Master's thesis

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

Morten Austnes

Sensor system for measuring water level in nature

Master's thesis in Cybernetics and robotics
Supervisor: Geir Mathisen
June 2022

NTNU
Norwegian University of
Science and Technology

Morten Austnes

Sensor system for measuring water level in nature

Master's thesis in Cybernetics and robotics
Supervisor: Geir Mathisen
June 2022

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

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of Science and Technology

Faculty of Information Technology and Electrical engineering

Department of Engineering Cybernetics

NTNU

Master's Thesis Description

Candidate:	Morten Austnes
Course	TTK4900 - Engineering Cybernetics, Master's Thesis
Thesis title (Norwegian)	Sensorsystem for måling av vannstand i naturen
Thesis title (English)	Sensor system for measuring water level in nature

Thesis description:

Water level is measured in many cases; water level in rivers, lakes, groundwater, sea level, open and closed tubs / pools etc. We want to investigate instrumentation for measuring water levels in rivers, streams and in groundwater. The goal is to produce an instrumentation that can be deployed for a long time without maintenance and that measures the water level and sends the measurements wirelessly to a cloud service. Important keywords here will be robustness and low energy consumption.

This thesis shall build on the experience from the specialization project "Sensors for measuring water level in nature" and extend the scope to investigate a complete sensor system including a wireless front end sensor, sky service connection and information presentation on the operator's video screen. The focus should be on the wireless part of the system and important keywords here will be robustness and low energy consumption.

The tasks will be:

1. Conduct a literary study of systems used for monitoring of environmental parameters.
2. Propose a complete sensor system for water level measurement with a wireless front end sensor, sky service connection and information presentation on the operator's video screen.
3. As far as time permits, implement the suggested system.

Start date:	17. January, 2022
Due date:	13. June, 2022

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

Abstract

IoT technology has created new opportunities for measuring important aspects of nature, and is especially important as the focus on climate increases. Water has previously proven to be challenging to monitor, but with new low-power devices one can measure groundwater, rivers, lakes, and seawater more accurately, and at a much larger scale than before.

In this thesis a system for collecting water level data in nature has been designed, implemented, and tested. A PCB was designed and ordered, two sensors have been ordered to be used alongside the designed electronics, and a cloud service has been used to log data to some extent. The goal for the thesis is to produce a system which can be deployed to collect water level data for long periods of time without maintenance. The thesis investigates the collected data, wireless technology for use in remote IoT devices, and energy consumption.

The system succeeded to log data in nature in two separate occasions, the longest of them lasting for seven days. Energy consumption of the system was measured, and it was found that the system can be deployed for months. Some improvements are left to be done, as the database and GUI for storing and plotting the data was not fully implemented.

Sammendrag

IoT-teknologi har skapt nye muligheter for å måle viktige aspekt i naturen, og blir stadig viktigere ettersom fokuset på klima øker. Vann har tidligere vist seg vanskelig å overvåke, men med ny lav-energi elektronikk kan en måle grunnvann, elver, innsjøer, og havvann mer nøyaktig, og på en mye større skala enn før.

Et system for innsamling av vanndata ble designet, implementert, og testet. Et kretskort ble designet og bestilt, to sensorer ble bestilt og brukt med den designede elektronikken, og en skytjeneste ble brukt til å loggføre dataene til en viss grad. Målet er å produsere et system som kan bli utplassert for å samle inn data om vannivå i lange tidsrom uten vedlikehold. Oppgaven ser på de innsamlede dataene, trådløs kommunikasjon i IoT-enheter, og energiforbruk.

Systemet lyktes med å logge vanndata i naturen ved to separate anledninger, hvor den lengste av dem varte i sju dager. Energiforbruket til system ble målt, og det ble funnet at systemet kan være utplassert i måneder av gangen. Noe arbeid gjenstår, ettersom databasen og brukergrensesnittet for lagring og plotting av data ikke ble fullstendig implementert.

Preface

This thesis continues the work done in the specialization project TTK4450. Much of the hardware from the specialization project was reworked and improved upon. Some important literature was also fetched from the specialization project to be presented here.

The making of this thesis has proved to be much more challenging that first expected. I hope this paper has provided enough information for others to be able to re-create and improve on it.

I would like to thank my supervisor Geir Mathisen for his help and guidance throughout this thesis. I also want to thank Oskar Østby and Ingebrigt Reinsborg for being great friends during my time at NTNU.

Table of Contents

List of Figures

List of Tables

Abbreviations

Expression	Meaning
ADC	Analog-to-Digital Converter
API	Application programming interface
BLE	Bluetooth Low Energy
DAQ	Data acquisition (system)
GPIO	General-purpose input/output
I2C	Inter-Integrated Circuit
IC	Integrated circuit
IoT	Internet of Things
ISR	Interrupt service routine
LED	Light emitting diode
LoRa	Long Range
MCU	Microcontroller unit
RTC	Real-time clock
SMD	Surface-mount device
SMT	Surface-mount technology
SPI	Serial Peripheral Interface
TTN	The Things Network
UART	Universal Asynchronous Receiver Transmitter
UMS	Ultrasonic measurement system
USART	Universal Synchronous Asynchronous Receiver Transmitter

Chapter 1

Introduction

1.1 Background

The interest for the Internet of Things (IoT) has increased over the past years. New technology has enabled the industry to combine embedded systems and wireless communication to create wholly new applications. These systems are often called IoT devices. Advances in embedded engineering has created better low-power microcontrollers, wireless communication, and sensor technology. The ability to measure and process data, and then send it wirelessly with very low energy consumption gives us the ability to create small, cheap, and long-lived devices for new applications. This progress has made IoT a viable solution to many of the problems facing today's society. IoT describes systems utilizing sensors and communication technology which work either by themselves or in groups to perform tasks. This technology has a wide range of uses, and are often employed for large scale data collection, and when small and scalable solutions are needed. They see use everywhere, from consumer products to industry applications; Smart homes, health monitoring in medicine, in the transportation industry, manufacturing, agriculture, martime systems, infrastructure, environmental monitoring, and military operations are all fields in which IoT has seen use.

As mentioned, there have been created numerous IoT applications for environmental monitoring and infrastructure, differing wildly in use-cases. These include measuring the air and water quality, measuring atmospheric and soil conditions, and early detection of earthquakes and tsunamis. These devices can provide early warnings in the case of natural disasters or other events which could lead to the damage of important infrastructure. By doing so, they provide great societal benefits.

Systems for measuring water can be useful in a wide variety of cases. Groundwater, rivers, lakes, and seawater are all important parts of nature which we still could benefit from measuring more closely. IoT devices for measuring groundwater can prove useful for the residents of the area. Rivers can be

measured to protect local infrastructure, or possibly provide warnings for anomalies and to prevent natural disasters.

Can a system for measuring water levels be made such that it can be placed outdoors for prolonged periods of time to collect useful data? If so, how is such a system designed, and what sensors should be used to make the application both reliable and cheap? It would need to utilize wireless communication to transmit its data to an operator's desk. Designing such a system comes with challenges. It needs to be created with limited resources, especially with cost in mind to make such a product scalable. To ensure the longevity of the system one needs to consider its power consumption and its accuracy over time. Simultaneously it needs to be precise enough to return any meaningful data, all while being waterproof. The system developed in this thesis will be used to measure both water level in controlled environments, as well as in nature. It should be able to detect water levels in both stationary waters, e.g. lakes and ground water, as well as in more rapidly moving waters such as rivers.

1.2 Motivation

I chose this thesis due to my interest in embedded systems, as well as wanting to expand my knowledge on wireless sensor networks. Since I started as a student at the university I knew I wanted to work on embedded systems. I have previously worked with sensor networks and measurements related to water during my summer internships, and found it enjoyable.

I also wanted a practical assignment, to work with something hands-on in the field. Collecting data in nature is vital, and especially water. Flooding, for example, is a prevalent natural disaster and to work on systems which can provide an early warning for such events serves as a strong motivational factor.

1.3 Limitations

The system was only tested for a week at most. Ideally the system should have been tested for at least a month to truly test the battery time and real-time clock drift. The test in Estenstadmarka only lasted for a few hours, due to fear of potential theft or interference by strangers. It should also have been tested during different weather conditions, to see how robust it truly is. Other sensors could also have been tested, to see if the energy consumption could be reduced, and if cheaper and less accurate sensors could be used. There was little preexisting knowledge about LoRa, so the cloud service implementation was somewhat done in a rush.

1.4 Disposition

Chapter 2 Literature Review is divided in three parts. First it contains studies done on water level measurement, then a section on different long-range wireless technologies, and finally theory and studies on sensors for measuring water level, as well as a brief look at sensor systems already on the market.

Chapter 3 Design presents design specifications for a system that can measure water level in nature. It proposes a design for the complete system, from embedded hardware and software, to the server-side software.

Chapter 4 Implementation contains the details for the implementation of the hardware and software for the proposed system from Chapter 3.

Chapter 5 Testing and results details how the system was tested, as well as presenting the results of said tests.

Chapter 6 Discussion discusses the findings from Chapter 5 in detail.

Chapter 7 Conclusion concludes the thesis and its findings.

Chapter 8 Future work suggests improvements that can be done to the system that was proposed, implemented, and tested.

Appendix A contains the part list and the complete schematic for the PCB.

Appendix B contains screenshots of the TTN GUI and control panel.

Chapter 2

Literature review

The purpose of this chapter is to investigate different measurement systems for use in logging water levels. The literature review will discuss different aspects of measuring water with embedded hardware.

Section 2.1 covers surveys done to compare different IoT water measurements systems. It discusses how the systems have been implemented, based on what hardware is used, which sensors have been chosen, and whether or not energy consumption was considered.

Section 2.2 discusses different wireless technologies to find a good candidate for the wireless communication module used in this thesis.

Section 2.3 investigates research done on the topic of water level measurement, with a focus on sensors. 2.3.3 compares different already existing products on the market to get a benchmark for what to expect of the specifications for such a system.

2.1 Systems

2.1.1 Survey of the Systems for Water Level Detection

The following is fetched from K. S. Mehta, P. T. Maru, and N. P. Shaha, 2020 [1]

In *Survey of the Systems for Water Level Detection* [1], Mehta, Maru, and Shah conducted a survey IoT device for measuring of water levels. As described in the paper, they focused on devices which could calculate water rates. Collecting quantitative data measured from different consumption units could then be used to predict water demand. This data could then prove useful by load sharing, making the process of supply and demand more efficient.

The survey is relatively new, conducted in December of 2020. It compares several different papers discussing IoT-based water level measurement. Mehta, Maru, and Shah compare many different aspects of water level measurement, including which microcontroller was used, which sensor the paper used, cloud solutions, cost of production, and more. The IoT solutions compared are listed in Table 2.1.1. As mentioned, IoT devices can support a wide array of different types of sensors. The comparison shows that many of the papers settled on basic ultrasonic measurement systems (UMS). Some mentioned the use of pH sensors and water (conductivity) sensors. Some of the papers also employed machine learning algorithms to improve the accuracy of the water level measurement.

It was found that there are few existing systems, all with their respective drawbacks. The survey mentions some of the challenges still left to solve for measuring water levels using IoT-based technology. Providing viable data is one such challenge. The survey mentioned machine learning algorithms to improve accuracy of the measured water level. The survey also mentioned improving or reconstructing a better IoT packet, providing offline support, and creating a better method for prediction of water levels.

Many of the papers discussed in the survey seem to prefer Arduino or similar boards for their solutions. While being fast to implement and easy to use, they have a few drawbacks. The most important drawbacks for this thesis being power consumption and real-time capabilities. The findings in this survey seem to support the work done in the specialization project [2] concerning sensors. As stated in the survey: "Upon review of literature it is being noticed that ultrasonic sensor and transmission of data into cloud storage is a common practice for these systems." Manufactured ultrasonic sensors are often easy to use both in software and in setup, as they provide data without being in contact with the medium. They provide accurate and repeatable results at the expense of requiring somewhat more current draw than other types of sensors. The use of machine learning algorithms presented in some of these papers are also out of scope for this thesis.

The different papers discussed in Mehta, Maru, and Shah managed to measure water as planned. However, little thought seems to have been put into the energy consumption and longevity of their devices. The Arduinos as an example have many features such as LEDs that draw unecessary current, as well as no built-in deep-sleep functionality. Some devices seem to only have been tested in a lab, and only for relatively short periods of time, or with a power supply and not a battery.

Table 2.1.1: Comparative Study of system for Water Level Measurement. From [1], shortened.

[Citation] Title	Approach	Boards	Sensors	Others	Cost
[3] IOT based water level meter	To detect the water level of the dams	Node MCU	-	IoT packet transmit data to an online cloud platform used to gather and display data	Low
[4] Smart water quality monitoring and metering using LoRa for smart villages	To detect the quality of water	M2M LoRa, KT-LoRa mote	PH sensor water quality sensor	-	High
[5] IOT based water level monitoring and implementation on both agriculture and domestic areas	To estimate water dimension and to improve the moisture level of the soil	Arduino	Ultrasonic sensor, moisture sensor, GSM module	-	Medium
[6] Non-Contact Water Level Monitoring System Implemented using LabVIEW and Arduino	Non-contact water level monitoring system	Arduino UNO	Ultrasonic sensor	-	Low
[7] Water Demand Prediction Using Support Vector Machine Regression	Water demand prediction using machine learning algorithm	-	-	Machine learning algorithm along with support Vector Regression	-
[8] A Water Level Detection: IoT Platform Based on Wireless Sensor Network	Water detection and transmit data to the cloud storage for further analysis	Node MCU	HC-SR04 (ultrasonic), ESP8266 WiFi module	Thingers an online cloud platform used for plotting the gathered data	High
[9] A Novel ANN Based Adaptive Ultrasonic Measurement System for Accurate Water Level Monitoring	To use ultrasonic sensor for accurate water measurement and monitoring system	Arduino UNO	HC-SR04 (ultrasonic), Bluetooth module, temperature and humidity sensor	Machine learning algorithm ANN (artificial neural network).	Medium
[10] Flash Flood Detection in Urban Cities Using Ultrasonic and Infrared Sensors	In this paper the flash floods are detected by the author with the help of ultrasonic rangefinder sensor and some infrared temperature sensors.	32-bit micro-controller	Pir motion sensor, zigbee module, ultra-sonic sensor, infrared temperature sensor	-	High
[11] Assessment of Surface Water Quality by Using Satellite Images Fusion Based on PCA Method in the Lake Gala, Turkey	In this proposed system satellite images are used to detect water quality of the lake	-	-	Satellite images from various satellites are used. PCA, MLR, ANN, SVM machine learning algorithm used.	-
[12] Water Level Measurement Device And Shoreline Extraction Method	Water level measurement device and shoreline extraction method.	-	-	Machine learning for identifying water areas and non-water.	-
[13] Smart Water Quality Monitoring System	A smart water quality monitoring system used to check the quality of the water	32-bit micro-controller	ESP8266 WiFi module, GSP module, pH sensor, humidity sensor	-	Low

2.1.2 IoT and ICT based Smart Water Management, Monitoring and Controlling System: A Review

The following is fetched from H. Yasin, S. Zeebaree, M. M. Sadeeq et al., 2021 [14]

Similarly to [1], H. Yasin, S. Zeebaree, M. M. Sadeeq et al. compare different papers on IoT-based systems which monitor and control water supplies. As stated in their article, "intelligent monitoring is defined using different computational methods that provide the customers with relevant tools and information in monitoring, control, manage, and optimize the network." While the applications for water supply monitoring and control are somewhat different to those that only monitor water level, many of the same solutions and technologies are involved. H. Yasin, S. Zeebaree, M. M. Sadeeq et al. compare and discuss many different aspects of smart water monitoring. The articles discussed in the paper are compared based on their chosen microcontroller, programming language, sensors, and commuication module and protocol. Table 2.1.2 shows a summarized comparison of various smart water applications based on IoT technology, as done in [1].

Although many of the systems discussed in the paper were successful in their task, few actually used a custom design. Many of the implementations relied on popular pre-made microcontroller boards such as the Arduino, Raspberry Pi, and NodeMCU. These are often seen as hobbyist tools due to being easy to program and use for a specialized task, but they would not fit tasks which require ultra low-power. These systems consume too much energy to be used in remote locations, which would require the system to be solely powered by battery.

Similar to the survey done by Mehta, Maru, and Shah, the systems discussed in Yasin et al. show promising results in their collected data, but not all take power consumption into consideration. Many were only tested in a lab, and for short periods of time. Some of the devices discussed are expected to have a constant voltage supply available. Some do have energy consumption in mind however, such as [15], and state that "power consumption is a major constraint for IoT applications, because the applications are most likely to operate on batteries. Communication of data is a major source of power consumption."

Table 2.1.2: Comparison of various Smart Water Applications features based on IoT Technologies. From [14], shortened.

[Citation] Title	Microcontroller	Sensors	Comm. Module	Protocol	Result
[15] Internet of things enabled real time water quality monitoring system	TI CC3200	Water level, pH, YL69, Conductivity, turbity	ZigBee	HTTP	The low-cost water quality control system is less complex
[16] IoT based smart irrigation monitoring and controlling system	ATmega328	LM393, soil moisture, M116 water level	ZigBee	HTTP	The system optimizes the use of water for irrigation purposes. Furthermore, water consumption has decreased.
[17] A smart irrigation system using IoT and fuzzy logic controller	Mamdani Fuzzy	Soil moisture, DHT11	ZigBee	HTTP	Reducing water consumption during irrigation.
[18] IoT based smart water system	PIC16F877	Ultrasonic, flow, pH	GSM	WAP	Control the water impurity, water wasted, and low water flow.
[19] Smart water management in housing societies using IoT	Raspberry Pi	Ultrasonic, turbity	WiFi build-in	MQTT	Enable the users to monitor and manage the water management systems remotely from their smartphone.
[20] An IoT Based water monitoring system for smart buildings	MSP430	Flow, pressure	CC2650	MQTT	Detect water leakages, control water wastage, and avoid natural water waste.
[21] Consumer's activity prediction in household water consumption based-IoT	NodeMCU	Flow rate	ESP8266	HTTP	Increase people's awareness about saving water for sustainable water resources.
[22] Design and development of IOT based SMART water distribution network for Residential areas	Raspberry Pi/Arduino	Ultrasonic, turbity, flow	WiFi build-in	HTTP	Distributed the same amount of water to all customers, maintain water quality, and maintain a water level in the main water tank.
[23] IOT based smart water management, monitoring and distribution system for an apartment	Arduino	Ultrasonic, flow meter	ESP8266	MQTT	Water wastage is fully controlled, a cost-effective system to save water and money.
[24] IOT based water level monitoring and implementation on both agriculture and domestic areas	Arduino	Ultrasonic, soil moisture	Ethernet shield	HTTP	Reduce the burden of the user in monitoring the water level and make it a user-friendly system.
[25] IoT based intelligent domestic water management system	NodeMCU	YF-S201	ESP8266	MQTT	Design detection model of water usage anomaly in households.
[26] Smart water management in agricultural land using IoT	Arduino	LM35, DHT11, pH, moisture	GSM/GPRS	WAP	The model is to come up with a solution for conserving water.
[27] IoT-based smart platform to manage personal water usage	NodeMCU	pH, turbity	ESP8266	MQTT	Refining the water bottle with fuzzy theory to fine-tune the calculation goal of water and smartly suggestions technique.
[28] Cloud-based internet of things for smart water consumption monitoring system	Raspberry Pi	YF-S201	WiFi build-in	HTTP	The system utilizes live water usage data from water flow meters at household and draws proper inferences from it.
[29] Exploiting constrained IoT devices in a trustless blockchain-based water management system	Raspberry Pi	YF-201, hall-effect	RFM95W	HTTP	Incur an additional 6% of the energy consumed for their typical interactions with a gateway.
[30] IoT and cloud computing based smart water metering system	NodeMCU	YF-S201	WiFi build-in	MQTT	Detect excess water consumption by using machine learning.
[31] IoT based water quality monitoring system and evaluation	NodeMCU	pH	ESP8266	MQTT	Automatically updates the status of water quality, real-time monitoring, less operational maintenance.
[32] Smart water leakage and theft detection using IoT	Arduino	YF-S201	ZigBee, Rola	HTTP	Save water resources in areas where pipeline connection is in use.
[33] The Internet of Things (IOT) based smart rain water harvesting system	NodeMCU	pH, rainfall, ultrasonic	ESP8266	MQTP	Retain the quality of precious rainwater, collect rainwater in areas of tiny size houses.
[34] IOT based water quality monitoring system	LPC2148	pH, EC, turbity, LM35	ESP8266	MQTP	Low cost, efficient, real-time water quality monitoring.

2.2 Wireless communication

Most low-power devices consume the most energy when transferring data. With recent advances in IoT, new low-power communication protocols have been developed to increase the lifetime of battery powered systems. This section will focus on these low-power communication methods. Important aspects to consider when choosing the communication technology for an IoT-device is quality of service (QoS), energy consumption, scalability, range, price, and deployment model. The literature review will not cover short-range technologies, as these are deemed unsuitable for systems measuring in rural outdoors locations. While it is possible to connect devices in a mesh topology to increase range [35], only a star topology will be considered as Low Power Wide Area Network (LPWAN) technologies were considered to be more suitable for the system which will be designed and implemented for this thesis.

Two different LPWAN technologies are primarily used in IoT-devices to communicate over long distances, unlicensed LPWA and cellular technology. Sigfox and LoRaWAN are the two main technologies used in the unlicensed LPWA category [36]. Unlicensed LPWA has been developed for IoT-devices which require low power usage and long range. NB-IoT is cellular technology designed for use in IoT, based on a liscensed spectrum primarily used for mobile communication [37].

The following is fetched from K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, 2019 [35]

K. Mekki, E. Bajic, F. Chaxel, and F. Meyer compare different LPWAN technologies in their paper *Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT* [35]. The paper focuses on three leading LPWAN technologies, Sigfox, LoRaWAN, and NB-IoT. The paper compares quality of service, coverage, range, latency, battery life, scalability, payload length, deployment, and cost of each communication technology. It also considers different application scenarios and explain which of these technolgies best fit for each application.

As described in the paper, BLE and ZigBee are not adapted for long range transmission, and most cellular technolgies, namely 2G, 3G, and 4G consume too much energy to be suitable for small low-power IoT-devices. Therefore they are not included in the comparison. ZigBee and BLE can utilize mesh networks to circumvent their lack of range. This however, comes at the cost of requiring deployment of a large number of devices. Also, as some devices become more congested than others, the battery lifetime is also significantly reduced in the

most active devices. Sigfox, LoRaWAN, and NB-IoT can connect in a star topology, to let them send data directly to a back end. This saves on energy consumption and gives immediate access to data to an operator. Transmitting directly to a back end that is always-on also removes the need to listen before transmitting each message.

Comparison:

Quality of service — Quality of service (QoS) can guarantee some level of performance to wireless communication. Sigfox and LoRaWAN both employ an asynchronous protocol, while NB-IoT employs a synchronous protocol which provides quality of service. Quality of services comes at the detriment of cost however. K. Mekki et al. recommend using NB-IoT for applications which need guaranteed QoS.

Battery life — All three technologies utilize sleep modes as much as possible, but NB-IoT consumes additional energy due to its synchronous communication and QoS handling. NB-IoT also has a higher peak current while sending data. While Sigfox and LoRaWAN are useful for less power consumption, they come at the cost of latency and connectivity.

Scalability and payload length — NB-IoT offers more connected devices per base station and longer payload length. NB-IoT allows for 100K devices per station, while Sigfox and LoRaWAN offer 50K per cell. NB-IoT allows for transmission of data up to 1600 bytes, while LoraWAN allows a maximum of 243 bytes. Sigfox has the lowest payload length of 12 bytes, which could limit its utilization on various IoT-devices which need to send large amounts of data.

Coverage and range — Sigfox boasts a range of >40km from a single base station, and LoRaWAN provides a range of <20km. NB-IoT has the lowest range of <10km. NB-IoT also does not cover regions without LTE coverage.

Deployment model — Unlike Sigfox and NB-IoT, LoRaWAN offers a local network deployment, LAN using a LoRa gateway, as well as public network operation via base stations.

Cost — Sigfox and LoRaWAN offer a much cheaper price, at 2 Euro per device for Sigfox, and 3-5 Euro per device for LoRaWAN. With over 20 Euro per deivce for NB-IoT. NB-IoT base stations cost much more, while also covering less area. Table 2.2.1 shows a comparison of different prices for each of the three different communication technologies.

Table 2.2.1: Cost of Sigfox, LoRaWAN, and NB-IoT, from [35].

	Spectrum cost	**Deployment cost**	**End-device cost**
Sigfox	Free	>4000 €/base station	<2€
LoRaWAN	Free	>100 €/gateway >1000 €/base station	3-5€
NB-IoT	>500M€/MHz	>15000 €/base station	>20€

The work of K. Mekki, E. Bajic, F. Chaxel, and F. Meyer is summarized in Figure 2.2.1, which visualizes the different aspects of Sigfox, LoRaWAN, and NB-IoT. The Figure shows how NB-IoT provides better quality of communication, at the cost of energy consumption, price, and range. Table 2.2.2 shows a more detailed comparison of the three different communication technologies.

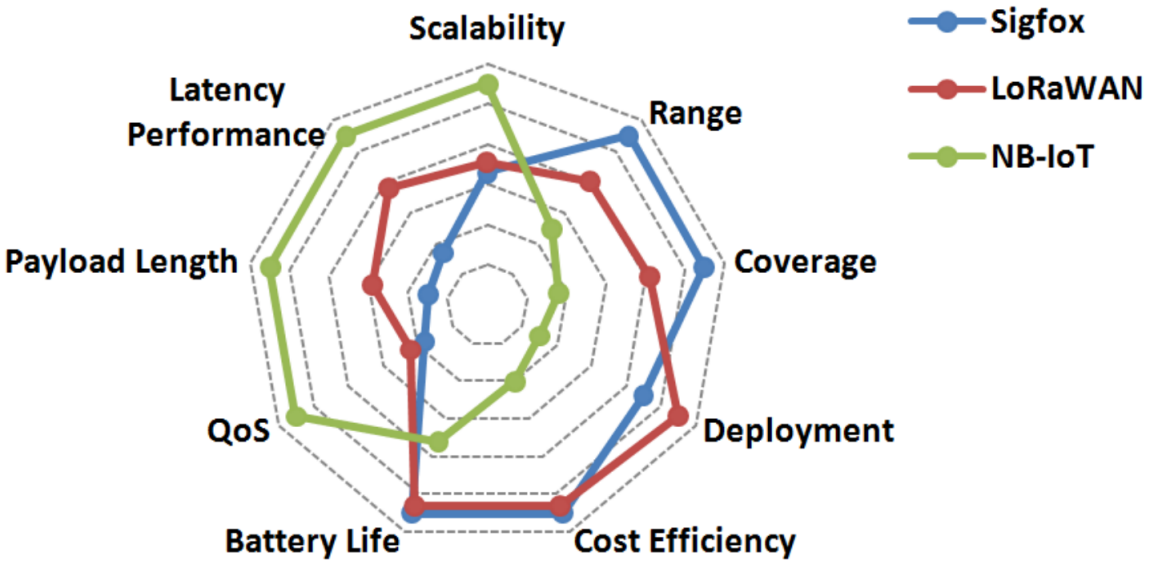


Figure 2.2.1: Respective advantages of Sigfox, NB-IoT, and LoRaWAN, from [35].

Table 2.2.2: Overview of LPWAN technologies: Sigfox, LoRaWAN, and NB-IoT, from [35].

	Sigfox	**LoRaWAN**	**NB-IoT**
Modulation	BPSK	CSS	QPSK
Frequency	Unlicensed ISM bands (868MHz in Europe)	Unlicensed ISM bands (868MHz in Europe)	Licensed LTE frequency bands
Bandwith	100Hz	250kHz and 125kHz	200kHz
Maximum data rate	100bps	50kbps	200kbps
Bidirectional	Limited / Half-duplex	Yes / Half-duplex	Yes / Half-duplex
Maximum messages/day	140 (UL), 4 (DL)	Unlimited	Unlimited
Maximum payload length	12 bytes (UL), 8 bytes (DL)	243 bytes	1600 bytes
Range	10km (urban), 40km (rural)	5km (urban), 20km (rural)	1km (urban), 10km (rural)
Interference immunity	Very high	Very high	Low
Authentication & encryption	Not supported	Yes (AES 128b)	Yes (LTE encryption)
Adaptive data rate	No	Yes	No
Handover	End-devices do not join a single base station	End-devices do not join a single base station	End-devices join a single base station
Localization	Yes (RSSI)	Yes (TDOA)	No (under specification)
Allow private network	No	Yes	No
Standarization	Sigfox company is collaborating with ETSI on the standardization of Sigfox-based network	LoRa-Alliance	3GPP

2.3 Sensors

The specialization project *Sensors for measuring water level in nature* [2] by Austnes investigates instrumentation for measuring water levels in rivers, streams, and in groundwater. The goal is to investigate sensors which can provide measurements of water level wirelessly for extended periods of time. It examines different measurement principles, looking at the availability and suitability of equipment, assessing robustness and energy needs.

The specialization project assignment is comprised of three parts. Firstly, a literary study is conducted on systems used for water level measurements, with a special focus on the sensors used for the measurement. Secondly, sensors for a battery powered water level measurement system are proposed. Thirdly, experiments are conducted to verify the viability of the proposed sensors.

This thesis is a natural progression of the work done in the specialization project [2]. As the assignment consisted of performing a literature review on systems used for water level measurement, with a special focus on the sensors used in such systems, the literature is highly relevant for this thesis. Further, the specialization project proposed sensors to be used in a battery powered water level measurement system, which also proves as useful literature. For the sake of completeness of this thesis, parts of the specialization project is summarized here with minor changes. Sections 2.3.1, 2.3.2, and 2.3.3 are from

the specialization project [2].

2.3.1 Virtual techniques for liquid level monitoring using differential pressure sensors

The following is fetched from G. Nikolov, B. Nikolova, 2008 [38]

G. Nikolov, B. Nikolova (2008) explore possibilities for a software centric system for continuous liquid level measurement and monitoring of various liquids. The reason for developing a software centric design was to simplify the equipment design. By using only a few sensors one can easily extract user-defined data from various physical phenomena using software solutions. As part of their research, they have compared different sensors as shown in table 2.3.1. The table compares sensors used commonly for measuring water level and water density.

The experiment in by G. Nikolov, B. Nikolova (2008) was based on detection by hydrostatic pressure. The experiment was conducted in a closed tank. Performing experiments in such a controlled environment also allowed for a controlled instrument setup. The technique was based on two differential pressure sensors, as well as a thermocouple for temperature measurements. These were all connected to a portable multifunction data acquisition board (DAQ). One differential pressure sensor was mounted so to measure the difference between the atmospheric pressure and the pressure at the bottom of the tank. The other was mounted such that it measured the pressure difference between two points within the liquid, with a known distance between each other. The mounting of the equipment is shown in figure 2.3.1.

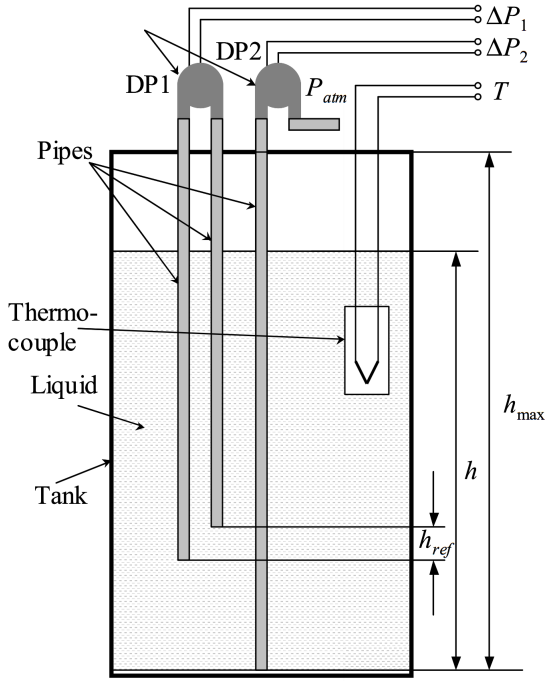


Figure 2.3.1: Mounting of the two differential pressure sensors and the thermocouple, from [38]

Table 2.3.1: Comparison between different methods for continuous liquid level measurement, from [38]

Method/ Sensor	**Advantages**	**Disadvantages**
Emerging Technologies (Time-of-Flight Measurements)		
Ultrasonic/Sonic	Easy to install; Non-contact measurement; No moving parts; Can measure corrosive and volatile liquids	Need high power; Low accuracy; Not operate on vacuum or high pressure applications; Expensive; Temperature correction is needed
Laser Level Transmitters	Use in vessels with numerous obstructions; High level of accuracy (better than 1 mm)	Expensive; Fails if dust, smoke, etc. are present in the vessel; Sensitive to dirt
Radar (Microwaves) Level Transmitters	Non-contact measurement; The transmission time is unaffected by ambient temperature and pressure fluctuations (can be used in closed tanks, where the liquid is turbulent and in the presence of obstructions and steam condensate)	Internal piping and multiple reflections can cause erroneous readings; Transmitter setup can be tedious and changes in the process environment can be problematic; The appropriate licences have to be obtained
Level measurement by hydrostatic pressure		
Bubbler-type sensor	Simplicity of design; Low initial purchase cost	Not suitable for use in non-vented vessels; Used gas may affect the contents of the tank
Differential Pressure Silicon Sensors	Inexpensive, Wide range measurements; Can be isolated safely from the process; Measurements can be digitally networked for remote computer access	Need power and active electronic; Require two vessel penetrations; Depends on the density and the temperature of the liquid
Level Measurements by Detecting Electrical Properties		
Capacitance level transmitters	Suitable for use in extreme conditions; Only a single tank penetration is required	Large errors caused by coatings; Normally limited to water-like media; Temperature correction is needed
RF Capacitance	Wide range of process conditions; No moving parts; Only a single tank penetration; Easy to use, Simple to clean	Special considerations is needed, to minimize errors caused by probe movement; Specific circuitry is needed
Up thrust buoyancy (Float systems)		
Displacers/-Floats	The only available technique for a cryogenic application; Adaptable to wide variations in fluid densities	Only for relatively clean fluids; High installation cost; Depends on the specific gravity of the liquid; Temperature correction is needed
Magnetostrictive Sensor	High level of accuracy; Reliable and repeatable; Non-contact; Low maintenance cost; Wide operating temperature range; Low and stable offset and low sensitivity to mechanical stress	Can work only if the auxiliary column and chamber walls constructed by nonmagnetic material; Magnets must not be operated beyond their Curie point

The article briefly mentions theory on hydrostatic pressure, and how to measure the liquid height based on measurements using two pressure sensors. Three

types of pressure measurement are listed in the paper.

- Absolute pressure, which is the pressure of a medium relative to vacuum.
- Differential pressure, which is the pressure difference between two mediums.
- Gage pressure, which is the differential pressure between a medium relative to the atmospheric pressure.

The paper also mentions useful parameters to compare pressure sensors by. Designing systems with pressure sensors for high accuracy requires considering linearity, ratiometricity, stability, repeatability, hysteresis, null set point, span set point, and null temperature shift. G. Nikolov, B. Nikolova (2008) use Freescale's MPX5100DP integrated pressure sensor, which was used for liquid level measurement. It has temperature compensation, signal conditioning, and calibration on chip (DP2 in figure 2.3.1). A similar sensor, Freescale's MPXV5004DP was used to measure the liquid density (DP1 in figure 2.3.1).

By measuring the liquid density through the differential pressure in the liquid to extract the density, as well as the pressure at the bottom of the medium, one can extract the level. The hydrostatic pressure can be found by using equation 2.1.

$$P = P_{atm} + \rho g h \tag{2.1}$$

P is the pressure of the medium measured in Pa, P_{atm} is the atmospheric pressure in Pa, ρ is the mass density of the liquid, g is the gravitational constant, and h is the vertical distance from the submerged sensor to the liquid surface.

In G. Nikolov, B. Nikolova (2008), the liquid density is found through the differential pressure between two points in a liquid with a known distance between them. Using 2.1 one finds that

$$\Delta P_1 = \rho g h_{ref} \tag{2.2}$$

where P_1 is the measured pressure by the sensor DP1 in figure 2.3.1. This gives the equation for the liquid density

$$\rho = \frac{\Delta P_1}{g h_{ref}} \tag{2.3}$$

Using this in 2.1, as well as the differential pressure measured by DP2 in figure 2.3.1 one can find the height by

$$h = \frac{\Delta P_2}{\rho g} = \frac{\Delta P_2}{\Delta P_1} h_{ref} \tag{2.4}$$

This method can be used to measure the liquid level even if the density of the medium is changing over time.

2.3.2 Water level and discharge measurements

The following is fetched from F. Larrarte, M. Lepot, F. Clemenset al., 2021 [39]

With urban drainage and stormwater management in focus, F. Larrarte et al. (2021) discusses important aspects of collecting data from urban sewage pipe systems. Various concepts for both water level and flow velocity are described and discussed. This thesis is mostly concerned with measuring water level, but the flow could also prove useful when extending the functionality of the measurement system being proposed for this project. F. Larrarte et al. (2021) conciders the velocity as a function of the water level. It would however likely have hysteresis, as shown in figure 2.3.2. In conclusion it would likely only occur in uniform steady flows, or controlled environments (in sewer systems). They recommend to carefully select the measurement site and using two separate sensors to measure the water level and the velocity.

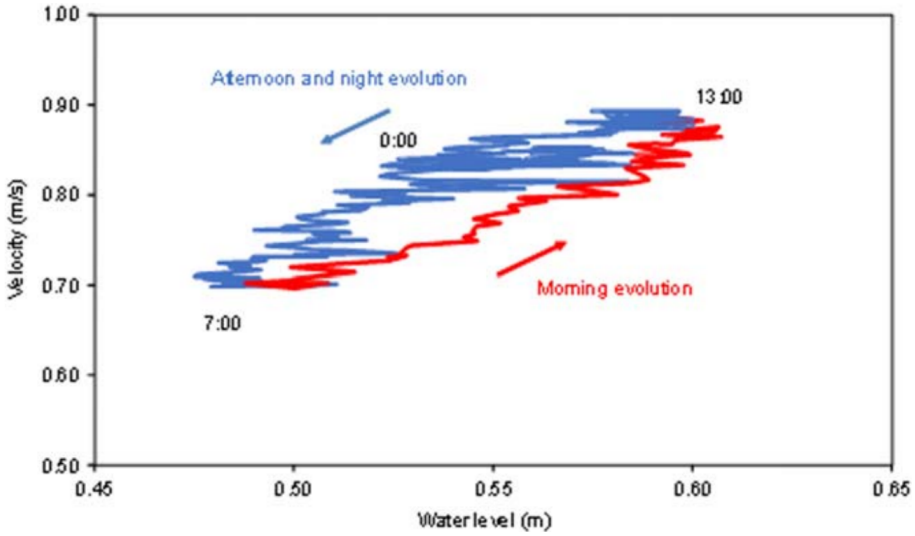


Figure 2.3.2: Water level/velocity hysteresis in sewer systems, from [39].

Pressure sensors are mentioned as a method for measuring water level. For moving liquids, the method is based on the Bernoulli relation [39]. With conservation of the sum, the equation is given by:

$$\frac{P}{\rho g} + z(x,y) + \frac{V(x,y)^2}{2g} \tag{2.5}$$

where P is the pressure, ρ is the density, z is the water height at a point (x, y), and V is the velocity at a point (x, y) in a cross section of the stream. One could calculate the water level by using this relation, but this phenomenon poses a problem for extracting water level through pressure. As the water velocity is inversely related to the pressure, the measurement will be affected by changing water velocity. Therefore if one wishes to measure the water level using pressure, it needs to be placed such that the velocity of the water is low around the sensor. As mentioned in F. Larrarte et al. (2021), a velocity of 1m/s leads to an overestimation of the water level of 5cm, and for 2m/s the overestimation will be 20cm [39]. These types of sensors do have benefits however. They can often be installed into locations where ultrasonic sensors would be too large to fit. Ultrasonic sensors also have dead zones, which could also be problematic in tight areas.

F. Larrarte et al. (2021) also touches on ultrasonic sensors. These types of sensors have been a well-received solution for long term monitoring stations [39]. By measuring the travel time of acoustic waves which are emitted by the sensor and then reflected by the water surface one can measure the distance from the sensor to the water surface. Ultrasonic sensor mounted above water can measure water height h through the following equation [39]:

$$h = h_0 - \frac{c_{air} T_r}{2} \tag{2.6}$$

Where h_0 is the distance from the sensor to the bottom of the water, c_{air} is the speed of sound in air, and T_r is the travelling time of the ultrasound echo. Likewise, the equation for the water level for ultrasonic sensors mounted at the bottom of the water is given by [39]:

$$h = h_s + \frac{c_{water} T_r}{2} \tag{2.7}$$

where h_s is the vertical distance between the sensor membrane and the ground under water, c_{water} is the speed of sound in water, and T_r being the travelling time of the ultrasound echo.

Aerial ultrasonic sensors, i.e. sensors mounted above in the air above the water surface, are the most widespread [39]. According to F. Larrarte et al. (2021), they are small, cheap, not really prone to drift and require less maintenance than the submerged sensors. However, the measurements become inaccurate if either foam or floating debris covers the surface below the sensor, or if the composition, temperature, pressure, and/or moisture of the atmosphere influences the speed of sound in air [39]. It is also mentioned that one might not necessarily assume that the speed of sound in air is constant. Since the temperature, pressure, composition, and/or moisture of the air could affect the speed of sound enough to cause variations in the measurements, one might need to account for these changes. Measuring additional parameters, such as temperature and air humidity could partially correct these variations [39].

A table listing the advantages and disadvantages of different types of sensors can be found in F. Larrarte et al. (2021), as shown in table 2.3.2. As not all of them are relevant to this paper, so only ultrasonic sensors and pressure sensors have been listed in table 2.3.2.

Table 2.3.2: Pressure/ultrasound comparison, from [39].

Technology	**Advantages**	**Disadvantages**
Piezometric (pressure) sensor	- Continuous measurements - No dead zone - Average investment cost - Works also for pressurized flow - Easy to calibrate	- Contact with the water - Requires regular maintenance as it is sensitive to fouling - Drifts easily
Ultrasonic sensor	- Continuous measurements - Easy to install and maintain - No contact with the effluent - Low drift over time - Rather low cost	- Presence of a dead zone - Does not measure when the water level goes up to the sensor - Several disturbance factors (foams, floats, temperature gradients, haze, etc.)

2.3.3 Systems on the market

To get a sense of what kinds of technology are used today, one can look to complete logging systems which are available on the market. These can provide useful benchmarks for comparison when developing a system for use in measurement of water levels. There are many of these products to choose from. They range heavily in size, cost, lifetime, communication technology, and which and how many sensors they use. This section will only list a few of these products, as these most closely would resemble the system developed in this thesis in form factor and price.

One such system is Onset's HOBO MX2001 [40]. They are pressure-based and send data wirelessly with Bluetooth Low Energy (BLE). They come with different operation ranges, the simplest having an operating range of 0 to 145kPa, approximating to 0 to 4m water depth at sea level. The resolution of these measurements are 0.14cm depth. They weigh approximately 136g, and are run on two AA batteries, lasting for 1 year of logging at 1 minute intervals. The BLE transmission range is 30.5m in line-of-sight. It costs approximately 6000 NOK.

Figure 2.3.3: MX2001, from [40].

Another such system is the EM500-SWL by Milesight [41]. It is based on a gauge pressure sensor, with a customizable range from 0 up to 10m water depth and a resolution of 1cm. It utilizes LoRa for wireless data transmission. It uses an ER34615 battery, and has a lifetime of 4 years when logging at 10 minute intervals. It also costs approximately 6000 NOK.

Figure 2.3.4: EM500-SWL, from [41].

2.4 Summary of literature

The survey done by Mehta, Maru, and Shaha [1], and the survey by Yasin et al. [14] were investigated in Section 2.1. A brief summary of their surveys are shown respectively in Tables 2.1.1 and 2.1.2. The systems discussed in these surveys are designed for different applications relating to measuring water. Many of them do not create any custom embedded hardware for their respective device, and instead use pre-made microcontroller boards such as

the Arduino. This heavily restricts the system's longevity, as these microcontroller boards are not meant for ultra low-power usage. This is also reflected in that most systems discussed were only tested in controlled environments, with a stable power supply. The devices in these papers seem to fit poorly in an outdoor environment. Many of them do use an ultrasonic sensor, which support the findings in the specialization project [2], where it was found that the ultrasonic sensor provides a good trade-off between sensor accuracy and energy consumption.

K. Mekki, E. Bajic, F. Chaxel, and F. Meyer [35] compare different LPWAN technologies and focuses on Sigfox, NB-IoT, and LoRaWAN as they are the leading long-range low-power communication technologies. They compare quality of service, coverage, range, latency, battery life, scalability, payload length, deployment, and cost of each of the three technologies. A simple comparison of the advantages of Sigfox, NB-IoT, and LoRaWAN are shown in Figure 2.2.1. Based on what is discussed in Section 2.2, it seems that LoRa is the best option for the system in this thesis. It provides ample range, with both low energy consumption and low cost of implementation.

Section 2.3 discusses different sensors for use in measuring water level and is based on the work in the specialization project [2]. Two articles are discussed in this section. The first by G. Nikolov, B. Nikolova [38], which implement a system to measure water level in tanks. They also provide a brief comparison of different sensors with their respective advantages and disadvantages, as shown in Table 2.3.1. From this, it again seems that ultrasonic sensors are favored for measuring water level in nature, as they provide a non-contact method for measurement, with no moving parts.

The second article, by F. Larrarte et al. [39], discusses aspects of measuring water in urban sewage pipe systems. It mentions both ultrasonic sensors and pressure sensors as good candidates for the measurement of water level. Ultrasonic sensors, again, for providing non-contact measurements which imply easy deployment and less maintenance. Ultrasonic sensors however, have dead zones in which they cannot provide measurements. They can also be disturbed by several factors, such as foams, floats, temperature gradients, haze, etc. [39]. Pressure sensors are favored where the smaller form factor is advantageous. Pressure sensors can also be used to extract the flow of the water if two sensors are used. This can be done by measuring the pressure in opposite directions, both towards and away from the flow of water. Pressure sensors are however, more susceptible to drifting.

Chapter 3

Design

This chapter covers the functional specification, technical specification, acceptance criteria, and design. The requirements for the complete system will be discussed and a proposal for implementation will be given.

3.1 Overview

A battery powered embedded system is to be designed in this thesis, complete with sensors and a communication module. It is based on the work done in the specialization project [2], where different sensors were tested for how viable they were for measuring water levels. This thesis will continue the work from the specialization project. The goal is to design and implement a complete system for measuring water levels, with a focus on the embedded device.

The system will be designed such that it can be placed in some remote location to collect water level data. It should then transmits its data wirelessly to a LoRa gateway, which then forwards it to an operator's desk with a user interface. The operator's interface in this case can be a computer with a standard operating system, running a custom application. The user interface should allow the the user to view the collected data, as well as to send configuration messages back to the system. Messages are mainly sent back to synchronize the time on the device, or to adjust the sampling interval. Figure 3.1.1 shows a simple block diagram of how the end-device sends and receives data from the user.

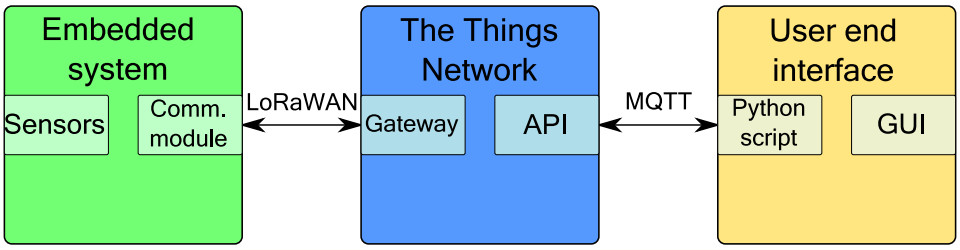


Figure 3.1.1: Block diagram of the system, from sensors to operator's desk.

3.2 Functional specification

The embedded system must be able to collect data outdoors without maintenance. This requires it to be waterproof, and battery powered with a focus on low energy consumption. The system should have a battery life spanning months to be useful. It also needs a wireless communication module such that it can send and receive data. Since many different sensors can be used to measure water level, the system should allow for multiple communication protocols, and preferably simultaneously for the sake of testing. The system should also utilize accurate sensors, which produce repeatable results over long time. To make it more customizable and future proof, it should support easy reprogramming and debugging. The system should also have reset functionality, both manual and through a watchdog timer in case it stops working while deployed. It should have a small form factor such that it can be deployed more easily, and have a relatively low cost to alternatives already on the market. In the case that the wireless link is broken, it should be able to operate without LoRa coverage by saving its data in a non-volatile storage. Table 3.2.1 shows a list of the functional specifications for the system.

Table 3.2.1: Functional specification of the system.

1. The system should be able to withstand harsh weather conditions.
2. The system should support multiple types of sensors.
3. The system should accurately represent water level.
4. The system should be accurate over time.
5. The system should operate wirelessly.
6. The system needs 2-way communication for remote access of data.
7. The system should be capable of being reset.
8. The system needs functionality for reprogramming and debugging.
9. The system should be able to operate for long periods of time without maintenance.

10. The system should have a small form factor (portable).
11. The system should be scalable (cheap).
12. The system should be able to store data in local non-volatile memory.

3.3 Technical specification

3.3.1 Hardware

Based on the functional specification in Table 3.2.1, the technical specifications for the embedded hardware should be as given in Table 3.3.1.

Table 3.3.1: Technical specification of the system hardware.

1. The system should be placed in a waterproof package.
2. The system should cost approximately 1000 NOK.
3. The system should be battery powered.
4. The system should support multiple types of sensors simultaneously.
5. The system should support reading analog voltages.
6. The system should support I2C, UART, and SPI interfaces.
7. The system should have multiple UART interfaces.
8. The sensors should have an accuracy of $\pm$1cm to provide useful data.
9. The system should support an SD card for data storage.
10. The system should be designed such that it has a lifetime of at least a few months.
11. The chips and sensors should have low operating voltage and current draw.
12. The system needs to support a range of input voltages to be more compatible with batteries.
13. The system should have a reset button.
14. The system should support debugging interfaces, along with LEDs to be easy to use and test.

15. The system should have a USB connection for both debugging and charging.
16. The system should have a LoRa module for sending and receiving data.

3.3.2 Software

Based on the functional specification in Table 3.2.1 and hardware specification in Table 3.3.1, the software specifications for the embedded system is as shown in Table 3.3.2.

Table 3.3.2: Technical specification of the system software.

1. The system should utilize a sleep mode to save energy.
2. The system must be able to acquire and process sensor data.
3. The system should be able to send and receive messages through the LoRa module.
4. Should have a real-time clock which is synchronized by incoming messages.
5. The system must be able to store data to the SD card.
6. The system should reset in case of unexpected errors.
7. Support debugging through some interface.

3.3.3 Operator's interface

Based on the functional specification in Table 3.2.1, the technical specification for the operator's interface is as shown in Table 3.3.3.

Table 3.3.3: Technical specification of the operator's interface (user interface).

1. The user interface needs to be able to send time data to the system.
2. The user interface needs to be able receive data from the system.
3. The user interface needs to display the collected data in a GUI.
4. The user interface needs to store the received data in a non-volatile way.

3.4 Acceptance criteria

Based on the technical specifications listed in Tables 3.3.1, 3.3.2, and 3.3.3, the acceptance criteria for the complete system are listed in Table 3.4.1. These will be used as basis for the testing in Chapter 5 to verify the functionality of the system.

Table 3.4.1: Acceptance criteria for the complete system.

Label	**Description**
AC1	The system can send sensor data wirelessly.
AC2	Data collected by the system can be received and displayed on a computer.
AC3	The system can store data on a SD card.
AC4	The system supports most sensor interfaces, such as analog, I2C, and UART.
AC5	The system can detect water level with centimeter accuracy.
AC6	The system can be deployed anywhere with LoRa coverage.
AC7	The system can survive harsh weather conditions, including heavy rain and wind.
AC8	The system can operate for months without maintenance.
AC9	The system can be configured wirelessly.
AC10	The system has a local clock which can be synchronized.
AC11	The system has LEDs which indicate its status.
AC12	The system utilizes sleep modes to save energy.
AC13	The system can reset on unexpected errors.
AC14	The system can send status data wirelessly.
AC15	The system should cost less than 1000 NOK to produce.

3.5 Design

3.5.1 Hardware

A PCB design has been proposed based on the specifications above. Figure 3.5.1 shows a block diagram of the proposed design, and is further detailed in section 4.1.7. The PCB should be powered by an external battery, which is connected to a voltage regulator. The voltage regulator ensures that a variety of inputs are supported, and transforms the voltage to a value which can be used by all components and sensors. A microcontroller must be chosen to support the functionality required by the system. All GPIO pins on the MCU are connected to header pins such that they can be accessed in case the system requires more functionality than expected. A LoRa transmitter is also needed to send measurements wirelessly such that the system can be placed in a remote location. An SD card is also suggested along with the RF transmitter for backing up data in case the wireless communication fails. An array of LEDs are also present on the board to help in both debugging and testing.

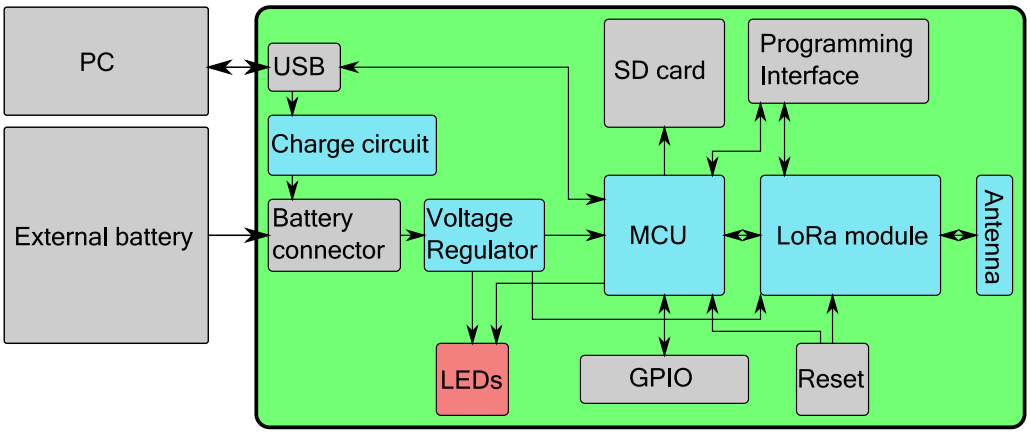


Figure 3.5.1: PCB design block diagram.

Low-power wireless protocols were discussed in the literature review. LoRaWAN was ultimately chosen to be the wireless commuication protocol due to its combination of both cost, coverage, and low energy consumption. It provides an easy to implement solution for energy efficient two-way commuication.

3.5.2 Sensors

This section covers some sensors considered for the system, and discusses how viable they are for measurement of water level in nature for long periods of time. The task for this section is to choose two different sensors, one ultrasonic and one pressure sensor, to be implemented with the system.

The reason for choosing two different sensors is that they both possess their respective advantages and disadvantages, as discussed in the literature review, and as tested in the specialization project [2]. Ultrasonic sensors are easy to use and deploy, provide solid measurements, but are expensive in energy consumption. Pressure sensors provide a smaller form factor and less energy consumption, but the sensor needs to be in contact with the medium and the measurements need to be derived from the pressure. The latter can result in more variable measurements due to changes in the liquid. The system does not strictly need both sensors simultaneously. The final product would only need one of these in fact. However, testing them both at the same time could be beneficial such that one could compare data from both sensors given the same environment. The tests could also provide useful information on which of the two sensors could best satisfy the requirements for the system.

The following paragraphs on ultrasonic sensors are from the specialization project [2], as it would be reasonable to continue using sensors which have proven to be viable from the specialization project.

Ultrasonic sensors

MaxBotix's MB7092 from the XL-MaxSonar series [42] is a weather resistant ultrasonic distance sensor, with an IP67 waterproof grade. It has an operating voltage between 3 to 5.5V, with a average current draw of 2.1mA and a current peak of 50mA. It can operate in temperatures between -40 to 65°C. The MB7092 can measure from 20cm up to 765cm. It costs approximately 900 NOK, depending on vendors. The specifications of the MB7092 make it a good candidate for measuring water level. It is resistant to harsh weather conditions, and has a long enough range to be placed in a more safe location. Its power consumption is also relatively low compared to other sensors of its price range.

Figure 3.5.2: MB7092 sensor, from [42].

DFRobot's A01NYUB [43] ultrasonic sensor is somewhat similar to MaxBotix's MB7092. It boasts waterproofness with an IP67 grade, and an operating temerature of -15 to 60°C. Its operating voltage is between 3.3V to 5V, with an average current draw of 15mA. It has a detection range of 28 to 750cm. It costs approximately 300 NOK. The A01NYUB is very similar to the MB7092 in specifications, but the the main difference this project is concerned about is the conciderably higher current draw.

Figure 3.5.3: A01NYUB, from [43].

Pressure sensor

CUI Devices' PS02-G350KP-4W is a pressure sensor with a range of 35kPa, which is equivalent to 0.35 bar or approximately 3.5m depth in water. It is powered with a 3.3V supply, and has an operating range from -40 up to 105°C. It is interfaced with I2C. It has a small form factor compared to the ultrasonic sensors, and costs approximately 750 NOK. The sensor seems fit for measuring shallow waters.

Figure 3.5.4: PS02, from [44].

The A01NYUB ultrasonic sensor used in the specialization project [2] was also chosen to be used for this thesis, as it has proven to reliably collect useful data as demonstrated in [2]. The pressure sensor tested in the specialization project was discarded due to its high measurement range and relatively low resolution. The pressure sensor replacing it is the PS02 as shown above. It was chosen as due to it being a waterproof pressure sensor, able to be powered by 3.3V. It was also chosen for its low measurement range, as this provides a better resolution when measuring water level only a few meters deep.

3.5.3 The Things Network

The Things Network (TTN) is a global LoRaWAN network for IoT devices. It is possible to use private LoRa gateways for communicating with end-devices, but TTN was chosen for its coverage. It allows the system to broadcast to its messages to be picked up by any gateway in reach. While using TTN somewhat limits the customizability of the gateway, it is outweighed by the fact that it allows the system to be deployed in many more locations due its wide coverage. The Things Network also has an online console which can be used for faster implementation, and an API. The API can be used with HTTP or MQTT, such that messages can be sent to TTN from a back-end and then

forwarded to the embedded system.

3.5.4 Software

The software required on the embedded system has to be made to periodically perform five tasks:

1. Handle incoming configuration messages
2. Retrieve data from sensors
3. Transmit the sensor data
4. Save data to an SD card
5. Sleep

Due to the simplistic flow of operations, it would be reasonable to implement the as a state machine. The details of the software implementation are discussed in detail in section 4.2.1.

The complete system also needs software to send receive data on the user end. This back end application should be able to communicate with TTN, to be able to send messages to, and receive messages from, the system. A database could prove useful for storing large amounts of data. Since the collected data mostly consists of measurements from two different sensors, a database with two tables would suffice. A simple database schema is proposed and shown in Figure 3.5.5. The user end application should also have some GUI to allow for plotting of collected data.

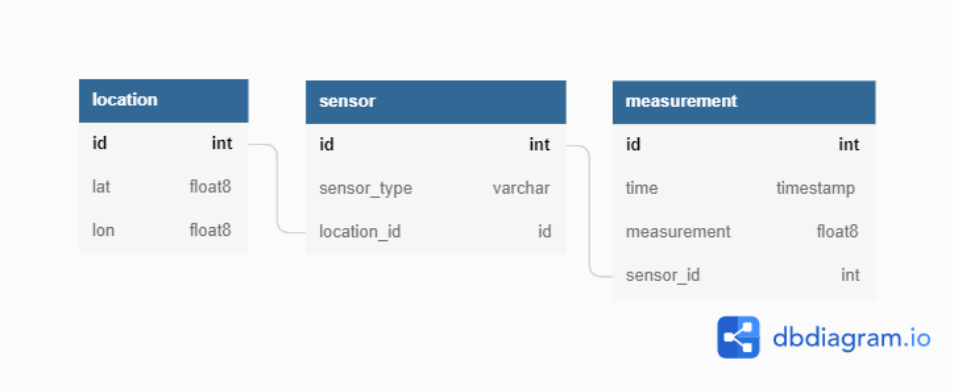


Figure 3.5.5: Database diagram.

A simple GUI is proposed to be implemented as shown in Figure 3.5.6 such that the collected sensor data can be retrieved from the database and displayed with ease.

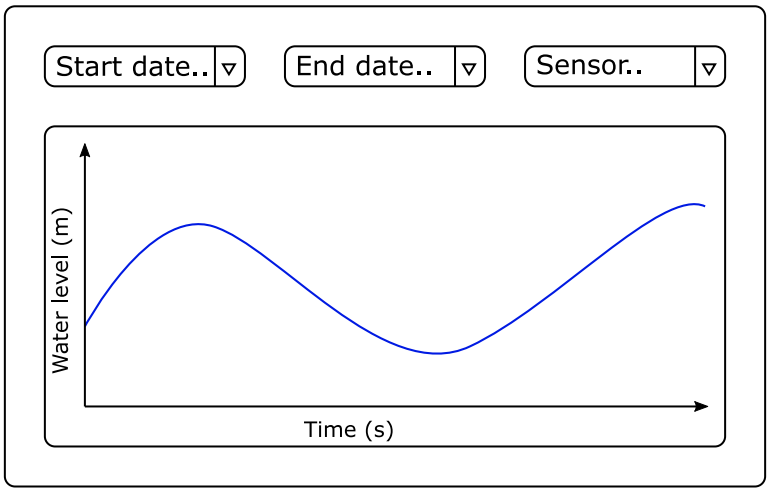


Figure 3.5.6: Proposed design for a GUI displaying collected sensor data.

Chapter 4

Implementation

This chapter covers how the system was implemented. It covers how the circuitry was made as well as a detailed explanation of the accompanying software. The PCB was designed in *EasyEDA* and the software for the device written in C++ using *Microchip Studio*. The user back end was made with Python.

4.1 Description of hardware

The embedded hardware for the system is based on the work done in the specialization project [2]. The PCB design is similar, with both new and improved features. The remaining hardware for this thesis is already implemented in the sense that The Things Network's gateways and sky service are used, and any common computer can be used to run the back-end software. This section covers the implementation of the embedded hardware.

4.1.1 Components

ATmega324PB

The ATmega324PB is an 8-bit microcontroller based on the AVR RISC architecture, and was developed by Microchip. It features 32KB of flash program memory, 1KB EEPROM, and 2KB SRAM. It supports three programmable serial USART interfaces, two SPI interfaces, and two I2C interfaces. It also has an internal ADC, GPIO, 8 and 16 bit timers/counters, sleep mode, and more. It was primarily chosen for two reasons. Firstly due to time constraints for the implementation of the system. The ATmega324PB was used previously in the specialization project [2], as well as other courses. The previous experience with the ATmega made it an easy choice for this reason. Secondly, it was chosen due to the lack of availability. Many newer and more suitable mi-

crocontrollers were out of stock with long lead times during the developement of this system. Microcontrollers from STMicroelectronics and the EFM32 by Silicon Labs support the same features as the ATmega324PB, but with lower power consumption. These were also considered in the process of making the system.

LDL1117S33R

The LDL1117 by STMicroelectronics supports input voltages between 2.6 and 18V. The LDL1117 has several models with different fixed output voltages. An output voltage of 3.3V was chosen to be suitable for this system, such to provide all electronics with the correct voltage. It was also chosen for its low quiescent current to save on energy consumption, as a part of the specification.

MCP73830L

The MCP73830L is a single-cell li-ion battery charge management controller made by Microchip. It requires few external components, and the constant-current value is set by one external resistor. The chip supports charge currents up to 1000mA. It supports input voltages between 3.75 and 6V, meaning that the battery can be charged with 5V supplied through USB.

CH340C

The CH340 is a USB bus conversion chip, creating USB to UART interfacing. It supports both 5V and 3.3V power voltage. The CH340C model has a built-in crystal, reducing the number of components needed to realize the implementation. It was chosen to create a simple USB-UART interface to connect the system to a computer through USB for debugging.

74LVC1G125QW5-7

The 74LVC1G125Q is a non-inverting buffer/bus driver with a 3-state output. Its operating voltage ranges from 1.65V to 5.5V. The inputs support voltages up to 5.5V, allowing the device to support a mixed-voltage environment. It is used together with the CH340C to implement the USB-UART interface.

RN2483A

The RN2483 by Microchip is a low-power LoRa transceiver module. It features an on-board LoRaWAN protocol stack and ASCII command interface over UART. It operates on voltages between 2.1 and 3.6V, the typical being 3.3V. It has an integrated MCU, crystal, EUI-64 node identity serial EEPROM, radio

transceiver with analog front end, and matching circuitry. It features 14 GPIO, and 13 analog inputs. It was chosen due to being an easy-to-use, low-power solution for LoRa data transmission. The typical idle current consumption is 2.8mA, the typical transmit current being 38.9mA, and sleep current at approximately 16µA. It is however an old model. The main drawback of this chip is that it has relatively high energy consumption compared to newer LoRa modules. One such newer chip being Microchip's WLR089U0. The RN2483 was chosen mainly because of the lack of availability for other LoRa modules.

4.1.2 Power circuit

The schematic for the power circuit is as shown in Figures 4.1.1 and 4.1.2. JP1 represents a screw terminal which can be connected to a battery. The battery is then connected to the voltage regulator U6 and the battery charging circuit which is shown in Figure 4.1.2. The voltage regulator U6 supports a wide range of input voltages, ranging between 2.6 and 18V as stated in the data sheet [45]. It outputs 3.3V which powers the rest of the system excluding the battery charging circuit. Both the input and the output of the voltage regulator are decoupled with capacitors C4 and C7. LED3 from Figure 4.1.3 is used to indicate that the system is powered on.

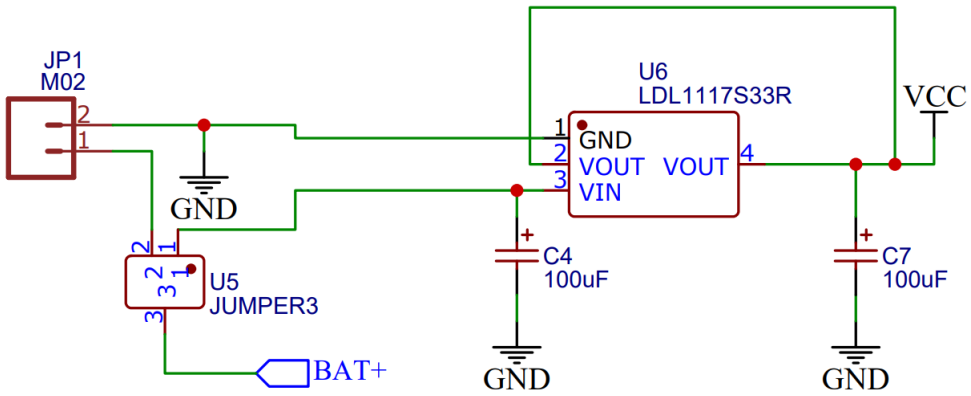


Figure 4.1.1: Schematic of battery connection and voltage regulator.

As an optional feature, a Li-Ion battery charging circuit was implemented as shown in Figure 4.1.2, similar to what was done by Rasmussen [46]. This was done to simplify the use of the system, but not strictly required. For instance, it is not needed in the case a non-Li-Ion battery is used, or in the case that the USB connection is used without the need to recharge the battery. It was also made optional because the MCP73830L available for order had a very small footprint. As the board was to be soldered by reflow (by hand), it was made optional in the case that the soldering was not good enough.

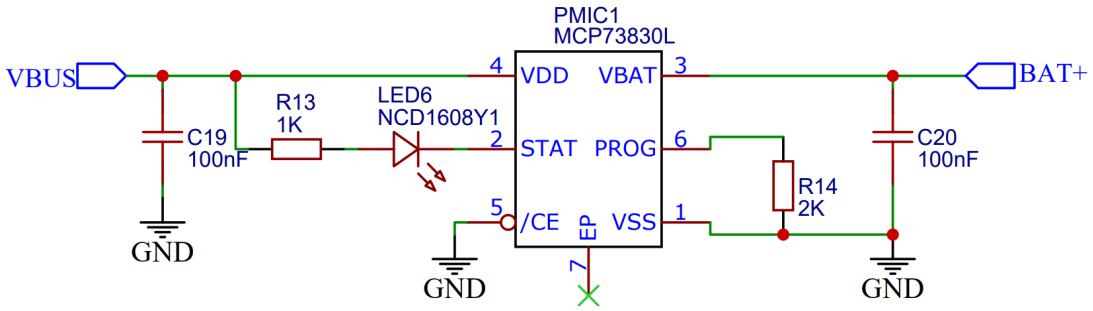


Figure 4.1.2: Schematic of battery charging circuit.

The battery charger can be used by connecting the VBAT output of the PMIC1 in the charging circuit to the battery through the jumper U5. The charging circuit is powered by the $5V$ provided through the USB connection USB1 as shown in Figure 4.1.6. The battery charge management controller is connected similarly to what is suggested in its data sheet [47]. C19 and C20 were used to create a low impedance path to reduce noise, and the resistor R14 is used to control the charge current. The charge current was selected by using the following formula:

$$I_{charge} = \frac{1000}{R} \tag{4.1}$$

where I_{charge} is the charge current in mA, and R is the value of the resistor R14 in Ω in the schematic. R14 was chosen to be 2kΩ to provide a charge current of 0.5mA. Additionally, LED6 is used to indicate the charging status.

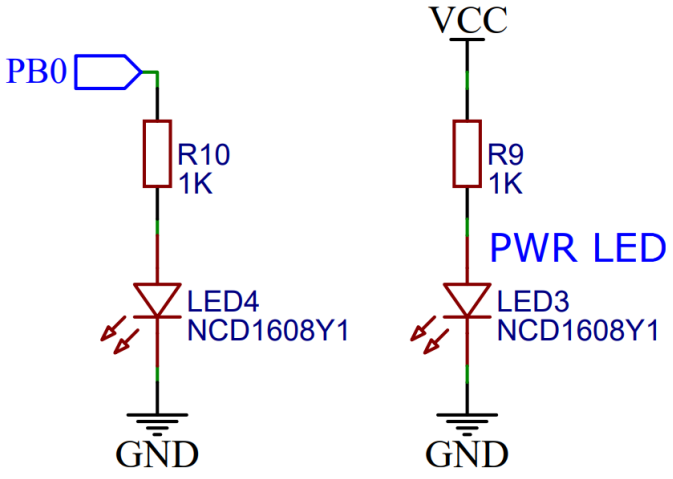


Figure 4.1.3: Schematic of two LEDs used for showing status.

4.1.3 RF circuit

Figure 4.1.4 shows the schematic for the LoRa transmitter module. The RN2483 was chosen to allow for wireless communication. RFH and RFL on the RN2483 are connected to two antenna connectors, JP2 and JP3. The

module's power inputs are decoupled with C5 and C6 to provide a more stable power voltage. The RN2483 is interfaced with UART to the ATmega324PB, with resistors R6 and R19 with a value of 0Ω. The resistors were used in case the two UART lines needed to be swapped or used for something else. The LoRa module is also connected to the global reset button.

There was an oversight made when designing the circuitry around the RN2483. The ICSP pins are not connected, but according to the data sheet [48] this should not pose a problem. As stated in the data sheet: "PGC_INT (Pin 30) and PGD_INT (Pin 31) are internal program pins used during manufacturing. For normal operation, these pins can be left unconnected. The normal firmware upgrade method is through the internal bootloader of the module via the UART. The method is documented in the [data sheet]. However, for backup firmware update purposes the user can place a 6-pin ICSP header on their host PCB with PGC_INT (Pin 30), PGD_INT (Pin 31), RESET (Pin 32), power and ground." As such, the ICSP pins are not strictly necessary for this system.

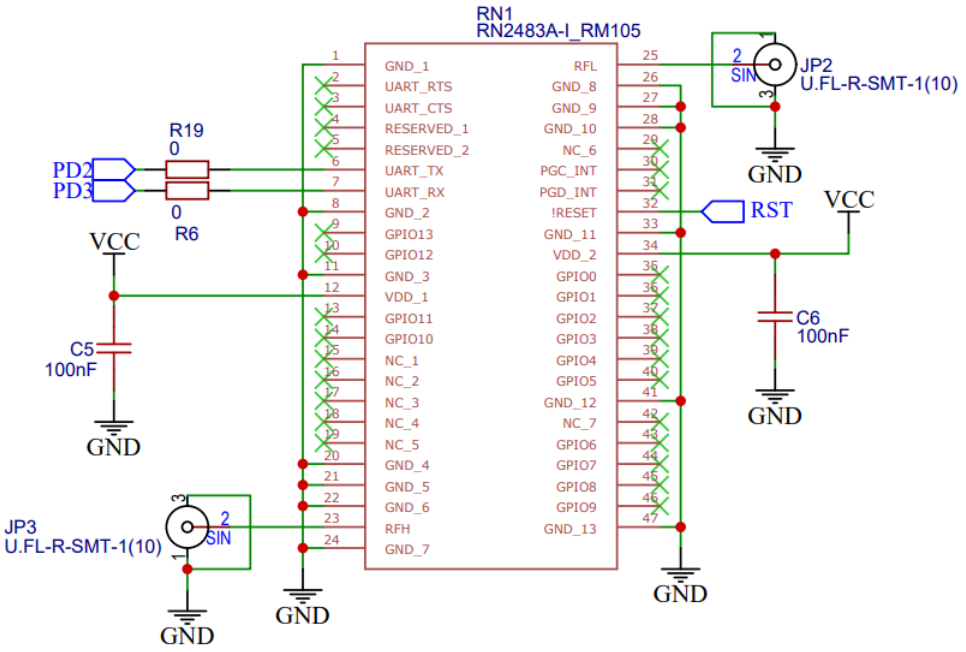


Figure 4.1.4: Schematic of LoRa transmitter module.

4.1.4 Microcontroller

The schematic surrounding the micro controller is shown in Figure 4.1.5. The supply voltage inputs on the ATmega have been decoupled with several capacitors to provide a more stable voltage level. C10, C12, C13 have been used to decouple VCC, AVCC and AREF on the ATmega. While the ATmega has an internal oscillator, an external crystal X1 has been added in case it is needed. This is connected to capacitors C8 and C9 to reduce noise.

All useable pins on the micro controller have been connected to header pins so they are readily available in case any of them are needed. This was done to

provide a more flexible design for testing, and many of them are not strictly needed. The pins include PA0-7, PB0-7, PC0-7, and PD0-7. PB0 is also connected to LED4 as shown in Figure 4.1.3. This is to make debugging slightly easier, as the pin can be made to signal that the system is running correctly after being deployed. The micro controller is also connected to the J1 and H1 headers. These are respectively the AVR ISP and JTAG headers, to allow for programming and debugging.

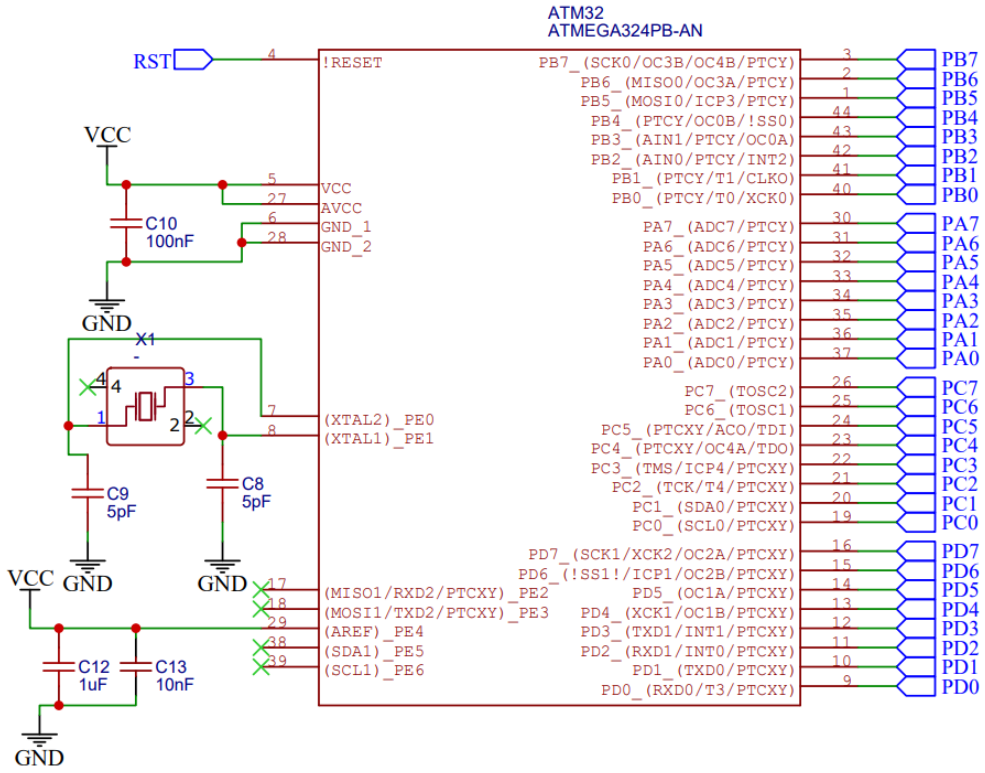


Figure 4.1.5: Schematic of ATmega324PB.

4.1.5 USB

The schematic for connecting the ATmega to an USB interface is shown in Figure 4.1.6. This circuitry has been included to allow for easier connection to a PC to allow for live debugging. The USB port is connected to the CH340C with a fuse F1. VCC on the CH340C is also decoupled with capacitor C11. Both U2 and U3 are 74LVC1G125QW5-7 bus drivers to interface the CH340C with the ATmega. U2 and U3 have been connected to LED1 and LED2, which respectively indicate data transfer on TX and RX on the ATmega. U2's and U3's VCC pin have also been decoupled with capacitors C2 and C3.

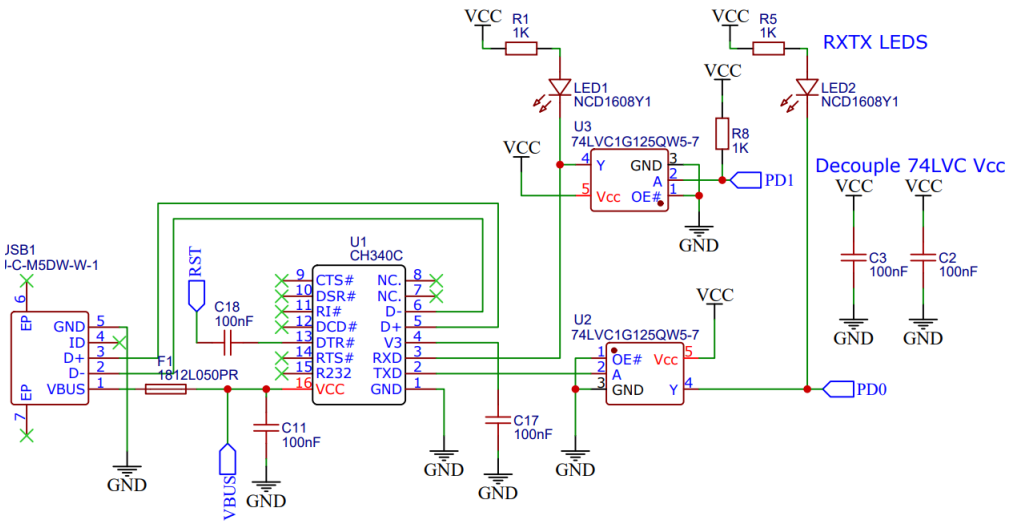


Figure 4.1.6: Schematic showing the circuitry required for USB connection.

4.1.6 Peripherals

A reset button has been implemented as shown in Figure 4.1.7. The RST line is usually pulled up through the resistor R4, and can be pulled low using the RESET button. This passes current through LED5 to indicate that the system is being reset. The capacitor C1 is used as a low-pass filter to smooth out the voltage drop.

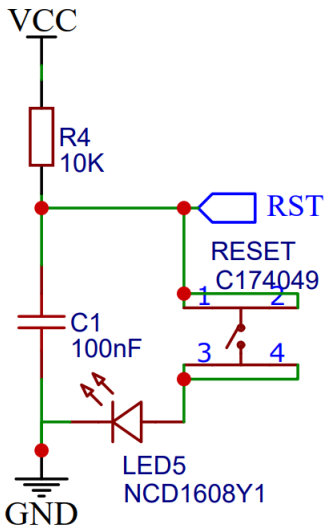


Figure 4.1.7: Schematic of reset button.

The ATmega has been connected to an SD card using the micro controller's SPI interface, with pins PB4, PB5, PB6, and PB7 on the ATmega. This is shown in Figure 4.1.8. The resistors R2, R3, R11, and R12 pull up each pin. The resistors R7, R15, R16, and R17 are 0Ω resistors, which provide a footprint on the PCB to reconnect the SPI lines in case they have been connected incorrectly.

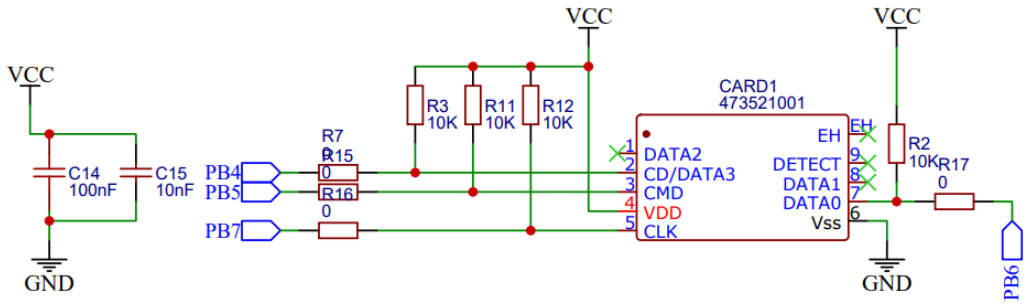


Figure 4.1.8: Schematic of SD card insert.

4.1.7 Printed circuit board

The circuit board was designed as a 4-layer PCB. The top and bottom layers were used for traces, while the two middle planes were used as a 3.3V power plane and a ground plane to simplify routing. The top plane has also been covered with a ground area. An image of the final PCB is shown in 4.1.9. The traces for the PCB, excluding the middle planes and top layer area is shown in Figure 4.1.10.

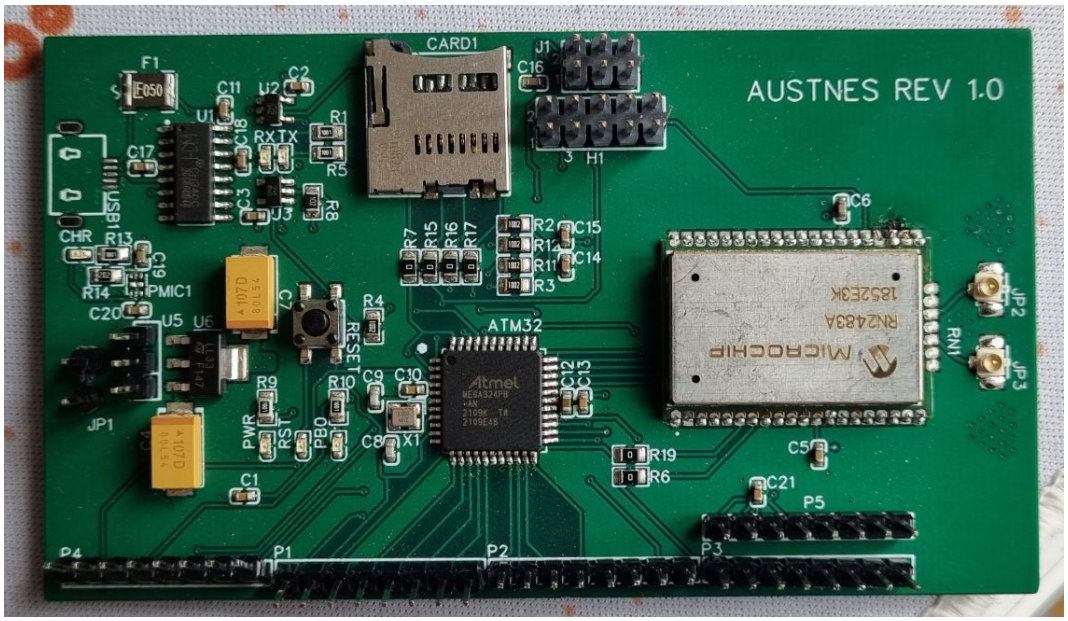


Figure 4.1.9: Image showing the final PCB.

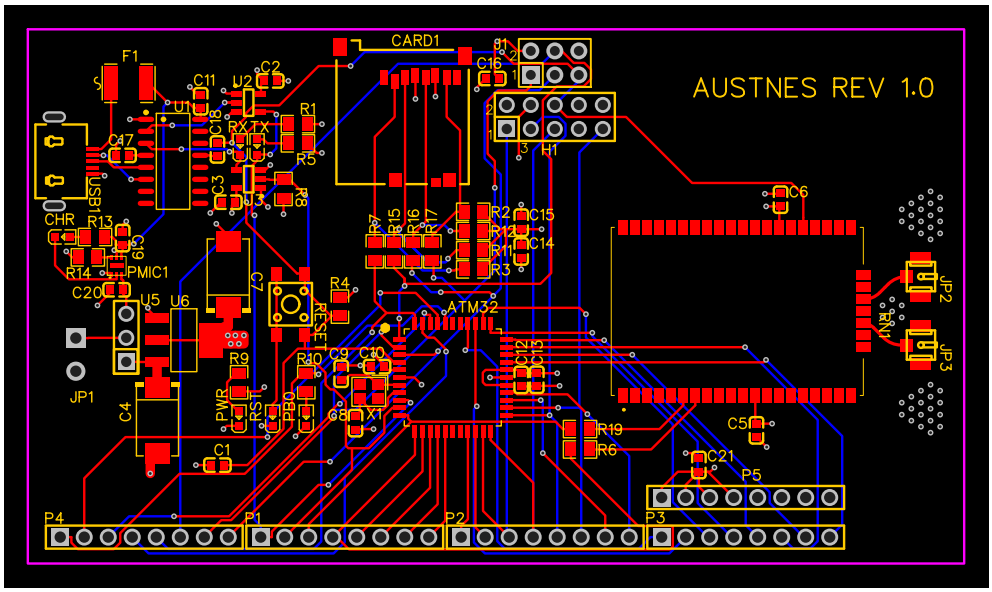


Figure 4.1.10: PCB trace without planes.

The PCB was designed in *EasyEDA*, and ordered from *JLCPCB* at https://jlcpcb.com/. The *EasyEDA* Designer is a well-known web-based PCB design tool found at https://easyeda.com/. Most parts were also ordered from *LCSC* found at https://lcsc.com/. All three companies are owned by the JLC Group, based in Shenzhen, China, which made the ordering process somewhat easier. Some parts were also ordered from *Mouser*, based in Texas, USA. Their website can be found at https://no.mouser.com/. The header pins, jumper cables, and the RN2483A were bought at the electrical workshop on campus. The last component due to limited availability online. The part list for the PCB can be found in Appendix A.

The RF traces were chosen to be 0.34mm to impedance match the antennas, using *JLCPCB*'s impedance calculator. As *JLCPCB* manufactured the PCB, it was a natural choice to calculate the trace width with their tool. The routing was mainly done through *EasyEDA*'s routing tool, which automatically routes selected labels. Some manual rework was performed on the traces to simply the routing, as the automatic routing tool was not deemed optimal. Several vias were also placed near the antenna connectors to reduce noise.

All SMD components were soldered on through reflow. The paste was applied by hand, and an oven at the electrical workshop on campus was used to heat the board up to the correct temperature.

A feature which was implemented only after ordering the PCB was to use a transistor to toggle the sensors on and off with each measurement. This was done as it was realized only after ordering, and ordering a new PCB at the time would not be feasible due to time constraints. By connecting one of the GPIO pins of the ATmega to the transistor, one can toggle whether or not the sensor is connected to the voltage supply. This is shown in Figure 4.1.11. The sensors need some time to adjust after being turned on, but this is outweighed by the

fact that they draw very little current when the system is sleeping, which is considerably longer than the time it takes to sample data. After testing, this proved to save some energy consumption. This is further discussed in Chapter 5.

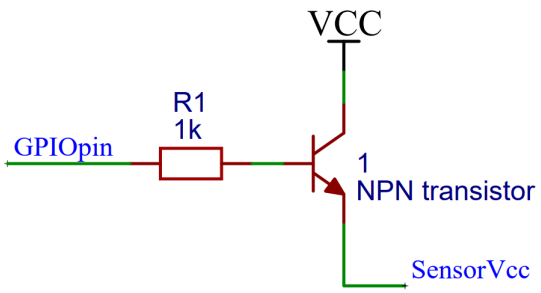


Figure 4.1.11: Transistor connected to ATmega GPIO pin, V_{cc}, and the sensor voltage input.

The complete schematic for the circuit board is shown in Figure 4.1.12, and a higher resolution schematic can be found in Appendix A.

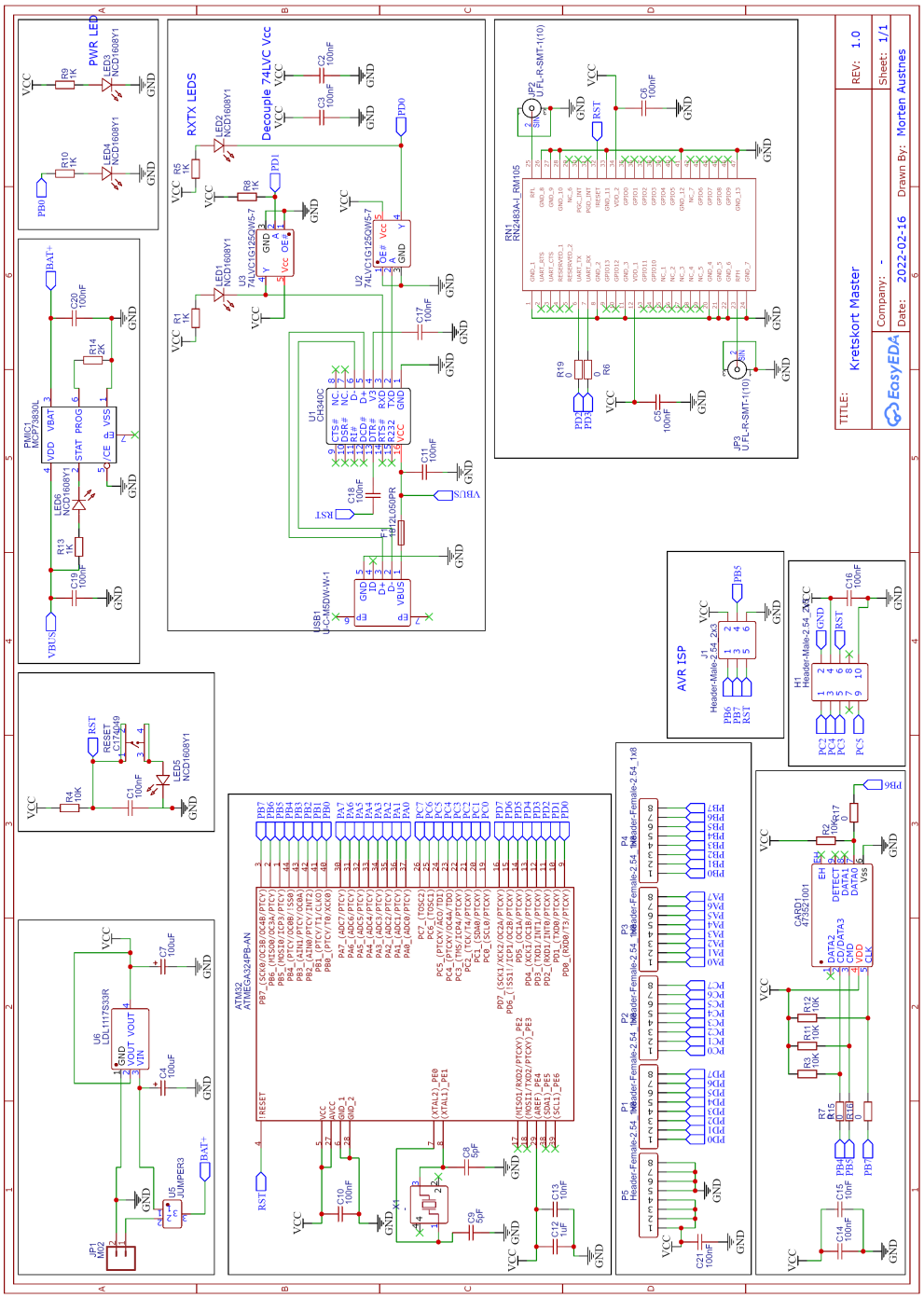


Figure 4.1.12: Complete schematic of the system.

4.1.8 Battery

The USB connection was not implemented due to time constraints. This meant that the recharging IC did not get the 5V it needed to recharge batteries. Therefore, an easier solution would be to use a power bank, which can easily be recharged by a computer. A 10Ah power bank was chosen as the battery for the system.

4.1.9 Box

A container and stand was made to store the embedded hardware. Figure 4.1.13 shows a simple schematic for the setup, and Figure 4.1.14 shows the final result, and figures 4.1.15a and 4.1.15b show images of the box with sensors mounted to it and hardware inside. It was made at NTNU's workshop in the Electrical Engineering building.

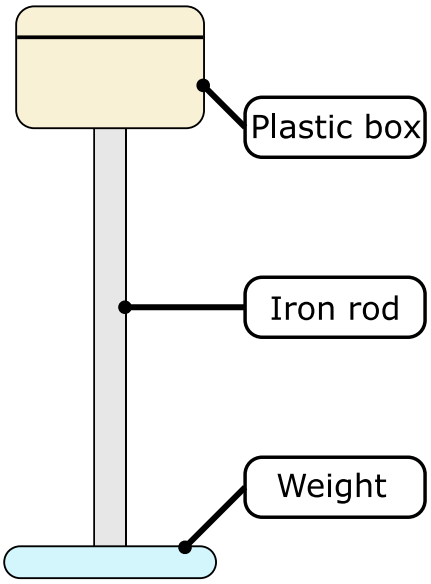


Figure 4.1.13: Simple schematic showing the container and stand for the embedded hardware.

Figure 4.1.14: Image of the final container and stand for the embedded hardware.

(a) Image of box for containing hardware.

(b) Image of box for containing hardware.

Figure 4.1.15: Sensors mounted for long term testing outdoors.

4.2 Description of software

The software for the ATmega was developed using Microchip Studio 7.0, and flashed and debugged using a JTAGICE3. Microchip Studio allows for both programming and debugging of the device, as it provides direct support of the JTAGICE3. The software was written in C++. Examples from the RN2483 code examples [49] and the A01NYUB ultrasonic sensor documentation [50] were used as inspiration to create the software necessary to operate the RN2483 LoRa module and the ultrasonic sensor respectively. The following sections cover the implementation of the software for the system.

4.2.1 State machine

This section covers the software for the system, implemented as a state machine as proposed in Section 3.5.4. Figure 4.2.1 shows the state diagram of the system, excluding the ISR. The ISR is described in detail in Section 4.2.1.

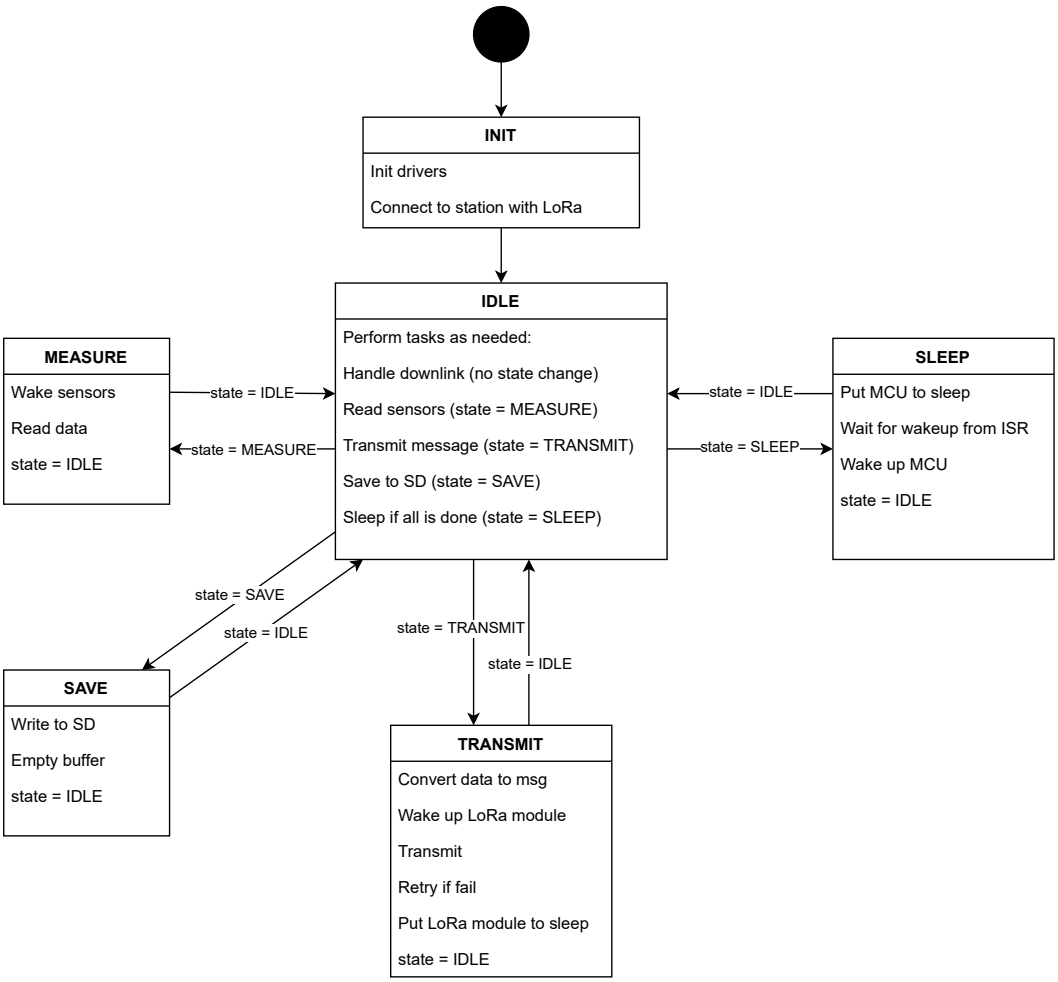


Figure 4.2.1: State diagram of the system's normal state flow.

Idle state

The idle state functions as the main state which checks a set of conditions and sets the new state accordingly. The program has to go through the idle state to reach all other states except for the initialization. The system is designed to perform a few simple tasks as needed, and then go to sleep for some time. Therefore it would be natural to implement the idle state as it is. The tasks which need to be performed by the system include handling configuration messages, reading the sensor values, sending those values, and saving those values to the SD card. After there is nothing left to do, it goes into the sleep state.

The conditions which the idle state checks are as illustrated in the flowchart in Figure 4.2.2. Firstly, it checks if there are new downlinks received by the LoRa module. Secondly, the sensors are read if they have not already been read this wake cycle. Then, if the sensors have been read, those messages are sent wirelessly through the LoRa module. The sensor data is stored in a buffer temporarily, and if that is full, the data is then stored to the SD card. This is to avoid writing to the SD card more times than necessary. Lastly, if everything is complete, the system goes to sleep.

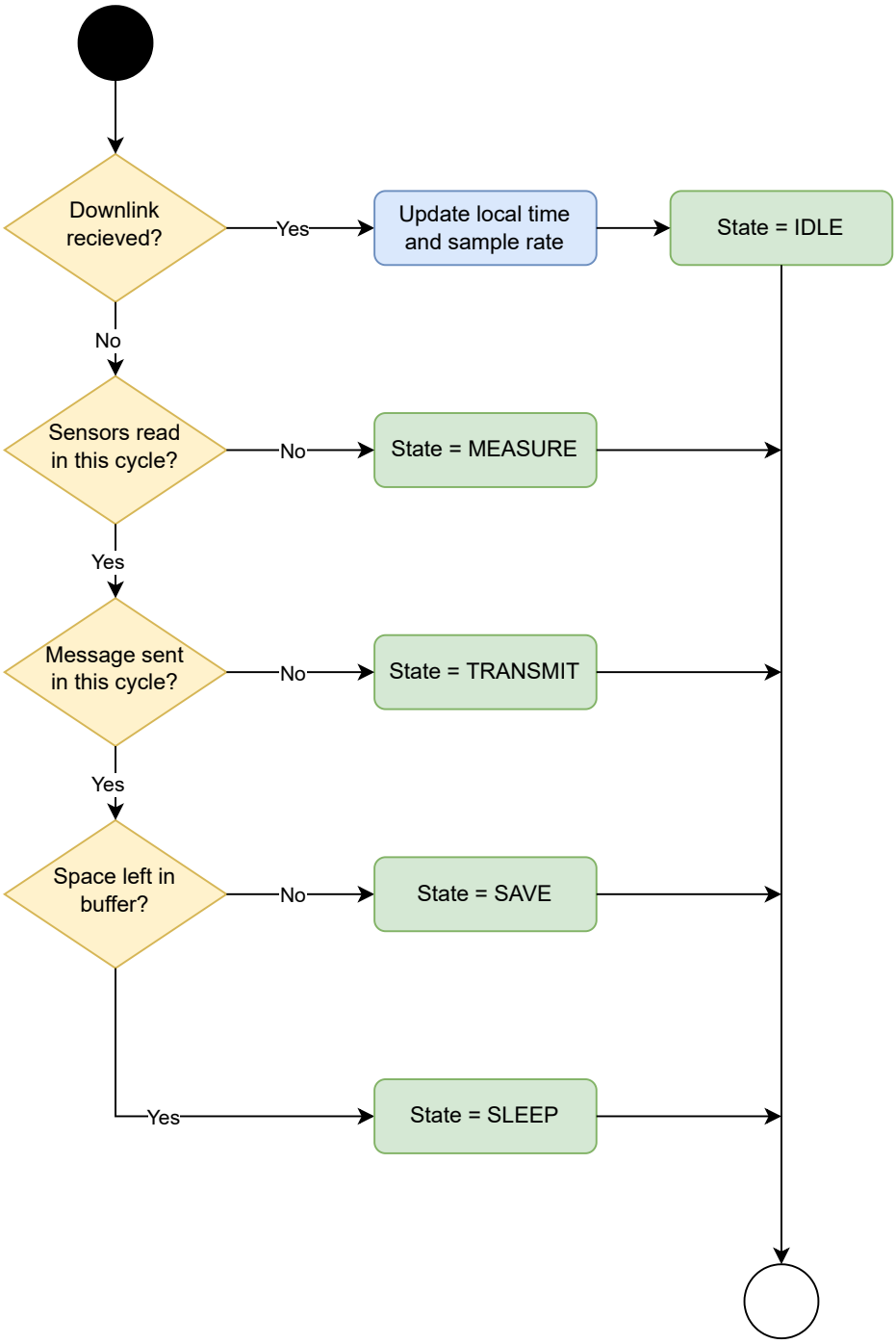


Figure 4.2.2: Flowchart of the operations in the system's idle state.

Measure state

The measure state is used to retrieve data from both the ultrasonic sensor, the pressure sensor, and the ADC. The ADC is used to read the battery level

of the system. Firstly, the state powers up the sensors. The ultrasonic sensor requires reading four bytes over UART per measurement, two of which are used for error detection in the data. If the system is configured to average over several samples, the sensor data is stored, and more is collected until it is sufficient. The state then moves on to collect the pressure sensor data. This requires two bytes over I2C. As with the ultrasonic sensor, more data is collected and averaged if needed. The ADC is then read to extract the battery level. This measurement is averaged regardless, due to the built-in ADC in the ATmega being noisy. Finally, the sensors are put to sleep, and the state is changed to idle again. The functionality of the measure state is as shown in the flowchart in Figure 4.2.3.

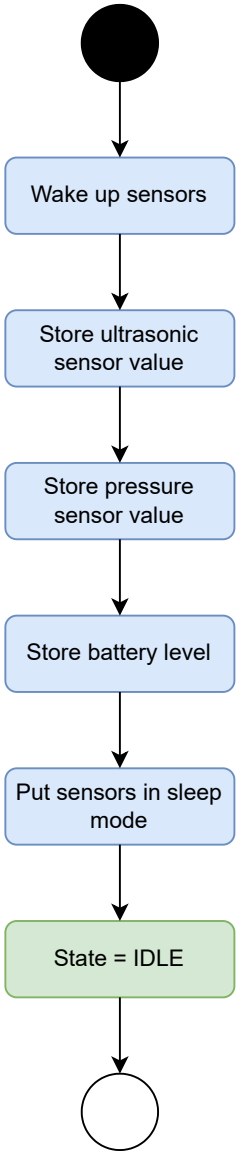


Figure 4.2.3: Flowchart of the operations in the system's measure state.

Transmit state

Before each transmit, the software converts the local timestamp, battery level, ultrasonic measurement, and pressure measurement into a message payload.

Table 4.2.1: Table of the data packed for the LoRa message payload.

Uplink message payload								
Unix timestamp (4 Bytes)				Battery level (1 Byte)	Ultrasonic data (2 Bytes)		Pressure data (2 Bytes)	
Bit 31-24	bit 23-15	Bit 15-8	Bit 7-0	Bit 7-0	Bit 15-8	Bit 7-0	Bit 15-8	Bit 7-0

Afterwards, the RN2483 is woken up and the message is sent. The system then waits for a confirmation message. If no confirmation message is received, the message is resent until either a confirmation is received or too many retries have been attempted. The LoRa module is then put into sleep mode to save on energy consumption.

A separate “alive” message is not sent by the system to tell whether it still functions. The sensor data transmission is the only message that is sent by the system, and therefore it also functions as a “alive” message. This leaves the user with less information about whether the system is operating or not, but it is a trade-off made to save on energy consumption. No single data point is considered valuable enough to be sent indefinitely, so it is better to discard a message and move on to the next one rather than to resend it until the measurements are outdated. The functionality of the transmit state is as shown in the flowchart in Figure 4.2.4.

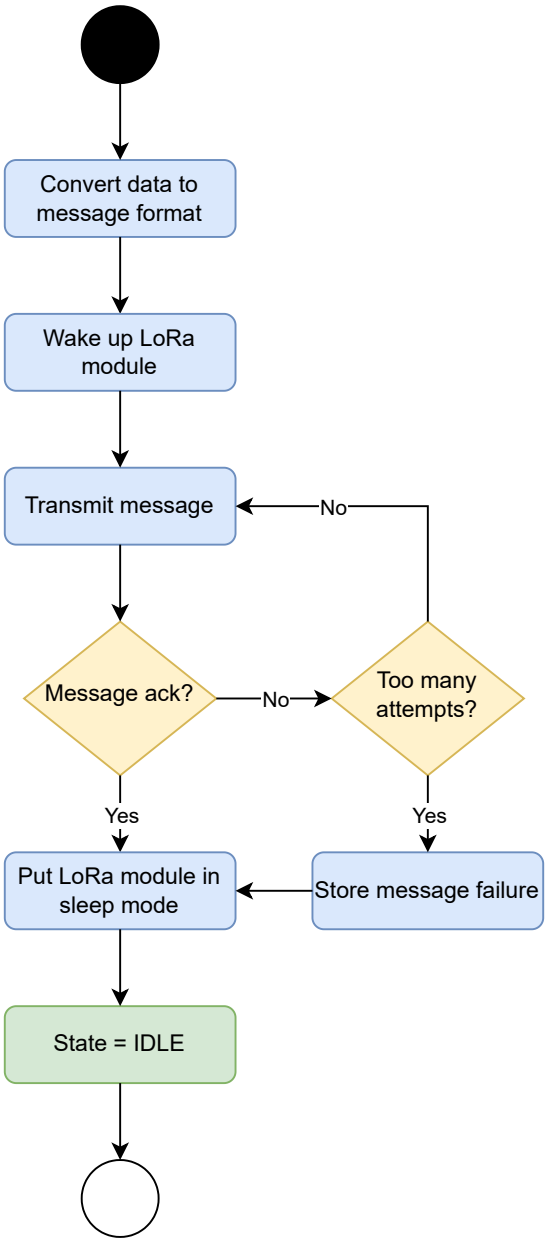


Figure 4.2.4: Flowchart of the operations in the system's transmit state.

Save state

The save state is entered when the buffer of sensor data is full. Here, the buffer is transferred to the non-volatile memory of the SD card, and the buffer is emptied. This is done to avoid writing to the SD card more than necessary, as it could reduce the lifespan of the card. A flowchart of the functionality of the save state is as shown in Figure 4.2.5.

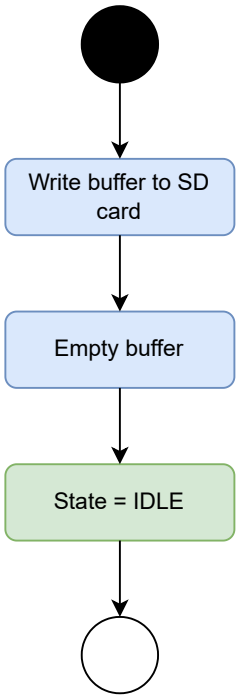


Figure 4.2.5: Flowchart of the operations in the system's save state.

Sleep state

The sleep state utilizes the ATmega's power-down mode, as described in the data sheet [51]. This is to significantly save on energy consumption while the system is in essence just waiting for some time to pass before performing the next measurement. While in this mode, however, the MCU can only be woken up by an interrupt. Timer 2 can still run in power-down mode, so the ISR can continue to keep track of time and be responsible for waking up the MCU after enough time has passed. This functionality is further explained below. A flowchart of the functionality of the sleep state is as shown in Figure 4.2.6.

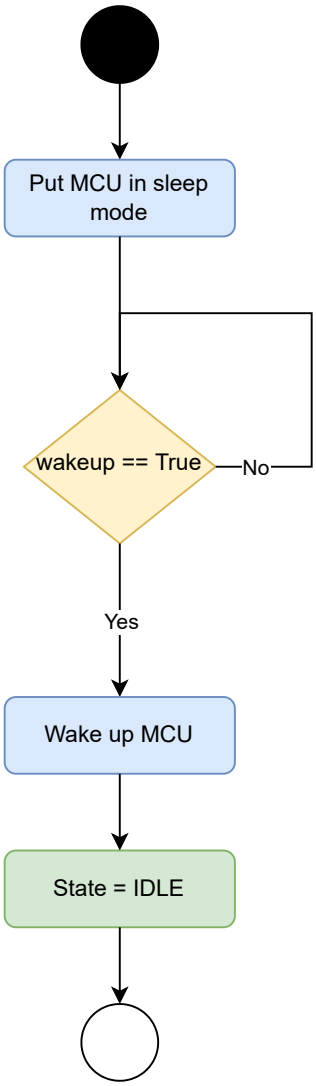


Figure 4.2.6: Flowchart of the operations in the system's sleep state.

ISR

The ISR serves as a way to both maintain a real-time clock, and to wake the microcontroller up from sleep mode. It uses Timer 2 on the ATmega to increment the unix_time variable. The variable holds seconds in unix time, as described in section 4.2.2. It increments the unix time every time it is called, and then, if the MCU is currently sleeping, it compares the current time to the scheduled wake-up time. If sufficient amount of time has passed, it will schedule a new wake-up time and start the MCU again to take new measurements.

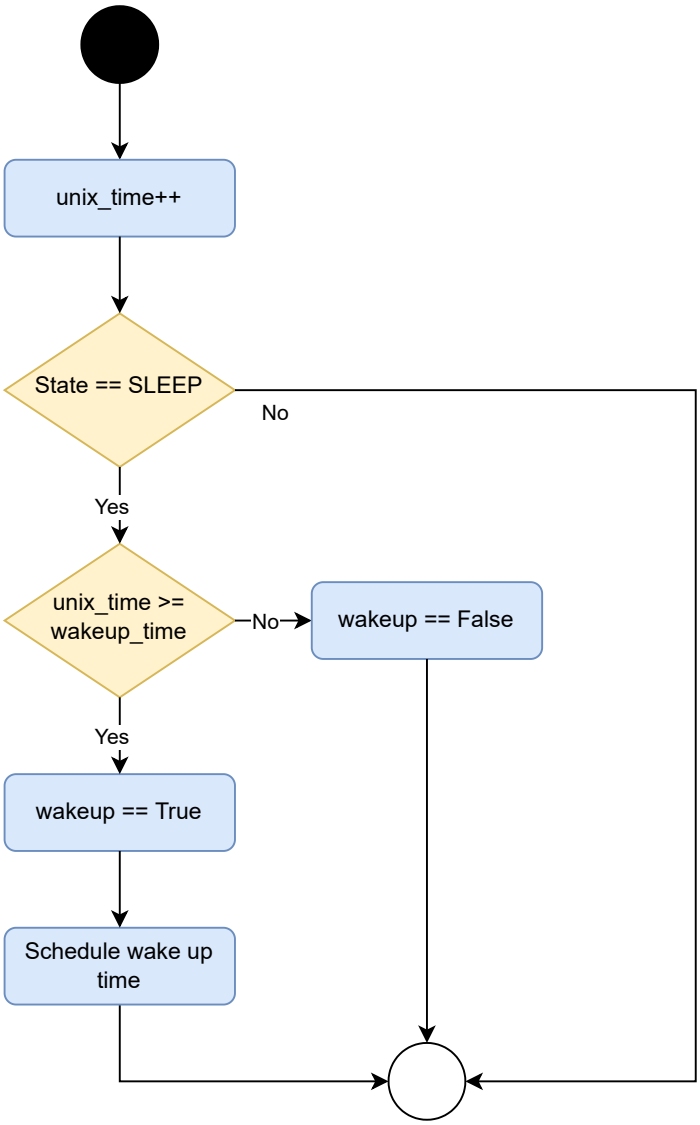


Figure 4.2.7: Flowchart of the operations in the system's ISR.

4.2.2 RTC

The system requires some knowledge about time to be able to store and send sensor data, as well as to sleep for a designated amount of time. An implementation of a real-time clock for the ATmega was done by Rasmussen [46]. A similar design has been chosen to be implemented for this system as well. As done in [46], the RTC utilizes Timer 2 on the ATmega, and an ISR to keep track of time. A timer with millisecond resolution was created by Rasmussen, but it was chosen to only implement a whole second timer for this system. The decision was made as it was not deemed strictly necessary to utilize so small time increments. The system satisfies its acceptance criteria even though it can only sleep for integer amount of seconds. The measurements are also deemed accurate enough when the time of measurement is rounded to seconds.

The real-time clock is implemented using Timer 2, an 8-bit timer, on the

ATmega, using a 32.768kHz crystal. The reason for choosing Timer 2, is that it is still enabled in power-down mode, as described in the data sheet [51]. This is the easiest way to trigger a timed interrupt while the MCU is sleeping. The crystal frequency value was chosen as it is equivalent to 2^{15}, as done in [46]. This proves useful as the 8-bit timer can have its prescaler set to 128, such that it overflows every second. This can then be used to call the ISR, which increments the unix time by a second, as described in 4.2.1. A prescaler of 64 was used in [46], such that the timer overflows every 0.5 second. This resolution was again deemed unnecessary for the system in this thesis.

4.2.3 Watchdog

The ATmega has a built-in watchdog timer, as described in the data sheet [51]. It uses a 128kHz internal oscillator with a prescaler between 2K and 10K. This can provide a timeout between 16ms and 8s. The timeout can then trigger either a reset, ISR, or both. As it could take more than 8 seconds to receive an acknowledgement, a longer period must be used for a watchdog timeout. Similarly as implemented in [46], an ISR was created to count how many timeouts have been called, and to then trigger a reset if a certain threshold of timeouts have been called. The watchdog could prove useful in the case of deadlocks, or bugs in the code which could cause the program to spin. Due to the long waiting time for acknowledgement messages, the timeout was chosen to be 180 seconds.

4.2.4 ADC

The ATmega has 7 channels with 10-bit ADC as described in the data sheet [51]. This can be used to read the battery voltage using a voltage divider. This lets the user see how much battery life the system has left.

4.2.5 Other drivers

A UART driver had to be written for the system. The ATmega has three separate UART channels, where UART0 and UART1 are used for the ultrasonic sensor and the RN2483 LoRa module respectively. UART0 is used to read four bytes at a time from the ultrasonic sensor, such that it can retrieve the distance information from the sensor.

UART1 is used to send commands to the RN2483. It has a multitude of commands, ranging from setting communication parameters, sending and receiving data, and also putting it into sleep mode. The sleep mode is important as it saves on energy consumption while the system is not active. It was implemented based on documentation (code examples) from Microchip [49]. The RN2483 was set up for OTAA join procedures, with the keys auto-generated

by TTN. When a message is received, an internal flag is set such that the received message can be handled by the state machine. This is described further in 4.2.1. The RN2483 also allows for sending and receiving messages on different ports, but this was not implemented due to time constraints.

Since LED4 was connected directly to the GPIO pin PB0, a simple driver was written to make it blink when booting, to signal that the initialization was successful. It was also used for some simple testing, such as making sure the real-time clock ticked every second.

4.3 Recieving data on computer

A working GUI and database was not implemented due to time constraints. A Python script was used to fetch data through TTN's API and stored the data in a file.

TTN provides an easy way to connect LoRa nodes to a computer/server. TTN was used to set up an application server, which multiple end-devices can connect to. It generates EUIs and access keys automatically for the application, as well as supports APIs such as MQTT, as described in 3.5.3. The console itself can also be used as a GUI for configuring the devices. Screenshots of the TTN GUI can be found in Appendix B.

The Python script was made using the TTN library[1]. It was used to retrieve data from the embedded system through the API provided by TTN, and the script wrote the retrieved data to a file on the computer it was running on. A simple script was also made to send clock synchronization data for when the end-device boots up.

When storing the data, the computer's timestamp is also stored along with the message that was received. The log file was made in .csv format, with comma-separated columns. Table 4.3.1 shows how the data is stored.

Table 4.3.1: Table of how the log file sorts incoming data.

Log file columns				
Computer timestamp (Unix timestamp)	Message timestamp (Unix timestamp)	Battery level (%)	Ultrasonic data (Meter)	Pressure data (Bar)

The name of the file was changed manually, such that the data from the three tests that were performed could be easily distinguishable. The tests and the data collected from them is further discussed in Chapter 5.

[1] https://www.thethingsnetwork.org/docs/applications/python/

4.4 Location

The device was placed and tested in three different locations. First, a test in a tank at NTNU, then in Estenstadmarka, and finally in Nidelva. Figure 4.4.1 shows a map of where the three tests were conducted. Figures 4.4.2 and 4.4.3 shows a closeup of the test locations in Estenstadmarka and Nidelva respectively. The tests are further discussed in Chapter 5.

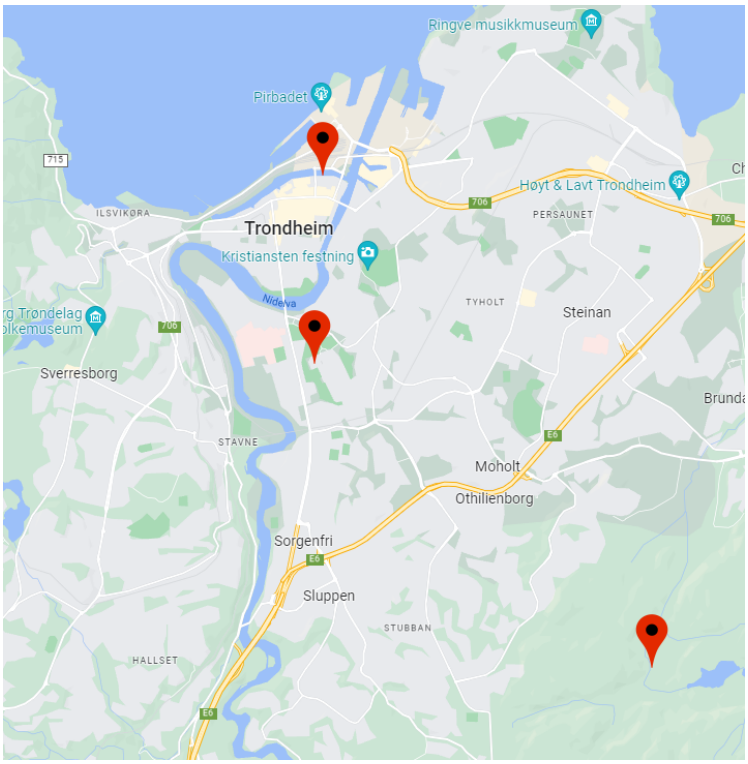


Figure 4.4.1: Map of all test locations.

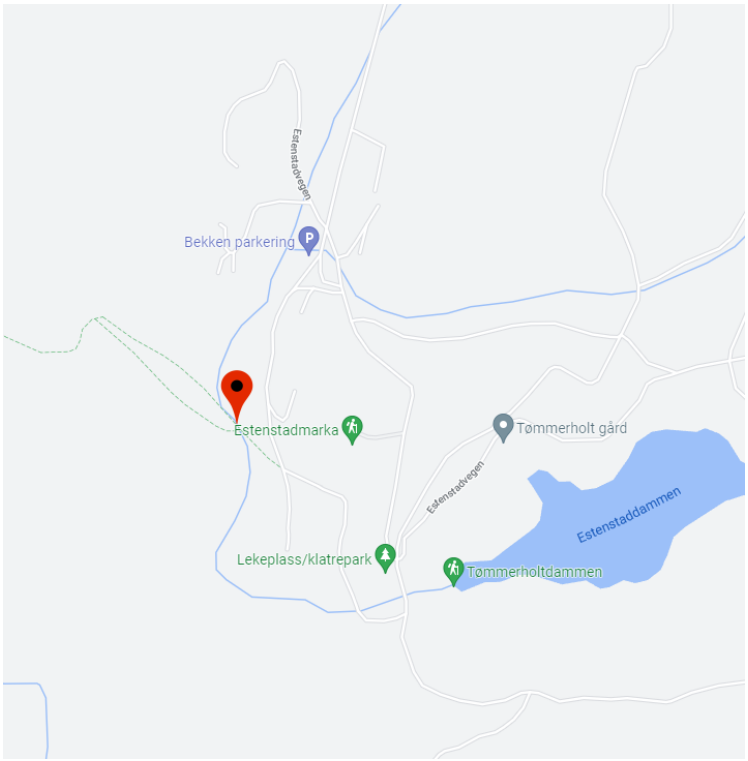


Figure 4.4.2: Map showing the test location in Estenstadmarka.

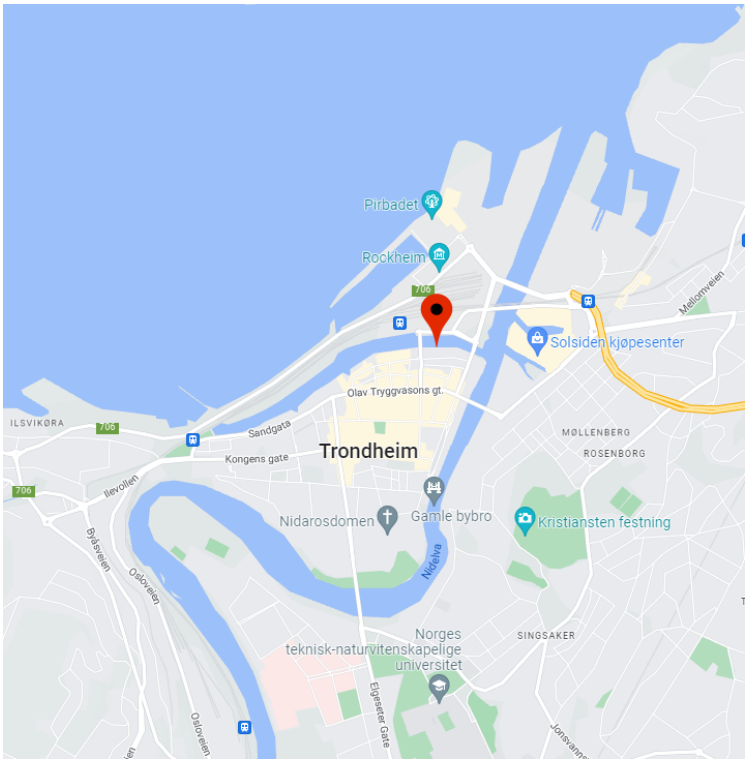


Figure 4.4.3: Map showing the test location in Nidelva.

Chapter 5

Testing and results

This chapter covers the testing and results of the system. The chapter is divided in three parts. First the implemented embedded system is tested against the acceptance criteria from 3.4, Then, the sensor data collected by the system is presented. Finally, a test is performed to measure the current draw of the embedded system.

5.1 Hardware

After the PCB was soldered, it was thoroughly tested using an oscilloscope. The power and data lines were scoped to verify AC4. Simple software was programmed on the ATmega to test the individual function of the PCB. This includes collecting data with the sensors, and sending that data wirelessly using the LoRa module. This data was then compared to the data written on the SD card to verify AC1, AC3, AC6, and AC14. The data was received through the TTN console. The system was also programmed to toggle the LED lights on certain conditions, to verify that the LEDs were working as intended, confirming AC11. AC13, concerning the watchdog timer was tested by making the hardware spin in a while loop, and to blink an LED if the watchdog was triggered.

The battery was supposed to be connected to one of ATmegas ADC inputs, but this was not done as it was not prioritized. This means that the system could not send its status wirelessly, as stated in AC14. However, as the messages are expected to be delivered on a pre-programmed interval, it is possible to use each message as a heartbeat to indicate that the system is still operational.

Messages were sent to the device using the TTN to verify that the system could be configured while active. This was done by using a Python script and the TTN API. AC2, which states that "Data collected by the system can be received and displayed on a computer." fails in the sense that there was no database or GUI was implemented to display the data. The data could be

received however. This proved that the system could change sampling time, as well as received a timestamp to synchronize its clock. This verified AC9 and AC10.

5.2 Sensor data

Three tests were performed where the system was to collect data outdoors, to verify AC5 and AC7. These tests were:

1. A test in an indoor environment
2. A test in a small water stream
3. A test in a river

The test indoors was conducted to verify the repeatability and accuracy of the sensors. It provided proof that the sensors could in fact measure water level and send the data wirelessly for longer periods of time. The outdoor tests were conducted to verify that the system can collect data from various types of waters in outdoor conditions. The first outdoor test was conducted in a small stream in Estenstadmarka in Trondheim, and the second was conducted in Nidelva in Trondheim. No averaging was done on the collected data, the sensors only sampled once and then turned off until the next cycle.

5.2.1 Indoor test

Test procedure

The indoor test was performed in a water tank at the university. The ultrasonic sensor was mounted approximately 51cm above the water surface, and the pressure sensor was fastened approximately 30cm below the water surface. The test was conducted for 24 hours and 9 minutes, which is roughly 86940 seconds. The sensors were sampled every 10 seconds. It was done to test the drift and accuracy of both sensors. This was done to provide more information on the reliability of the data collected when measuring in nature.

Results

Figures 5.2.1 and 5.2.2 show the data collected from the ultrasonic and pressure sensor respectively.

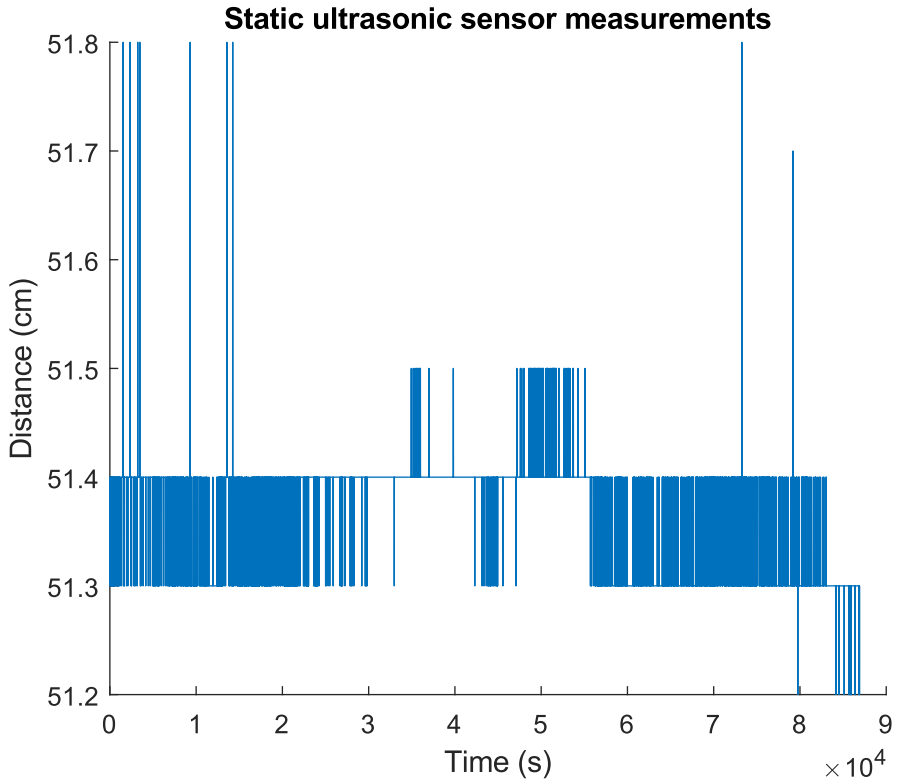


Figure 5.2.1: Ultrasonic sensor data from logging in an indoor environment.

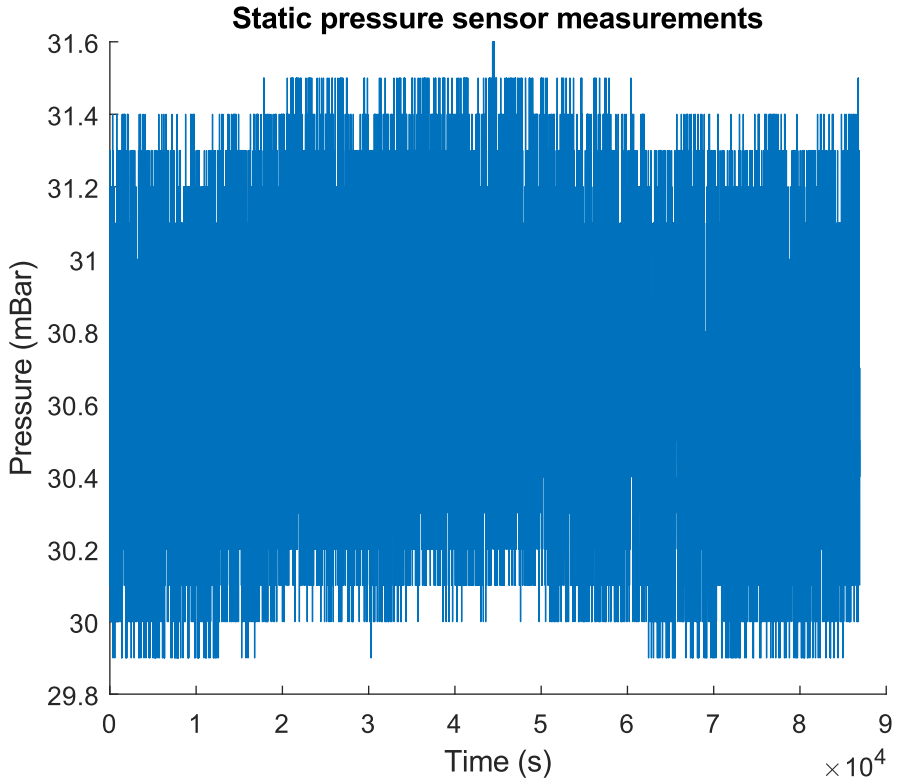


Figure 5.2.2: Pressure sensor data from logging in an indoor environment.

5.2.2 Stream test

Test procedure

To verify that the system can collect data in an unpredictable environment, it was placed alongside a small stream in Estenstadmarka. The test was conducted between 10:54 and 15:27 on the 14. of May. It lasted approximately 4 hours and 33 minutes, which is roughly 16380 seconds. The water level was sampled every 20 seconds. The ultrasonic sensor was mounted approximately 114cm above the water surface at the start of the test, and the pressure sensor was placed approximately 31cm below the water surface at the start of the test. Figure 5.2.3 shows the system as it was set up in Estenstadmarka.

Figure 5.2.3: Image of the system deployed in a stream in Estenstadmarka.

Results

Figures 5.2.4 and 5.2.6 show the data collected from the ultrasonic and pressure sensor respectively when logging the water level in a stream in Estenstadmarka. Note that the ultrasonic sensor data is flipped, as the sensor measures distance from the water surface, and not the water level itself. Assuming that the water level was 31cm from the stream bed at the start of the test, Figure 5.2.5 shows the water level data collected by the ultrasonic sensor.

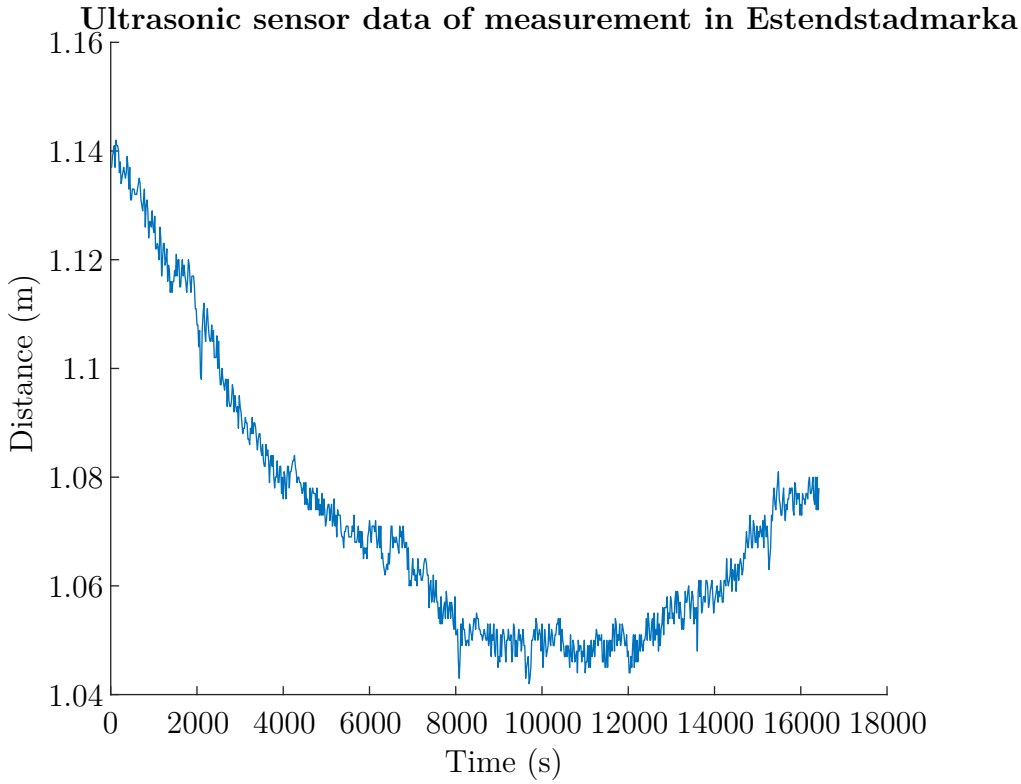


Figure 5.2.4: Ultrasonic sensor data from logging in a stream in Estenstadmarka.

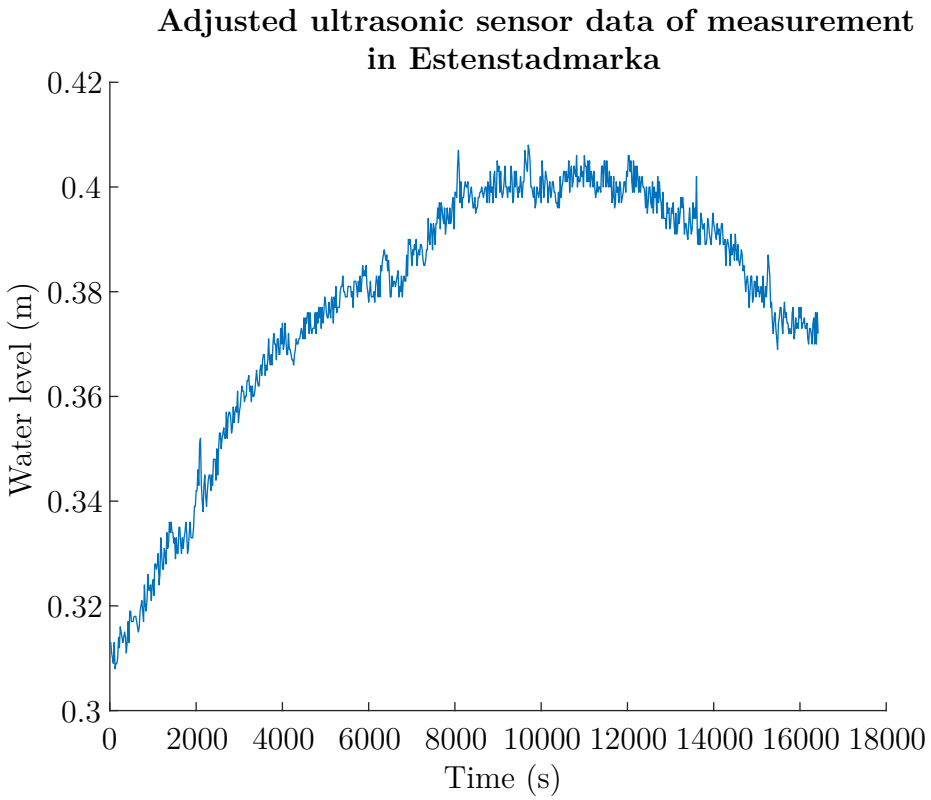


Figure 5.2.5: Adjusted ultrasonic sensor data from logging in a stream in Estenstadmarka.

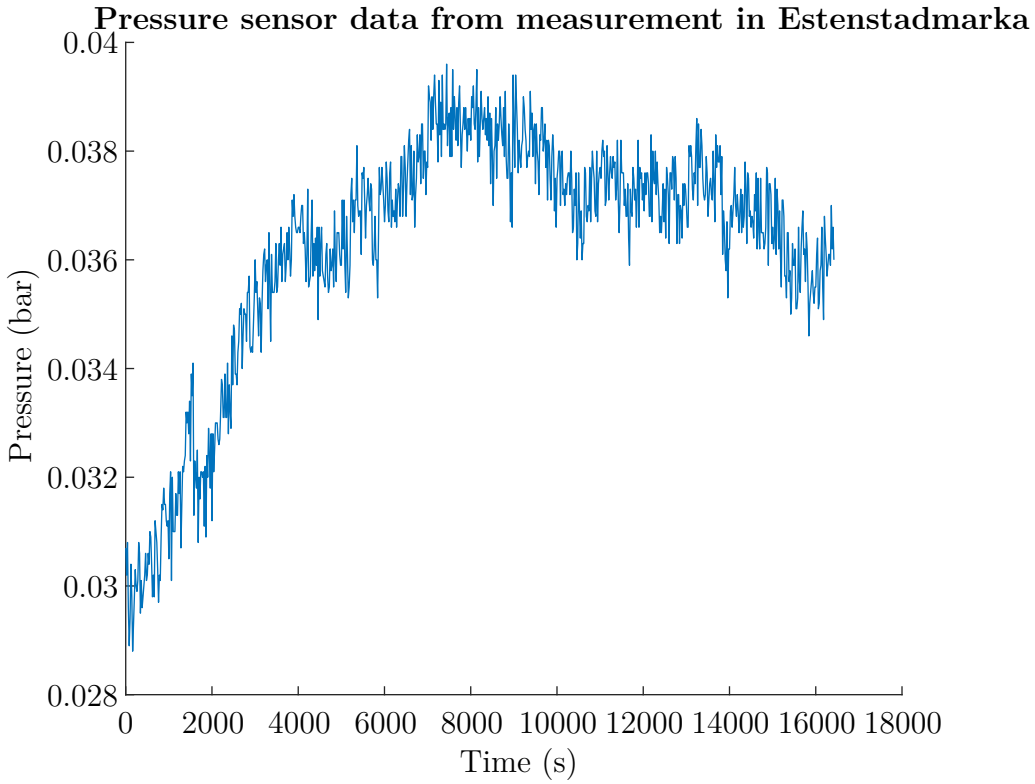


Figure 5.2.6: Pressure sensor data from logging in a stream in Estenstadmarka.

5.2.3 River test

Test procedure

A final test was conducted in Nidelva, to measure water levels for a week. The test lasted from 18:03 on the 25. of May 2022 until 14:23 on the 1. of June 2022. This corresponds to approximately 591600 seconds. The water level was sampled every 100 seconds. The ultrasonic sensor was mounted approximately 3.3 meters above the river bed, with the pressure sensor on the river bed at the same location. Figure 5.2.7 shows the location of where the system was placed. The image was taken at low tide.

Figure 5.2.7: Image of the location measured in Nidelva.

Note that since the system needs to be placed close to land, the sensors can not reach far enough out to measure the lowest point of the tide. Figure 5.2.8 shows how the sensors cannot detect water levels below a certain threshold. The principle applies both to the ultrasonic and pressure sensor, as they cannot be placed far out on the river bed.

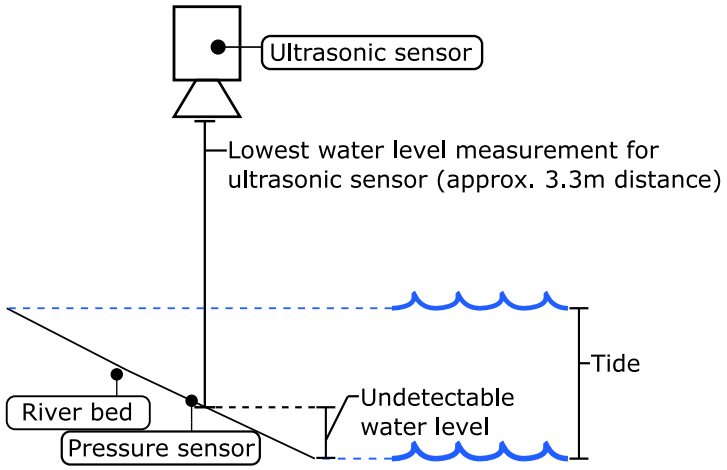


Figure 5.2.8: Illustration of how the sensors can only detect water level above a certain level.

Results

Figures 5.2.9 and 5.2.11 show the data collected from the ultrasonic and pressure sensor respectively when logging the water level in Nidelva. Note that the ultrasonic sensor data is flipped, as the sensor measures distance from the water surface, and not the water level itself. Correcting for this, Figure 5.2.10 shows the water level data collected by the ultrasonic sensor.

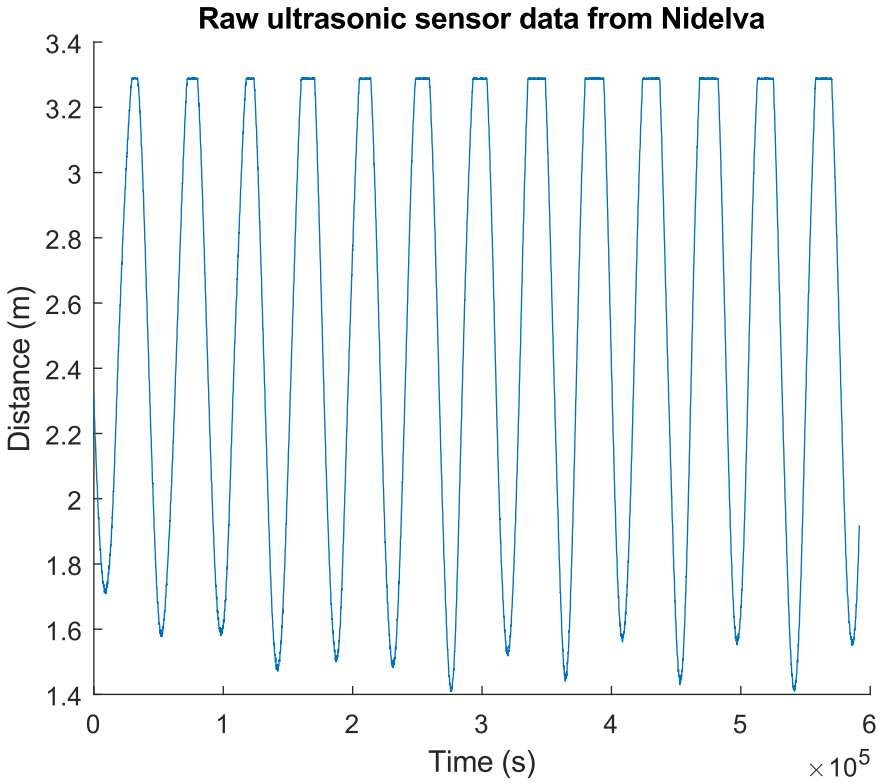


Figure 5.2.9: Ultrasonic sensor data from logging in Nidelva.

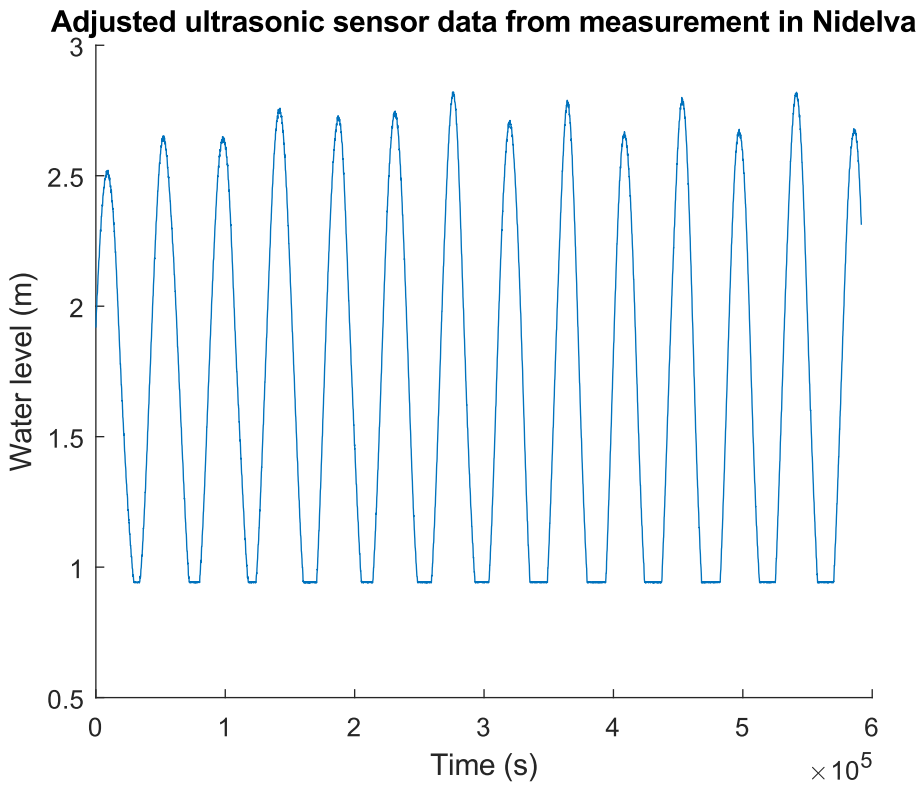


Figure 5.2.10: Adjusted ultrasonic sensor data from logging in Nidelva.

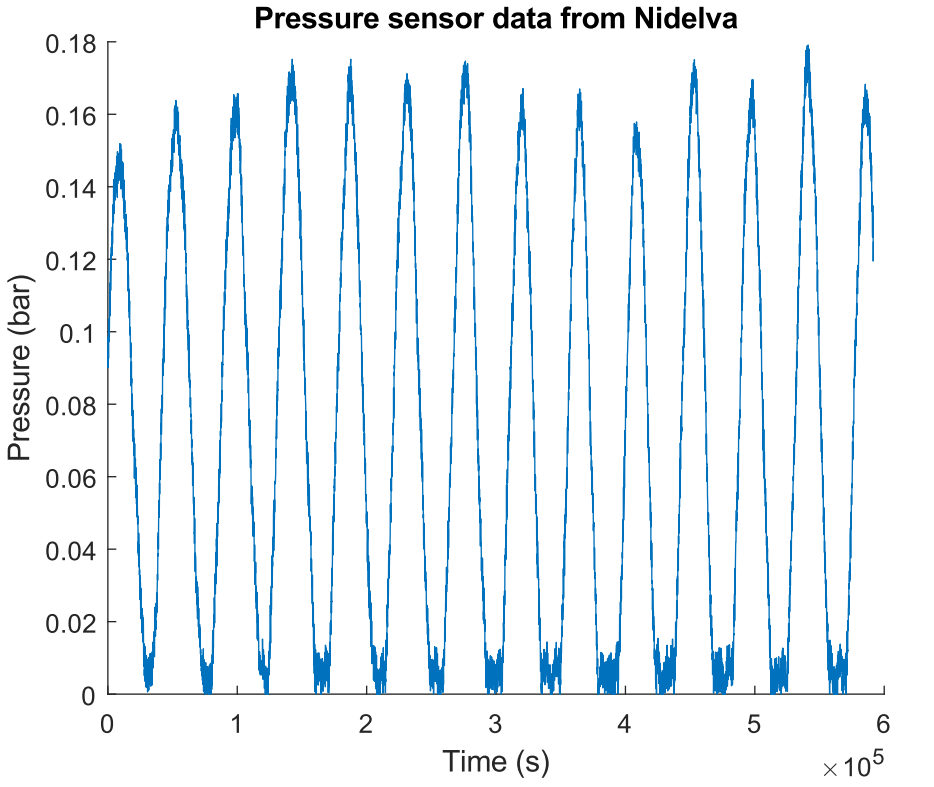


Figure 5.2.11: Pressure sensor data from logging in Nidelva.

5.3 Power consumption

To verify AC8 and AC12, several tests were conducted to measure the power consumption of the system. By knowing how much energy the system consumes for each action, and for how long each action is performed, one can calculate the lifespan of the system.

During testing it was found that acknowledging messages takes a long time, as well as consumes a lot of unnecessary energy. It was chosen to discard the acknowledgement of messages and simply send 10 messages without waiting for acknowledgement.

5.3.1 Test procedure

To test the current draw of the system, Nordic Semiconductor's Power Profiler Kit II was used. It has µA accuracy, and can provide its own voltage supply. The kit can sample up to 100 000 times/second. It has its own software and is interfaced to a computer using an USB cable, which can be used to extract the measured current draw. Figure 5.3.1 shows an image of the Power Profiler Kit.

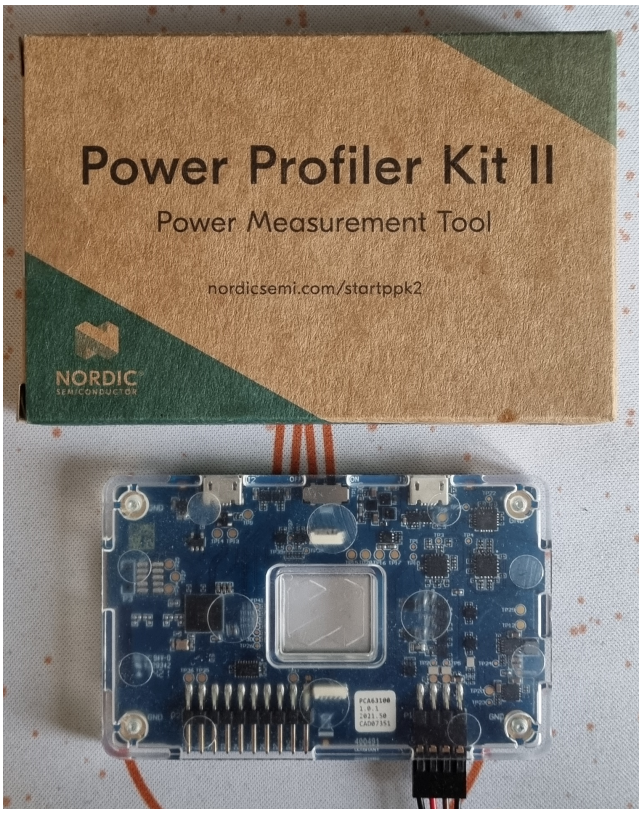


Figure 5.3.1: Image showing Nordic's Power Profiler Kit II.

It was connected to the PCB and the kit's own voltage supply was used and set to 3.3V through Nordic's provided software. An image of the system being

tested is shown in Figure 5.3.2.

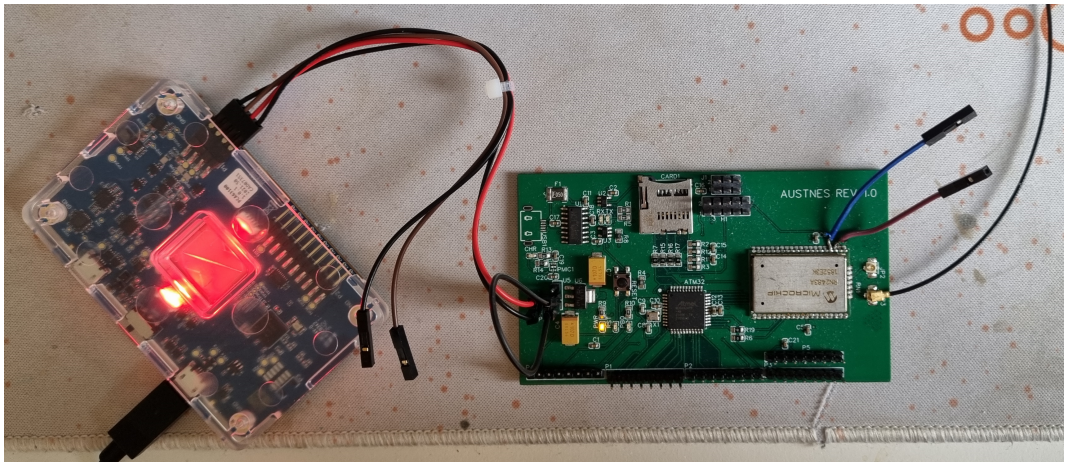


Figure 5.3.2: Image showing the system being tested while transmitting data on the RN2483.

5.3.2 Results

Four different functionalities were measured for their current draw. This includes the RN2483 while it was sending, the sensors when they sampled data, sleep mode on both the ATmega and the RN2483, and when the system was running through its idle state. The last includes writing to the SD card.

Figure 5.3.3 shows the current draw while transmitting data on the RN2483, and a closeup of one message being sent is shown in Figure 5.3.4. In Figure 5.3.4, the RN2483 is woken up from sleep at $2.7 * 10^4$ms, and the transmission starts at $2.8 * 10^4$ms. A single send draws 12.65mA on average and lasts approximately 2.5 seconds. The spreading factor is 7, with a bandwidth of 125kHz, and with a physical bit rate of 5470 bit/s.

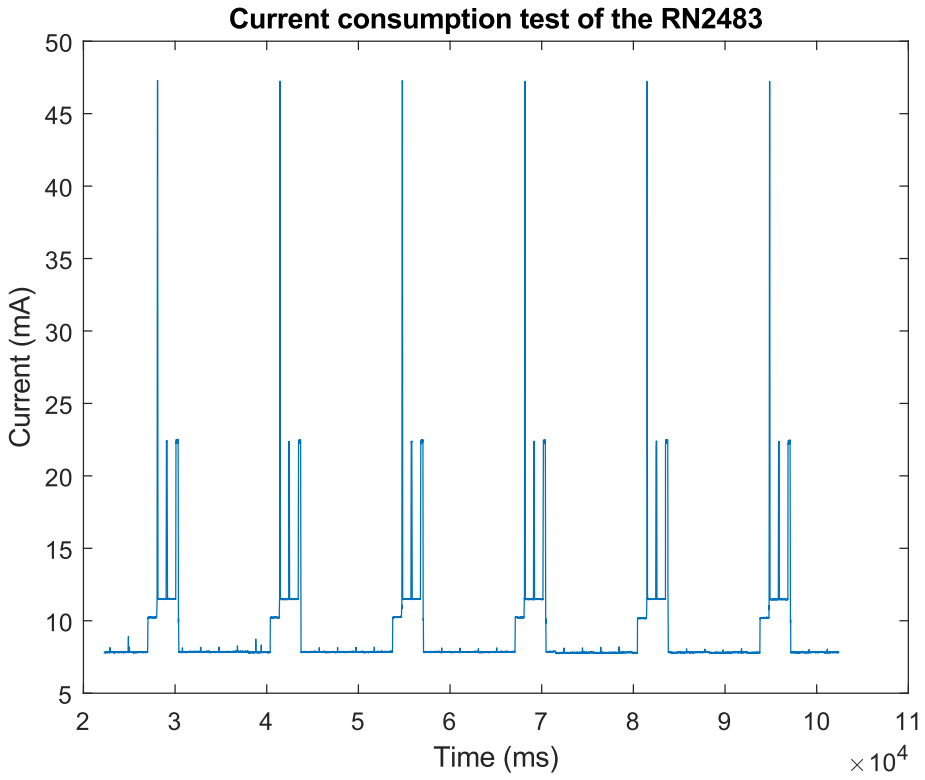


Figure 5.3.3: RN2483 power consumption during sending.

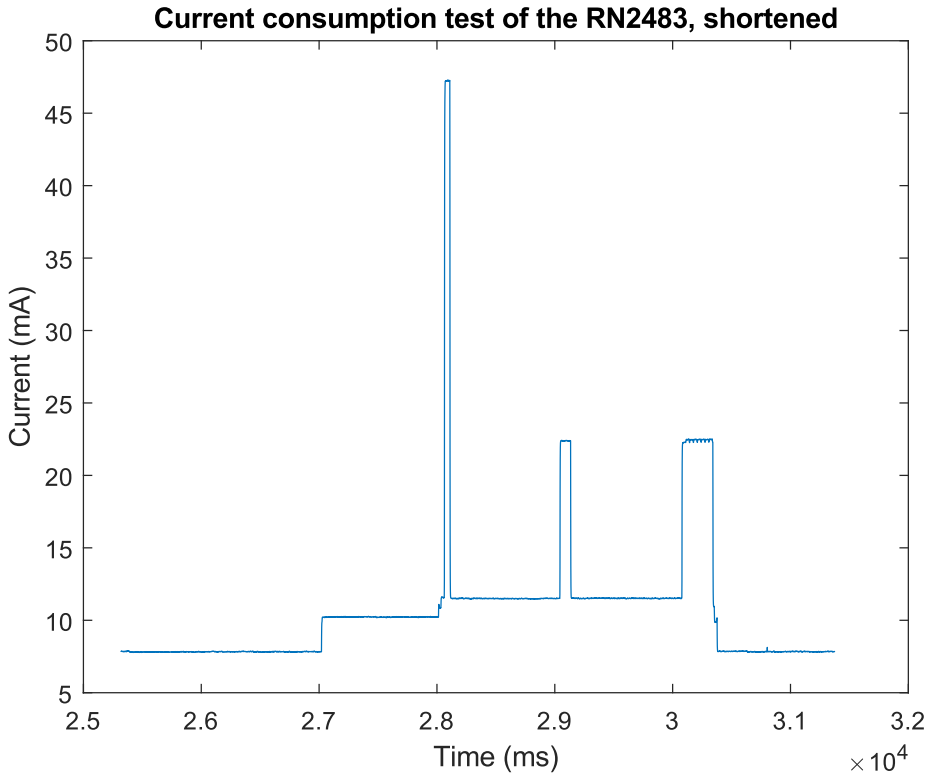


Figure 5.3.4: Power consumption of the RN2483 during a single send.

Figures 5.3.5 shows the current draw for the system while the ultrasonic sensor is sampling and being toggled off by the transistor as described in Chapter 4. 5.3.6 shows the current draw for a single sample being taken by the ultrasonic sensor. The current draw is 18.6mA on average, and drops to approximately 8mA when turned off. The pressure sensor was measured to draw a constant 10.2mA when on. Table 5.3.1 summarizes the average current draw for the different operations performed by the system.

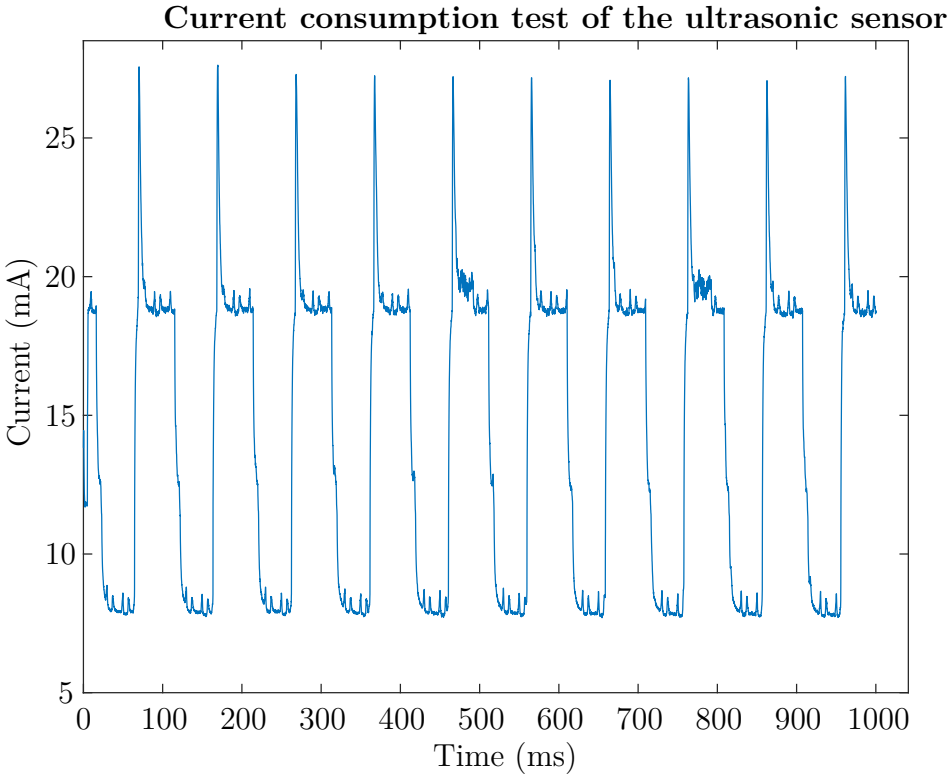


Figure 5.3.5: Power consumption of the ultrasonic sensor.

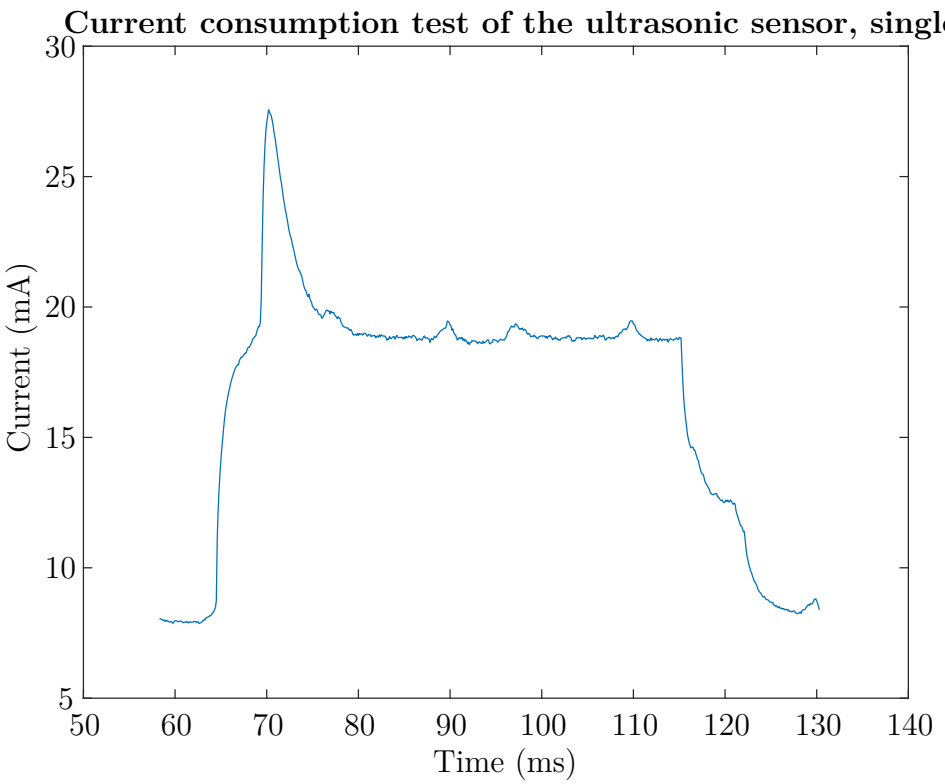


Figure 5.3.6: Power consumption of the ultrasonic sensor during a single sample.

Table 5.3.1: Summary of the system's average current draw for different operations.

Operation	**Current draw**	**Time 1 cycle**
Message transmission	12.65mA	2.5s
Sensor sample	18.6mA	120ms
Other operations	8mA	100ms
Sleep	743uA	(Cycle time - 2.72s)

Chapter 6

Discussion

This chapter discusses the obtained results from Chapter 5. First, a discussion on how well the system fulfilled the acceptance criteria as stated in 3.4, as well as a discussion on design flaws and possible improvements which can be done. Then the sensor data is discussed and analyzed, then a brief discussion on the system's power consumption.

6.1 System

The system has proven to be able to measure water levels in remote locations, and has potential to be useful for measuring sea level, rivers, drainage system, road ditches, and more. The cost of the system came at less than 1000 NOK per device assuming only one sensor is used. This fulfills AC15. It would likely be much cheaper to produce for large scale water monitoring, as the unit price for each component is much higher when only producing a single device. It fulfills most acceptance criteria except for AC2, as a database and GUI is left to be desired. Handling files manually is much harder than if they were stored in a database, especially if multiple devices are deployed and the devices log data for longer periods of time. A lack of a user interface also makes checking the data harder, as it needs to be imported into Matlab or a Python script for further processing. The end goal for the system would be that it only needs to be configured once and deployed, and then for the logs to be automatically created for the user to review or otherwise use.

There are also other aspects of the system which could be improved upon. The tests that were performed in this thesis only took a single sample for each data point. By sampling a few more times and averaging the results, one could possibly have removed outliers gotten a more reliable result. The sampling period was relatively small, up to 100 seconds. For testing in Nidelva this should be enough to provide a clear picture of how the water level changes, as no sudden changes are expected to take place within a 100 second span. While measuring road ditches, streams connected to dams, etc., the sampling

period should be increased to detect and warn about sudden changes faster. Shorter periods in between sampling increases the amount of messages sent however. This is by far the biggest factor in energy consumption as seen in Chapter 5. Decreasing the period would also mean that each data point is less important. More data means that each single data point is of less importance, and averaging over samples would not be as necessary to combat sensor inaccuracy as this could be done in the back end.

Another way to save energy is to implement the system with more efficient hardware. The implementation of the system is this thesis is limited both by previous knowledge and the currently high lead times on integrated circuits. The ATmega324PB, for example, was chosen as it was deemed good enough for this thesis, and also due to familiarity. There are newer microcontrollers on the market today which are both easier to work with and more energy efficient, such as the EFM32 produced by Silicon Labs. Another improvement which could have been made is to put the unnecessary circuitry on its own power line, for instance the LEDs. There is currently no way to turn off the LEDs to reduce current draw, other than physically removing them from the circuit. Including an optional jumper which could supply voltage to a seperate power line could make the strictly unnecessary circuitry optional such that it is only active while debugging. This would save on energy consumption as the extra functionality could be turned completely off.

Some flaws in the design were also found only after ordering the board. One of them being that thicker traces should have been used for the power lines, for less resistance and heat buildup. The screw terminal also had the wrong footprint to that which was ordered, which led to header pins being used instead. This makes the system as a product significantly worse, as the power connection is much easier to disconnect. The USB-connectors which were ordered also had the wrong footprint, and is due to the vendor shipping the wrong item. As there was not enough time to order new ones, the USB-connection was not implemented. This also meant that the recharging circuitry did not work, as it relied on the 5V supplied through USB. This however, is not strictly needed, and is the reason it was discarded.

A better real-time clock and clock synchronization method could also have been implemented. The implemented RTC only has 1 second resolution, and the clock synchronization does not account for the skew that appears in sending data to and from the device. The watchdog timer could also have been more thoroughly tested, but this was not prioritized.

As the ADC was not connected to the battery, it was not possible to receive battery status. This was not prioritized, as it was only a minor feature. If one were to implement this, it would likely suffice to use a voltage divider to lower the voltage such that it is tolerated by the ATmega's ADC. The battery itself also an area of improvement; A regular battery could be used instead of a power bank.

Table 6.1.1 shows a table that summarizes the acceptance criteria again for

the sake of readability, and whether or not the system passed each criteria.

Table 6.1.1: Table showing the acceptance criteria and whether or not the system passed.

Label	Description	Passed
AC1	The system can send sensor data wirelessly.	Yes
AC2	Data collected by the system can be received and displayed on a computer.	No
AC3	The system can store data on a SD card.	Yes
AC4	The system supports most sensor interfaces, such as analog, I2C, and UART.	Yes
AC5	The system can detect water level with centimeter accuracy.	Yes
AC6	The system can be deployed anywhere with LoRa coverage.	Yes
AC7	The system can survive harsh weather conditions, including heavy rain and wind.	Yes
AC8	The system can operate for months without maintenance.	Yes
AC9	The system can be configured wirelessly.	Yes
AC10	The system has a local clock which can be synchronized.	Yes
AC11	The system has LEDs which indicate its status.	Yes
AC12	The system utilizes sleep modes to save energy.	Yes
AC13	The system can reset on unexpected errors.	Yes
AC14	The system can send status data wirelessly.	Yes
AC15	The system should cost less than 1000 NOK to produce.	Yes

6.2 Sensor data

The data analysis assumes that the pressure is approximately linearly proportional to the water depth such that 1Bar $\implies$ 10m, as no measurements of the flow of the water was done.

6.2.1 Estenstadmarka

Figure 6.2.1 shows the data collected in Estenstadmarka. Here the two data sets are overlapping such that the data from the ultrasonic sensor and the pressure sensor can be compared. The results from the specialization project [2] support that the chosen ultrasonic sensor is very accurate, and should be used as a baseline for comparing the other sensor. The noise in the ultrasonic measurements might be due to more more bubbles, ripples, and floating debris in the stream. The most notable features from 6.2.1 are that the pressure sensor is more noisy, and that it deviates from the ultrasonic sensor, especially as the water level increased. The fact that the pressure sensor is more noisy than the ultrasonic sensor is also reflected in the test done indoors, as shown in Figures 5.2.1 and 5.2.1. The flow of water in the stream can also contribute to both the noise and the deviation from the measurements. As the water moves more rapidly, the pressure drops, and this can significantly affect the pressure measurements [39]. During the first hours, the pressure sensor is approximately 2cm below what is measured by the ultrasonic sensor, and then deviates by up to 4cm when the water level is at its peak.

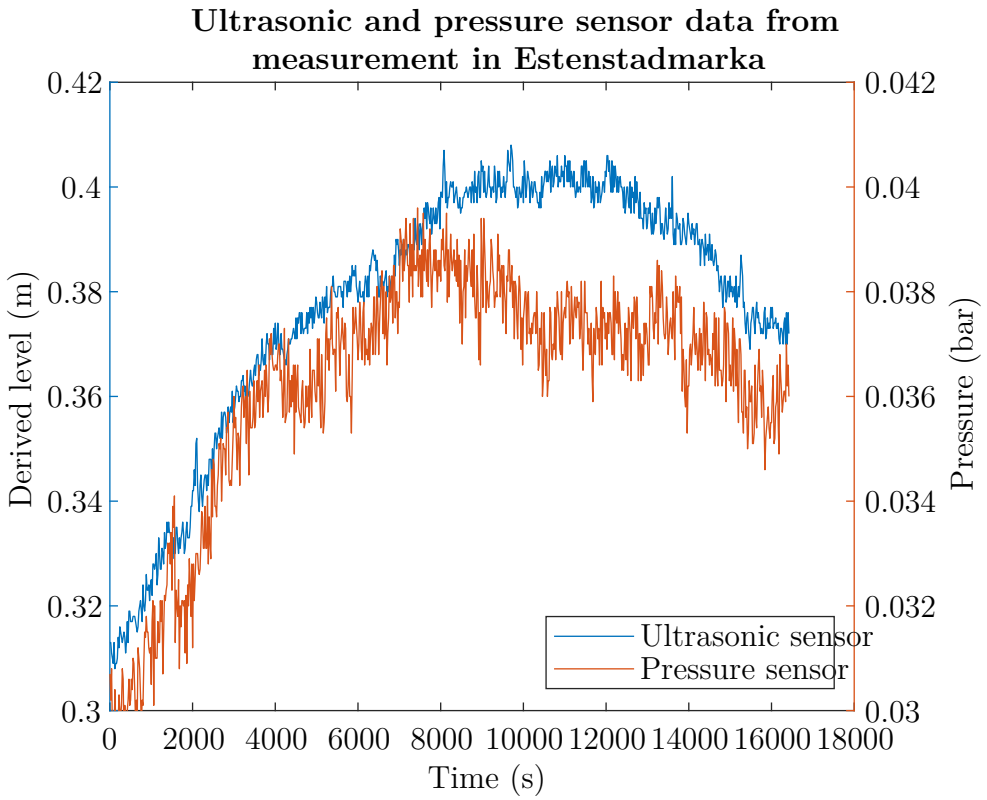


Figure 6.2.1: Ultrasonic and pressure sensor data from logging in Estenstadmarka.

6.2.2 Nidelva

One of the reasons Nidelva was chosen was due to the fact that Kartverket provides tidal data on their website at https://www.kartverket.no/. This provides a useful baseline for comparing both the ultrasonic sensor and the pressure sensor to more reliable data. Figure 6.2.2 shows tidal data provided by Kartverket between 00:00 on 25. of May 2022 until 23:59 on the 1. of June 2022.

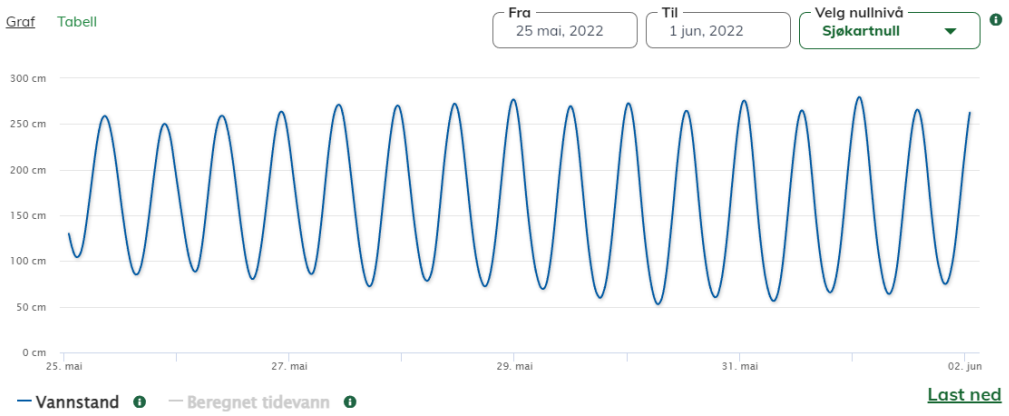


Figure 6.2.2: Measured tide by Kartverket [52].

The tidal data can be downloaded directly from their website, but it only provides a sample for every hour. To be able to better visualize the data, some work had to be done. The 193 samples provided from Kartverket in this one week period was interpolated in Matlab using spline interpolation, and the result is shown in 6.2.3. The Figure shows the downloaded data as blue circles, and the interpolated data in orange. The interpolation was only done on the data between 18:03 on the 25. of May and 14:23 on the 1. of June, as this was when the system was active.

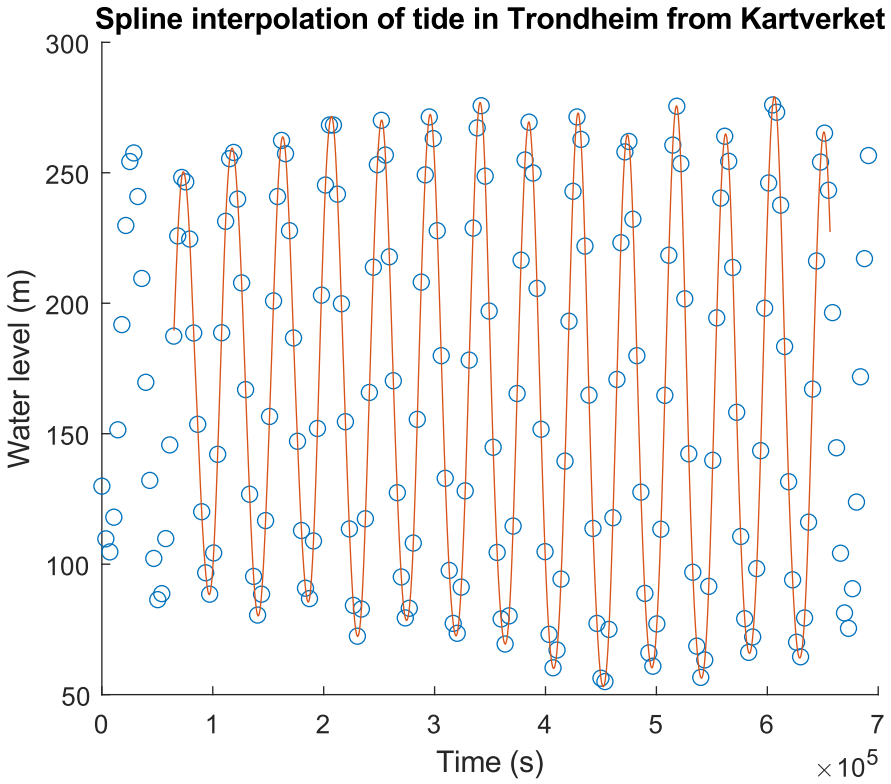


Figure 6.2.3: Spline interpolation of tide data from Kartverket.

Figure 6.2.4 shows the collected ultrasonic sensor data compared to the interpolated data from Kartverket. 6.2.5 shows the first two tides of that data. Figure 6.2.6 shows only the first high tide in the data set, and shows that the ultrasonic sensor closely follows the data from Kartverket, only deviating with approximately 2cm. The second high tide is shown in 6.2.7 and again shows that the ultrasonic sensor tracks the water level in Nidelva extremely well, with only a deviation of approximately 4cm at worst.

However, as explained with Figure 5.2.8, the low tides are not detectable, as the ultrasonic sensor points at the river bed below a certain water level. This explains the flat regions when the water level is less than approximately 0.95m. This could have been fixed by pointing the ultrasonic sensor at a slight angle, such that it measures in the middle section of the river, but it was chosen to set it pointing vertically, such that the data did not skew.

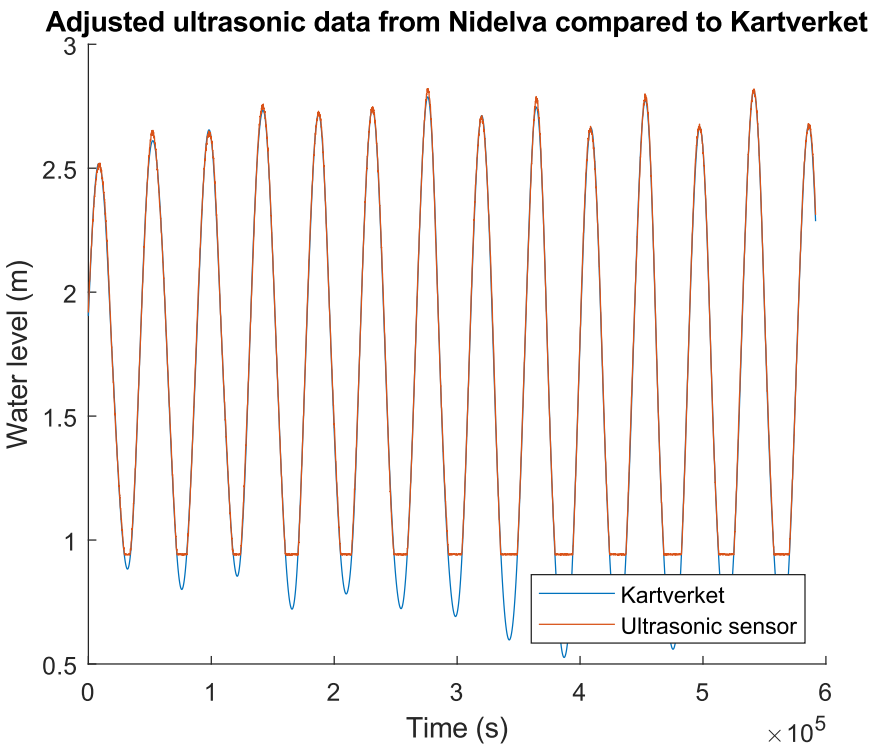


Figure 6.2.4: Ultrasonic sensor data from logging in Nidelva compared to data from Kartverket.

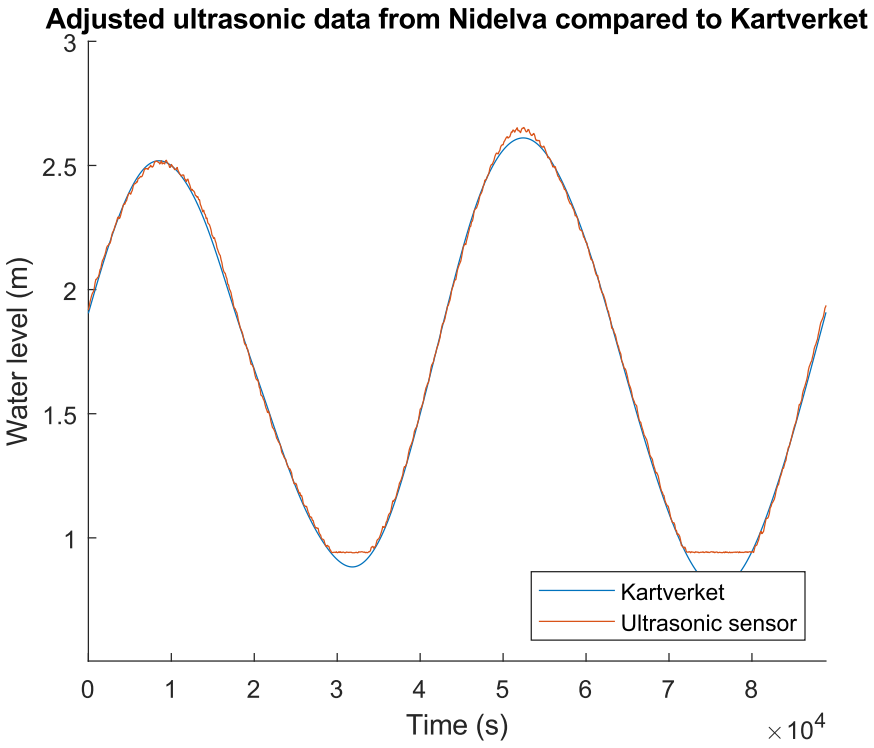


Figure 6.2.5: First two tides measured with ultrasonic sensor in Nidelva compared to Kartverket.

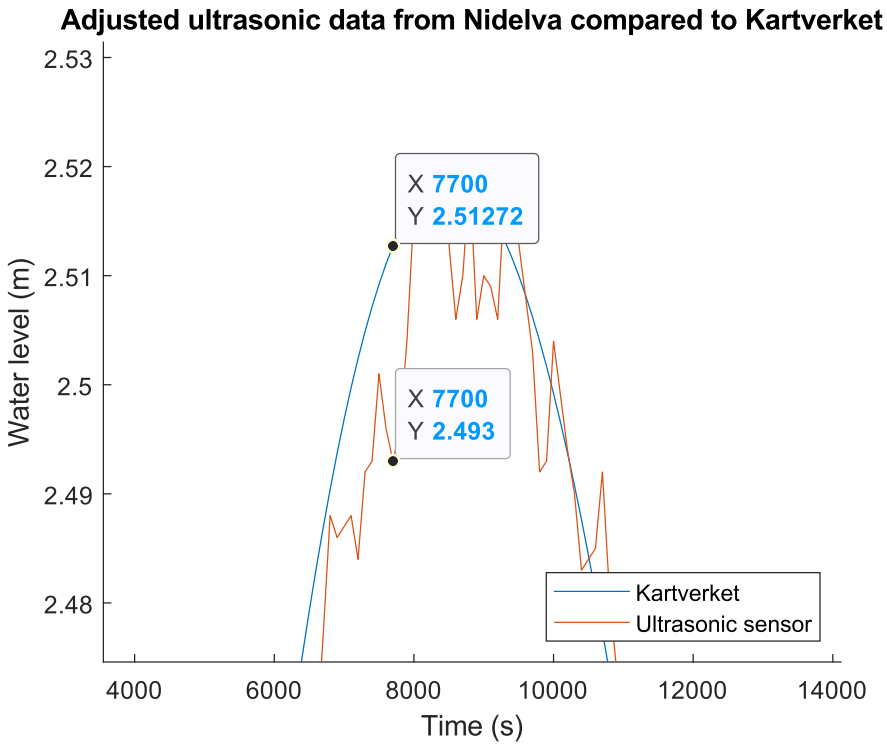


Figure 6.2.6: First high tide measured with ultrasonic sensor in Nidelva compared to Kartverket.

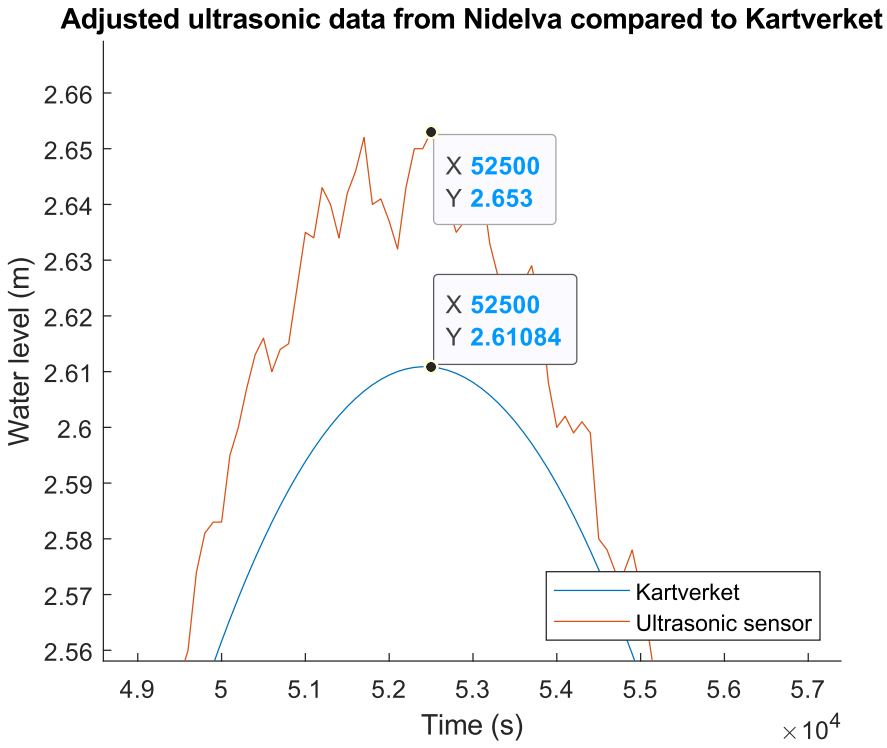


Figure 6.2.7: Second high tide measured with ultrasonic sensor in Nidelva compared to Kartverket.

Figure 6.2.8 shows the measured pressure in Nidelva compared to the interpolated data from Kartverket. Figure 6.2.9 shows the two first tides from the same data set. Again, the pressure data is more noisy, as expected from the results from both the indoor test and the test in Estendstadmarka. The pressure sensor was placed at the river bed directly below the ultrasonic pressure, so the same flat line is to expected during low tide. The pressure sensor is however very noisy during this period, as seen in 6.2.9 at around $3 * 10^4$ seconds after starting the test.

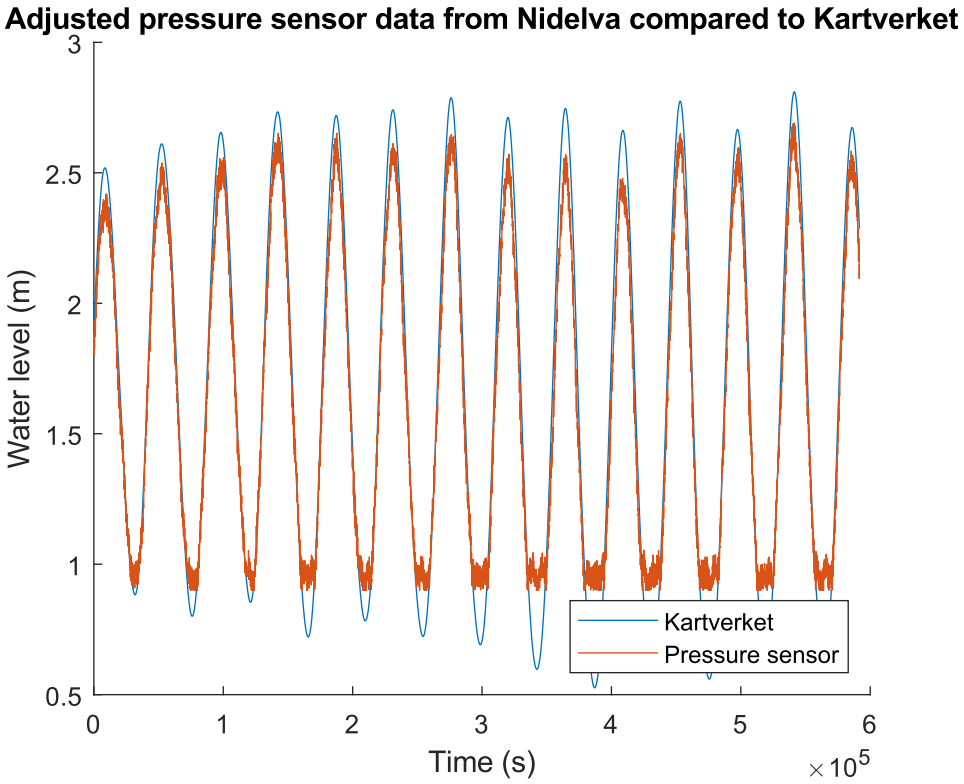


Figure 6.2.8: Pressure sensor data from logging in Nidelva compared to data from Kartverket.

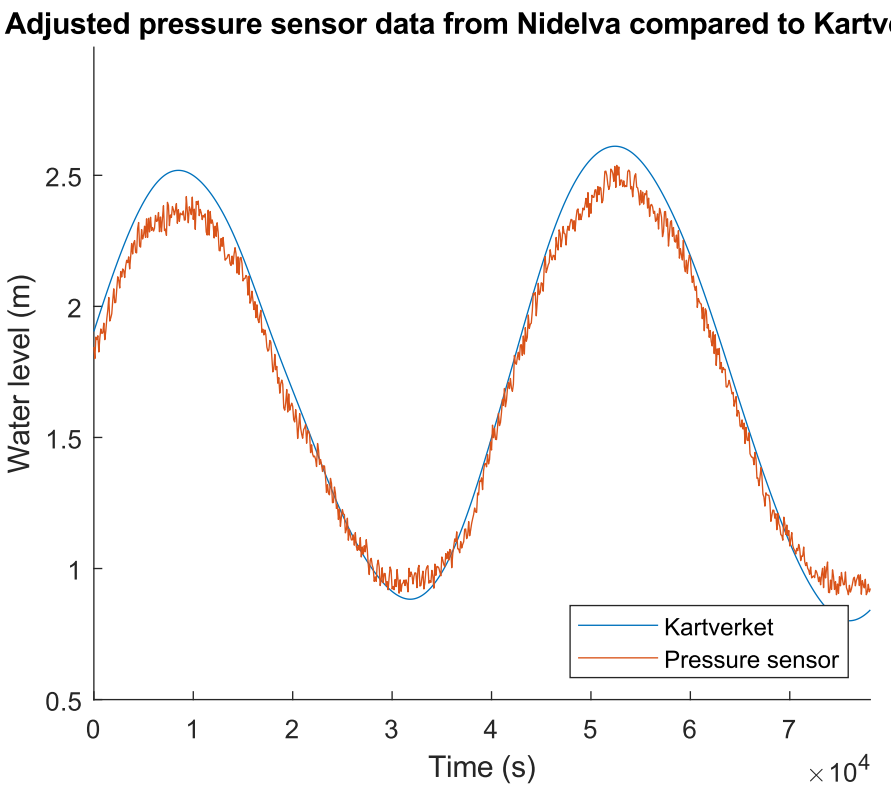


Figure 6.2.9: First two tides measured with pressure sensor in Nidelva compared to Kartverket.

Finally, the two data sets are compared in Figure 6.2.10. From previous tests, and from the specialization project [2], it is likely that the ultrasonic sensor gives a more accurate representation of the water level that the pressure sensor. The water level from the pressure sensor is significantly lower that what is measured by the ultrasonic sensor. This could be due to the drop in pressure from water flow, or maybe a poorly calibrated sensor. The latter is unlikely, as the static test showed that the sensor neither drifts, and can represent the water level if the water is still.

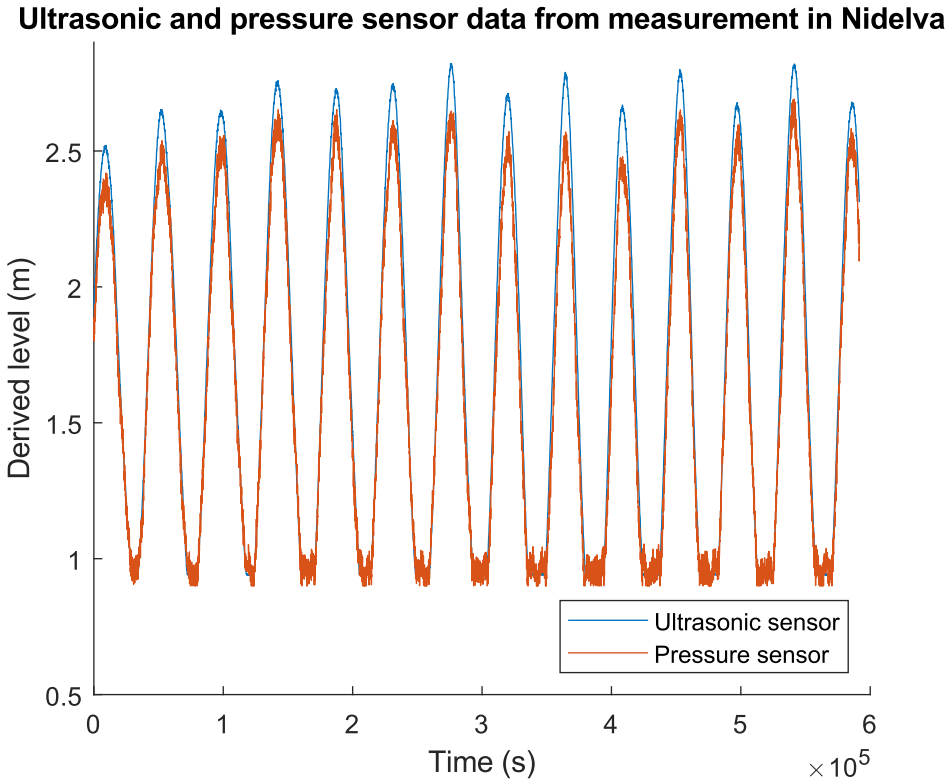


Figure 6.2.10: Ultrasonic and pressure sensor data from logging in Nidelva compared to eachother.

6.3 Energy consumption

By looking at current draw data collected with the power profiler kit in Table 5.3.1, the average current draw of the system can be calculated with the following formula:

$$\frac{12.65\text{mA} * 2.5\text{s} + 18.6\text{mA} * 0.12\text{s} + 8\text{mA} * 0.1\text{s} + 0.743\text{mA} * (\text{Cycle time} - 2.72\text{s})}{\text{Cycle time}} \tag{6.1}$$

The cycle time is the time between each sensor sample and transmission. Assuming only one message is sent every time, by choosing a cycle time of 10 seconds gives an average current draw of 4.00mA. A cycle time of 100 seconds gives an average current draw of 1.07mA.

Batteries of 10Ah on the market today are both cheap and portable enough to be used alongside this system. This would offer a life expectancy of 104 days with a 4.00mA current draw, and 389 days for an average current draw of 1.07mA. This should be more than enough to satisfy AC8, which states that the system should be able to operate for months without maintenance.

However, this life expectancy is a best-case scenario. Several factors could affect the longevity of the system. If data from the sensors should be averaged before sending, the sensors need to stay active for longer, and if more that one message is sent, significantly more current is drawn. The same applies to if the messages are acknowledged. In the case for this system, it was deemed unnecessary to acknowledge each message, as a single data point is not worth the extra cost of more energy consumption. Very little package loss was experienced during testing, but in case the system is located somewhere with poor LoRa coverage, it is expected to lose a significant amount of messages. Therefore either more messages need to be sent, or acknowledged. In case 20 messages need to be sent each time with a sample interval of 100 seconds, the average current draw would be approximately 6.73mA. With a 10Ah battery, this implies a life expectancy of approximately 62 days at best, and would not satisfy AC8 to the same degree. Sending more messages also limits the sample interval. With the current system, if 20 messages are sent, the system can only sample every 50 seconds at best.

The battery size could be increased, of course, Doubling the battery size would in theory also double the longevity, but comes at the cost of both price and portability. The system could also utilize solar energy to recharge itself during sunny days, but this would not alleviate the problem if the system is to be deployed during rainy periods, or winter months.

Chapter 7

Conclusion

A system for the purpose of measuring water levels in nature has been designed and implemented. Some modifications were done to the original design during implementation, and some features were not fully implemented. Mostly, a thorough implementation of the serverside software is lacking. The database for storing collected data, and the GUI for plotting said data were not implemented due to time constraints for this thesis.

The embedded system was placed in both Estenstadmarka and by Nidelva to log water levels. The longest test was conducted by Nidelva, and lasted for a week. Properties such as data accuracy, and device life expectancy were investigated. The system can detect water levels in nature with high accuracy, assuming an ultrasonic sensor is used, and can operate for long periods of time without maintenance as planned in the acceptance criteria. The collected data has shown that it is possible to deploy the system in unstable environments, such as streams, or larger bodies of water such as rivers.

The system was made by designing and ordering a custom PCB, and ordering sensors which can be used to measure water levels. Most acceptance criteria were fulfilled, but there are still some improvements which can be done however.

Chapter 8

Future work

The following is a list of suggestions and improvements which can be implemented based on the discussion in Chapter 6.

More energy efficient IC and sensors — It is worth looking into more energy efficient hardware. More specifically, replacing the microcontroller, LoRa module, and sensors with an equivalent that draws less current. As an added benefit to this, newer ICs are often easier to work with and have less clunky toolchains.

Redesign PCB — Parts of the PCB design draw unnecessary current, such as the LEDs, and the USB-serial interface. Both were made to make debugging easier, but they are not needed for the system to operate normally. These could be removed to make the system more energy efficient. A more general redesign could also be done to

New battery — A power bank is used currently. This could be swapped out with a regular battery. It should also be connected to the microcontroller's ADC such that the battery level can be read and broadcast.

Concider solar energy — Solar energy can be used in conjunction with a battery to further increase the life expectancy of the embedded system.

An integrated cloud solution — As the database and user interface is lacking in this thesis, it is suggested to create a better implementation for storing the collected data.

Improve the real-time clock — Currently, the real-time clock only has a resolution of 1 second. This needs be further improved if data with higher time resolution is needed.

Deploy several devices — It would be interesting to see if the system is useful when deployed on a larger scale.

Bibliography

[1] K. S. Mehta, P. T. Maru and N. P. Shaha, 'Survey of the systems for water level detection', *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, Dec. 2020.

[2] M. Austnes, 'Sensors for measuring water level in nature', p. 37, Dec. 2021.

[3] Y. Pachipala, C. Nagaraju, A. Raju, A. Yeswanth, K. Karthik and P. Surendra, 'Iot based water level meter', Dec. 2018, pp. 448–456. DOI: 10.1109/ICSSIT.2018.8748838.

[4] A. M. Manoharan and V. Rathinasabapathy, 'Smart water quality monitoring and metering using lora for smart villages', Aug. 2018, pp. 57–61. DOI: 10.1109/ICSGSC.2018.8541336.

[5] R. Jisha, G. Vignesh and D. Deekshit, 'Iot based water level monitoring and implementation on both agriculture and domestic areas', Jul. 2019, pp. 1119–1123. DOI: 10.1109/ICICICT46008.2019.8993272.

[6] R. M. Shrenika, S. S. Chikmath, A. V. R. Kumar, Y. V. Divyashree and R. K. Swamy, 'Non-contact water level monitoring system implemented using labview and arduino', *2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)*, pp. 306–309, 2017.

[7] A. Tamang and S. Shukla, 'Water demand prediction using support vector machine regression', Mar. 2019, pp. 1–5. DOI: 10.1109/IconDSC.2019.8816969.

[8] A. Prafanto and E. Budiman, 'A water level detection: Iot platform based on wireless sensor network', Nov. 2018, pp. 46–49. DOI: 10.1109/EIConCIT.2018.8878559.

[9] S. K. U. Ajit Kumar Sahoo, 'A novel ann based adaptive ultrasonic measurement system for accurate water level monitoring', *IEEE Transactions on Instrumentation and Measurement*, pp. 3359–9456, 2019.

[10] M. Moussa, X. Zhang and C. Claudel, 'Flash flood detection in urban cities using ultrasonic and infrared sensors', *IEEE Sensors Journal*, vol. 16, pp. 1–1, Oct. 2016. DOI: 10.1109/JSEN.2016.2592359.

[11] E. Batur and D. Maktav, 'Assessment of surface water quality by using satellite images fusion based on pca method in the lake gala, turkey', *IEEE Transactions on Geoscience and Remote Sensing*, vol. PP, pp. 1–7, Nov. 2018. DOI: 10.1109/TGRS.2018.2879024.

[12] IN202047005455, *Water level measurement device and shoreline extraction method.* https://patentscope.wipo.int/search/en/detail.jsf?docId=IN283945449&_cid=P20-K95Y1N-18979-2, Accessed: 2022-02-15.

[13] IN201941002597, *Smart water quality monitoring system*, https://patentscope.wipo.int/search/en/detail.jsf?docId=IN237708484&tab=NATIONALBIBLIO&_cid=P12-K96P6L-98432-3, Accessed: 2022-02-15.

[14] H. Yasin, S. Zeebaree, M. M.Sadeeq *et al.*, 'Iot and ict based smart water management, monitoring and controlling system: A review', *Asian Journal of Research in Computer Science*, vol. 8, pp. 42–56, May 2021. DOI: 10.9734/AJRCOS/2021/v8i230198.

[15] S. Geetha and S. Gouthami, 'Internet of things enabled real time water quality monitoring system', *Smart Water*, pp. 1–19, 2016.

[16] S. B. Saraf, 'Iot based smart irrigation monitoring and controlling system', *IEEE International Conference On Recent Trends in Electronics Information & Communication Technology*, pp. 815–819, 2017.

[17] B. Alomar and A. Alazzam, 'A smart irrigation system using iot and fuzzy logic controller', *ITT 2018*, pp. 175–179, 2018.

[18] M. S. B. Praba, N. Rengaswamy and D. O. Vishal, 'Iot based smart water system', *Proceedings of the 3rd International Conference on Communication and Electronics Systems, ICCES*, pp. 1041–1045, 2018.

[19] K. Gupta, M. Kulkarni, M. Magdum, Y. Baldawa and S. Patil, 'Smart water management in housing societies using iot', *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT*, pp. 1608–1613, 2018.

[20] D. Paula, Gomes, Affonso, Rabelo and Rodrigues, 'An iot based water monitoring system for smart buildings', *IEEE International Conference on Communications Workshops, ICC Workshops 2019*, p. 6, 2019.

[21] A. A. Hasibuan and F. Fahrianto, 'Consumer's activity prediction in household water consumption based-iot', *7th International Conference on Cyber and IT Service Management (CITSM)*, pp. 1–7, 2019.

[22] C. S. Patel and J. A. Gaikwad, 'Design and development of iot based smart water distribution network for residential areas', *2019 International Conference on Communication and Electronics Systems (ICCES)*, 2019.

[23] N. Rapelli, A. Myakal, V. Kota and P. R. Rajarapollu, 'Iot based smart water management, monitoring and distribution system for an apartment.', *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pp. 440–443, 2019.

[24] R. Jisha, G. Vignesh and D. Deekshit, 'Iot based water level monitoring and implementation on both agriculture and domestic areas', *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pp. 1119–1123, 2019.

[25] I. S. Herath, 'Iot based intelligent domestic water management system', *2019 International Conference on Advancements in Computing, ICAC*, pp. 380–385, 2019.

[26] A. Arun, J. A. Sugirtharani, P. J. M. Carolina and C. A. Teresa, 'Smart water management in agricultural land using iot', *2019 5th International Conference on Advanced Computing and Communication Systems, ICACCS*, pp. 708–711, 2019.

[27] J. Vithanage, R. D. Silva, K. Karunarathne, M. D. Silva, P. Bogoda and R. K. et al., 'Iot-based smart platform to manage personal water usage', *2019 International Conference on Advancements in Computing, ICAC*, pp. 398–403, 2019.

[28] L. Harika, 'Cloud-based internet of things for smart water consumption monitoring system', *IEEE, International Conference on Communication and Electronics Systems (ICCES 2020)*, pp. 967–972, 2019.

[29] M. Pincheira, M. Vecchio, R. Giaffreda and S. S. Kanhere, 'Exploiting constrained iot devices in a trustless blockchain-based water management system', *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2020*, pp. 1–7, 2020.

[30] A. Ray and S. Goswami, 'Iot and cloud computing based smart water metering system', *2020 International Conference on Power Electronics and IoT Applications in Renewable Energy and its Control, PARC 2020*, pp. 308–313, 2020.

[31] S. A. Hami, A. Rahim, S. Y. Fadhlullah, S. Abdullah and Z. Muhammad, 'Iot based water quality monitoring system and evaluation', *Proceedings - 10th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2020*, pp. 102–106, 2020.

[32] A. K. Sarangi, 'Smart water leakage and theft detection using iot', *IEEE 2020 International Conference on Industry 4.0 Technology, I4Tech 2020*, pp. 46–50, 2020.

[33] V. Ranjan, M. V. Reddy, M. Irshad and N. Joshi, 'The internet of things (iot) based smart rain water harvesting system', *IEEE, 2020 6th International Conference on Signal Processing and Communication, ICSC 2020*, pp. 302–305, 2020.

[34] S. Konde and D. S. Deosarkar, 'Iot based water quality monitoring system', *SSRN Electronic Journa*, p. 6, 2020.

[35] K. Mekki, E. Bajic, F. Chaxel and F. Meyer, 'A comparative study of LPWAN technologies for large-scale IoT deployment', *ICT Express*, vol. 5, no. 1, pp. 1–7, Mar. 2019. DOI: 10.1016/j.icte.2017.12.005. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01670379.

[36] J. P. Queralta, T. Gia, Z. Zou, H. Tenhunen and T. Westerlund, 'Comparative study of lpwan technologies on unlicensed bands for m2m communication in the iot: Beyond lora and lorawan', *Procedia Computer Science*, vol. 155, pp. 343–350, 2019, The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2019),The 14th International Conference on Future Networks and Communications (FNC-2019),The 9th International Conference on Sustainable Energy Information Technology, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.08.049. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919309639.

[37] Ericsson, *Cellular networks for massive iot*, https://www.ericsson.com/en/reports-and-papers/white-papers/cellular-networks-for-massive-iot--enabling-low-power-wide-area-applications, Accessed: 2022-04-08.

[38] G. Nikolov and B. Nikolova, 'Virtual techniques for liquid level monitoring using differential pressure sensors', *Recent*, vol. 9, Jul. 2008.

[39] F. Larrarte, M. Lepot, F. Clemens *et al.*, 'Water level and discharge measurements', in Aug. 2021, pp. 35–104, ISBN: 9781789060119. DOI: 10.2166/9781789060119_0035.

[40] Onset. 'Mx2001'. (2015), [Online]. Available: https://www.onsetcomp.com/products/data-loggers/mx2001 (visited on 15th Dec. 2021).

[41] Milesight. 'Em500-swl'. (2015), [Online]. Available: https://www.milesight-iot.com/lorawan/sensor/em500-swl/ (visited on 15th Dec. 2021).

[42] MaxBotix. 'Mb7092, xl-maxsonar-wrma1'. (2005), [Online]. Available: https://www.maxbotix.com/ultrasonic_sensors/mb7092.htm (visited on 15th Dec. 2021).

[43] DFRobot. 'A01nyub waterproof ultrasonic sensor'. (2019), [Online]. Available: https://www.dfrobot.com/product-1934.html (visited on 15th Dec. 2021).

[44] C. Devices. 'Ps02 pressure sensor'. (2021), [Online]. Available: https://www.cuidevices.com/product/sensors/pressure-sensors/ps02-series (visited on 25th May 2022).

[45] STMicroelectronics, *Ldl1117 datasheet*, https://no.mouser.com/datasheet/2/389/dm00366442-1799212.pdf, Accessed: 2022-04-19.

[46] T. U. Rasmussen, 'Investigation of connectivity, energy consumption and real-time properties in a lora-network, using vibration sensors as case', p. 94, Feb. 2018.

[47] M. Technology, *Mcp73830/l data sheet*, https://ww1.microchip.com/downloads/en/DeviceDoc/MCP73830-L-Data-Sheet-DS20005049E.pdf, Accessed: 2022-04-19.

[48] ——, *Rn2483 data sheet*, https://ww1.microchip.com/downloads/en/DeviceDoc/RN2483-Low-Power-Long-Range-LoRa-Technology-Transceiver-Module-DS50002346F.pdf, Accessed: 2022-05-19.

[49] ——, *Rn2483 github examples*, https://github.com/search?q=topic:rn2483+org:MicrochipTech, Accessed: 2022-04-24.

[50] DFROBOT, *A01nyub documentation*, https://github.com/DFRobot/DFRobot_RaspberryPi_A02YYUW, Accessed: 2022-06-06.

[51] M. Technology, *Atmega324pb data sheet*, https://ww1.microchip.com/downloads/en/DeviceDoc/40001908A.pdf, Accessed: 2022-04-28.

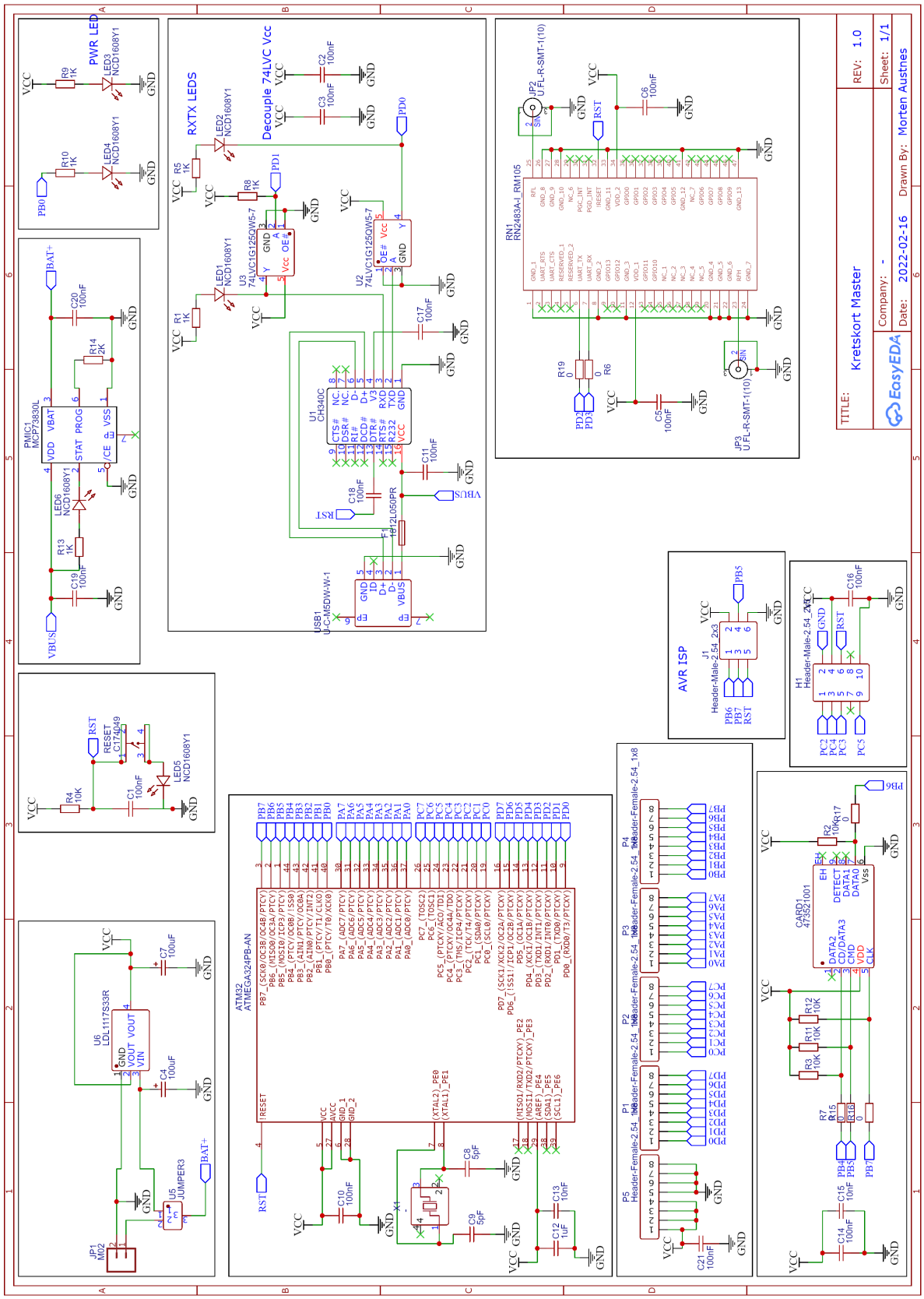
[52] Kartverket. 'Havnivå, tidevann og vannstand'. (2021), [Online]. Available: https://www.kartverket.no/til-sjos/se-havniva/resultat/?id=558690 (visited on 19th Dec. 2021).

Appendix A

A1: Parts list

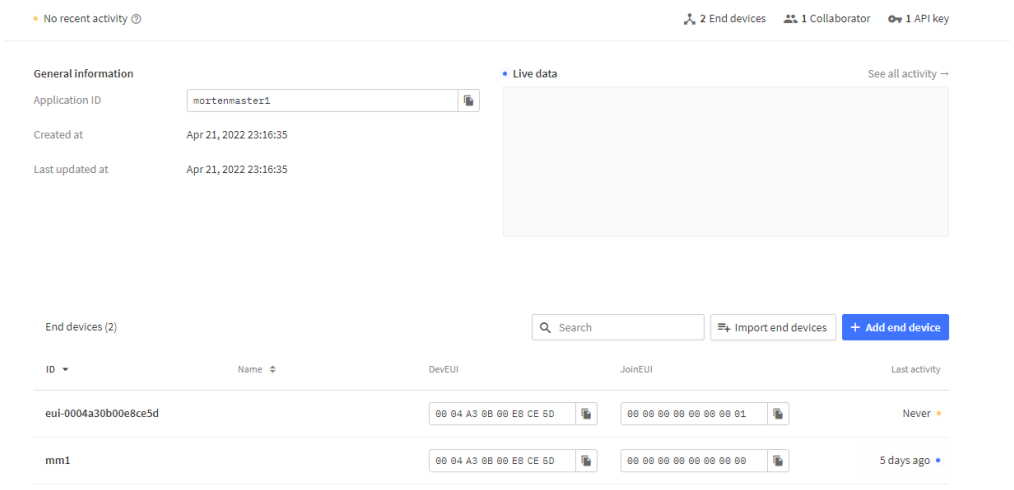
Designator	Description	Footprint	Value
X1	Crystal	SMD2520-4P	
H1	Male Header 2.54 2x5	Through-hole	
J1	Male Header 2.54 2x3	Through-hole	
RX,TX,PWR,PB0,RST,CHR	SMD LED	LED0603	
ATM32	ATMEGA324PB-AN	44-TQFP	
C1,C2,C3,C5,C6,C10,C11,C14,C16,C17,C18,C19,C20,C21	SMD capacitor	C0603	100nF
C4,C7	SMD capacitor	D7343	100uF
C8,C9	SMD capacitor	C0603	5pF
C12	SMD capacitor	C0603	1uF
C13,C15	SMD capacitor	C0603	10nF
CARD1	SD card holder	SD-SMD	
F1	Fuse	F1812	
JP1	Screw terminal	Through-hole	
JP2,JP3	Antenna connector	U.FL-R-SMT-1(10)	
P1,P2,P3,P4,P5	Female header 2.54 1x8	Through-hole	
PMIC1	MCP73830L charge manager	TDFN-6	
R1,R5,R8,R9,R10,R13	SMD resistor	R0805	1K
R2,R3,R4,R11,R12	SMD resistor	R0805	10K
R6,R7,R15,R19,R16,R17	SMD resistor	R0805	0
R14	SMD resistor	R0805	2K
RESET	Switch	SMD	
RN1	LoRa Module	RM	
U1	USB bus converter CH340C	SOP-16	
U2,U3	Bus driver 74LVC1G125QW5-7	SOT-25-5	
U5	Male Header 2.54 1x3	Through-hole	
U6	Voltage regulator LDL1117S33R	SOT-223-4	
USB1	USB connector	Micro-USB through-hole	

A2: Schematic

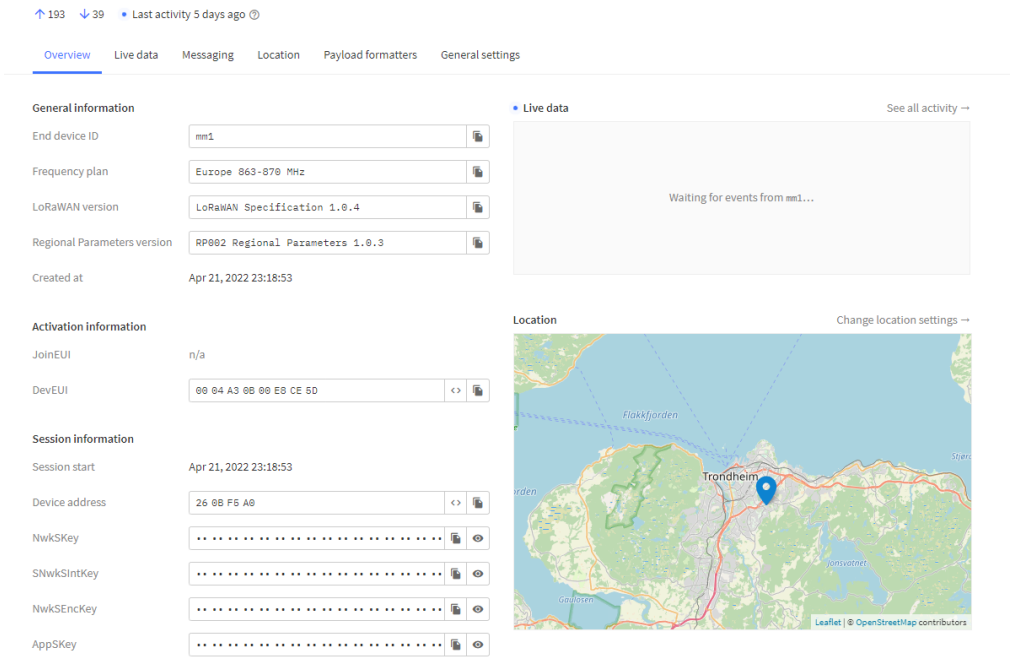

PWR LED
RXTX LEDS
Decouple 74LVC Vcc
AVR ISP
TITLE: Kretskort Master
REV: 1.0
Company: -
Sheet: 1/1
Date: 2022-02-16
Drawn By: Morten Austnes
EasyEDA

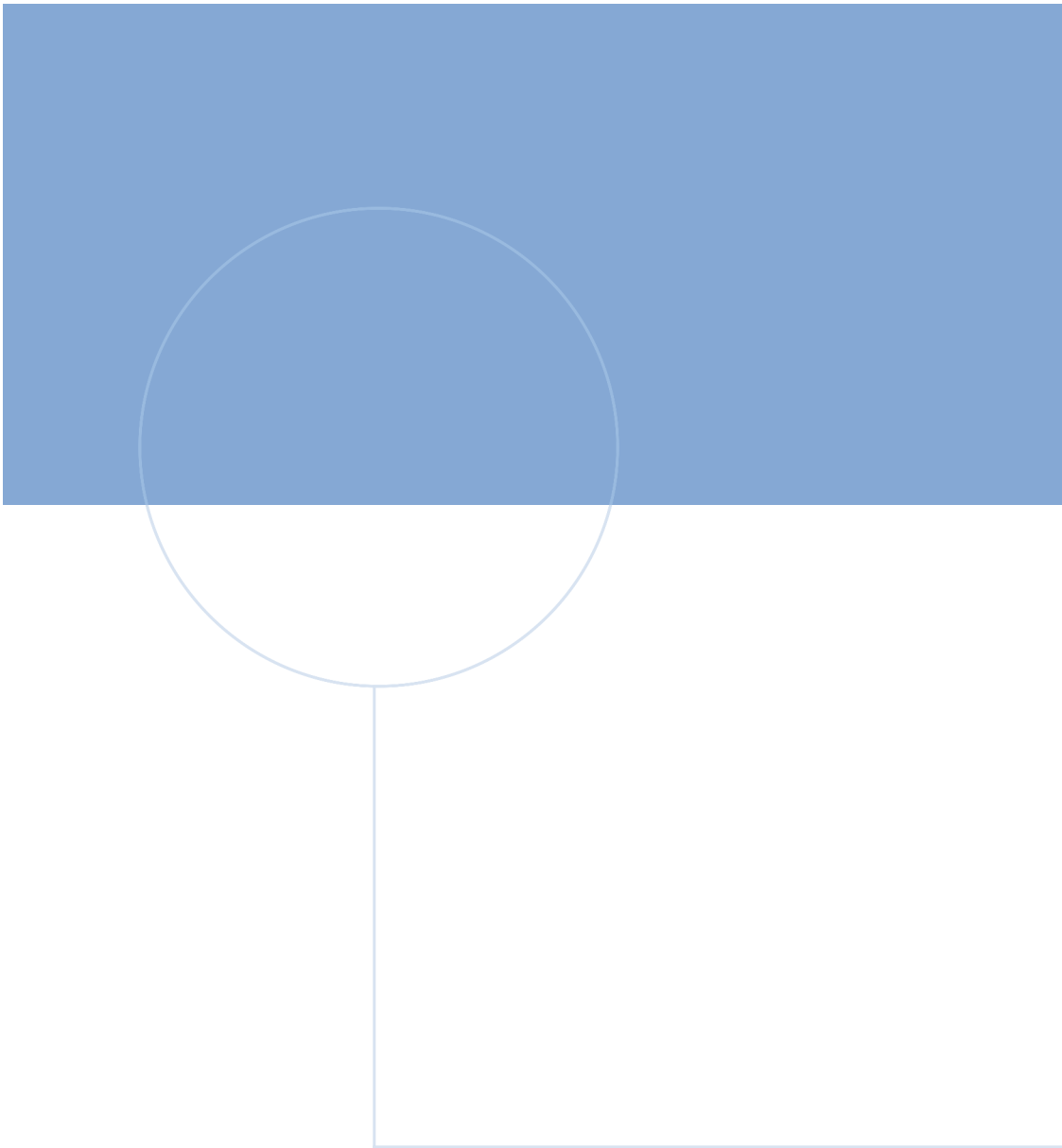
Appendix B

B1: TTN server GUI



B2: TTN device control panel





NTNU

Norwegian University of
Science and Technology