as it will depend on how frequently the camera trap is visited by rodents. In all versions of the node, this is a small procedure that will take about a second to be completed. Hence, neither should have a significant influence on the yearly energy usage.

12.2 Results

The power consumption of the operative mode of the sensor nodes is very similar between the different versions. This is because it performs essentially

the same tasks. The only meaningful difference is a slight reduction in the time it takes to complete it's tasks in version 2.0.

	V1.0	V1.1	V2.0
Low power mode			
Current	442µA	443µA	24µA
Yearly energy usage	3, 871mAh	3, 881mAh	210mAh
Measurement mode			
Current	70mA	70mA	70mA
Time spent	14.9s	14.8s	14.1s
Yearly energy usage	5, 076mAh	5, 042mAh	4, 803mAh
Communication mode			
Current	120mA	120mA	120mA
Time spent	180s	180s	180s
Yearly energy usage	2, 190mAh	2, 190mAh	2, 190mAh
Total yearly energy usage	11, 138mAh	11, 113mAh	7, 203mAh

We can see from the table there is a significant difference in the yearly consumption of the first two versions and version 2.0. Most of this is due to the low-power mode current draw being much lower in version 2.0. Our experiment shows that fine-tuning the software can significantly improve the efficiency of a given hardware configuration. This illustrates the importance of optimization of power consumption for the most common state. Again, it is important to note that there are no significant hardware differences between versions. These gains in power efficiency are nearly entirely due to software improvements.

The deployed nodes have a total of 14, 000mAh capacity in the batteries. This implies that those nodes should be able to operate for 459 days for version 1.0 and 460 days for version 1.1. Version 2.0 have been equipped with 21, 000mAh total capacity. This, together with with the improved efficiency, results in a maximum deployment time of 1, 064 days, or almost 3 years.

Increasing the deployment period is not beneficial in our case. Instead, we could increase the measurement frequency. Versions 1.0 and 1.1 could both be configured to take measurements every 19 minutes instead of the 30 they currently are configured to. Version 2.0 could be configured to take measurements every 7.5 minutes. This would result in versions 1.0 and 1.1 recording in excess of 75 measurements each day, while version 2.0 recording 192 readings each day. A plot of the maximum number of measurements possible each day is presented in Figure 12.1.

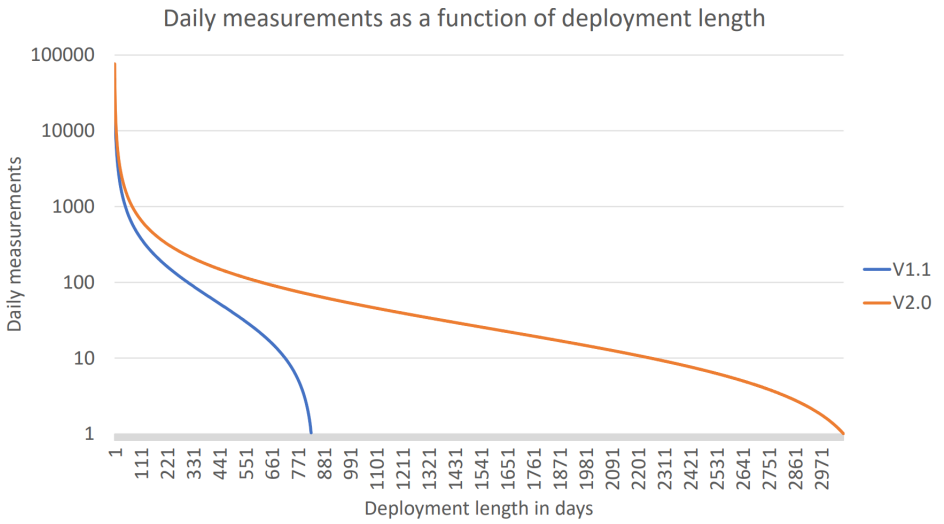

Figure 12.1: Graph showing daily measurements as a function of deployment length (Graph: Øystein Tveito)

We can also use the measurements to estimate the required battery capacity for each of the nodes dependent upon the deployment period. A graph plotting the battery capacity needed for a given deployment period is presented in Figure 12.2.

Measurement frequency, communication frequency, battery capacity, and deployment length can be changed in case this is desired. If only a single reading is needed each day, the sensor node can be deployed for more than eight years according to these calculations. If communication was not important, the node can operate for more than four years before the batteries needs to be replaced, if measuring at a frequency of two measurements per hour. However, one aspect that is not taken into consideration is the self depletion rate of the batteries. All batteries will lose some of its capacity over time. For the batteries we are using, the shelf life is stated to be 20 years. In this time, 10% of the capacity will be dissipated. As such, this is not an important concern, provided that we over-estimate energy usage by at least five per cent.

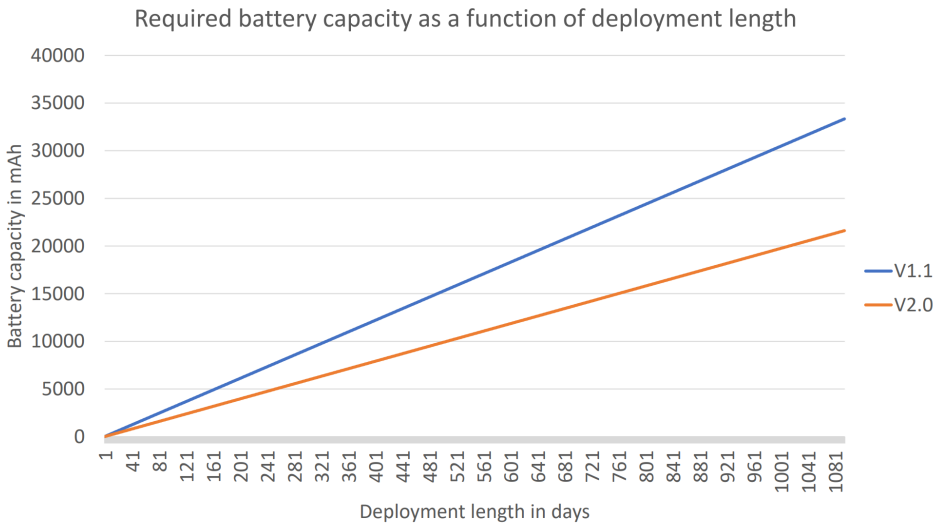

Figure 12.2: Graph showing required battery capacity as a function of deployment length (Graph: Øystein Tveito)

13

Summary of the project

13.1 Status

The status of the project is currently that we have ten units in the field. Of these, we have been in communication with six, but only two of these are communicating frequently. None of the sensors from which we have received measurements have provided CO_2 readings that we believe are correct.

Some of the sensors are constantly giving the value zero for CO_2. This may indicate that the sensor itself is broken. Others report CO_2 values that are far outside the range we expected. If those prove to be correct, that would be very surprising and of high importance to the ecology study. However, we deem it far more likely that the data is either corrupted by the missing calibration, or are heavily dependent upon humidity data to be able to accurately report the CO_2 levels.

The data reported back will be examined further when the nodes are eventually retrieved from the deployment site. The data and logs from the SD-cards will then be analyzed. The hope is that we are able to decorrelate the temperature with the CO_2 readings, providing a somewhat useful result.

The logs of the units that have not reported back will also be subject to investigation. We expect the reason we have not heard from them are that they are without network coverage. If this is not the case, it would be interesting to investigate why they have failed.

Version 2.0 should remedy the problem with incorrect data with a sensor that is better suited for the temperatures to which we are exposing the sensor nodes. The new version is also easier and faster to assemble. This could allow us to deploy far more units to widen the scope of the experiment.

Version 2.0 of the node would be ready for the next deployment. However, due to the global COVID-19 pandemic the deployment plans are more uncertain than when the project started. It is conceivable that this year's deployment will be delayed. It is also possible that it will be cancelled entirely. The current traveling advice is that unnecessary traveling should be avoided.

13.2 Challenges

Challenges, small and big, are a part of any project worth doing. Here we will reiterate some of the more important challenges we encountered throughout this project. Finally, we will take a look on some of the lessons that can be extracted from this.

13.2.1 Part selection

Assembling a complete sensor node based on the data available in data sheets has been challenging. The way the technical data is provided in a data sheet can be quite cryptic for the untrained eye. Due to our somewhat limited experience in interpreting these types of documents, some of the caveats were overlooked when picking components. The CO_2 sensor is an example of such a mistake. Here the information on calibration ranges and humidity dependency was interpreted incorrectly. This lead us to using a sensor which was not calibrated for operation in negative temperatures.

13.2.2 Antenna problems

We did not receive communication from any of the deployed nodes on the first deployment. This was a major problem that we encountered in this project. Our somewhat lacking understanding of antenna theory at the time led to us spending several days in search of the cause of the problem.

Compounding to the antenna placement problem, the reception in the area was lower than we expected. According to the coverage maps we had access to, the reception should have been sufficient. After the antenna placement had been fixed, four units were still not able to communicate due to the sparse

coverage.

13.2.3 Time management

Deadlines are an ever-present concern for most projects. The deadline we had to abide by was set by the planned maintenance trip done yearly by the ecologists at COAT. Planning, designing, implementing, and testing our sensor node in time for this deadline was a challenge. Software development and hardware assembly was ongoing right up to the deadline. This meant that little time was spent on testing the device before deployment. There was also not allocated time for code reviews. Code reviews are usually useful for finding less than optimal code and bugs.

While the hardware design was being developed from the beginning of the project, the software development did not start until the first hardware prototype had been created. This contributed to the time crunch at the end of the project.

Finally, in the start of the project, a lot of time was spent on planning the project and design. No parts were ordered until this stage was completed. While some components were changed in this phase, most of the components were selected early in the planning stage. This meant we had to wait for parts to arrive once the planning stage was completed, subtracting from our available development time.

13.3 Lessons learned

From the challenges we encountered, we have noted some lessons that we have learned. These lessons could have helped us to quickly find good solutions, or avoid these problems completely.

13.3.1 Part selection

Many data sheets do not present their technical data in an easily readable manner. The data sheet is meant for reference and comparison, not as a manual. Spending more time on reading the data sheets might be useful. What we ended up doing after discovering the strange measurements was to e-mail the manufacturer. After a couple of emails back and forth, we had understood how the data sheet was meant to be read. If we had been in direct contact with the manufacturer at an early stage of the project this problem could have

been completely avoided.

The manufacturer's sales representatives usually have comprehensive knowledge about their products and prior use-cases. It is in their interest that we, the customer, are getting the correct information and are successful in our project. Both the resulting sales and the publications that are produced may be profitable or useful for their continued business. Therefore, one should not hesitate to contact the company if any uncertainty exists after reading the data sheet for a component.

Confounding variables must be considered carefully. More thought should have been given to determining which variables might influence the results, apart from what we are trying to measure. In our case, humidity is likely a crucial variable that needs to be included in the equation to calculate the correct CO_2 level.

13.3.2 Antenna problems

The lesson that have been ingrained in us from our encounter with the antenna problem is an important one. When developing any solution, the solution should always be tested thoroughly. We did some testing in and around our lab, but none of the tests done tried to imitate the actual deployment. Testing parts of a solution is of course important, but testing the entire solution as it is meant to be deployed is crucial.

Our testing should have included placing the node in a camera trap enclosure, with the antenna secured to the lid as in the deployment. It should also have included a test of this setup in an area with poor reception. If this had been done we would likely have located the problem before the actual deployment. This would have saved us the second deployment altogether.

While some of the units now are communicating, others are not. The coverage maps provided by Telenor and Telia are clear about the fact that the maps are generated based on algorithms, and not actual gathered data. In our case, shadowing caused by the local terrain is likely to be the reason for the deviation between the map and reality. Measuring the signal strength in the area would have given us some valuable information about the conditions in advance. We would have had the time to create a signal strength test in a simple node. This could then have been shipped to COAT's local technician, who could have taken it to the actual deployment site early on in the project. The deployment site could potentially have been changed, as COAT operates several deployment sites like the one we are using.

13.3.3 Time management

While the deadline for this project could not have been changed, there were several things we could have done differently in our time management. One of the first lessons have extracted from this is that software development should have started earlier.

As stated earlier, software development did not start until we had an assembled hardware prototype. It may be slightly frustrating developing for a system that has not been built yet. Nevertheless, it would have saved us some time to have a base program ready when the first hardware was built. The main operations of the sensor node could be coded on a stand-alone FiPy. The missing hardware functionality could have been emulated with place-holder functions. Programming is usually more time consuming than first anticipated. With software development started early, we could have had time for both code reviews and testing of the system before the deployment. This would also have given us a way of properly testing the hardware early on in the hardware development.

Allocating time for reviewing code and tests is useful. While it does take time away from development, it may save the developer from spending time going in the wrong direction. Additionally, it often is useful to formulate what your code is trying to do in a human language.

Components could have been ordered as soon as we were reasonably sure that the final selection for that component was done. This way, we could have saved some time waiting for components to arrive. This would have resulted in about a week of extra development time. There would be a risk of some parts being removed or replaced from the project after it was ordered, but the added cost of this would be negligible compared to the overall budget.

13.4 Future work

There is still some work that needs to be done until this project can be considered complete. Version 2.0 of the sensor node is still not deployed. This will be done in conjunction with retrieval of the currently deployed version 1.1 nodes. With the old nodes retrieved, we can start analyzing the logs and data. Diagnosing the nodes that have not been able to communicate will then be the main priority. If the logs are not able to give us enough information about why the nodes failed, we would then have to do some more experimentation.

Furthermore, we will take a closer look at the data. The goal is that we will be

able to interpret the data, estimating the real CO_2 values. If this can be done, we would be able to deliver on our original task. The new and improved sensor node should in any case be able to deliver useful data to the ecologists by the following winter.

Additional functionality has been suggested in order to improve its versatility. Top amongst this is the addition of a local network. This would allow sensor nodes with no LTE CAT M1 coverage to use surrounding nodes as gateways. Having this functionality would vastly expand the amount of suitable deployment locations.

14

Conclusion

The task we set out to complete was to create a small, robust, and power efficient node to measure CO_2 concentrations underneath the snow in the Arctic tundra. We have created three distinct versions of a sensor node, fulfilling all the requirements set by the ecologists and the requirements set by us. The experience from the first two deployments has allowed us to test what works, and helped us identify design- and implementation flaws. We have still not been able to deliver the requested data to COAT's ecologists. This is something that we think will change once we are able to deploy the newest versions of our sensor node.

In the end, we have created a scientific instrument that will be able to provide measurements of previously unknown variables. The data gathered might be an important part of the ecological research done in COAT. The sensor node that has been presented here is novel, small, power efficient, and functional. It is able to autonomously monitor the conditions in the camera traps for several years on batteries, while delivering the data wirelessly to our backend servers.

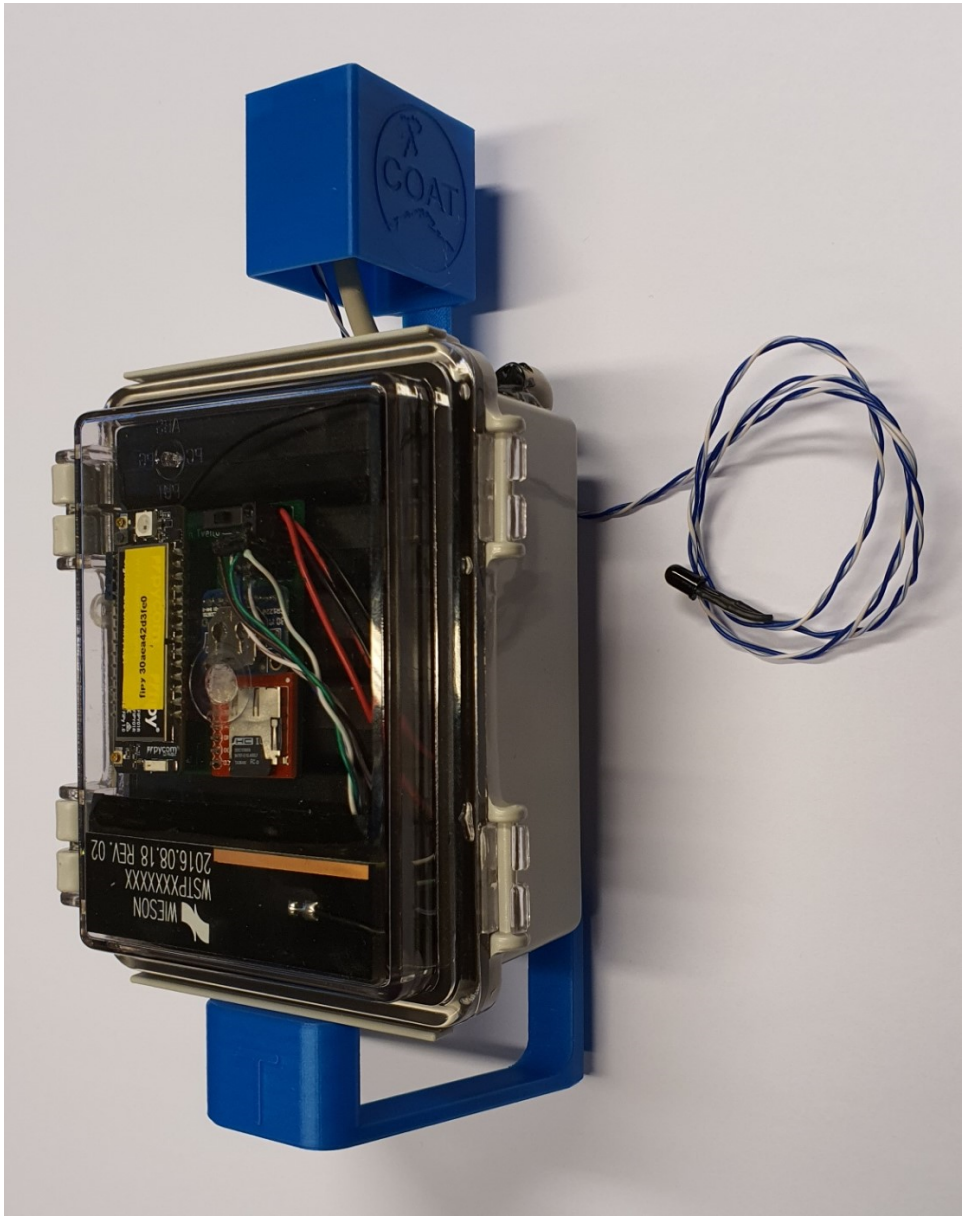

Figure 14.1: Picture of a finished version 2.0 sensor node (Photo: Øystein Tveito)

Bibliography

[1] APT. *Industrial Grade microSD datasheet*, 2020. https://media.digikey.com/pdf/Data%20Sheets/ATP%20Electronics%20PDFs/ATP%20microSD_SDHC.pdf.

[2] Aline Baggio. Wireless sensor networks in precision agriculture. In *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN 2005), Stockholm, Sweden*, pages 1567–1576. Citeseer, 2005.

[3] Guillermo Barrenetxea, François Ingelrest, Gunnar Schaefer, Martin Vetterli, Olivier Couach, and M. Parlange. Sensorscope: Out-of-the-box environmental monitoring. pages 332–343, 05 2008.

[4] Elizabeth A. Basha, Sai Ravela, and Daniela Rus. Model-based monitoring for early warning flood detection. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 295–308, New York, NY, USA, 2008. ACM.

[5] C. Chen, X. Jun-ming, and G. Hui-fang. Polluted water monitoring based on wireless sensor. In *2011 International Conference on Electronics, Communications and Control (ICECC)*, pages 961–963, Sept 2011.

[6] M. Chen, Y. Miao, Y. Hao, and K. Hwang. Narrow band internet of things. *IEEE Access*, 5:20557–20577, 2017.

[7] COAT. *Climate-ecological Observatory for Arctic Tundra*, 2020. https://www.coat.no/.

[8] Earth System Research Laboratory - Global Monitoring Division. *Global Greenhouse Gas Reference Network*, 2020. https://www.esrl.noaa.gov/gmd/ccgg/trends/.

[9] Energizer. *ENERGIZER L91 datasheet*, 2005. https://data.energizer.com/pdfs/l91.pdf.

[10] Espressif. *ESP32 datasheet*, 2020. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[11] Bramer et al. Chapter three - advances in monitoring and modelling climate at ecologically relevant scales. *Adcances in Ecological Research*, 58:101–161, Feb 2018.

[12] S. Evans. Dielectric properties of ice and snow–a review. *Journal of Glaciology*, 5(42):773–792, Jan 1965.

[13] N. J. Poole F. Sarvar and P. A. Witting. Pcb glass-fibre laminates: Thermal conductivity measurements and their effect on simulation. *Journal of Electronic Materials*, 19(12):1345–1350, Dec 1990.

[14] Antonio-Javier Garcia-Sanchez, Felipe Garcia-Sanchez, and Joan Garcia-Haro. Wireless sensor network deployment for integrating video-surveillance and data-monitoring in precision agriculture over distributed crops. *Computers and Electronics in Agriculture*, 75(2):288 – 303, 2011.

[15] Gas Sensing Solutions. *ExplorIR-W CO2 Sensor Datasheet*, 2018. https://web.archive.org/web/20190522120607/https://www.gassensing.co.uk/wp-content/uploads/2018/08/ExplorIR-W-CO2-sensor-datasheet.pdf.

[16] P. Hsieh, Y. Jia, D. Parra, and P. Aithal. An experimental study on coverage enhancement of lte cat-m1 for machine-type communication. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–5, 2018.

[17] Intel. *MCS 51 MICROCONTROLLER FAMILY USER'S MANUAL*, 1994. http://datasheets.chipdb.org/Intel/MCS51/MANUALS/27238302.PDF.

[18] International IOR rectifier. *IRLB8721PbF datasheet*, 2009. https://cdn-shop.adafruit.com/datasheets/irlb8721pbf.pdf.

[19] Thomas C. Krohn, Axel Kornerup Hansen, and Nils Dragsted. The impact of low levels of carbon dioxide on rats. *Laboratory Animals*, 37(2):94–99, 2003. PMID: 12689419.

[20] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, pages 8 pp.–, April 2006.

[21] Françoise Martz, Jaana Vuosku, Anu Ovaskainen, Sari Stark, and Pasi

Rautio. The snow must go on: Ground ice encasement, snow compaction and absence of snow differently cause soil hypoxia, co2 accumulation and tree seedling damage in boreal forest. *PLOS ONE*, 11(6):1–18, 06 2016.

[22] Maxim Integrated. *DS3231 datasheet*, 2015. https://cdn-shop.adafruit.com/product-files/3013/DS3231.pdf.

[23] Maxim Integrated. *DS18B20 datasheet*, 2019. https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

[24] Mbed. *Building IoT devices with JavaScript*, 2020. https://os.mbed.com/javascript-on-mbed/.

[25] Wesley J. McBride and Jason R. Courter. Using raspberry pi microcomputers to remotely monitor birds and collect environmental data. *Ecological Informatics*, 54:101016, 2019.

[26] James G. Mickley, Timothy E. Moore, Carl D. Schlichting, Amber DeRobertis, Emilia N. Pfisterer, and Robert Bagchi. Measuring microenvironments for global change: Diy environmental microcontroller units (emus). *Methods in Ecology and Evolution*, 10(4):578–584, 2019.

[27] MicroPython org. *MicroPython*, 2013. http://micropython.org/.

[28] M. A. Nasirudin, U. N. Za'bah, and O. Sidek. Fresh water real-time monitoring system based on wireless sensor network and gsm. In *2011 IEEE Conference on Open Systems*, pages 354–357, Sept 2011.

[29] Sajid Nazir, Scott Newey, R. Justin Irvine, Fabio Verdicchio, Paul Davidson, Gorry Fairhurst, and René van der Wal. Wiseeye: Next generation expandable and programmable camera trap platform for wildlife research. *PLOS ONE*, 12(1):1–15, 01 2017.

[30] NEMA. *NEMA Enclosure Types*, 2015. https://www.nema.org/Products/Documents/nema-enclosure-types.pdf.

[31] Polycase. *Polycase WH-02*, 2019. https://www.polycase.com/wh-02.

[32] Pycom. *FiPy datasheet*, 2017. https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_FiPy_v2.pdf.

[33] Raspberry pi foundation. *Raspberry pi*, 2020. https://www.raspberrypi.org/.

[34] Luis Ruiz-Garcia, Loredana Lunadei, Pilar Barreiro, and Ignacio Robla. A review of wireless sensor technologies and applications in agriculture and food industry: State of the art and current trends. *Sensors*, 9(6):4728–4750, 2009.

[35] Karl E. Schaefer. Effects of increased ambient co2 levels on human and animal health. *Experientia*, 38(10):1163–1168, Oct 1982.

[36] Senserion. *SHT1X datasheet*, 2011. `https://cdn-shop.adafruit.com/datasheets/Sensirion_Humidity_SHT1x_Datasheet_V5.pdf`.

[37] Eeva M. Soininen, Ingrid Jensvoll, Siw T. Killengreen, and Rolf A. Ims. Under the snow: a new camera trap opens the white box of subnivean ecology. *Remote Sensing in Ecology and Conservation*, 1(1):29–38, 2015.

[38] Telenor. *Dekningskart IoT*, 2019. `https://www.telenor.no/bedrift/iot/dekning/#map`.

[39] Telia. *Dekningskart IoT*, 2019. `https://www.telia.no/dekning/`.

[40] Dagrun Vikhamar-Schuler, Ketil Isaksen, Jan Erik Haugen, Hans Tømmervik, Bartlomiej Luks, Thomas Vikhamar Schuler, and Jarle W. Bjerke. Changes in winter warming events in the nordic arctic region. *Journal of Climate*, 29(17):6223–6244, 2016.

[41] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, OSDI '06, pages 381–396, Berkeley, CA, USA, 2006. USENIX Association.

[42] Ming Xia, Yabo Dong, Wenyuan Xu, Xiangyang Li, and Dongming Lu. Mc2: Multimode user-centric design of wireless sensor networks for long-term monitoring. *ACM Trans. Sen. Netw.*, 10(3):52:1–52:30, May 2014.