Master's thesis

Andreas Moan

Developing a Small and Power Efficient Depth Measuring System for Use at Remote Locations

Master's thesis in Mechanical Engineering
Supervisor: Bjørn Haugen
June 2019

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

agder energi
God kraft. Godt klima.

Developing a Small and Power Efficient Depth Measuring System for Use at Remote Locations

Andreas Moan

TMM4960 - Master's thesis at The Department of Mechanical and Industrial Engineering

andremoa@stud.ntnu.no

June 2019

Abstract

The hydroelectric power production company Agder Energi, wants to develop a new depth measuring system for use at remotely located rivers and reservoirs. This system shall be small, light, power efficient, user friendly, reliable and cost efficient. The goal is to supplement, and to some degree replace, the large and costly stations they are using today. The objective of this master's thesis is to develop this depth measuring system for Agder Energi. The majority of the report deals with the creation of a single-node system; one node that does all the measurements and sends the data back to Agder Energi's contol central via the NB-IoT mobile network protocol. The later part of the report deals with the development of a multi-node system, one that is able to operate outside of areas where mobile network is available. It does so by utilizing radio communication in addition to NB-IoT. Agder Energi has the single-node system as their top priority, and the work on this system will by the end of this project be concluded. The multi-node system will however have work remaining by the end of this project. Although long-term testing still remains for the final version of the single-node system, the preliminary results seems quite promising regarding all the criteria set by Agder Energi. The system will be put into use soon after the completion of this master's thesis.

Sammendrag (abstract in Norwegian)

Vannkraftproduksjonsselskapet Agder Energi ønsker å utvikle et nytt dybdemålingssystem tilegnet bruk i avsidesliggende elver og reservoar. Dette systemet skal være lite, lett, energieffektivt, brukervennlig, pålitelig og kostnadseffektivt. Målet er å supplere, og til en viss grad erstatte, de store og dyre stasjonene som blir brukt per dags dato. Formålet med denne masteroppgaven er å utvikle dette dybdemålingssystemet for Agder Energi. Mesteparten av rapporten omhandler utviklingen av et enkelt-node-system; én node som tar alle målingene, og sender dataen tilbake til Agder Energis driftssentral via mobilnettverksprotokollen NB-IoT. Den senere delen av rapporten omhandler utviklingen av et multi-node-system som kan operere utenfor områder med mobildekning. Den gjør dette ved å ta i bruk radiokommunikasjon i tillegg til NB-IoT. Agder Energi har enkelt-node-systemet som sin førsteprioritet, og arbeidet på dette systemet vil i løpet av prosjektet bli fullført. Multi-node-systemet vil derimot fremdeles ha gjenstående arbeid ved slutten av prosjektet. Selv om langtids-testing av den endelige versjonen av enkelt-node-systemet fremdeles gjenstår, ser de foreløpige resultatene svært lovende ut sett opp mot kriteriene satt av Agder Energi. Systemet vil bli tatt i bruk like etter at denne masteroppgaven er fullført.

Preface

First of all, I want to thank my supervisor; associate professor Bjørn Haugen, and my contact person at Agder Energi; Trym Bjønnes. Haugen has had an open door-policy, which have allowed me to drop by whenever I had things to discuss. This is something I have appreciated, knowing I don't have to wait two weeks for a scheduled meeting. My talks with Haugen have let me know whether I'm approaching the task in an appropriate manner. Bjønnes have given valuable feedback throughout the entire project. This feedback has been crucial for the en product becoming what it is today. I would also like to thank Agder Energi for initiating and sponsoring this project, and NTNU for giving me access to useful equipment and tools.

The skills I have gained during the course of my study at NTNU, including the last semester working on this project, will become useful as I journey from being a student into my new job as a product developer in the field of mechatronics. A lot have been learned throughout the duration of this master's thesis. In addition to having gained pure technical knowledge, I have also experienced how it feels to relate to a company as both a partner and a customer.

I would like to end this preface with a quote by Maya Angelou which is both inspiring and comforting.

"Do the best you can until you know better. Then when you know better, do better."

- Maya Angelou

Andreas Moan
June 2019, Trondheim

Contents

Abbreviations

328P	ATmega328P
3GPP	3rd Generation Partnership Project
A	Ampere
ADC	Analog-to-digital converter
AE	Agder Energi
CAD	Computer Aided Design
CRC	Cyclic Redundancy Check
CSV	Comma Separated Value
GPIO	General-Purpose Input/Output
GSM	Global System for Mobile communications
IDE	Integrated Development Environment
IoT	Internet of Things
ISR	Interrupt Service Routines
JSON	JavaScript Object Notation
MCU	Microcontroller Unit
NB-IoT	Narrow Band Internet of Things
NKOM	The Norwegian Communication Authority
NMEA	National Marine Electronics Association
NVE	Norges vassdrags- og energidirektorat (Eng: The Norwegian Water Resources and Energy Directorate)
PCB	Printed Circuit Board
RF	Radio Frequency
RMC	Recommended Minimum Navigation Information
RSSI	Received Signal Strength Indication
SAMD21	ATSAMD21J1
SCADA	Supervisory Control And Data Acquisition
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
V	Volts
$\mathbf{V_{CC}}$	Voltage common collector
WDT	Watchdog Timer
XML	Extensible Markup Language

Supplementary files

The code referred to throughout this report can be found in the attached files. These files aren't needed in order to read this report. They can however give a deeper insight to some of the subjects discussed. There are 42 code files, all in the format *.cpp, *.h or *.ino. The code files can be opened and read in most text editors, although it's recommended to use code compatible editors like Notepad++ or Visual Studio to properly highlight the syntax. Agder Energi have chosen to have an open source approach to this project. All the code can therefore also be found online at *https://github.com/andremoa/depthStations*.

Code files for the single-node system:

- singleNode.ino
- alarmFunctions.ino
- dataProcessingFunctions.ino
- dataAquizitionFunctions.ino
- networkFunctions.ino
- userInputFunctions.ino
- portDefines.h
- globalVariables.h
- bitPatterns.h

Note: Several other files are needed to run the main code (SODAQ_node.ino). These files can however be downloaded and included directly from the integrated development environment. They are therefore not included in this list.

Code files for the sensor node in the multi-node system:

- main.cpp
- bitPatterns.h
- clock.cpp
- clock.h
- Framebuffer.cpp
- Framebuffer.h
- GPS.cpp
- GPS.h
- I2C.cpp
- I2C.h
- LoRa.cpp
- LoRa.h
- powerManagement.cpp
- powerManagement.h
- SDI12.cpp
- SDI12.h
- SPI.cpp
- SPI.h
- SSD1306.cpp
- SSD1306.h
- USART.cpp
- USART.h

Note: Framebuffer.cpp, Framebuffer.h, SSD1306 and SSD1306.h are not self-written. They are open source libraries found online. Their lisence agreement can be found in their header files. The same goes for SDI12.cpp and SDI12.h. They are not self written, but they have been modified for the purpose of this project.

Code files for the gateway node in the multi-node system:

- multiNode_gateway.ino
- functions.ino
- bitPatterns.h
- globalVariables.h
- portDefines.h
- RHReliableDatagram.cpp
- RHReliableDatagram.h
- RH_RF95.cpp
- RH_RF95.h
- RHDatagram.cpp
- RadioHead.h

Note: RHReliableDatagram.cpp, RHReliableDatagram.h, RH_RF95.cpp, RH_RF95.h, RHDatagram.cpp and RadioHead.h are not self-written, but they have been modified for the purpose of this project. They are open source libraries found online. Their lisence agreement can be found in their header files.

1 Introduction

1.1 Background and motivation

Agder Energi (AE) is one of the major power production companies in Norway, producing and distributing electrical power to the Agder region (Agder Energi 2018). Their production consists of 99% hydropower, produced by their 47 hydroelectric power plants. It's important for AE to at all times have knowledge about the water levels of their reservoirs and the water flow in the downstream rivers. The Norwegian Water Resources and Energy Directorate (NVE) have regulations which requires a minimum water flow in the downstream rivers of their reservoirs. AE wants to stay as close to these minimum requirements as possible in order to direct more of the water into their turbines. At the same time, to avoid breach of the regulations, AE releases more water into these rivers than the minimum specification requires. This water is released directly from the dams, and thus bypasses the turbines. The potential loss of income due to this buffer margin from only one of AE's dams is approximately 8 million NOK per year[1,2] (Austrud 2018). In addition to the financial importance of the sensor data, the data can also be used in flood prevention. AE serves an important role in flood control. By knowing the depths and flows in various location, they can actively open and close their hatches to minimize the flood (Bjønnes 2019).

AE uses depth measuring stations for both depth measurements and for water flow measurements. By knowing the depth of the water at given locations in the rivers, they can calculate the water flow[3]. The depth measuring stations AE are using today are hydrology stations delivered by Scanmatic. Although these stations are reliable in terms of giving the right data, they do have their drawbacks. They are large and heavy, they are expensive, they consumes a lot of power, the batteries often go flat if the solar panel is covered by i.e. snow for a longer period of time, and they aren't intuitive to operate, meaning only trained personnel can operate them. Due to these inconveniences, the Scanmatic stations are deployed at fewer location than what AE considers optimal. At some locations it takes roughly 20 hours from a valve is opened, to the results can be measured by a sensor in the downstream river. AE therefore wants to create a new depth measuring system which don't have the same drawbacks as the ones they are using today. They want stations that they can deploy with a higher and more optimal quantity.

1.2 Previous work

In the spring of 2018, AE hired two students to work on a summer project where the goal was to make a proof of concept of a cheap, small and low powered depth measuring system. The outcome of the summer project was a system consisting of three units. A sensor node, doing the measurements and sending it onward using radio communication. A repeater, receiving the message from the node before sending it onward again. And a gateway, receiving the message from a node or repeater, and storing the data. All communication between the units happened via radio. Although the system proved successful as a proof of concept regarding range, prize and power consumption, it also had some crucial drawbacks. The system was unstable. Sometimes it froze during its operation. Sometimes it collected wrong or corrupted data, and sometimes it collected no sensor data at all. Some of the desired features like automatic time update were unsuccessfully attempted to implement. AE asked the students to take this project onward to a specialization project and master's thesis, whereof one of the students (the author) accepted.

[1]This is the *theoretical* maximum saving from this dam if the water flow had been exactly on the minimum requirement at all time.

[2]The dam can't be named here for confidential reasons.

[3]More about this in chapter 2.1

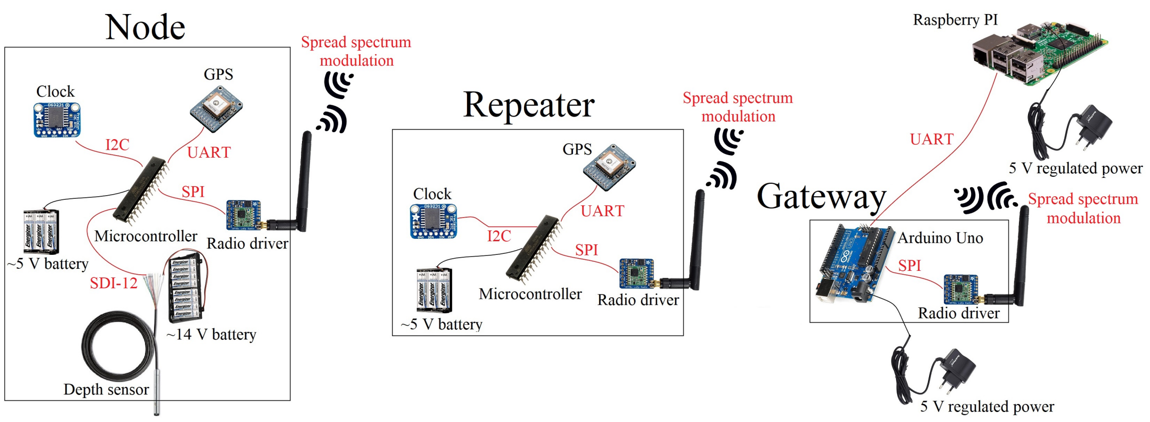


Figure 1: A simplified diagram of the system from the specialization project. Component names are in black letters, while their communication protocols are in red letters.

The focus of the specialization project was to improve upon the system that was created during the summer project. The code was rewritten from scratch in the microcontrollers native AVR-C language. This proved to stabilize the system's operation in terms of acquiring the correct data without freezing. At the same time it reduced the microcontrollers RAM usage from 82% down to 17%. A simplified diagram of the outcome of the specialization project is shown in figure 1. The system was not finished by the end of the specialization project, and the plan was to continue and finish the project as a master's thesis.

1.3 Goals and priorities

The system that was begun developed in the specialization project was solely based on radio communication for transmitting the data from the field to a base station. This setup could at some locations result in a long series of repeaters, if the nearest base station was far away. As a possible solution to this problem, it was suggested to AE that the master's thesis could change focus to instead use NarrowBand-IoT (a mobile network communication protocol) in combination with radio drivers in order to reduce the numbers of nodes needed to send data back from the field. The mobile network coverage map form Telia (2019) shows that most locations of interest are either in a zone where mobile network is available, or has a distance to an area with mobile network available that is shorter than the distance to the nearest base station. AE liked the idea of utilizing NarrowBand-IoT. They had already thought about the idea of exploring this technology themselves. It was therefore agreed that the focus of this master's thesis should be directed towards utilizing NarrowBand-IoT.

This new strategy means that two separate systems should be created. One of the systems can consist of one single node that does all the measurements before sending it to AE via NarrowBand-IoT. This node must be placed in a location where mobile network is available. The other system must consist of two or more nodes. One sensor placed outside of the mobile network's coverage area. This node does the measurements and sends the data via radio to a gateway node placed inside the coverage area, which then sends the data onward to AE via NarrowBand-IoT. If necessary, repeaters can be placed in between these two nodes.

AE wants the single-node system to be the main priority of this master's thesis, since the majority of the desired measuring locations are within coverage of mobile network. The multi-node system shall be developed only if there's time, and if it doesn't negatively affect the development of the single-node system.

1.3.1 Must-haves and nice-to-haves

AE has several goals for the systems. These goals can be split into two categories; *must-haves* and *nice-to-haves.*

Must-haves, or requirements, are the features of the end-product that must be present in order for AE to consider using the system. This is first and foremost that the system is able to send the correct sensor data back from the field. AE's operations center will perform actions based on the received data. If the data is wrong, it can do more harm than good. The node must be able to do measurements relatively often. AE thinks that one set of measurements every 15 minutes is a good frequency. The end-product should also be relatively durable and maintenance free. If a remotely stationed nodes requires physical handling every few weeks, it will become a burden to use them.

Nice-to-haves are the features that should be present in order to make the product more attractive to use. These features are low cost, a simple and efficient user interface, small size, light weight and a low power consumption. All of these features can be thought of as non-discrete, qualitative properties since there aren't any given tangible goals. It's hard to say exactly when the properties are good enough. These features should instead be pushed as far as possible in the right direction.

1.4 Problem description

The aim of this master's thesis is to develop a reliable, low powered, user friendly, cheap, small and light depth measuring system for use at remote locations. This system should send data back to Agder Energi via NarrowBand-IoT once every 15 minutes. The primary goal is to create a single node system which can operate at locations where mobile network is available. If there's time, a second system consisting of two or more nodes, using both NB-IoT and radio communication, shall be created. This system should then be able to operate outside of the mobile network coverage area.

1.5 Work methodology

Following up on AE's requests, this development project is executed in accordance with the given priorities. The focus is first and foremost directed towards the primary goal of creating the single-node system. The multi-node system is second priority, and is thus worked on after the development of the single-node system has delivered a satisfying outcome. Work on these systems is at times also done in parallel where it seems advantageous, since some features can be used in both setups.

The importance of user interaction in the process of product development is heavily emphasized in the literature (Taha, Alli, and Abdul-Rashid 2011). AE is involved throughout the entire development processes via user co-design. User co-design is the acts of participatory and cooperative creation (Chisholm 2014). AE are both giving feedback on the choices being made, as well as suggesting new features that can be implemented. This will in turn lead to iterations of the solutions presented to them.

Time constraints often lead engineers/designers to focus on product realization instead of iterating on several solutions (Schrage 1999). Experiments have nonetheless shown that this is not the optimal strategy, even when there's a short deadline (Dow, Heddleston, and Klemmer 2009). Iterative prototyping helps the designer unveil both new issues and new opportunities. Several aspects of the depth measuring system(s) created here are iterated upon several times in order to find the best solutions to the challenges.

Since this is a practical product development project, not a pure scientific thesis, some non-traditional sources will be used to acquire information on certain topics and problems encountered during the process. Data sheets, product information pages, and to a small degree forums, will be used in addition to the scientific papers to gather the necessary information.

1.5.1 Structure of the report

This report is structured into four main parts. Part one is the introduction, theory and an analysis of some existing depth measuring systems. Part two is the development process of the single-node setup. Part three is the development process of the multi-node setup. Lastly, part four contains the discussion and conclusion related to the project as a whole. The header text on each page shows which part of the report the page belongs to. All parts, except for the first, are clearly introduced with a headline on a separate page.

2 General theory

This report describes the development of a remote monitoring system. A vast variety of topics will be encountered throughout this process. Covering the theory of all the topics in a few sections at the beginning of the report would most likely feel out of place. New theory will therefore to some degree be introduced later on in the report when the information is needed. However, some theory which can be useful throughout the report will be presented here.

This theory chapter consists of three subsections. First some general theory about water flow and depth measurement will be presented. The second subsection will present some basic theory about microcontrollers. Lastly, the third subsection will briefly explain what NarrowBand-IoT and radio communication is and how it works. These three topics are chosen because they build the foundations as to how the products should be designed.

2.1 Water flow and depth measurement

There are several ways of measuring a stream flow rate (Suneco Hydro 2019). For household devices like showers, small measuring tools can be attached. *The bucket method* can be used to measure the water flow of small rivers. Directly measuring the water flow of large rivers on the other hand can be a bit trickier. A method that is often used, including by AE, is to direct the river to flow through sections of simple geometry at certain checkpoints. If one is able to measure the height of the water over these checkpoints, the water flow rate can then be calculated. This method is known as the weir method.

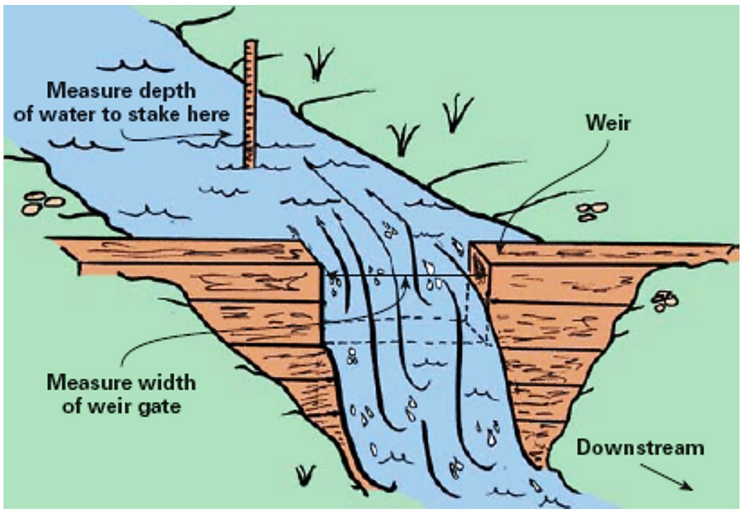

Measure depth of water to stake here
Weir
Measure width of weir gate
Downstream

Figure 2: Water flow measurement using the weir method. Figure by Suneco Hydro (2019).

The depth measurements shouldn't be made directly at the geometry, since the readings will have a tendency to get distorted by the river when it rapidly changing its geometry. Instead the measurements should be taken a bit upstream, or in a nearby still backwater if available. At these close-by locations, the water will rise and fall by the same amount at the same time as through the geometry. The measurements must however be calibrated to correspond to the water height through the weir. If the depth upstream is 140 mm shallower than at the weir, then 140 mm must be added to every measurement.

In terms of predicting the upcoming water flow rate, air temperature measurements are also of interest (Bjønnes 2019). Temperature affects how the water flows from one point to another. There are many factors playing a role here. For example if it's hot, some water will vaporize and thus decrease the water flow. At the same time it's more likely that snow will melt from the mountains, increasing the water flow. Interpreting this data to make predictions is a science of its own, and it requires knowledge of the local geography.

There are several ways of measuring the depth itself. Some sensors use sound (Flowline Inc. 2010), while other use floaters (Seba Hydrometrie 2019). AE uses two different sensors which they connect to their measuring stations. These are the OTT PLS and the UNIK 5000. Although the first is a digital sensor (OTT HydroMet 2018) and the latter is an analogue sensor (Tormatic AS 2019), they both measure pressure in order to calculate the depth. At some locations, like in reservoirs, it's the depth itself that is of interest to AE. Not the water flow.

2.2 Basic theory of microcontrollers

To put it simply, a microcontroller is a toned-down version of a computer. They're cheap, they're small, they use little power and they are good at interacting with the real world. This is why microcontrollers will serve as the "processing- and command centre" of the systems that will be created in this project. It's therefore useful to have some general knowledge of what microcontrollers are and how they work. The field of microcontrollers is quite vast, so what will be covered here is only the basics which is needed to understand the topics discussed later on in the report. The book *Make: AVR Programming* by Elliot Williams can be recommended if further reading is of interest.

2.2.1 Overview

A microcontroller is basically a single chip computer. It has a central processing unit (CPU), random access memory (RAM), flash memory and an arithmetic logic unit for mathematical processing (Williams 2015). Most microcontrollers also have additional dedicated integrated hardware for other tasks like analog-to-digital conversion (ADC), serial communication, two-wire interface (TWI) and more. Despite being a computer, a microcontroller is different from a normal laptop or stationary PC. The most noticeable difference might be that a microcontroller usually don't have an operating system, and the computational power is lower than what a x86 (e.g. Intel or AMD) processor has. Where the CPU of a PC has a clock speed of 2-3 GHz or more, microcontrollers usually operates in megahertz. This gives the microcontrollers certain limitations in advanced graphical processing and other high-demanding tasks. What microcontrollers are good at on the other hand, is interacting with the real physical world. Microcontrollers are for example used in microwaves, electrical toothbrushes, drones, washing machines, subsystems of cars and smoke detectors just to mention a few. They register inputs, processes the information, and then gives the appropriate output.

Most microcontrollers have only digital outputs and inputs. The output of a microcontroller is either a logic high voltage or a logic low voltage. The same goes for the input. A microcontroller can only read an input voltage, and compare the voltage to a reference voltage to see if the input is higher than the reference or not. However, microcontrollers who also have an ADC, can use the boolean comparison to find an approximate value of an analogue input. The way an ADC works is by taking a constant voltage, say 3.3 V, and regulate it down to $3.3V \times \frac{1}{2}$. Then it checks to see if the input voltage is higher or lower than this. If the input voltage is higher, it regulates the constant voltage to $3.3V \times \frac{3}{4}$ before comparing them again. This comparison and regulation happens as many times as the resolution of the ADC allows. A 10-bit ADC has 10 comparisons. In the end, the microcontroller knows which two increments the input voltage lies between.

One of many commonly used microcontrollers is the AVR ATmega328 manufactured by Microchips (originally made by Atmel). A diagram of this microcontroller is shown in figure 3. This is the microcontroller that was used in the specialization project, and it might be used for the multi-node system here as well. It will therefore be used as a general purpose example of a microcontroller.

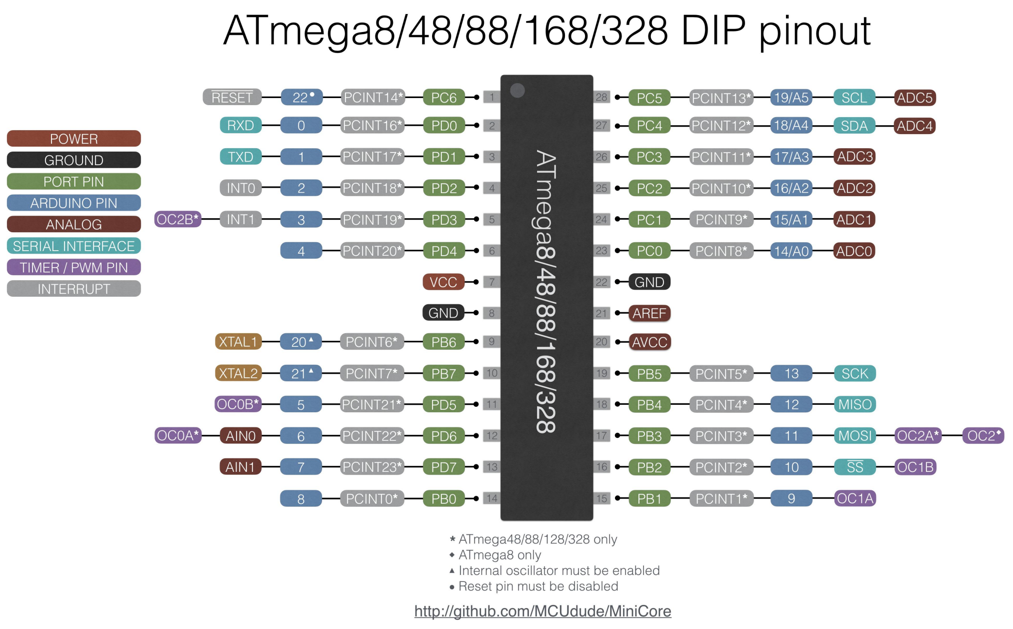

ATmega8/48/88/168/328 DIP pinout
POWER
GROUND
PORT PIN
ARDUINO PIN
ANALOG
SERIAL INTERFACE
TIMER / PWM PIN
INTERRUPT
ATmega8/48/88/168/328
* ATmega48/88/128/328 only
ATmega8 only
Internal oscillator must be enabled
Reset pin must be disabled
http://github.com/MCUdude/MiniCore

Figure 3: Pinout diagram of the ATmega328.

The ATmega328 has a single core CPU, meaning it can execute only a single instruction at a time[4]. This might seem limiting to a microcontrollers application, but it is usually not a major problem. This is due to what is called *interrupt service routines* (ISR). Although microcontrollers only can execute one instruction at a time, they can always simultaneously listen for external and internal events that, if triggered, makes the microcontroller abort its current task, execute a separate set of instructions, and then return to the block of code where it left of. These separate sets of instructions are the ISRs. An internal event can for example be a timer overflow, while an external event can be a pin changing state (going from low to high, or vice versa).

2.2.2 Programming of microcontrollers

A microcontroller can be programmed in several languages. Almost every microcontroller in existence has associated compilers that can interpret either *C*, *Assembly* or both. The code for the microcontrollers can be written in any preferred text editor. After the code is written, it must be linked to a compiler which converts the code from the chosen language to machine code in a *.hex format. Another program then flashes[5] the code onto the microcontroller.

Writing code for microcontrollers is a lot about handling registers. A register is a location in the memory of the microcontroller whose value determines one or more properties of the microcontroller. Modifying bits in a register can change one or several of these properties. A basic example of how to

[4]However, since the AVR's registers are directly connected to the Arithmetic Logic Unit, the microcontroller can in certain cases access two registers in a single clock cycle (Microchip Technology Inc. 2018a)

[5]*Flashing* a microcontroller means transferring the code to the microcontroller and storing it in it's flash memory.

edit registers is shown below, where a GPIO pin on the ATmega328 is set to a low voltage output. GPIO is short for General-Purpose Input/Output. GPIO pins can, as the name implies, be used to both input and output. For the ATmega328, all pins labeled *PORT PIN* can be used as GPIO pins. How much current these pins can supply/drain differs from microcontroller to microcontroller. The ATmega328 for example, can supply/drain a maximum of 40 mA per pin. Pin 12 of the ATmega328 is used in this example. Figure 3 shows that pin 12 is labeled PD6. This means that pin 12 is the sixth bit located in the D-bank register.

$$DDRD \;\; |= 0b01000000; \quad //Set\ PD6\ as\ output\ in\ the\ data\ direction\ register$$

$$PORTD \;\; \&= \;\; 0b10111111; \quad //Set\ the\ pin\ to\ a\ logical\ low$$

These commands can of course also be written in a shorter form using bit-shifting:

$$DDRD \;\; |= (1 << 6); \quad //Set\ PD6\ as\ output\ in\ the\ data\ direction\ register$$

$$PORTD \;\; \&= \;\; \sim (1 << 6); \quad //Set\ the\ pin\ to\ a\ logical\ low$$

2.3 Wireless communication

Two useful types of wireless communication will be looked into here. These are peer-to-peer radio communication and NarrowBand Internet of Things.

2.3.1 Radio waves

Radio waves, or radio frequency (RF) signals are electromagnetic waves consisting of electric and magnetic waves with amplitudes oriented perpendicularly to each other (Moorthy, Sankar, and Kumar 2017). The distinct orientation and amplitudes of the waves gives the signal its polarity. This has practical implications, as the receiving antennas must be oriented to match the polarity of the signal. The frequency of radio waves are the number cycles per second the electromagnetic waves oscillates at. This frequency is also important in practical applications for several reasons, including in the choice of antennas. As electromagnetic waves travels nearly constant to the speed of light[6], the frequency determines the wavelength of the signal (National Radio Astronomy Observatory 2018). The design of an antenna determines what wavelengths it's able to transmit and receive at. Especially the length of the antenna is an important parameter. Antennas must therefore be chosen with a design that effectively transmit and receive at the desired frequency range.

2.3.2 Peer-to-peer radio communication

Radio communication in a peer-to-peer network structure is when two or more devices communicates directly to each other via radio waves without the need for a central server (Tiwana 2014). A server is a unit in the system that provides a service. A client is a unit in the system asking the server for a service. All units in a peer-to-peer system acts as both a server and a client, sharing services like information, processing power or storage space with each other.

The acquisition of radio transmitters can legally be done by anyone. There are however rules as to how one are allowed to operate these radio drivers. The Norwegian Communication Authority (NKOM) states that frequencies in the electromagnetic frequency spectrum can only be used if NKOM or The Ministry of Transportation have given the proper permission to do so (NKOM 2019). There are however certain exceptions to this rule. *Fribruksforskriften* specifies frequencies and associated maximum power outputs that can be used by anyone (Samferdselsdepartementet 2012). It also gives

[6]Some variations occur depending on the medium the signal travels through.

permission to use reserved frequencies and powers if certain criteria are met. § 7 states that the frequency band 870–875,6 MHz, with a maximum bandwidth of 200 kHz and a maximum effective radiated power of 500 mW with a maximum transmission time of 2.5 % can be used by among other measuring equipment monitoring water and meteorology.

2.3.3 NarrowBand Internet of Things

An Internet of Thing (IoT) device is basically any sensory device that is able to send data via the internet (HCL Technologies 2019). For many of these devices, battery life is one of the most important aspects, while having a low bandwidth internet connection is enough for it to send the data. The 3rd Generation Partnership Project (3GPP), a telecommunications standard development organizations (3GPP 2019), have developed a telecommunication protocol from the ground up for this specific purpose. This protocol is called NarrowBand IoT (NB-IoT). The maximum transmission peak is at 250 kbps at a bandwidth of 200 kHz (Ericsson 2019). This low bandwidth allows thousands of IoT devices to connect to a singe base station by operating both at and in between the normal cellphone bandwidths. NB-IoT have a high latency of 1.5 to 10 seconds, and it's not designed for roaming since it can only be connected to one cell-tower at a time. Another 3GPP protocol called LTE-M can be used as a compromise between NB-IoT and the normal LTE protocol. LTE-M have a 1 Mbps peak data-rate, a latency of 50-100 ms, and supports continuous cell-tower connection which allows for roaming, while at the same time use little battery. NB-IoT does however have cheaper hardware and a better signal penetration through obstacles like concrete and trees. For stationary devices like parking meters or depth sensors that only need to send little data once in a while, where a few seconds latency isn't critical, NB-IoT is usually concidered the best option.

Mobile networks also use RF signals for communication. The major difference from peer-to-peer communication is that this communication is based on the *client-server model*, where a client communicates with a server which again processes or re-directs the data (IBM 2019).

3 Analyzation of two existing depth measuring systems

Two existing depth measuring stations will be analyzed here in order to see how well the solutions they have chosen works. The positives can then be taken onward into the system to be created in this project, while the negatives can be avoided. The depth *sensors* themselves will not be discussed here since they are bought separately from the stations.

3.1 Scanmatic

AE are using two depth measuring systems as of today (Bjønnes 2019). The most used system is one delivered by Scanmatic. These HydMet stations, as they're called, are powered by both a 12 V battery and by a solar panel where grid power isn't available (Scanmatic AS 2018). One of these nodes and its battery can be seen in figure 4. This is the *mobile briefcase* version of the station. Most of the stations are inside cabinets, where there's also room for the battery. The Scanmatic nodes sends data via GPRS on the 2G and 3G cellular network, and some also uses a separate modem to send data via SMS (Høgevold 2019). Where mobile network aren't available, they use satellite communication. The data is sent in a non-standard format, so all messages must be translated by Scanmatic software into a standard format before being usable by AE[7]. In addition to depth sensors, sensors for measuring temperature, rainfall and more can also be connected to these nodes.

[7] This is not the same as encryption/decryption for security reasons.

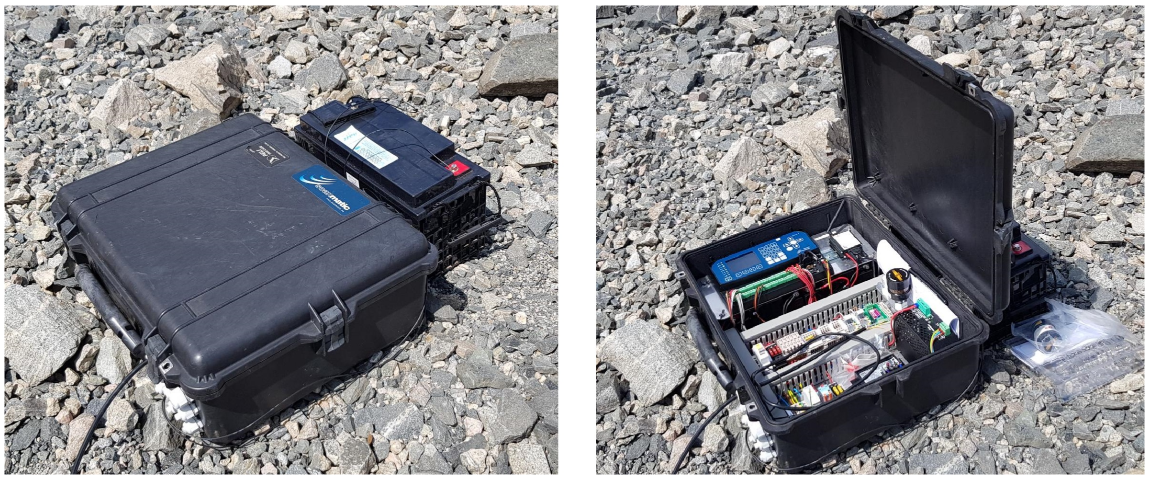


Figure 4: One of the Scanmatic depth measuring stations.

The strengths of the Scanmatic stations is that they are reliable in giving the right measurements every time. They have an interface which easily allows the user to change functionality like data upload intervals. And if any technical problems should occur, Scanmatic will provide good support. However, there are a few drawbacks. The price is quite high. Each HydMet node costs between 55,000 to 115,000 NOK, depending on technical specifications (Bjønnes 2019). The dimensions of the station shown in figure 4 are 560 x 450 x 260 mm, excluding the battery. The pelicase and the electronics weighs roughly 10 kg, and the 12 V battery weighs about 22 kg, giving the system a total weight of roughly 32 kg. These stations have a relatively high power consumption, which is partially due to the HydMet stations ability to constantly monitor the depths. At the same time, the high power consumption makes the stations vulnerable to battery drainage at winter times when snow covers the solar panel.

It seems that these stations are designed using several standard general-purpose modules which aren't optimized for the specific job of the HydMet stations. Scanmatic delivers a range of equipment to a variety of customers like Statkraft, Jernbaneverket and Statens Vegvesen to mention a few (Scanmatic AS 2019). Instead of designing the stations from the ground up, they have probably chosen components and setups they already are familiar with and that they know will work[8]. This however, will in turn make the stations unnecessary big, heavy and expensive.

The HydMet station is a good and reliable product, but it's just not specialized for the task demanded by AE.

The other depth measuring system that AE uses, the RTU500, will not be discussed here. This is because the RTU500 is constantly connected to grid power and cabled network (ABB 2018). They are in other words not concerned with many of the challenges relevant to this project.

3.2 Libelium

A IoT company called Libelium have developed a water measuring system called Smart Water, where different sensors can be connected via easily available ports on the device. A range of compatible sensors are available, like depth measurement and water quality (Libelium 2014). Smart Water is

[8]Note that these are only assumptions based on a look at the stations. The statements have not been confirmed, nor denied, by Scanmatic

intended for both companies, cities/official institutes, as well as private customers. The price of the device ranges from 1,000 to 5,250 €, depending on the technical specifications. Also the size and weight varies slightly with the model. Figure 5 shows one of the standard Smart Water nodes. Without any sensors or solar panel attached, the weight of this node is 1000 grams, and the dimensions are 124 x 122 x 85 mm (Libelium 2018). This node would in other words be a lot more manageable to carry around in the forest than the Scanmatic node.

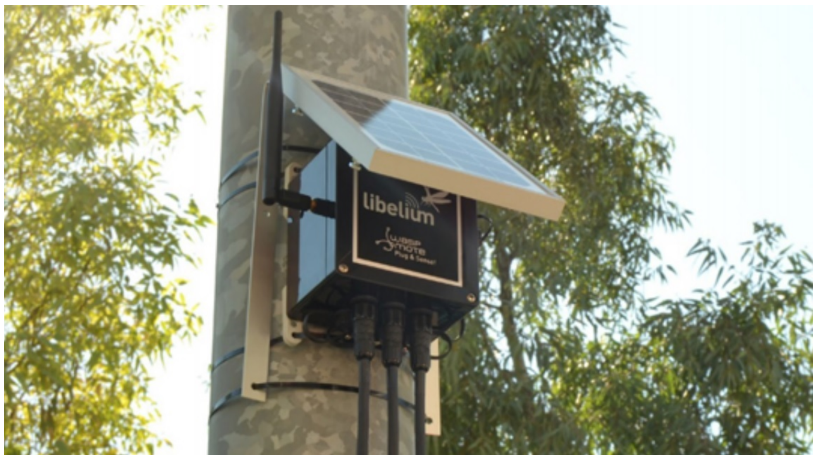


Figure 5: The Libelium Smart Water node together with the standard solar panel. Photo: Libelium (2019).

Power is drawn from batteries, which again can be charged by a solar panels. Communication happens via cellular network, radio drivers or WiFi, depending on the model. User interaction, like choosing functionality, happens via a computer. A USB cable, connected to one of the outside sockets, must be used with the first interaction (ibid.). Later interaction can be done via WiFi or 4G if the model supports this feature. It seems that one of the reasons why this product is so popular, is because of how user friendly it is. It's a "plug-and-play" product that requires little technical knowledge from the user. Nor does it seem to require hours of reading the instruction manual in order to understand how to operate it. This user friendliness is a key element to keep in mind later on in this product development project.

System I:
The single-node setup

4 Plan, overview and choice of microcontroller

This part of the report will go through the process of designing and creating the single-node system. Even though some requirements probably will change, and new ideas will be thought of as the project goes along, it's important to have a general overview of the challenges from the beginning. This way it will be easier to know where to begin and what precautions to take. The list below is based on AE's requests so far, and thus presents the current main goals for the system.

- Measure depths
- Measure temperature
- Do measurements every 15 minutes
- Have a user friendly interface
- Use little power
- Be small and light
- Send data back to AE via NB-IoT
- Focus on cost, as long as it doesn't interfere with the other goals

The parts chosen for the product should be as optimized for the job as possible. A good place to start is by choosing which microcontroller the system will use. This is a good starting point since all other units will be directly or indirectly in contact with the microcontroller.

4.1 Choice of microcontroller

As mentioned, AE have already had plans on exploring the possibilities of using NB-IoT for communication with remote sensors. They have therefore bought two development boards that has on-board NB-IoT modems. These boards have until now been unused. The name of these boards are SODAQ SARA AFF N211 (SODAQ 2018a). The SODAQ SARA AFF N211 (hereby referred to as *the SODAQ board*) is depicted in figure 6. In addition to having an on-board NB-IoT modem, the SODAQ board also includes a U.FL antenna connector, a microSIM slot, a Li-Ion battery charger and a GPS, among others.

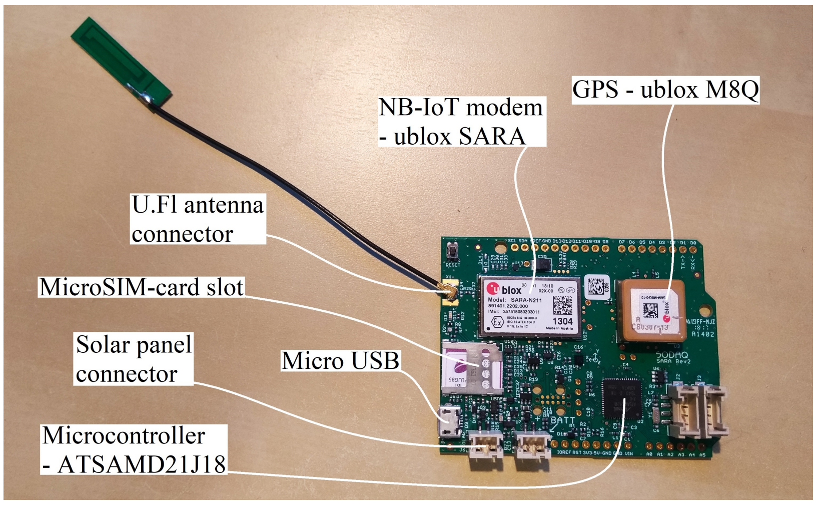


Figure 6: SODAQ SARA AFF N211 and some of the on-board components.

The SODAQ board uses the ARM ATSAMD21J18 microcontroller by Microchip Technology Inc. The ARM ATSAMD21J18 (hereby referred to as SAMD21) has a 32-bit processor, 256 kB flash memory, 32 kB SRAM[9], and has a maximum clock frequency of 48 MHz. For comparison, the AVR ATmega328P used in the specialization project is an 8-bit microcontroller. It has more modest specifications with 32 kB flash memory, 2 kB SRAM and a maximum clock frequency of 8 MHz. Similarly to the ATmega238P, the SAMD21 has integrated hardware for common communication protocols like UART, I^{2}C and SPI. On their website, SODAQ claims that this board has a power consumption of 26 µA while in low-power-mode (SODAQ 2018a).

The SODAQ board costs 930 NOK through the official web shop (SODAQ 2019). Although this price is higher than what an equivalent microcontroller costs by itself, the combined cost of all the components on the board wouldn't be that much lower if bought separately. The board seems to be of good quality, and the SAMD21 should be suited for the task. Since the SODAQ board seems like a good platform to build this system on, it will for now be chosen for this single-node system.

4.2 Choice of programming language

The SAMD21 microcontroller on the SODAQ board comes pre-loaded with what is called an Arduino Bootloader. This is a small script that runs on the microcontroller before entering the main program that allows it to be flashed with a new code via its serial line (using USB for example) instead of the six-wire[10] SPI bus which it's normally programmed through (Arduino 2019). The microcontroller also has associated board files in the Arduino Integrated Development Environment (IDE). This combination allows the SAMD21 to be programmed in the Arduino IDE using the Arduino C++ language. This language has the same syntax for all microcontrollers that has associated board files. So code written for one type of microcontroller can be used on another type of microcontroller since the board files can translate the common syntax into microcontroller specific syntax when being compiled. Since the Arduino C++ doesn't use microcontroller specific syntax, it's more abstract, but at the same time more intuitive, than the microcontroller specific syntax. The Arduino overlay is also a bit more demanding on the processor. An operation in the Arduino language usually uses more clock ticks than the same operation if written in the native language.

Whether the microcontroller should be programmed in native ARM-C or in the Arduino C++ language is not a straight forward question to answer. They both have their advantages and disadvantages. The Arduino language has it's strength in being easy to use, and the same code can run on other microcontrollers that are able to use the Arduino language. On the other hand, since the native languages of microcontrollers are specific to every model, they are therefore faster and less resource demanding than the Arduino language. It was because of the limited resources that it was decided to program in native AVR-C on the ATmega328P in the specialization project. The difference now is that the SAMD21 has much more resources than the ATmega328P, and in theory it should be more than enough for a program like the one to be written here. It must also be taken into consideration that the code might need to be adapted to new requirements after the end of this master's thesis. If so, it will be easier for AE to implement these changes themselves if the Arduino language is used. Because the Arduino language is more user friendly and more adaptable, and it likely won't have any downsides when running on the SAMD21 in this project, the Arduino C++ will be chosen as the programming language for the single-node system.

5 Power consumption

One of the goals for the system is that it should be able to use little power. Power consumption is something that should be thought of throughout the entire development process. The components must

[9]SRAM (Static Random Access Memory) is a type of RAM.

[10]The SPI protocol requires only four wires for communication, but it also need two wires for the power supply.

be chosen with respect to their power consumption in addition to fulfilling their primary function. The physical setup and wiring must be made so that it allows for as little current leakage as possible, and the code must be written to accommodate the low current leakage from the physical setup. The majority of components, especially the power demanding ones, should be active as little time as possible to lower the consumption. Lastly, the battery must be chosen to last the longest under the given conditions. One should preferably also be able to remotely monitor the systems battery level.

5.1 Design strategy for reduced current leakage in peripheral units

This subsection will look into some general design strategies to lower the power consumption of both peripheral units when not in use and the microcontroller itself. Even though a peripheral unit, for example a temperature sensor, has been given the command to power down or be inactive, current can still flow through it.

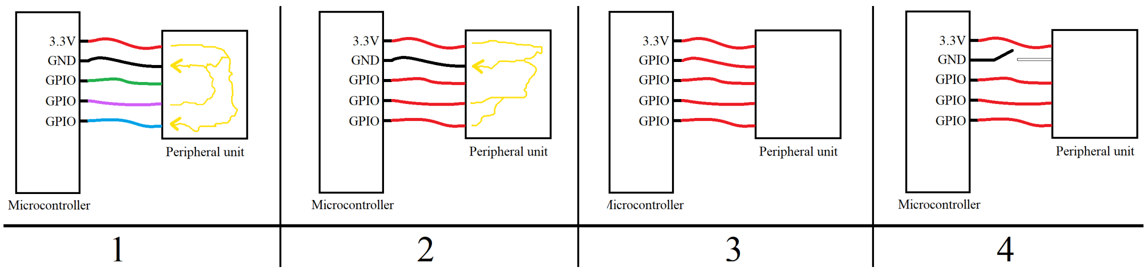


Figure 7: Different pin configurations allow for different amounts of leak current. Red wire indicates a high voltage, black is low voltage, while green, purple and blue are undefined or unknown voltage states.

Drawing 1 in figure 7 shows a microcontroller connected to a peripheral unit. The GPIO pins are left in an arbitrary state, marked with the green, purple and blue wires. This is the state the pins were in after the last bit of communication with the unit. The peripheral unit is inactive, but current still flows from some high voltage pins, via the peripheral unit, to some low voltage pins. Note that the path of the current in the figure is mostly drawn at random, and that not all pins will necessarily have current leakage. One way to gain more control is to set the GPIO pins of the microcontroller to a defined state. Many microcontrollers, including the SAMD21, have the possibility to set the GPIO pins to input in addition to pulling them high via an internal pull-up resistor (Microchip Technology Inc. 2018e). This state is often recommended for unused pins as it both prevents the pin from being left floating, and it protects the pin from short circuiting since its connected to high via a resistor, contrary to an output state who is connected directly to a high or a low logic level voltage. This scenario is shown in drawing 2 in figure 7. Even so, current can still flow from high to ground. A solution to this could be to use a GPIO pin as either the main voltage supply or as ground when active, and then set the pin to the inverse when not in use. The latter case is shown in drawing 3 in figure 7. Now there can be no leak current since all the pins have the same voltage level. Though what might be a problem here is if the peripheral unit requires more current than what a GPIO pin of the microcontroller can provide. The GPIOs of SAMD21 can supply and drain relatively little current compared to other microcontrollers. As it's only 7 mA, the previously presented solution will often not be applicable. If this is the case, another solution can be to completely cut the main voltage supply or the ground when the unit is not in use. This can for example be done using a MOSFET or a power switch. This scenario is shown in drawing 4 in figure 7. Using an electrically controlled switch will again introduce a new peripheral unit to the circuit, but these units are designed for this task and do therefore themselves have low leak currents.

Two general strategies for reducing the current consumption of the microcontroller itself is to shut down as many internal unused features as possible since they consume power, and to avoid leaving pins floating. Floating pins will cause an additional power consumption because they can pick up interference that affects other internal components of the microcontroller, causing them to consume unnecessary power (Microchip Technology Inc. 2018b).

5.2 Choice of battery and solar panel

The batteries used in most of the Scanmatic measuring stations that AE uses today are large 12 V batteries. Since on of the goals for the system is to be small and light, these batteries will not be considered here due to their weight and size. Several parameters must be taken into consideration when deciding what battery to choose for an application. For a long-lifetime operation like the one hoped to achieve here, self-discharge might be one of the most important factors to take into account. Self-discharge is the phenomena of batteries loosing more capacity than what the active drainage would imply (Simpson 2011). Primary lithium batteries are often considered the go-to battery for IoT applications. This is due to the superior weight to capacity ratio (University 2017), the close to constant voltage over discharge cycle (Energizer 2018a) as well as the low self-discharge rate (Stoehr 2012). They aren't very costly either, costing 19 NOK per 1.5 V 3500 mAh battery (BatteriOnline 2019).

One downside of primary lithium batteries is that they aren't rechargeable. As previously mentioned, the SODAQ board has an on-board charger which for example can be connected to a solar panel. The schematics of the board shows that it uses the MCP73831 charge management controller (Sikken 2018). According to the MCP73831's datasheet, it has a maximum charge current of 500 mA (Microchip Technology Inc. 2014a). Combined with for example a 1 W solar panel, a battery should theoretically be able to charge at 300 mA during direct sunlight. This solar panel can be bought from SODAQ's own website for 47 NOK. Before deciding whether to use rechargeable or non-rechargeable batteries, the solar panel should be tested to see what charging current it actually will provide in typical Norwegian conditions. The solar panel is bought together with a 6000 mAh, 3.7 V, 235 NOK, lithium ion polymer battery, which is compatible with the specifications of the charge management controller (ibid.).

In early February with clouded sky, without enough direct sunlight to cast any visible shadow, the solar panel was able to charge the battery through the SODAQ board with a current of 3.5 mA. Even though this is much lower than the theoretical maximum charge current of 300 mA, it shows that the solar panel and charger works even at poor conditions. Some calculations must be done in order to figure out how much improvements the solar panel and rechargeable battery would bring compared to the non-rechargeable primary lithium battery. Several assumptions must be made, so this is just a rough estimate. It will be assumed that the solar panel is subjected to poor charging conditions for five hours a day, as can be expected during the winter. The rest of the time it will not charge. This means that the system will receive 3.5 mA for 5 hours every 24 hours. It'll be assumed that the system uses 50 µA during sleep (since the SODAQ board uses 36 µA in sleep by itself). Lastly the system will be asleep for roughly 14.5 minutes every 15 minutes. During the 30 seconds of operation every 15 minutes, the system might be expected to use an average of 20 mA. So during 24 hours, the power consumption would be:

$$3.5mA \times 5hours - 0.05mA \times \frac{14.5minutes \times 4\frac{times}{hour}}{60\frac{minutes}{hour}} \times 24\frac{hours}{day} - 20mA \times \frac{0.5minutes \times 4\frac{times}{hour}}{60\frac{minutes}{hour}} \times 24\frac{hour}{day} = 0.34mAh$$

This means that even in abnormally bad charging conditions, given that the assumptions are correct, the batteries will be able to gain 0.34 mAh during 24 hours. If this is correct, the battery will never have to be replaced until the battery itself reaches the end of it's lifetime. Lithium ion polymer batteries are often said to have a cycle life of $\geq$300 charge/discharge cycles (Tracer Power 2019), although this isn't relevant here since the battery most of the time will be fully charged. Unfortunately there were no articles or datasheets to be found on the fully-charged lifetime of lithium ion polymer batteries.

The solar panel and lithium ion polymer battery combination seems like the best option since it provides an "unlimited" battery life. Further tests will be done later in the project to see if the previously stated assumptions, and thus the power consumption calculations, are correct. The battery can easily be changed later in the project. For now, the lithium ion polymer battery and the 1 W solar panel will be chosen as the power source.

5.3 Voltage measurement

In a remotely stationed, battery/solar powered instrument like the one being developed here, there is an interest of knowing the state of the battery. This way it's easy to see if and when the device needs power-related maintenance. The battery level can also indicate other problems with the device, like a sudden voltage drop due to a short circuit.

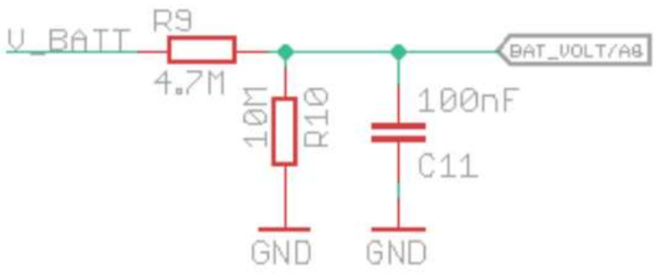


Figure 8: A voltage divider on the SODAQ board, used to lower the voltage from the battery so that the ADC can read the value. Image by Sikken (2018).

As can be seen from the cut-out of the SODAQ boards' schematics in figure 8, the battery voltage can indirectly be read from pin PB5 (here labeled BAT_VOLT/A8) of the microcontroller. The current from the battery flows through a voltage regulator, lowering the voltage from what ever voltage the battery has[11], down to a constant 3.3 V. A small part of the battery current also flows through the voltage divider in figure 8. The battery voltage is here lowered to a voltage that is usually lower than 3.3 V, and this voltage is in direct contact with pin A8. Pin A8 is connected to the ADC of the SAMD21. The ADC can read the voltage, and return the value as a number between $0 \leq X \leq 1023$. The maximum voltage the ADC can read is 3.3 V since this is the reference voltage. A voltage of 3.3 V on A8 means that V_BAT in figure 8 must be $3.3V \times \frac{10M\Omega + 4.6M\Omega}{10M\Omega} = 4.851V$. A returned value of 0 simply means 0 V on V_BAT. A general formula to interpret the returned values from the ADC can be written as:

$$V\_BAT = \frac{3.3V}{1023} \times \frac{R10 + R9}{R10} \times ADC\_VALUE$$

where ADC_VALUE is the value returned by the ADC, V_BAT is the battery voltage, and the values of R10 and R9 can be found in figure 8. In the code, this equation is implemented into the function getBatteryVoltage(), which returns the battery voltage in the unit mV.

[11]The SODAQ board uses the XC6220B311MR-G voltage regulator, which has a maximum input voltage of 6 V (TOREX SEMICONDUCTOR LTD. 2019).

5.4 Current measurement

During the development process it's useful to measure the direct current consumption of both the system as a whole, and of single components. The UNI-T UT33D multimeter will for the most part be used to do this measurement. One thing to note is that when trying to measure the current consumption of the SODAQ board by connecting the multimeter in series with the battery, the board often halts or resets itself. Even when setting the multimeter to "high current reading" (200 mA), the same thing would still happen. The reason for this is most likely because the SODAQ board at times, like at start-up, has a short sudden spike of current consumption (SODAQ 2018b). The multimeter might not be able to handle this sudden spike, and therefore blocks the needed current to the board. This results in the board not being able to boot properly, thus either halting or trying to reset itself. The solution to this problem is to short the two terminals of the multimeter against one another until the SODAQ board has booted properly. Then the two terminals can be separated and the current consumption can be measured.

6 NB-IoT connectivity and usage

The NB-IoT modem mounted on the SODAQ board is the SARA-N2 211 produced by u-blox. The modem is powered by setting the microcontroller's PA27 pin (called the SARA_ENABLE pin) high, which then activates a load switch connected to the modem. No wiring is necessary here since the modem is already integrated on the board. However, an antenna must be connected and a SIM card must be inserted. The antenna is connected to the u.fl. connector on the board. The telecommunications company Telia agreed to give away two NB-IoT SIM cards to be used in this project and the time afterwards. They did so because they themselves are interested in exploring the practical usage and possibilities of the relatively new NB-IoT technology.

6.1 Communication between the modem and the microcontroller

The SARA-N2 modem communicates with the microcontroller via the universal asynchronous receiver-transmitter (UART) protocol. UART is a common serial communication protocol that among others can be used to send data between the microcontroller and the serial monitor terminal window on the computer via USB. The SODAQ board's schematics shows that the serial ports of the modem is separate from the serial ports of the USB. In other words; both the modem and the serial monitor terminal window on the computer can be used at the same time. This will come in handy when writing code for the modem, and debugging it. When initiating the UART protocol on the microcontroller, a baud rate must be chosen. The baud rate must be agreed upon by both units, since there are no shared clock pulse line to synchronize the communication. Thereby the word *asynchronous* in UART. The modem supports several baud rates, but 9600 bps is the default baud rate, so it will be used here.

The SARA-N2 modem uses the same command set as GSM/GPRS modems and mobile phones, namely the AT commands. AT commands are ASCII strings that are sent to the modem, where the modem interprets the commands and performs the appropriate actions. Every command starts with the 'AT' prefix. For example to make the modem reboot, the command *AT+NRB* is used. The command *AT+NUESTATS* makes the modem reply with a string containing connection statistics. The complete list of commands can be found in the modem's command manual (u-blox 2018).

6.2 Connecting to Telia's network

Different telecommunications companies have different settings that needs to be configured on the modems in order to make them connect to their network. No official information from Telia were found online regarding what settings they require. However it seems that another person have contacted Telia with the same question, and have posted the answer he received on a forum (Kim 2018). These settings will be tested before alternatively contacting Telia personally if they don't work. The following commands will be sent to the modem in order to change the settings to try to connect with Telia's network:

```
AT+NCONFIG: "AUTOCONNECT","TRUE"
AT+NCONFIG: "CR_0354_0338_SCRAMBLING","TRUE"
AT+NCONFIG: "CR_0859_SI_AVOID","TRUE"
AT+NCONFIG: "COMBINE_ATTACH","FALSE"
AT+NCONFIG: "CELL_RESELECTION","FALSE"
AT+NCONFIG: "ENABLE_BIP","FALSE"
```

In addition, the command manual specifies that the following setting also should be changed to always remain enabled (u-blox 2018):

```
AT+NCONFIG: "NAS_SIM_POWER_SAVING_ENABLE","TRUE"
```

These commands are put into the function configureModemSettingsTelia() in order to not having to do this configuration manually for every new modem. Once the settings have been configured, they are stored in NVRAM, a non-volatile memory on the modem. This means that they only need to be entered once during the modems lifetime.

The settings seems to have worked to some degree. In Trondheim the connectivity has for the most part been good. Usually contact with the network is established within 25 seconds. Although at times, roughly once every 10 connections, the duration increases to about 120 seconds. And a few times the registration takes more than two hours. For the final product, this amount of time is not acceptable. So hopefully this can be improved upon. In Setesdalen (Øst-Agder, Norway), one of the places where the final product will be deployed, the situation is a lot worse. During a four day visit there, connection was established only once, and it had the Received signal strength indication (RSSI) of only -101 dBm. This is quite critical since this depth measuring system is supposed to be used in this area. Due to the seriousness of this issue, Telia is contacted directly. Telia confirms that the previously configured settings are correct, and they don't know what might have caused this issue. The area in Setesdalen should have good coverage, and there have been no other reports of the network being down. Both parties will continue to look into this issue.

6.3 Sending data and server selection

The SARA-N2 modem sends data via the NB-IoT network using the user datagram protocol (UDP). The transmission control protocol (TCP) is not supported on these modems. Although the plan is to eventually be able to check if the message have been received at the other end, having a UPD-only modem is not a problem. The server can be programmed to reply to the sender after a successful reception. If the node then receives this acknowledgement, it can continue it's operation. Though if no acknowledgement is received before the timeout, the node will try to send the message once again. This will work sort of like a simulated TCP. For now, only the basic UDP procedure will be implemented. The first step in sending a package using the modem, is to open a UDP socket. This is done by sending the following AT command:

```
AT+NSOCR="DGRAM",17,42000,0
```

DGRAM is the socket type and 17 is the standard internet protocol definition value for UDP. These are the only two allowed parameters here for this modem. 42000 is the listen port when receiving a message. 42000 is the default value, but it can have any value from 0 to 65535 except for 5683. The last parameter is the receive control, where a 0 will ignore incoming messages, while a 1 will trigger a *received message indication*. After this command is sent, the modem will return a socket identifier number. This number must be referenced later in order to use the opened socket. To send data, the following AT command is used:

AT+NSOST=[socket identifier],"[receiver IPv4 address]",[receiver port],[data size in bytes],"[data in hex format]"

The square brackets and their contents are to be replaced with numbers in dec format, except for the data itself, which must be in hexadecimal format. Both the IP address and the data are passed on to the modem as character arrays. This is denoted by the quotation marks. The modem will return a message indicating whether the message was successfully sent or not. When all the messages have been sent, the socket can be closed using the following command:

AT+NSOCL=[socket identifier]

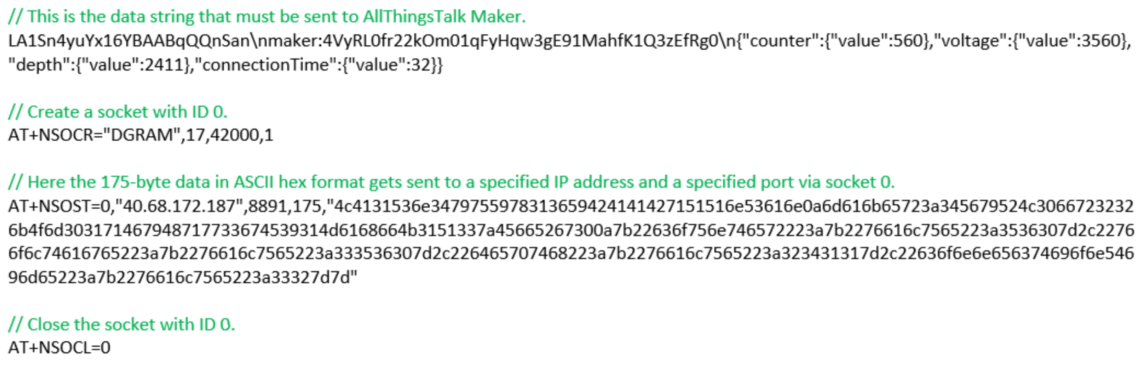
The data from the node needs to be sent somewhere it can be monitored by AE. The plan is to eventually send the data into AE's SCADA system. This is where most of AE's data is gathered (Bjønnes 2019). But due to the high sensitivity of the data in AE's SCADA system, it's very restrictive which devices that gets to send data here. It would take a lot of time before being able to monitor the data stream from the device if the SCADA system is to be used during the development. Because of this, the SCADA system will not be used for now.

There are several other servers where the data can be sent. One option is to make a Python server on a computer or Raspberry Pi. The downside of doing this is that the server would always need to be running (while testing the sensor node). The other problem is that private internet service provider subscriptions often has dynamic IP addresses, as is currently the case. This means that the IP address changes from time to time. So for long term testing, this is not a good solution. Another alternative is to use Azure, a cloud based computing platform from Microsoft (Microsoft 2019). Though the problem with Azure is that it has poor UDP support, and as mentioned, the SARA-N2 modems only supports UDP. Another alternative server is one that can be accessed via a site called AllThingsTalk Maker. AllThingsTalk Maker is an IoT cloud that receives data from an IoT device, and the data can be displayed in graphs or monitored in a live feed. This service has UDP support and is free for an unlimited amount of time. AllThingsTalk Maker will therefore be used as a cloud server during the development process.

Five parameters are needed to send data to AllThingsTalk. The first two are the IP address and the port used by AllThingsTalk Maker. These are 40.68.172.187 and 8891 respectively (AllThingsTalk 2019). To identify which device the data belongs to, each device has a unique device ID and a device token. The format looks as presented below:

Device ID: LA1Sn4yuYx16YBAABqQQnSan
Device Token: maker:4VyRL0fr22kOm01qFyHqw3gE91MahfK1Q3zEfRg0

These are the example device ID and device token given on their UDP information page. The real device ID and device token can be found on the 'Settings' page after creating an account. In addition to the four parameters mentioned so far, the message must also contain the exact asset name that has been entered on the device page of AllThingsTalk Maker. The full UDP payload for a device sending *counter = 560*, *voltage = 3560 [mV]*, *depth = 2411 [mm]* and *connectionTime = 32 [s]* would look like this:



```
// This is the data string that must be sent to AllThingsTalk Maker.
LA1Sn4yuYx16YBAABqQQnSan\nmaker:4VyRL0fr22kOm01qFyHqw3gE91MahfK1Q3zEfRg0\n{"counter":{"value":560},"voltage":{"value":3560},
"depth":{"value":2411},"connectionTime":{"value":32}}

// Create a socket with ID 0.
AT+NSOCR="DGRAM",17,42000,1

// Here the 175-byte data in ASCII hex format gets sent to a specified IP address and a specified port via socket 0.
AT+NSOST=0,"40.68.172.187",8891,175,"4c4131536e347975597831365942414142715151 6e53616e0a6d616b65723a345679524c3066723232
6b4f6d30317146794871773367453931 4d6168664b3151337a45665267300a7b22636f756e746572223a7b2276616c7565223a3536307d2c2276
6f6c74616765223a7b2276616c7565223a333536307d2c226465707468223a7b2276616c7565223a323431317d2c22636f6e6e656374696f6e546
96d65223a7b2276616c7565223a33327d7d"

// Close the socket with ID 0.
AT+NSOCL=0
```

The data is converted into a string where each single letter and each single digit of the numbers are represented by its ASCII character in hexadecimal format. For example the number 2411 would become "32343131". The number 2 in ASCII format is represented by the dec number 50, which again is represented by the number 32 in hex. The only exception to this rule is the '\n' part. This must be converted to 0a, which is hex for the ASCII representation of '\n', instead of 5c6e where 5c = \and 6e = n.

Several support functions must be written in the code in order to create one single flexible function which will take care of the sending. It's important that this part of the code is user friendly since AE themselves later will use this part of the code when entering the device ID and device token for new devices. All the asset names can now be entered as defines on the top of the code, and the values of the assets can be entered as arguments into the function sendToAllThingsTalk(). For example like this:

```
#define DeviceID "My5sVHzEqwUYcit057T3DoRu"
#define DeviceToken "maker:4IxXLoJVNIZzG0lqFyyQSHtmNOs3ZSyfO2bDM3Q1"
#define Asset1 "temperature"
#define Asset2 "date"
sendToAllThingsTalk(temperatureValue, dateToday);
```

Up to 10 assets can be created per device as the code is now, but this can easily be expanded. The arguments of the function has the type uint16_t, in other words maximum a 16 bit number. As of now, this function sends data in a format that is specific to AllThingsTalk Maker. The plan is to later on in this project make a general purpose function which sends data in a standard format; for example JSON. The string sent to AllThingsTalk Maker is in fact in a JSON format, all except the prefix.

7 Depth measurement

The sensor AE wants to use is the OTT PLS (Bjønnes 2019). This is the same sensor that they are using today at many of their stations. OTT PLS is a digital pressure level sensor. The sensor measures both water pressure and atmospheric pressure in order to determine the true depth of the sensor probe. Air pressure is measured via a tube that goes from the sensor probe, through the cable together with the rest of the wires, up to the microcontroller.

7.1 Powering the sensor - Boost converter

The OTT PLS was also used in the specialization project. Then the sensor was powered by a separate battery package consisting of eight 1.5 V AA batteries in series. The reason the depth sensor was

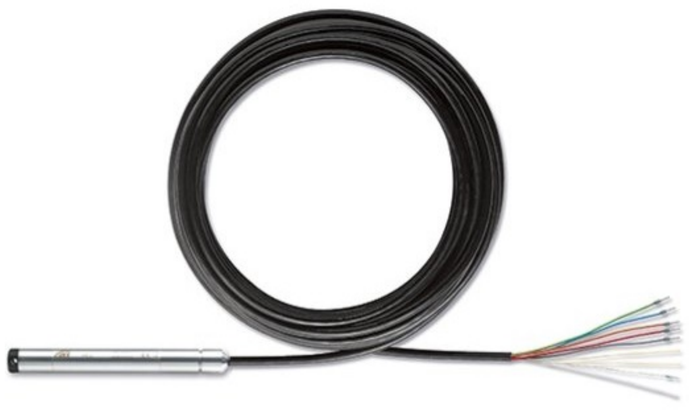


Figure 9: The OTT PLS depth sensor. Photo: OTT HydroMet (2018).

powered using its own battery package is because it requires higher voltage than the rest of the system. The sensor's datasheet specifies an accepted operating voltage range from 9.8 V to 28 V DC. All other sensors used only 3.3 to 5 volts. There are some disadvantages to using a separate high voltage battery package for a system like this. The main disadvantage is that it causes extra expense, extra weight and requires extra space. In the system being developed here, the batteries will hopefully never be completely discharged since the batteries are connected to a solar panel. The problem if the extra high-voltage batteries are to be included, is that they will not be able to charge and will thus eventually discharge. The charge management controller used in the SODAQ board, the MCP73831T-2ACI, can charge batteries up to a maximum of 4.232 V (Microchip Technology Inc. 2014b). Although simple calculations estimates the lifetime of the high voltage batteries to be about 15 years[12], it's still an extra factor to keep an eye out for. Lastly, some extra components would be needed to separate the high and low voltage levels from each other.

One way to avoid having to include an extra high-voltage battery package is by using a step-up converter (also known as a boost or buck converter). A step-up converter can convert a lower voltage to a higher voltage. So in this case, it can take the 3.3 V used in the rest of the system and step it up to a voltage in the desired range. This method will be tested here. If successful, it will also be implemented. The step-up converter MT3608 is chosen for this project due to it's power efficient design (Xi' an Aerosemi Technology Co. 2018). MT3608 has a soft-start feature, meaning it prevents a large inrush current at start-up, and thus reduces it's power consumption (RICOH Electronic 2018). This converter can take input voltages from 2 V to 24 V, and step them up to 28 V at a maximum of 2000 mA. According to the efficiency curves in the step-up converters datasheet, the converter should have a total efficiency of about 91% with the voltage and current drain measured for the sensor. The depth sensor will get an input voltage of 14 V even though it can run on as little as 9.8 V. This is to have a buffer margin to the lower limit. It will be powered by the constant 3.3 V instead of directly from the battery voltage in order to get a constant output voltage.

[12]The sensor is measured to use an average of 2.5 mA while active. The active time per measurement is 8.3 seconds. Measurements are taken every 15 minutes. The Energizer Ultimate Lithium batteries used in the spezialization project has a capacity of roughly 3500 mAh (Energizer 2018b). The batteries also looses about 10 % of their capacity after 10 yeas due to temperature effects and self-discharge.

7.2 Wiring and code

OTT PLS uses a communication protocol called SDI-12. This protocol allows for bidirectional communication via one single wire. The official SDI-12 specification documents states that the bidirectional three-state data line has a binary state of 1 if the voltage is between -0.5 and 1 V, a binary state of 0 if the voltage is between 3.5 and 5.5 V, and is undefined between these two intervals. This could be a problem for two reasons. The SAMD21 is a 3.3 V logic level microcontroller. So the highest output voltage from the microcontroller is only 3.3 V. According to the official SDI-12 documents, this is interpreted as an undefined state. The other problem is that the SDI-12 protocol allows for high voltage levels up to 5.5 V. A voltage this high could damage the microcontroller, since it has a maximum operating voltage of 3.63 V (Microchip Technology Inc. 2018e). No further information are to be found on using a 3.3 V logic with the OTT PLS in the datasheet, nor on the producers website. Some forums however say that many SDI-12 sensors actually are compatible with 3.3 V microcontrollers. Forum-posts are quite useful and are often correct, but they aren't considered reliable sources. Both of these potential problems must therefore be tested for. The wiring of the sensor is quite simple, as shown in figure 10. Any GPIO pin can be used as the data pin. A MOSFET is used to completely cut the power to both the OTT PLS and the step-up converter when they're not in use.

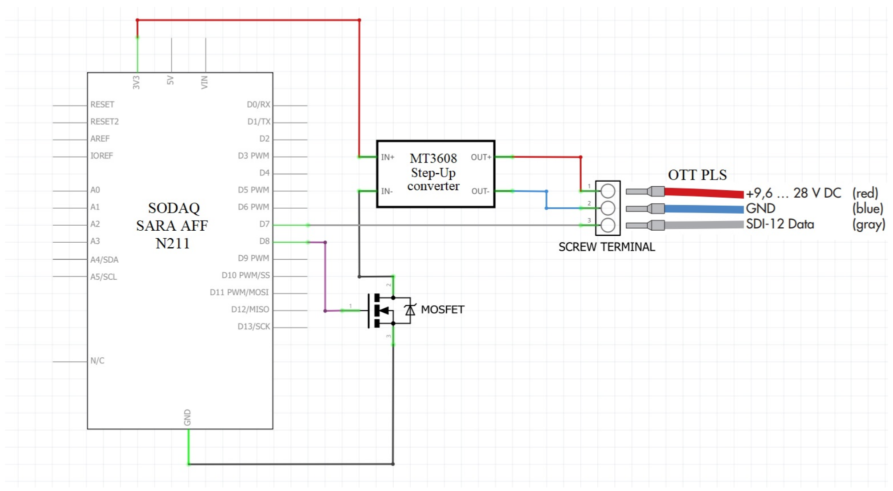


Figure 10: Wiring of the OTT PLS depth sensor.

The SDI-12 protocol works a lot like the UART protocol. The biggest difference is that it uses only a single wire for communication instead of having dedicated transmit and receive channels. The SAMD21 microcontroller doesn't have dedicated hardware for SDI-12 communication. Instead the protocol must be implemented via code. Writing communication protocols from scratch can take a lot of work. And protocols are, as the name implies, a standard. So writing one from scratch would most likely mean doing the exact same work as someone else have done before. Therefore it's more time effective to reuse somebody else's code if the licence allows it. A library called *Arduino-SDI-12*, written by Kevin M. Smith, is available on GitHub (Smith 2014). This library is distributed under the GNU Lesser Public License (LGPL 2.1), and can thus be used for the means and purposes of this project (Free Software Foundation Inc. 2018).

Similarly as with the u-blox modem, communication here works by sending a command to the sensor and then listen for a reply. The OTT PLS' datasheet lists the available commands. Before sending any commands, the sensor needs to power up. This power-up takes five seconds according to the datasheet. So a 5000 ms delay is added after switching on the step-ups' MOSFET. The majority of the code for the depth sensor is put into a function called getDepth(). First the string *?M!* is sent. This tells the OTT PLS to start doing the measurements. The first character in the command string should be the sensor address. Addressing is useful if multiple SDI-12 sensors are connected to the same pin on the same microcontroller. Though by using "*?*" instead of the address, all active sensors responds to the command. Here, where there is only one SDI-12 sensor connected to the pin, there's no need to bother with addresses. Each OTT PLS unit can have different addresses, so by using "*?*", they don't have to be put into the code manually for each sensor. The response from the sensor after being given this command contains the time in seconds that the sensor needs to complete the measurements. These sensors say that they need two seconds to do the measurements. Unfortunately the resolution of the time is in seconds, so it's hard to say whether it only needs 2 seconds, or 2.4 seconds, or even 2.9 seconds if it doesn't round up. The datasheet doesn't address this issue any further either. Most likely it only needs two seconds. And later performed tests shows that giving a delay of two seconds is enough to complete the measurements. Though just to be sure that any problem won't come up later, a delay of 3000 ms is added after sending the command. As long as the extra power consumption or the extra duration due to the added second doesn't become a problem, it will stay this way. The next command sent to the sensor is *?D!*, which tells the sensor to send the results from it's measurements to the microcontroller.

Before running the code on the SAMD21, the same setup is made with an ATmega328P running on 3.3 V. The 328P has a maximum operating voltage of 6 V (Microchip Technology Inc. 2018a). The voltage will be measured on the bidirectional data line to see if the voltage surpasses the maximum operating voltage of the SAMD21. Luckily the voltage seems to stay below 3.6 V the whole time. That means that it's safe to use the sensor with the SAMD21 without including extra components like logic level shifters. The reply from the sensor to the microcontroller looks like this:

$$20022$$

$$2 + 0.105 + 20.8$$

It seems that the 3.3 V communication logic from the SAMD21 microcontroller to the sensor works fine as well since the sensor responds correctly. The whole reply is stored in a character array, but only the '0.105' are of interest since that's the depth (in millimeters)[13]. The output format is fixed, therefore the depth value can be extracted by knowing their position in the character array. The ASCII number is then converted to a uint16_t dec number, and then returned by the getDepth() function. This is the raw depth value of the sensor probe. Depth calibration will be implemented after the implementation of the user interface in chapter 12.

8 Sleep and time keeping

The plan is to have the node take measurements and send the data back to AE once every 15 minutes. In between these measurements, the node will enter a low power state called *sleep mode*. This chapter will look at the procedure of entering sleep mode, as well as how to wake up from sleep at the desired intervals.

8.1 Sleep mode

SAMD21 has two types of sleep mode; one called ide mode and one called standby mode. Standby mode uses the least power, so that's the one that will be implemented here. Page 147 of SAMD21's datasheet shows the steps of putting the microcontroller in standby mode (Microchip Technology Inc.

[13]What the rest of the numbers mean can be found in the OTT PLS' datasheet.

2018d). The system control register of the microcontroller contains a so called *SLEEPDEEP* bit. This bit controls which sleep mode the microcontroller will use. Standby mode is configured by setting this bit to 1. Now standby mode can be initiated by using the WFI (Wait For Interrupt) command. The microcontrollers then enters sleep mode and can only be wakened by a interrupt or a reset (ARM 2007). The code looks like this:

```
SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;
__WFI();
```

All clock sources except for a select few are stopped in order to save power when entering standby mode. Also the voltage regulator is running in a low-power mode. This means that the user must make sure that as many peripherals and clocks as possible are stopped before entering sleep mode in order not to overload the regulator (Microchip Technology Inc. 2018d).

8.2 Internal Real Time Clock

The microcontroller is now asleep after executing the previous commands. Now the next step is to wake the microcontroller at the right time. The SAMD21 makes it easy to keep track of the time, as well as providing interrupt signal at the desired times. Most microcontrollers must rely on an external real time clock (RTC) for timekeeping long term timekeeping. The SAMD21 on the other hand has an integrated internal RTC. In addition to keeping track of the time, the RTC can send interrupt signals, and it has a pre-configured calendar.

Setting the time, setting alarms and reading the time is done by accessing registers and performe read/write actions. Since the SODAQ uses the Arduino bootloader and uses the same microcontroller as the Arduino Zero, some convenient functions for accessing these specific registers are already written. These functions are accessible by including the *RTCZero* header file. A function called setAlarmNext15Minute() is created. This function will first check the current minute and hour value, then it will set the alarm for the next whole 15 minutes (i.e. if the time is now 09:18, the alarm will be set at 09:30). It will then enable the alarm upon the next match of the set minute, hour and second value. Lastly it will attach an interrupt to this alarm. This function is the last one to be called in the event loop before the microcontroller goes to sleep. The event loop is the part of the code that will run over and over again indefinitely. When the interrupt is triggered, the interrupt service routine will disable the alarm before allowing the microcontroller to continue its operations.

8.3 Updating the time and time drifting

The microcontroller can use the SARA-N2 modem to acquire the time and date, and then set the time of the RTC thereafter. The modem returns the time and date when the *AT+CCLK?* command is sent, if it has a good enough connection to the network. All subroutines of acquiring and setting the time is put into a function called getTheTime();. This function will actually serve two purposes. One of them, as the name partially implies, is to get and set the current time and date. Secondly it will be used to check the network connection status. Although the modem says that it is connected to the network when performing a signal quality check (*AT+CSQ*), it has shown to sometimes take a couple of seconds more before the modem is ready to send and receive messages. Since the modem needs to receive a message before having the time and date available, it can therefore be used as an indication for when the modem has a true contact to the network. The function will be called once for each run of the event loop, i.e. once every 15 minutes. Only once the updated time has been acquired, the modem will try to send the data package containing the sensor values.

The way the function works now is by looping indefinitely until the time/date have been updated. Eventually the plan is to have a timeout on this function so that it doesn't try to register to the network for several days straight if the network is down. Instead it should then timeout, save the unsent package, and try to resend on the next run of the event loop. Though for now it's interesting to see how long time it actually will use to get a signal, and to see how the connection time varies throughout the day and week.

One of the downsides of the internal RTC of the SAMD21 compared to other external RTCs, is that the measured time has an increased tendency to drift. The datasheet doesn't mention the accuracy of the clock, but a simple test shows that the drifting is abut 0.17 seconds per hour. That's equivalent to about 47 parts per million. Since the plan is to update the time once every 15 minutes, this isn't really a problem for this depth measuring system.

9 First early stage long duration test

From February 21st to March 5th, an early stage, long duration test will be performed. The goal of this almost two week long test is to analyze several aspects of the system. Will the system run into a glitch and stop working after a while? Will it do the right measurements? Will it be able to send all the data? How will the network connection time vary? How much will the battery be drained?

9.1 Setup

The test is performed indoor in a room with low humidity and a relatively constant temperature, so there is no need to protect the electronics. Most of the wiring is similar to the one in figure 10 since the depth sensor is the only peripheral so far. The only difference is that a battery now is connected to the V_{IN} and GND of the SODAQ board. Every 15 minutes, measurements will be taken and data will be sent to AllThingsTalk Maker.

Four parameters are to be measured: Depth, connection time, voltage and the accumulated numbers of times data have been sent. The depth sensor is placed in a small container filled with water, with an initial depth of 102 mm. The container is left untouched for the duration of the test, so the depth is expected to decrease slowly as the water naturally evaporates. The connection time is the time it takes for the u-blox modem to connect to a base station and acquire the time/date. The voltage should have been the voltage of the battery mentioned in chapter 5.2. Unfortunately this rechargeable 3.7 V lithium ion polymer battery was unavailable at the beginning of this test. Instead three 1.5 V primary lithium batteries in series are used. A multimeter is used to find that this battery package have an initial voltage of 5.16 V. This voltage, even through the voltage divider, is higher than the maximum voltage the ADC can measure, so the microcontroller will interpret it as a constant 4.85 V until it goes below 4.85 V (see chapter 5.3). A multimeter must therefore be used at the end to see how the battery voltage have changed. It's important to note that this voltage won't damage the microcontroller since the voltage delivered to the sensor pin via the voltage divider is $5.16V \times \frac{10M\Omega}{14.7M\Omega} = 3.51V$, which is lower than the maximum input voltage (Microchip Technology Inc. 2018e). Lastly, the counter counts how many times the data has been sent to the server. At power-up/reset of the system, the number starts at 1 with the first transmission. This counter will show if the system for some reason has reset itself, or if packages have been sent but not received.

9.2 Results and analysis of the test

After 13 days, the results can now be analyzed. It seems that the microcontroller as been running continuous without encountering glitches causing it to freeze. The graph from AllThingsTalk Maker in figure 11 shows the data from one week of the test. The counter started at 1 with the first upload and ended at 1155 at the last upload, before the power was deliberately cut. This shows that the counter, and thus the system, has not reset itself during the test. When zooming in on the graph and scrolling through the days, one can see that the counter has incremented by one every time, which means that all the uploads have been successful.

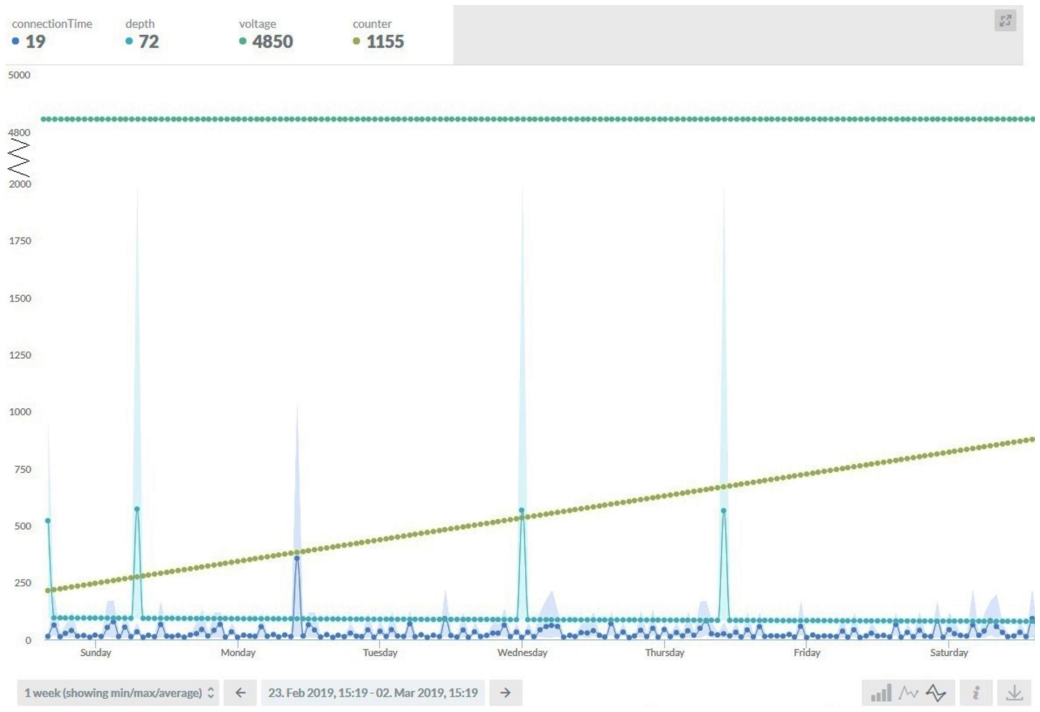


Figure 11: The graph from AllThingsTalk Maker showing seven days of the test. Each dot shows the data averaged over one hour, while the colored area shows the minimum and maximum value.

As expected, the voltage measurement is shown to be stable, with a constant reading of 4.850 V. As mentioned, this abnormally stable measured voltage is because of the battery being at a higher voltage than what the microcontroller can differentiate between. The voltage now measures 5.15 V using the same multimeter as earlier. Due to the flat voltage discharge curve of the primary lithium batteries (Energizer 2018b), it's hard to say how many mAh that has been used. At least it shows that the batteries haven't been drastically discharged.

For the most part the connection time seems to be between 8 to 23 seconds. Though there are two interesting characteristics here. Firstly there seems to be a pattern in how long it takes to connect to the network throughout the day. The time seems to repeat in a "wave pattern". A 24 hour graph of this pattern is shown in figure 12. There are no clear indications as to when the time is the shortest and the longest, since the "wave" is shifted from day to day. The other interesting characteristic is that the connection time is usually between 8 to 23 seconds, 60 to 80 seconds or 118 to 130 seconds. And sometimes up around 170 to 180 seconds. The connection time is rarely outside of these intervals. One might wonder if it's the timer keeping track of the connection time that might have a bug, making it show numbers only in these intervals. One of few exceptions is when the connection time reached it's highest value of 1038 seconds. According to private conversations with Telia, the non-roaming[14] connection time should never be more than 60 seconds. The long connection times experienced here are therefore most likely a bug that should be fixed.

[14]When the device isn't moving.

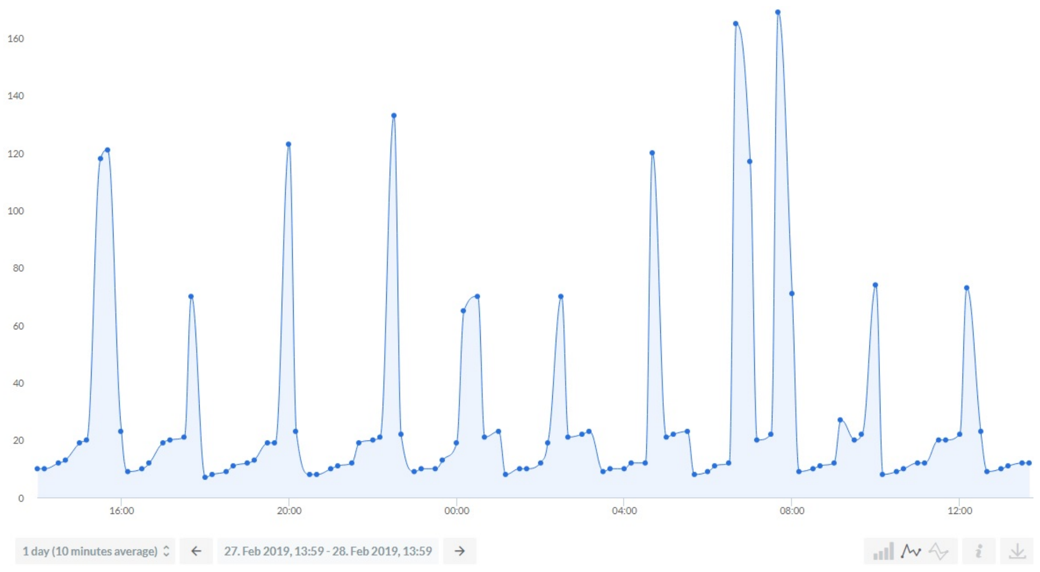


Figure 12: The repeating wave pattern of the connection time.

The data from the depth sensor were for the most part stable, slowly decreasing from 102 mm to 72 mm. But as can be seen in the graph in figure 11, there are some major deviations. These deviations displays the depth as a lot deeper than it should be. When studying the graph, two different patterns can be seen. The depth is either a little more than 2000 mm, or it's ten times plus two more than the correct number. In the first case, if the true depth for example is 100 mm, the false data is 2010 mm. If the correct depth is 75 mm, the false data shows 2007 mm. In the latter case, if the true depth is 68 mm, the false data would be 682 mm. These false measurements happened 11 times during the 13-day test. In other words in about 1 out of 100 measurements. False depth data is not acceptable in a system like this where the primary purpose is to report back on this very measurement. This issue must therefore be fixed.

10 Resolving the problems from the test

The two issues discovered in the test, being the occasional long connection time and the occasional false depth data, must be fixed. The depth data will be looked into first since it's the most critical.

10.1 Depth data

A potential cause of the data corruption is that the boost converter causes interference in the data line of the depth sensor. Figure 13 shows a simplified schematics of a boost converter. A boost converter works by rapidly switching on and off a transistor (S). Current flows through the transistor when its base is saturated, which means that current also flows through the inductor (L) making it build up an electric field. Then when no current flows through the transistor, the magnetic field collapses, inducing a lesser current with a higher voltage to flow through the diode (D) before the energy gets stored in the capacitor (C). The diode is there to prevent the current from flowing back when the transistor opens again. For each cycle of the transistor turning on and off, the capacitor builds up more and more charge. The rapid fluctuating magnetic field created by the coil can induce unwanted currents in nearby wires. This unwanted induced current might interfere with the data line to such extent that the data gets corrupted.

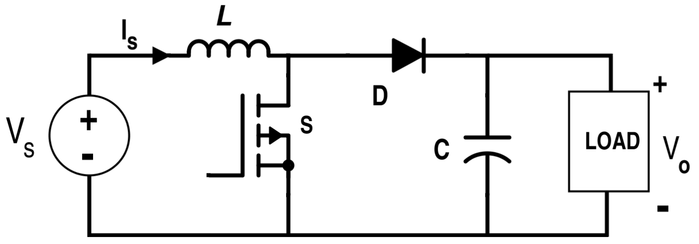


Figure 13: Schematics of a basic boost converter. Figure is taken from Electrical Engineering Stack Exchange (Granger 2016)

The hypothesis that the step-up converter causes interference in the data line of the depth sensor can be tested using an oscilloscope. If the hypothesis is correct, noise in the same frequency as the step-up's switching frequency should be detectable on top of the 1200 bits per second communication of the depth sensor. The MT3608 step-up converter has a constant switching frequency of 1.2 MHz. The first test, shown in figure 14, is where the signal wire gives a constant output of 0 V. Half way through, the step-up is activated. The wire is as close to the coil of the step-up as possible. The chart shows a significant increase in noise. So yes, the coil does cause interference at very close range. The interference shows to have a peak-to-peak of approximately 0.9 V.

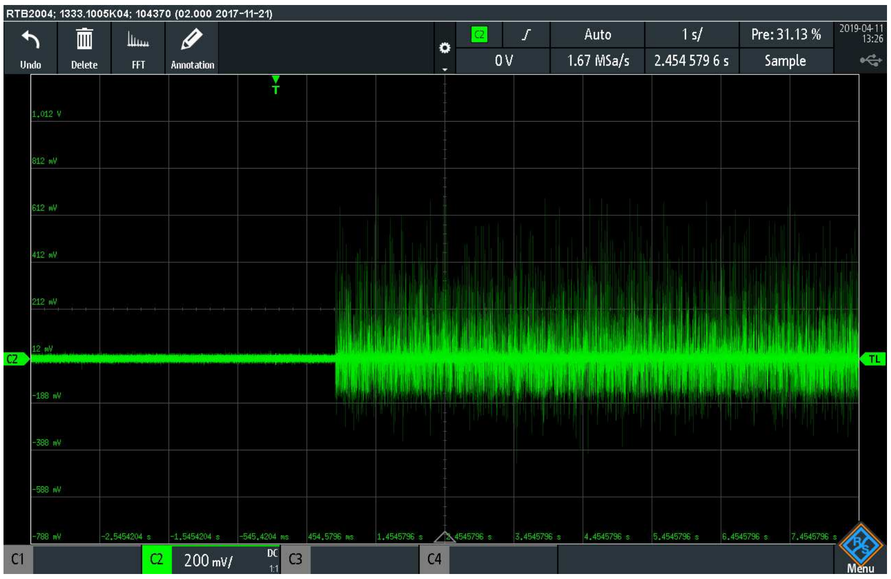


Figure 14: A constant low signal being distorted by the step-up converter. The oscilloscope used here is the Rohde & Schwarz RTB2004.

However when moving the wire away from the coil, the amplitude of the noise rapidly decreases. The noise is no longer detectable at a distance of 5 mm. Figure 15 shows the actual communication between the microcontroller and the depth sensor. Here the data wire is at a distance of approximately 5 mm away from the coil. No noise from the step-up converter can be seen. In the setup of the 13-day test, the distance between the wires and the coil were more than 5 mm, so it seems that this is not the cause of the issue. So a solution to the problem needs to be found somewhere else. Though it should be noted for later that all wires must have a safe distance from the coil.

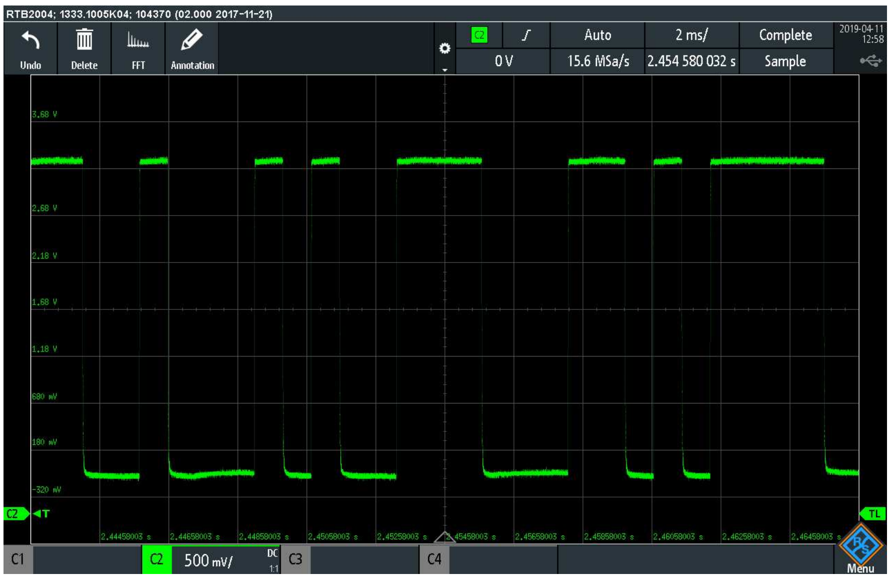


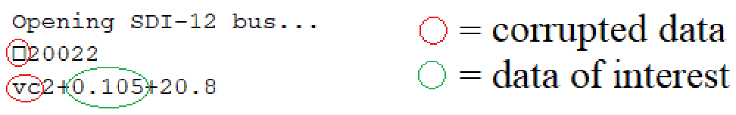
Figure 15: Communication between the microcontroller and the OTT PLS depth sensor. The wires have a distance of approximately 5 mm from coil of the step-up converter. No noise is detected.

Another possible cause of the problem might be that the SAMD21 microcontroller is a 3.3 V microcontroller, while the SDI-12 protocol should be used with a 5 V logic (SDI-12 Support Group Technical Committee 2017). If this is the problem, a logic level shifter can be used to convert the data signal in between these two voltage levels. To test this hypothesis, the setup from the 13-day test is used, except that the ATmega328P running on 5 V has replaced the SAMD21. Now the microcontroller uses the same logic level as the one recommended by the official SDI-12 documents. Unfortunately the problem still occurs at about the same rate.

The same setup as in the 13-day test was made once again, except this time the measurements are to be taken once every 16 seconds, and the full response string from the sensor are to be printed to the terminal window of the computer. After doing 1000 measurements, 14 responses were corrupted. Figure 16 shows a typical response with corrupted data[15]. It seems that the corrupted response has either added or removed characters from the supposed-to-be fixed-length response string. In figure 16, three extra characters (or symbols) has been added. The position of the depth data in the string has been shifted, so reading the data from the position where the depth was supposed to be gives the wrong output. The positive thing is that the corruption only happens at the beginning of each line. The raw depth data, as well as the two "+/-" signs, never shows to get corrupted. This means that the data still can be extracted even when the beginning of the string is corrupted. The code just needs to be improved upon.

A new and improved getDepth() function is written, that works by exploiting a few characteristics of the response string. If there are two, and only two "+" or "-" signs, and if the distance between these signs are six characters, then extract and convert the characters at position 1, 3, 4 and 5 away from the first "+" or "-" sign. If some of these criteria are not met, then the function shall return the number 50000. This is a value that is out of range for the depth sensor itself. So if this number is

[15]The string *"Opening SDI-12 bus..."* generated by the microcontroller, not the sensor.



```
Opening SDI-12 bus...
□20022
vc2+0.105+20.8
```

Figure 16: Typical sample of corrupted data output from the depth sensor. Here the microcontroller has read more characters than it was supposed to do.

returned, then the viewer of the data knows not to trust that specific measurement. Now the function can return the correct depth even if the position of the depth data in the string has changed. After flashing the new code and running 1000 new measurements, the function never returns a false value. Although the root cause of the problem haven't been solved, the practical issues of the problem have been fixed.

10.2 Connectivity

The problem with the connectivity is that the connection time is varying more than it should. At times it goes far beyond the 30-60 seconds that Telia said is normal. This strange behaviour might have correlations with the long roaming time when connecting to a new base station, and the abnormally bad connection in Setesdalen.

After doing some research online, it was found that u-blox have released a new firmware update for their N211 modems. The update specifications reads as follows: "*Faster network registration using a roaming SIM. The first time module attempts a network registration in "automatic operator selection" mode (AT+COPS=0), the registration procedure is completed in 1- 2 minutes.*" This is one of the mentioned problems experienced with this modem. u-blox had to be contacted via e-mail in order to get log-in information to the firmware download site. The firmware update process works by connecting the microcontroller to a PC via USB and flashing it with a *serial pass-through sketch*. This sketch makes the microcontroller pass all the data received via USB directly onward to the u-blox modem. Then another program on the PC sends the new firmware to the microcontroller. The whole process takes about five minutes.

The update process were successful, and the modem confirms that the new firmware has been installed. So far, the impressions after the update have been satisfying. Except for the first network connection which took about 70 seconds, all connections are made within about 10 seconds with only a few seconds variation. Prior to this update, the modem had quite the difficulty to get any network connection in Setesdalen (see chapter 6.2). Now the results are just as good there as in Trondheim. Connection time is at a stable 9-12 seconds, and the signal quality is mostly equivalent to what Telia's coverage map claims it to be. Experiences so far indicates that the firmware update have solved all the connectivity issues.

11 Temperature measurement

As discussed in chapter 2.1, temperature affects how water flows from one location to another. It is therefore of interest to the operators at AE to know the temperatures in addition to the water depths/flow. Public services like *www.yr.no* can be used to get this information, but it's more troublesome than getting the local temperatures displayed along with the depth data in real time. A temperature sensor will therefore be included in this system.

The temperature sensor chosen for this job is the BMP280 produced by Bosch. This is an accurate, low powered and relatively cheap sensor. The price for the sensor mounted on a breakout board with convenient pinouts, is as little as \$1. Although the sensor itself has an accuracy of ±0.01°C, the resolution can become as fine as ±0.0003°C due to oversampling (Bosch Sensortec 2015). The peak current consumption during measurement is only 325 µA at 3.3 V.

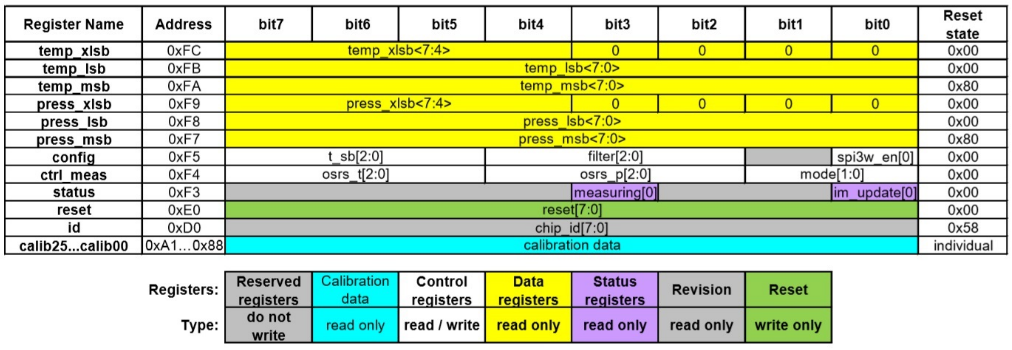
11.1 Wiring

This sensor can communicate using both I^2C and SPI. I^2C will be chosen as the communication protocol simply because it requires only two wires for communication, compared to four wires for SPI. I^2C is also a more commonly used protocol for temperature sensors. So this setup will also make it easier to swap the temperature sensor in the future if desired. SCK and SDI of the sensor is connected to the microcontrollers SCL and SDA pins respectively. Since the BMP280 uses so little current, the sensor can easily be powered directly by one of the microcontrollers GPIO pins. In order to set all the pins as input with pull-up resistors when the sensor is inactive, the ground pin of the sensor must be connected to a GPIO pin. Pin PB1 (also labeled A1) will be chosen here as ground. The power supply pin of the sensor, V_{IN}, is connected directly to the 3.3 V rail. The wiring can be seen in the complete schematics of the node, shown in appendix A.

The temperature sensor can't be placed inside the enclosure together with the microcontroller. The enclosure will from time to time be subjected to direct sunlight, which in turn will increase the temperature inside the enclosure more than the surrounding air. So instead of soldering the temperature sensor onto the board, it's instead connected via a screw terminal.

11.2 The code

There are two operations that must be performed before reading the temperature data from the BMP280. First some settings in the modules control register must be altered. As can be seen in the memory map of the BMP280 in figure 17, the *ctrl_meas* register contains three parameters. Bit 7, 6 and 5, named *osrs_t*, controls the oversampling of the temperature. Bit 4, 3, and 2 controls the oversampling of pressure. And bit 1 and 0 controls the power mode of the device. This register will be set to *0b00100011*. Doing so gives temperature an oversampling of x1, pressure measurement is skipped entirely, and the power mode is set to *normal mode*. An oversampling of x1 of the temperature gives a resolution and accuracy of 16 bits (0.0050°C). This is accurate enough since the plan is to display the temperature with two decimals.



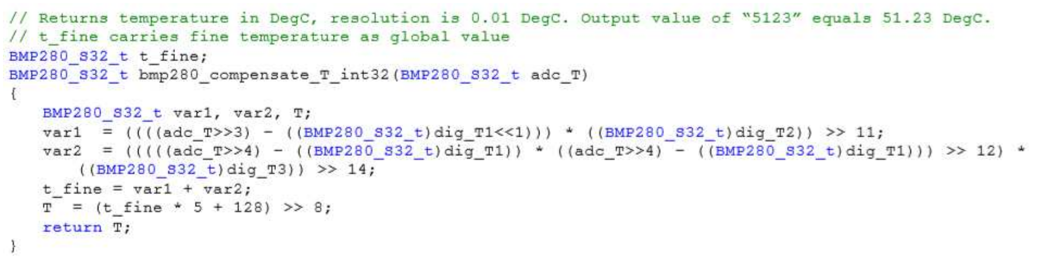
Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
temp_xlsb	0xFC	temp_xlsb<7:4>				0	0	0	0	0x00
temp_lsb	0xFB	temp_lsb<7:0>								0x00
temp_msb	0xFA	temp_msb<7:0>								0x80
press_xlsb	0xF9	press_xlsb<7:4>				0	0	0	0	0x00
press_lsb	0xF8	press_lsb<7:0>								0x00
press_msb	0xF7	press_msb<7:0>								0x80
config	0xF5	t_sb[2:0]			filter[2:0]				spi3w_en[0]	0x00
ctrl_meas	0xF4	osrs_t[2:0]			osrs_p[2:0]			mode[1:0]		0x00
status	0xF3					measuring[0]			im_update[0]	0x00
reset	0xE0	reset[7:0]								0x00
id	0xD0	chip_id[7:0]								0x58
calib25...calib00	0xA1...0x88	calibration data								individual

Registers:	Reserved registers	Calibration data	Control registers	Data registers	Status registers	Revision	Reset
Type:	do not write	read only	read / write	read only	read only	read only	write only

Figure 17: Memory map of the BMP280. Found on page 24 in the BMP280's datasheet (Bosch Sensortec 2015).

Secondly, the temperature data from the sensor needs to be calibrated. The BMP280 uses analogue sensors to measure both the air temperature and air pressure. Although the sensing elements used in the different BMP280 devices behave roughly the same way, there are some minor differences. Calibration is therefore needed. Three 16-bit numbers are factory programmed onto the BMP280's non-volatile memory which are used for this calibration. The addresses of these numbers can be found in table 17 of the datasheet. These numbers will later be used in the compensation formula.

The 24-bit raw temperature sensor data can now be read from the three 8-bit temperature data registers. This number is then used in the compensation formulas found on page 22 in the datasheet, together with the three compensation parameters found earlier. The formulas are shown in figure 18. All the code for the temperature sensor is put in a function called getTemperature(). The only exception is the powering of the sensor, which is done in the event loop.



```
// Returns temperature in DegC, resolution is 0.01 DegC. Output value of "5123" equals 51.23 DegC.
// t_fine carries fine temperature as global value
BMP280_S32_t t_fine;
BMP280_S32_t bmp280_compensate_T_int32(BMP280_S32_t adc_T)
{
    BMP280_S32_t var1, var2, T;
    var1  = ((((adc_T>>3) - ((BMP280_S32_t)dig_T1<<1))) * ((BMP280_S32_t)dig_T2)) >> 11;
    var2  = (((((adc_T>>4) - ((BMP280_S32_t)dig_T1)) * ((adc_T>>4) - ((BMP280_S32_t)dig_T1))) >> 12) *
        ((BMP280_S32_t)dig_T3)) >> 14;
    t_fine = var1 + var2;
    T  = (t_fine * 5 + 128) >> 8;
    return T;
}
```

Figure 18: The compensation formulas for calculating the correct temperature. Found on page 22 in the BMP280's datasheet (Bosch Sensortec 2015).

Most of the code written in this chapter is specific to this very brand of temperature sensor. If the sensor is to be swapped for another I^{2}C temperature sensor, the device address, register addresses, settings and calibration formulas (if any) must be edited. This is however usually not too difficult or time demanding if the datasheet is used properly.

12 User interface

The node should have a user interface so that the technician deploying the node can calibrate the depth, as well as use it to check other parameters if necessary. The user interface must consist of two basic parts; a way of getting information from the system to the user, and a way of getting information from the user into the system. A principle to follow throughout the process is that the interface should be as simple and easy to use as possible, while at the same time have all the required functionality.

12.1 Display

A small amount of data can be transmitted to the user via LEDs, like power status or connection status. However a display can be used to transmit more detailed information clearly. The display that is chosen is the SSD1306. This is a monochrome, 0.96" OLED screen, with a resolution of 128x64 pixels. It can be seen in figure 19. There are several reasons for why this screen is chosen. Firstly it has a low power consumption. At 3.3 V it usually consumes 4-15 mA, depending on how much of the screen is lit. The screen is also cheap, costing about 20 NOK. The display also has a convenient way of communicating with microcontrollers by having a built in I^{2}C driver. Where some other displays, like the popular 1602A LCD uses up to 16 pins for communication, the I^{2}C driver can get by with only two pins in addition to power and ground. Wiring will almost be the same as the temperature sensor: SCL and SDA of the display to SCL and SDA of the microcontroller respectively, and V_{IN} to 3.3 V. The difference is that the GPIO pins of the SAMD21 can't provide, nor drain, enough current for the display. A MOSFET will therefore be used in between the ground of the display and the common ground.

The display works by either lighting up or not lighting up pixels on the screen. Each pixel is represented by one bit of data in the code. This way of building a picture is known as a bitmap (Techopedia 2019). If the bit is 1, the pixel is lit up (if the *inverted* option is in its default state). The library *ssd1306.h* by Dynda (2019) is used to ease the work of displaying images on the display. The library includes a function that places a bitmap image of a given size on a given position on the screen. Bitmap images can be generated using the software *LCD Assistant* (Kwiecień 2019). This program takes a monochrome .bmp image file as input, and outputs the corresponding hexadecimal bitmap representation of the image in a .txt file. The .bmp image can be created using a photo editing program like Microsoft Paint.

A full screen image is 128x64 = 8192 bits = 1024 bytes. If every image to be displayed were to be stored in the RAM of the microcontroller, most microcontrollers would fill up their RAM with relatively few images. To avoid this problem, the bitmaps can instead be stored in the microcontrollers flash memory, in which there usually is a lot more storage capacity. The 256 KB flash storage of the SAMD21 has a maximum guaranteed write cycles of 10,000. This is due to the degradation of oxide layers of the memory cells caused by each erase/write cycle (National Instruments 2019). Since the flash is only written to when the microcontroller is being programmed, this will not be a problem. Reading from flash on the other hand, which will be done a lot more often, doesn't have a maximum theoretical limit.

12.2 Input method

To enable user interaction, the KY-040 rotary encoder is acquired. This is a cheap (8 NOK) input method with a practical design. It can rotate indefinitely in both directions, and it has a built-in push-button. This 20-step rotary encoder have five pins. Two pins for constant high and low reference voltage, two pins for determining which way the knob is being rotated and one pin for the push-button. A good overview of the technicality and the inner workings of the encoder can be found on henrysbench.capnfatz.com (Henry 2015). A rotary encoder doesn't keep track of the position of the rotor, only its movement. The code for interpreting the movement can be found in the code in the file *userInputFunctions.ino*. The main procedure of this code is to read the current states of the pins, and then see which pins change state and in what order. The programmer must also be aware of the *debouncing* effect. When a connection is made or broken, like what's happening inside the encoder, what really happens is that the two connectors connects and disconnects rapidly within a few milliseconds (or less) before settling at their final state (Williams 2015). Although it all happens too fast for humans to observe it with the naked eye, the microcontroller is able to register each state within this bouncing as separate input states.

12.3 Functionality

The primary function of the user interface is to allow the user to calibrate the depth measurements. As discussed in chapter 2.1, the probe of the sensor is placed at an arbitrary depth, whereas the depth of interest is the height of the water over a threshold. The change in water height the two places are equivalent, so the microcontroller only has to calculate an offset and then add or subtract this offset at every measurement. It would also be practical if the user could be able to check the signal quality while moving around before deciding where to place the node. In other words; the system should have two modes. One mode to check signal quality, and one mode to calibrate the measurements. A *mode switch* is added to the circuit in order to change between the two modes.

When asked about their language preference in the user interface, AE said that Norwegian would be nice. The whole user interface will therefore be in Norwegian.

12.3.1 Calibration

If the switch is in position one (later defined as the *calibrate position)* before the system is powered, the interface will go into calibration mode. Here the user will be able to use the encoder to input the manually measured depth over the threshold. If the knob is rotated, the selected digit will change its value. If the knob is pushed down, the next digit will be selected. The calibration screen is shown in figure 19. The interface displayed here consists of six different bitmaps. The static text is one image, each number is one image, and the dash below the digit being changed is one image. The screen is refreshed each time one of these parameters are changed. It would take some time to explain the code here, so the reader is instead recommended to look at the documented code if interested. The code for calibrating the node can be found in the functions inputCurrentDepth() in userInputFunctions.ino and line 226 to 245 in the main program.

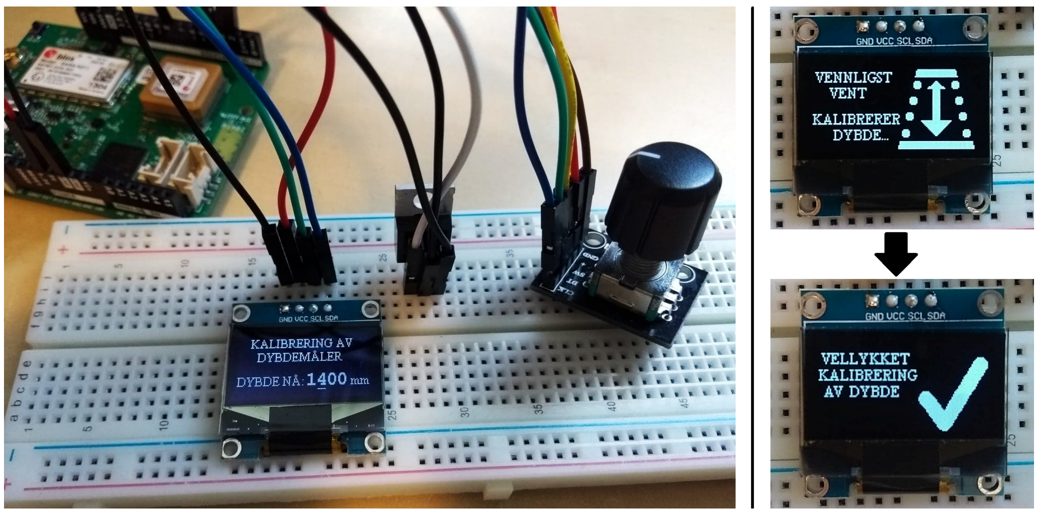


Figure 19: Here the depth is being calibrated. It's currently the second digit from the left that is being changed. The *mode switch* was not implemented yet at the time this picture was taken.

After the manually measured depth has been entered by the user, the node reads the measurement of the OTT PLS' probes depth, and then calculates an offset and whether to add or subtract this offset. These two parameters should be stored in a non-volatile memory so that the node doesn't have to be re-calibrated every time the power is cut or the node is restarted. Many microcontrollers use EEPROM (Electrically Erasable Programmable Read-Only Memory) for storage of data that mustn't get lost by a reset or a power loss. Unfortunately EEPROM is a feature that the SAMD21 lacks. Instead the flash memory can be used as a simulated EEPROM by including the *FlashAsEEPROM* library (Maglie 2018). In practice these two methods works almost the same way. One must just be aware that contrary to EEPROM, the flash memory gets erased and re-written every time the microcontroller is flashed with a program. Two additional functions must be created in order to use the flash memory as intended. The included library only deals with single bytes, but the depth data needs to be stored as a uint16_t. So the functions *FlashUpdate16bits()* and *FlashRead16bits()* are created to handle writing and reading of two-byte numbers. Now the two values can be stored in flash memory.

12.3.2 Signal quality check

If the switch is in position two (later referred to as the *normal position* before the system is powered, the modem will try to register with the network, and then the signal quality will be displayed on the screen. The position is called *normal* because after displaying the signal quality for five seconds, the system is supposed to go straight down to the event loop, skipping the calibration. If the system were to reset itself for some reason during it's normal operation, it will show the signal quality and then go back to normal operations. These two features are combined in order to not have to include an extra button or a three-position button, and thus ease the user experience.

When sending the command to the modem to return the signal quality, the returned value is a unitless number from 0 to 31 (or the number 99 if there is no connection). A table in the modems datasheet shows how these numbers maps to the RSSI value. 0 means a quality of -113 dBm or worse, 31 means a quality of -51 dBm or better. The rest of the numbers are mapped in between these values. Unless the user has specific experience with the SARA-N2 modems, it's hard to understand what these unitless numbers mean. The microcontroller will therefore be programmed to map the numbers to their corresponding RSSI [dBm] values before displaying them on the screen. Then again, a user without experience within telecommunication might have a hard time understanding the RSSI values as well. A more user friendly approach is to display the signal quality as a percentage. Although dBm isn't a linear scale, a linear mapping gives a good enough approximation for the user to understand how good the signal is. In some use cases, Agder Energi might actually want to display the signal quality in terms of dBm. Both options are therefore kept in the code. Agder Energi can choose to switch back and forth between these two displaying options by un-commenting or commenting out two lines of code in the displaySignalStrength() function.

13 First prototype

Now that the most important features of the system have been implemented, it's time to make the first prototype. This prototype will be tested out in the field, subjected to the same conditions as the final product will be subjected to. The wiring of the preliminary design have been tested and validated on the breadboard, so the next step is to make the same circuitry on a prototype board. A waterproof and practical enclosure will also be designed, where all the electronics can be safely placed while out in the field.

13.1 Prototype board

A prototype board (also called a perfboard) is a thin board, usually about 1.5 mm thick, with pre-drilled holes (Cunningham 2018). These holes have the same spacing as the holes on a breadboard, 2.54 mm, and they often have a ring of copper around them. The components are manually placed on the board, and the connections can be made by either soldering on wires, or by making the paths directly on the board using soldering tin. A prototype board allows the developer to test the physical layout of the design before ordering a manufactured circuit board. A prototype board reduces the chance of wires falling loose, compared to having the same design on a breadboard. This will be convenient when the prototype is to be tested in the field.

The board is created as a shield that can be put directly on top of the SODAQ board as shown to the right in figure 20. The three screw terminals serve as entrances for the depth sensor, the battery and the temperature sensor. The solar panel will be connected via its original socket on the SODAQ board. Since the rotary encoder is relatively tall, it's connected to the bottom of the prototype board so that the base of the encoder is on the same level as the SODAQ board. Two capacitor are added to the circuit to stabilize the power delivery. One capacitor is placed between the positive battery voltage and common ground. The other one is placed between the positive regulated 3.3 V and common ground.

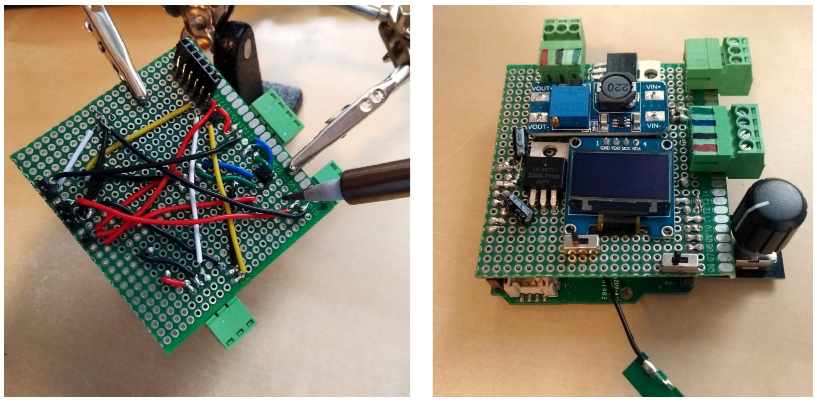


Figure 20: Soldering components to the prototype board. One switch is the *Mode switch*, while the other one turns on and off the power once a battery is connected. The screw terminals serve as inputs for the depth sensor, the temperature sensor, and the battery.

13.2 Enclosure

This depth measuring system, including the prototype, shall be deployed out into the field where there at times will be a harsh environment. The electronics must therefore be protected in a waterproof and robust enclosure. A custom enclosure shall be made. One which is designed for the specific usage of this system. Whether a custom made enclosure will be used for the final product is not yet decided. For the first field testing, a pre-made industrial IP68 rated enclosure will be used. It's likely that the finished system will continue to use one of these pre-made enclosure for the first period. Producing custom enclosure adds more expense to each node. If AE thinks that the pre-made enclosure works fine, they will most likely continue to use them. The custom enclosure will be made either way in order to show the potential of what a design from the ground up for this system could look like.

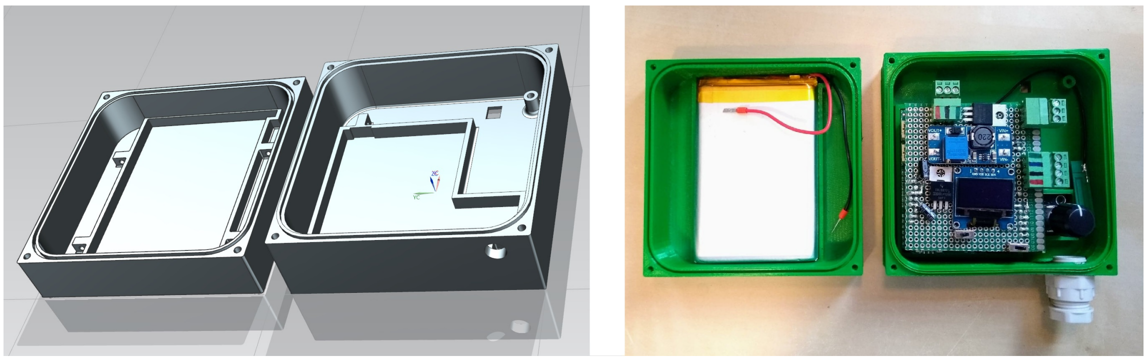


Figure 21: The 3D-model of the enclosure created using Siemens NX, and a 3D-print of it. The dimensions are 100 mm x 100 mm, with a combined height of 52 mm.

Figure 21 shows the 3D-model of the enclosure. It's created using Siemens NX. All parts of the system are measured using a caliper in order to get the correct dimensions for the model. The model measures 100 mm x 100 mm, and the two parts has a combined height of 52 mm. The part on the left has room for the battery, as well as fasting points for cords to keep the battery in place. On the top it has an indent going all the way around where there's room for a gasket. In each corner there are holes for bolts to fasten the two parts together. The right part is where the rest of the electronics is going to be placed. The outline at the bottom shows where the microcontroller with the shield will be placed. The circular hole is where the water resistant grommet for the depth- and temperature senor will be mounted. The extruded part next to the rectangular hole is an anchor point for the depth and temperature sensor. By having them anchored here, the potential pulling force from the wires will be unloaded here instead of in the screw terminals. The rectangular hole is a pass-through for the wires to the solar panel. The solar panel itself is mounted in a groove on the backside.

A full scale version of the enclosure is 3D-printed using an Ultimaker 2+. The settings are set to low quality since it for now only will be used for demonstration and user-testing, not field testing. The enclosure will be shown and given demonstration of to AE in order to get feedback on the design from the ones who will become the end-user of the product.

13.3 Field test

A thorough NB-IoT network signal quality check is performed at several important locations in Setesdalen. Both the standard small PCB antenna used so far, and a larger antenna in the same frequency range is tested. Close to the main roads and in villages, the large antenna proved a little bit better, but no major difference is seen. However on the mountains and other remote locations where the signal quality is worse, the larger antenna performed significantly better. Some places the large antenna was able to send messages where the small antenna couldn't get any reception. The larger antenna will therefore be used from this point onward.

The 3D-printed enclosure were printed for demonstration purposes only, and is therefore not suitable for actual field testing. A pre-purchased IP68 rated box is going to be used instead. This temporary enclosure has the dimensions 160 x 240 x 90 mm, and weighs roughly 300 grams. Holes are drilled into the box in order to fit the IP-67 vent, the grommets for the antenna and the cables, and a hole in the lid for the wires to the solar panel. The electronics are secured inside along side with a silica gel bag that will absorb potential moisture. The "ready to go" prototype can be seen in figure 22. Five values will be sent back from the field to AllThingsTalk Maker. These are the depth, the battery voltage, the connection time, the temperature and the counter value.

The prototype is placed a few kilometers away from AE's operations center in Brokke, Setesdalen. This is a good location to deploy the prototype because it has easy access to one of the larger rivers. In addition, AE already has another depth measuring station a little bit further downstream that can be used to check if the measurements from the prototype are correct. The picture on the left in figure 23 shows the prototype where it's been placed, calibrated, and has started it's normal operation. The probe of the depth sensor is placed and secured in the banks of the relatively still backwater of the river behind the enclosure. The river flows onward to the right of the backwater. The picture to the right in figure 23 is taken about 6 meters away from the prototype. The water height over the concrete threshold (the weir) seen here is the height that prototype has been calibrated to match. At the time of deployment, the height over the threshold was 110 mm.

13.4 Results from the prototype's first week in the field

The results from the prototypes' first week seems promising. The node has sent every package it was supposed to at the correct time. There have been no false or corrupted depth measurements. The connection time have always been between 9 and 11 seconds. Figure 24 shows the battery voltage from the first week of the test. This graph shows that the battery has been charging during the day, and that the voltage have been quite stable during the night. The graph shows that the voltage measurements overshoots a bit during the day before getting stable during the night. This is probably because of the constant 4.2 V output voltage of the charge regulator while the solar panel provides charging current (Microchip Technology Inc. 2014b). Before deploying the prototype, the sleep current of the system was measured. Measurements showed it to be 30 µA with all the peripherals connected. Although this

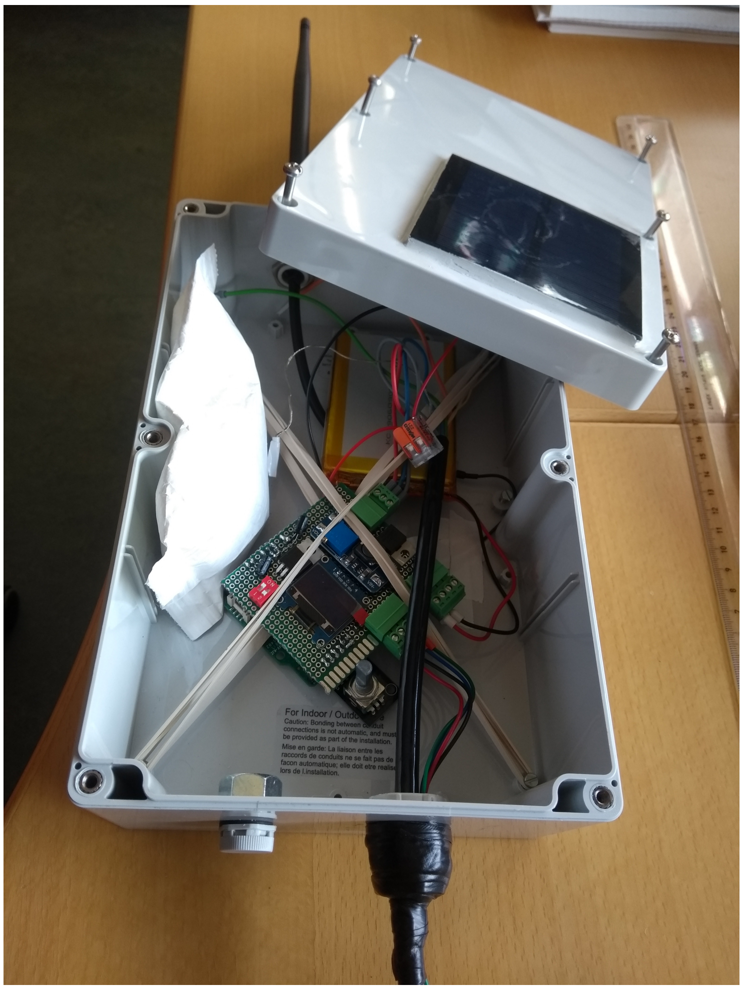


Figure 22: The prototype inside the temporary enclosure.

power consumption is well within the acceptable range, attempts will be made to lower it even further.

One obvious mistake that was discovered after the first night in the field, was that the code was not able to interpret negative values. The temperature the first night went down to minus 0.41°C. The node sent this data back as the value 65495. Luckily this is an easy fix that will be changed for the next version. The prototype will stay in the field for for as long as it seems useful.

13.5 Feedback from AE and discussion regarding further work

AE are generally pleased with the prototype (Bjønnes 2019). They especially liked how easy it was to use, how small and light it was able to become, and that it had such a low power consumption.

Høgevold (2019), engineer at ICT and process control at AE, have some constructive feedback regarding the 3D-printed enclosure after having tested it. Firstly, the enclosure was missing the hole for the vent. The vent is used to make the air pressure inside the enclosure equal to the air pressure outside. This allows the depth sensor to use its tube to compensate for atmospheric pressure. The tube should also be pointing downwards inside the enclosure. This makes less moisture escape from

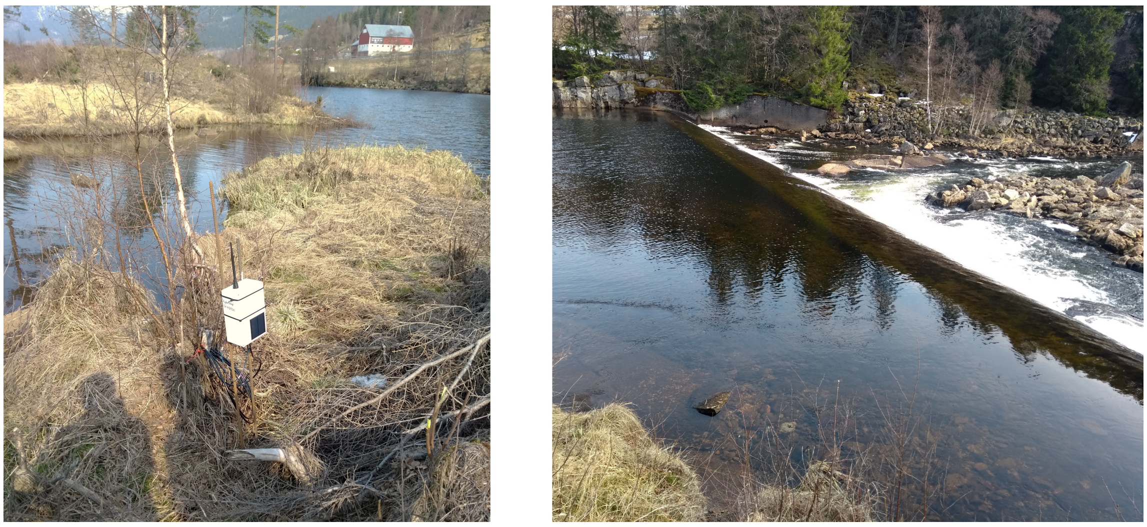


Figure 23: First field test. The picture to the right shows the node mounted to a thin steel rod. The picture to the right shows the threshold that depth measurements are calibrated to.

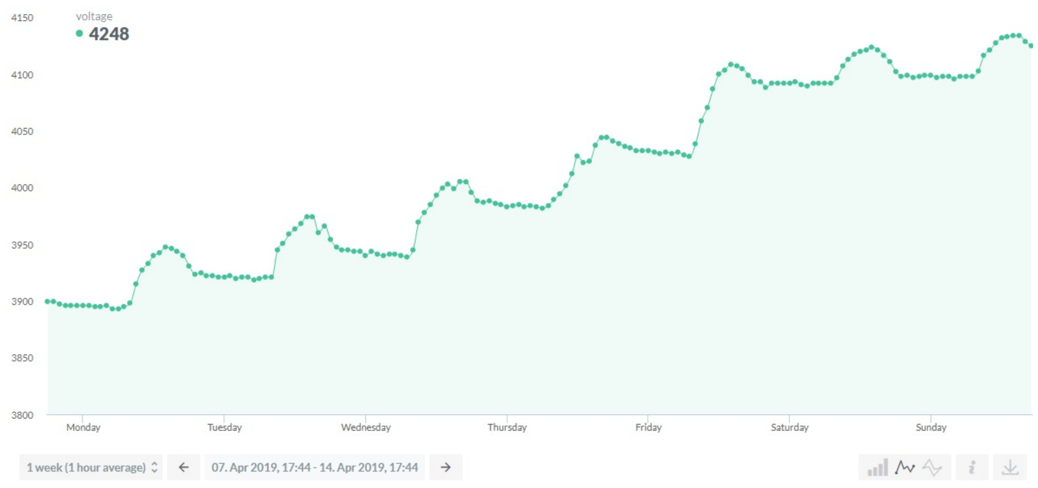


Figure 24: Battery voltage the first week of the test. AllThingsTalk Maker is used to display the data.

the tube into the enclosure. AE liked how small the enclosure was able to become while still fitting all the electronics inside. In fact, it was a bit too small. A enclosure of this size gives too little room for cold fingers without fine motor skills, which often is the case half the year. It would also be easier to lose the node with a enclosure this small. So the enclosure should be a bit bigger. Lastly, it was discussed whether the new big antenna should be on the inside or the outside of the enclosure. If the antenna is on the outside, it's less likely to be covered in snow than what the enclosure itself is. Especially if it is pointing downward under the enclosure. The antenna will therefor be on the outside, and the new enclosure will thus need a hole for the antenna.

Some ideas are thought of as the project goes along. For example in addition to sending data back from the field, AE would now also like all the data to be stored locally on the sensor node. This is to have the possibility to retrieve the data history in case something were to happen to the antenna or the modem or some other network-related feature.

At this point it's also time to add two features that have intentionally been left out so far. These are the network registration time-out and re-sending of unsent data. The network registration time-out has not been implemented before because it's interesting to see how long it takes for the modem to connect if it's always given an unlimited amount of time. Now that the first prototype reports back on this matter, the timeout can be implemented for the next prototype.

14 Implementing the new ideas and requests

14.1 Connection timeout

As discussed in chapter 8.3, the getTheTime() function is used to check the network connection in addition to updating the time and date. The timeout must therefore be implemented in this function. In the NB-IoT's firmware update specifications it said that the roaming connection time should be less than two minutes. This is the time it takes to get connection if the modem switches base station. Although the modem should be connected to the same base station the whole time after the first connection, it might happen to switch if one station goes down for a while. The connection timeout should therefore be set to 130 seconds, so that it will have time to reconnect to a new base station if needed. Ten seconds is added to the two minutes as a buffer margin. The implementation is relatively simple. A timer is started inside the function once it's called. If it connects to the network within 130000 ms, it sets the time/date and returns 1. If not, the function returns 0. getTheTime() is renamed getTime130sTimeout().

14.2 Re-sending unsent data

Ideally, the data from the node should be sent to a server that forwards the messages to AE's SCADA system, which is the plan eventually. This server will be able to send an acknowledgement message back to the sensor node, informing it that the data has been received. As of today the data is sent to AllThingsTalk Maker. AllThingsTalk Maker does not acknowledge received data. The node is thus not informed whether or not a package has been successfully delivered. The criterion for saving and re-sending data will now instead be based on whether the modem has connected successfully or not. The functions for saving and re-sending will be the same either way. So the criterion can simply be changed later once the server is up and running.

Two functions are created. One named storeUnsentData(), which is called if getTime130sTimeout() returns 0. This function stores the data in an array along with a *unsent status identifier*. If the identifier is 1, this set of data has not yet been sent. If it's 0, the data has been re-sent earlier, and the content in this position of the array can be overwritten. The other function is called sendUnsentData(), and is called if the modem is able to connect. This data checks the array used by storeUnsentData(). If the identifier is 1, the function sends the data and clears the identifier.

14.3 Local data logging

All the data, both the sent and the unsent, should be stored locally on the node itself so that it later can be manually extracted. The SAMD21 microcontroller does have some memory to spare. The program uses only 17% of the total 256 KB in-system flash memory. As of today the node produced 10 bytes of data every 15 minutes, which gives enough free flash memory for 221 days of data. This is not enough, as the node is supposed to operate for many years without needing physical handling. Also, extracting data from the flash memory of the microcontroller means unnecessary work as the SODAQ board needs to be connected to a computer. A better solution is to log the data onto a commonly used external storage unit. A relatively cheap and effective solution is to use a microSD card for storage. A microSD card breakout board can be bought for 10 NOK, and a 16 GB microSD card can be bought for 100 NOK (OmegaVerkstedet 2019). 16 GB is more than enough for hundreds of years of storage.

The microSD module contains no drivers at all. It only provides logic level shifters, resistors, and convenient pinouts. Communication happens directly with the card itself (SD Association 2006). SD cards can communicate using two different protocols. One called SD and one called SPI. The SD protocol is a lot faster, and is the one used by for example video cameras. The problem is that this protocol is quite complex and is generally not recommended as the interface with a microcontroller. SPI on the other hand is simple to use with microcontrollers. As mentioned in chapter 4.1, SAMD21 has dedicated hardware for this very communication protocol. With a bit rate of 6 MB/s, SPI is more than fast enough for this project. Storing data to-, and reading data from an SD card in an organized way is a bit tricky. It's not as simple as handling a few registers as with the temperature sensor. A library called *SD* made by SparkFun Electronics (2010) will therefore be used to handle the reading and writing of the SD card. Using a pre-written library for more than just communication protocols can be a bit risky because it allows external code to take control of the microcontroller. This isn't necessarily bad. The problem is if the code changes the configuration of registers in the microcontroller that affects later operations, without the programmer being aware of it. One should therefore make oneself familiar with the procedures of the code in the libraries before using them.

Data is stored onto the microSD in a .txt document in a comma separated value (CSV) format. Each set of measurements are separated by a new line. A new .txt fil is created for each month. By creating a new file every month, there won't be one single large file on the card after a few years. Opening that one large file before being able to write to it would take longer and longer time as it gets larger. The name of the files are on the format mmyy.txt, where mm is the month and yy is the last two digits of the year. A few functions are created to get the data in the right format for storage, but all the read/write operations are handled by the functions in the SD library.

In an unfortunate scenario, the the microcontroller could be unable to communicate with the microSD card, or the files on the microSD card are unable to open. This could leave the microcontroller in a waiting state, waiting for the files to open. As a fail-safe for this not to happen, the microcontroller now writes to the file only *if* the microcontroller establishes contact, and *if* the file opens. Otherwise it skips this part of the code.

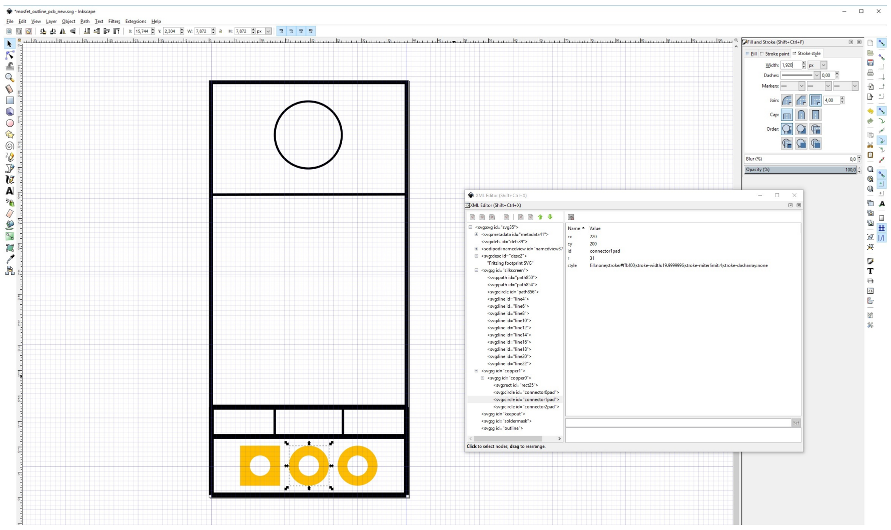
15 Printed circuit board

Now that the prototype has been tested, the new features has been added, and everything seems to work fine, it is time to create a printed circuit board (PCB).

A computer program called Fritzing will be used to create the PCB design. When the design is finished, Fritzing can export the PCB files into a format that can be sent to a PCB manufacturer. Fritzing has a part-list containing components which can be used in the PCB design. Although there are many components in this list, and components files can be downloaded form communities and manufacturers online, there are some components used in this project that aren't to be found. These components must therefore be created. Some parts must be created from scratch, while others can be made by modifying existing parts.

15.1 Creating custom parts

The component files used in Fritzing are created using vector graphics. A free and open source vector editor program called Inkscape will be used here. Figure 25 shows the creation of the PCB part file for a flat-laid MOSFET using Inkscape. There are three important features that are essential for the PCB components file. These are the holes, the copper and the silkscreen. The components used in this project are all through-hole components, meaning they have pins that goes through the board[16]. The copper feature tells Fritzing where the copper pads should be and how they should look. Lastly, the silkscreen is the visual markings on the board, which among others are used to draw the layout of the components. The silkscreen lets the designer know how big the components are so that they won't be placed on top of each other. It also helps when soldering, to show where the components are supposed to be placed.



```
<svg:svg id="svg35">
  <svg:metadata id="metadata41">
  <svg:defs id="defs39">
  <sodipodi:namedview id="namedview37
  <svg:desc id="desc2">
    "Fritzing footprint SVG"
  <svg:g id="silkscreen">
    <svg:path id="path850">
    <svg:path id="path854">
    <svg:circle id="path856">
    <svg:line id="line4">
    <svg:line id="line6">
    <svg:line id="line8">
    <svg:line id="line10">
    <svg:line id="line12">
    <svg:line id="line14">
    <svg:line id="line16">
    <svg:line id="line18">
    <svg:line id="line20">
    <svg:line id="line22">
  <svg:g id="copper1">
    <svg:g id="copper0">
      <svg:rect id="rect25">
      <svg:circle id="connector0pad">
      <svg:circle id="connector1pad">
      <svg:circle id="connector2pad">
  <svg:g id="keepout">
  <svg:g id="soldermask">
  <svg:g id="outline">
```

Name	Value
cx	220
cy	200
id	connector1pad
r	31
style	fill:none;stroke:#ffbf00;stroke-width:19.9999996;stroke-miterlimit:4;stroke-dasharray:none

XML Editor (Shift+Ctrl+X)
Click to select nodes, drag to rearrange.
Fill and Stroke (Shift+Ctrl+F)
Join:
Cap:
Order:
Blur (%)
Opacity (%)

Figure 25: The creation of a new PCB part in Inkscape. Here a laid-down MOSFET is created. The XML-editor is the window to the right of the part.

The XML editor in Inkscape contains the code for the graphical representation of the part. This is a useful tool when designing components. Fritzing recognizes certain names in the XML Editor as features. Like the name "copper1" is the copper connector on the top side of the board, "copper0" the copper connector on the bottom of the board, "silkscreen" is the outer markings of the parts etc. The "levels" of the features in the editor are also important. For example must copper must always be on the top level, which in Inkscape is on the bottom of the XML Editor.

When the component is finished in Inkscape, the vector graphic .svg file can be imported into Fritzing. Here the pins and ports designed in Inkscape can be associated with corresponding pins and ports labels in Fritzing. The final PCB design process can start once all the part files have been created and properly imported.

15.2 Design rules and creating the PCB

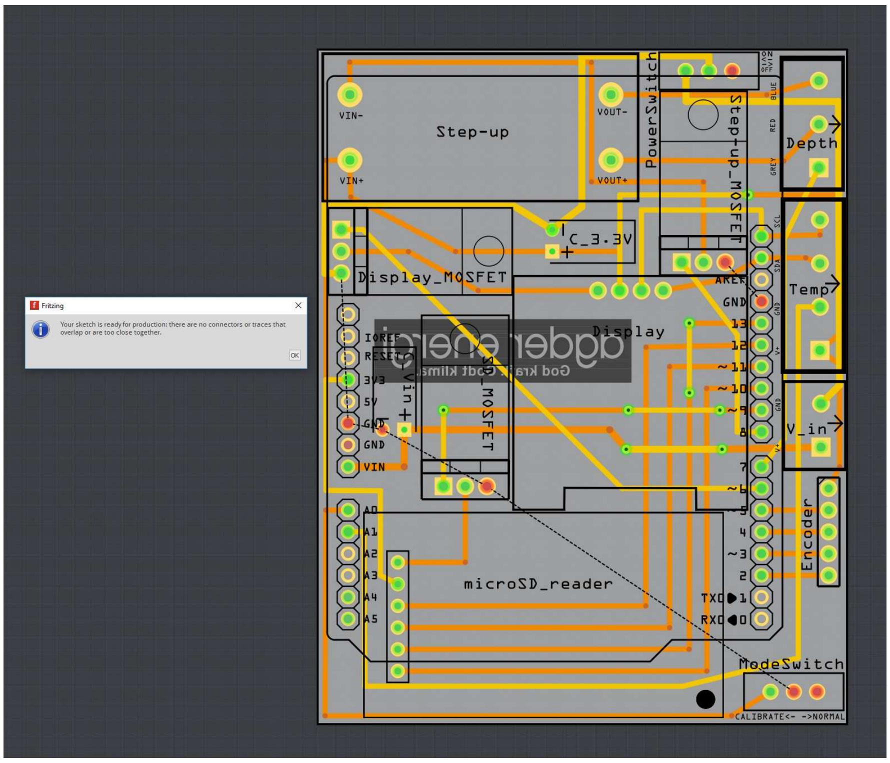
A common myth in the engineering world is that the traces on a PCB never should have 90 degree angles. This is due to the fear of electromagnetic interference caused by electron bouncing when high frequency signals must pass sharp corners. This myth has however been proven wrong for most practical cases (Johnson 2000). If the frequency of a signal is in the 10 GHz range, unwanted noise may occur. However since the maximum clock frequency of the MCU used in this project is 48 MHz, and no peripheral components talk directly to each other, it's given that none of the signals passing through

[16]Contrary to surface-mount components.

the traces on the PCB created here will be anywhere near the "danger zone".

A PCB can have several layers. A simple PCB might only need one layer, but here there are too many crossing connections that needs to be made for only a single layer. Having two layers greatly helps routing. A tool called *autoroute* can automatically create the routes on the board. Even though this tool works, autoroute is rarely recommended. Most of the time autoroute creates quite messy and overly complex routes. Therefore it's often best to do it manually. The list below presents some criteria to take into consideration when placing the components and creating the routes.

- Convenient placement of the screw terminals.
- Encoder and switches should be easily accessible.
- The two switches should not be too close to one another to prevent that one gets flipped by mistake.
- Traces should not go too close to coil of the step-up converter in order to not get interference.
- The main power supply should have thick tracks.
- It must be easy to insert and remove the microSD card from the reader.
- The functions of the switches and the screw terminals inputs should be labeled.


Fritzing

Your sketch is ready for production: there are no connectors or traces that overlap or are too close together.

Figure 26: The design rules check have here been used to check for crossing- or too close connections. None were found, and it says that the PCB is ready for production. The only thing missing on this PCB is the ground fill connection all the common ground pins.

The PCB shown in figure 26 was created by following the previously presented criteria. Several layouts were tested before concluding with this one. With the components placed as tight as practically doable, the board's dimensions are 62 mm × 79 mm. The routes have two different colors. Yellow routes are on the top layer of the board, while orange routes are on the bottom. Some places the routes changes which layers they are on. This is done using a feature called *vias*. Vias makes a copper-filled hole between the two layers of the board. The black box with the inverted Agder Energi logo is a silkscreen image that will be printed on the bottom side of the board. In the bottom right corner of the microSD reader, there is a black circle. This is simply a hole where the module can be fastened to unload some of the force from the soldered pins when inserting and ejecting cards.

A tool called *Design Rules Check* is used to check if routes overlap or are too close to each other. As can be seen from figure 26, no problems are reported in this design. An observant eye will notice that there still are some connections that have not been made yet. These are the common ground connections. The reason they're not connected using routes is because both the top and bottom layer will serve as a ground plane. This is done using a tool called ground fill. Ground filling the unused parts of the layers is a common practice since fewer connections needs to be made. Also, the ground plane can serve as a heat sink, although this will hopefully not be necessary here since the goal is to use little power and thus create little access heat. Before applying the ground fill, two parameters must be set. The ground fill seeds and the ground fill keepout. Ground fill seeds are the connections which will serve as common ground. Ground fill keepout is the buffer zone between the routes/connectors and the ground plane. Here the ground fill keepout is set to 14 mills[17]. Note: figure 26 does not display the PCB with the ground fill because the ground fill makes it hard to see some of the routes. After the ground fill has been added, Fritzing can export the design into a format called Extended Gerber (RS-274X) which consist of several files containing different properties of the different layers of the PCB. These files must be compressed into a singe file (like .zip) before being sent to a manufacturer.

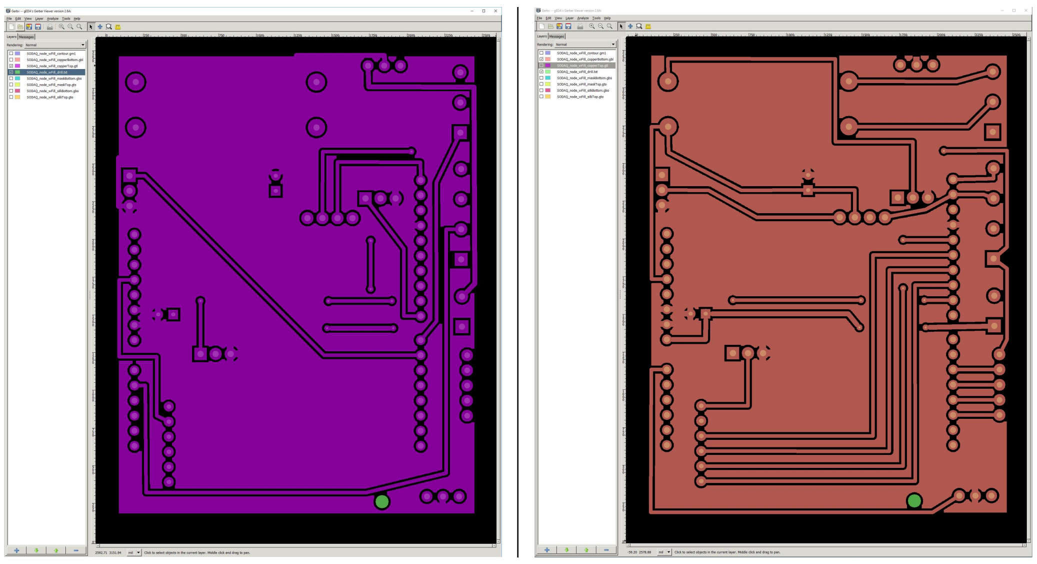


Figure 27: Gerbv is used to check the final design of the PCB. The top layer is to the left, the bottom layer is to the right.

One last check should be made before sending the file to the manufacturer. A program called Gerbv can open each single file contained in the extended gerber file, and visualizes the properties. To the left in figure 27, Gerbv shows the top copper layer and the drill holes. Note that the middle connector in the bottom right corner is disconnected from the rest of the common ground pins even though the plane has a ground fill. This could be a problem. Luckily the bottom layer connects this connector to some of the other common ground pins, which again is connected to the rest via the top layer. The middle connector of the power switch (as seen in figure 26) at the top of the board had to be given an

[17]The unit mils is equal to 0.001 inches.

additional trace above it in order to get it connected to common ground. This trace can be seen on the top layer in figure 27. Things like these are important to check in programs like Gerbv.

The gerber file is uploaded to a manufacturer called JLCPCB. JLCPCB offers five 2-layered high quality PCBs for only $2 plus shipping (JLCPCB 2019). A final check of the PCB can be performed on their web page using the tool *Gerber Viewer* before placing the final order.

15.3 The finished PCBs

Five PCSs arrived after six days. All the connections on the board is tested using a multimeter, and everything seems to be in order. The plane PCBs and a PCB with the components solders on can be seen in figure 28. Both the PCB and the system as a whole seems to be working as intended.

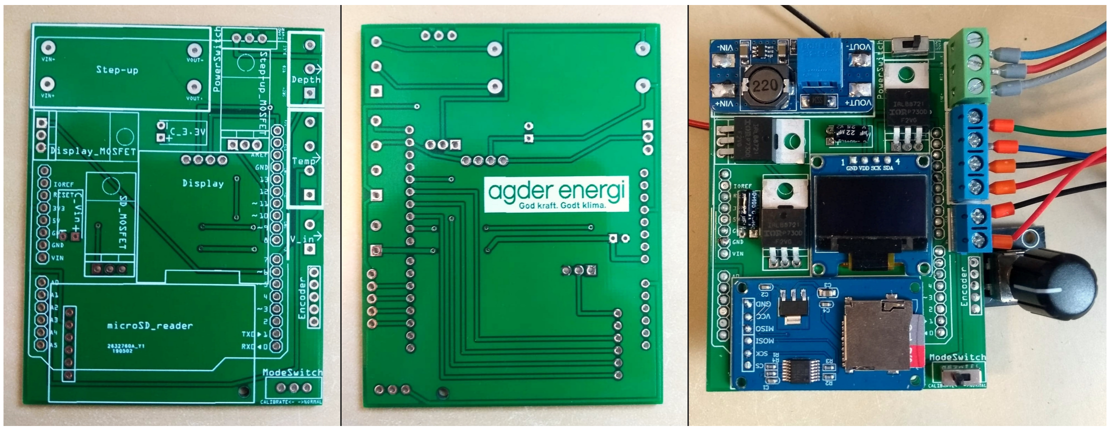


Figure 28: Left: Front of the PCB. Middle: Back of the PCB. Right: The PCB with all the components soldered on.

There are however some design choices that will be changed before ordering the next batch. These are as follows:

- Some of the silkscreen text, like the marking of the screw terminals inputs, are too small. They must be enlarged.
- Some of the components silkscreen markings, like the MOSFET, is a bit off. This can be changed to fit better.
- It's easy to inset the microSD card, but a bit more troublesome to eject it. The microSD card module will therefore be placed so that the card is pointing out of the board.
- The encoder should be placed further down on the board so that it's not so close to the wires from the screw terminals.
- There haven't been any problem with interference from the coil of the step-up converter. However the MOSFET closest to the coil can be moved a bit further down since there is room for it.

After using the PCB for a while, a periodically occurring problem was discovered. At times the microSD card module wouldn't write data to the card. An attempt to solve this problem will be made in the next chapter.

16 New requests and final touches

In addition to fixing the issues discussed in the previous chapter, AE have also come up with some additional features they would like to have implemented in the system.

16.1 Password protection

To be able to calibrate or re-calibrate the depth measurement of the stations AE are using today, the user must enter a four digit password. AE would like this feature on this new system as well.

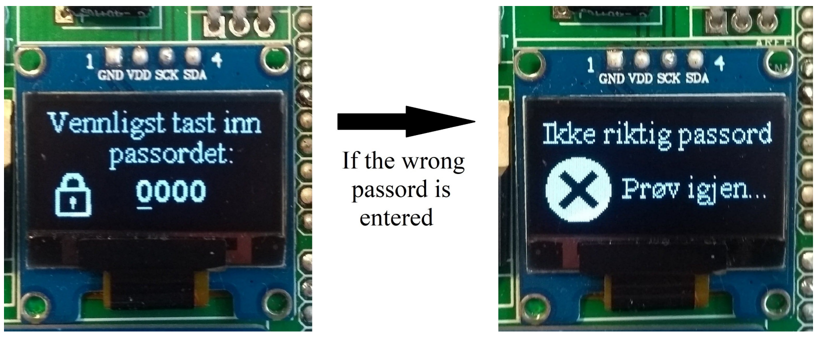


Figure 29: If the password is wrong, the user must wait for two seconds before being able to try again. If the password is correct, the program goes to the calibration screen.

The previously created inputCurrentDepth() function can be used to implement this feature with just a few tweaks. The function is duplicated, and the new copy is renamed inputPassword(). Instead of using the user input to calibrate the depth sensor, the function will be used to input a password. The password will be defined in the code, and can easily be changed by AE later. The inputPassword() function is put inside a while loop with the function's return statement as the criteria. If the password input is correct, the function returns 1, and the program will continue to the depth calibration. If the password is wrong, the function returns 0, and the user is displayed a *wrong password screen* before being given the chance to try again. This scenario is shown in figure 29. In addition to informing the user that the password is wrong, *the wrong password screen* also serves as a delay so that it will take longer to guess the password if someone is trying to "hack" the system.

16.2 GPS

The subject of whether or not to use GPS to log the position of the nodes have been discussed with AE before. It was decided that in order to save power, the user/operator would instead manually log the position when the nodes are being deployed. However, now that the system has proven to charge it's batteries more than it's using them, the extra one-time power consuming operation is no longer an issue. AE would therefore like a GPS position acquisition feature included. The SODAQ board already includes a GPS with a built in antenna, so no additional components are required to implement this feature.

When the position of the node is changed, the depth measurements also needs re-calibration. The GPS will therefore be activated only when the user chooses to re-calibrate the depth senor. However, the depth measurements might need re-calibration without the node having changed its position. It should therefore be an option to abort the position acquisition in order not to use unnecessary time.

To make the user experience more time efficient, the GPS will be powered, and thus start searching for satellite connection before the user starts to calibrate the depth measurements. After the depth calibration is done, the user is presented with a information screen explaining how to abort the GPS position acquisition. Tests with the GPS have shown that the GPS needs contact with six satellites or more in order to get a reasonably accurate location (within a 10 meter radius). The microcontroller will therefore be programmed to automatically store the location once the GPS have had contact with six satellites for eight seconds. However, if the GPS uses an unreasonably long time to achieve contact with six satellites, the user is given the possibility to abort the operation and instead store the less precise location given by the current numbers of satellites. The abortion is done by flipping the *mode switch*. Whether the process is aborted, or if the desired numbers of satellites is reached, the latitude and longitude will be stored in the non-volatile flash memory. The number of satellites the GPS has contact with are shown on the display as shown in figure 30. In the top right corner there is an image of a satellites with some *signal bars* coming out of it. The number of bars will change each second, going from zero to three before starting over again. This is to show the user that the system hasn't run into a glitch if the numbers of satellites hasn't changed for a while.

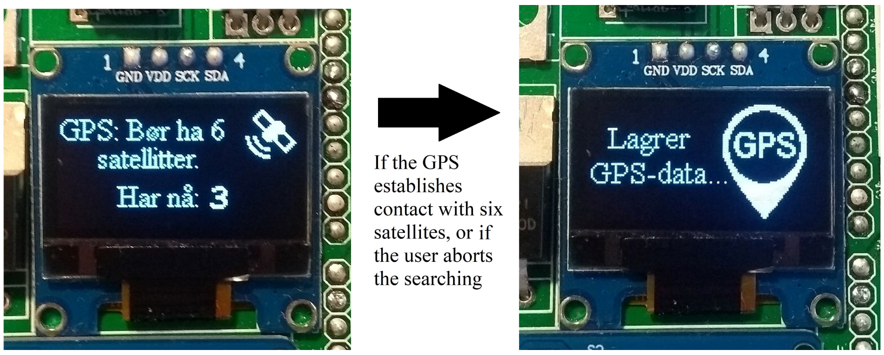

GND VDD SCK SDA
GPS: Bør ha 6 satellitter.
Har nå: 3
If the GPS establishes contact with six satellites, or if the user aborts the searching
Lagrer GPS-data...
GPS

Figure 30: The GPS searching for contact with satellites. If the GPS establishes contact with six satellites, or if the user aborts the process, the current GPS data will be stored in non-volatile memory. The satellite icon changes the numbers of bars every second.

Both latitude and longitude are represented by nine digit numbers. Having numbers this large have implications on several other functions used in the program. The program have until now operated using 16-bit or smaller numbers. This is however too few bits to store numbers as large as the longitude and latitude. The numbers can of course be partitioned into several smaller numbers before being re-assembled as a ASCII hex character array[18] before being sent, but this makes the procedure more complex than it has to be. Instead the whole program will be modified to handle up to 32-bit numbers. 32-bit numbers can be as large as $2^{32} = 4294967296$, which is more than enough for both the longitude and latitude.

16.3 Altitude calibration

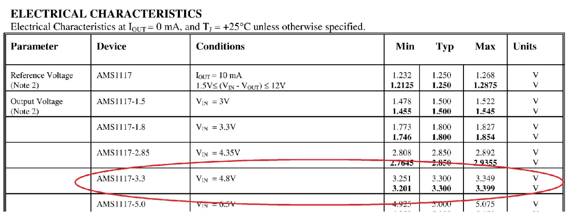
The way the depth calibration works now is by entering the water height over a threshold at the time of deployment. After this, the system measures the depth of the depth sensor's probe and calculates the offset. The possible values that can be entered are from 0 mm to 9999 mm, which is well within the range that the water normally can be above the threshold. In addition to being able to calibrate the depths relative to a local threshold, AE would also like to be able to calibrate the height of the water plane relative to a global reference, namely the height above the sea level. This is of interest at some locations. Especially at reservoirs, where the potential energy is given by the absolute water height (minus the height of the turbines at the power plant). There are permanent markings at the locations where this is of interest that shows the meters above sea level. So the way that AE wants to be able

[18]See chapter 6.3 for more information about the format.

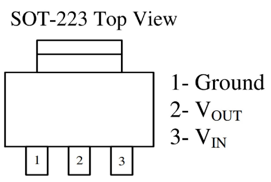
to calibrate the measurements isn't really much different to the way it's done now: Entering the water height relative to some reference, being either a local or a global reference, and then calculate an offset. However, the current calibration range is too short if the 1 mm precision is to be kept. If a node is to be placed on top of a 2000 meter mountain, the input must contain four digits for whole meters and three digits for millimeters. A total of seven digits are needed. Now that the whole program has been adapted to handle 32-bit numbers (see chapter 16.2), the range can easily be extended. Though some graphical work is needed to make the display show the extended calibration screen. To make the calibration more intuitive, the graphical interface now says *Vannhøyde nå:* (Water height now:) instead of *Dybde nå:* (Depth now:) when calibrating. This instruction should be easier to understand for both reference systems.

16.4 Fixing local data logging issues

As mentioned, at times there seems to be a problem writing to the microSD card. The code is thoroughly examined to see if there has been some mistake here. No bugs are found in the software, so maybe the fault is in the hardware. All conventional SD cards operates at 3.3 V (SD Association 2019), the same voltage as the SAMD21 microcontroller. The microSD card module used here has logic level shifters and a voltage regulator to make the module compatible with 5 V microcontrollers. Many regulators can have a varying input voltage from $V_{\mathrm{OUT}} \leq V_{\mathrm{IN}} \leq V_{\mathrm{IN\_MAX}}$. However, it seems that the voltage regulator used on this module, the AMS1117-3.3, must have a fixed input voltage of around 4.8 V to give an output at around 3.3 V (Advanced Monolithic Systems 2019). This seems to match the the characteristics of the issue, since the regulator has a hard time converting the 3.3 V when it's expecting a higher one. To test if this is the problem, and the only problem, the voltage regulator can simply be removed with a pincer. The regulator isn't needed since the SODAQ board has its own 3.3 V regulator. As shown in figure 31, the pads where the input and output terminal of the regulator used to be can now be soldered together.


ELECTRICAL CHARACTERISTICS
Electrical Characteristics at I_{OUT} = 0 mA, and T_J = +25°C unless otherwise specified.

Parameter	Device	Conditions	Min	Typ	Max	Units
Reference Voltage (Note 2)	AMS1117	I_{OUT} = 10 mA $1.5V \leq (V_{IN} - V_{OUT}) \leq 12V$	1.232 **1.2125**	1.250 **1.250**	1.268 **1.2875**	V V
Output Voltage (Note 2)	AMS1117-1.5	V_{IN} = 3V	1.478 **1.455**	1.500 **1.500**	1.522 **1.545**	V V
	AMS1117-1.8	V_{IN} = 3.3V	1.773 **1.746**	1.800 **1.800**	1.827 **1.854**	V V
	AMS1117-2.85	V_{IN} = 4.35V	2.808 **2.7645**	2.850 **2.850**	2.892 **2.9355**	V V
	AMS1117-3.3	V_{IN} = 4.8V	3.251 **3.201**	3.300 **3.300**	3.349 **3.399**	V V
	AMS1117-5.0	V_{IN} = 6.5V	4.925	5.000	5.075	V



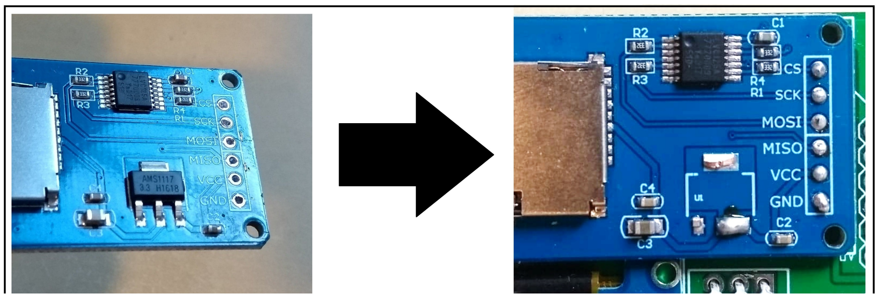


Figure 31: The voltage regulator on the microSD module can't handle 3.3 V, nor is it needed. So it is removed and bypassed.

Bypassing the voltage regulator seems to have worked. The data gets written to the microSD card every time now. Although bypassing the regulator is a simple procedure, it's still an operation that shouldn't be required for the final product. Instead of continuing to use these modules, a pre-compatible module is bought. The Adafruit Micro SD Card Breakout Board is chosen since it's from a renowned producer who uses high quality components on their breakout boards. The module itself is more expensive than the previous one, costing 64 NOK per unit. AE sees this as a good investment.

The new module has pin-outs for both V_{CC}, going through the the voltage regulator, and for a direct 3.3 V pin bypassing the regulator (Adafruit 2019). V_{CC} is the voltage common collector, a generic term used for a positive-voltage supply. Both pins can be used to power the microSD card since the LP298X voltage regulator on the breakout board can take input voltages down to 2.1 V (Texas Instruments 2019). Though only voltages between 3.3 V and 16 V gets regulated down to 3.3 V. Tests with this regulator shows that there is a slight voltage drop of approximately 80 mV when the input voltage is 3.3 V. This isn't critical since SD cards (including microSD cards) has a operating voltage from 2.7 to 3.6 V (SD Association 2019). However, since the 3.3 V voltage regulator on the SODAQ board combined with both a high side and a low side capacitor have shown to provide a stable voltage, the LP298X regulator on the Adafruit module will be bypassed in the new design by using the 3.3 V pin on the module.

16.5 Watchdog timer

The plan was initially to use a watchdog timer (WDT) on the system. A WDT is an integrated timer in the microcontroller that runs independently from the CPU. This timer can be activated at one point in the code, and must be stopped or started over again later in the code. If the WDT experiences a timeout, it's able to reset the microcontroller, starting the whole system over. A WDT can be useful in a system that has a tendency to freeze, or in devices that are hard to physically get to, like a satellite. However, seeing that this system never have frozen, it's questionable whether or not a WDT should be included here. Even when there were problems with the local storage, the system did what it was supposed to do by skipping the operation instead of waiting for the card to respond. And the first field test prototype described in chapter 13.3 have been running for 67 days continuously without encountering any errors. In theory there aren't many downsides to using a WDT. It does use a little extra power, but nothing more than what is manageable. However, it does introduce an extra process that one must be aware of if new features are to be implemented in the future. This means that there would be an extra element that could go wrong.

In other words; if the WDT isn't needed, including it would do more harm than good. However, it's hard to say if it later on turns out that a WDT should have been included after all. As a compromise, some functions and notes related to the WDT will be included in the code so that the process of activating it at a later time will be easy if it turns out that it's needed.

16.6 New PCB and enclosure design

Both the PCB and the enclosure design is updated based on the feedback from AE, and to accommodate the new features. The updated PCB and enclosure design can be seen in figure 32. Their designs are updated with the following features:

PCB: The new microSD module is included, the microSD card is now pointing out of the board with room below it, the smallest labeling is enlarged to become more readable, all pins and screw terminal inputs are labeled, the encoder is moved further away from the screw terminals, the silkscreen now match the components' actual size better, and components are moved even further away from the coil of the step-up converter as an extra safety margin. The new PCB has the dimensions 63 mm x 78 mm, one millimetre wider but one millimetre shorter than the previous one.

Enclosure: Generally a larger box which gives more room inside and it's harder to lose. There are now holes for the antenna, the vent, and the cable grommet. The antenna is pointing downwards below the box so that it won't be covered in snow[19]. The vent and the atmospheric pressure compensation tube from the depth sensor are now physically isolated from the rest of the electronics by a separate compartment to avoid moisture. The compartment gets closed when the top and the bottom of the enclosure are screwed together. All electronics, including the solar panel and the battery, are now mounted in the same part of the enclosure. Lastly, the wires from the depth- and temperature sensor now goes through a tract before reaching the screw terminals. The tract blocks the wires from getting in the way of the rotary encoder. The dimensions of the new enclosure are 204 x 140 x 50 mm.

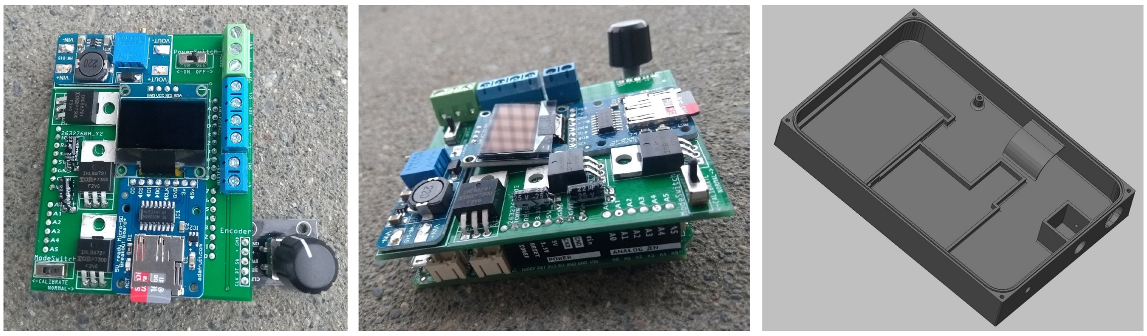


Figure 32: The new PCB and enclosure design.

16.7 Server update and standardization of data format

Telenor have been contacted regarding an agreement on server rental and an NB-IoT subscription. Some of the Scanmatic HydMet stations that are being used today, sends their data via GPRS on Telenor's network to Telenor's servers, before being re-transmitted to AE's servers and then passed on to the SCADA system. This is concidered as a secure way of sending data into the SCADA system. The plan is to extend this agreement to also include Telenor's NB-IoT network.

Since AllThingsTalk Maker soon no longer will be needed, there will be no need to keep sending the data in the specific format required for this application. AE now wants the data to be sent in a standard format that they can use without the need for further translation (Bjønnes 2019). AE have experienced problems related to data translators before. A lot of the equipment they are using, sends data in non-standard format that are specific to the model or the brand. Translators must therefor be placed in between the units and the SCADA system, and in between the units themselves if two different brands are to speak to each other. If this could be avoided in this system, it would be appreciated. The JavaScript Object Notation (JSON) format is one of the data formats AE would prefer. Information about the JSON format can be found on https://json.org.

A function called convertDataToJSON() is created, which takes the raw data as arguments and converts them into a JSON objects. It also uses definitions defined at the top of the main code to match the input arguments to the corresponding asset names and the values' logarithmic factor. The structure, as suggested by Bjønnes (ibid.), is shown in figure 33. This string is the output of the JSON conversion function, which is then sent to another conversion function to turn the string into to the hexadecimal representation of the ASCII characters. This ASCII hex string can now be passed as an argument to a newly written function called NBIoTSendData(). The NBIoTSendData() function uses three parameters; the server IP-address, the server port and the data itself. The function is created to be more flexible and user friendly than the previously used sendToAllThingsTalk() function.

[19]A downward pointing and a upward pointing antenna are equivalent in terms of RF signal polarity.

```
{
 'SensorID':'<HS_IoT1>',
 'Readings':{
         'counter':{'value':1234,'logscale':0},
         'depth[m]':{'value':1234,'logscale':-3},
         'voltage[V]':{'value':1224,'logscale':-3}
         ...}
}
```

Figure 33: The JSON data format structure suggested by Bjønnes (2019) at AE. This type of string is the output of the new convertDataToJSON() function.

16.8 Naming

The last thing this system is missing for now, is a name. The single-node system will hereby be called MiniMåler'n (The small measurer). This name is chosen because it describes what the product is, it is short, and some might even find it catchy.

17 Final results and further work

17.1 Results

A new low powered depth- and temperature measurement system for use at remote locations have been developed in this project. This system have been given the name *MiniMåler'n*. MiniMåler'n is designed with the aim to be reliable, use little power, be small and light, be easy and intuitive to operate, while at the same time be cost-effective. The schematics of the node can be seen in figure 34, as well as in a larger scale in appendix A.

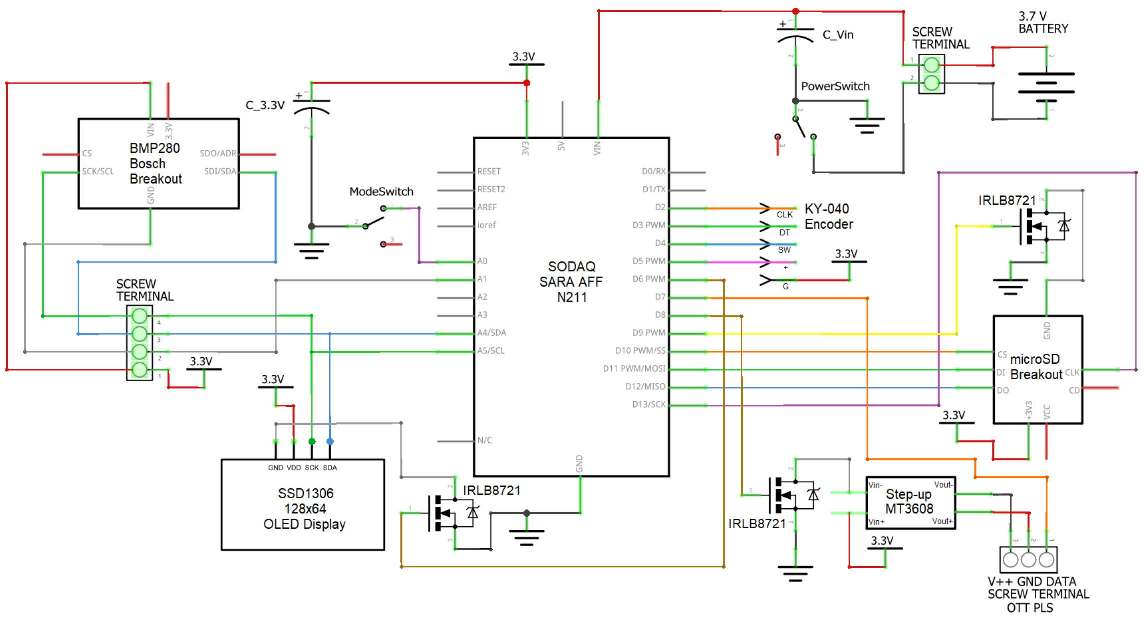


Figure 34: The schematic of the node in the single-node system. The schematic is made using Fritzing.

Communication from the field to the operations central happens via the power efficient NB-IoT protocol. The power consumption of the device is now 16 µA during sleep, and an average of approximately 20 mA while awake. With an average connection time to the mobile network of 10 seconds, the node is awake about 13 seconds every 15 minutes. Since the battery used as of today has a capacity of 6000 mAh, the battery life of the system without the use of a solar panel should be two years and two months. However, the prototype have shown that the energy calculation is positive when a solar panel is attached.

MiniMåler'n weighs 212 grams including the battery and the solar panel. All the electronics can fit inside the dimensions of 100 x 100 x 53 mm of the original custom designed enclosure. However, to give more room for the operator to work inside the enclosure, and to make it easier to spot, a new enclosure have been designed with the dimensions 140 x 204 x 53 mm. Whether this new enclosure will be used or not, haven't been decided yet. A pre-made IP67 rated enclosure will be used at least for the first five nodes to be placed in the field. The combined weight of the electronics, the battery, the solar panel and the IP67-enclosure, is 594 grams.

The user interface consists of two slide-switches, one rotary encoder which also includes a push-button, and a display. One of the slide-switches is used to turn on and off the power to the micro-controller, while the other one chooses which mode the system should be in. If the mode switch is in the *Calibrate* position, the user must enter a password before being allowed to calibrate the depth measurements. The password-input and the calibration is done using the rotary encoder. Once the calibration is done, the interface will continue to a GPS information screen. If the user wants to skip this part, he/she must slide the mode switch to the *Normal* position before continuing. Otherwise the GPS will try to connect to satellites. Once the GPS has a good enough accuracy (contact with six satellites) or the user chooses to abort half way through, the system will store the latitude and longitude. If the user haven't aborted the GPS search, the mode switch will still be in the *Calibrate* position. The interface will tell the user to flip it back to the *Normal* position. Once this is done, the display says *thank you and goodbye* to the user before modifying some internal settings, setting the alarm and going to sleep. If the mode switch already is in the *Normal* position when the system is powered, it will try to register with the mobile network, display the signal quality on the screen, and then go straight into its normal operation after modifying some internal settings and setting the alarm.

The total cost for the parts to make one node is 1667 NOK. Table 1 in appendix D shows the prize of all the components used to build one single node. Weight, dimensions and cost all excludes the OTT PLS depth sensor and the SIM card subscription, since these must be bough separately no matter what system is being used.

Other features on this system includes temperature measurements, local data logging to a microSD card, date- and time stamping of data, battery voltage measurement, JSON formatting of the data, connection timeout and re-sending of unsent data. Data such as depth measurement compensation, latitude and longitude are stored in a non-volatile memory so that they aren't lost if the microcontroller loses power or is reset.

17.2 Further plan and work

Long-term testing: Five new prototypes of MiniMåler'n, with all the features included, will be deployed in Setesdalen by the end of June 2019. Hopefully the agreement with Telenor will be in place by then. Although everything seems to be working fine with the for-now finished system, long term testing is still required. These five new prototypes will give new insight to how the system works over a longer period of time, through various seasons.

Pre-soldered PCB: The components on the PCB shield now have to be soldered on manually. Several manufacturers, like Seeed Studio, offers PCB assembly services. If AE are going to mass produce this system, it might be advantageous ordering the PCB pre-soldered.

System II: The multi-node setup

18 Plan, overview and choice of microcontrollers

This is the second depth measuring system that Agder Energi want to develop. The goal of the system is much the same as the previous one; do depth and temperature measurements every 15 minutes and send the data back to AE's servers. The main difference is that this system will be able to operate where there aren't any mobile network reception. Although the end goal is the same, measure and send data, the new requirements imposes new technical challenges.

The plan is to use two or more nodes in this system. One of them is the sensor node. This node will do the measurements at the desired measuring location where there aren't any mobile reception. When the measurements are done, the node will send the data onward via radio communication to a gateway node stationed at a location with mobile reception. This node will then send the data to Agder Energi's servers via NB-IoT. Repeaters can also be used in between the sensor and the gateway if the distance is too long for them to communicate directly. The focus will however be on the sensor node and gateway node for now. The list below presents criteria and challenges for the system.

- Measure depths
- Measure temperature
- Do measurements every 15 minutes
- Have a user friendly interface
- Use little power
- Be small and light
- Store data locally
- Send data from the gateway node to Agder Energi via NB-IoT

New criteria and challanges:

- Send data from the sensor node to the gateway node via radio
- Re-send data if the gateway node doesn't acknowledge the message
- Ensure that the two nodes are synchronized at all times
- Ensure that the two nodes have contact between them
- Make the deployment procedure as convenient as possible

Some of the challenges presented here have been solved in parallel with the development of the single-node system, since they were solvable with the same technical solutions. Unfortunately there isn't too much time left before the deadline of the project. The development of this multi-node system will therefore most likely not be concluded. The ground work will however be laid down for a continuation of the development, and further remaining work will be discussed at the end.

As with the single-node system, the process will start of by choosing an appropriate microcontroller for the nodes.

18.1 Microcontrollers, IDE and language

The sensor node and the gateway node will have different tasks they need to perform, and they will thus have different requirements. Both the microcontrollers as well as all other components will be chosen separately for each node, in order to choose the ones that are most suited for the specific tasks.

18.1.1 Gateway node

The gateway node will send data back to AE via NB-IoT. The SODAQ board used in the single-node setup, is, as previously discussed, a good NB-IoT platform. A lot of the code that has already been written for the single-node system can probably be re-used here, either directly or with minor tweaks. The SODAQ board, and thus the on-board SAMD21 microcontroller, will therefore be chosen for the gateway node. Also the Arduino IDE and its compiler will be used here to program this node.

18.1.2 Sensor node

Since the sensor node will be stationed somewhere without mobile reception, there is no use for the NB-IoT modem found on the SODAQ board. So continuing to use the SODAQ board will add an unnecessary expense to the project. The SODAQ board will therefore not be used here. A suitable option for the microcontroller is the ATmega328P. This is the same microcontroller as the one used in the specialization project. The microcontroller was chosen for the specialization project because of its low power consumption, its useful integrated hardware (UART, I^2C, SPI) and its low cost (16 NOK). The 8-bit microcontroller has 32KB flash memory, 1024B EEPROM and 2KB SRAM. Using the Arduino language and IDE along with its libraries works well for small hobby projects. However, the microcontrollers limited resources, especially the RAM, can easily overflow when using the non-specialized libraries for larger projects. Instead, the code can be written from scratch in the low-level AVR-C language. By using AVR-C to write the code for this project's specific tasks, limited resources shouldn't be a problem with this microcontroller. The RAM consumption was, as previously mentioned, reduced from 82 % to 17 % for the code in the specialization project by switching from Arduino C++ to AVR-C. Resource usage must of course be kept in mind when writing the code in AVR-C in order to achieve this low resource usage.

Although it's possible to write code in AVR-C for the ATmega328P in the Arduino IDE, a better option is to use Atmel Studio. Atmel Studio is an IDE created by Atmel (which is now a part of Microchips Technology Inc.), the creators of the AVR microcontrollers. The advantage of using Atmel Studio is that it has a more practical graphical interface[20], it has auto-completion for the code, and it properly highlights the microcontroller-specific syntax. Since the microcontroller doesn't have an Arduino bootloader, an in-system programmer must be used to flash the code onto the chip. The in-system programmer that will be used here is the AVR Dragon. The AVR Dragon is a relatively cheap programmer that also can be used as a debugger (Microchip Technology Inc. 2018c). The SPI protocol (the same protocol used by the microSD card) is used to flash programs onto the microcontroller. Both the ATmega328P and the AVR Dragon is shown in figure 35.

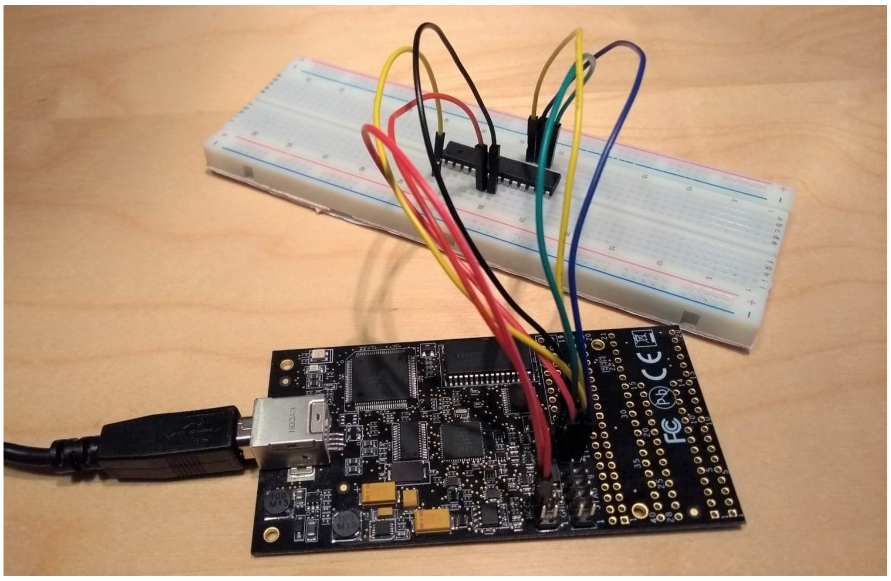


Figure 35: The in-serial programmer AVR Dragon connected to the ATmega328P.

[20]This is a personal opinion.

Some of the code written in the specialization project can probably be re-used for this node, since many of the peripheral units most likely will be the same. However, much work still remains. Both the structure of the main program as well as subroutines, must be modified to fit the tasks of this new system. Extra components for user interface, local storage, solar panel, charge controller, step-up converter etc. that was not used in the specialization project must be introduced here.

19 Basic physical setup

The SAMD21 mounted to the SODAQ board includes all the necessary support components for the microcontroller to work. The standalone ATmega328P on the other hand needs a couple of components and wires to ensure a stable operation. The ATmega328P and its support components are shown in figure 36. The 100 µF capacitor provides a stable main power voltage to the microcontroller. Pin 1 (the pin numbering starts at 1 in the bottom left corner, and increases counter clockwise) is the reset pin. If this pin is pulled low ($0V \leq X \leq 0.3V_{\text{CC}}$) for more than one clock cycle, the microcontroller will reset (Microchip Technology Inc. 2018a). A 10 $k\Omega$ pull-up resistor is therefore added here, in addition to the internal pull-up resistor which also will be activated. Pin 8 and 22 connects each side of the board to ground. There is an internal connection between the grounds of both sides, but this line has a 2 Ω resistance. Although it's not much, it's better to avoid the extra resistance if possible. Pin 7 is the V_{CC}, and will therefore be connected to the positive power supply. Lastly, pin 20, AV_{CC}, is the power supply to the internal analog-to-digital converter, and pin 21, A_{REF}, is the ADC's reference voltage. These are also connected to the main power supply.

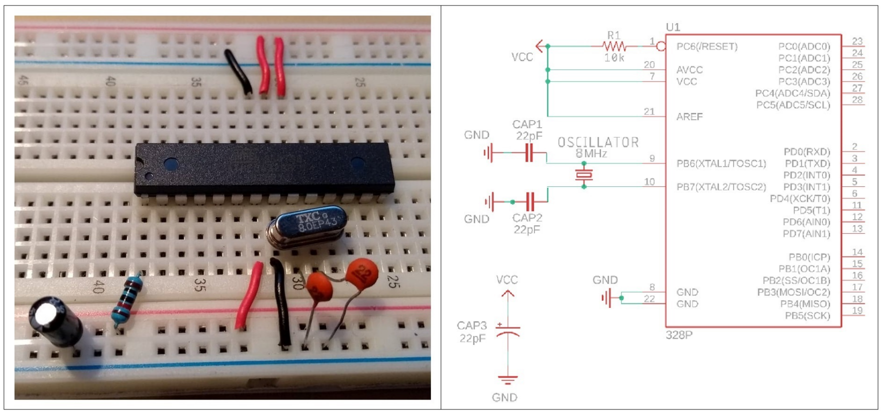


Figure 36: The basic setup of the ATmega328P. This schematic is made using Eagle.

To increase and stabilize the clock frequency, it's normal to include a quartz crystal (often 16 MHz as the one used by the Arduino Uno) and two 22 pF capacitors on pin 9 and 10 of the microcontroller. Initially, the plan was to not include these components here. The microcontroller could instead use it's own internal 8 MHz clock, which is fast enough since none of the planned tasks are very demanding on the processor. However, if non-synchronized communication protocols[21] are to be used, i.e. UART or SDI-12, the internal 8 MHz clock might be too off its targeted frequency for the communication to work. The ATmega328P's datasheet specifies that the factory calibration can only guarantee a ±10% accuracy. UART can allow a half bit error on the last bit (Cook 2012). This means that it requires a 5% accuracy with a 10-bit transmission word. If the factory calibration is more than 5% off, the data will be corrupted. The plan was to not include an external crystal because it will consume some extra power, and the standard 16 MHZ crystal would reduce the possible operating voltage range

[21]Where two devices communicates without a shared time pulse. An agreed upon baud rate must be used instead.

of the microcontroller as can be seen from the graph in figure 37. However, seeing that one of the microcontrollers used in this project have been more than 5 % off in their calibration, it was decided that it is worth including a crystal. Having to check each microcontroller and potentially having to calibrate some of them would introduce extra work for AE in the future. An 8 MHz crystal oscillator will be used in order to keep the high operating voltage range.

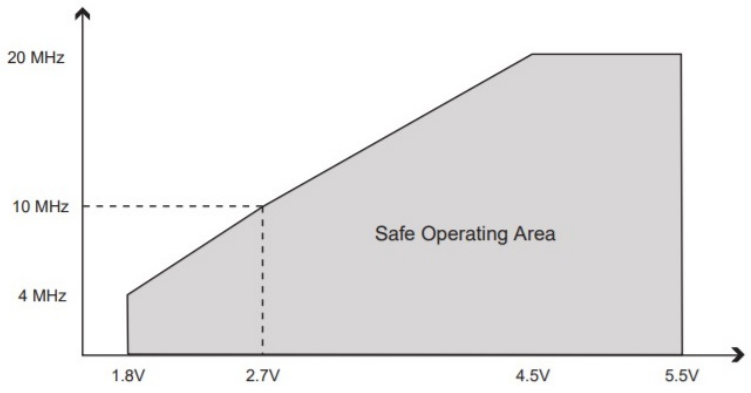


Figure 37: The accepted voltage range of the ATmega328P at different clock frequencies. The image is taken from the ATmega328P's datasheet (Microchip Technology Inc. 2018a).

20 Peripheral units

This chapter will discuss the choices of the peripheral units, how they are wired as well as briefly discuss their coding for initialization and communication with the microcontroller. Several of these aspects have been gone through more in-depth earlier. The communication with-, and wiring of the peripherals will therefore be discussed relatively briefly here. The components presented below are the once that have been found suiting to fulfill the requirements presented earlier. A complete schematics of the sensor node will be presented in figure 41 in the last subsection of this chapter.

20.1 User interface

As with the single-node system, this system also needs a user interface. The same SSD1306 display and the same KY-040 encoder will be used here. Both the gateway node and the sensor node will be given a display, but only the sensor node will be given an encoder. This is because no calibration is needed on the gateway. All user inputs on the gateway node can be done via the power- and mode-switches, which will be included om both nodes.

The user interface code for the ATmega328P looks quite similar to the one on the SAMD21. The biggest difference is how the communication protocol works "under the hood". Where the I^2C on the SAMD21 is controlled by an external library, the setup for the communication will be done from scratch on the ATmega328P. The datasheet of the microcontroller shows which bits in which registers to toggle to get the desired functionality. One thing to note is that the term I^2C is a registered trademark by Philips. Microchips Technologies have therefore avoided using this term in their datasheets. Instead they have created a protocol called *Two wire interface* (TWI), which is fully compatible with I^2C. This report will continue to use the term I^2C because it is the most common term used in other literature, and because the datasheets of the peripheral units uses this name for the protocol.

Contrary to the gateway node, the sensor node don't need a MOSFET to control the powering of the display. The ATmega328P can deliver 20 mA through the GPIO pins, which is enough to power the display. Instead of setting all high when the module is not in use, all pins will instead be set to low (GND). This is because of the external RTC, as will be explained shortly. A GPIO pin will therefore control the V_{IN} pin of the display, while GND of the display is connected directly to the common ground. The rest of the wiring for both the display and the encoder can be seen in the schematics in chapter 20.8.

20.2 Real-time clock

Both the gateway node and the sensor node needs to be able to keep track of the time quite accurately. If their clocks are synchronized, then the time window that the gateway needs to be awake to listen for incoming messages from the sensor node can be shorter. Unlike the SAMD21, the ATmega328P doesn't have an internal RTC. The RTC chosen for the ATmega328P in the specialization project will also be used here. This is the DS3231 by Maxim Integrated mounted, on a breakout board created by Adafruit. Normal crystal oscillators will have a varying frequency with varying temperature. The DS3231 however has an integrated temperature compensated crystal oscillator which is able to give it an accuracy of $\pm$3.5 parts per million from -40°C to +85°C (Maxim Integrated 2015). An integrated calendar includes leap year compensation up to the year 2100, and the alarm can be set to trigger a falling signal on any combination of matching seconds, minutes, hours, date, month or year. This falling signal can then be used to trigger an interrupt on the microcontroller to for example wake it from sleep. Communication between the RTC and the ATmega328P happens via the I^2C protocol, the same protocol used for the display.

The DS3231 can be powered from two different sources. The V_{IN} pin on the breakout board is the main power supply. The RTC must be powered via this pin when communication between the microcontroller and RTC is needed. The secondary power supply is the BAT pin. When the RTC is powered only through this pin, not the V_{IN}, it reduces the power consumption to a minimum. This reduces it's power consumption from about 70 µA to about 1 µA. An important thing to note is that the RTC by default won't keep the oscillator running when powered only through the BAT pin, even though it's crucial for timekeeping. The functionality of keeping the oscillator running, must be set manually by setting the EOSC bit to 0 in the control register.

Since the RTC needs to be powered even when it's not directly in use, the ground pin of the RTC must also be connected to common ground all the time. This means that the strategy of setting all pins high when not in use, as done previously, can not be done here. Instead, all pins must be set low except the BAT pin. This has shown to reduce the current consumption in the low power state from 300 µA due to current leakage when all pins except ground are set high, down to the expected 1 µA when all pins except BAT are set low. Since this RTC uses the I^2C communication protocol, it means that all other devices connected to the ATmega328P via the common I^2C bus must use the same strategy by setting all pins to low when not in use.

20.3 Temperature sensor

Only the sensor node will have a temperature sensor. The sensor to be used will be the BMP280, which were also used for the single-node system. This sensor has not yet been configured for the ATmega328P. However now that the code for the I^2C protocol have been written, and convenient start-, send-, receive- and stop-functions have been created, it's easy to copy the getTemperature() function from the single-node system and replace the I^2C function with the ones for the ATmega328P. Nothing else needs to be done code-wise to get the temperature. The wiring will almost be the same as previously, except that V_{CC} now will be connected to a GPIO pin instead of GND. This is so that all pins can be set to low instead of high when not in use due to the requirements from the RTC, which uses the same I^{2}C bus.

20.4 Depth sensor

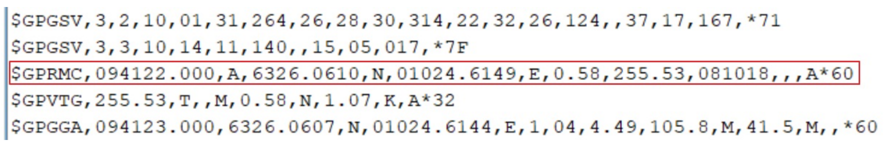
The OTT PLS depth sensor will be used for the sensor node here as well. Since the ATmega328P doesn't have dedicated hardware for the SDI-12 protocol either, the protocol must be implemented via software. Luckily, this work doesn't have to be done from scratch. A programmer named Frank Natvidad has rewritten the Arduino-SDI-12 library to work on the ATtiny85 microcontroller (Natividad 2015). The ATtiny85 also uses the AVR architecture, which means that it's easy to rewrite Natvidad's modified code to work on the ATmega328P as well. Some register names and addresses needs to be changed to correspond to the ones of the ATmega328P. Both Natvidad's library, and the one it's based on, is free to modify and use as long as the modified code is shared.

The physical setup will almost be the same as with the single-node system. Since the step-up converter already have been proved to work successfully, it will be used here as well.

20.5 GPS

Both the sensor node and the gateway node will use GPS to acquire information about their locations. The GPS mounted on the SODAQ board will be used on the gateway node since it's already there and it already has been configured. The GPS to be used on the sensor node will be the same as the one used in the specialization project. This is the FGPMMOPA6H by GlobalTop Technology Inc., which again is mounted on an Adafruit breakout board for easy access to it's ports. This GPS has an integrated $15 \times 15 \times 2.5$ mm ceramic patch antenna (GlobalTop Technology Inc. 2011). The current consumption of the GPS is usually between 25 and 30 mA, but the consumption will at times peak above 40 mA. So even the ATmega328P can't power it using it's GPIO pins. So a MOSFET must be included to cut the ground connection when not in use.

Communication between the GlobalTop GPS and the microcontroller happens via the UART protocol. Since the ATmega328P has integrated hardware for the UART protocol, the initial setup can be done using the same logic as when setting up the I^{2}C protocol. This GPS by GlobalTop doesn't need any input signal or commands before transmitting information to the microcontroller. When it's powered up, it immediately starts transmitting the available data from it's transmit (TX) pin to the microcontrollers receive (RX) pin. Data is sent as ASCII characters in the The National Marine Electronics Association (NMEA) format. The NMEA data is organized in five categories, and the information useful for this project can be found in the Recommended Minimum Navigation Information (RMC) category. Figure 38 shows the output from the GPS, and the RMC category marked in red.



```
$GPGSV,3,2,10,01,31,264,26,28,30,314,22,32,26,124,,37,17,167,*71
$GPGSV,3,3,10,14,11,140,,15,05,017,*7F
$GPRMC,094122.000,A,6326.0610,N,01024.6149,E,0.58,255.53,081018,,,A*60
$GPVTG,255.53,T,,M,0.58,N,1.07,K,A*32
$GPGGA,094123.000,6326.0607,N,01024.6144,E,1,04,4.49,105.8,M,41.5,M,,*60
```

Figure 38: The data transmitted to the microcontroller by the GPS. The content in the red box is the data of interest.

The output data is not a fixed-length string. Instead, the latitude and longitude data must be extracted using the commas as position indicators. Once the GPS has satellite connection and the data is valid (indicated by the 'A' close to the end of the RMC category string), the extraction can begin. All the code for the GPS is put inside a function called getGPSData(). This function has no return value, but all its arguments are taken in as references. This way the global variables latitude and longitude can be altered directly inside the function. See GPS.h and GPS.cpp for the full details of the code.

20.6 Radio communication

The two nodes will communicate with each other using radio communication. The radio drivers that has been chosen is the LoRa SX1276 by Semtech. These drivers uses Semtech's patented spread spectrum modulation technique to provide long range communication with high interference immunity, while at the same time use little power (Augustin et al. 2016). Self-performed tests show that these drivers can reach a range of more than 9 km while still using omni-directional antennas[22]. The range and location of this test can be seen in figure 39. Current consumption is about 120 mA when sending and 20 mA while listening. In accordance with Fribruksforskriften (see chapter 2.3.2), the drivers can use its maximum power of 100 mW at the compatible frequency of 870.1 MHz.

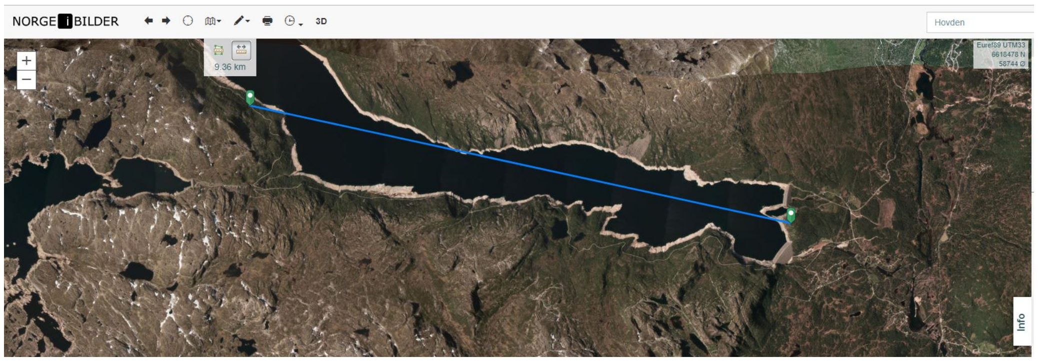


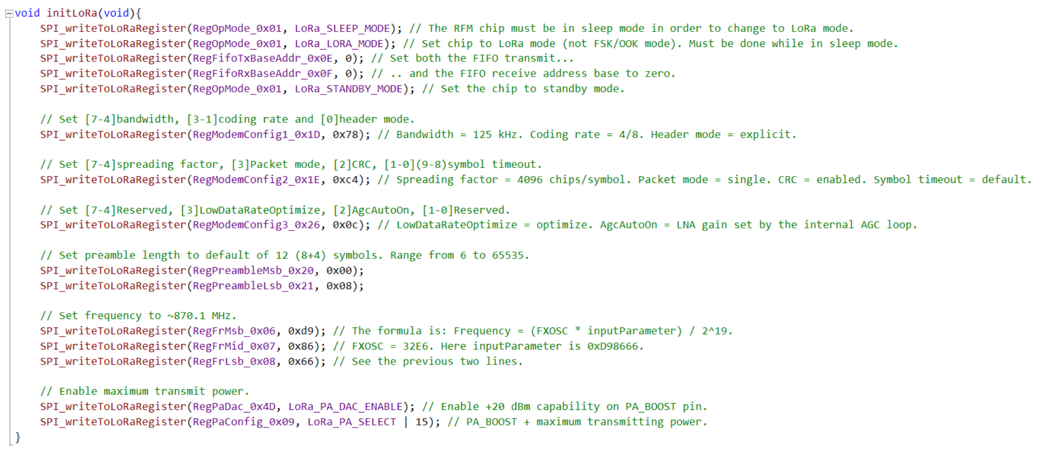
Figure 39: At a range of 9.36 km, the LoRa radio drivers were still able to communicate with each other. The lake seen here is Vatnedalsvatn in Setesdalen.

In terms of programming, the radio driver is the most complex of all the peripheral units. The driver uses the SPI communication protocol for communicating with the microcontroller. A protocol which by itself is convenient to use once the initialize-, read- and write-functions has been created. The tricky part is to figure out the proper way to handle the radio driver. A 132 page datasheet can be downloaded from the producers product page (Semtech 2019). This datasheet presents a lot of considerations that must be taken when initializing and using the driver. Contrary to most datasheets, this datasheet can not simply be used as a look-up table. To understand how to operate the driver, the datasheet must be read from start, up to page 89. Some of the content might however already be familiar to experienced radio programmers.

On the ATmega328P, both the communication protocol and the code for handling the radio driver will be written from scratch. The code snipped in figure 40 shows how the chosen settings for initializing the LoRa radio driver on the ATmega328P.

These settings puts the driver in its most powerful and most robust (most immunity to interference) state. It also makes the transmitting of data quite slow. Sending 16 bytes of data with these settings takes about 1150 ms. When deciding which settings to choose, there are two factors that needs special consideration. These are the time usage and the power usage. If these radio drivers were to be used to for example control a drone, the time usage of each transmission would be critical. Here however,

[22]At 90 cm, these antennas were however longer than the 19 cm ones to be used in this system.



```c
void initLoRa(void){
    SPI_writeToLoRaRegister(RegOpMode_0x01, LoRa_SLEEP_MODE); // The RFM chip must be in sleep mode in order to change to LoRa mode.
    SPI_writeToLoRaRegister(RegOpMode_0x01, LoRa_LORA_MODE); // Set chip to LoRa mode (not FSK/OOK mode). Must be done while in sleep mode.
    SPI_writeToLoRaRegister(RegFifoTxBaseAddr_0x0E, 0); // Set both the FIFO transmit...
    SPI_writeToLoRaRegister(RegFifoRxBaseAddr_0x0F, 0); // .. and the FIFO receive address base to zero.
    SPI_writeToLoRaRegister(RegOpMode_0x01, LoRa_STANDBY_MODE); // Set the chip to standby mode.

    // Set [7-4]bandwidth, [3-1]coding rate and [0]header mode.
    SPI_writeToLoRaRegister(RegModemConfig1_0x1D, 0x78); // Bandwidth = 125 kHz. Coding rate = 4/8. Header mode = explicit.

    // Set [7-4]spreading factor, [3]Packet mode, [2]CRC, [1-0](9-8)symbol timeout.
    SPI_writeToLoRaRegister(RegModemConfig2_0x1E, 0xc4); // Spreading factor = 4096 chips/symbol. Packet mode = single. CRC = enabled. Symbol timeout = default.

    // Set [7-4]Reserved, [3]LowDataRateOptimize, [2]AgcAutoOn, [1-0]Reserved.
    SPI_writeToLoRaRegister(RegModemConfig3_0x26, 0x0c); // LowDataRateOptimize = optimize. AgcAutoOn = LNA gain set by the internal AGC loop.

    // Set preamble length to default of 12 (8+4) symbols. Range from 6 to 65535.
    SPI_writeToLoRaRegister(RegPreambleMsb_0x20, 0x00);
    SPI_writeToLoRaRegister(RegPreambleLsb_0x21, 0x08);

    // Set frequency to ~870.1 MHz.
    SPI_writeToLoRaRegister(RegFrMsb_0x06, 0xd9); // The formula is: Frequency = (FXOSC * inputParameter) / 2^19.
    SPI_writeToLoRaRegister(RegFrMid_0x07, 0x86); // FXOSC = 32E6. Here inputParameter is 0xD98666.
    SPI_writeToLoRaRegister(RegFrLsb_0x08, 0x66); // See the previous two lines.

    // Enable maximum transmit power.
    SPI_writeToLoRaRegister(RegPaDac_0x4D, LoRa_PA_DAC_ENABLE); // Enable +20 dBm capability on PA_BOOST pin.
    SPI_writeToLoRaRegister(RegPaConfig_0x09, LoRa_PA_SELECT | 15); // PA_BOOST + maximum transmitting power.
}
```

Figure 40: Initialization of the radio driver on the ATmega328P. The SPI_writeToLoRaRegister() function is a routine using the SPI protocol in order to use fewer lines per LoRa-command.

one second give or take for the transmission is nothing to worry about. The most important thing is that the message gets received. The extra power consumption from the extra second of transmission shouldn't be anything to worry about either if this system performs as well as the single-node system in terms of power consumption. For the SAMD21, there exists pre-made libraries for both the SPI protocol and for controlling the radio driver. It will use the standard Arduino SPI library as well as a library for the radio driver called RadioHead. This library will have to be modified to include all the functionality desired for this project. Luckily the RadioHead library is offered under a free GPL V2 license, which means that it can be used and modified as long as the code is shared (Free Software Foundation Inc. 2018). As with all the code written in this project, the files can be found publicly on *https://github.com/andremoa/depthStations*.

20.7 Local data logging

Local data logging will only be done on the sensor node. The only additional information the gateway node provides is its own location, which is static information. Local data logging could also happen on the gateway node instead of on the sensor node, but then there might be a chance that data is lost if the radio communication on one or both of the nodes breaks down for some reason.

Both the radio driver and the microSD card uses SPI for communication. This isn't a problem since the SPI bus allows for multiple devices to be attached at the same time. Instead of using addresses for for choosing which device to address, like the I^2C protocol does, SPI uses a chip select line (often called slave select) to tell the devices which device the microcontroller (called the master) intends to communicate with. The chip select line is pulled low on the active device, while the rest must be pulled high. No functions related to storing data on the microSD card have been written yet. The microSD module will still be included in the schematics in order to show how it can be included in the system.

20.8 Multiplexer and the complete schematics

The ATmega328P have a total of 28 pins, whereof 8 are reserved for the reset pin, the crystal oscillator, the reference voltage, power and ground. This leaves 20 pins as GPIO pins. 20 pins are not enough for all the peripherals to directly connect to the microcontroller. A solution to this problem is to use a multiplexer. A multiplexer provides the ability to send one common digital input signal out to one out of many output ports, or vice versa (Crowe and Hayes-Gill 1998). Most multiplexers, including the CD4051B that will be used here, are designed for carrying signals, not high currents for powering

(Texas Instruments 2018b). This means that the multiplexer can't directly be used to provide power to the peripherals. It can however be used to give either a high or a low signal the gate of the MOSFETs, either saturating or desaturating the MOSFETs. A multiplexer connects the chosen output pin to the common input signal, while all other output pins are disconnected. They're not pulled high or low. The gate of a MOSFET must be given a high logic voltage to be saturated, or a low logic voltage to be desaturated. Leaving the gate floating will result in an unpredictable saturation state. Since the default state of all the MOSFETs are desaturated, the multiplexer will be used to saturate the gates one by one, since only one can be chosen at a time. A pull-down resistor of 100 $k\Omega$ will be used to pull the gates low when the pins would otherwise be floating. The multiplexer is given the common input signal of 4.5 V from V_{CC}. The resistor values are chosen to be large enough to not lead too much current when the multiplexer connects 4.5 V to it, while at the same time be small enough keep the MOSFETs desaturated when they're not connected to 4.5 V.

The gateway node only needs to be connected to two peripheral units; the display and the radio driver. A multiplexer is therefore not needed here. The schematics of the gateway node can be seen in appendix C, while the schematics of the sensor node can be seen if figure 41, and in a larger scale in appendix B.

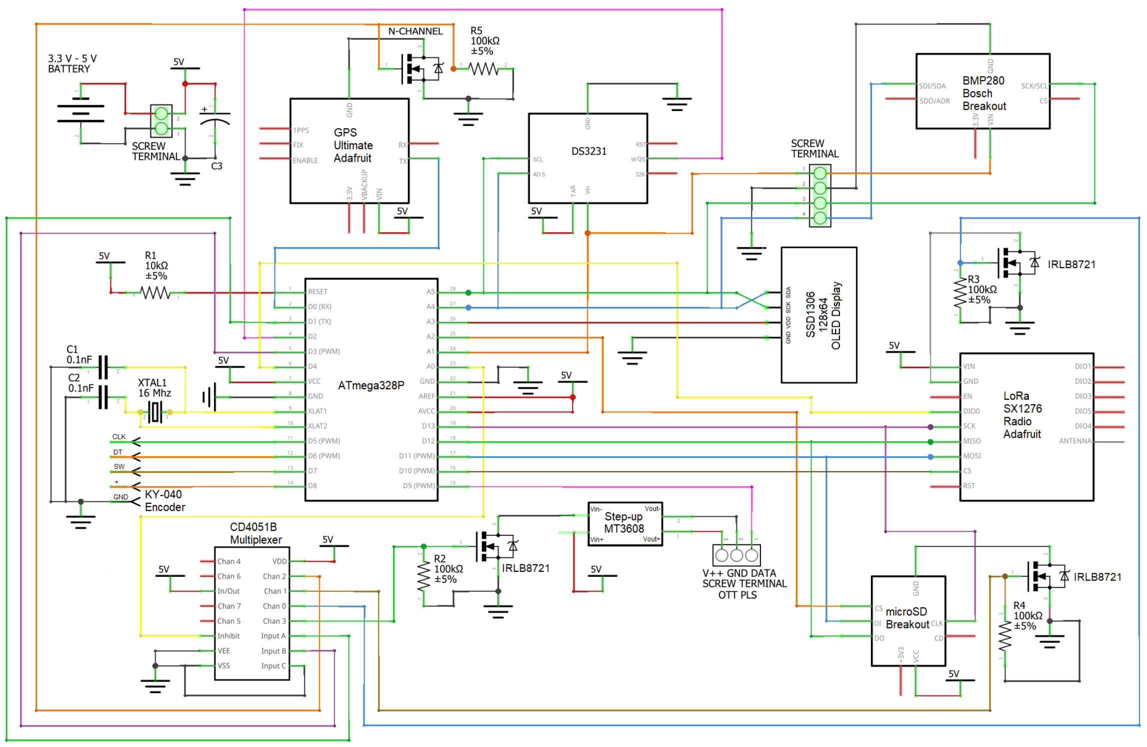


Figure 41: The preliminary schematic for the sensor node. A larger format of this schematic can be found in appendix B. The schematic is made using Fritzing.

21 Functionality and the procedure of deploying the nodes

The procedure of deploying the nodes should be made as simple and effective as possible. For the single-node system, this was relatively easy since everything happens at one location. Here however, there are two nodes at different locations that are going to communicate with each other. The gateway node must be placed somewhere with mobile network coverage, while the sensor node must be at a point of interest for depth measurements. At the same time, one must know that these two nodes are able to communicate with each other via radio communication. The clocks of the two nodes must also be synchronized so that both nodes will wake up from sleep at the same time every fifteen minutes.

21.1 Step 1: Placing the gateway

The gateway needs to be placed somewhere with mobile network coverage. In the same way as with the single-node setup, the function showSignalStrength() will be used display the signal quality on the screen. The operator will have to put the *Mode switch* in the *Normal* position before flipping the *Power switch* to access this feature. A good location for the gateway should be as close as possible, and preferably in line of sight to the desired position of the sensor node. Once a good location has been found, the gateway can be fastened to a pole, a tree or something similar.

21.2 Step 2: Establishing continuous contact between the two nodes

AE says that the deployment of a multi-node system like this normally would require two persons; one at each node (Bjønnes 2019). However, if this procedure could be made so that only one person were required, it would be considered a major advantage. This is what will be attempted here.

When the gateway node has been placed in its final position, the *Mode switch* can be flipped into position two, which for this system will be be called the *Deployment* position. This is the last time the operator have to touch the gateway node. Once the switch is in this position, the microcontroller will initiate the LoRa radio driver and start listening for incoming messages via the radio driver. The sensor node can now be powered with its *Mode switch* also in the *Deployment* position[23]. This will make the senor node start sending messages every two seconds via its radio driver. This message contains the ID of the sender (sensor node), the ID of the intended receiver (gateway node) as well as a short meaningless string of numbers, which for now are all equal to 255. When the gateway node receives this message, it will acknowledge it with its own message.

When the sensor node receives the acknowledgement, it will display the signal quality of the received message on the display so that the operator at all time can see whether he/she is still within radio range of the gateway node. This screen is shown in image 42. The operator can now start moving toward the desired location of the sensor node. The "ball" to the right on the sensor node's screen will rotate one position every two seconds, i.e. once every time the node is expecting to receive a confirmation message. This is to show the user that the system hasn't run into a glitch if it shows the same signal quality for a longer period of time.

21.3 Step 3: Automatic clock synchronization

As previously mentioned, it's important that the clocks of the two nodes are synchronized at all times. Synchronization is something that the user doesn't have to worry about at all since the system will take care of this on its own. The acknowledgement messages that the gateway node sends while the sensor node is being deployed, contains a time stamp. Upon the reception of the first valid acknowledgement message, i.e. a message that passes the cyclic redundancy check (CRC), the sensor node will set it's RTC equal to this time stamp plus one second, due to the time it takes to transmit the message. All following time stamps received while being deployed will be ignored since the time is already set.

[23]The *mode switch* on the sensor node is for now only a wire that is either disconnected from-, or connected to common ground.

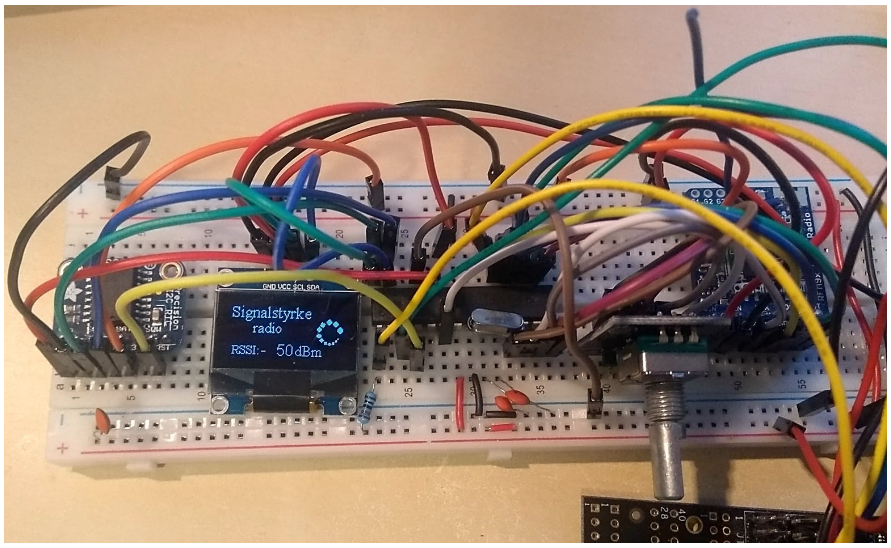


Figure 42: The radio driver on the sensor node (barely showing to the top right) is here communicating with the radio driver on the gateway node, located in another room about 15 meters away. A simple wire antenna is used. The signal strength is updated once per second. The power rail of the breadboard can be connected to a battery, and the whole node can then be moved around to see how the signal quality changes.

When the nodes goes into their normal operation after the deployment, the sensor node will update its RTC once every 15 minutes. After the sensor node has sent its sensor data to the gateway node, the gateway node will acknowledge that it has received the data. This acknowledgement contains the updated time stamp. If the acknowledgement message passes the CRC, the clock of the RTC on the sensor node will be updated.

21.4 Step 4: Remotely putting the gateway node into its normal operation

Once the operator has reached/found a final location for the sensor node, there is no longer a need for the gateway node to continuously keep listening for messages. The operator can now flip the *Mode switch* of the sensor node from *Deploy* to *Normal*. The meaningless string of numbers that has been sent to the gateway node until now, are changed to a defined string of numbers that both nodes are aware of. When the gateway node receives this string, it will acknowledge that it has received it, then listen for 10 more seconds, before setting its alarm for the next whole 15 minute and go into sleep mode. The gateway listens for 10 seconds after acknowledging the message in case the acknowledgement weren't received by the sensor node, and the sensor node tries to send the same message again. The sensor node will send the *go-to-normal-operation-message* until it has received the acknowledgement, or until it hasn't received an acknowledgement for five turns. One weakness of this procedure is if the gateway node don't receive any of the five *go-to-normal-operation-messages*. Then it will keep on listening after the sensor node is done sending. The sensor node could of course acknowledge the acknowledgement, but then the gateway node should acknowledge the acknowledgement of the acknowledgement etc. The two nodes can't bi-laterally know that the other node knows what it knows. On the other hand, if one node sends five messages in a row which the other node can't receive, they are probably too far apart in the given terrain, and one of them should thus be moved. As a final safety, the gateway node will always go into its normal operation if it has been one hour since the last received message.

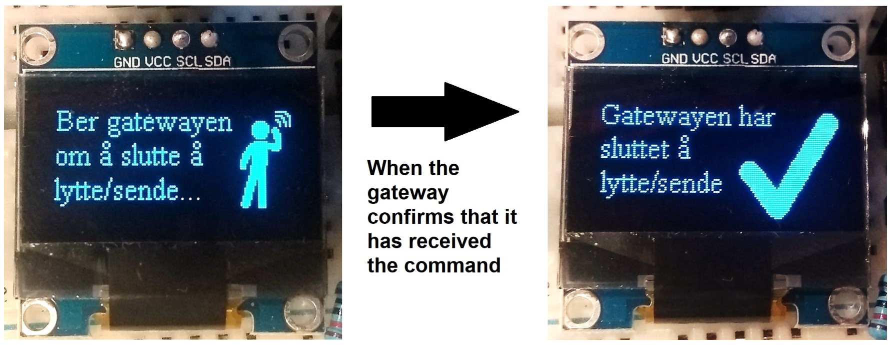

GND VCC SCL SDA
Ber gatewayen om å slutte å lytte/sende...
When the gateway confirms that it has received the command
GND VCC SCL SDA
Gatewayen har sluttet å lytte/sende

Figure 43: The sensor node sends a command to the gateway node to stop listening for/acknowledging messages. The gateway node will acknowledge this command, listen for 10 more seconds, then enter its event loop. The sensor node can now continue to the depth calibration.

21.5 Step 5: Depth measurement calibration and GPS

After the sensor node has told the gateway node to stop continuously listening for messages, and the depth sensor probe has been placed in the water, the depth measurement can be calibrated. This will happen the same way as with the single-node system. The code is a bit difference since it happens on another microcontroller, but from the user's perspective it looks all the same.

When the depth measurement calibration is done, the microcontroller will search for a GPS signal and get the position. Also this will to the user look the same as on the single-node system. The acquisition can be aborted by flipping the *Mode switch* back and forth. Where the GPS data on the SAMD21 are stored in simulated EEPROM in Flash, the data is here stored in actual EEPROM.

21.6 Step 6: Normal operation

If the *Mode switch* is in the correct (*Normal*) position, and the GPS has stored its data, the node sets the alarm of the RTC for the next whole 15 minutes and then goes to sleep.

22 Preliminary results and further work

22.1 Preliminary results

Unfortunately there is not enough time to conclude the work on this multi-node system. The deployment procedure as described in the previous chapter, is finished, all except for a proper GPS implementation. Improvements to the procedure can be made. This will be discussed in the next subsection. Though for now it now works as intended.

The sensor node of this multi-node system uses the ATmega328P microcontroller. In sleep mode, the sensor node uses 4 µA. The microcontroller alone have been registered to use 2 µA in sleep mode, so there is a tiny bit of current leakage somewhere. Many of the components used to make this node are the same as the ones used for the previous system. This includes the display, the encoder and the local storage module, the step-up converter and the temperature sensor. The DS3231 external RTC is used for timekeeping and to initiate an interrupt at the desired times. The SX1276 LoRa radio driver is used for communication with the gateway node.

The gateway node uses the same SODAQ board, and thus the same SAMD21 microcontroller as the node in the single-node system. Several important subroutines for this system have not been implemented yet. This includes routines for lowering the power consumption. The current consumption in sleep mode of this node is measured to be 35 µA. In addition to support components like switches, MOSFETs and capacitors, this node uses only two peripheral modules. These are the SSD1306 dispaly, and the SX1276 LoRa radio driver.

Some routines for the two node's event loops have been written. This includes functions for handling the time and alarm, general purpose sending and receiving functions of data using the radio driver, temperature measurement and depth measurement. These functions have yet to be implemented into the main program.

22.2 Further work

Finish the code and test for stability: The code for both the sensor node and the gateway node of the multi-node system is still unfinished. Although some subroutines for the event-loops of the two nodes have been written, much work still remains. This includes designing the structure and functionality of the event-loops, and writing the necessary subroutines to archive this functionality if they have not already been written. The routines that have been written, have yet to be tested for stability in order to find out how well they perform over a longer period of time, and how they perform together in a complete program.

Replace the multiplexer: A *high load multiplexer* can be used to replace the multiplexer used on the sensor node. The difference between these two multiplexers is that the high load multiplexer can lead high currents, often a few amps (Texas Instruments 2018a). A high load multiplexer would probably be a better suited component than the one used now, since it can directly connect the main power supply of the components to V_{CC}, and thus also make the MOSFETs redundant.

Charging and battery level indication: As the physical setup is now, the sensor node is not compatible with solar panel charging like the gateway node is. A charge management controller and required supporting components can be included in the setup to make this possible. The sensor node can't read the battery voltage either as of today. There are two reasons for this. Firstly, there aren't any unused pins left on the microcontroller. A power switch, as described earlier, can be used to free one of the ATmega328P's analogue ADC pins. Secondly, the ADC reference voltage pin on the ATmega328P isn't fed with any constant reference voltage. This is however easy to fix since several of the peripheral units, like the microSD card module, has a pin-out for the constant 3.3 V from the on-board voltage regulator. The ADC can then be used to read the battery voltage via a voltage divider, in the same way as on the SODAQ board.

Improved placement procedure: In hindsight, it would probably be better to place the sensor node before placing the gateway node. There is often a specific location where it's the most convenient to do the depth measurements, while the placement of the gateway node is more flexible. The procedure of placing the nodes could therefore be reverted so that the sensor node is placed first. Or maybe even better; a feature where the operator chooses which node to place first could be implemented.

Physical design: All the wiring of the components are now done using breadboards. A robust physical design, including a PCB desing, must be created. The circuitry for the gateway node can probably be made as a shield, like with the single-node system. The sensor node on the other hand has a more flexible design since all the components for the system can be included directly on the PCB (excluding the battery, the solar panel, the depth sensor and the temperature sensor.). If AE chooses to produce the custom made enclosures for the single-node system, custom enclosures can be made for this system as well.

Create repeaters: The distance from some desired locations of depth measurement might be in too tough terrain or be too far away from an area with mobile reception for the radio drivers to be able to communicate directly with each other. If this is the case, a repeater, or even several repeaters, can be placed in between the two nodes to receive and re-transmit the data. It would therefore be an improvement to the system if compatible repeaters were created.

Discussion and conclusion

23 Discussion and conclusion

The aim of this master's thesis was to develop a reliable, low powered, user friendly, cheap, small and light depth measuring system for use at remote locations. Agder Energi wanted this system so they could supplement, and to some degree replace, the large, heavy and expensive stations they are using today. By having cheaper and smaller measuring stations, AE can do measurements at more locations than they were able to before (Bjønnes 2019). This will in turn make the operators at AE's control central able to regulate the water flow in the rivers more precisely than they used to. Better regulation of water flow is good for both AE and the community/environment. AE will be able to release less excess water from their dams, staying closer to the healthy lower limit set by NVE, thus being able to send more water through their turbines to increase their power production. AE will also be less likely to breach the lower limit due to their improved overview. Lastly, AE will be better suited to actively prevent flood by acting upon the larger quantity of data available.

The choices made during this development project have been discussed in details throughout the report. This chapter will not repeat this discussion. It will rather discuss based upon the results from the project.

Setups
Two different setups have been under development. A single-node system which is able to operate where mobile network communication is available, and a multi-node system that should be able to operate outside the coverage area of mobile network communication. The single-node system was AE's top priority, while the multi-node system was to be worked on if there were time after the development of the first-priority system. The work on the single-node system is concluded, while the multi-node system is still unfinished. Since only the single-node system have proper results to discuss, it is the only system that will be discussed here. Some of the topics can however relate to both systems. The discussion of the multi-node system alone is for the most part covered under *Further work* in chapter 22.2.

The node of the single-node system have been named *MiniMåler'n*. Although long term testing still remains before a proper conclusion can be drawn on how the final version of the system performs, the preliminary results looks promising.

Reliability
As previously mentioned, the finished system still requires long term testing throughout various seasons in order to determine how well it works. During an early stage test, there were some issues regarding false depth measurements and long network registration times. These issues do however seem to have been solved. After fixing the issues as discussed in chapter 10, they have never appeared since. The first field prototype have now been running for 67 days straight without loosing any message or sending wrong depth data. The connection time is usually between 9 to 11 seconds, with a maximum of about 20 seconds once or twice a week. There were some issues with the temperature data on the field prototype when the temperature went below zero. The temperature was then shown to be very high. This wasn't actually a false measurement. It was merely because the microcontroller wasn't programmed to handle two's complement representation of negative numbers. This issue have been resolved for the later prototypes.

The final version of the system have only been tested for shorter periods of time. No false data, nor any other glitches or bugs, have been registered so far. Five nodes of the final version of MiniMåler'n will be deployed into the field by the end of June 2019. More data regarding this subject will be available after this.

Power consumption
Power consumption have been an important aspect of the system, and a focus have been directed

toward reducing the consumption as much as possible. MiniMåler'n has shown to have a power consumption of 16 µA while in sleep mode. This is actually lower than what SODAQ themselves reported the sleep current of the board alone to be (SODAQ 2018a). The most likely explanation for this is the fact that the pins of the microcontroller are given a defined pin state instead of being left floating. Having a defined pin state has proved to reduce current consumption (Microchip Technology Inc. 2018b). The example sketch SODAQ was using to demonstrate the power consumption, did not define most pin states.

The long-running field prototype shows that the node gets charged by the solar panel more than the power it uses. After being charged the first week, the battery level have been full and stable since. Calculations with the measured current consumption shows that the battery of the node will last for about two years and two months using the 6000 mAh lithium ion polymer battery, without being charged by a solar panel. One of the downsides with the HydMet stations from Scanmatic is that they sometimes experiences power-downs during the winter because their solar panels are covered with snow. This is an issue that shouldn't be relevant to MiniMåler'n since it can operate on battery power alone for a longer period of time than there is snow during the winter.

User experience
The user interface have been designed with the aim of being easy and intuitive to operate. This goes for both the navigation on the display, and the physical handling of the device. All user interaction happens via two slide-switches and one rotary encoder which includes a push-button. The system communicates with the user via a 0.69" OLED display. The goal seems to have been accomplished since AE finds the interface convenient to use. One drawback if the system ever is going to be sold to a consumer market, is that layout of the interface isn't exactly pretty. Both the display, the buttons and the encoder is located so the user can see much of the open electronics of the system while using the interface. Though this shouldn't be of any concern as long as AE is the only one using the system.

User experience extends further than just the interface on the node. The fact that data can be sent in a standard, universally used format, is seen as a major advantage. AE have experienced issues with different equipment using different data formats (Bjønnes 2019). Translator units are often needed in between equipment and the control central, as well as in between equipment that needs to communicate with each other. This translation can at times cause trouble. Also the Scanmatic stations that AE are using today, needs translation by a Scanmatic software before being usable by AE. The data sent from the system developed here is on the standard JSON format. A format that can be read by the software AE already are using.

Further comparison to other systems
The Libelium's Smart Water nodes have a smaller volume than what is practically convenient with MiniMåler'n as of today. This is because user interaction, like using the interface or connection external sensors, must be done from the inside of MiniMåler'ns enclosure. With Smart Water on the other hand, almost all interaction happens from the outside of the enclosure (Libelium 2018). Though at 594 grams including battery, solar panel and enclosure, MiniMåler'n weighs a bit less than the 1 kg Smart Water node. Both nodes do however weigh considerably less than the 32 kg HydMet node.

Just like MiniMåler'n, the HydMet stations sends data from the field to Agder Energi at certain intervals. The main difference between these two systems regarding this subject is that MiniMåler'n does measurements once every 15 minutes and sends the data back right afterwards, while the HydMet stations can continuously monitor the depths before sending a sample of the measurements at given intervals. This adaptable interval is often set to be once per hour. Being able to continuously monitor the depth, as well as having an adaptable transmit-interval, could be an advantage. If the depth is changing very rapidly, the HydMet stations can give a data series with a higher resolution. For monitoring the depths of rivers and reservoirs however, a 15 minute resolution is usually more than good enough for AE (Bjønnes 2019). When monitoring the data from the first field prototype, which is still operating, one can see that the depth is usually changing quite slowly. One exception was during heavy rainfall and snow melting on May 21st. At times the depth changed by as much as 27 mm during 15 minutes. Continuous measurement could be interesting at times like these, but it is however not a necessity.

The price of one MiniMåler'n node is 1667 NOK, which is less than both the 55,000 to 115,000 NOK HydMet node and the 1,000 to 5,250 € Smart Water node. Some of these savings does of course come from the fact that there have been no additional development or marketing cost, and no profit margin, when creating this system. AE will now own this depth measurement system, so there will be no second or third party costs that needs to be covered other than to the retailers of the required components. Owning the entirety of the system has both advantages and disadvantages. The advantage is, as just discussed, the cost. Owning the system also means access to the source code of the microcontroller, making it possible to adapt the features of the system. However, if AE chooses to continue to use the HydMet stations, or chooses to buy the Smart Water system or any other depth measuring system on the market for that sake, they will get a finished and complete product that has customer support if anything is to go wrong. If anything goes wrong with the system they own themselves, AE is the one that needs to find a solution to the problem.

Adaptability
In later discussions with AE, they said that they have thoughts about using the basic design and framework of MiniMåler'n to develop other measurement systems. For example snow measurement, which today is done manually. Doing so does require knowledge in both programming and electronics. AE does however have skilled people within these fields, so expanding upon the framework of the single-node system is achievable.

Concluding words
Agder Energi wanted to create their own measuring station suited for their specific needs. Whereas stations like HydMet and Smart Water are designed to suit different customers with different needs, the system developed in this master's thesis does exactly what AE needs it to do. Nothing more, nothing less. If AE want to add new features, they have the possibility to do so in an even greater extent than with a purchased system. They do however have to arrange for this adaptation themselves. The fact that this system is designed to fulfill AE's exact needs, is an important part to why it was able to become so cheap, small and power efficient as it is.

AE is pleased with the results from this project. Although the multi-node system isn't finished, the ground work for the continuation has been laid down. The single-node system on the other hand, is finished, and long term test of the final setup will show if it performs as well over time as the preliminary results indicates.

References

3GPP (2019). *About 3GPP.* URL: https://www.3gpp.org/about-3gpp.

ABB (2018). *RTU500 series Intelligence distributed across your power grid.* URL: https://new.abb.com/substation-automation/products/remote-terminal-units.

Adafruit (2019). *adafruit$_p$roducts$_s$chem.png.* URL: https://learn.adafruit.com/assets/35635.

Advanced Monolithic Systems (2019). "Adjustable/Fixed Low Dropout Linear Regulator". In: URL: https://static.chipdip.ru/lib/552/DOC001552809.pdf.

Agder Energi (2018). *Om Agder Energi.* URL: https://www.ae.no/konsernet/.

AllThingsTalk (2019). *UDP Messaging API.* URL: https://docs.allthingstalk.com/developers/api/udp-messaging/.

Arduino (2019). *Bootloader Development.* URL: https://www.arduino.cc/en/Hacking/Bootloader?from=Tutorial.Bootloader.

ARM (2007). "Application Note 179 Cortex™-M3 Embedded Software Development". In: p. 147. URL: http://infocenter.arm.com/help/topic/com.arm.doc.dai0179b/AppsNote179.pdf.

Augustin, Aloÿs et al. (2016). "A Study of LoRa: Long Range Low Power Networks for the Internet of Things". In: *Sensors* 16.

Austrud, Bjørn (2018). *Agder Energi Kraftforvaltning.* Personal communication.

BatteriOnline (2019). *Energizer Ultimate Lithium L91 / AA B2B Batterier (10 Stk. Pakning).* URL: https://batterionline.no/energizer-lithium-l91-aa-b2b-batterier-10-stk-pakning?gclid=CjwKCAjw8e7mBRBsEiwAPVxxiB5V0iNMis-wgIa0kiRDmWFmtjMpjfjSACyF-Vf87F8ZFWyAQ-CzIhoC8wsQAvD_BwE.

Bjønnes, Trym (2019). *System engineer and Process control at Agder Energi Vannkraft AS.* Personal communication.

Bosch Sensortec (2015). "BMP280 Digital Pressure Sensor". In: URL: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMP280-DS001.pdf.

Chisholm, John (2014). "What is co-design?" In: *Design For Europe.*

Cook, David (2012). *Timing Errors in Serial Communication.* URL: http://www.robotroom.com/Asynchronous-Serial-Communication-2.html.

Crowe, John and Barrie Hayes-Gill (1998). "Introduction to Digital Electronics". In: Newnes, pp. 88–124. URL: http://www.sciencedirect.com/science/article/pii/B978034064570350006X.

Cunningham, Collin (2018). "Collin's Lab: Breadboards Perfboards". In: *Adafruit learning system.* URL: https://cdn-learn.adafruit.com/downloads/pdf/collins-lab-breadboards-and-perfboards.pdf?timestamp=1559909863.

Dow, Steven P., Kate Heddleston, and Scott R. Klemmer (2009). "The Efficacy of Prototyping Under Time Constraints." In: *Stanford University HCI Group, Department of Computer Science.*

Dynda, Alexey (2019). *SSD1306/SSD1331/SSD1351/IL9163/ILI9341/ST7735/ILI9341 OLED display driver, PCD8544 LED display driver.* URL: https://github.com/lexus2k/ssd1306.

Energizer (2018a). *Cylindrical Primary Lithium - Handbook and Application Manual.* Page 11. URL: http://data.energizer.com/pdfs/lithiuml91l92_appman.pdf.

— (2018b). "ENERGIZER L91 Ultimate Lithium PRODUCT DATASHEET". In: URL: http://data.energizer.com/pdfs/lithiuml91l92_appman.pdf.

Ericsson (2019). *Know the difference between NB-IoT vs. Cat-M1 for your massive IoT deployment.* URL: https://www.ericsson.com/en/blog/2019/2/difference-between-NB-IoT-CaT-M1.

Flowline Inc. (2010). *EchoPod DX10 Ultrasonic Level Sensor (Voltage or Frequency).* URL: https://docs-emea.rs-online.com/webdocs/1425/0900766b814258f4.pdf.

Free Software Foundation Inc. (2018). *GNU Lesser General Public License, version 2.1.* URL: https://www.gnu.org/licenses/old-licenses/lgpl-2.1.html.

GlobalTop Technology Inc. (2011). "FGPMMOPA6H GPS Standalone Module Data Sheet". In: URL: https://cdn-shop.adafruit.com/datasheets/GlobalTop-FGPMMOPA6H-Datasheet-VOA.pdf.

Granger, Joshua (2016). *Boost converter help.* URL: https://electronics.stackexchange.com/questions/273938/boost-converter-help.

HCL Technologies (2019). *WHAT IS AN IOT DEVICE?* URL: https://www.hcltech.com/technology-qa/what-is-an-iot-device.

Henry (2015). *Keyes KY-040 Arduino Rotary Encoder User Manual.* URL: http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/keyes-ky-040-arduino-rotary-encoder-user-manual/.

Høgevold, Knut (2019). *ICT and Process control at Agder Energi Vannkraft AS.* Personal communication.

IBM (2019). *The client/server model.* URL: https://www.ibm.com/support/knowledgecenter/en/SSAL2T_8.1.0/com.ibm.cics.tx.doc/concepts/c_clnt_sevr_model.html.

JLCPCB (2019). *Fast, Reliable Deliveries.* URL: https://jlcpcb.com/.

Johnson, Dr. Howard (2000). "Who's Afraid of the Big Bad Bend?" In: *EDN magazine.* URL: http://www.sigcon.com/Pubs/edn/bigbadbend.htm.

Kim (2018). *IoT in Norway, settings?* URL: http://forum.sodaq.com/t/iot-in-norway-settings/1578.

Kwiecień, Radosław (2019). *LCD Assistant.* URL: http://en.radzio.dxp.pl/bitmap_converter/.

Libelium (2014). *Smart Water Sensors to monitor water quality in rivers, lakes and the sea.* URL: http://www.libelium.com/smart-water-sensors-to-monitor-water-quality-in-rivers-lakes-and-the-sea/.

— (2018). "Smart Water Technical Guide". In: URL: http://www.libelium.com/downloads/documentation/smart_water_sensor_board.pdf.

— (2019). *IoT made easy!* URL: http://www.libelium.com/products/plug-sense/technical-overview/.

Maglie, Cristian (2018). *FlashStorage library for Arduino.* URL: https://github.com/cmaglie/FlashStorage.

Maxim Integrated (2015). "Datasheet: DS3231 Extremely Accurate I2C-Integrated RTC/TCXO/Crystal". In: URL: https://cdn-shop.adafruit.com/product-files/3013/DS3231.pdf.

Microchip Technology Inc. (2014a). "Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers". In: URL: http://ww1.microchip.com/downloads/en/DeviceDoc/20001984g.pdf.

— (2014b). "Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers". In: URL: http://ww1.microchip.com/downloads/en/DeviceDoc/20001984g.pdf.

— (2018a). "ATmega48A/PA/88A/PA/168A/PA/328/P Data Sheet". In: URL: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf.

— (2018b). "ATmega48A/PA/88A/PA/168A/PA/328/P Data Sheet". In: pp. 88–89. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf.

— (2018c). *AVR Dragon.* URL: https://www.microchip.com/Developmenttools/ProductDetails/ATAVRDRAGON#additional-summary.

— (2018d). "SAM D21 Family". In: p. 147. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/SAMD21-Family-DataSheet-DS40001882D.pdf.

— (2018e). "SAM D21 Family (Datasheet)". In: URL: http://ww1.microchip.com/downloads/en/DeviceDoc/SAMD21-Family-DataSheet-DS40001882D.pdf.

Microsoft (2019). *What is Azure?* URL: https://azure.microsoft.com/en-us/overview/what-is-azure/#most-popular-questions.

Moorthy, C.Ganesa, G. Udhaya Sankar, and Raj Kumar (2017). "WHAT IS THE POLARITY OF AN ELECTROMAGNETIC WAVE?" In: *Indian Journal of Scientific Research* 13.

National Instruments (2019). *Understanding Life Expectancy of Flash Storage.* URL: http://www.ni.com/product-documentation/10126/en/.

National Radio Astronomy Observatory (2018). *How Radio Waves Are Produced.* URL: https://web.archive.org/web/20140328011153/http://www.nrao.edu/index.php/learn/radioastronomy/radiowaves.

Natividad, Frank (2015). *SDI-12-ATTiny84a.* URL: https://github.com/frankyn/SDI-12-ATTiny84a.

NKOM (2019). *Hva er frekvenstillatelser.* URL: https://www.nkom.no/teknisk/frekvens/tillatelser/hva-er-frekvenstillatelse-spektrumstillatelser.

OmegaVerkstedet (2019). *Component search: SD.* URL: https://omegav.ed.ntnu.no/komp?search=sd.

OTT HydroMet (2018). "Operating instructions Pressure Probe OTT PLS". In: URL: https://www.ott.com/en-us/products/download/operating-instructions-pressure-probe-ott-pls/.

RICOH Electronic (2018). *Soft-Start Function.* URL: https://www.e-devices.ricoh.co.jp/en/products/power/ap_note/dcdc/softstart.html.

Samferdselsdepartementet (2012). *Forskrift om generelle tillatelser til bruk av frekvenser (fribruksforskriften).* URL: https://lovdata.no/dokument/SF/forskrift/2012-01-19-77.

Scanmatic AS (2018). "Hydrologistasjon". In: URL: https://www.scanmatic.no/energi/hydrologistasjoner/.

— (2019). *Referanser.* URL: https://www.scanmatic.no/referanser/.

Schrage, Michael (1999). "Serious Play: How the World's Best Companies Simulate to Innovate." In: *Harvard Business School Press.*

SD Association (2006). "SD Specifications". In: URL: http://users.ece.utexas.edu/~valvano/EE345M/SD_Physical_Layer_Spec.pdf.

— (2019). *SD Standard Overview.* URL: https://www.sdcard.org/developers/overview/index.html.

SDI-12 Support Group Technical Committee (2017). "SDI-12 A Serial-Digital Interface Standard for Microprocessor-Based Sensors". In: p. 3. URL: http://www.sdi-12.org/specification.php.

Seba Hydrometrie (2019). *FLOAT LEVEL SENSOR / FOR WATER / STAINLESS STEEL / IP65.* URL: http://www.directindustry.com/prod/seba-hydrometrie-gmbh-co-kg/product-63216-1983299.html.

Semtech (2019). "Datasheet: SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver". In: URL: https://www.semtech.com/uploads/documents/DS_SX1276-7-8-9_W_APP_V6.pdf.

Sikken, Jasper (2018). *SODAQ Sara schematics.* URL: http://support.sodaq.com/wp-content/uploads/2018/10/SODAQ-SARA-AFF-Rev3-Schematic-sheet-1.png.

Simpson, Chester (2011). *Characteristics of Rechargeable Batteries.* URL: http://www.ti.com/lit/an/snva533/snva533.pdf.

Smith, Kevin M. (2014). *Arduino-SDI-12.* URL: https://github.com/Kevin-M-Smith/Arduino-SDI-12.

SODAQ (2018a). *SODAQ SARA AFF N211.* URL: https://support.sodaq.com/sodaq-one/sara/.

— (2018b). *$SODAQ_ArcMainCurrent$.* URL: http://support.sodaq.com/wp-content/uploads/2018/03/sodaq_sara_sleep_no_led.png.

— (2019). *SODAQ SARA Arduino Form Factor (AFF) N211 including PCB Antenna.* URL: https://shop.sodaq.com/sodaq-sara-aff-n211.html.

SparkFun Electronics (2010). *SD Library for Arduino.* URL: https://github.com/arduino-libraries/SD.

Stoehr, Martin (2012). *Getting Started With A Short-Range Radio Design.* URL: http://www.electronicdesign.com/communications/getting-started-short-range-radio-design.

Suneco Hydro (2019). *How to Measure Water Flow.* URL: https://www.micro-hydro-power.com/how-to-measure-water-flow.htm.

Taha, Zahari, Hassan Alli, and Salwa Abdul-Rashid (2011). "Users Involvement in New Product Development Process: A Designers' Perspectives". In: *Industrial Engineering and Management Systems* 10.

Techopedia (2019). *Bitmap (BMP).* URL: https://www.techopedia.com/definition/792/bitmap-bmp.

Telia (2019). *DEKNINGSKART.* URL: https://www.telia.no/dekning/.

Texas Instruments (2018a). "Basics of Power MUX". In: URL: http://www.ti.com/lit/an/slvae51/slvae51.pdf.

Texas Instruments (2018b). "CD405xB CMOS Single 8-Channel Analog Multiplexer/Demultiplexer With Logic-Level Conversion". In: URL: http://www.ti.com/lit/ds/symlink/cd4051b.pdf.

— (2019). "LP2981-N Micropower 100-mA Ultralow Dropout Regulator in SOT-23 Package". In: URL: http://www.ti.com/lit/ds/symlink/lp2981-n.pdf.

Tiwana, Amrit (2014). "Platform Architecture". In: *Elsevier B.V.* URL: https://www.sciencedirect.com/book/9780124080669/platform-ecosystems.

TOREX SEMICONDUCTOR LTD. (2019). "XC6220 Series". In: URL: https://no.mouser.com/datasheet/2/760/XC6220-837466.pdf.

Tormatic AS (2019). *UNIK 5000 - Trykksensorserie i 316 SS*. URL: http://www.tormatic.no/prosess/trykk/unik-5000---trykktransmitterserie.

Tracer Power (2019). "LITHIUM ION POLYMER BATTERIES – DATA SHEET". In: URL: http://www.tracerpower.com/user/pdf/Li-Poly_Data_Sheet.pdf.

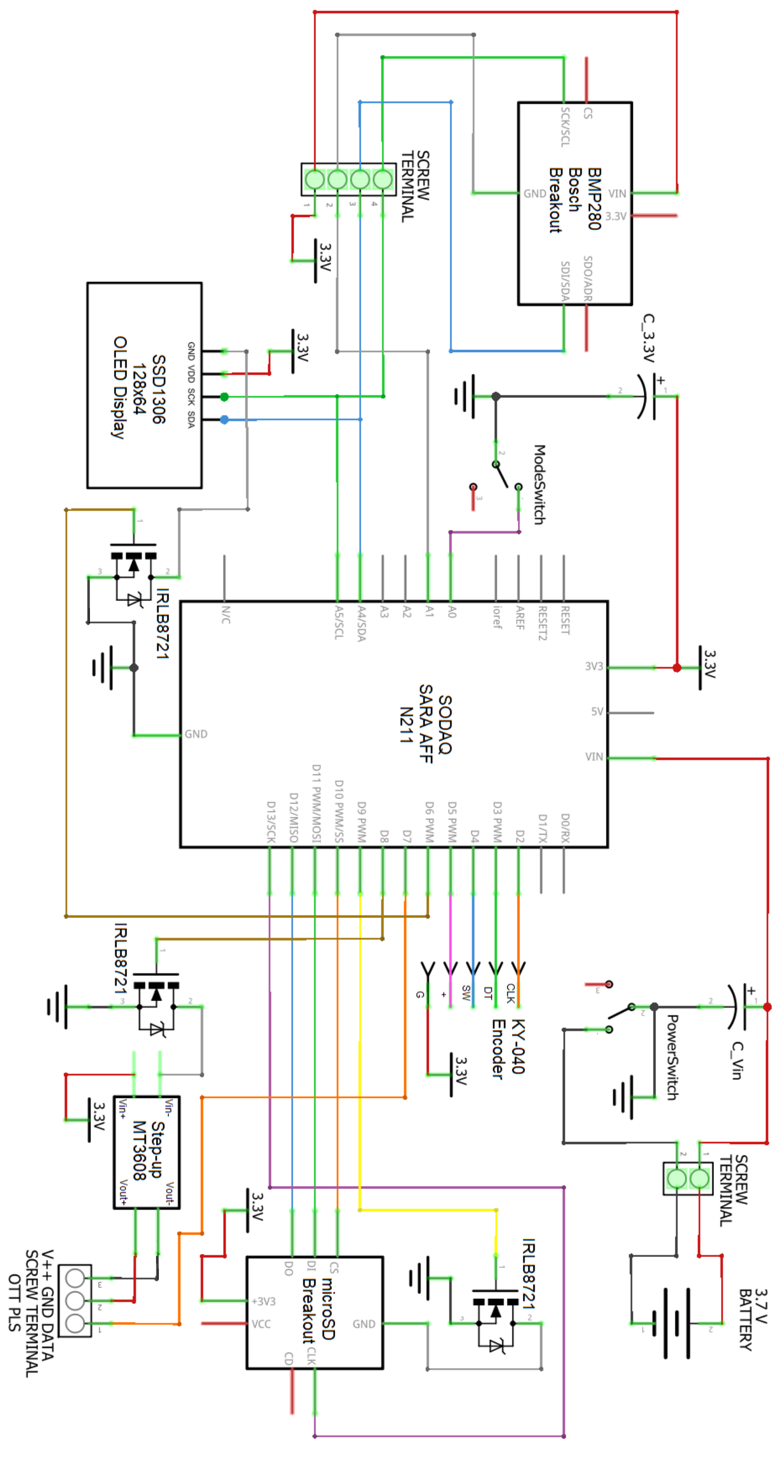
u-blox (2018). "SARA-N2 Modules - Power-optimized NB-IoT modules - AT Commands Manual". In: URL: https://www.u-blox.com/sites/default/files/SARA-N2_ATCommands_%5C%28UBX-16014887%5C%29.pdf.

University, Battery (2017). *BU-106: Advantages of Primary Batteries*. URL: batteryuniversity.com/learn/article/primary_batteries.

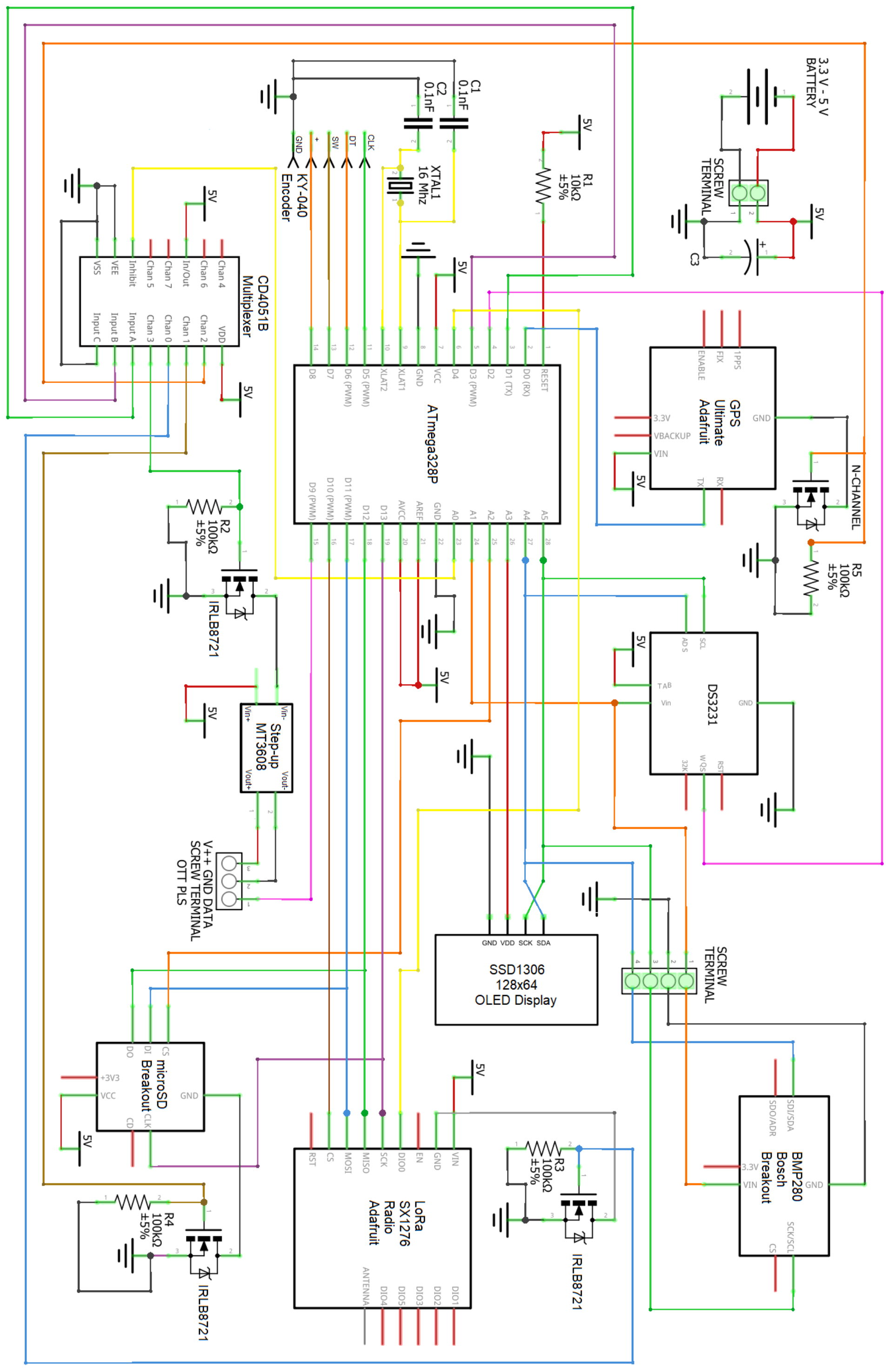
Williams, Elliot (2015). *Make: AVR Programming*. Maker Media, Inc.

Xi' an Aerosemi Technology Co. (2018). "High Efficiency 1.2MHz 2A Step Up Converter". In: URL: https://www.olimex.com/Products/Breadboarding/BB-PWR-3608/resources/MT3608.pdf.

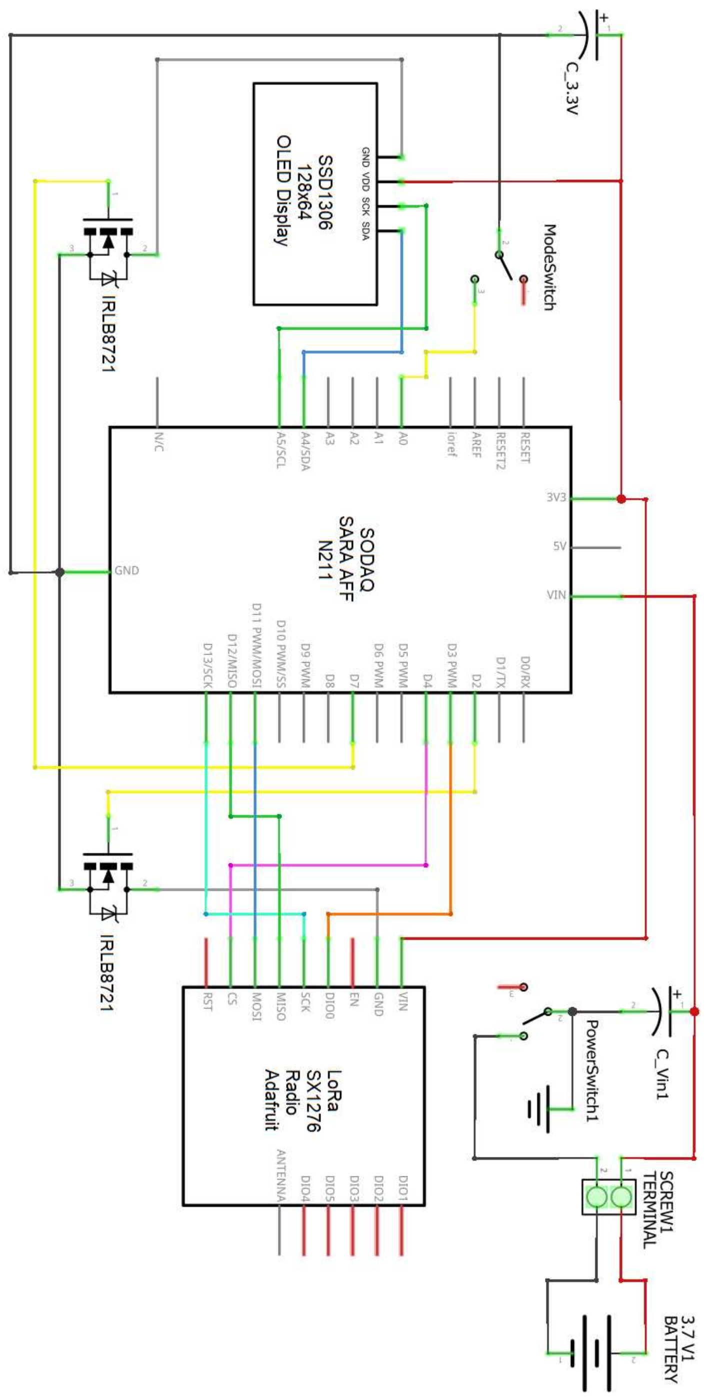
Appendix A: Schematic - Single-node system



Appendix B: Schematic - Sensor node of multi-node system



Appendix C: Schematic - Gatewat of multi-node system



Appendix D: Components list and prices

Table 1: All the components required for one node of the single-node system.

Component	**Price (NOK)**	**Example site**
SODAQ SARA AFF N211 (including PCB Antenna)	928	https://shop.sodaq.com/sodaq-sara-aff-n211.html
Lithium Ion Polymer Battery 6000 mAh	234	https://shop.sodaq.com/battery-lithium-polymer-6000-mah.html
Casing IP67 CHDX8-323	189	https://www.elfadistrelec.no/no/robust-boks-med-hengsler-185x135x85mm-gra-abs-ip66-ip67-camdenboss-chdx8-323/p/30113299
Antenna IP67 including U.FL connector cable	92	https://no.mouser.com/ProductDetail/Pycom/LoRa-Antenna?qs=sGAEpiMZZMve4%2FbfQkoj%252BCdToa3Jb5bySpbgGrc4L1I%3D
MicroSD Card Breakout Board	64	https://no.mouser.com/ProductDetail/Adafruit/254?qs= sGAEpiMZZMsMyYRRhGMFNu9gxkKdu10TYKPWLeYCwDQ%3D
1W Solar Panel	46	https://shop.sodaq.com/1w-solar-panel.html
Vent IP67	22	https://www.data-alliance.net/vent-plug-for-network-enclosure-waterproof-ip67/
I2C Display SSD1306 0.96" 128X64 OLED	21	https://www.ebay.com/itm/0-96-Yellow-Blue-White-128X64-OLED-I2C-IIC-Serial-LCD-LED-SSD-Display-SSD1306/282658584700?hash=item41cfc3b87c:m:motyoLeb9AuoziMk8Mb2VBw
3 x MOSFET N-chanel IRLB8721	18	https://omegav.ed.ntnu.no/komp?search=irlb8721
Cable Grommet IP68 Polyamide	13	https://uk.rs-online.com/web/p/cable-glands/8229684/
BMP280 Temperature and Pressure Sensor Module	9	https://www.ebay.com/itm/BMP280-3-3V-Pressure-Sensor-Module-High-Precision-Atmospheric-Arduino-BMP180/201511252133?epid=1655551950&hash=item2eeb01b0a5:g:EA0AAOSwBSxbIIlB
MT3608 Step-up converter	5	https://www.ebay.com/itm/2pcs-MT3608-Step-Up-Power-Apply-Booster-Module-DC-DC-2V-24V-2A-NEW/171907688472?hash=item28067f2818:g:8G8AAOSwxYxUsl2k
Rotary Encoder KY-040	8	https://www.ebay.com/itm/10PCS-KY-040-Rotary-Encoder-Module-Brick-Sensor-Axle-Plastic-Button-Cap/172226545771?epid=506379107&hash=item281980886b:g:bVQAAOSwzJ5XTu-R
Custom PCB shield	4	https://jlcpcb.com/quote
2 x SPDT Slide Switch	2	https://www.ebay.com/itm/20PCS-3-Pin-SS12D00G3-2-Position-SPDT-1P2T-PCB-Panel-Mini-Vertical-Slide-Switch/311654835782?hash=item4890136646:g:tVwAAOSw0kNXhD-R
2-pin screw terminal	2	https://www.kjell.com/se/produkter/el-verktyg/elektronik/montering/luxorparts-skruvplint-5-mm-2-pol-10-pack-p90760
3-pin screw terminal	3	https://www.kjell.com/se/produkter/el-verktyg/elektronik/montering/luxorparts-skruvplint-5-mm-3-pol-10-pack-p90761
4-pin screw terminal	4	https://www.kjell.com/se/produkter/el-verktyg/elektronik/montering/luxorparts-skruvplint-5-mm-4-pol-10-pack-p90762
2 x Capacitor ∼5 µF Ceramic	1	https://omegav.ed.ntnu.no/komp?search=kondensator
Single Row Straight Female Pin Header Strip 2.54mm	1	https://www.ebay.com/itm/10PCS-Male-Female-40pin-2-54mm-Header-Socket-Row-Strip-PCB-Connector-Cool/382264400396?hash=item5900bbda0c:g:8B4AAOSwzXxZ80Hu
Single Row Straight Male Pin Header Strip 2.54mm	1	https://www.ebay.com/itm/10PCS-Male-Female-40pin-2-54mm-Header-Socket-Row-Strip-PCB-Connector-Cool/382264400396?hash=item5900bbda0c:g:8B4AAOSwzXxZ80Hu
SUM	**1667**	