Master's thesis

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

Eivind Holen Jølsgard

LPWAN connectivity and embedded solutions for smart ocean monitoring buoys

Master's thesis in Cybernetics and Robotics
Supervisor: Jo Arve Alfredsen
August 2021

NTNU
Norwegian University of
Science and Technology

Eivind Holen Jølsgard

LPWAN connectivity and embedded solutions for smart ocean monitoring buoys

Master's thesis in Cybernetics and Robotics
Supervisor: Jo Arve Alfredsen
August 2021

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

MASTER'S THESIS ASSIGNMENT

Name: Eivind Holen Jølsgard
Program: Engineering Cybernetics
Title: LPWAN connectivity and embedded solutions for smart ocean monitoring buoys
Credits: 30 SP

Project description:

This project concerns the development and evaluation of the performance of a marine buoy concept based on LPWAN connectivity. The marine buoy is designed for carrying a specific acoustic telemetry receiver and relaying data received underwater from acoustic transmitters to an Internet backend. The objective of the buoy is to enable remote and near real-time acquisition of underwater telemetry data and the operational state of the receiver/buoy system, while preserving the desirable properties of current off-line logging receivers. These include ease of deployment, long operational life, and low cost. The buoy should thus be lightweight and allow deployment for several months in a remote fjord or coastal site consuming minimal amounts of power, while still being capable of transferring receiver detections wirelessly to a shore-based gateway/base station in near real time. Using multiple buoys for underwater hyperbolic positioning of acoustic transmitters should also be supported by incorporating distributed and accurate GNSS time synchronization of the buoys/receivers. The buoy controller should therefore include a GNSS solution that satisfies the strict power budget. Regarding power consumption, the requirement is set minimum three months operational life on a DD lithium-thionyl battery cell (nominally 35Ah@3.6V). The project will focus on exploring and comparing the performance of LoRa and NB-IoT as LPWAN technologies for the buoy concept, including demonstration of the concept in representative sea trials. Integration of the buoy concept as a resource into the DUNE framework of cooperating autonomous vehicles should also be explored and discussed. The project includes the following tasks:

- Survey of relevant technical literature on the proposed LPWAN technologies and an inquiry into their general properties.
- Background study of the existing solution for the LoRa-based buoy controller (SLIM – Synchronization and Lora Interface Module), acoustic telemetry receiver interfacing/protocol, distributed GNSS time synchronization, and underwater transmitter localization.
- Derive a requirements specification and a test specification/plan for the buoy controller prototypes.
- Explore the technical capabilities and requirements of the Nordic Semiconductor nRF9160 SiP with respect to integration of NB-IoT as a solution for both LPWAN connectivity and the other requirements defined for the buoy controller.
- Explore the feasibility of using a local high-accuracy ultra-low power temperature compensated RTC to assist GNSS time synchronization.
- Develop a buoy controller prototype based on the nRF9160.
- Plan and carry out trials suitable for analyzing the performance of the buoy systems with respect to energy consumption/battery life, coverage/range of wireless link, synchronization, and other relevant parameters.
- Implement and demonstrate a bridge for passing messages from the buoy system to the DUNE software framework.
- Make detailed documentation of the buoy system and test results, and discuss its capabilities and limitations.

Project start: 6 January 2021
Project due: 16 August 2021
Host institution: NTNU, Department of Engineering Cybernetics
Supervisor: Jo Arve Alfredsen, NTNU/DEC

Trondheim, 22 January 2021
Jo Arve Alfredsen

Abstract

This project continues the work on the Internet of Fish (IoF) project at NTNU. IoF aims at realising near real-time monitoring and bio-telemetry of fish for migration and behavioral research. The solution is based on synchronized stationary acoustic telemetry receiver buoys forwarding detections using Low Power Wide Area Network (LPWAN) technology in near-real-time.

This project has multiple aspects, focusing on the buoy controller at sea and communication with the mainland.

A buoy controller prototype has successfully been designed, built and tested. The prototype shows promising results and is fulfilling many of its requirements. Work remains before a commercially deployable solution is finalized.

Battery lifetime and ease of deployment are of major concern in the IoF project. The thesis explores the appliance of two standard LPWAN technologies: LoRaWAN and NB-IoT. Different aspects of the technologies are described, measured, and discussed, focusing on energy efficiency and suitability for the IoF project. Both LPWAN technologies have shown strengths and weaknesses concerning the IoF project, dependent on the use case, number of marked fish and deployment area. Sea trials are advised until further conclusions are made.

Effects of including a precise Real-Time Clock (RTC) to the buoy controller has been explored to increase the period between energy-demanding Global Navigation Satellite System (GNSS) time synchronization procedures, further increasing battery life. Test results show the RTC performing well within its stated drift of 1.5ppm. Using RTC frequency estimation, a local time drift down to 306us in 9.8 hours has been measured in stationary temperature environments. Further testing is required for evaluating the performance in changing temperature areas.

A DUNE: Unified Navigation Environment bridge is made to allow unmanned surface vehicles access to buoy observations, partly enabling USVs to cooperate in the IoF project.

Sammendrag

Denne oppgaven fortsetter arbeidet med Internet of Fish (IoF) prosjektet ved NTNU, med mål om å realisere overvåkning og posisjonering av fisk i nær sanntid. Dette gir verdifull innsikt i fiskeatferd og bevegelsesmønstre. Løsningen baserer seg på utplassering av stasjonære havbøyer med synkroniserte akustiske sensorer, hvor en kontrollenhet på havbøyen videresender deteksjoner til fastlandet ved hjelp av lavenergi og strømgjerrig nettverksteknologi (LPWAN).

Prosjektet har mange aspekter, med hovedfokus på kontrolleren i havbøyene og kommunikasjon med fastlandet.

En prototype for bøye-kontrolleren har blitt designet, satt sammen og testet, med lovende resultater. Prototypen oppfyller mange av kravene satt i starten av prosjektet. Videre arbeid gjenstår før en fullverdig løsning er på plass.

Et viktig aspekt ved IoF prosjektet er enkel utplassering og vedlikehold av havbøyene. Her er batterilevetid viktig for å øke vedlikeholdsintervallet og redusere kostnader. Prosjektet presenterer, undersøker og diskuterer fordeler og ulemper ved to LPWAN teknologier; NB-IoT og LoRaWAN. Ulike aspekter har blitt utforsket med hovedfokus på energieffektivitet og egnethet for IoF. Begge nettverksteknologiene viser styrker og svakheter, avhengig av bruksområde, antallet fisk og område systemet opererer i. Testing i reelle miljøer anbefales før videre konklusjoner trekkes.

Oppgaven utforsker også nytten av en presis sanntidsklokke (RTC) i bøyekontrolleren. Målet er å øke batterilevetiden ved å redusere behovet for energikrevende satellittoppdateringer. Oppdateringene brukes for å synkronisere tiden til havbøye-kontrolleren. Tester viser lovende resultater, med lavere drift enn forventede 1.5 PPM. Ved bruk av frekvensestimering har driften av lokal tid blitt målt til 306 us etter 9.8 timer i stabile omgivelsestemperaturer. Videre testing trengs for å evaluere bruksnytten i områder med varierende temperatur.

En løsning for å videresende IoF data til DUNE: Unified Navigation Environment er utviklet. Dette gir ubemannede overflatefartøy adgang til bøye-observasjoner, som delvis tillater fartøyene å samarbeide i IoF prosjektet.

Preface

This Thesis forms the foundation of evaluation for my Master's Thesis at the Cybernetics Department (ITK) at the Norwegian University of Science and Technology (NTNU). The project is supervised by Jo Arve Alfredsen and continues the work done by Marius Rundhovde and others before him in the IoF project.

I want to thank my supervisor, Jo Arve Alfredsen, for providing resources and knowledge of the former built system as well as being available for questions and discussions. I would also like to thank the staff at ITKs electrical workshop for access to tools and equipment and Nikolai Lauvås for providing me with IMC message formats for the DUNE bridge.

At the beginning of the project, available resources consist of an nRF9160 development kit, a previously built SLIM module, TBR 700 RT acoustic receivers, Multitech Conduit and IP67 LoRa gateways, an MQTT broker on the otter01 virtual machine at NTNU, and access to previously written reports and software.

Although not a large part of this thesis, hardware and driver debugging have been a large part of the project. A Digilent Analog Discovery 2 has been very useful for debugging these problems, providing oscilloscope, voltage source, and a logic and bus protocol analyzer in one package. I would therefore like to applaud Omega Verksted for providing these at a low cost to students. For those working on similar projects, I would recommend having a tool similar to the Analog Discovery. Also, concerning recommendations and hardware design, I would like to add: If assembling PCBs by hand, think twice before designing hardware using small components!

I would also like to thank Micro Crystal AG for providing samples of their new RV3032-C7 RTC before becoming available on the market.

Eivind H. Jølsgard
August 2021

Table of Contents

Master's Thesis Assignment i

Abstract ii

Sammendrag iii

Preface iv

Abbreviations xi

List of Figures xiv

List of Tables xvi

1 Introduction 1

1.1 Problem description . . . 1

1.2 Previous work . . . 1

1.3 Organisation of thesis . . . 2

2 Background information 3

2.1 Hardware . . . 3

2.1.1 Acoustic tags . . . 3

2.1.2 Acoustic Telemetry Receiver . . . 3

2.1.3 The ARM Cortex-M microprocessors . . . 3

2.1.4 The nRF91 Series and the nRF9160 microcontroller . . . 3

2.1.5 The nRF9160 Development Kit . . . 4

2.1.6 LoRa gateway . . . 4

2.1.7 Server . . . 5

2.1.8 PCB hardware . . . 5

2.2 Software . . . 6

2.2.1 Zephyr RTOS . . . 6

2.2.2 Server software . . . 7

2.2.3 DUNE: Unified Navigation Environment . . . 7

2.3 Communication technology and protocols . . . 7

2.3.1 Fish tag acoustic protocols . . . 7

2.3.2 IoF message formats . . . 7

2.3.3 Transmission from sea buoy to the mainland (LPWAN) . . . 8

2.3.3.1 Communication technology basics . . . 8
2.3.3.2 Energy efficiency metrics . . . 9
2.3.3.3 Issues in IoT energy conservation . . . 9
2.3.3.4 NB-IoT . . . 10
2.3.3.5 NB-IoT networks . . . 10
2.3.3.6 LoRa . . . 11
2.3.3.7 LoRaWAN . . . 12
2.3.3.8 MQTT . . . 14
2.3.3.9 CoAP . . . 16
2.4 Development Environment . . . 16
2.4.1 KiCad - Schematics and PCB design . . . 16
2.4.2 LibraryLoader . . . 16
2.4.3 FreeRouting . . . 16
2.4.4 VS Code - Code editor . . . 17
2.4.5 nRF Connect and the nRF Connect SDK . . . 17
2.4.6 Microchip studio . . . 17
2.4.7 Git . . . 17
2.4.8 LSTS toolchain . . . 17

3 System Requirements **19**
3.1 Buoy prototype requirements . . . 19
3.1.1 All-use-case requirements . . . 19
3.1.2 Use case dependent requirements . . . 20
3.1.3 Shield requirements . . . 20
3.2 Standalone buoy module requirements . . . 20
3.3 LPWAN requirements . . . 22
3.4 DUNE bridge requirements . . . 22

4 System overview **23**
4.1 Acoustic tags and telemetry receiver . . . 23
4.2 cSLIM buoy controller and sea buoy . . . 24
4.3 LoRa Gateway and MQTT broker . . . 24
4.4 IoF application . . . 25
4.5 DUNE bridge . . . 25

5 Shield development **26**

5.1 Interface requirements . . . 26
5.2 PCB design considerations . . . 27
5.2.1 PCB manufacturing and assembly considerations . . . 27
5.2.2 Component selection . . . 27
5.3 Schematics and PCB layout . . . 29
5.4 Production and assembly . . . 30
5.5 Testing and verification . . . 31

6 Software development **34**
6.1 cSLIM software overview . . . 34
6.1.1 boards/arm/cSLIM . . . 35
6.1.2 src/buffers . . . 35
6.1.3 src/devices . . . 35
6.1.4 src/drivers . . . 35
6.1.5 src/messages . . . 35
6.1.6 src/mqtt . . . 36
6.1.7 src/tasks . . . 36
6.1.8 src/time . . . 36
6.1.9 src/ugui . . . 36
6.2 Driver and device adaptions . . . 36
6.2.1 LED and button drivers . . . 36
6.2.2 SPI driver . . . 36
6.2.3 OLED device . . . 36
6.2.4 GNSS device . . . 37
6.2.5 I2C driver . . . 37
6.2.6 RTC device . . . 37
6.2.7 LoRa device . . . 37
6.3 Software tasks . . . 37
6.4 Time synchronization . . . 38
6.5 RTC frequency estimation . . . 40
6.6 LoRa module . . . 41
6.6.1 LoRaWAN RN Parser user interface . . . 41
6.7 DUNE Bridge . . . 43

7 Putting it together **45**

8 Testing and results **46**

8.1 Test setup . . . 46

8.1.1 RTC drift and local time correction . . . 46

8.1.2 Time synchronization . . . 46

8.1.3 Energy consumption . . . 47

8.1.3.1 Overall energy consumption . . . 47

8.1.3.2 LPWAN power consumption . . . 47

8.1.4 Data throughput . . . 48

8.1.5 DUNE Bridge . . . 48

8.2 Energy consumption . . . 49

8.2.1 cSLIM system energy consumption . . . 49

8.2.2 LPWAN energy consumption . . . 49

8.3 LPWAN coverage . . . 49

8.4 Airtime and data throughput . . . 50

8.4.1 LoRaWAN . . . 50

8.5 Timing requirements . . . 50

8.5.1 RTC-drift and local time measurements . . . 50

8.5.2 Local time drift with RTC frequency estimation . . . 55

8.5.3 Time synchronization . . . 55

8.6 DUNE bridge . . . 57

8.7 Validation of cSLIM module . . . 58

8.7.1 Underwater Acoustic Receiver(UAR) . . . 58

8.7.2 Cellular SLIM module . . . 58

8.7.3 nRF9160-DK cSLIM shield requirements . . . 60

8.7.4 nRF9160 cSLIM standalone module (cSLIM-SA) requirements . . . 60

8.8 Validation of DUNE bridge . . . 61

9 Discussion **62**

9.1 cSLIM hardware performance . . . 62

9.1.1 Effect of including RTC . . . 62

9.1.2 Static energy consumption . . . 62

9.2 Use of a RTOS in cSLIM module . . . 63

9.3 Pros and cons of NB-IoT and LoRaWAN in IoF application . . . 63

9.3.1 Spectral efficiency . . . 63

9.3.2 Distance to gateway/cell tower . . . 63

9.3.3 Energy per bit . . . 64
9.3.4 Battery and network lifetime . . . 64
9.3.5 Performance in low activity areas . . . 65
9.3.6 Performance in high activity areas . . . 65
9.3.7 Operation complexity and costs . . . 66
9.3.8 Summary of LPWAN technology in IoF . . . 66
9.4 cSLIM power consumption and operational life . . . 66
9.5 DUNE Bridge . . . 67
9.6 Goal and method . . . 67

10 Conclusion **69**

11 Future Work **70**
11.1 Standalone cSLIM module . . . 70
11.2 Energy savings on local time calibration and GNSS module . . . 70
11.3 GNSS time synchronization procedure . . . 71
11.4 Verification of TBR time synchronization . . . 71
11.5 Verification of RS232 . . . 71
11.6 Final implementation and verification of local log . . . 71
11.7 LoRa module improvements . . . 71
11.8 NB-IoT communication improvements. . . . 72
11.9 Extended evaluation of LoRaWAN and NB-IoT performance by field tests . . . 72
11.10 Extended evaluation of RTC and local time performance by field tests . . . 72
11.11 IoF Application, DUNE Bridge and JSON MQTT payload . . . 72
11.12 Bug fixing . . . 72

A Getting started with further development **73**
A.1 nRF9160 . . . 73
A.2 WLR089u0 . . . 73

B System requirements **75**
B.1 Underwater Acoustic Receiver(UAR) requirements . . . 75
B.2 Cellular SLIM (cSLIM) module requirements . . . 75
B.3 nRF9160-DK cSLIM shield requirements . . . 77
B.4 DUNE bridge requirements . . . 78

C Digital appendixes **80**

C.1 Application source code 80
C.2 PCB project files 80
C.3 Test Specifications 80

D Shield schematics **81**

E Shield PCB layout **92**

Bibliography **94**

Abbreviations

API Application Programming Interface.

AT Attention, referring to instructions often used to control a modem.

ATR Acoustic Telemetry Receiver.

CE Coverage Enhancement.

CoAp Constrained Application Protocol.

CPU Central Processing Unit.

CS Chip Select.

cSLIM Cellular Synchronisation and Lora Interface Module.

CSS Chirp Spread Spectrum.

DK Development Kit.

DRX Discontinuous Reception.

DUNE DUNE: Unified Navigation Environment.

e-ink electronic ink.

Eb Energy per bit.

ECAD Electronic Computer Aided Design.

ECL Extended Coverage Level, same as CE-level.

eDRX Extended Discontinuous Reception.

EVI External Event Interrupt Input.

FRAM Ferromagnetic Random Access Memory.

GNSS Global Navigation Satellite System.

GPIO General purpose input/output.

GPS Global Positioning System.

GSM Global System for Mobile Communications.

GW Gateway.

HTTP Hypertext Transfer Protocol.

I2C Inter-integrated circuit, serial communication bus.

ID Identification.

IMC Inter Module Connect.

IoF Internet of Fish.

IoT Internet of Things.

IP Internet Protocol.

ISM Industrial, Scientific, Medical.

ITU International Telecommunication Union.

JSON JavaScript Object Notation.

LCD Liquid crystal display.

LED Light Emitting Diode.

LLC Logic level converter.

LoRa Modulation technique.

LoRaWAN LPWAN Standard using LoRa modulation.

LPSAN Low Power Short Area Network.

LPWAN Low Power Wide Area Network.

LTE Long Term Evolution.

LTE-M Long Term Evolution Machine Type Communication.

MAC Medium Access Control.

MCU Microcontroller unit.

MIC Message Integrity Check.

MQTT Message Queuing Telemetry Transport.

NB-IoT Narrowband Internet of Things.

OLED Organic Light Emitting Diode.

OPC Open Protocols Coding.

PCB Printed Circuit Board.

PPM Pulse Per Minute.

PPS Pulse PerSecond.

PSM Power Saving Mode.

QoS Quality of Service.

RAM Random Access Memory.

RF Radio Frequency.

RTC Real Time Clock.

RTOS Real-Time Operating System.

RX Receive.

SCM Supply Chain Management.

SDK Software Development Kit.

SF Spreading Factor.

SIM Subscriber Identity Module.

SiP System in Package.

SLIM Synchronisation and Lora Interface Module.

SMD Surface Mounted Device.

SNR Signal to Noise Ratio.

SPI Serial Peripheral Interface, serial communication bus.

SSH Secure Shell.

SSL Secure Sockets layer.

STM State Machine.

TAU Tracking Area Update.

TCP Transmission Control Protocol.

TDoA Time Difference of Arrival.

TLS Transport Layer Security.

TTL Transistor-transistor logic.

TX Transmit.

UART Universal Asynchronous Receiver-Transmitter, serial communication standard.

UDP User Datagram Protocol.

uSD micro SD(secure digital).

UTC Coordinated Universal Time.

VPN Virtual Private Network.

List of Figures

2.1 nRF9160 development kit . . . 4
2.2 Device tree source example . . . 6
2.3 cSLIM non-secure device tree bindings . . . 6
2.4 IoF message frame . . . 8
2.5 IoF message tag detection frame . . . 8
2.6 IoF message TBR status frame . . . 9
2.7 IoF Message buoy controller status frame . . . 9
2.8 Coverage of LTE-M and NB-IoT networks worldwide. . . . 11
2.9 Coverage of NB-IoT in Norway by Telenor . . . 12
2.10 Coverage of NB-IoT in Norway by Telia . . . 13
2.11 Chirp Spread Spectrum modulation . . . 13
2.12 Spreading factor transmit time . . . 14
2.13 LoRa MAC frame . . . 15
2.15 MQTT frame . . . 15
2.16 TCP frame . . . 15
2.17 CoAp frame format . . . 16
2.18 UDP frame . . . 16
2.14 The Things Network LoRa gateways in south of Norway . . . 18
3.1 Single-buoy deployment . . . 21
3.2 Multi-buoy deployment . . . 21
4.1 IoF overview . . . 23
4.2 Acoustic telemetry receiver and tag . . . 23
4.3 cSLIM buoy controller modules and interfaces . . . 24
4.4 IoF sea buoy . . . 25
4.5 MultiTech Conduit LoRaWAN base station . . . 26
5.1 3D view of PCB . . . 30
5.2 SMD solder paste appliance . . . 30
5.3 PCB ready for the reflow oven . . . 31
5.4 Solder paste reflow temperature profiles . . . 31
5.5 Final PCB prototype . . . 32
6.1 cSLIM overview . . . 34
6.2 GPS Task STM . . . 38
6.3 Time Task STM . . . 39

6.4 Local time update sequence diagram 39
6.5 Local time calculation and frequency estimation 40
6.6 Successful LoRa join procedure 42
6.7 Dune Bridge flow while running 44
7.1 cSLIM buoy controller hardware 45
7.2 Fixing hardware with solder iron 45
8.1 Expected millisecond part of tag detections 47
8.2 LPWAN power consumption test setup 48
8.3 Current measurements of nRF9160 and WLR089u0 in standby mode 50
8.4 NB-IoT current measurements of tag detection transmissions 51
8.5 NB-IoT retaining connection with PSM 52
8.6 LoRaWAN current measurements of tag detection transmissions 53
8.7 RTC drift without frequency estimation 54
8.8 RTC drift with frequency estimation 56
8.9 DUNE Bridge IMC::FishTag message 57
8.10 DUNE Bridge IMC::TBRSensor message 58

List of Tables

2.1 Key features of NB-IoT and LoRaWAN 14

5.1 cSLIM peripherals 27

6.1 LoRa RN Parser command overview 41

6.2 LoRa RN Parser response messages 41

8.1 cSLIM current consumption 49

8.2 Energy consumption of transmitting IoT messages using NB-IoT 49

8.3 Energy consumption of transmitting IoT messages using LoRaWAN 52

8.5 Local time drift without RTC frequency estimation 52

8.4 LoRaWAN throughput measurements 55

8.6 Local time drift with RTC frequency estimation 55

1 Introduction

1.1 Problem description

Early on, fish movement and behavior studies were based on observation, marking, and recapturing of fish. Based on principles of sonar, developed to detect submarines during World War I, acoustic fish monitoring has been ongoing since the late 1950s, with the examination of salmon mitigation [44] being one of the first publications. The use of acoustic telemetry brought a new advantage to fish research; locating and identifying individuals without recapturing [15].

Since then, it has been common to equip fish with small devices that transmit or reflect acoustic signals to a receiver. The acoustic receivers are placed in the sea or under a boat, gathering information for some time. The sensors are then collected before the data is downloaded and reviewed. For instance, this method was used to gather information on fish travel in Mjøsa and Gudbrandsdalslågen, Innlandet, Norway in 2020 [16].

New hardware and communication technology enables connecting more diminutive and more power-efficient devices to a network for the past decades. The phenomenon is commonly known as the Internet of Things (IoT). It has lead to the development of real-time monitoring systems of fish, giving the ability to receive information of fish location and movement observations. At the same time, the sensors are still active at sea. The IoF (Internet of Fish) project is such a project, ongoing for the past years.

Fisheries and aquaculture have gained a significant and growing role in providing food, nutrition, and employment [11]. Moreover, as monitoring fish location gain a "variety of insights into migration, habitat use, behavior, productivity, or survival of fish" [15], this can be used to improve living conditions and reduce mortality in aquaculture, hence increased earnings for the fish farmers.

Due to a harsh marine environment and costly maintenance operations, marine life monitoring and aquaculture research as fish biotelemetry is a cost-demanding field of study. Therefore, finding simple and cost-effective ways to gather and publish data is critical to increasing utility costs. This includes reducing hardware costs and reducing operational costs, for instance, by increasing the lifetime of observation buoys and hence increasing maintenance intervals. Also, for wirelessly connected monitoring buoys, a cost-effective way of transmitting observations is appropriately wise.

1.2 Previous work

Since the first publication by Parker S. Trefethen in 1956 [44], there has been numerous research on fish biotelemetry. After the large sonic tags used by Trefethen, attached to adult salmon with a nickel-chromium hog-ring, multiple tags have been developed, both active and passive. With its 0.2 g and a life of 18-22 days, the active miniature transmitter by Naef-Daenzer et al.[25] is one of the smallest developed. The small form factor and lightweight provide opportunities to study species that previously were too small to be tagged.

Hassan et al. [14] introduced the Internet of Fish (IoF) concept, aiming to realize a real-time system for monitoring and positioning of fish. The system consists of three layers; the perception layer, network layer, and the presentation layer. The perception layer consists of monitoring buoys equipped with submerged acoustic receivers, receiving signals from acoustic tags in the fish. The network layer consists of a gateway receiving packets from the monitoring buoys using LPWAN radio links. The packets are forwarded through the Internet to a server application presenting the information to the user. Using a Time Difference of Arrival (TDoA) algorithm, the fish location in a commercial marine farm in Norway could be computed in 3D.

For the IoF project, the SLIM module, initially was developed by NTNU PhD candidate Hassan. The SLIM module is a hardware module designed for monitoring buoys, synchronizing the acoustic receivers using GPS signals, receiving tag observations and transmitting them to the Gateways using LPWAN radio technology. The SLIM module aims to be an energy-efficient solution, increasing

battery lifetime and deployment time. In his Master's Thesis, Rundhovde [33] developed the SLIM software further. Based on measured crystal oscillator behavior at different temperatures, Rundhovde estimated the actual frequency of the crystal driving the microcontroller based on the device temperature. The modifications resulted in a reduction of device time drift in the SLIM module, increasing the time between GPS updates. Together with some other fixes for the GPS module, the result was a reduction in power consumption, from 25.8 mA to 10.4 mA with tag detections and 25.5 mA to 7.1 mA without tag detections.

In his Master thesis, Kjelsvik [29] presents an application layer software for enabling real-time monitoring of IoF data. The IoF application consists of a back-end service receiving observation data and the status of the acoustic receivers and a front-end presenting the data to the user. Also, Kjelsvik aimed to implement the TDoA based positioning service.

In addition to the buoy observation, acoustic receivers have been placed on Autonomous Surface Vehicles running the DUNE framework. Also, an MQTT Transport into DUNE has been developed by the author [18] as part of the course TTK22 last autumn. However, it has not been thoroughly tested or included in the DUNE repository.

Regarding comparing LPWAN technologies, there have been case studies exploring NB-IoT and LoRa's strengths and weaknesses. Mohan [22], in brief, compared ten key points, from deployment status and options to transmit-current, data rates, and mobility. Zanaj et al. [51] presents an extensive review of LPWAN (and LPSAN) technologies regarding energy efficiency and power consumption, as well as discussing current limitations and challenges. Santiago and Dr. Arockiam [36] presents an overview of energy efficiency in the Internet of Things and discuss "issues and ways to minimize the energy consumption in IoT environment." This is only a drop of papers and articles in the LPWAN research ocean.

1.3 Organisation of thesis

The following describes the organization of the rest of this thesis. Chapter 2 introduces the hardware and software used and the development environment and describes LPWAN technologies and other relevant communication technology. Chapter 3 presents system requirements set at the start of the project period, including buoy prototype requirements, LPWAN requirements, and requirements for the DUNE bridge. Chapter 4 gives an overview of the system, from the acoustic tags, through the buoy and LPWAN communication to the end applications. Chapter 5 then describes the requirements, design, production, and verification of the cSLIM shield developed for the nRF9160-DK based prototype. Chapter 6 describes the software developed for the application, including the cSLIM buoy controller and DUNE bridge. Chapter 7 briefly describes the system put together, including pictures of the buoy controller prototype. Chapter 8 then explains the test setup used for this thesis before presenting test results and verifying system requirements introduced in chapter 3. Chapter 9 discusses the cSLIM buoy controller performance and the advantages and disadvantages of the explored LPWAN technologies concerning the IoF project before giving some comments on the goals and method of this project. Finally, concluding remarks are given in chapter 10, while chapter 11 provides an overview of suggested improvements and future work.

2 Background information

The following subsections present background information of the work done in this thesis. Section 2.1 presents the acoustic tags and receiver, the cSLIM module, including the nRF9160 System in Package (SiP) and other PCB hardware. The LoRa Gateway and MQTT broker is also presented. Section 2.2 presents software, including the Zephyr RTOS and DUNE: Unified navigation environment. Section 2.3 introduces relevant communication technology and protocols, with LoRaWAN and NB-IoT being the main focus. At last, section 2.4 presents the development environment.

2.1 Hardware

2.1.1 Acoustic tags

An acoustic tag is a small sound-emitting device emitting sound waves in subsurface environments, where sound waves carry over long distances. The tags at least consist of a battery, modulator, and transmitter. Some tags also contain one or multiple sensors gathering data transmitted in the acoustic messages. The acoustic messages are modulated, often using a differential pulse position modulation, containing information about the tag-id and potentially sensor data. The acoustic tags are required to be physically small and at low weight while still being capable of transmitting over long distances and having a long operational lifetime, which represents conflicting requirements. Practical and balanced compromises must therefore be utilized in the tag design. According to Kjelsvik [29] acoustic tags often use pulsed transmitters, transmitting information at pseudo-random intervals. Different tag types exist with varying intervals of transmission and transmission power, dependent on the application's requirements.

2.1.2 Acoustic Telemetry Receiver

An acoustic telemetry receiver (ATR) is a hydrophone specifically designed to receive and decode signals from acoustic tags. The ATR transforms the acoustic signals to electrical energy. Most hydrophones are made using a piezoelectric element, producing small electrical signals when applied to pressure. Acoustic underwater receivers can be either omnidirectional, receiving signals from all directions, or unidirectional, receiving signals from a single direction.

2.1.3 The ARM Cortex-M microprocessors

The Cortex-M series developed by ARM is a collection of 32-bit microprocessor architectures. The collection consists of various microcontroller designs optimized for cost and energy efficiency. ARM, which does not produce microcontrollers themselves, sells their design architecture as intellectual property. This enables multiple microcontroller vendors to base their design on the same architecture, reducing time to market and development cost.

2.1.4 The nRF91 Series and the nRF9160 microcontroller

The nRF91 Series is Nordic Semiconductors collection of low-power cellular devices. The series aim to bring advanced application performance and possibilities to cellular IoT, making "single chip solution a real possibility in many cases"[26]. The first device released in the series is the nRF9160 System in Package(SiP), featuring a modem with LTE capability, an RF multiband radio, and incorporating the Arm Cortex M-33 CPU architecture. The SiP has 1MB of flash, 256kB of RAM, 32 GPIOs, analog interfaces, and four digital interfaces that support SPI, I2C, and UART. For operation and connectivity, the SiP needs an external power source, SIM card, and antenna.

The nRF9160 supports both LTE-M and NB-IoT with eeDRX and PSM power-saving modes. According to the product brief [28] the floor current of the LTE connection is in the range of a few

uA when using the power-saving modes.

2.1.5 The nRF9160 Development Kit

The nRF9160 Development Kit (DK) from Nordic Semiconductor, shown in figure 2.1 is a development platform provided by Nordic Semiconductor to ease the development of applications using the nRF9160 SiP. With a dedicated NB-IoT and LTE-M antenna, a GPS antenna and a 2.4GHz antenna the board can be used for developing a range of connected IoT applications, both using short-range wireless protocols as Bluetooth, openThread and Mesh as well as long-range connectivity with NB-IoT and LTE-M. For clarification, the 2.4 GHz antenna is connected to the nRF52840 board controller, not the nRF9160.

The nRF52840 board controller can connect and disconnect the nRF9160 SiP to external peripherals on the development kit. This includes LEDs, buttons and switches, external memory and UART logging capabilities, as well as the interface pins between the nRF9160 SiP and the nRF52840. The complete list of analog switches and their routing options is given in the nRF9160 board controller documentation [27].

The header pins of the nRF9160-DK are compatible with Arduino Uno boards. However, as the nRF9160-DK uses 1.8V or 3V transistor-transistor logic (TTL), where Arduino Uno uses 5V, not all pre-developed boards are suitable.

The development kit has an internal voltage regulator and can be powered over USB, an external battery, or another stable power source. The onboard Segger J-Link Debugger enables the microcontroller to be programmed and debugged through the USB connector. In addition, 10-pin `debug` headers are available for access using an SWD interface.

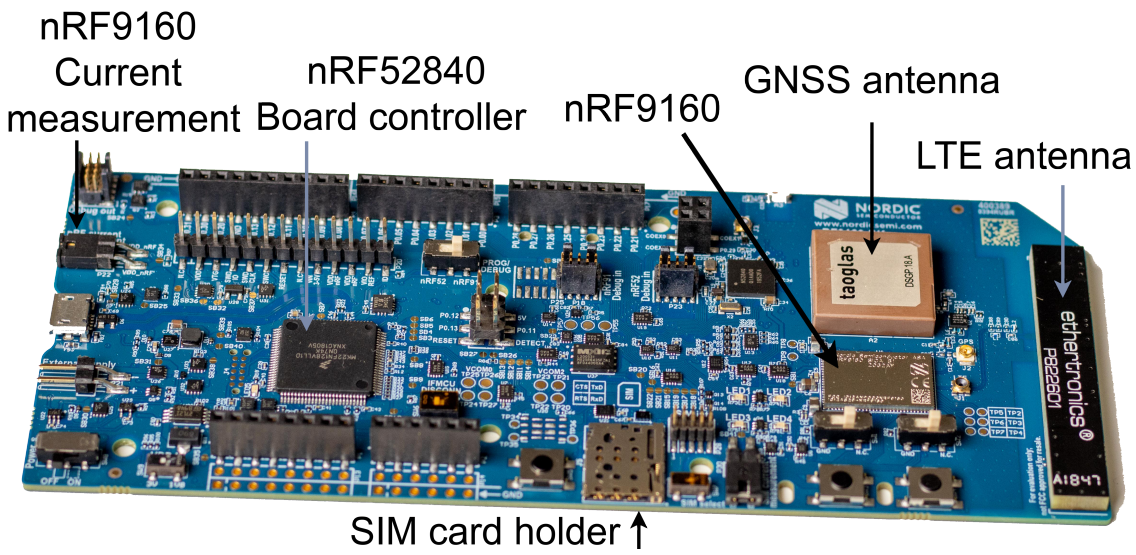


Figure 2.1: nRF9160 development kit. Photo: Eivind Jølsgard/NTNU (CC BY 4.0)

2.1.6 LoRa gateway

A LoRa gateway is a network router equipped with a LoRa concentrator (transmitter/receiver module) to convert packets between a LoRaWAN and another network interface as Ethernet, GSM, or WiFi. In other words, the LoRa gateway can receive LoRa messages from an end device over a LoRaWAN and forward the messages to a server through, e.g., the Internet.

2.1.7 Server

The MQTT Broker is running at a virtual server provided by the IT department at NTNU, named `otter01.it.ntnu.no`. The server is accessed through SSH and NTNU-user credentials. NTNUs VPN tunnel (Cisco AnyConnect) must be used if the client computer is not connected to the NTNU (Eduroam) network.

2.1.8 PCB hardware

The following subsection presents some of the hardware used or considered for the PCB designed in this project. Specific components and component selection are further described in section 5.2.2.

Voltage Regulators

Voltage regulators can mainly be divided into two categories: linear voltage regulators and switching voltage regulators.

Linear Voltage Regulators

For linear voltage regulators, the input voltage must be higher than the required output voltage. According to Schweber [35] linear Voltage Regulators are simple, cheap, and give little noise to the surrounding circuitry. On the other hand, they are not very power efficient, especially when there is a gap between input and output power.

Switching Voltage Regulators

Schweber [35] describes switching regulators as "highly efficient and able to step up (boost), step down (buck), and invert voltages with ease. Contemporary modular chips are compact, reliable, and available from multiple suppliers". On the other hand, switching voltage regulators often require more external components than linear ones, resulting in more costly solutions. In addition, frequent switching causes noise due to ripples at the switching rate. Therefore, when designing devices with noise-sensitive components, care must be taken to avoid interference. This applies to RF-enabled devices.

Displays

There are many different techniques for displaying pixels on a screen. For embedded solutions, there are primarily two display types: LCD and OLED. LCD and OLED displays use LEDs to display information, either by having the LEDs providing backlight (LCD) or illuminating the pixels themselves(OLED). This results in a continuous power drawn while the display is active. Lately, e-ink displays also have become prominent. These displays use electronic ink to display information, only drawing power when changing the display and keeping pixel color on power loss. One typical application for e-ink displays is price tags in consumer stores.

Logic level converters

Logic level converters are, as the name implies, regularly used to connect devices with different TTL levels (transistor-transistor-logic). Common is conversions between the three standard voltage levels 5V, 3.0V-3.3V, and later 1.8V. As the components selected for this design work with 3V TTL levels, no voltage conversion is necessary. However, some devices are powered down during operation while being connected to pins at the microcontroller used by other devices. Therefore, Logic level converters with enable pins are added to the SPI lines before powered-off devices to ensure tri-state operation.

2.2 Software

2.2.1 Zephyr RTOS

The Zephyr RTOS is an open-source real-time operating system for resource-constrained embedded devices. It is part of the Linux foundation, having multiple contributors. With its Kconfig, Makefile, and defconfig configuration, the Linux build system provides a portable operating system, supporting devices and boards across multiple architectures. As listed by the Zephyr documentation [52] this is including but is not limited to arm-cortex-based Arduinos, EFM32s from Silicon Labs and nRF-devices from Nordic Semiconductor.

Kconfig is a configuration and build system developed for the Linux kernel as it mitigated to Git. Kconfig provides configuration options for the developer without having to change any source code. The Kconfig configuration is converted to a header file (.h) that can be tested and included at build-time.

Device Trees are hierarchical structures used to describe a device's hardware. The device tree files describe hardware available on the current board and default values of configuration parameters. The device tree consists of two types of files; source files and bindings, where source files contain the device tree itself, while bindings describe the content of the device tree, for example, data types. Figure 2.2 shows an example of a device tree source file while figure 2.3 shows the cSLIM bindings for the non-secure partition.

```
/dts-v1/;

/ {
        leds {
                led_yellow: led_1 {
                    gpios = <&gpio0 3  (GPIO_ACTIVE_LOW)>;
                                label = "Yellow LED";
                };
        };
};
```

Figure 2.2: Device tree source example

```
identifier: cSLIMns
name: cSLIM-Non-Secure
type: mcu
arch: arm
toolchain:
  - gnuarmemb
  - xtools
  - zephyr
ram: 128
flash: 192
supported:
  - i2c
  - pwm
  - watchdog
```

Figure 2.3: cSLIM non-secure device tree bindings

Device tree configurations are overruled by .overlay files.

In addition to hardware support, the real-time operating system has a kernel providing, e.g., threading, interrupt, timers and scheduling, operating system services including debugging, logging, peripheral drivers, networking and a watchdog, and application services.

More information of the Zephyr RTOS is available in the project documentation [54].

2.2.2 Server software

The IoF server software consists of a frontend and backend written in Python. The backend receives MQTT messages from the SLIM gateways, unpacking the messages and storing them in a database. In the newest version, the messages from the gateway is packed in JSON format. The IoF server software has not been used in this thesis. However, it is included as being part of the IoF project. The reader is therefore referred to Rundhovde's [33] and Kjelsvik's Master Thesis [29] for further information.

2.2.3 DUNE: Unified Navigation Environment

The DUNE project Github [10] describes DUNE: Unified Navigation Environment as a "runtime environment for unmanned systems on-board software. It is used to write generic embedded software at the heart of the system, e.g., code or control, navigation, communication, sensor, and actuator access. In addition, it provides an operating-system and architecture-independent platform abstraction layer, written in C++, enhancing portability among different CPU architectures and operating systems".

DUNE is divided into tasks communicating with each other through IMC(Inter Module Connect)-messages. The IMC messages are provided in XML-format, which can be ported to C++ bindings. Moreover, IMC messages can be exchanged between vehicles and systems on the DUNE network using numerous Transports, e.g., UDP, TCP, Serial, GSM, Iridium, and more.

2.3 Communication technology and protocols

2.3.1 Fish tag acoustic protocols

As most transmitter manufacturers use their schemes, there are multiple acoustic protocols available. For example, some support over one million unique tags IDs, others support 256. Also, some active fish tags collect extra sensor data that is encoded in the message. Some examples of existing protocols are "R64K", "R256", "S64K", and "DS256". Lately, newer open protocols coding (OPC) have been developed by Thelma Biotel and more. The OPCs aim is to allow "transmitters and receivers from separate manufacturers to work together to obtain the best and most relevant data"; however, older protocols are still used.

2.3.2 IoF message formats

The IoF message format is developed for transferring acoustic tag detections and status messages through LPWAN's in a cost-effective way. The format is divided into a header and payload. The header is specifying the acoustic receiver ID, message content and UTC timestamp for the first message in the payload. The payload contains one or more acoustic detection and acoustic receiver sensor data frames or the buoy controller status frame. Figure 2.4 presents the IoF message frame with header fields, while figure 2.5 and 2.6 presents the tag detection and acoustic sensor payload, respectively. Multiple tag detection and sensor data frames can be concatenated in a single IoF payload. Figure 2.7 presents the buoy controller status payload.

0	2	4	6	8	10	12	14
TBR serial number [0,16383]							Header flags [0,3]
Reference timestamp(UTC) [0,4294967296]							
Reference timestamp(UTC)							
Payload							

Figure 2.4: IoF message frame. Header flag is zero if the message contains tag detections or TBR sensor data frames. Header flag is 1 if message contain the buoy controllers' status. Values of 2 and 3 are left for future message formats.

0	2	4	6	8	10	12	14
Seconds since reference timestamp [0,255]				Code type [0,254]			
Tag ID (all protocols)				Tag ID (protocol R04K, R64K, R01M, S64K, HS256, DS256) / Tag payload (protocol S256) / Absent(protocol R256)			
Tag ID (protocol R01M) / Tag payload (protocol S64K, HS256, DS256) / Absent (protocol R256, R04K, R64K, S256)				SNR [0,60]		Milliseconds	
Milliseconds [0,999]							

Figure 2.5: IoF message tag detection frame. Field contents and the message size is dependent on the code type defined in the second byte. The total size of the tag detection payload is five to seven bytes (if absent bytes, following bytes are left shifted successively).

2.3.3 Transmission from sea buoy to the mainland (LPWAN)

In order to forward acoustic tag detections and send information about the monitoring buoys' status and position to the mainland, the buoy must be equipped with some communication device. As the buoys should be easy to deploy and sustain long battery life, a Low Power Wide Area Network (LPWAN) technology is appropriate. There are multiple technologies available, with NB-IoT and LoRa being two that has shown momentum and probably will have a large share of the market in the coming years, according to Pelaez [30]. The protocols, being similar in some areas, has differences in technology and commercializing approaches.

2.3.3.1 Communication technology basics

The following subsection describes some communication technology terms being used in this thesis.

Receiver sensitivity is the amount of power required to demodulate a signal. Higher receiver sensitivity will increase the range of the wireless link and increase robustness.

Energy per bit (Eb) is the amount of energy required to send a single bit.

Thermal noise spectral density (No) is the noise power per unit of bandwidth and is expressed as power over frequency (watt per Hz)

Eb/No ratio is a term used to describe the link-performance at a given data rate. The better the Eb/No ratio, the better receiver sensitivity at a given data rate [17].

Spectral Efficiency is the amount of data that can be transmitted in a single link. It can be defined as

$$\eta \approx \frac{R/BW}{K}$$

Where R is the bit rate, BW is the bandwidth, and K is the cluster size.[49].

The Shannon-Hartley theorem describes the maximum error-free digital data that can be transmitted over a communication channel with specified bandwidth and the presence of noise.

0	2	4	6	8	10	12	14
Seconds since reference timestamp [0,255]				Code type [255]			
Temperature [0,65535]							
Noise average [0,255]				Noise peak [0,255]			
Frequency (kHz) [63,77]				Upper timing error [0,255]			

Figure 2.6: IoF message TBR status frame. The code type is fixed to 255 to indicate a TBR sensor message.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Battery Status [0,255]							Air temperature [0, 127]							Lon..	
Longitude [0,67108863]															
...Longitude								PDOP [0,127]							Lat..
Latitude [0, 33554431]															
...Latitude								Fix[0,7]			Number of tracked satellite [0, 31]				

Figure 2.7: IoF Message buoy controller status frame

With a bandwidth, BW (Hz), white Gaussian noise of power N (watts), and a received signal of power S (watts), the information rate, I (bits/sec), is given in equation 1.

$$I = BWlog_2(1 + \frac{S}{N}) \tag{1}$$

2.3.3.2 Energy efficiency metrics

When measuring energy efficiency in IoT applications, Santiago and Dr. Arockiam [36] presents the following most common aspects :

1. **Energy per bit**
2. **Energy per reported event**, i.e., the energy spent to report one single event.
3. **Delay/energy tradeoff**, i.e., the importance of reporting events in (near) real-time compared to the energy usage
4. **Network lifetime**, i.e., the time the device is able to fulfill its task. This includes both the lifetime of the device itself as well as the infrastructure needed to fulfill the task, e.g. network infrastructure or servers.

Zanaj et.al. [51] presents the additional aspects:

1. **Battery Lifetime**, i.e., how long the device can live without replacing the battery.
2. **Duty Cycle**, i.e., the time of which the node is active on the network (either transmitting or listening) compared to the overall lifetime.

2.3.3.3 Issues in IoT energy conservation

Santiago and Dr. Arockiam [36] summarizes the issues related to energy efficiency in IoT in the following five key points:

- **Idle listening** is the state where a node (end device) is awaiting ready to transmit or receive data while not doing any of those above.

- **Collision** occurs when the node receives multiple transmissions at the same time. As the transmissions interfere with each other, none of the received data can be used.
- **Overhearing** occurs in dense sensor networks where neighboring nodes receive and process information that is of no use to the node.
- **Protocol overhead** results in extra energy usage. Reduction of protocol header information will reduce the amount of data transmitted and processed and reduce energy consumption.
- **Traffic fluctuation** occurs when multiple nodes are active at the same time, resulting in congestion or long delays.

2.3.3.4 NB-IoT

Narrowband Internet of Things, in short, NB-IoT, is a cellular wireless technology standard introduced in release 13 of the 3GPP project [1], introducing low power wide area network (LPWAN) to the cellular network standard. The protocol offers direct communication between the low-power end-device and a TCP/IP connected server, or another Internet device, through base stations and infrastructure provided by cellular network operators. The network operators take a fee per device connected to the network or the amount of data transferred. According to Le Bras [13] the maximum payload size for NB-IoT is 1600 Bytes, while the peak data rate is 66Kbps.

NB-IoT Power Saving features
Increased battery lifetime operations are one of the key features of LPWANs. While not utilizing power-saving features, a device connected to the network is idle listening by paging the network. In other words, the device is listening at a short periodic interval decided by the network operator. NB-IoT support two essential power-saving features: Extended Discontinuous Reception (eDRX) and Power Saving Mode (PSM).

eDRX
eDRX is an extension to the regular DRX (discontinuous reception) that is used by other LTE networks. eDRX allows the device to be unreachable for a longer time (5 to 2621 seconds) before listening to the network. A device in eDRX mode is still quite receptacle to data while limiting the energy consumption.

PSM
In PSM mode, the device can set timers that are forwarded to the network: Periodic Tracking Area Update (TAU) and Active time. This contains information on the sleep and activation time of the device. If the network accepts the configuration, the device can sleep for a given time (up to 14 days) before reattaching is necessary. During the sleep interval, the device is not able to receive packets. However, based on the timer values, the network knows when the device will be reachable and available for receiving packets. Hilal et al. [4] state the PSM mode is introducing a large down-link delay of packets, dependent on the device sleep time.

2.3.3.5 NB-IoT networks

As part of the 3GPP release 13 in 2015 [1] NB-IoT is a quite new protocol, and cellular network operators worldwide are currently working on facilitating the protocol. Figure 2.8 shows the worldwide coverage of Mobile IoT networks. Although some areas might support NB-IoT, all of its features as PSM and eDRX might not be fully supported. Figure 2.9 shows the NB-IoT coverage provided by Telenor in Norway, while figure 2.10 shows the coverage provided by Telia.

Coverage Enhancement Level
The coverage enhancement level (CE-level), also referred to as the extended coverage level (ECL), is included to provide reliable communication to devices in harsh environments. Each level has its radio profile, including transmission parameters as transmission power, a subset of sub-carriers, number of repetitions, and maximum number of transmission attempts. Deliang et al. [50] and Foivos et al. [21] presents the three ECL-levels:

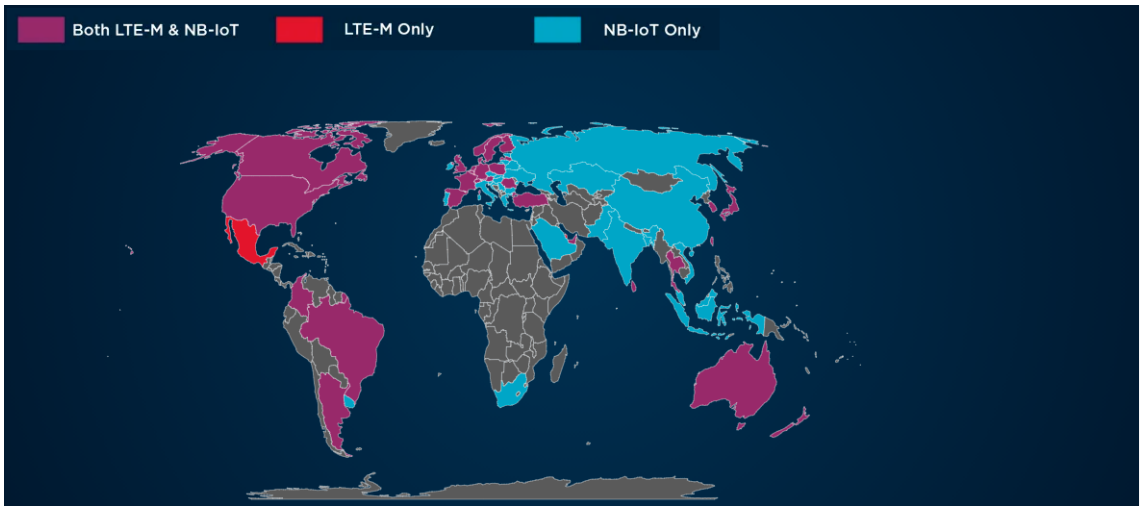


Figure 2.8: Coverage of LTE-M and NB-IoT networks worldwide. Figure by GSMA [12]

- ECL 0: For operation in good conditions, typically at distances shorter than 0.7 km.
- ECL 1: For robust operation in obstructive environments, as indoor or at longer distances.
- ECL 2: For operation in extreme conditions. Here the number of repetitions might be as many as 2048 with a transmission delay of 10 s.

The network operators decide the threshold for which ECL to use.

2.3.3.6 LoRa

The term LoRa (Long Range) specifies a physical layer standard owned by Semtech, using Chirp Spread Spectrum (CSS) modulation to transmit data over large distances. CSS symbols have four important parameters: Spreading Factor (SF), the minimal and maximal frequency (f_{min}, f_{max}) in the frequency band and the starting frequency. The starting frequency (f_0), $f_{min} <= f_0 <= f_{max}$ is calculated based on the symbol being sent. The raw chirp, $f_c(t)$ can be described as in equation 2.

$$f_c(t) = \pm\frac{BW}{T}t \tag{2}$$

BW is the bandwidth of the ISM band ($f_{max} - f_{min}$), t is the time, and T is the symbol period given in equation 3.

$$T = \frac{2^{SF}}{BW} \tag{3}$$

$SF \in [7, 12]$ is the spreading factor of the chirp, indicating how fast the frequency is changing within the bandwidth.

The starting frequency is added to the raw chirp frequency, resulting in the frequency given in equation 4.

$$f(t) = f_c(t) + f_0 \tag{4}$$

As the ISM band limits the output frequency, the output signal chirps over all frequencies, starting at f_0 to f_{max} before continuing from f_{min} to f_0. Thus, the final signal frequency is shown in equation (5).

$$s(t) = e^{i2\pi f(t)t} \tag{5}$$

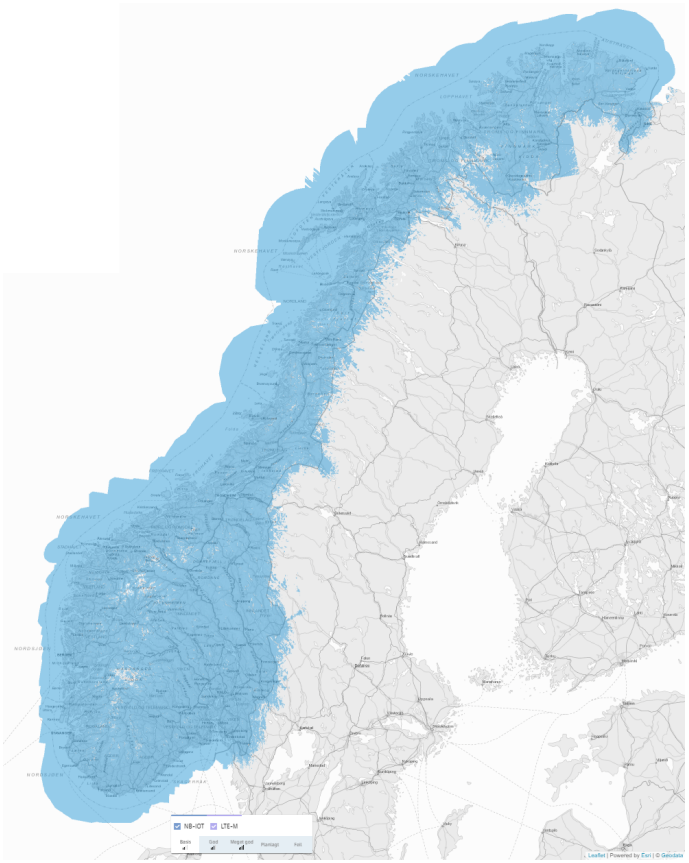


Figure 2.9: Coverage of NB-IoT in Norway by Telenor. White areas are areas without coverage. Figure by Telenor [40]

Figure 2.12 shows the transmit time of a symbol for different spreading factors.

Zanaj et al. [51] state that the bit rate of a LoRa transmission is given by equation 6.

$$R_{LoRa} = SF \frac{\frac{4}{4+CR}}{\frac{2^{SF}}{BW}} 1000 \tag{6}$$

CR is the coding rate of the transmission, i.e., the number of transmitted bits that carry information. The rest of the transmitted bits are used for forward error correction. Typical coding rates are 4/5, 4/6, 4/7 and 4/8.

According to Marco [6] the spectral efficiency of LoRa is dependent on its spreading factor. The spectral efficiency with respect to spreading factor, code rate and cluster size is given in equation 7:

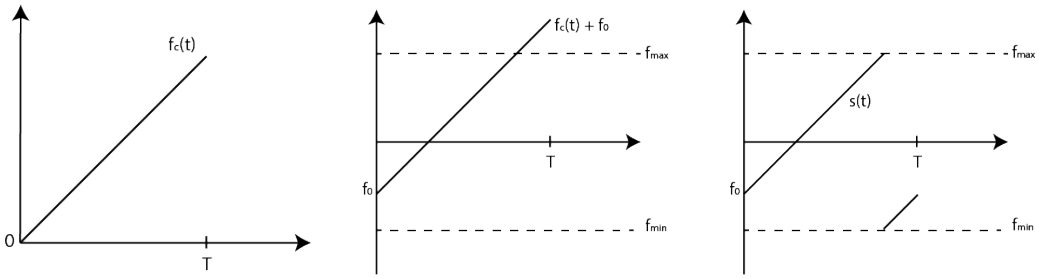
$$\eta_{LoRa} \approx \frac{R/BW}{K} = \frac{SF \frac{\frac{4}{4+CR}}{\frac{2^{SF}}{BW}} 1000}{BW * K} = \frac{\frac{SF}{2^{SF}} \frac{4}{4+CR} 1000}{K} \tag{7}$$

The modulation technology is further described by Rundhovde [33], Zanaj et.al. [51] and Reynders et.al.[32].

2.3.3.7 LoRaWAN

A LoRaWAN is a Wide Area Network based on the LoRa modulation technique. LoRaWAN includes a scalable bandwidth of 125kHz, 250kHz or 500kHz. Based on this and the Signal to Noise

Figure 2.10: Coverage of NB-IoT in Norway by Telia. White areas are areas without coverage. Figure by Telia [41]



(a) Raw Chirp

(b) Raw chirp with starting frequency

(c) Final CSS signal

Figure 2.11: Chirp Spread Spectrum modulation

Ratio (SNR), a variable spreading factor is selected to make it less affected by noise. LoRaWAN operates within the ISM bands, including 169MHz, 433 MHz, 868MHz (Europe) and 915 MHz (North America).

ABI Research [31] lists some key features of NB-IoT and LoRaWAN. These are given in table 2.1

Figure 2.13 shows the content of a LoRa MAC frame.

Figure 2.14 shows the LoRa gateways registered to The Things Network in the southern part of Norway. As stated in The Things Network documentation [45] devices on their network can be active transmitting 30 seconds per day and are limited to 10 downlink messages per 24 hours. In Norway, other LoRaWAN operators are also available, as Altibox and Last_Mile. Unfortunately, no coverage map is found. However, Altibox' solution [2] is referred to as a "smart-city" solution.

A device connected to a LoRaWAN can be registered in one of the following device classes:

- Class A: Enable uplink (to the server) and downlink (to the end device) transmissions. However, a downlink transmission can only occur after an uplink transmission. The end

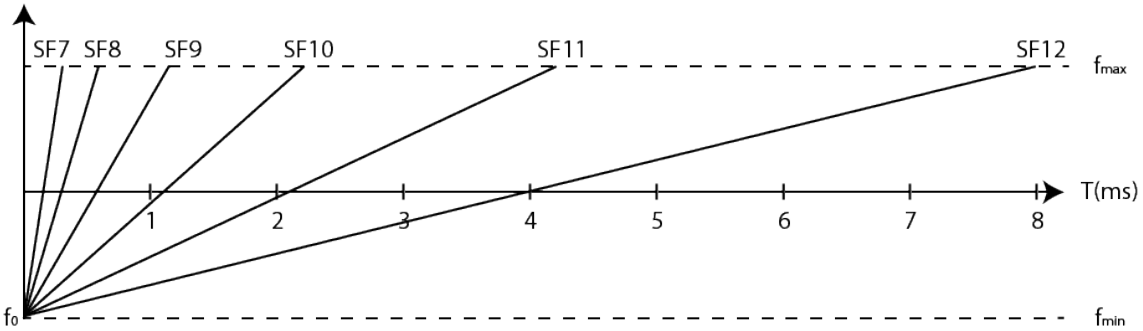


Figure 2.12: Spreading factor transmit time
for a signal transmitted using a bandwidth of 500 kHz. For simplicity, the starting frequency f_0 equals the lower band frequency f_{min}.

Parameter	NB-IoT	LoRaWAN
Bandwidth	180kHz	125kHz
Coverage	164 dB	165dB
Estimated Battery Life	10+ years	15+ years
Throughput	60 Kbps	50Kbps
Latency	¡10s	Device Class dependent
Security	3GPP (128 to 256 bit)	AES 128 bit
Geolocation	Yes (3GPP rel 14)	Yes (TDoA)
Cost efficiency	Medium	High

Table 2.1: Key features of NB-IoT and LoRaWAN. Source: ABI Research [31]

device listens to the network in two short receive windows after the uplink transmission. If no message is received, the end device goes offline until the next uplink transmission.

- Class B: Extend class A devices by adding scheduled receive windows. The device then listens to the network for downlink transmissions at regular intervals.
- Class C: Extend class A devices by listening to the network when not transmitting. In other words, the end device is always reachable unless performing an uplink transmission.

2.3.3.8 MQTT

Message Queueing Telemetry Transport is a lightweight publish/subscribe messaging transport protocol. The protocol is developed to connect remote devices with a small code footprint and limited network bandwidth, support bidirectional communication, and contain mechanisms for unreliable networks. This includes messages delivered at least once, delivered exactly once and a "fire and forget" configuration where messages are transmitted without fault detection. MQTT product pages [23] state the protocol is used in a wide variety of industries, such as automotive, manufacturing, telecommunications, oil and gas. MQTT is built on top of TCP, giving security through SSL/TLS-encryption. Using MQTT, the device will typically retain a connection to the broker at all times. However, on a sensor network (MQTT-SN), this is not required [24].

The MQTT packet format consists of three main parts: The fixed header, variable header and the payload. The MQTT packet format is shown in 2.15

0	8	16	24	32	40	48	56
Mac Header	Device Address				Frame Ctrl	Frame Count	
Frame Options([0-15] bytes)							
Frame Port	Frame Payload ([0-51/115/222] bytes in EU868, dependent on spreading factor and frequency)						
Frame Paylaod				MIC (4 bytes)			

Figure 2.13: LoRa MAC frame. The maximum allowed number of bytes in the payload is dependent on the spreading factor and transmission frequency, and is ranging from 51 to 222 bytes in EU868
.

0	1	2	3	4	5	6	7
Message Type				DUP	QoS		RETAIN
Remaining Length							
Variable header							
Payload (max 256 MB)							

Figure 2.15: MQTT frame. The variable header includes extra message information as protocol name, protocol level and message topic in that UTF is adjustable in size. The maximum payload is 260MB
.

The fixed header, required for all MQTT packets, consists of the following fields:

- Message type field indicating the package type, e.g. a connect request for setting up the connection to a broker, a subscribe message for subscribing to a topic or a publish message to send data to the broker.
- Message flags containing specific indicators for each message type. For a publish message, the four bits are the following[5]:
    1. Duplication flag (1-bit) indicating whether the package might have been previously received.
    2. Quality of Service (2-bit), selecting one of three QoS: At-least-once, at-most-once or exactly-once.
    3. Retain flag (1-bit) indicating if the broker should keep the message stored for future subscribers.
- Remaining length

As MQTT is a TCP based protocol, the message transmitted is a TCP package containing the MQTT message as payload. Figure 2.16 shows the content of the TCP frame.

0			16
Source Port			Destination Port
Sequence Number			
Acknowledgment Number			
Data Offset	Reserved	Flags	Sliding Window
Checksum			Urgent Pointer
Options (0-40 bytes)			Padding (0-4 bytes)
Data			

Figure 2.16: TCP frame

.

2.3.3.9 CoAP

CoAp (Constrained Application Protocol) is, according to its product pages, [7] a "specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation." Using UDP, bitfields and mapping from strings to integers, CoAp packets are small, much smaller than e.g., HTTP TCP flows. CoAp support both acknowledged packets as well as "fire and forget" transmissions [24]. CoAp normally runs over UDP. The UDP frame format is shown in figure 2.18

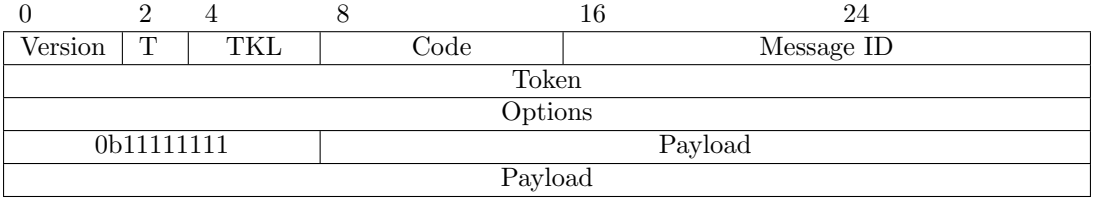


Figure 2.17: CoAp frame

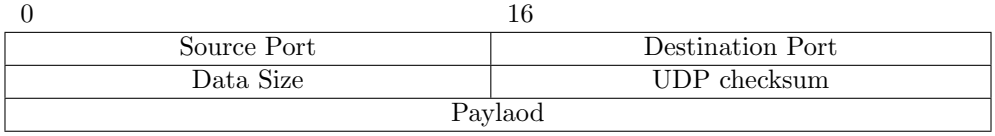


Figure 2.18: UDP frame

2.4 Development Environment

2.4.1 KiCad - Schematics and PCB design

KiCad is open-source software for electronic computer-aided design (ECAD). The program support schematic drawings and PCB layout design. The program consists of five main parts:

- KiCad project manager, used to manage the project and start sub-applications.
- KiCad Eeschema, used to create circuit schematics.
- KiCad PCBnew, used for PCB layout design.
- KiCad GerbView, used to view Gerber and drill files, i.e. PCB production files.
- KiCad Bitmap2Component, used for converting images to PCB footprints.

2.4.2 LibraryLoader

LibraryLoader is a small application for importing schematic and PCB footprints of components to KiCad libraries. While running, the components are added automatically on download.

2.4.3 FreeRouting

FreeRouting is an opensource routing tool for PCBs. The software is allowing different track widths and track-spacing, e.g., for power wires and ground wires. FreeRouting does not support differential pair routing and tuning or similar features.

2.4.4 VS Code - Code editor

Visual Studio Code is a Microsoft code editor available for Windows, macOS and Linux. Allowing open-source plugins, it has many extension available for aiding the development of source code in many languages, including syntax highlighting, themes, debuggers and connecting to online services as Git and other SCM repositories.

2.4.5 nRF Connect and the nRF Connect SDK

nRF Connect is a tool developed by Nordic Semiconductor to facilitate the programming, testing, evaluation and debugging of their nRF-series devices and the use of tools such as the power profiler, modem trace collector and LTE link monitor. The toolchain manager is used for downloading and installing the nRF Connect SDK.

The nRF Connect SDK is a software development kit for building applications for the nRF-series devices. It integrates the Zephyr RTOS together with hardware drivers, libraries and examples of use. The SDK source code is publicly available on GitHub through a series of Git repositories. The toolchain, therefore, uses West for managing the Git repositories and the Ninja build system for configuring and building the applications.

2.4.6 Microchip studio

Microchip Studio is a code editor and development environment by Microchip. It contains tools and examples for programming their AVR and SAM micro-controllers and includes data visualizing tools for the Microchip Power Debugger.

2.4.7 Git

Git is a version control system for tracking changes of source files and allowing multiple people to work on the same project. In addition, Git has features for synchronizing and merging local source code with a code base, often at a web host.

2.4.8 LSTS toolchain

The LSTS Toolchain, provided by LSTS Underwater Systems and Technology Laboratory in Portugal, is a virtual Linux machine image containing all necessary software to develop and run tasks in the DUNE environment.

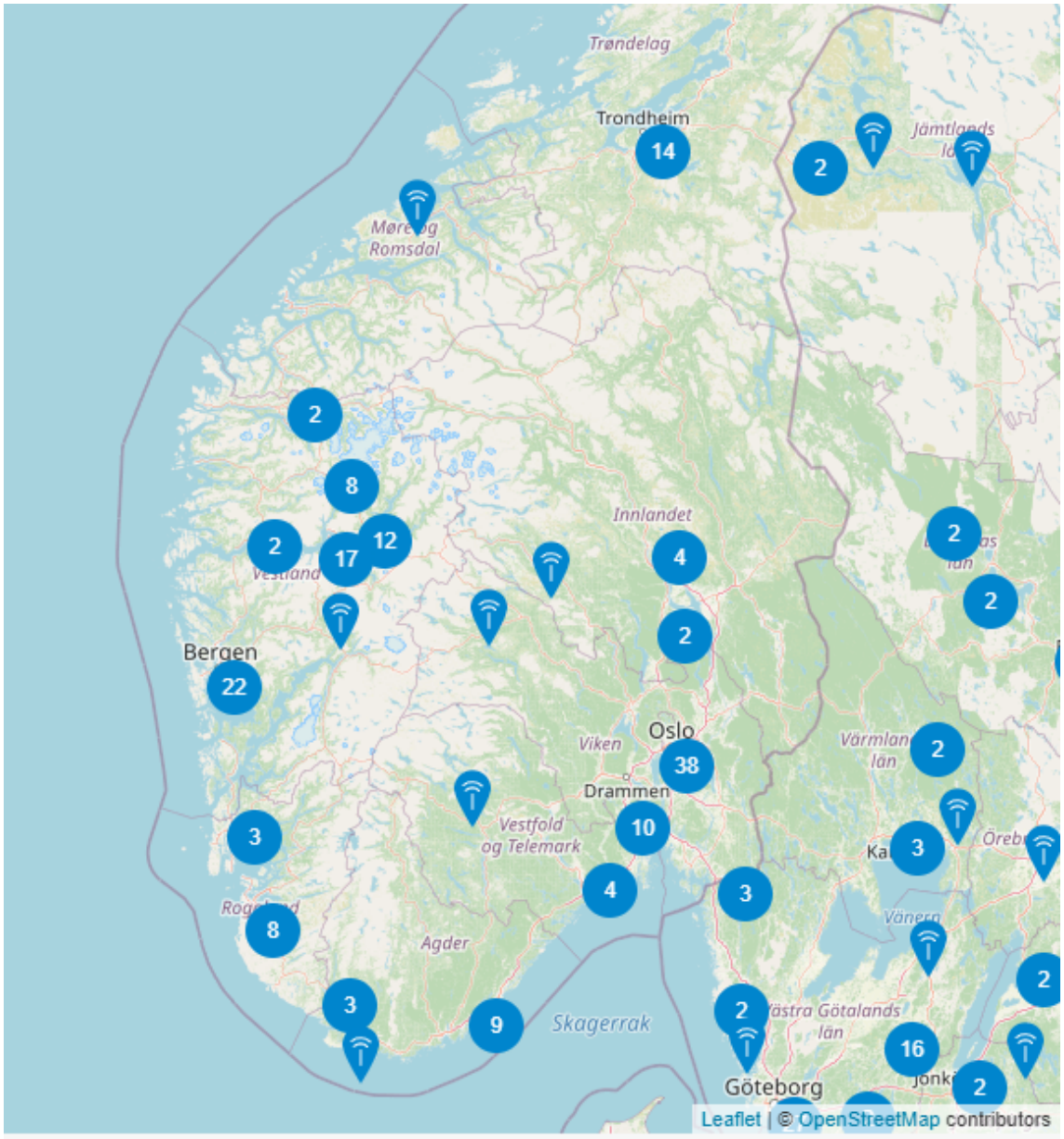


Figure 2.14: The Things Network LoRa gateways in south of Norway. Further north gateways are limited to the cities of Bodø, Tromsø and Alta. Figure by The Things Network[42].

3 System Requirements

It was decided to build a prototype based on the nRF9160-DK to explore the NB-IoT and LoRaWAN limitations and capabilities. Then, if the results are promising, this can evolve to a standalone module without the development kit.

The first month of the project was used to inspect the previously built SLIM buoy controller and read up on older reports. Previous work gave insight into the project, how the SLIM module was built and the new system requirements. In addition, earlier project reports helped locate problems with the previous system, reducing pitfalls when building new ones. The Master Thesis by Rundhovde [33] is especially useful as it is recent and thus contains up-to-date problems.

A list of requirements was developed for the system based on the project description and older reports. This chapter presents the key points of the requirements, while the complete list is given in Appendix B. Section 3.1 gives insight into the requirements for the Cellular SLIM (cSLIM) prototype, including the nRF9160-DK and hardware shield. Section 3.2 gives insight into the extra requirements for a standalone module not using the development kit, while section 3.3 gives insight into LPWAN requirements. At last, section 3.4 gives insight into the requirements for the DUNE bridge.

3.1 Buoy prototype requirements

The buoy consists of an acoustic telemetry receiver (ATR) and a buoy controller. The ATR should be able to detect acoustic signals from tags with high reliability, range and timing precision, process the signals and forward the detections to the buoy controller. To synchronize the time with other buoys, the communication with the buoy controller must be duplex, enabling data to be sent to and received by the acoustic telemetry receiver. As the distance between the controller module and ATR can be several meters, it is preferable with a robust communication interface for such distances. In addition, the ATR must also be able to detect, process and transmit sensor data like sea temperature. For the IoF project, two prominent use cases apply: Standalone use and bio-telemetry use. Some requirements are valid for both use cases, while others are unique for the two. At first all-use-case requirements are presented before presenting special requirements.

3.1.1 All-use-case requirements

An essential part of the system is regarding the fish tag observations. The buoy controller must communicate with the acoustic telemetry receiver and forward its tag detections and sensor data to the mainland. Furthermore, to synchronize the time of the ATR with other receivers, the controller must obtain a precise local time and make sure the acoustic receiver is synchronized appropriately.

A fish tag observation is of little use without knowing where the observation took place. This requires the buoy controller to obtain its position based on existing GNSS satellite networks as GPS, GLONASS, Galileo, or similar. The system should also keep operating without a constant position fix to facilitate periods without GNSS signal coverage and to reduce battery usage by powering off the GNSS module or putting it in sleep mode. Based on observations by Rundhovde [33], the GNSS module is one of the more energy-consuming parts of the buoy controller.

The buoy controller must transmit its status to the mainland, containing information about battery status, position, metadata (position loss, dilution of precision, number of tracked satellites), timing accuracy, and other relevant parameters. This is used both by the end applications to obtain buoy positions as well as to indicate the certainty of fish positions by TDoA calculations. Also, such information helps evaluate the performance of the system at sea. The buoy controller must set peripherals on standby or in sleep mode when they are not needed to reduce energy usage. This includes keeping operation without waking the peripherals more frequently than necessary. In other words, the system must work even if new data from, e.g. the GNSS module is not available upon request. To transmit the data from the buoy to the mainland in an energy- and cost-efficiently

way, the buoy controllers should use a LPWAN technology to transfer the data. In addition to being power efficient, the LPWAN technology must work at longer distances from the base stations to support deployments away from shore. The LPWAN communication must also follow ITU and local radio regulations to avoid overloading the frequency bands. As one of the goals of this project is to compare the performance, suitability, and cost-effectiveness of LoRa and NB-IoT, both LPWAN technologies should be supported by the system.

In addition to transmitting the observations and status of the system to the mainland, the data should be logged locally to support periods of no LPWAN connectivity, as well as enabling extra logging of events. Furthermore, logging system status and errors locally are preferable for debugging and further improvements while keeping a low wireless link budget. In order to ease deployment, the system should convey its state and relevant information to the user without debugging or serial communication equipment. The system state should also be observable at a distance.

To reduce maintenance costs, it is preferable that the system can be deployed at prolonged periods, ideally up to seven months, to keep up with the lifetime of the acoustic telemetry receivers. However, during months of deployment, errors will most likely occur at some point in time. Therefore, the system needs to detect these errors, recover, and preferably forward the status to the user.

3.1.2 Use case dependent requirements

There are mainly two use-cases applicable to the system: Single-buoy and multi-buoy deployment. The standalone use case, illustrated in figure 3.1, is based on placing a single monitoring buoy at sea to keep track of fish-mitigation of longer distances. As the fish moves across larger areas, e.g., from the fjord and upstream a river to spawn before returning to sea, the exact position is not of the highest interest. This partly removes the timing requirements of the system. However, the acoustic telemetry receiver should still be reasonably accurate. Therefore, a maximum drift of 500ms is set for the application. The other use case, shown in figure 3.2, is based on tag-observation being used for bio-telemetry, for instance, to monitor fish behavioral within fish farms. This requires multiple buoys, at least four deployed at a relatively small area, to get a 3-dimensional position. The acoustic receivers must be time-synchronized in order to calculate the position accurately. As the smallest resolution of ATRs as the TBR 700 RT is 1 ms, the time error should be minor to half of that, 500us, compared to the actual GNSS time. This ensures the ATR to be at the correct millisecond at all times.

3.1.3 Shield requirements

The shield must contain all components and connectors of the cSLIM module not being part of the nRF9160-DK. Shield requirements are further described in section 5.1.

3.2 Standalone buoy module requirements

This subsection describes the additional requirements for the standalone buoy controller. This has not been developed, and the requirements are given for future work.

The standalone module is expected to consist of a single PCB with all necessary components from the nRF9160-DK as well as the cSLIM shield, as described in section 3.1. The extra components needed include an nRF9160 with required circuitry as capacitors and resistors, a voltage regulator, reset button, antenna matching networks for the nRF9160 cellular antenna, and preferably matching networks for the nRF9160 GNSS antenna, if this is to be used. A SIM or eSIM is also required to register the device on an NB-IoT network.

The standalone module must also include connectors for the external antennas, power supply (battery) and an SWD connector for programming the nRF9160.

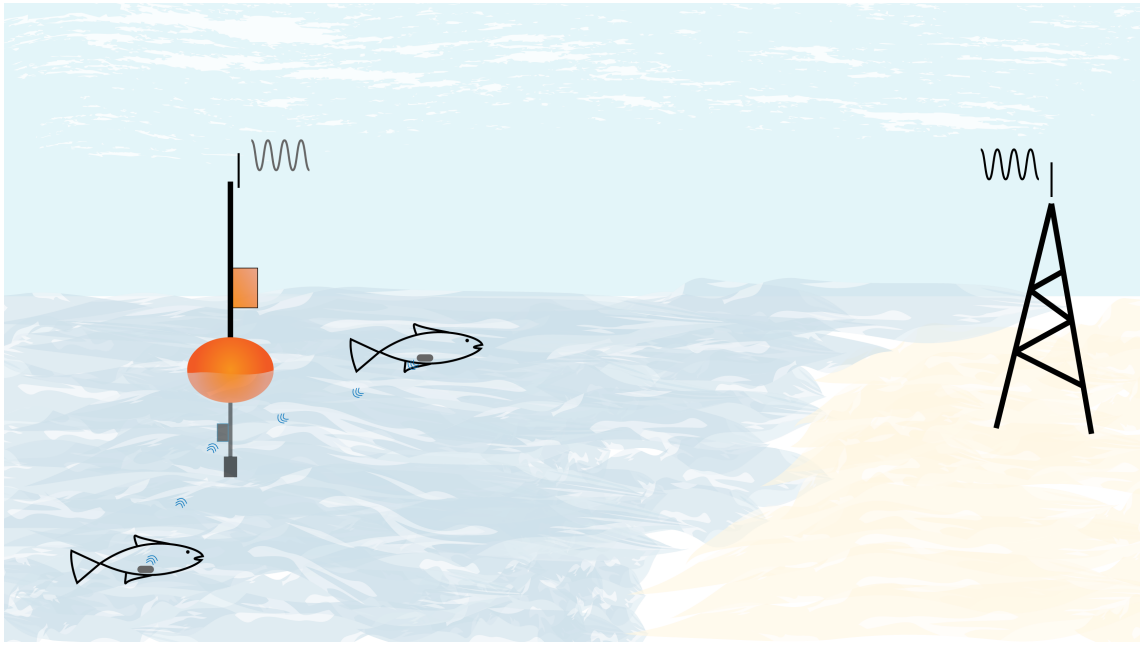


Figure 3.1: Single buoy deployment. In a single deployment scenario, the buoy is placed by itself in a river, fjord or ocean. This allows for tracking if the fish is close to the buoy. By deploying multiple buoys across larger areas, fish travel can be researched. Base stations on the mainland are receiving LPWAN messages from the buoys and forward the messages to an MQTT broker and end applications. Figure: Eivind Jølsgard/NTNU (CC BY 4.0)

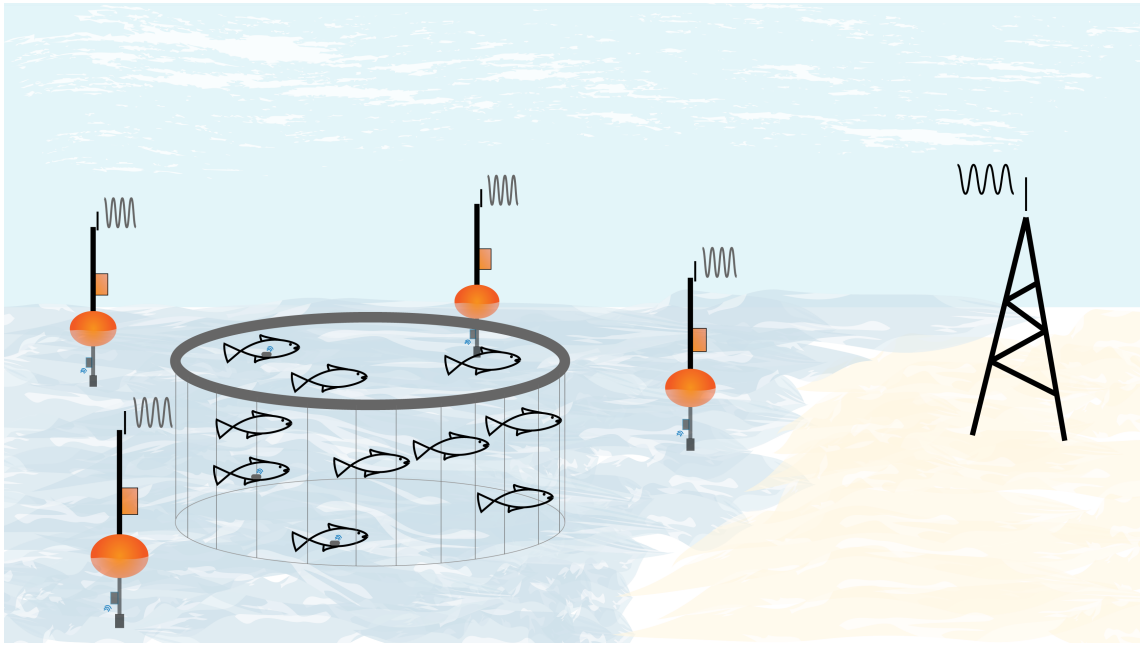


Figure 3.2: Multi-buoy deployment. In a multi deployment scenario, multiple buoys are receiving and forwarding the acoustic detections. With synchronized receivers, this allows for bio-telemetry applications. Base stations on the mainland are receiving LPWAN messages from the buoys and forward the messages to an MQTT broker and end applications. Figure: Eivind Jølsgard/NTNU (CC BY 4.0)

3.3 LPWAN requirements

An LPWAN is required to transfer the data from the acoustic telemetry receiver to the mainland. The LPWAN should allow the data to be transmitted cost-effectively, both regarding low power consumption and economic costs. All required infrastructure for the LPWAN should be easy to deploy if not already provided by network operators. With regards to data usage, the LPWAN should support data-transfers of an appropriate number of buoy controllers and fish-tag detections for the use cases described in 3.1.

To support deployment in remote environments and away from the mainland, the LPWAN should uphold a link with several kilometers between the buoy and other infrastructure. Also, the communication must follow ITU and local radio regulations.

3.4 DUNE bridge requirements

The DUNE bridge should receive the fish-tag and sensor data from the buoy controllers and forward this information to the DUNE framework. N. Lauvås at ITK, NTNU already define a IMC::FishTag and IMC::TBRSensor message format. It is, therefore, preferable to use these formats. The DUNE bridge should be easy to set up in an already defined DUNE framework and require minimal configuration.

4 System overview

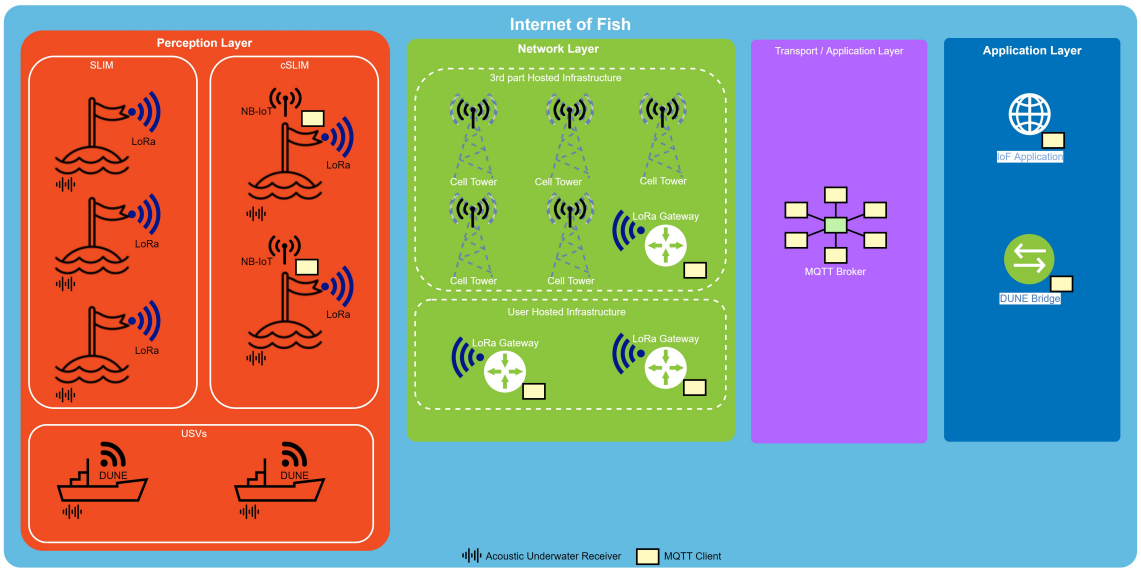


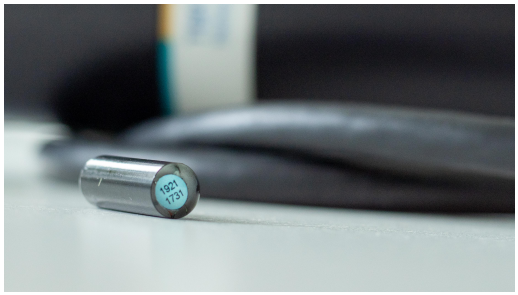
Figure 4.1: IoF overview illustrating the main components of the system and infrastructure. This enables the buoy to transmit its status and observations to the applications (DUNE bridge and the IoF application). Figure: Eivind Jølsgard/NTNU (CC BY 4.0)

This section gives an overview of the system in total, its components, and communication between the components. The main system components consist of the acoustic telemetry receiver detecting the acoustic tag transmissions, the cSLIM buoy controller synchronizing the acoustic receivers, and forwarding the data through the LoRa gateway or NB-IoT network to an MQTT broker. The MQTT broker forwards the published messages to subscribed applications as the IoF application and DUNE bridge. The following subsections describe the system components further.

4.1 Acoustic tags and telemetry receiver

The acoustic telemetry receiver used in this project is a TBR 700 RT from Thelma BioTel, receiving acoustic signals from a cylinder-formed tag with a diameter of 9 mm and a height of 2.9 cm. Other tags are also supported. The acoustic receiver and tag is shown in figure 4.2.

(a) Acoustic telemetry receiver



(b) Acoustic tag

Figure 4.2: Acoustic telemetry receiver and tag. The acoustic telemetry receiver is a TBR 700 RT from Thelma BioTel. The picture also shows the interface cable used to communicate with the buoy controller using the RS-485 interface. To the right is an acoustic tag from Thelma BioTel with a diameter of 9 mm. Photo: Eivind Jølsgard/NTNU (CC BY 4.0)

The TBR 700 RT has a battery with a lifetime of up to seven months. It communicates through an

RS485 interface using a differential pair of signal wires and can send or receive one character every millisecond. The acoustic sensor uses its internal clock and time to timestamp tag detections. The time can be set using a predefined message format, synchronizing the last received character. In addition, a shorter message without time data can be received every ten seconds to correct internal clock drift. The interface is further described in the TBR Live datasheet [39].

4.2 cSLIM buoy controller and sea buoy

The cSLIM buoy controller is the heart of the buoy at sea. Its prototype consists of the nRF9160-DK and a custom shield designed specifically for the application. The device synchronizes the time of the TBR acoustic receiver, receives tag detections, and forwards them to the mainland using LPWAN technology. The module also transmits its position, status and time drift at regular intervals. Figure 4.3 shows the buoy controller and acoustic receiver module interfaces. Figure 7.1 shows the final buoy controller prototype.

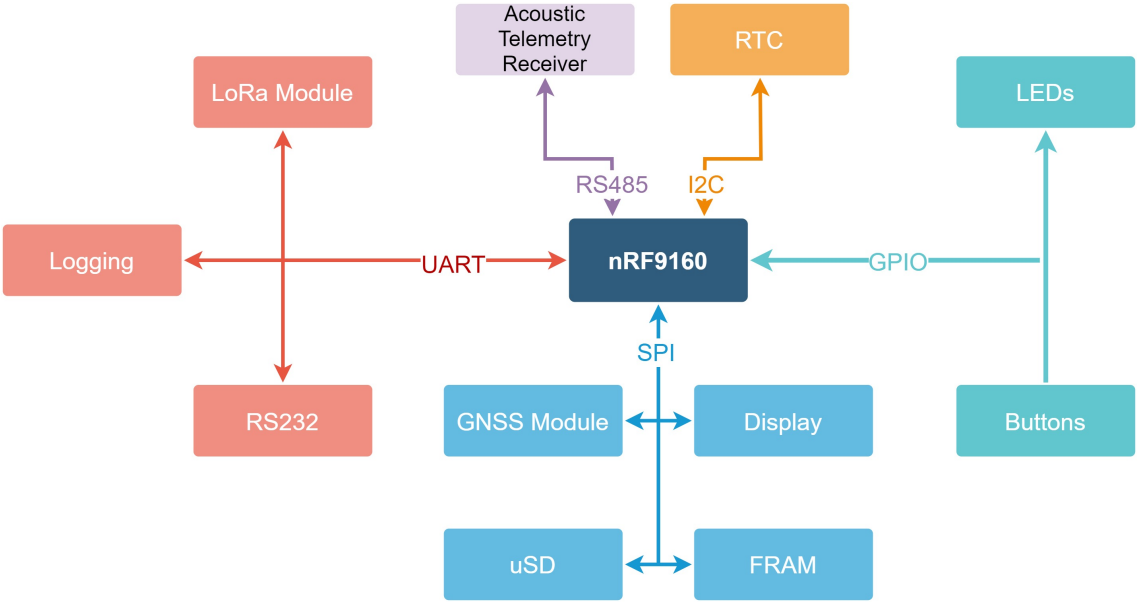


Figure 4.3: cSLIM buoy controller modules and interfaces.

The buoy and its components are shown in figure 4.4. The buoy itself consists of a floating device and a weight to keep it stable. Mounted on the buoy are the acoustic receiver and buoy controller in waterproof casings. In addition, it is possible to mount external antennas on the top of the buoy. No cSLIM device has been used at sea; however, buoys are deployed using SLIM modules.

4.3 LoRa Gateway and MQTT broker

The LoRa gateway used in this project is the MultiTech Conduit. The Conduit comes in both indoor desktop and outdoor (IP67) editions. Figure 4.5 shows the indoor version. The gateway can be connected to Ethernet and to the cellular network, supporting 2G, 3G, and 4G LTE. Running Node-RED; the gateway forwards LoRaWAN payload to the MQTT broker. For the cSLIM application, only the LoRaWAN payload is forwarded to make the messages equal to those transmitted using NB-IoT/MQTT. This is the format used by Kjelsvik [29] for the IoF application. However, it has later been edited by Rundhovde [33], packing the data in JSON objects together with LoRaWAN parameters.

The MQTT broker, running on the virtual machine provided by NTNU, is a mosquitto broker with its access restricted by username and password authentication.

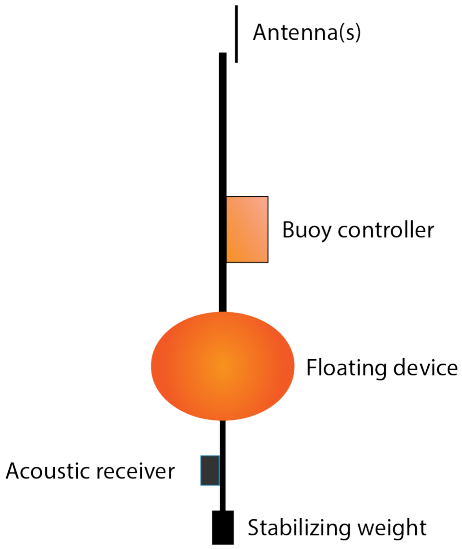


Figure 4.4: IoF sea buoy

4.4 IoF application

The IoF application is connected to the MQTT broker, and subscribed to the buoy controller publish topic. The newest back-end version receives IoF messages packed in JSON objects containing relevant LoRa parameters and saves received information in a database. The database is accessed by the front-end, displaying information to the user. As the newest IoF application expects additional LoRa information, it is currently incompatible with the LoRa Gateways used in the cSLIM project or communication using NB-IoT. However, for applications using LoRa Gateways running the SLIM Node-RED flow by Rundhovde, cSLIM should be supported.

4.5 DUNE bridge

The DUNE bridge, running in a DUNE environment, is subscribed to the MQTT broker and cSLIM publish topic. Upon receiving messages, the application converts the TBR status and tag detections to appropriate IMC messages that are dispatched in DUNE. In addition, information about the positions of the acoustic sensors is gathered using buoy controller status messages.

Figure 4.5: MultiTech Conduit LoRaWAN base station. Photo: Eivind Jølsgard/NTNU (CC BY 4.0)

5 Shield development

The following subsections present the steps of shield design, production and verification, as well as some design considerations taken into account for the cSLIM prototype.

5.1 Interface requirements

Based on the overall cSLIM buoy requirements, the shield should meet the following requirements:

1. The shield must fit on top of the nRF9160 development kit using an Arduino UNO revision 3 layout.
2. The shield must connect the nRF9160 to the peripherals needed for the cSLIM buoy controller. A list of peripherals is given in table 5.1.
3. The display and LEDs must be placed such that they are easy to observe.
4. The buttons must be placed such that they are accessible by the user.

As the shield is a first step for designing a standalone buoy controller based on the nRF9160 SiP, it was decided to include hardware needed for the final PCB. This includes a 3V voltage regulator and a USB connector for power and debugging capabilities. The SWD program/debug connector and SIM are left out as the required pins are not accessible by the Arduino headers.

Peripheral	Interface	Connection Pins
Display	SPI	Display_nCS(P0.22), SPI_MOSI(P0.18), SPI_SCLK(P0.19)
LEDs	GPIO	LED_RED(P0.02), LED_YELLOW(P0.03), LED_GREEN(P0.04), LED_BLUE1(P0.05)
Buttons	GPIO	RESET(nReset), PB1(P0.06), PB2(P0.07)
RS485 Tranceiver	UART	RS485_RX(P0.00), RS485_TX(P0.01), RS485_RE(P0.14), RS485_DE(P0.15)
RS232 Tranceiver	UART	RS232_RX(P0.23), RS232_TX(P0.24), RS232_nShutdown(P0.11)
RTC	I2C	I2C_SDA(P0.30), I2C_SCL(P0.31) RTC_Clockout(P0.26), RTC_nInt(P0.03)
LoRa module	UART/I2C/SPI	UART_RX(P0.08), UART_TX(P0.09), I2C_SDA(P0.30), I2C_SCL(P0.31), LoRa_SPI_nCS(P0.20), SPI_MOSI(P0.18), SPI_MISO(P0.17), SPI_SCLK(P0.19), LoRa_nEnable(P0.05), LoRa_PA06(P0.21)
GPS module	SPI	GPS_nCS(P0.10), SPI_MOSI(P0.18), SPI_MISO(P0.17), SPI_SCLK(P0.19), GPS_nEnable(P0.12),GPS_Timepulse(P0.27)
uSD Card reader	SPI	uSD_nCS(P0.07), SPI_MOSI(P0.18), SPI_MOSI(P0.17), SPI_SCLK(P0.19), uSD_nEnable(P0.06)
FRAM	SPI	SPI_nCS(P0.25), SPI_MOSI(P0.18), SPI_MISO(P0.17), SPI_SCLK(P0.19)
Battery measurement	Analog	Battery_voltage(P0.13)

Table 5.1: cSLIM peripherals with interface types and nRF9160 connected pins.

5.2 PCB design considerations

5.2.1 PCB manufacturing and assembly considerations

To use small signal track widths and have precise pads for component footprints, which make it possible to use smaller components, it was decided to manufacture the PCBs using a third-party producer. The author then performs design, assembly, and testing at the university and at home. As components are mounted on the PCB by hand before being soldered in the reflow oven, using the minor components available is non-ideal. When applying solder paste, further care must be taken so that components mount correctly to the PCB without any short circuits.

PCBs can be manufactured with many layers of copper. The copper layers are connected through small copper-covered holes (vias) and using through-hole components. PCB cost is increasing with the number of layers and, for simple PCB designs, it is common with only a front and a back layer. For advanced PCBs, more layers might be necessary. For PCBs with noise-sensitive components, ground and power planes are beneficial. This is to avoid ground loops and cross-wire communication. Ground and power planes can also have a capacitive effect for ensuring a stable power supply.

The cSLIM module has some space-demanding components, with the display, GNSS-module and LoRa module being the largest. Without covering the antennas of the development kit, there is not enough space for all components on a single layer. It was therefore decided to place components on both the front and back layer of the PCB.

5.2.2 Component selection

Resistors
Based on previous SMD designs, thick-film 0805 resistors of size 8x5 inches were selected. Handling approximately 0.1W should be enough. With a variation of $1 - 5\%$, they introduce an uncertainty in the analog battery measurement. However, this is not a problem as the measurement is only to be used to give an approximate charge of the battery.

Capacitors
Ceramic SMD capacitors work well in high-frequency operations. 0603 capacitors (6x3 inches) were selected as being used by the SLIM module.

LEDs
The LEDs should be low power and visible at a distance. As the LEDs are driven by a 3V power source, the voltage drop across the diode has to be lower than 3V. LEDs with voltage drops of 1.8V (red, yellow) and 2.65V (blue, green) was selected. With 820 Ω and 330 Ω series resistors, this gives a forward current of 1.46 mA and 1.06 mA, respectively. For the prototype, this is okay. However, if LEDs are to be turned on at all times, a milli-Ampere of current per LED will significantly reduce the buoy's operational lifetime.

Logic level converters
Logic level converters are, as the name implies, regularly used to connect devices with different TTL (transistor-transistor-logic) levels. Common are conversions between the three standard voltage levels 5V, 3.0V-3.3V and 1.8V. All peripherals in the system are able to run at a 3V TTL-level. However, as it is desired to power down parts of the system, logic level converters were placed between some peripherals and the SPI bus to ensure a tri-state behavior of the inputs. Also, as described by Rundhovde [33], there is a problem with the GPS module waking up by traffic on the SPI MOSI line. Placing a logic level converter before the GPS' MOSI input, enabled by the GPS chip-select signal, should solve this problem.

Voltage regulator
A stable 3V supply is required for a standalone cSLIM module, driving the nRF9160 and the peripherals. As the DK has an internal voltage regulator, this is not necessary for the module to work. However, it is implemented to reduce the gap to a freestanding solution without the development kit.

The system has to be low power. Therefore, it is desirable with a switching voltage regulator, as these typically are more power-efficient than linear voltage regulators. Based on the SLIM design, the TPS63000 was selected. According to its datasheet [43] this is a buck-boost switching voltage regulator with up to 96% efficiency that can provide an output current of up to at least 1.2 A in this application (step down mode, 3 V output 3.6 V input). Introducing switching noise, the regulator should be placed away from analog circuitry.

Display
The LS013B7DH03 from Sharp Microelectronics is a power-efficient transflective display with an SPI interface. With a typical average power consumption of 12 uW on standby and 50uW on display update, it is one of the less power demanding and still affordable displays on the market [19]. As the frequency of display update is small, an e-ink display was also considered for the application. These have the advantage of maintaining displayed pixels without drawing power. On the other hand, more power is needed when updating the display. As no suitable 1.28" or similar e-ink display with SPI or I2C interface was found, the display from Sharp Microelectronics was selected. As the SLIM module uses this, drivers can also be reused.

FRAM
In order to meet the goal of persistent storage while maintaining a low power design, a FRAM chip was selected for storing data. The selected FRAM chip CY15B108QN's datasheet [8] states a low power consumption of typical 300uA while active (at 100MHz), 100uA standby and 3uA in sleep mode.

uSD Card Holder
To give the ability of persistent storage and to make it easy to retrieve the information on an external computer, a uSD cardholder was added to the design. LLc's are added to turn the uSD card off when not used as this draws energy when active, dependent on card manufacturer and type.

LoRa Module
The WLR089u0 is by its product page [48] described as a low-power FCC, IC and RED certified LoRa module from Microchip. The module contains a SAM R34 family microcontroller, based

on the Arm Cortex M0+ MCU architecture, and a sub-GHz radio delivering up to 18.6dBm TX power and an RX-sensitivity down to -136 dBm. With sleep currents down to 790nA, the module is well suitable for battery-powered applications.

GNSS Module
The nRF9160 does contain a GPS radio. However, no functionality for gaining precise timing was found in the documentation. Therefore, a Ublox NEO-M8N module was selected based on the SLIM design for location and precise timing requirements. The module has an output pin for precise time synchronization, giving a pulse synchronized with the GPS clock. This partly enables the system to meet the time synchronization requirements. Rundhovde [33] describes some problems with the GPS module from the SLIM design. For instance, the module was found to wake up with activity on the MOSI line. However, this can be solved by adding logic that only forward the MOSI line to the GPS module when the chip-select is active. An updated version, the NEO-M9N, might not have these problems. However, delivery was not estimated until January 2022. The NEO-M9N is pin-compatible with the NEO-M8N, making design changes small to upgrade the module.

RTC
The previously built SLIM module uses the microcontrollers hardware counter and external crystal to track local time between GPS updates. As the crystal driving the microcontroller is not very precise, the GPS module has to be wakened up often to meet timing constraints. The RV-3032-C7 module from Microcrystal was added to the design to increase the interval between waking up the GPS. This is an ultra-low-power real-time clock, using 160nA at 3V, with a temperature-compensated crystal. The RTC's datasheet [34] states its drift of 1.5 parts per million (PPM) between 0°C and 50°C, and 3.0PPM down to -40°C. The daily drift of the RTC can be given as in equation 8.

$$Dailydrift = spd * d/10^6 \tag{8}$$

where *spd* is the number of seconds per day, equal to 24*60*60 and d is the RTC drift in PPM.

For a drift of 1.5 PPM this gives a daily drift of 0.1296 s or 90 us per minute.

5.3 Schematics and PCB layout

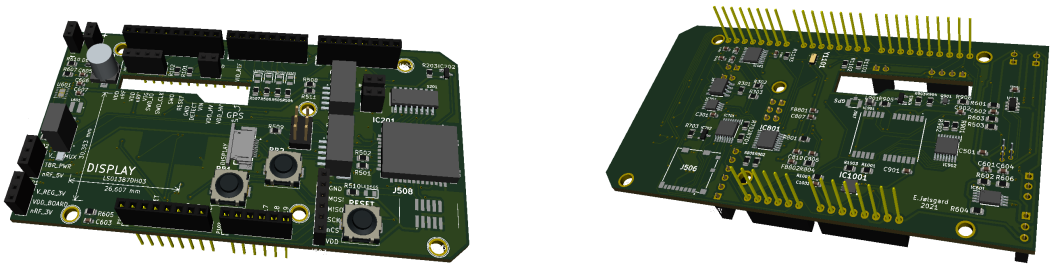
As the headers of the nRF9160-DK are Arduino Uno compatible, an Arduino Uno template was used to create a baseline for the nRF9160-DK. Pin names were rewritten to match the nRF naming convention and board edges were adjusted. As the nRF9160-DK has an extra header in the middle of the PCB, that is not used by this project, a hole was cut out for the header and GNSS antenna cable.

Component schematics and footprints were downloaded from Mouser and Component Search Engine. Using LibraryLoader, the components were imported to KiCad. Components were placed in the schematic and connected by wires and labels. In an attempt to make the schematic clear, the schematic was divided into sheets, with one sheet per main component and one sheet for connectors, LEDs and buttons. A top sheet was made for connecting it all. The schematic is shown in appendix D. When the schematic was complete, a netlist was generated. This includes information on all components and connections.

The netlist was imported in the PCB layout editor, PCBnew. Components were placed in the nRF9160-DK shield template. Those that needed to be accessible to the user were placed in the top layer, while most resistors and capacitors were placed on the back layer. An attempt was made to keep the voltage regulator and power source separated from the antennas and RF modules and to keep the LoRa and GPS radios away from each other.

When all components were placed, silkscreen was added to connectors and other relevant components. The tracks were separated into net classes, setting the track-width of power and ground tracks wider than regular signal tracks. All wires were placed on the board using the FreeRouting Autorouter. The batch optimizer reduces the length of tracks and the number of vias. The result was then imported to KiCad, and extra vias were added for ground and power (VDD) tracks. The

front and back layers were filled with copper, one connected to ground and one to VDD, creating a capacitive effect. Finally, the "design rule check" was run to ensure no unconnected items or unwanted connections before the fabrication Gerber and drill files were exported. A 3D view of the PCB is shown in figure 5.1.



(a) PCB front

(b) PCB back

Figure 5.1: 3D view of PCB using KiCad 3D Viewer

5.4 Production and assembly

PCBs were ordered from PCBWay. Components were ordered from Mouser, Digi-Key, RSComponents and Win-Source.

The PCB prototype was assembled at the electrical workshop of the Cybernetics Department at NTNU. High-temperature solder paste was added on the SMD-pads on the back layer. This is shown in figure 5.2. Using tweezers, components were placed on the PCB before it was put in the reflow oven, as shown in figure 5.3. In the reflow oven, the solder paste melted and, when cooling down, connected the SMD components. Due to the overflow of solder paste, there were short circuits on the logic level converters and real-time-clock. These were removed using a soldering iron and desoldering braid. A multimeter was used to verify that there were no short circuits in power lines or between the component legs.

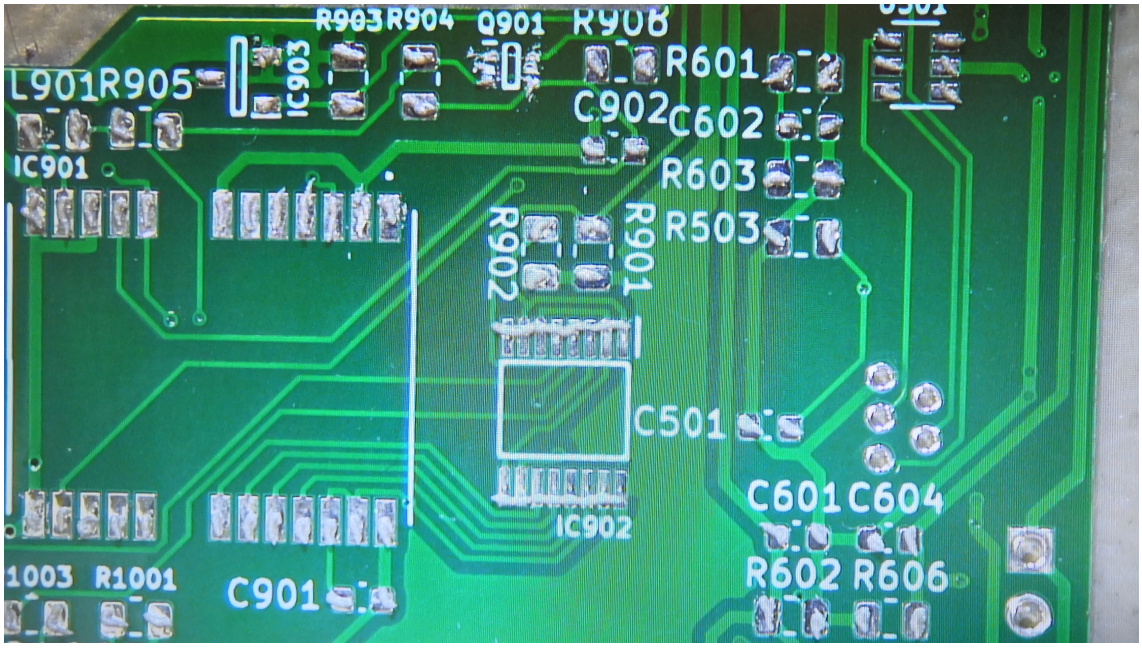


Figure 5.2: SMD solder paste appliance. The picture is showing the output of a microscope. Photo: Eivind Jølsgard/NTNU (CC BY 4.0)

The front side of the PCB was mounted in the same way as described above. To avoid desoldering

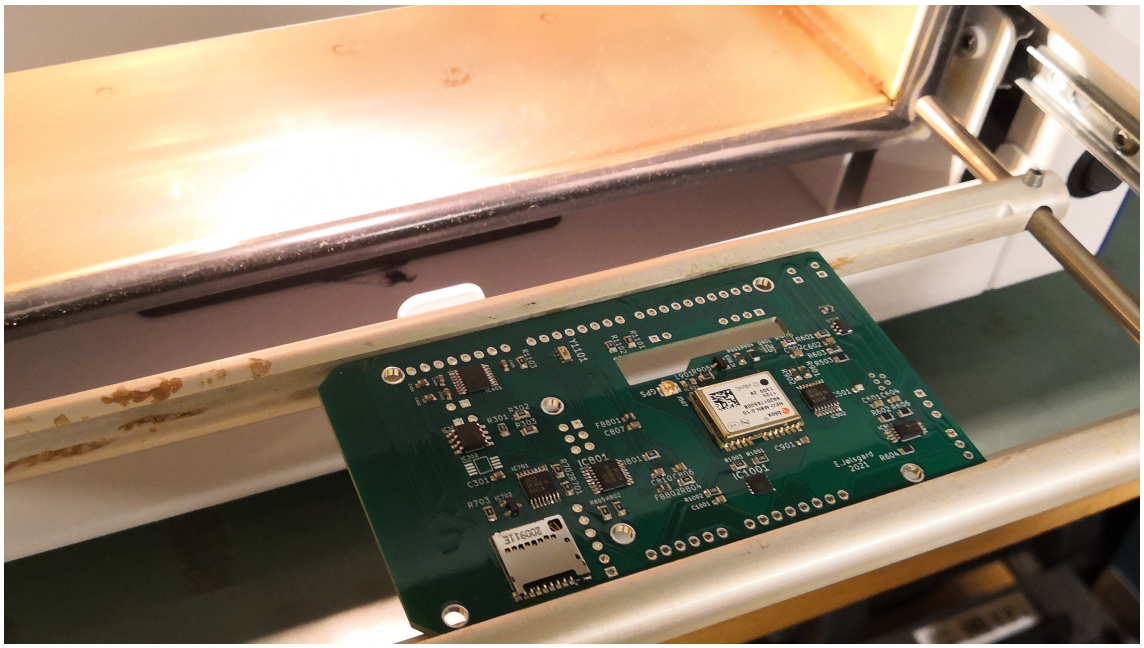
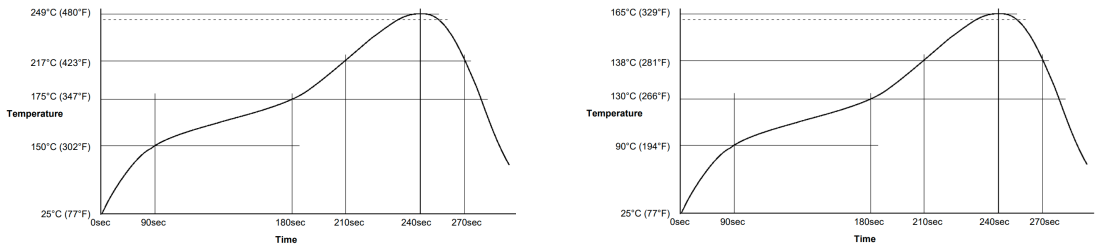


Figure 5.3: PCB ready for the first round in the reflow oven, showing the back side of the PCB. Photo: Eivind Jølsgard/NTNU (CC BY 4.0)

components at the back layer, a solder paste with a lower temperature profile was used. The temperature profile was added to the reflow oven to support the low-temperature solder paste. The recommended temperature profiles for the solder paste used in this project are shown in figure 5.4. At last, through-hole components were mounted using a soldering iron. The final PCB is shown in figure 5.5.



(a) SMD291SNL (high temperature)

(b) SMDLTLFP (low temperature)

Figure 5.4: Solder paste reflow temperature profiles. To solder components on both sides of the PCB using a reflow oven, different solder paste were used on each side, starting with the high temperature solder paste. Figures by Chipquick [37] [38]

5.5 Testing and verification

A single PCB was assembled to test the functionality and verify the design. In addition, software drivers were adapted as described in 6.2. These were used to verify the design. The peripherals were tested one by one with the following conclusion:

1. The 3V voltage regulator was small and difficult to solder. After some attempts, a short circuit led to desoldering it from the PCB. Instead, the nRF9160-DK voltage regulator is used to power the board. It is not believed that there are errors in the design, as it is similar to the SLIM module design.

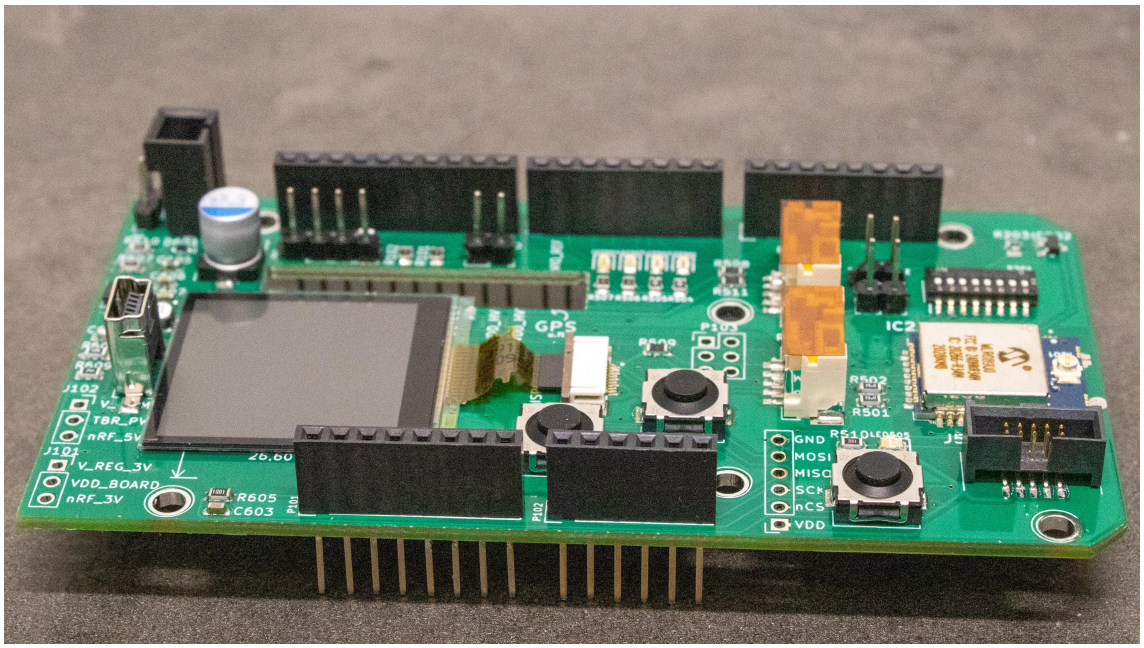


Figure 5.5: Final PCB prototype. Photo: Eivind Jølsgard/NTNU (CC BY 4.0)

2. LEDs and buttons work as expected.

3. After sending the PCB to production, it was discovered that the display had an active-high chip select (CS). This lead to problems as the logic level converter between the nRF and the display used the CS to enable its output. The problem was resolved by cutting the wire to the enable pin of the LLC and soldering it to ground. This makes the LLC useless. However, it has proven superfluous. As the problematic connection is underneath the LLC, the wire has to be cut before soldering the device. Other than this, the display works as expected.

4. The LoRa module was programmed via the SWD connector and respond to `mac reset 868` calls from the nRF9160. Also, the USB com port on the shield, connected to the WLR089u0, was verified using an example program. However, this is not implemented in the LoRa RN Parser.

5. The I2C bus and other pins connected directly to the LoRa module did not work unless the LoRa module is enabled and the pins are configured as inputs.

6. The RTC time was set and read without problems. The Uart COM port of the development kit had to be disabled in the board controller for the time-pulse signal to work. The board controller is described further in section 2.1.5

7. The RS485 interface with the TBR sensor is working as intended.

8. The RS232 was not tested due to the lack of an RS232 interface on the computer and not being vital to the final design. It should be tested before the next revision.

9. Testing of the FRAM chip, by writing to and reading from different parts of memory, shows some addresses working as expected, while others do not. No pattern of working addresses has been found. However, the results are reproducible. The FRAM is in a GQFN package with 0.35 mm pads and 0.65 mm BSC (Basic Spacing between Centers). During the production of the PCB the FRAM's SPI CLK and MISO line were not connected properly. The pins were attached using a soldering iron. However, it is assumed that the heat during soldering has been too high and have damaged the chip.

10. By adding FATFS support to the project, the uSD card was proven to work as intended. This was tested by writing strings to files in the memory card that was opened and read on a computer.

6 Software development

The following section gives insight into the buoy controller and DUNE Bridge software. Section 6.1 provides an overview of the software running on the cSLIM buoy controller. Section 6.2 describes the adaption of SLIM drivers and the development of new drivers to support the hardware on the cSLIM shield. Section 6.3 describes the development and workings of the application tasks scheduled by the Zephyr RTOS. Section 6.6 describes the software running on the WLR089u0 LoRa module and, finally, section 6.7 presents the DUNE Bridge.

6.1 cSLIM software overview

The cSLIM software uses the Zephyr RTOS to interface with hardware peripherals, schedule tasks, power management, and handle fault detection. Figure 6.1 shows an overview of the cSLIM buoy controller software.

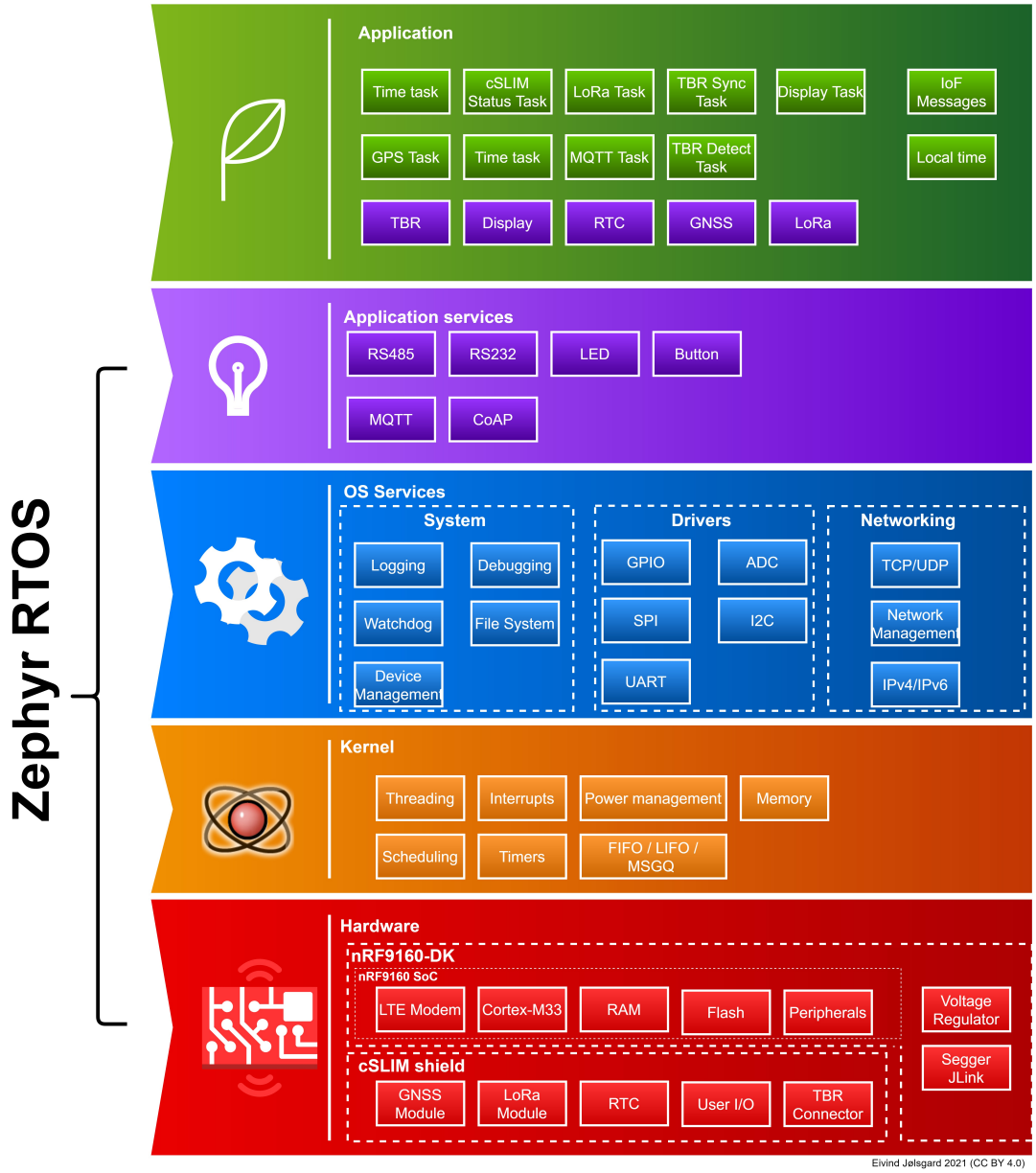


Figure 6.1: cSLIM overview

At the end of the project the cSLIM software is divided into the following sub-folders:

- boards/arm/cSLIM
- src/buffers
- src/devices
- src/drivers
- src/messages
- src/mqtt
- src/tasks
- src/time
- src/ugui

In addition, the Zephyr RTOS and nRF-drivers are in the nRF Connect SDK. The content of the application is further described below.

6.1.1 boards/arm/cSLIM

This folder contains the configuration of the cSLIM board, including Device Tree Sources and Kconfig. The configuration is based on the nRF9160 development kit configuration, and the Device Tree is adapted to suit the pinout and devices of the cSLIM shield. As the nRF9160 is divided into one secure and one non-secure partition, there are configuration files for each separate part. The cSLIM application is running in the non-secure partition.

6.1.2 src/buffers

This folder contains FIFO buffers to ease the implementation of code from the SLIM module. As Zephyr also has FIFO buffers implemented, the cSLIM code can be further adapted to use these instead. However, this has not been prioritized as the solution is working as intended.

6.1.3 src/devices

This folder includes device-specific functionality for the cSLIM shield components, e.g., control of the display, GNSS module, RTC and LoRa module. The functionality is further described in the source code, digitally available in appendix C.

6.1.4 src/drivers

This folder includes functions for interacting with the zephyr peripheral drivers for the nRF9160, i.e., SPI, I2C, UART and RS485. As the nRF9160 does not have an RS485 peripheral, UART is used with GPIO control of the RE and DE lines. The SPI driver uses a mutex to avoid multiple devices using the SPI driver at once.

6.1.5 src/messages

This folder contains cSLIM, IoF and TBR message formats and functions for converting TBR and cSLIM status messages to the IoF message format. This format is designed to reduce the number of bytes necessary for low power message transmissions. The message format is given in section 2.3.2.

6.1.6 src/mqtt

This folder contains functionality for sending and receiving messages using MQTT over NB-IoT, including enabling the modem and connecting to the network. MQTT broker configuration, like IP address, port and user credentials are set in the project configuration file (`proj.conf`).

6.1.7 src/tasks

This folder includes application tasks that the Zephyr RTOS schedules. Also, message queues for communicating between tasks and interrupt handlers for LoRa and GNSS are located here. Software tasks are further described in section 6.3.

6.1.8 src/time

This folder includes the local time library and time conversion functions. For precise timing, the Zephyr hardware timer value is used. Using `k_cycle_get32()` at the event, the value is passed to the local time library that computes the actual UTC with 30.56 us resolution, not using frequency estimation (Hardware timer has a frequency of 32.768 kHz).

6.1.9 src/ugui

This folder includes the μGUI graphic library for embedded systems [47]. The library is used for presenting graphics and text on display.

6.2 Driver and device adaptions

The drivers for the original SLIM module were implemented for the EFM32GG microcontroller hardware. Useful drivers from the original SLIM buoy controller were adapted for use with the nRF9160 to reduce the development time for the new system. The reused libraries are the GNSS and display driver, including supporting libraries. Zephyr includes driver APIs (Application Programming Interfaces) for interacting with nRF9160 hardware peripherals as I2C, SPI, UART and GPIO. The adaption of SLIM modules, therefore, included making a hardware interface layer for the devices. Although not being the Zephyr way of creating and using device drivers [53], it increased the usability of SLIM source code. The following subsections describe the changes and development of other device drivers.

6.2.1 LED and button drivers

A new driver was created for the LEDs, using a GPIO driver. In addition, a driver was also developed, assigning interrupt handlers to the button presses.

6.2.2 SPI driver

An interface layer was developed to facilitate the use of the Zephyr SPI driver by multiple devices, i.e., the display, GNSS module and uSD card. The interface was based on the SLIM SPI driver to ease reusing previous display and GNSS module drivers.

6.2.3 OLED device

The display driver from the SLIM module was slightly rewritten. However, it is primarily identical to the SLIM module.

6.2.4 GNSS device

The GNSS driver was rewritten to reduce the number of global variables used by the functions and to avoid function calls by the driver to higher-level software. Instead, the variables are passed to the functions as input parameters. Parameters that are manipulated by the function are passed by reference. Instead of calling functions in the time manager library from the driver, the calling function of the driver is using the manipulated data to assess its state and taking appropriate actions. Some calls to the display still exist in the GNSS initialization process and acquisition of navigation data. However this is only debugging information to the user.

6.2.5 I2C driver

An interface layer was developed to facilitate the use of the Zephyr I2C driver by multiple devices. For the current design, only the RTC is using I2C.

6.2.6 RTC device

An RV-3032-C7 header file was created for all register addresses of the RTC. Primary register read/write functions were created before adding functionality to get and set the date and time of the RTC. Also, support was added to enable and disable the RTC's clock-out pin, setting its frequency and enabling the EVI (External Event Interrupt Input) pin to synchronize the time with an external event on rising edge.

6.2.7 LoRa device

An interface for transmitting and receiving char-strings using UART was created for the LoRa module.

6.3 Software tasks

With the Zephyr RTOS providing task-scheduling support, the cSLIM software is divided into tasks, each responsible for specific hardware modules, communication or monitoring. The tasks communicate with each other by passing messages through buffers. Tasks that are scheduled periodically or awaiting messages are put to sleep by the operating system, reducing power consumption.

The software tasks include the:

- **cSLIM status task**, periodically gathering status information about the cSLIM module, forwarding the status to the MQTT and LoRa tasks.
- **Display task**, initializing the display, ugui library and continuously toggling it. It also updates the display with the content of the display buffer upon request.
- **GPS task**, managing the GNSS module. This includes waking up the module, selecting GNSS start level (cold, warm, hot) dependent on time since the last wake, requesting navigation and UTC data and forwarding data to the time task and cSLIM status task. The GPS task also puts the GNSS module to sleep when not used.
- **LoRa task**, configuring the LoRa module and establishing a connection with the LoRaWAN gateway. The LoRa task is then responsible for sending IoF messages to the LoRaWAN GW.
- **MQTT task**, enabling the nRF9160 modem and connecting to the cellular network and MQTT broker before sending IoF messages using NB-IoT.

- **TBR detect task**, responsible for receiving messages from the acoustic receiver, decode the messages, add synchronization offset and forward messages to the LoRa and MQTT tasks.
- **TBR sync task**, responsible for synchronizing the time of the acoustic receiver and store its synchronization offset.
- **Time task**, responsible for synchronizing the local time of the nRF9160 with the UTC. This task also computes the estimated frequency of the RTC and the allowed time before the next GNSS update. This task will also detect if the local time has drifted more than it should and is publishing the observed drift to the MQTT broker.

The following subsections are further describing some of the main tasks of the buoy controller as a whole.

6.4 Time synchronization

The time task wakes the GPS task to synchronize the local time of the buoy controller by sending a GPS_WAKEUP_REQUEST message to the GPS task. The GPS task will accept this request, waking the GNSS module, enabling its PPS line and replying with a GPS_UPDATE_ACCEPTED message. The tasks are then sleeping until the first event on the PPS line. The generated interrupt is sending a message to the GPS task indicating a time pulse has been received. The GPS task then requests nav-pvt data from the GNSS module that is pulled after the next time pulse. The navigation data is sent to the cSLIM status task. On the next time pulse, time-data is sent to the Time task that sets the time of the RTC and enables its EVI input. This enables the RTC to synchronize its internal clock, and clock-out, with the next clock pulse from the GNSS module. If another GNSS time pulse occurs before this is complete, the synchronization is restarted. At the next time pulse, if valid, the Time task is comparing the local time (before the update) with the GNSS time. The time task then uses the drift to estimate the maximum time to wait before a new GNSS update is required and the estimated frequency of the RTC. The time until the next update is halved and low-pass filtered to make certain constraints are met. Estimated frequency is also low-pass filtered to reduce the impact of a single, outlying measurement. Figure 6.2 and 6.3 shows a simplified state machine for the GPS task and time task, respectively. Figure 6.4 shows a successful update of the local time.

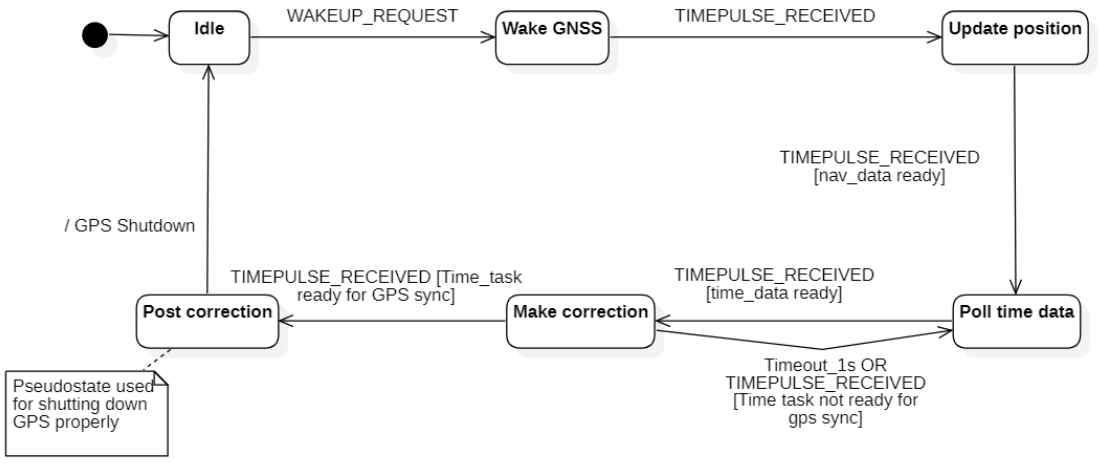


Figure 6.2: GPS Task STM

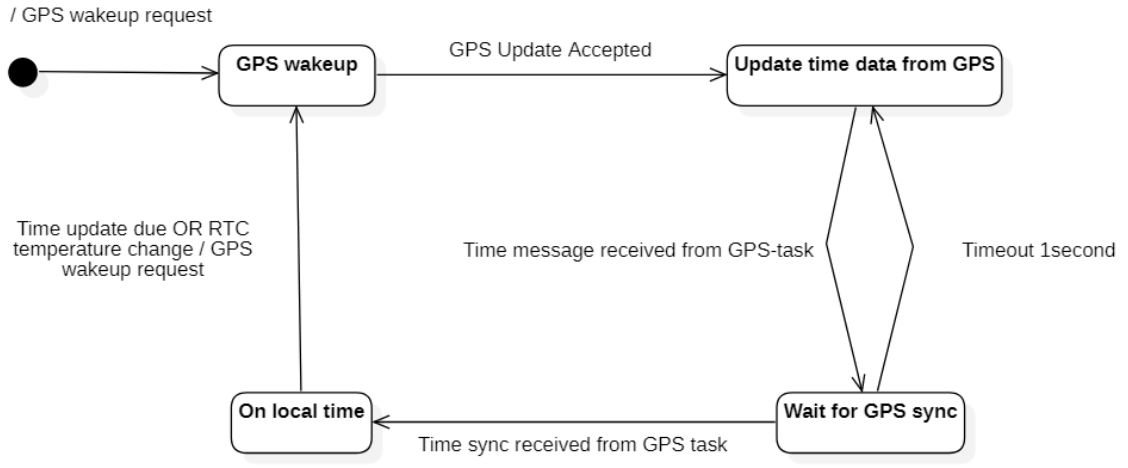


Figure 6.3: Time Task STM

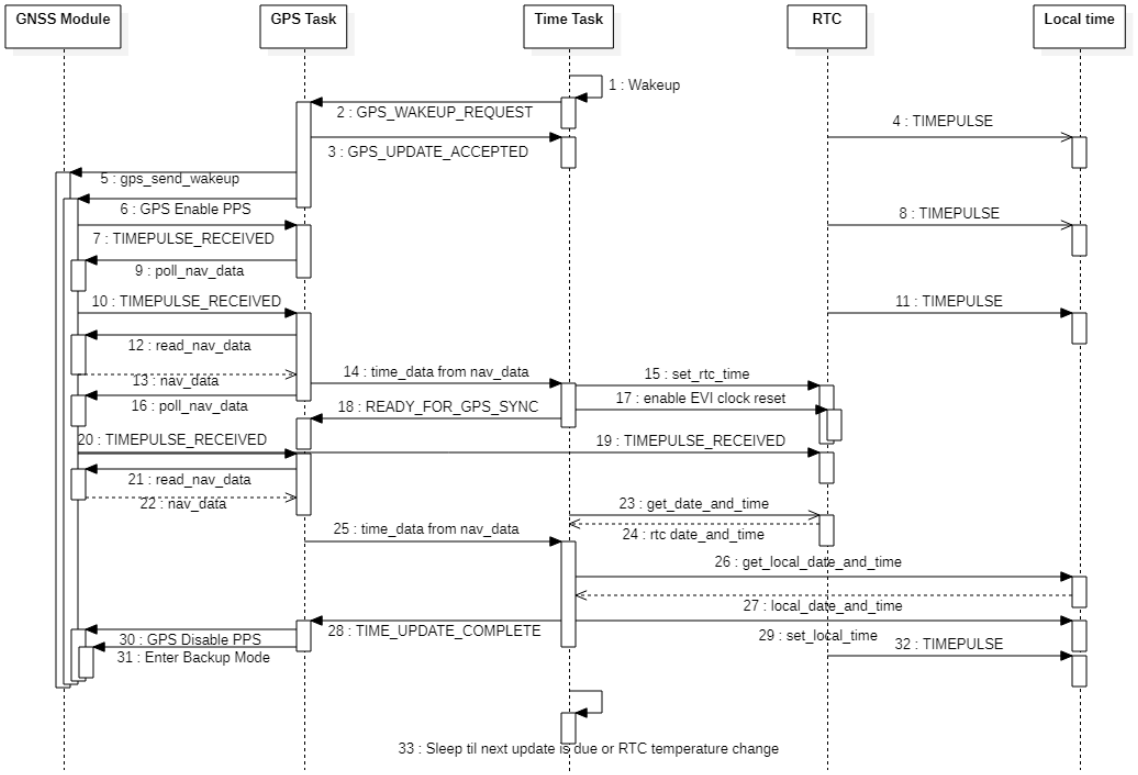


Figure 6.4: Local time update sequence diagram showing the sequence of events at a successful update. The update is not successful unless the two following time messages are one second apart, and no time pulse from the GNSS module occurs before the time task is ready to receive the second time pulse, synchronizing the RTC.

6.5 RTC frequency estimation

Compared to GPS time, the time since the last update and observed drift of the RTC, is used to estimate the RTCs frequency. This is then used to improve the local time of the buoy controller. The estimated frequency is added to the local time calculation using a low-pass filter.

Figure 6.5 shows the local time calculation and frequency estimation.

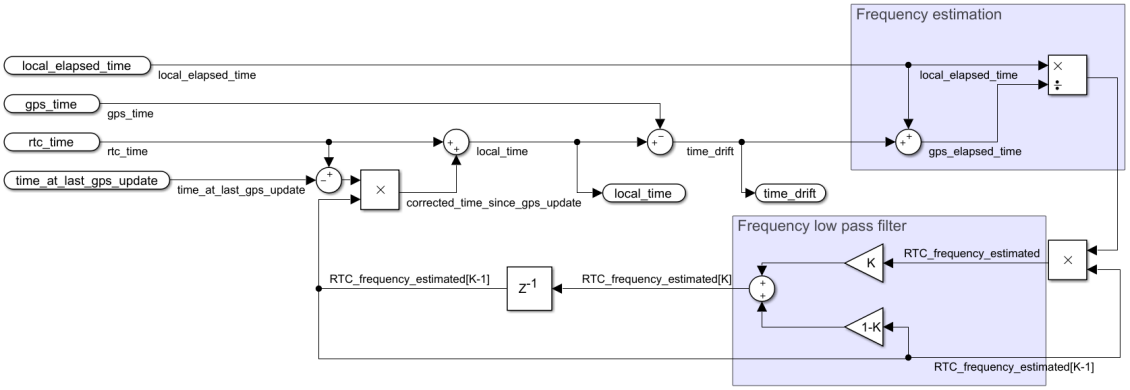


Figure 6.5: Local time calculation and frequency estimation. The calculations are run on every time calibration from the GNSS module. Initial estimated frequency, $RTC\_frequency\_estimated[0]$ is 1.0 Hz. Also, $0 < K \leq 1$.

6.6 LoRa module

The LoRa module is running the `ATSAMR34_LORAWAN_RN_PARSER`, referred to as the LoRaWAN RN Parser, available on GitHub [3]. The module, running a SAMR34 based on the ARM Cortex-M0+, is programmed through an SWD interface. The UART, I2C and SPI interfaces, and some other GPIO lines from the nRF9160 are directly connected to the LoRa module. The LoRaWAN RN Parser, therefore, has been modified to configure these pins as input, except the UART interface used for communication with the nRF. Also, pin PB03 has been configured as an output to enable the blue LED by the LoRa module, indicating that it is enabled. In addition, the LoRa module is connected to an 8-pin DIP-switch, designed for setting unique LoRa addresses when using multiple buoy controllers. Implementing the address switch is not prioritized as the address can be set using a command from the nRF9160 that is easily changed in code during development. However, adding this functionality will ease the deployment of multiple devices in the future.

6.6.1 LoRaWAN RN Parser user interface

Using the LoRaWAN RN Parser, the LoRa module is controlled from the nRF using text strings similar to AT-commands used by LTE modems. All commands are ended with `\r\n`. Table 6.1 shows an overview of some of the commands used for this application. All commands are acknowledged with a response message, as shown in table 6.2. Figure 6.6 shows the flow of connecting the LoRa module to the network.

Command	Description	Parameters
sys sleep <mode><length>	Sets the system in sleep mode for a given number of milliseconds	<mode>standby or backup <length>time in ms
sys get ver	Get system version	
mac reset <band>	Reset and initialize LoRa band	<band>representing band to reset, e.g. 868 for EU868
mac set <param><value>	Sets property of the MAC layer	<param>parameter to set, e.g. appkey, deveui, joineui, dr(data rate) and pwridx(power index). <value>Value to be set to the parameter
mac tx <type><portno> <data>	Transmit data over LoRaWAN	<type>cnf or uncnf, dependent on if the message should be confirmed by the receiver. <portno>decimal number representing port number [1,223] <data>Data to be sent
mac join <mode>	Join LoRa network	<mode>otaa or abp, dependent on join type.

Table 6.1: LoRa RN Parser command overview of the most important commands for this thesis. For a full overview, see the LoRa RN Parser documentation on Github [3].

Response	Description
ok	The request is valid and processed
accepted	Device is accepted to the network
denied	Device is rejected to the network or did not receive response
invalid_param	Request or parameters are not valid
mac_tx_ok	Transmission is successful
busy	Device is busy

Table 6.2: LoRa RN Parser response messages

```
mac reset 868
ok
mac set adr on
ok
mac set deveui 8278000000000600
ok
mac set joineui 0011224455119988
ok
mac set appkey 00110022003300440055006600770088
ok
mac set pwridx 0
ok
mac join otaa
ok
accepted
```

Figure 6.6: Successful LoRa join procedure. If the LoRa module replies with denied or invalid-param, the last command will be resent until accepted.

6.7 DUNE Bridge

The DUNE bridge is using the Paho MQTT c and c++ libraries for connecting and subscribing to the MQTT Broker. Nikolai Lauvås, NTNU ITK, provide IMC::FishTag and IMC::TBRSensor message formats. Libraries were downloaded and installed in the Virtual Machine provided by LSTS [20]. The Dune MQTT bridge was created in the `src/Transports/MQTT` folder using the python command.

`python ../source/programs/scripts/dune-create-task.py ../source 'Eivind Jølsgard' 'Transports/MQTT/SLIMMessageBridge'`.

The paho MQTT libraries were added to the CMake files. The following functions are implemented in the IoF MQTT message bridge:

- **Task(const std::string& name, Tasks::Context& ctx):Tasks::SimpleTransport(name, ctx),**
  m_sock(NULL)

  This is the function called when creating a task with the message bridge, setting default parameter values or getting them from the .ini file.

- **void onResourceAcquisition(void)**

  Called when the task is acquiring its resources, the function creates a new mqtt_client instance and establishes the connection to the MQTT-broker. If successful, the client then subscribes to the requested topic. If the connection to the MQTT-broker fails, an error will be thrown, requesting restarting the task.

- **void onResourceRelease(void)**

  Called when the task is releasing its resources, the function unsubscribes from the topic and disconnects from the broker.

- **int setTBRlocation(int tbr_sensor, int latitude, int longitude)**

  This function is storing the position of a TBR sensor, enabling the bridge to add the position to an IMC::FishTag or IMC::TBRSensor message.

- **int getTBRlocation(int tbr_sensor, int* latitude, int* longitude)**

  This function changes the value of latitude and longitude based on the TBR_sensor and locations stored in the bridge. If the TBR_sensor is not found, the latitude and longitude will be set to zero, and an error code returned (-1). On success returned value is 0.

- **fillTBRSensorMessage(IMC::TBRSensor* sensor_msg, uint8_t* message, uint32_t base_timestamp)**

  This function is filling an IMC::TBRSensor message based on received data in IoF format.

- **int fillTBRFishTagMessage(IMC::TBRFishTag* tag_msg, uint8_t* message, uint32_t base_timestamp)**

  This function is filling an IMC::FishTag message based on received data in IoF format.

- **onMain(void)**

  This function is receiving messages from the MQTT broker and decodes the header. The message content then fills an IMC::FishTag or IMC::TBR sensor message or stores the TBR sensor's location locally. When IMC::TBRSensor or IMC::FishTag messages are created, the locally stored TBR sensor's location is added, and the messages are dispatched to the DUNE framework. Figure 6.7 shows the flow of the bridge in normal operation.

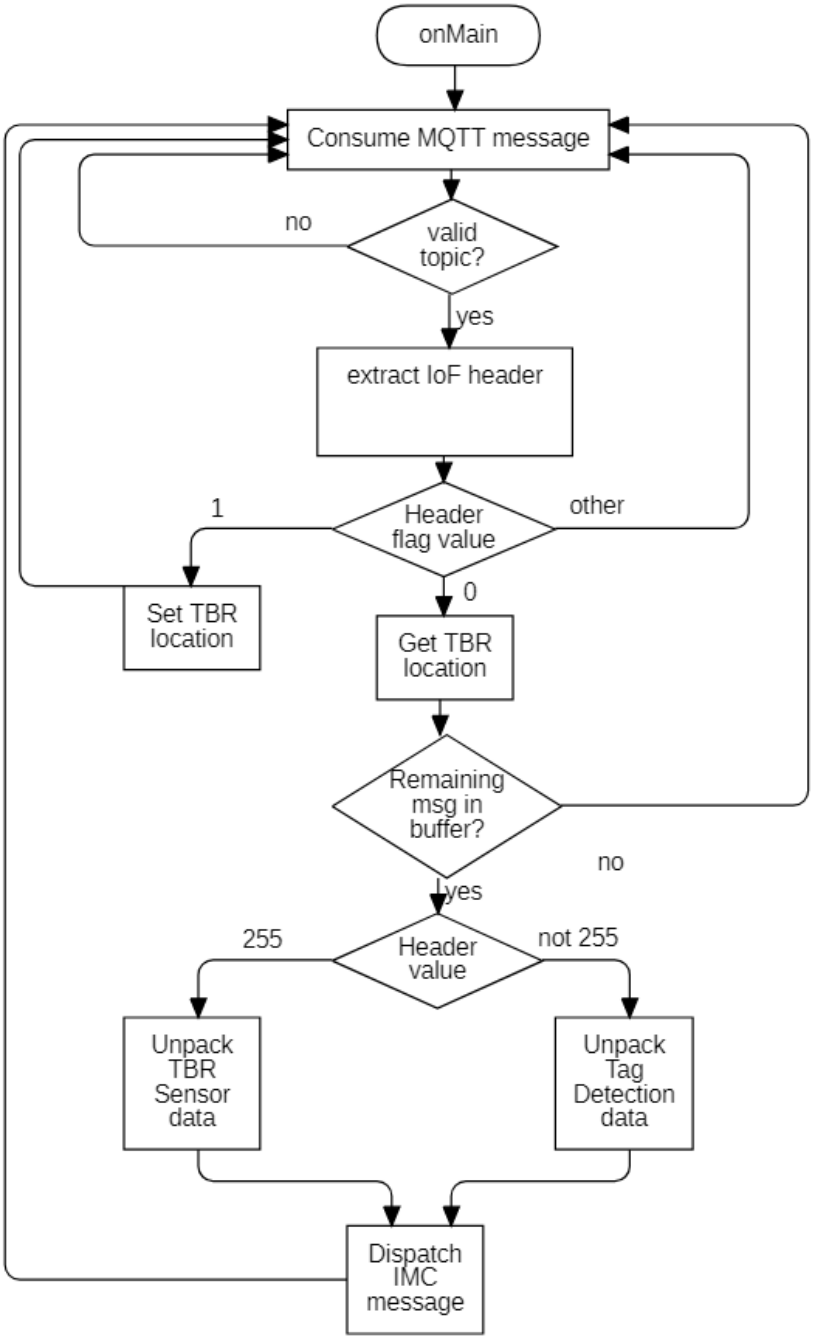


Figure 6.7: Dune Bridge flow while running

7 Putting it together

When initial prototypes for the software drivers and the hardware shield were developed, the shield was mounted on top of the nRF9160-DK as shown in 7.1. The drivers and hardware were tested, debugged, and further adapted. Hardware faults due to overflow or absent solder paste were solved using a soldering iron as shown in figure 7.2. The zephyr tasks were then added to the application, and the system was further debugged, adapted, and tested. The next chapter presents tests and results of the system.

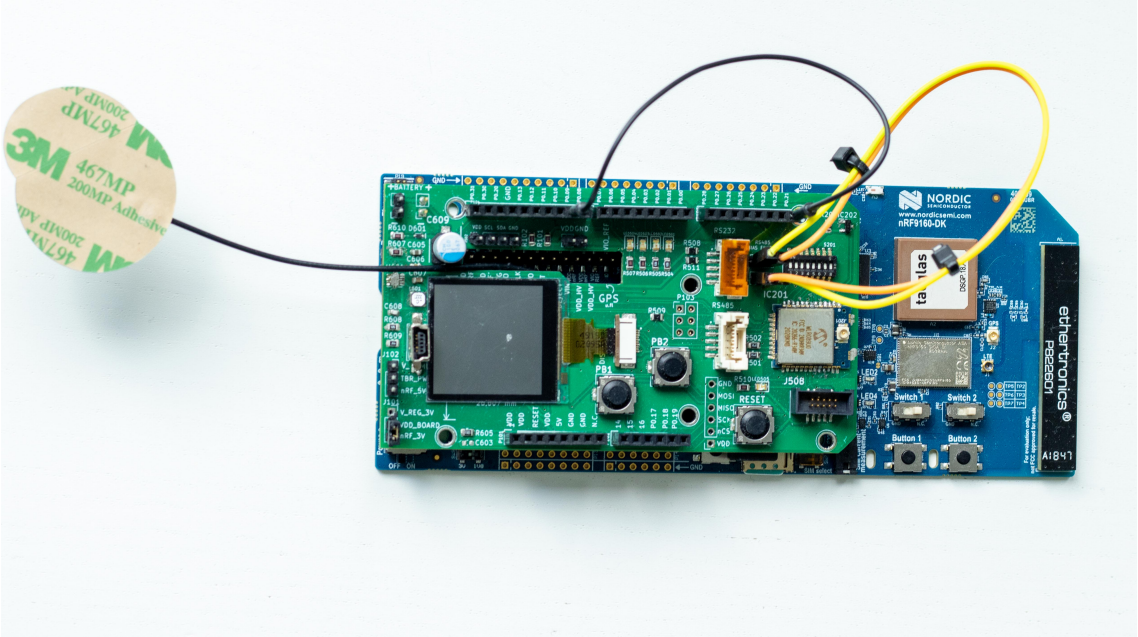


Figure 7.1: cSLIM buoy controller hardware. Photo: Eivind Jølsgard/NTNU (CC BY 4.0)

Figure 7.2: Fixing hardware with a solder iron. Photo: Eivind Jølsgard/NTNU (CC BY 4.0)

8 Testing and results

The following section explains the test setup used in this thesis. Not all tests have been successful in obtaining results. However, they are included as they are seen as relevant for future tests and system verification. The rest of the chapter presents test results and measurements and gives some comments on the results.

8.1 Test setup

8.1.1 RTC drift and local time correction

Goal: To verify that the local time of the nRF9160 is synchronized with the GNSS time with an error less than what is given in requirement **cSLIM-FR.5a** (appendix B)

Method: The local time is compared with the GNSS time at update intervals. The observed drift is transmitted to the MQTT broker and plotted. The system aims to have a drift of 250 us between GNSS updates, and most results should therefore lie in this area. It is, however, accepted with drift up to ±500 us from the actual GNSS time. The tests periods have been varying from a couple of hours up to almost 24 hours.

8.1.2 Time synchronization

Goal: For the system in total, the vital part to be synchronized with other devices is the TBR acoustic receiver (**UAR-FR.1b**, appendix B), as it is the time in the receiver that is added to the tag detections, and hence is used for positioning of fish.

Method: To verify if the time is correct, a TBR connected to the cSLIM module, is placed a couple of centimeters from a fish tag together with a TBR sensor connected to a previously built SLIM module. As the speed of sound is approximately 343 m/s = 34.3 cm/ms in air, a difference of 1 cm is negligible with a millisecond resolution in the acoustic receiver. Based on results from Rundhovde, testing three SLIM modules simultaneously, the timing accuracy of the SLIM module was found to be satisfactory. It has thereby been assessed to be sufficient to compare the cSLIM module with a single SLIM module. [33]. The acoustic tags are sending a signal every 60 to 180 seconds. As the tags internal clock is drifting compared to GPS time, the millisecond part of the tag detections is changing over time. This results in the millisecond part of the detection to be given as

$$T_{ms} = (k * t)\%1000 \tag{9}$$

where T_{ms} is the millisecond part of the detection, t is the time and k is a constant indicating the drift. Figure 8.1 shows an expected plot of the results, with cSLIM and SLIM tag detections superposed.

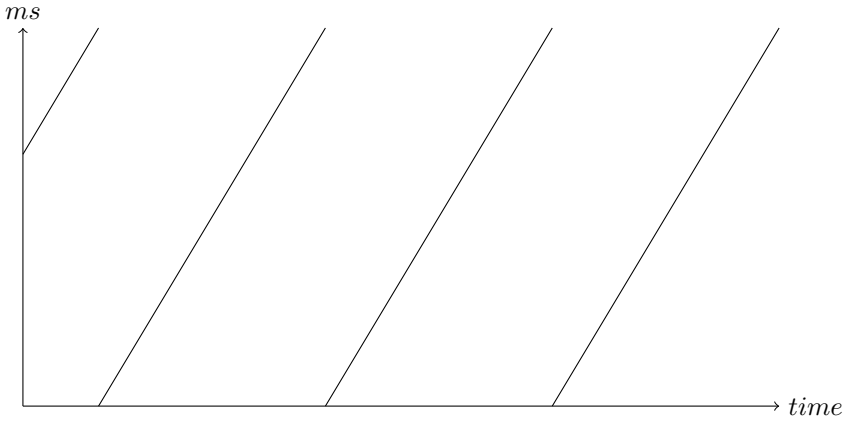


Figure 8.1: Expected results of plotting millisecond part of tag detections with respect to time

8.1.3 Energy consumption

Regarding energy consumption, it can be categorized into two main categories; static and dynamic. Static power consumption is due to current leakage in components and will be constant as long as the device is enabled. On the other hand, dynamic power consumption is due to electrical changes in the system, e.g., in transistors, registers, computation units, and wireless transmission amplification. The following explains the test setup for measuring and estimating energy consumption for the device and application.

8.1.3.1 Overall energy consumption

Goal: Verify that it is possible to develop a energy-efficient solution based on the prototype, fulfilling **cSLIM-NFR.1**.

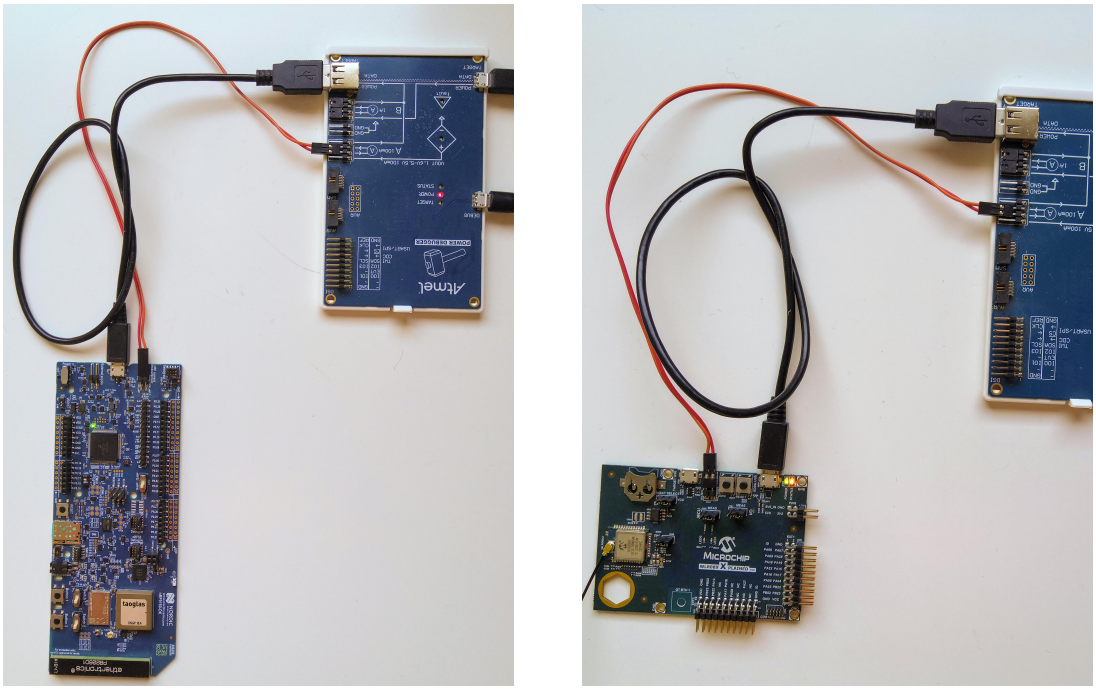
Method: The system's energy consumption was measured using a Microchip Power Debugger Kit measuring the current delivered through the USB connector of the nRF9160-DK. This was connected through the high current/low-resolution input of the power debugger. Also, as the nRF9160-DK has a connector for measuring the power consumption of the nRF9160 SiP, this was connected to the low current/high-resolution input of the Power Debugger. On higher loads of the microcontroller, e.g., when connecting to the network, resulting in current consumption's above 100 mA, the low current measurement cut power to the nRF, and the system restarted. This was therefore not always used. The shield energy consumption is measured by connecting the power profiler on the VDD header of the shield between the nRF9160 VDD pins and the components of the shield.

8.1.3.2 LPWAN power consumption

Goal: For choosing an energy-efficient LPWAN solution for the IoF project, energy consumption measurements of the proposed LPWAN technologies are required. This is also needed to verify the functional requirements in **cSLIM-FR.2a** and **cSLIM-FR.3a**, regarding power-efficient LPWAN communication.

Method: The cSLIM system itself was not used to measure the power consumption of NB-IoT and LoRaWAN. To reduce the amount of current drawn in total, hence increasing accuracy of the LPWAN measurements, the nRF9160-DK was used standalone for NB-IoT, while a WLR089u0 XPLAINED board was used for LoRa. This contains the same LoRa module like the one on the cSLIM module. The high-current measurement input of the Power Debugger Kit was connected to the board's USB-ports. The low-current measurement input was connected to the nRF9160 and WLR089u0 directly through power measurement connectors on the boards.

The test setup for LPWAN power consumption is shown in figure 8.2.



(a) NB-IoT, nRF9160-DK (b) LoRa, WLR089u0 XPLAINED

Figure 8.2: LPWAN power consumption test setup. The nRF9160-DK and WLR089u0 Xplained are connected to a Power Debugger from Microchip. The debugger is connected to a computer using Microchip Studio to read the power consumption. The LoRa module has an external antenna not shown in the picture. Photo: Eivind Jølsgard/NTNU (CC BY 4.0)

Although not using the cSLIM module, the code running on the WLR089u0 XPLAINED is identical to the one running on the cSLIM LoRa module. For the nRF9160, the tests use the MQTT client example provided by Nordic Semiconductor. This is the foundation upon which the cSLIM MQTT client is built. Adaptions are made to the example to mirror configuration and usage in the cSLIM application.

8.1.4 Data throughput

Goal: Verify that NB-IoT and LoRaWAN offers suitable data rates for operation conditions and data volume in the IoF project, as stated in requirement **cSLIM-FR.2c** and **cSLIM-FR.3c**.

Method: The active transmit time used to send a message of $n \in [1, 10, 33]$ tag detection's was measured for LoRaWAN using the power analyzer, by measuring when the power consumption was higher than usual. However, as NB-IoT is based on repetitive transmissions, dependent on network coverage, the method described for LoRaWAN does not give valuable results for the NB-IoT performance. Therefore, NB-IoT data throughput estimates are instead based on theoretical information.

8.1.5 DUNE Bridge

Goal: Verify that the DUNE bridge fulfills its input, output and message format requirements as stated in requirement **DUNE-BR-FR.1** to **DUNE-BR-FR.3**.

Method: Run system in simulation mode while the cSLIM buoy controller is running. Verify that tag detections and TBR sensor messages are dispatched by the DUNE Bridge using IMC formats.

The following sections present results from the measurements.

8.2 Energy consumption

8.2.1 cSLIM system energy consumption

The overall power consumption for various system applications is given in table 8.1.

Device	Current consumption (mA)
nRF9160-DK (idle)	39
nRF9160-DK(idle, IFMCU(1))	22
nRF9160(idle) + shield (GNSS off, LoRa on)	7.1
nRF9160(idle) + shield (GNSS off, LoRa off)	15.6(2)
nRF9160 (cSLIM-app, MQTT connected, GPS fix)	1.5
Shield (GNSS off, LoRa on)	6.5
Shield (GNSS search, LoRa joined)	35.8

Table 8.1: cSLIM Static current consumption measurement. (1) IFMCU disconnects some of the DK's components. (2) Lines from the nRF are connected directly to the LoRa module, drawing power when module is off.

8.2.2 LPWAN energy consumption

Figure 8.3 shows the energy consumption of NB-IoT and LoRa while in standby mode.

nRF9160 NB-IoT energy consumption

Table 8.2 and figure 8.4 shows measurements regarding the energy consumption of NB-IoT communication with nRF9160. Figure 8.5 shows the nRF MQTT-client retaining its connection with the MQTT broker, with eDRX enabled.

Event	Average Current Consumption [mA]	Measurement time[s]	Energy Consumption [uAh]	Energy Consumption per tag detection [uAh]
Standby mode (eDRX enabled)	0.4	–	400 / min	–
Paging mode	1.6	–	1570 / min	–
Connection to network	34.7	2-10 min	1156.7 - 5783.3	–
Retaining Connection	6.6	10	18.3	–
Single Tag detection	6.8	10	18.8	18.8
10 Tag detections	8.3	10	23.1	2.3
33 Tag detections	10.2	28.3		0.9

Table 8.2: Energy consumption of transmitting IoT messages using NB-IoT. Network connection consists of multiple phases. Measured current is of the most energy demanding phase of the connection procedure. Tag detection current measurements are the average over the entire transmission window. Retaining connection procedure is performed every minute at default configuration.

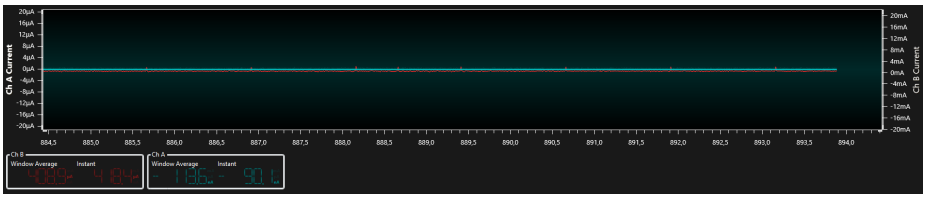
WLR089u0 LoRaWAN energy consumption

Table 8.3 and figure 8.6 shows measurements regarding the energy consumption of LoRaWAN communication with WLR089u0.

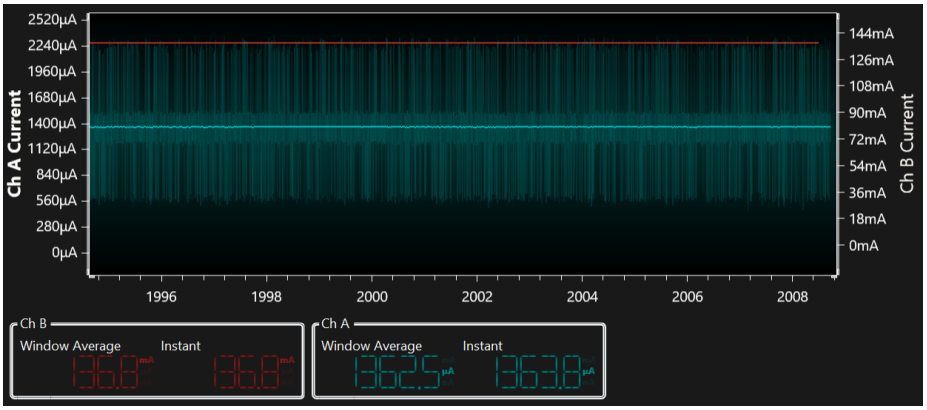
8.3 LPWAN coverage

Field tests to measure the LPWAN coverage has not been performed as part of this project. Rundhovde [33] previously tested the performance of SLIM and LoRa at sea, with a proven range up to 5.5 km. However, the system is expected to support even longer distances to the base station.

(a) nRF9160 (NB-IoT/MQTT) Paging (eDRX and PSM disabled)



(b) nRF9160 (NB-IoT/MQTT) eDRX enabled



(c) WLR089u0 (LoRaWAN

Figure 8.3: Current measurements of nRF9160 and WLR089u0 in standby mode.

The reader is further advised to look at the NB-IoT coverage maps in section 2.3.3 and the stated coverage in table 2.1.

8.4 Airtime and data throughput

8.4.1 LoRaWAN

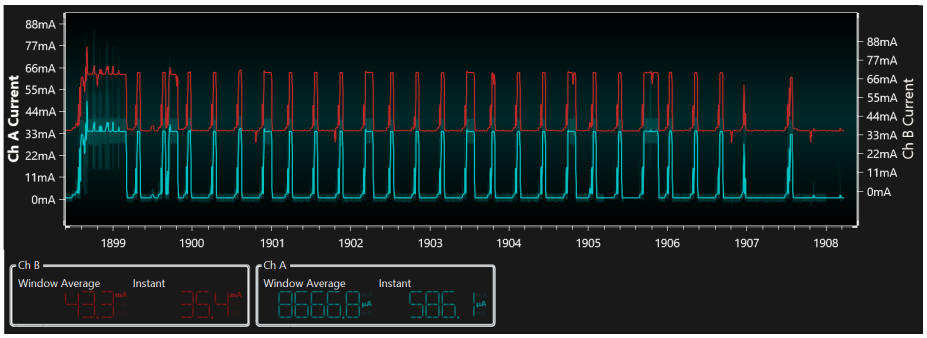
Table 8.4 shows the transmission time of tag detection messages using different data rates.

8.5 Timing requirements

8.5.1 RTC-drift and local time measurements

The drift of the RTC at approximately 23 $°C$ was measured for over 4 hours. The results are shown in table 8.5. The drift and times are identical at the us-part for multiple occurrences. It is most likely due to Zephyr cycles having a resolution of approximately 30.5 us because of the 32.768 kHz

(a) 1 Tag detection



(b) 10 Tag detections

(c) 33 Tag detections

Figure 8.4: NB-IoT current measurements of tag detection transmissions. eDRX and PSM power saving features are disabled for best response time.

oscillator in the nRF9160 SiP.

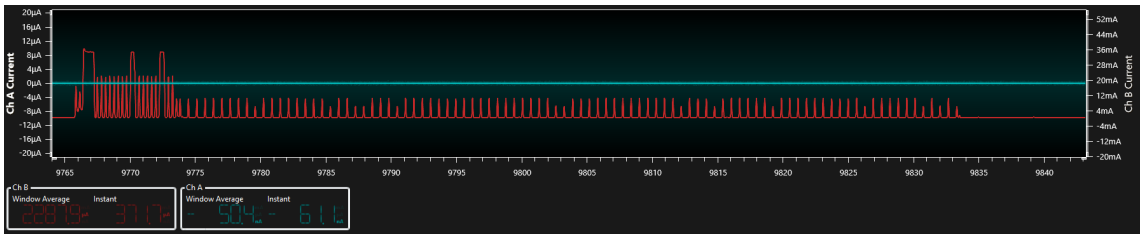


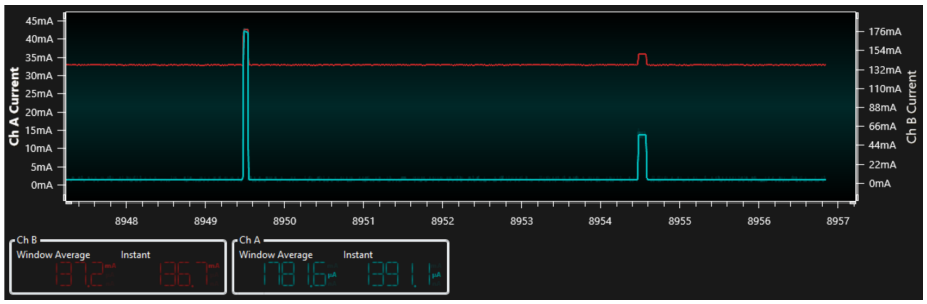
Figure 8.5: NB-IoT retaining connection with PSM. First part is the MQTT client retaining its connection. The modem is then paging for 60 seconds before eDRX is activated.

Event	Average Current Consumption [mA]	Measurement time[s]	Energy Consumption [uAh]	Energy Consumption per tag detection [uAh]
Standby mode	1.36	–	1360 / min	–
Connection to network	0.42	10		–
Retaining Connection	–	–	–	–
Single Tag detection (DR5, PwrIdx0)	0.42	10	1.17	1.17
10 Tag detection (DR5, PwrIdx0)	0.81	10	2.25	0.23
33 Tag detection (DR5, PwrIdx0)	1.74	10	4.83	0.15
33 Tag detection (DR5, PwrIdx5)	1.07	10	2.97	0.09
Single Tag detection (DR3, PwrIdx0)	1.08	10	3.00	3.00
10 Tag detection (DR3, PwrIdx0)	2.22	10	6.16	0.62
Single Tag detection (DR1, PwrIdx0)	4.31	10	11.97	11.97
6 Tag detection (DR1, PwrIdx0)	6.63	10	18.41	3.07

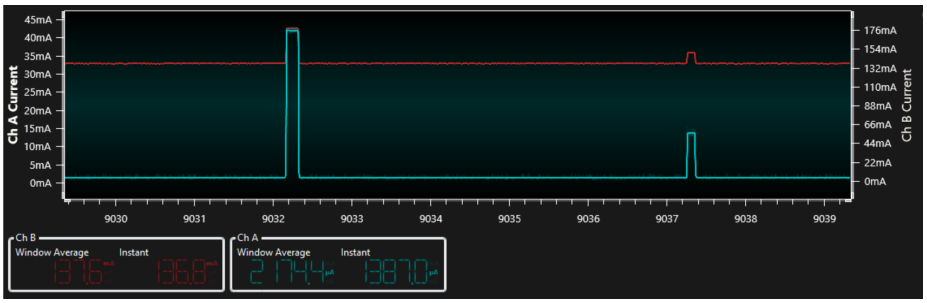
Table 8.3: Energy consumption of transmitting IoT messages using LoRaWAN. The data rate is given in parentheses to the left. All measurements are performed with a power index of 0. Results are read based on plots from Microchip studio.

Drift(us)	time(s)	Repetitions
999939	123	1
-31	140	1
-92	151	1
-122	236	1
-122	244	1
-214	424	1
-214	484	6
-244	484	2
-244	490	1
-244	544	8
-244	553	1
-275	484	1
-275	544	4
-275	576	1
-458	964	1
-31	140	1
-215851	389	1

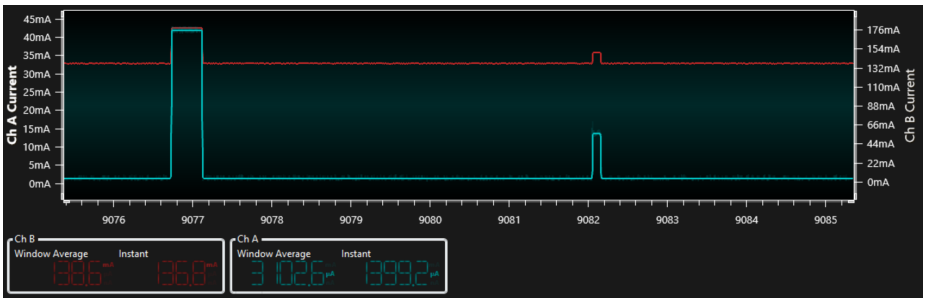
Table 8.5: Local time drift without RTC frequency estimation



(a) 1 Tag detection



(b) 10 Tag detections



(c) 33 Tag detections

Figure 8.6: LoRaWAN current measurements of tag detection transmissions. Tag detections are confirmed received by the gateway at the receive window.

Figure 8.7 shows the measured drift of local-time from one GPS update to another, using the time pulse from the GPS to synchronize the RTC and local time of the nRF9160. The RTC time pulse is used to correct the local time every second between GPS time pulse updates. The red line is a linear regression of the measurements. Two outlying measurements of 215 851 us in 389 s and 999 939 us in 123 s have been excluded, as they are from the startup of the system. These outliers are believed to be because of the GNSS module getting a proper time fix and due to UTC corrections transmitted every 12.5 min from the GPS satellites (here resulting in a 1 s correction of GPS time).

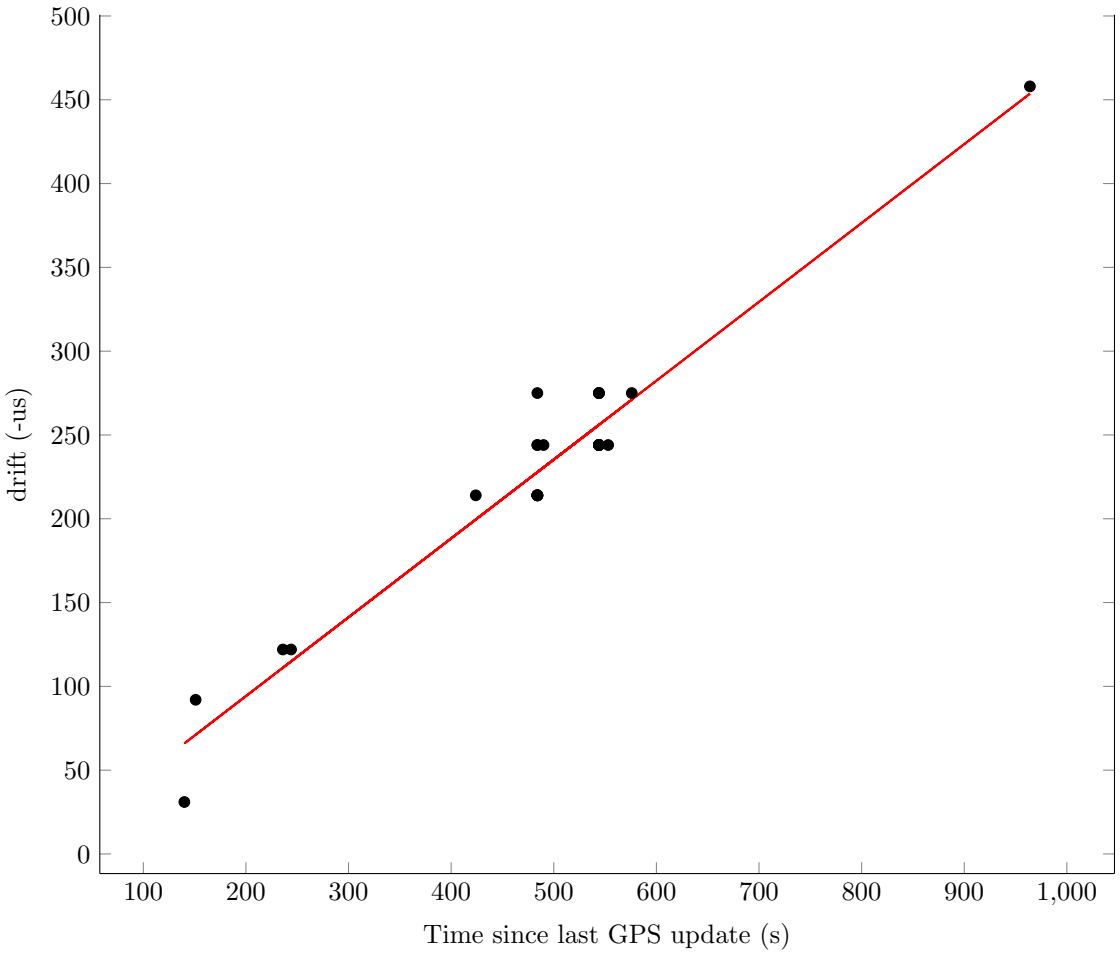


Figure 8.7: RTC drift without frequency estimation. The red line is a linear regression of the points. The time between GNSS updates is dependent on the previous drift, where observed drift and time since the last GNSS update is used to calculate how long the system can run on local time before needing a new GNSS update. This time is halved to make certain constraints are met. To avoid inaccurate drift readings, time is not set less than 2 min. The outlier of 458 us in 16 min and 4 s is due to a measurement of 31 us drift in 122 s. Outliers of 999939us and -215851us are excluded.

Number of tag detections	Time on air [s]	Throughput [Tag detections/minute]
Single Tag detection (DR5)	0.1	600
10 Tag detection (DR5)	0.2	3000
33 Tag detection (DR5)	0.4	4950
Single Tag detection (DR3)	0.4	150
10 Tag detection (DR3)	0.5	1200
Single Tag detection (DR1)	0.8	75
6 Tag detection (DR1)	1.5	240

Table 8.4: LoRaWAN throughput measurements. Measurements are read from power analyzer plots and are hence not very precise. Tag detections per minute is assuming the LoRa device can transmit all the time. However, LoRaWAN typically has a maximum transmission duty cycle of 1%.

8.5.2 Local time drift with RTC frequency estimation

Drift(us)	time(s)
-31	68
-109	544
999830	843
-100	424
-143	724
-112	964
-105	1566
-59	3123
-27	6546
-306	35279
+191	29463

Table 8.6: Local time drift with RTC frequency estimation. Results are given in occurring order, with UTC correction somewhere after 10 min 12 s. A fault occured after 22 hours, resulting in the program to restart. When restarting, the program is disregarding previous calculated RTC drift. Hence, readings after this have been excluded.

The estimated frequency, being low pass filtered, need some iterations before stabilising. This will result in the time between updates being shorter in the beginning than after the system has run for a while. The system is only tested in relatively constant temperature environments.

8.5.3 Time synchronization

As described in 8.1.2 a comparison between tag detections using the SLIM module was to be compared to tag detections using the cSLIM module. After connecting to the LoRa-GW, no tag detections or other status messages were received by the SLIM module. Debugging the SLIM module has not been prioritised. Hence no time synchronization comparison results are made. However, it has been verified that the TBRs time is set, both by receiving acknowledgement messages and observing its LED.

The time synchronization goal is therefore seen as partially fulfilled. Due to antenna delay to the

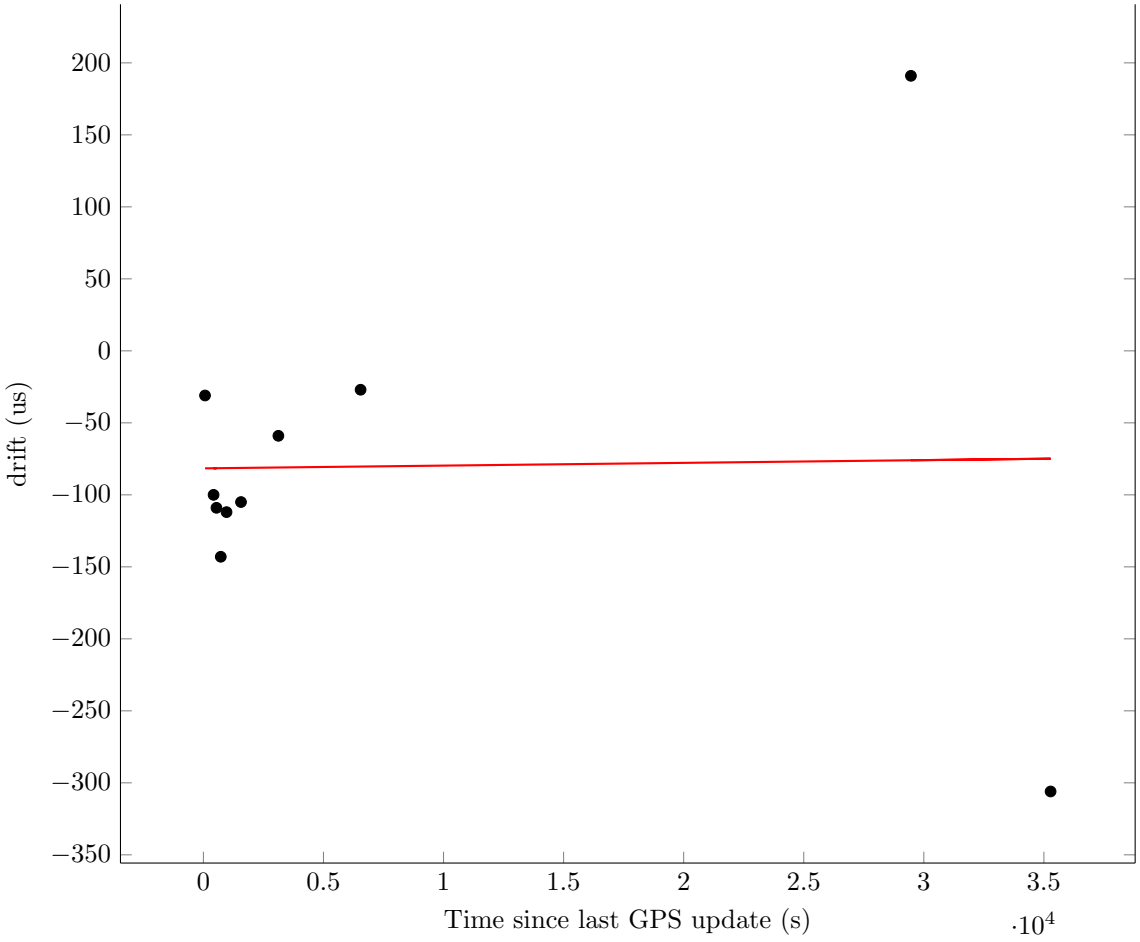


Figure 8.8: RTC drift with frequency estimation. The UTC update outlier of 999830 us have been excluded.

GNSS module, the tag detections might be skewed compared with the SLIM module detections. However, this can be fixed by setting the antenna delay of either the SLIM or cSLIM GNSS module to a non-zero value, dependent on which is before the other.

8.6 DUNE bridge

At the end of the project, the DUNE bridge connects to the MQTT broker and subscribes to a topic for raw IoF messages. The MQTT broker hostname, port, user credentials and topic is configured through a .ini file, standard for tasks in the DUNE framework. The IMC::FishTag and IMC::TBRSensor messages are dispatched to the network. This enables the bride to run directly on a TCP-IP connected vessel or at a dedicated server forwarding the messages through other DUNE transports. The DUNE bridge does not support JSON-packed messages from deployed LoRa-Gateways. Before location information of a given TBR sensor is received by the DUNE Bridge, the IMC messages will be dispatched with longitude and latitude 0. Also, as the IoF message format does not support transmission frequency and receiver memory address, these fields are left empty(0).

The client can only subscribe to a single topic. However, if more topics are required, multiple instances of the DUNE bridge can be run simultaneously on the same DUNE network. Figure 8.9 and 8.10 show example output.

```
{
  "abbrev": "TBRFishTag",
  "timestamp": "-1",
  "src": "65535",
  "src_ent": "255",
  "dst": "65535",
  "dst_ent": "255",
  "serial_no": "632",
  "unix_timestamp": "1623665464",
  "millis": "514",
  "trans_protocol": "3",
  "trans_id": "1731",
  "trans_data": "175",
  "snr": "45",
  "trans_freq": "0",
  "recv_mem_addr": "0",
  "lat": "2098438",
  "lon": "23119659"
}
```

Figure 8.9: DUNE Bridge IMC::FishTag message A IMC TBRFishTag is message dispatched for each fish tag observation sent from the cSLIM module through the MQTT broker to the DUNE Bridge.

```
{
  "abbrev": "TBRSensor",
  "timestamp": "-1",
  "src": "65535",
  "src_ent": "255",
  "dst": "65535",
  "dst_ent": "255",
  "serial_no": "632",
  "unix_timestamp": "1623665658",
  "temperature": "18",
  "avg_noise_level": "15",
  "peak_noise_level": "42",
  "recv_listen_freq": "35",
  "recv_mem_addr": "248"
}
```


Figure 8.10: DUNE Bridge IMC::TBRSensor message. A IMC TBRSensor is message dispatched for each TBR sensor frame sent from the cSLIM module through the MQTT broker to the DUNE Bridge.

8.7 Validation of cSLIM module

The following evaluates the performance of the cSLIM module with respect to the requirements set at the start of the development process. The requirements are given in appendix B.

8.7.1 Underwater Acoustic Receiver(UAR)

Functional requirements

The system uses a TBR 700 RT acoustic receiver from Thelma Biotel, communicating with the cSLIM module using the duplex RS485 protocol. The TBR 700 has proven to work as expected, both here and in previous SLIM applications [33] [14]. The time of the UAR is synchronized with the cSLIM module. However it is not validated that the time is set correctly. The UAR transmits sensor data to the cSLIM module.

8.7.2 Cellular SLIM module

Functional requirements

cSLIM-FR.2 – cellular IoT communication

In theory, NB-IoT has a long range, up to 10 km, to support use at a distance from base stations, however this has not been verified during this thesis. Suitability of datarate compared to operation conditions and data volume is further discussed in 9.3.

Nordic Semiconductor develops the nRF9160 modem software. This should ensure that the modem is using radio frequency bands in accordance with ITU and local regulations. The cSLIM module is sending status messages to the back-end at regular intervals of 10 minutes. The power efficiency of NB-IoT is further discussed in 9.3

cSLIM-FR.3 – LoRa communication

The power efficiency of LoRa is further discussed in 9.3. In theory, LoRa has a long range, up to 5 km in urban areas and 15 km in line of sight applications, however this has not been verified during this thesis. Suitability of datarate compared to operation conditions and data volume will be further discussed in 9.3. As the LoRa module is programmable, the user must make sure its application uses radio frequency bands in accordance with regulations. However, the LoRa RN Parser currently used on the module have built-in support for this.

cSLIM-FR.4 – Position awareness

The cSLIM module can acquire its position using GPS. Position and metadata (signal losses, dilution of precision, and the number of satellites) are transmitted to the back-end every 10 minutes as part of the cSLIM status message. On position loss, the system is still operational, using its last position and time-synchronization. However, this will reduce the usefulness in precise timing and positioning operations.

cSLIM-FR.5 – Timing accuracy:

Based on results given in section 8.5.1 the local time of the cSLIM module has proven not to drift more than 500us, except on startup before time is properly synchronized. However, it has not yet been verified that the time in the acoustic receiver is properly synchronized. Also, antenna delay has not been compensated for and must be done before using SLIM and cSLIM buoy controllers in the same deployment. The timing error is sent in its own message, with the time since the last synchronization.

cSLIM-FR.6 – Offline storage

The cSLIM module has support for saving logged data offline to persistent storage. As the FRAM chip was damaged using the soldering iron, only the uSD card is working on the prototype. However PCB design regarding the FRAM is verified. Implementing offline logging has not been prioritised and is left as future work.

cSLIM-FR.7 – Usability

The system conveys its state through LPWAN. LEDs ensure the operation can be observed at a distance. However, the LEDs are energy demanding and should be turned off in long-time deployments. The display shows information about LoRa, LTE, GNSS and TBR state and time synchronization.

cSLIM-FR.8 – Self-optimization during operation

The system chooses optimal parameters for ensuring time synchronization and power savings without any user interaction. The LTE modem is set to use eDRX. Using adaptive data rate, the LoRa device is acquiring ideal parameters through the LoRa-gateway. The GNSS module and other modules as the RS232 interface are put to sleep when not used.

cSLIM-FR.9 - Fault tolerance

Fault detection and recovery are mostly handled by the Zephyr RTOS, detecting faulty bus communication, illegal memory accesses and system freezes. Zephyr also uses the watchdog timer to reboot the system if needed. In addition, asserts are added to mutex acquisitions, throwing an error to the RTOS if deadlocks are detected. The module can log data locally through the uSD card. However, this has not been implemented.

cSLIM-FR.10 – Other hardware requirements

The module has antennas for c-IoT applications through the nRF9160-DK and LoRa and GNSS antenna connectors. In addition, the module has connectors for the acoustic receiver, programming and debugging. The module is equipped with an unused I2C and SPI header for the ease of future adaptions. The module is programmable via SWD and the nRF9160-DK USB connector through the Segger J-Link on board. This also provides debugging functionalities.

Non-functional requirements

cSLIM-NFR.1 – battery life:

The current prototype uses a lot of unnecessary energy because of the nRF9160-DK and some design flaws on the shield. With some further adaptions a standalone cSLIM module should have a long battery life for unattended installation in remote fjords and waters.

cSLIM-NRF.2 – Code quality:

Consistent variable and function naming have been attempted, mostly using lowercase letters and underscores. Most functions available through header files have been described using doxygen format [9], some more descriptive than other.

cSLIM-NRF.3 Code maintainability:

The code is divided in folders, each having a specific layer of functionality. The Zephyr RTOS is a bit heavy to get started with. However, by dividing the software into tasks and adding comments, the application should hopefully be easy to understand and maintain.

8.7.3 nRF9160-DK cSLIM shield requirements

Functional requirements:

The shield fit perfectly on top of the nRF9160-DK. Together with the DK it contains all peripherals and connectors needed to realise the cSLIM module requirements

Non-Functional requirements

The shield is easy to mount and de-assemble on the nRF9160-DK

8.7.4 nRF9160 cSLIM standalone module (cSLIM-SA) requirements

As the standalone cSLIM module have not yet been developed, its requirements are not discussed.

8.8 Validation of DUNE bridge

The following evaluates the performance of the DUNE bridge with respect to the requirements set at the start of the development process. The requirements are given in appendix B.

Functional requirements

The DUNE bridge is able to receive the fish-tag and sensor data from the cSLIM module, converting the data to previously defined IMC::FishTag and IMC::TBRSensor messages, before dispatching them to the DUNE network. The user can select a suitable transport (UDP, TCP ...).

Non-Functional requirements

The bridge can be deployed as part of an existing DUNE task, only requiring to add MQTT libraries to the project. How to do this is described with a readme.md available with the source code. To the best of the programmers' knowledge, the bridge has been written using DUNE conventions and should be easy to maintain.

9 Discussion

9.1 cSLIM hardware performance

9.1.1 Effect of including RTC

With a 1.5 PPM drift, the RTC is drifting 0.1296 seconds per day. With time constraints of 500us from UTC and using no compensation, the maximum time between two GPS updates is 5.55 minutes. Tests performed as part of this thesis shows the drift to be even smaller, with times between updates of 7 minutes to 8 minutes at roughly 250 us drift. This equals a drift of 0.60 PPM and 0.52 PPM, respectively. Using frequency estimation allowed the GNSS module to sleep for up to almost 10 hours with a drift of 300 us. This gives a drift of 0.0083 PPM for the local time, which is very low. In comparison, Rundhovde [33] states the SLIM module to wake its GNSS module every 60 to 220 seconds to correct its local time.

On the other hand, these tests have only been performed indoors with an approximately constant temperature. Rundhovde [33] based his correction of the local time in the SLIM module by measuring frequencies for different temperatures . Although the RTC being temperature-compensated, it is believed that the frequency will vary depending on temperature, although having a PPM of less than 1.5 for the temperature range given in section 5.2.2. As the outdoor temperature changes during the day, results might not be as good in field deployment, forcing the GNSS module to wake up more frequently. Still, even with a PPM of 1.5, the system can wake up less frequently than the SLIM module. The GNSS module uses much energy while active, and the RTC is very energy efficient, using 160nA at 3V on average. The solution should therefore reduce the overall energy efficiency.

As a side note on the RTC performance, the suggested RTC solution could also benefit resource-constrained sub-sea vessels needing precise timing on board. With reduced time drift, vessels can continue sub-sea operations at longer intervals before returning to the surface to calibrate their local time. As sea temperature is relatively constant during operation, keeping the RTC at a constant temperature should be possible, where results have shown the solution to work quite well. Increasing the interval between updates from minutes to hours will open for many operations. Today, precise atomic clocks are available. However, they are in a completely different price range and operate at a considerably higher energy budget.

The GNSS update interval and estimated frequency are low-pass filtered to avoid a single measurement throwing the calibration off course. The filter parameters might have to be adapted to suit environments with temperature oscillations.

9.1.2 Static energy consumption

As the nRF9160-DK includes many electronic components in addition to the nRF9160 that could not be fully disabled, it is drawing much power. This makes the solution with the DK and the shield not applicable for a final device.

Also, as per the first version, the shield has some unnecessary components for the application. For instance, the LLC in front of the Display. Also, due to the lack of GPIO pins, some of the LEDs and buttons are connected with other components. For instance, this is forcing the blue LED to be on while using the LoRa module. This adds a milli-Ampere to the power consumption.

As some of the components and the nRF9160 are supporting 1.8 V TTL while running on 3.0 V, reducing the voltage level might save static power consumption. On the other hand, some components require higher voltages, resulting in extra logic and two LLCs: One for 1.8 V and one for 3.0 V. Therefore, the energy savings of introducing 1.8 V voltages must at least make up for the extra energy drawn from additional components as well as additional components costs and hardware complexity.

9.2 Use of a RTOS in cSLIM module

Using an RTOS has the advantage of dividing the software into tasks that are scheduled periodically. Zephyr, being a real-time operating system, schedules the tasks to meet its timing constraints. That being said, by setting a thread to sleep for a given time, Zephyr will wake the task after the time has run out. It might, therefore, not be suitable for a hard real-time constrained system. However, it works well for near real-time applications such as the cSLIM buoy controller.

With built-in FIFOs, message queues, watchdog, peripheral drivers and communication libraries, it reduces the programming load of the user compared to bare bone register configuration. At least after getting familiar with the operating system and toolchain. Also, the fault detection and correction mechanisms restarting the system are helpful for a device being deployed for more extended periods. During that time, errors will happen, and the system must be able to revert to a working state by itself.

9.3 Pros and cons of NB-IoT and LoRaWAN in IoF application

Data transfer to the mainland is an energy-demanding part of the system. It is, therefore, essential to gain insight into relevant LPWAN technologies and their usefulness in the IoF application. As part of this thesis, two popular technology standards are investigated, the NB-IoT and LoRaWAN. The following subsections discuss the advantages and disadvantages of the two.

9.3.1 Spectral efficiency

The frequency spectrum where radio transmission is allowed is often a constrained and valuable asset. Hence, it is essential to ensure high spectral efficiency to ensure throughput to facilitate as many devices as possible. However, the spectral efficiency might not be as crucial for protocols using license-free frequencies in the ISM-bands as for cellular networks and operators. For the latter, frequency spectra are expensive, hence increasing costs for operator and end-user. 3GPP Rel. 14 and Rel. 15 focuses, among other things, on improving the spectral efficiency of NB-IoT.

As shown in equation 7, the LoRaWAN spectral efficiency is dependent on the spreading factor and coding rate. A higher spreading factor, typical for long-distance transmissions, results in worse spectral efficiency. Therefore, spectral efficiency might be an issue for multi-buoy deployments, allowing a maximum of 75 to 240 tag detections per minute per channel at the most significant spreading factor (SF12).

For IoF applications in remote areas, spectral efficiency might not have a significant impact. However, as more devices are becoming connected in factories, homes, and cities, this might be a factor to consider for fjord deployments, at least near urban environments. Poor spectral efficiency might result in increased transmission collisions, increasing transmission energy and reducing the buoy lifetime.

9.3.2 Distance to gateway/cell tower

Tests including distance and power measurements have not been performed as part of this thesis. The following is therefore based on theoretical aspects of the protocols and results by others researching the topic.

Table 2.1 shows the NB-IoT and LoRa technologies to have approximately the same coverage of around 165 dB. This implies the technologies to support close to the same distance to its base stations. However, energy efficiency with respect to distance might diverse for the technologies.

As shown from the Shanon-Hartley theorem, the information rate and link capacity decrease if the Signal to Noise Ratio decreases, as is the case for longer distances. LoRa solves increased distance by upping its spreading factor, resulting in a longer transmission time and increasing its

transmission power. NB-IoT increases its ECL level that also increases transmission power, as well as the delay time and the number of retries before cancelling the transmission. Yang et al. [21] shows that an end device will suffer from high energy drain in signal-limited environments where ECL2 is used. Measurements by Fovios et al. [21] show the ECL2 class being used for approximately 30% of samples at bad signal conditions, where the rest is divided between ELC0 and ECL1. Results are dependent on device and network operator conditions. For instance, does one operator not enforce any quiet period between each idle paging while another operator is. This is severely impacting energy consumption.

It is no surprise that energy consumption increases for both technologies when the distance to the base station or gateway increases. Field tests should be performed further to assess the energy consumption with respect to distance and technology.

9.3.3 Energy per bit

The LoRaWAN and NB-IoT/MQTT communication protocols are pretty different concerning header overlay. A LoRaWAN header and MIC has a size of 13-28 bytes, dependent on frame options. An MQTT header can be as little as two bytes. However, the optional header includes the MQTT topic as UTF8, needed to publish to the topic, and some other information. The header will also be longer for messages containing more than 256 bytes of data. As the MQTT message is packed in a TCP frame with an overhead of 20-60 bytes, typically 40 bytes, it is likely to include more overhead than a LoRaWAN message. The MQTT topics should be kept short to reduce the header.

The goal of the cSLIM application, concerning LPWAN, is to transfer tag detections, system status and TBR status in a reliable and energy effective way. Rather than looking at the energy per transmitted bit in total, it is therefore beneficial looking at the energy per transmitted tag detection and status message. As a status message is approximately the same size as a tag detection, compared to overhead, only tag detections have been investigated.

The header overlay reduces the energy efficiency of transmitting a few tag detections simultaneously, compared with transmitting in bursts. For NB-IoT a single tag detection is measured to consume 18.8 uAh if transmitted alone. If transmitted in bursts of 10 messages, the energy required per detection is 12.2% of that, while for 33 tag messages, it is 4.7% of transmitting detections one at a time. Similar results are shown for LoRaWAN. At data rate 5 (DR5), a single tag detection costs 1.17 uAh, while sending 10 and 33 at a time reduces the energy per tag detection to 19.7% and 12.8%, respectively.

An alternative to using MQTT for NB-IoT is using CoAP. With a smaller message header and using UDP headers that are smaller than TCP headers, the overhead of the transmission will be reduced. On the other hand, using CoAP require creating an additional bridge to the MQTT broker.

9.3.4 Battery and network lifetime

In the IoF application, the device with the shortest lifetime is likely the end device at sea, either the buoy controller or the acoustic receiver. Battery and network lifetime is therefore assumed to go hand in hand. Therefore, increasing the battery lifetime of the buoy controller up to the lifetime of the acoustic receiver (up to seven months) should be prioritized.

Retaining its connection with the MQTT broker, the nRF draws an average of 5.4 mA for ten seconds. With a period of 60 seconds, this adds an average consumption of 0.9 mA. This is a significant amount, reducing the operational lifetime by 15-20%. It is, however, possible to increase the interval up to one hour, making the energy consumption close to neglectable. Alternatives would be using CoAP or MQTT-SN(sensor network), not using keep-alive transmissions.

Using NB-IoT without eDRX or PSM, the device is idle paging every second, listening to the network for packages (see the spikes in figure 8.3a). In eDRX, the idle current is reduced by a

factor of four. However, the device will periodically wake up listening to the network, such that the total effect of PSM is somewhere between that and idle paging. The exact number is dependent on eDRX parameters. eDRX and PSM will, naturally, positively impact the power savings of the NB-IoT connection. However, it will also be affecting the near real-time performance of the system, as the period between IoF updates might increase, especially when using PSM.

Using LoRaWAN, the device does not need to retain its connection and, using class A, it only listens to the network after transmitting data. This helps saving energy.

9.3.5 Performance in low activity areas

In low activity areas, tag detections occur less frequent. This result in fewer transmissions, with TBR and system status messages sent periodically as the major payload.

Using mode A, a LoRa device will only consume power when having something to transmit. This gives an advantage in low activity areas over NB-IoT, as the system does not need to use power for the LPWAN connectivity if not having something to send.

As described above in section 9.3.3, headers add much overhead for both NB-IoT/MQTT and LoRaWAN. In low activity areas, where tag detections are few, one must weigh the near-real-time goals towards the energy savings of withholding a transmission until more tag detections are made. This is also applicable when using LoRaWAN with a low data rate and power index, where the cost of transmitting one byte is high. For example, using DR1 and power index 0, the cost of sending a single tag detection alone is more than four times higher than waiting until six detections are made.

9.3.6 Performance in high activity areas

There will naturally be more tag detections in high activity areas, resulting in the delay/energy trade of not being quite as important. Here, it might be more valuable that the LPWAN connection can support a higher data rate. A device on a LoRaWAN is typically allowed to transmit 1% of the time to avoid traffic fluctuation and collisions. The WRL089u0 has been shown to withstand an average of up to 49 tag detections per minute, using messages with 33 tag detections each. Reducing the tag detections per packet will severely reduce throughput, with a maximum of 30 detections per minute in 10-tag-packages. These numbers are also assuming the transmissions to be performed at the highest data rate. At a minimum, in harsh conditions, the system will only be allowed up to 2-3 tag detections per minute, with six detections per packet. Using The Things Network, with a limitation of 30 seconds on-air per day, the system will be able to send up to 2475 tag detections per day in perfect conditions, or an average of 1.7 tag detections per minute. This is close to eliminating The Things Network for all IoF applications. It might still be applicable in standalone use with very few tags deployed or monitoring a single fish.

NB-IoT does not have the same time-on-air limitation. Instead, the network decides how often a device is allowed to transmit data. With a theoretical limit of 66 Kbps and packet size of 1600 bytes, it could withstand up to 68000 tag detections per minute in perfect conditions. This is assuming a TCP header of 40 bytes, an MQTT header of 10 bytes, IoF header of six bytes, fish tag detections being seven bytes and perfect transmission conditions. The result is also messages with 220 tag detections, which is a relatively high number of detections. The above estimate must therefore be seen as a theoretical limit for the number of detections.

Assuming a single acoustic receiver transmitting messages to the buoy controller using the TBR protocol, with one character every millisecond and roughly 44 characters per detection, a total of 1360 detections can be made per minute. This is also assuming the acoustic receiver can process as many detections, which is not likely. Furthermore, the acoustic channels used for transmissions limits the number of tag detections further. For instance, the S256 protocol has a theoretical limit of 11 messages per minute. With three parallel channels of the TBR 700 acoustic telemetry receiver, a maximum of 33 messages can be received every minute. Other telemetry receivers

might support a more significant number of channels. Anyhow, for high activity areas, the NB-IoT solution might have an advantage over LoRaWAN in respect to transmitting all tag detections.

9.3.7 Operation complexity and costs

NB-IoT utilizes cellular provider infrastructure, making it easy to deploy connected systems worldwide, as long as cellular networks are in reach by operators that facilitate the NB-IoT protocol. As of summer 2021, the coverage rate in Norway is relatively high. However there are still some blind spots where an NB-IoT enabled system will not be applicable, as seen in figure 2.9 and 2.10. Using NB-IoT will therefore not assure the system to be deployed all over. However, it will be usable in most places. Connecting devices on the NB-IoT network adds costs, dependent on the number of devices and transmitted data.

LoRaWAN, using the unlicensed ISM bands, has one of its advantages with the ability to place LoRa-Gateways where they are needed. Public subscription LoRaWAN is provided. However, no network with suitable coverage has been found. The Things Network coverage, as shown in figure 2.14, has gateways centred around larger cities and attractions, and with the uplink limitations provided in section 9.3.6 it is not a viable solution. The coverage of Altibox and Last_Mile has not been found, nor information on limitations to air time. Using LoRaWAN will, therefore, most likely include the cost of installing and operating LoRa gateways. This will add costs and deployment complexity that must be weighed against the cost of using a cellular network provider.

Buoy-deployment and maintenance are costly operations in marine environments. Therefore, operation costs must also factor in the battery life and replacement costs. Design and production costs of a long-battery-lasting buoy controller can compensate for reduced maintenance costs.

9.3.8 Summary of LPWAN technology in IoF

The investigation and results in this thesis are not sufficient to conclude one of the LPWAN technologies over the other for the IoF project in total. LoRaWAN tends to be a more energy-efficient solution, which suits well with other articles researched in this project. Also, ABI Research [31] states LoRaWAN being more cost-efficient than NB-IoT with a longer estimated battery lifetime. On the other hand, LoRaWAN, having lower throughput than NB-IoT, might not support as many tag detections. This can be problematic in high-activity areas. As no suitable LoRaWAN network operator has been found, it will likely add extra overhead in increased infrastructure.

NB-IoT has the advantage of having its infrastructure provided by the cellular network operators and supporting more tag detections than the acoustic receiver and sound channels can handle. On the other hand, it has been shown to have a worse energy per tag detection ratio compared to LoRaWAN. As accepted parameters regarding paging, eDRX and PSM are chosen by the network operator, the energy efficiency might vary between operators. As shown by Fovios et al. [21] there can be a lot to save in idle paging by choosing the correct operator. As LPWAN is a growing market, it is expected that operators continue improving their NB-IoT support. Continuously adding functionality and features, supporting more configurations, is expected to even the advantage of one operator over another. Anyhow, for an NB-IoT enabled device, a third-party cellular network operator and more frequent buoy maintenance might increase costs.

Both solutions have their strengths and weaknesses, and which LPWAN technology being best suited depends on the use case and deployment area. Further development and field tests should be performed before any final decision can be made.

9.4 cSLIM power consumption and operational life

As mentioned earlier in section 9.1.2, the prototype developed as part of this thesis assignment is not suited for deployment, as the nRF9160 development kit contains power-hungry components not needed for the application. With wlr089u0 on and the GNSS module off, the resulting current

of the shield was measured to 7.1 mA. When the LoRa module is on, two blue LEDs are also turned on. Based on calculations in 5.2.2 the LEDs are drawing around 1 mA each. With the WLR089u0 module is measured using roughly 1.4 mA, the rest of the shield has a static current consumption of 3.7 mA. This only considers static power, resulting in the device working on a single 36Ah battery for approximately 13 months.

The application itself will use energy as well. As is, the nRF9160 uses approximately 1.6 mA when registered on the NB-IoT network and connected to the MQTT-broker, without eDRX or PSM. This is also the case when running the cSLIM application. On connect, the power consumption of the nRF is in the scale of 36 mA. However, this is only ongoing for some minutes when starting the application. With the application constantly drawing at least $3.74(static) + 1.6(nRF) = 5.34$ mA, this will only reduce operational life by an hour or so and is seen as negligible for a system being deployed several months.

The same logic is used to argue that the initial period of the GNSS module searching for satellites can be neglected, only leaving the intervals of which time and position is updated. It is here assumed the GNSS module is drawing around 23 mA when active tracking, as given by the datasheet [46]. Using the sub-minute time between GNSS updates in the local time drift measurements, the GNSS module is normally active for a period of four seconds at a time. However, periods up to 42 seconds have been observed. This is natural as the time is dependent on GPS signals, and there might be periods satellites are out of reach. For an update period of four seconds every three to five minutes, as is expected without the RTC without frequency estimation, the GNSS module adds an average current consumption of 0.30 mA to 0.51 mA to the system. Using frequency estimation, the consumption might be even lower.

Further cSLIM hardware should be designed such that the LoRa module can be entirely shut off or set in deep sleep as it is unlikely that both NB-IoT and LoRaWAN is needed simultaneously. By only enabling one of the two, this should save 1.4 mA to 1.6 mA for the final application. By eliminating the extra components in a standalone module, and doing some small changes to the shield hardware, the result might be a quite energy-efficient device suitable to the application.

9.5 DUNE Bridge

The DUNE bridge prototype has been tested and verified to work as expected for the project and should be easy to set up for passing buoy detections to a DUNE-enabled system.

Limitations to the DUNE bridge
As the focus of the project was creating a prototype DUNE bridge for IoF messages from the cSLIM module, JSON-packed messages used by Rundhovde's LoRa Gateway [33] is not supported. It should not be difficult to add support for JSON packed messages by adding JSON support directly to the bridge. An alternative is making an MQTT client subscribe to the JSON-IoF topic and unpacking the messages before publishing them to another (or the same) topic. It should also be investigated if adding the JSON support is desired, as this is done to add LoRaWAN parameters to the message. If using NB-IoT this is highly useless. Also, the former IoF application by Kjelsvik uses the IoF message without JSON.

9.6 Goal and method

The goal of this project has been to develop a hardware platform for the IoF project that, in addition to being a working buoy controller prototype, gives the ability to evaluate LoRaWAN and NB-IoT as LPWAN technologies for the project. In addition, the goal has been to investigate the aforementioned LPWAN technologies and their suitability in the IoF project, and make a bridge for including IoF data to DUNE in near real-time.

Having a previously built SLIM prototype has been invaluable when designing new hardware, as suitable components have been researched earlier. This has reduced the time of design, component exploration and driver adaptions. On the other hand, better suitable components might be

launched since the design of the SLIM module. Even though some effort has been put into finding updated or better-suited products, some components might not have been found as the research have been more limited than expected without a previous prototype. Also, trusting an existing design is not always beneficial, as there might be flaws not yet detected or updated in the documentation. This was the case with the RS485 interface, where a former SLIM design interchanged the A and B line bias resistors. Trusting this design, together with the TBR sensor releasing the RS485 lines a short period before a new transmission, lead to problems for the nRF UART module. The added ability to disable the bias resistors through jumpers enabled the resistors to be flipped the correct way without assembling new hardware (see the yellow and orange wire in figure 7.1).

When ordering a few modules, PCB assembly is a costly affair when looking at the unit costs. It was therefore decided to order the PCBs from a professional PCB manufacturer before soldering by hand. In hindsight, ordering PCB assembly by the PCB manufacturer could have been worth it as numerous hours have been spent removing overflowing solder paste and debugging hardware (and software). In the end, problems are traced back to bad soldering. In particular, this has been chiefly due to small components with pins underneath the chip, i.e. the FRAM, Voltage regulator and RTC. However, solder paste has also short-circuited pins of the GNSS module, partly underneath the chip. As the RTC is not yet released to market per mid-August 2021, it must have been hand soldered anyhow. However, it would have saved time for the rest of the components. It seems close to luck that the hardware is working fine at the end of the project. Except for the de-soldered voltage regulator, FRAM being damaged to heat while soldering and the RS232 interface not being tested. For further development third-party PCB assembly should be considered, as well as using component evaluation kits to verify software design.

Hardware design, manufacturing and software development have been a large part of this thesis, including getting started with the Zephyr RTOS, device trees and other configurations. Getting a stable LoRaWAN connection has also taken more time than expected due to short distances to the LoRa gateway and mismatching protocol versions with join requests accepted by the gateway anyhow. In contrast, join accepts mostly was not accepted by the end device. Therefore, test results regarding LPWAN technologies is unfortunately quite limited in this thesis compared to what was initially planned.

The working buoy controller prototype and DUNE Bridge fulfil many of its requirements set at the start of the project. There are still work to be done. Future work is described further in chapter 11. Anyhow, the prototype is a good step towards a combined NB-IoT and LoRa buoy controller.

10 Conclusion

The buoy controller prototype developed in this project fulfils many of its requirements set at the start of the project. There are still work remaining. However, it is a practical and valuable step towards a combined NB-IoT and LoRa buoy controller based on the nRF9160 System in Package. The nRF9160 proves suitable for the project, especially with respect to integrating NB-IoT, and with respect to integrating a real-time operating system with embedded features.

Regarding time synchronization, adding an ultra-low-power, high-accuracy and temperature compensated RTC has shown momentum increasing the GNSS update interval. Furthermore, adding RTC frequency estimation improves the solution severely. However, tests in non-stable temperature environments are needed to verify the solution for real-life deployment scenarios.

NB-IoT and LoRaWAN both have strengths and weaknesses. The best suited LPWAN technology for deployment is dependent on factors as use-case, distance to the base station and maintenance costs. LoRaWAN has its main advantages in low energy transmissions and the possibility of deployment outside of cellular network coverage. On the other hand, NB-IoT has the advantage of supporting a high number of tag detection's and reducing deployment complexity by using existing cellular network infrastructure. Test results in this thesis do not cover field tests, which is advised for further improvements of the buoy controller and selection of the LPWAN technology in a deployment scenario.

The DUNE Bridge is ready to be deployed in a DUNE enabled system, passing IoF tag detections and TBR sensor data to the system.

11 Future Work

In short, future work can be summarised as:

- Revision of cSLIM shield design and creating cSLIM standalone module.
- Rewriting Device Tree to match new PCB design.
- Energy savings on local time calibration and GNSS module.
- Simplify GNSS time synchronization procedure.
- Verification of TBR time synchronization.
- Verification of RS232 interface.
- Finalizing implementation and verification of local log.
- Improvements of LoRaWAN module software (interrupt-based solution and add address switch).
- Improvements of NB-IoT communication.
- Extended evaluation of LoRaWAN and NB-IoT performance by field tests.
- Extended evaluation of RTC and local time performance by field tests.
- Rewriting the newest IoF application to accept non-JSON IoF messages, or adding JSON support to the cSLIM module and DUNE Bridge.
- Bug fixing

Details and suggested approaches are given in the following subsections.

11.1 Standalone cSLIM module

The cSLIM module made in this thesis is purely a prototyping device meant to develop software and evaluate the performance at a first level. For future work, the hardware and software should be further developed for creating a cSLIM module able to fulfil all requirements described in appendix B. Some key points for further hardware development are given in section 3.2. Also, see comments in the hardware schematics in appendix D.

Making a new hardware design should require the device tree to be rewritten. However, this is a small task.

11.2 Energy savings on local time calibration and GNSS module

For now, the wake interval of the GNSS module has been extended. However, the RTC updates the local time of the nRF9160 with a time pulse every second, driving the clock forward. As this requires the nRF to wake up every second, it is power consuming. For further improvements of the local time synchronization, increasing the period of RTC time synchronization should be investigated. With a period of 10 seconds, the energy usage of the RTC updates are reduced by approximately a factor of ten, neglecting any extra code being executed on the nRF to increase the interval. As the maximum period of the RV3032-C7 clock-out is 1 second, the nRF must enable and disable the RTC clock-out or its GPIO interrupt handler at requested intervals. An alternative approach can be using the RTC clock-out to increase a hardware counter and set a top value that triggers an interrupt at a requested interval.

11.3 GNSS time synchronization procedure

The procedure used to synchronise the RTC and local time with the GNSS module was designed for being used with TIM-TP messages from the UBLOX-m8n GNSS module, receiving timing data for the next time pulse before synchronising. However, readings of TIM-TP messages from the GNSS module showed the seconds to jump, i.e. giving x seconds, then x seconds again before x+2 seconds or x, x+1, x+3. The reason for this strange behaviour was not found, and the problem was solved by using the time from the NAV-PVT message. The NAV-PVT messages are containing timing information of the previous time pulse. Therefore, it should be possible to reduce the number of states in the synchronization STMs, especially in the GPS Task. On the other hand, the TIM-TP message contains more detailed information on the time pulse. As an alternative approach, the problem with jumping seconds could be investigated, and if solved, the synchronization rewritten to use TIM-TP messages.

11.4 Verification of TBR time synchronization

TBR time synchronization has been implemented. However, the synchronization has not been verified due to problems receiving fish tag observations from the SLIM module. This must be done before the system is applicable in a multi-buoy and biotelemetry deployment. An alternative to comparing timestamps with the SLIM module is, if more cSLIM modules are made, to compare multiple cSLIM modules against each other. However, this will not verify SLIM and cSLIM buoy controllers working together.

11.5 Verification of RS232

Although being a simple task, the verification of a working RS232 interface has not been prioritized and is left as future work. As the nRF9160 only has four peripheral drivers, where one is used for RS485 (uart), one for LoRa / logging (uart), one for SPI and one for I2C, there is no available uart peripheral driver available. Adding extra hardware for multiplexing one of the uart devices (i.e. RS232 and LoRa/logging) can be beneficial. However, only one of the two can be active at a given time.

11.6 Final implementation and verification of local log

The FRAM chip on the cSLIM prototype is not working, most likely due to overheating while soldering. The uSD card interface has been tested by writing to the card and reading the uSD card using a computer. However, logging tag detections, cSLIM status and TBR status to the uSD card or FRAM has not been fully implemented.

11.7 LoRa module improvements

For now, the LoRa parser software is running almost unchanged at the WLR089u0, with minor changes described in section 6.6. For future work, the 8-pin DIP switch should be included to change LoRaWAN device EUI without changing to the code quickly. Also, as the LoRa module is connected to the USB connector, support should be added for forwarding log messages from the nRF through the LoRa module. This might be useful while developing the software further. Alternatives are to include logging through RS232, using the uSD card or connecting a logic analyser with UART support, e.g. a Digilent Analog Discovery.

11.8 NB-IoT communication improvements.

To reduce the current consumption of LPWAN transmission using NB-IoT, further improvements should be investigated. For example, ideas are implementing an adaptive update interval dependent on the application type (single-deployment/multi-deployment), optimising eDRX, and possibly PSM mode. It should also be investigated if using CoAP and a CoAP to MQTT bridge is suitable for reducing header overlay, especially for deployment in low-activity areas.

11.9 Extended evaluation of LoRaWAN and NB-IoT performance by field tests

LoRaWAN and NB-IoT have only been tested in an urban environment, with short distances to gateways and base stations. Performing tests indoors should somewhat compensate for the short distance. However, tests should also be performed at sea, in areas where the system is later deployed.

11.10 Extended evaluation of RTC and local time performance by field tests

RTC performance and drift measurements have so far been performed in a fairly stable-temperature environment. The performance should also be tested with changing temperatures, as is expected in the deployment area, to validate the solution for real-life scenarios.

11.11 IoF Application, DUNE Bridge and JSON MQTT payload

The cSLIM application is sending IoF messages directly as payload in the MQTT messages. This is done such that the MQTT messages transmitted using LoRaWAN and NB-IoT are identical. While this is the format used by Kjelsvik according to project documentation [29], Rundhovde [[33]] placed the IoF messages in a JSON object with LoRa transmission parameters. If using the newest IoF application, either the application or MQTT payload must be adapted. As adding LoRaWAN parameters does not suit the MQTT messages transmitted using NB-IoT, the focus should be put on the application. This prevents making changes to the DUNE Bridge, only supporting raw IoF payload. However the Node-RED flow of deployed LoRa gateways must be updated. An alternative solution is using Kjelsvik's original IoF application. However, this also requires changes to the LoRa gateways. If transmission of LoRa parameters from the gateway is preferable, it should be separated to another MQTT topic to avoid unneeded processing at the DUNE Bridge.

11.12 Bug fixing

During RTC tests over extended periods, the application tended to restart once in a while. As the period between restarts was up to 22 hours, the problem has not been investigated. The system restarts by itself and is still operational. However, the estimated frequency resets, and the GNSS module is waking from a cold start. This increases power consumption and will leave the acoustic receiver not synchronized for up to 12.5 minutes after GPS fix.

Appendix

A Getting started with further development

The following sections explain how to get started with further development using the nRF Connect SDK

A.1 nRF9160

Toolchain

1. Download and install the `nRF Connect SDK` from `https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/getting_started.html`
2. Go to the toolchain manager in nRF Connect and download SDK v.1.5.1

Software

1. Download the cSLIM application from the GitHub repository in appendix C. Try to make the path short, for instance by adding it directly to the drive, for instance `C://cSLIM/`
2. In the nRF Connect application open the `Toolchain Manager`. Press the arrow next to the downloaded version and click `Open command prompt`.
3. Navigate to your desired location of the cSLIM application.
4. Open Visual Studio Code with `code .` It is reccomended to do this through the nRF Connect command promtp to resolve links to the nRF Connect SDK.

Upload to nRF9160-DK

1. Navigate to the project folder in the terminal opened through the nRF Connect Toolchain manager
2. Connect the micro-usb to the nRF development kit at the side of the robot (not the back).
3. Compile and upload to the nRF9160-DK with `west build b cSLIMns -p` and `west flash`

A.2 WLR089u0

Software and toolchain

1. Download and install the `Microchip Studio` from `https://www.microchip.com/en-us/development-tools-tools-and-software/microchip-studio-for-avr-and-sam-devices`

Software

1. Download the LoRa parser application from the GitHub repository in appendix C.
2. Navigate to and open the `APPS_ENDDEVICE_DEMO2.atsln` project file. This will start Microchip studio and show all project files.

Upload to WLR089u0

1. Connect a Atmel ICE to the SWD/JTAG header on the cSLIM shield. verify that the red and green lights on the Atmel ICE is showing (you need to power the cSLIM).
2. Flash the program through Atmel Studio. The blue light by the LoRa module should be illuminated (as long as the program is not changed). Try rebooting the cSLIM module if this does not work.

B System requirements

B.1 Underwater Acoustic Receiver(UAR) requirements

Functional requirements

UAR-FR.1a – Underwater acoustic detections: Detect acoustic signals from acoustic tags with high reliability and range, process the signals and transmit to the cSLIM module.

UAR-FR.1b – Time synchronization: To compute precise tag-locations, the acoustic receivers time must be synchronized with other cSLIM (and preferably also SLIM) modules).

UAR-FR.1c Underwater sensor data: Detect, process, and transmit sensor data, e.g. water temperature

UAR-FR.1d cSLIM Communication: The UAR must have a duplex communication with the cSLIM module for time synchronization and data transmission.

B.2 Cellular SLIM (cSLIM) module requirements

Functional requirements

cSLIM-FR.1 – UAR communication: See UAR-FR.1d.

cSLIM-FR.2 – cellular IoT communication

cSLIM-FR.2a – Power: The c-IoT communication must be power efficient.

cSLIM-FR.2b – Range: Have a long range to support use at a distance from base stations.

cSLIM-FR.2c – Data rate: Use a data rate suitable to operation conditions and data volume.

cSLIM-FR.2d – ITU and local radio regulations: Use radio frequency bands in accordance with regulations.

cSLIM-FR.2e – Heartbeat: Send status messages to the back-end at regular intervals.

cSLIM-FR.3 – LoRa communication

cSLIM-FR.3a – Power: The LoRa communication must be power efficient.

cSLIM-FR.3b – Range: Have a long range to support use at a distance from base stations or other buoys.

cSLIM-FR.3c – Data rate: Use a data rate suitable to operation conditions and data volume.

cSLIM-FR.3d – ITU and local radio regulations: Use radio frequency bands in accordance with regulations.

cSLIM-FR.4 – Position awareness

cSLIM-FR.4a – position acquiring: The cSLIM module must be able to acquire its position based on existing satellite networks.

cSLIM-FR.4b – position transmission: Position and metadata (signal losses, dilution of precision, and number of satellites) must be transmitted to the back-end.

cSLIM-FR.4c – operability on position loss: System must work with the position acquiring turned off and without having an accurate position. This will reduce the usefulness in precise timing operations.

cSLIM-FR.5 – Timing accuracy:

cSLIM-FR.5a – maximum timing error: The timing error should be less than 500us for use cases involving transmitter localization where precise time synchronization is required, for example when using multiple cSLIM modules in a receiver grid and/or mobile receiver platforms (USVs). For stand alone operation the timing error should be less than 500ms.

cSLIM-FR.5b – Back-end timing error awareness: The backend should be made aware of timing errors in the cSLIM modules.

cSLIM-FR.6 – Offline storage

cSLIM-FR.6a – Save logged data offline to persistent storage, e.g a uSD card or F-RAM chip.

cSLIM-FR.7 – Usability

cSLIM-FR.7a The system should convey its state without the need for debug and serial communication equipment.

cSLIM-FR.7b Show system status during booting, operation and failure in a way that can be observed at a distance.

cSLIM-FR.7c Use display to convey relevant information about state and working conditions (radio, network and peripheral connections, status and parameters)

cSLIM-FR.8 – Self-optimization during operation

cSLIM-FR.8a – Choosing optimal parameters: The system should choose optimal parameters without user-interaction to minimize the power usage while maintaining stable radio communication and required timing accuracy.

cSLIM-FR.9 - Fault tolerance

cSLIM-FR.9a – fault detection: The module must be able to detect faults in the different parts of the system and, if possible, forward them to the user.

cSLIM-FR.9b – fault recovery: The module must be able to recover from software errors, for instance by resetting the cSLIM module

cSLIM-FR.9c - local data logging: The module must be able to log data locally, that is not corrupted on system reset or power loss. See cSLIM-FR.5.

cSLIM-FR.10 – Other hardware requirements

cSLIM-FR.10a – Antennas: The module must have antennas for c-IoT applications LoRa and GNSS, either internal or external.

cSLIM-FR.10b – Connectors: The module should have connectors for external GNSS, LoRa and c-IoT antennas as well as connectors for the acoustic receiver, programming, debugging etc. The module should also be equipped with extra connectors for the ease of future adaptions.

cSLIM-FR.10c- Programming: The module should be programmable via SWD or another suitable programming interface.

cSLIm-FR.10d – Debugging: The module should have a suitable debugging interface

Non-functional requirements

cSLIM-NFR.1 – battery life: The module should have a long battery life for unattended installation in remote fjords and waters. Ideally up to seven months, as the acoustic receivers.

cSLIM-NRF.2 – Code quality: Consistent variable and function naming, descriptive comments

cSLIM-NRF.3 Code maintainability: Well defined code structure, modulation and function scopes.

B.3 nRF9160-DK cSLIM shield requirements

Functional requirements:

cSLIM_SHIELD-FR.1 – The shield must be designed such that it, In association with a nRF9160-DK and cSLIM application source code, satisfies the cSLIM functional requirements.

cSLIM_SHIELD-FR.2- The shield must fit on top of the nRF9160-DK

cSLIM_SHIELD-FR.3 – The shield must contain peripherals and connectors needed to realise the cSLIM module requirements

cSLIM_SHIELD-FR.3a – The shield must have the following peripherals on board:

- uSD card reader
- offline storage
- Status LEDs (Status_green, status_red, status_gps, status_cellular)
- Configuration switches
- Low power display
- Battery measurement circuit
- RS485 interface module
- RS232 interface module
- Powerdownable LoRa module F.eks RN2483 eller WLR089U0 (Microchip) Est.kost 150,- pluss antenne

cSLIM_SHIELD-FR.3b – The shield must have peripheral connectors to:

- Accoustic receiver
- SPI interface for debugging/extra connectors
- I2C interface for debugging/extra connectors
- UART interface for debugging/extra connectors
- Battery connector
- RS485 interface debugging
- RS232 connector

- LoRa antenna connector
- USB connector(?)

(in addition the nRF9160-DK have connectors for external antennas (GPS and NB-IoT/LTE-M), power supply and programming/debugging(interfaces [Embedded Segger J-link])

Non-Functional requirements

cSLIM_SHIELD-NFR.1 – The shield must be easy to mount and de-assemble on the nRF9160-DK

nRF9160 cSLIM standalone module (cSLIM-SA) requirements: **Functional requirements**

cSLIM_SA-FR.1 – The standalone cSLIM module (cSLIM-SA) must satisfy the cSLIM and cSLIM_SHIELD functional requirements

cSLIM_SA-FR.2 – The module should consist of a single PCB with all components and connectors necessary to fulfill the functional requirements in cSLIM_SA-FR.1.

cSLIM_SA-FR.3 – The module must have the following peripherals on board (in addition to the cSLIM-SHIELD):

- Voltage regulator
- Reset button
- Antenna matching networks for GNSS, LoRa and cellular antenna
- power multiplexer to be powered over USB when available.

cSLIM_SA-FR.4 – The module must have the following connectors (in addition to the cSLIM_SHIELD):

- external GPS antenna
- external cellular antenna (NB-IoT/LTE-M)
- external LoRa antenna
- power supply (battery)
- programming/debugging.

Non-functional requirements

cSLIM_SA-NFR.1 – The cSLIM module should have a form similar to the current SLIM module to ease replacement in a bouy.

B.4 DUNE bridge requirements

Functional requirements

DUNE_BR-FR.1 – Input – The DUNE bridge should receive the fish-tag and sensor data from the cSLIM module

DUNE_BR-FR.2 – output - The DUNE bridge should put the IMC messages, resulting from cSLIM input, to the DUNE network through a suitable transport (UDP, TCP ...)

DUNE_BR-FR.3 – Message Format - The DUNE bridge must convert fish-tag observations and acoustic receiver sensor messages into IMC messages that can be transported into DUNE.

DUNE_BR-FR.3a - The fish-tag observations should use the already defined IMC "Fish-Tag" message format

DUNE_BR-FR.3b - A suitable IMC message should be defined/used for acoustic sensor data

Non-Functional requirements

DUNE_BR-NFR.1 - The bridge should be easy to set up and maintain

C Digital appendixes

Application source code and project files are available on Github.

C.1 Application source code

cSLIM bouy controller source code

`https://github.com/eivinhj/cSLIM_buoy_controller`

cSLIM nRF9160-DK board controller source code

`https://github.com/eivinhj/cSLIM_nRF9160-DK_board_controller`

LoRa module source code

`https://github.com/eivinhj/cSLIM_WLR089u0_LoRa_RN_Parser`

DUNE Bridge

`https://github.com/eivinhj/cSLIM_DUNE_bridge_2/tree/master/src/Transports/MQTT/SLIMMessageBridge`

C.2 PCB project files

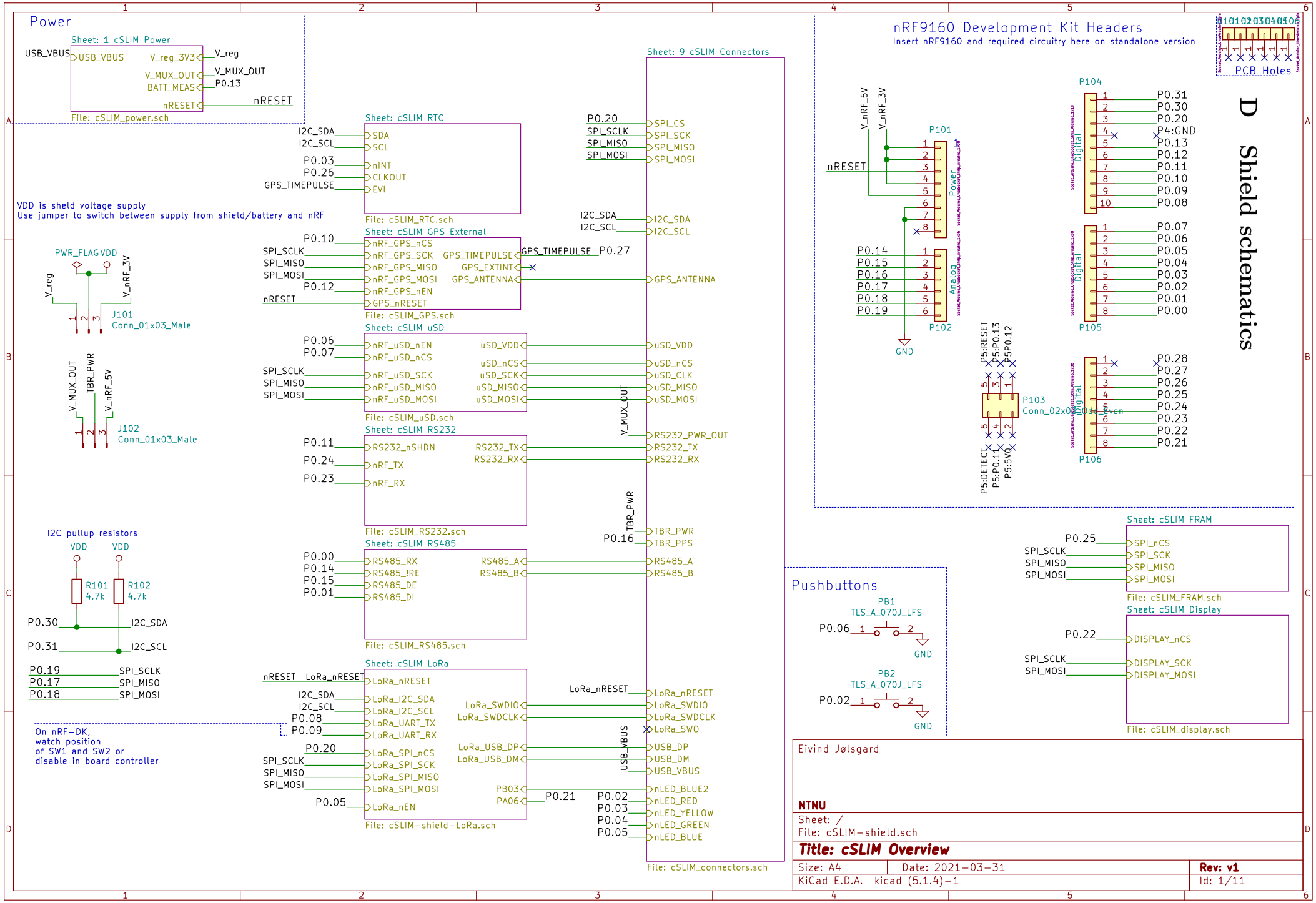
cSLIM shield kiCad project files

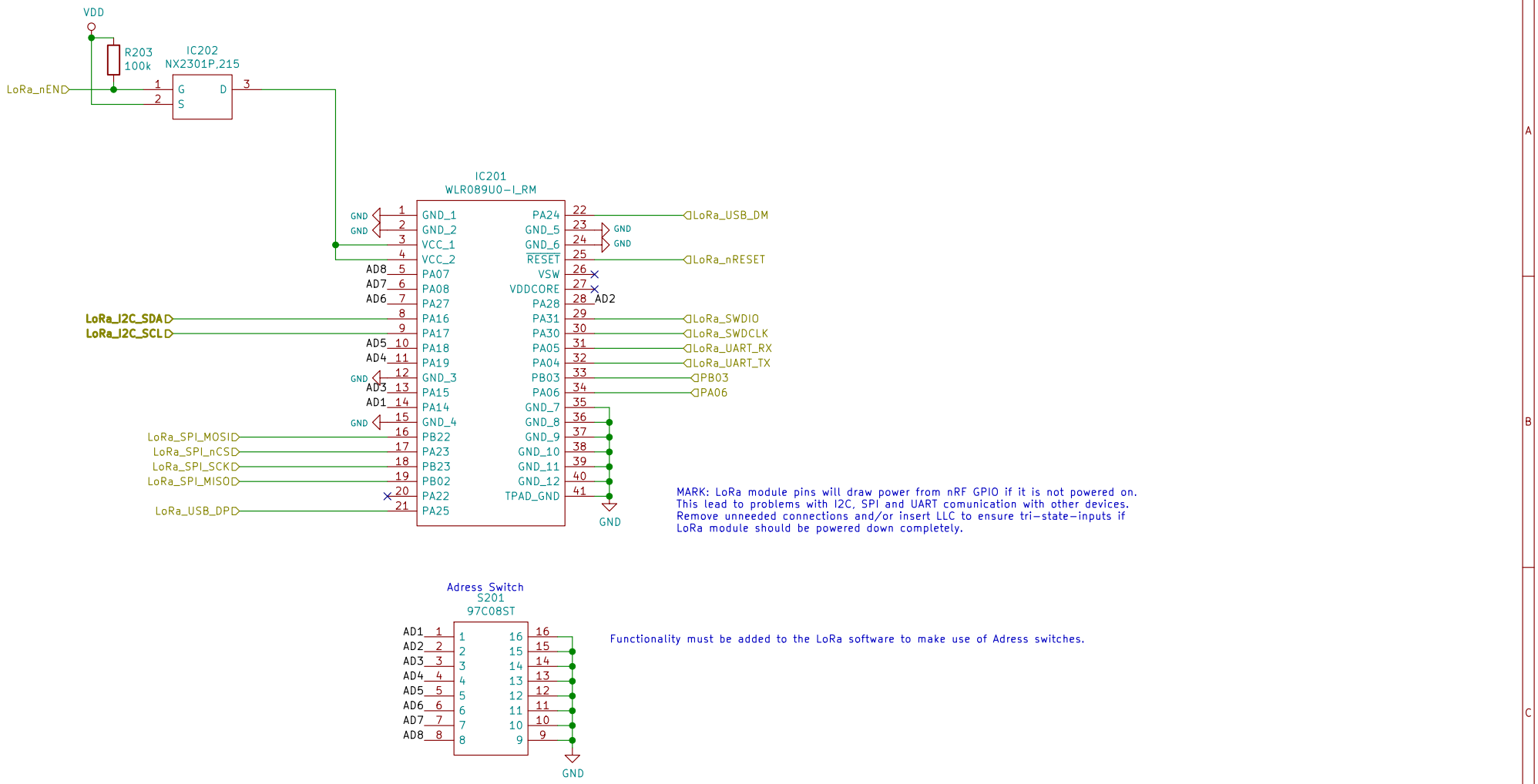
`https://github.com/eivinhj/cSLIM_nRF9160-DK_shield`

C.3 Test Specifications

Initial test specification (OneDrive)

`https://1drv.ms/u/s!AmerrbhDji3ziuJGxTTHqWwdbtuehA?e=qA4d99`


Power
Sheet: 1 cSLIM Power
USB_VBUS
V_reg_3V3
V_reg
V_MUX_OUT
BATT_MEAS
P0.13
nRESET
File: cSLIM_power.sch
Sheet: cSLIM RTC
I2C_SDA
I2C_SCL
P0.03
P0.26
GPS_TIMEPULSE
SDA
SCL
nINT
CLKOUT
EVI
File: cSLIM_RTC.sch
Sheet: cSLIM GPS External
P0.10
SPI_SCLK
SPI_MISO
SPI_MOSI
P0.12
nRESET
nRF_GPS_nCS
nRF_GPS_SCK
nRF_GPS_MISO
nRF_GPS_MOSI
nRF_GPS_nEN
GPS_nRESET
GPS_TIMEPULSE
GPS_EXTINT
GPS_ANTENNA
GPS_TIMEPULSE
P0.27
File: cSLIM_GPS.sch
Sheet: cSLIM uSD
P0.06
P0.07
SPI_SCLK
SPI_MISO
SPI_MOSI
nRF_uSD_nEN
nRF_uSD_nCS
nRF_uSD_SCK
nRF_uSD_MISO
nRF_uSD_MOSI
uSD_VDD
uSD_nCS
uSD_SCK
uSD_MISO
uSD_MOSI
File: cSLIM_uSD.sch
Sheet: cSLIM RS232
P0.11
P0.24
P0.23
RS232_nSHDN
nRF_TX
nRF_RX
RS232_TX
RS232_RX
File: cSLIM_RS232.sch
Sheet: cSLIM RS485
P0.00
P0.14
P0.15
P0.01
RS485_RX
RS485_!RE
RS485_DE
RS485_DI
RS485_A
RS485_B
File: cSLIM_RS485.sch
Sheet: cSLIM LoRa
nRESET
LoRa_nRESET
I2C_SDA
I2C_SCL
P0.08
P0.09
P0.20
SPI_SCLK
SPI_MISO
SPI_MOSI
P0.05
LoRa_nRESET
LoRa_I2C_SDA
LoRa_I2C_SCL
LoRa_UART_TX
LoRa_UART_RX
LoRa_SPI_nCS
LoRa_SPI_SCK
LoRa_SPI_MISO
LoRa_SPI_MOSI
LoRa_nEN
LoRa_SWDIO
LoRa_SWDCLK
LoRa_USB_DP
LoRa_USB_DM
PB03
PA06
P0.21
File: cSLIM-shield-LoRa.sch
Sheet: 9 cSLIM Connectors
P0.20
SPI_SCLK
SPI_MISO
SPI_MOSI
I2C_SDA
I2C_SCL
GPS_ANTENNA
V_MUX_OUT
TBR_PWR
P0.16
LoRa_nRESET
USB_VBUS
P0.02
P0.03
P0.04
P0.05
SPI_CS
SPI_SCK
SPI_MISO
SPI_MOSI
I2C_SDA
I2C_SCL
GPS_ANTENNA
RS232_PWR_OUT
RS232_TX
RS232_RX
TBR_PWR
TBR_PPS
RS485_A
RS485_B
LoRa_nRESET
LoRa_SWDIO
LoRa_SWDCLK
LoRa_SWO
USB_DP
USB_DM
USB_VBUS
nLED_BLUE2
nLED_RED
nLED_YELLOW
nLED_GREEN
nLED_BLUE
File: cSLIM_connectors.sch
VDD is sheld voltage supply
Use jumper to switch between supply from shield/battery and nRF
PWR_FLAG
VDD
V_reg
V_nRF_3V
J101
Conn_01x03_Male
V_MUX_OUT
TBR_PWR
V_nRF_5V
J102
Conn_01x03_Male
I2C pullup resistors
VDD
VDD
R101
4.7k
R102
4.7k
P0.30
I2C_SDA
P0.31
I2C_SCL
P0.19
SPI_SCLK
P0.17
SPI_MISO
P0.18
SPI_MOSI
On nRF-DK,
watch position
of SW1 and SW2 or
disable in board controller
nRF9160 Development Kit Headers
Insert nRF9160 and required circuitry here on standalone version
PCB Holes
V_nRF_5V
V_nRF_3V
nRESET
P101
Power
Analog
P102
GND
P104
Digital
P105
Digital
P106
Digital
P0.31
P0.30
P0.20
P4:GND
P0.13
P0.12
P0.11
P0.10
P0.09
P0.08
P0.07
P0.06
P0.05
P0.04
P0.03
P0.02
P0.01
P0.00
P0.28
P0.27
P0.26
P0.25
P0.24
P0.23
P0.22
P0.21
P0.14
P0.15
P0.16
P0.17
P0.18
P0.19
P5:RESET
P5:P0.13
P5:P0.12
P5:DETECT
P5:P0.11
P5:5V0
P103
Conn_02x03_Odd_Even
Pushbuttons
PB1
TLS_A_070J_LFS
P0.06
GND
PB2
TLS_A_070J_LFS
P0.02
GND
Sheet: cSLIM FRAM
P0.25
SPI_SCLK
SPI_MISO
SPI_MOSI
SPI_nCS
SPI_SCK
SPI_MISO
SPI_MOSI
File: cSLIM_FRAM.sch
Sheet: cSLIM Display
P0.22
SPI_SCLK
SPI_MOSI
DISPLAY_nCS
DISPLAY_SCK
DISPLAY_MOSI
File: cSLIM_display.sch
D Shield schematics
Eivind Jølsgard
NTNU
Sheet: /
File: cSLIM-shield.sch
Title: cSLIM Overview
Size: A4
Date: 2021-03-31
Rev: v1
KiCad E.D.A. kicad (5.1.4)-1
Id: 1/11



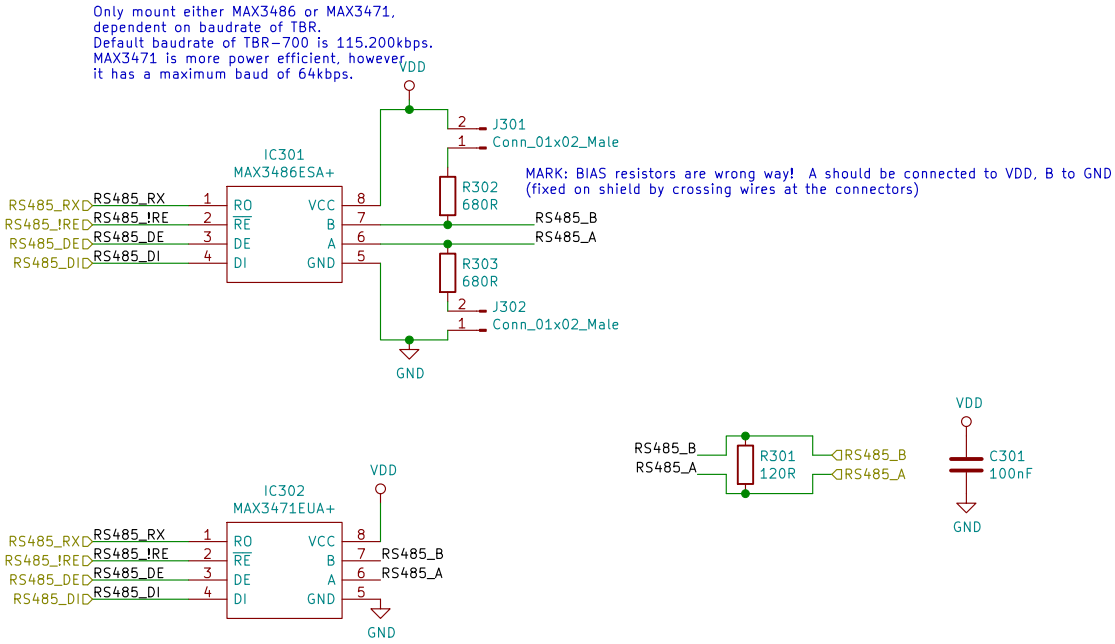
Eivind Jølsgard

NTNU

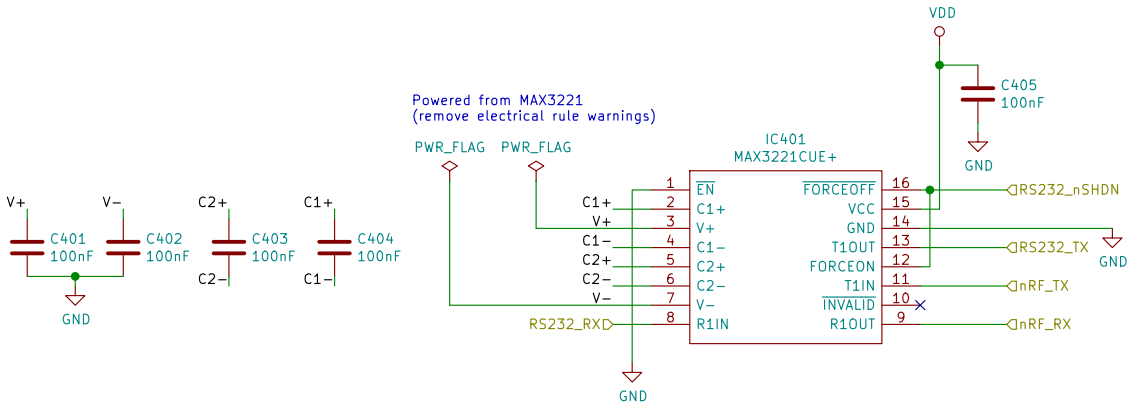
Sheet: /cSLIM LoRa/
File: cSLIM-shield-LoRa.sch

Title: cSLIM LoRa Module

Size: A4	Date: 2021-03-31	**Rev: v1**
KiCad E.D.A. kicad (5.1.4)-1		Id: 2/11

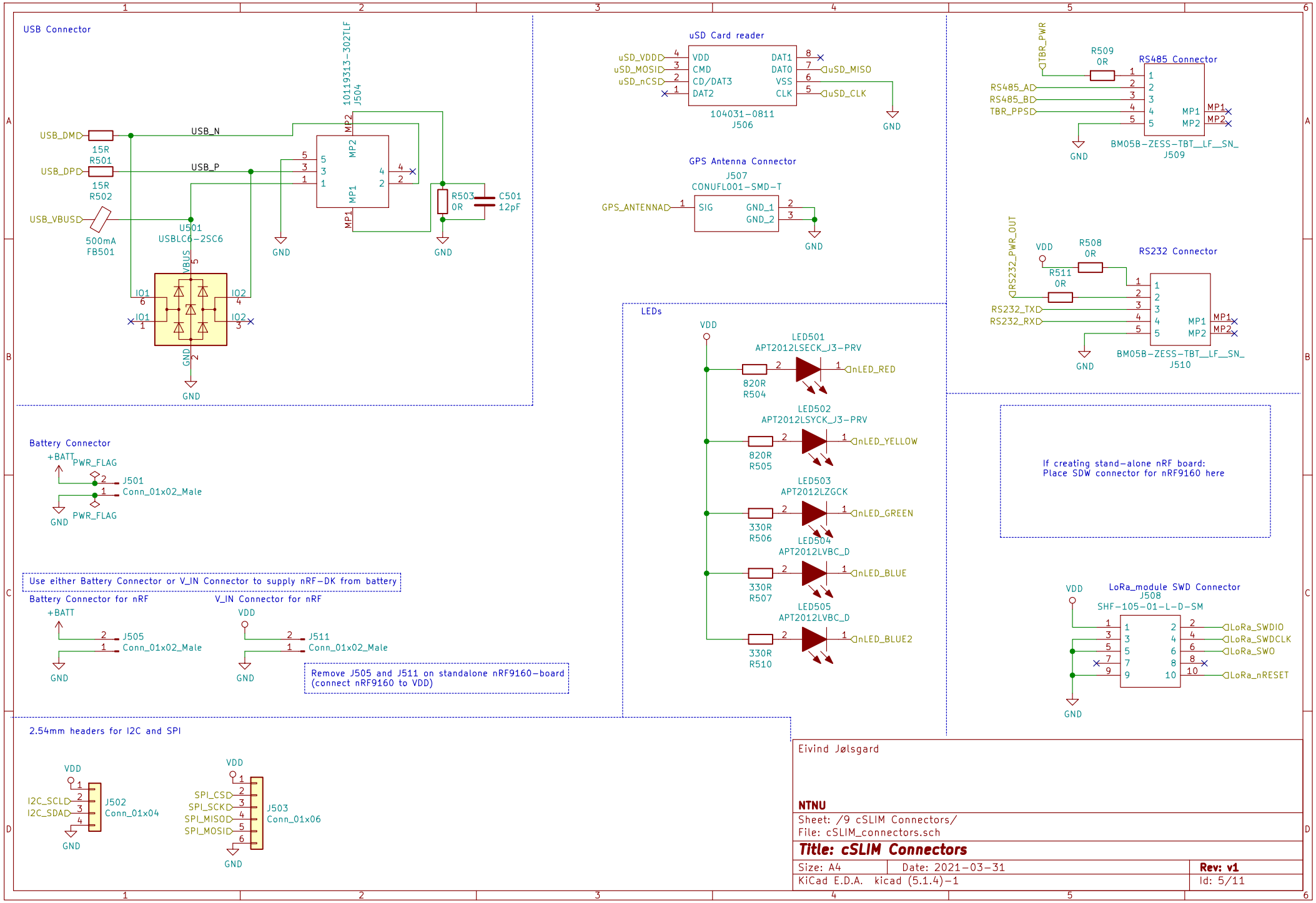


Eivind Jølsgard		
NTNU		
Sheet: /cSLIM RS485/		
File: cSLIM_RS485.sch		
Title: cSLIM RS485		
Size: A4	Date: 2021-06-30	**Rev: v1.1**
KiCad E.D.A. kicad (5.1.4)-1		Id: 3/11



Eivind Jølsgard	
NTNU	
Sheet: /cSLIM RS232/	
File: cSLIM_RS232.sch	
Title: cSLIM RS232	
Size: A4	Date: 2021−03−31
KiCad E.D.A. kicad (5.1.4)−1	Id: 4/11

Rev: v1



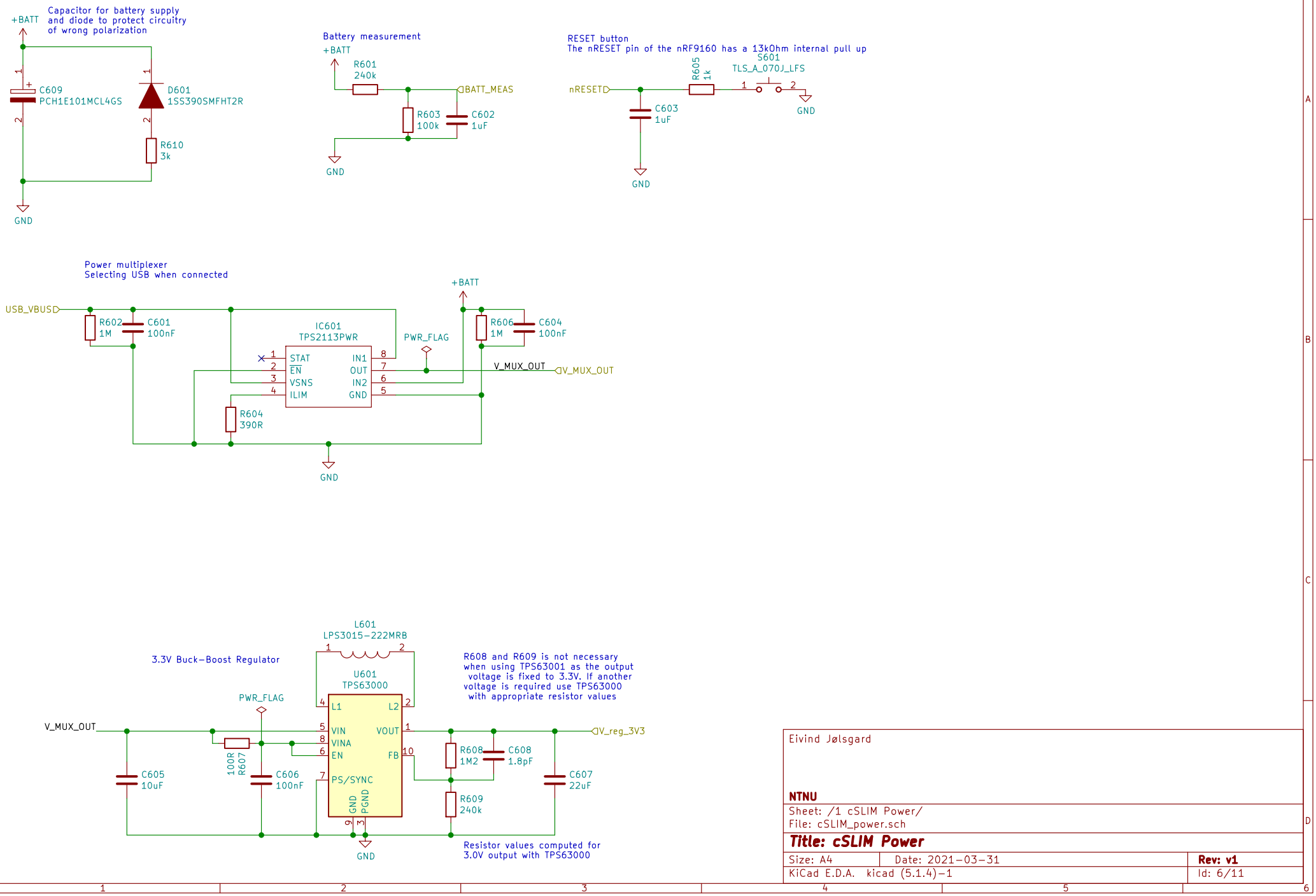
Eivind Jølsgard

NTNU

Sheet: /9 cSLIM Connectors/
File: cSLIM_connectors.sch

Title: cSLIM Connectors

Size: A4	Date: 2021-03-31	**Rev: v1**
KiCad E.D.A. kicad (5.1.4)-1		Id: 5/11



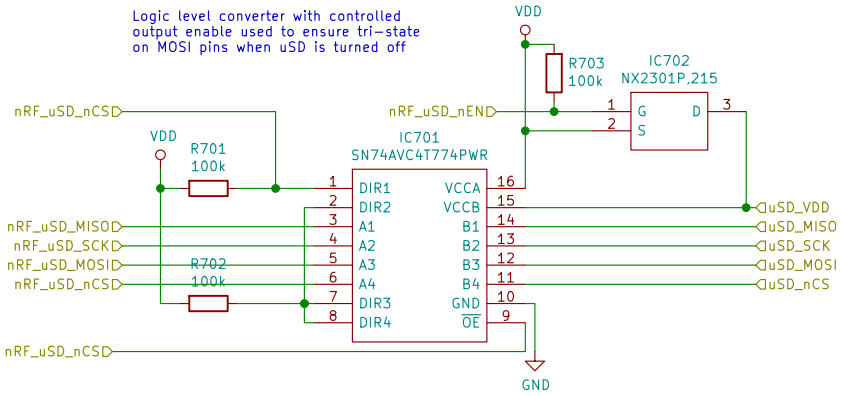
Eivind Jølsgard

NTNU

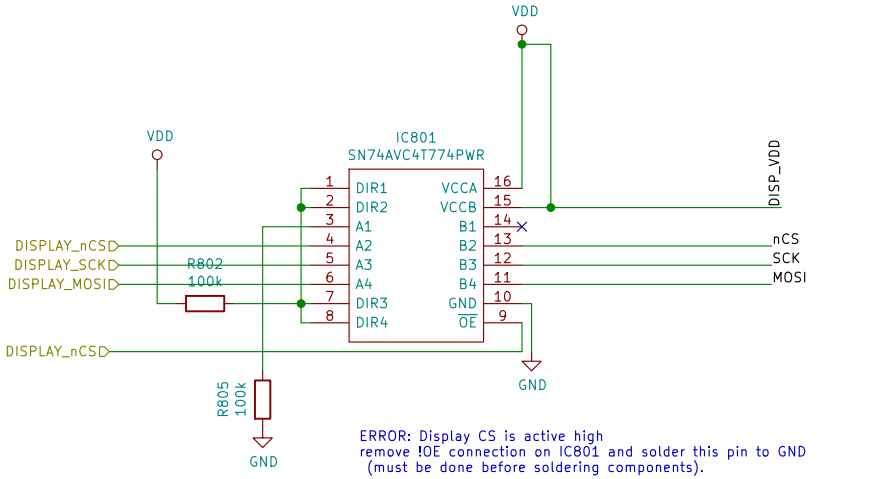
Sheet: /1 cSLIM Power/
File: cSLIM_power.sch

Title: cSLIM Power

Size: A4	Date: 2021-03-31	**Rev: v1**
KiCad E.D.A. kicad (5.1.4)-1		Id: 6/11

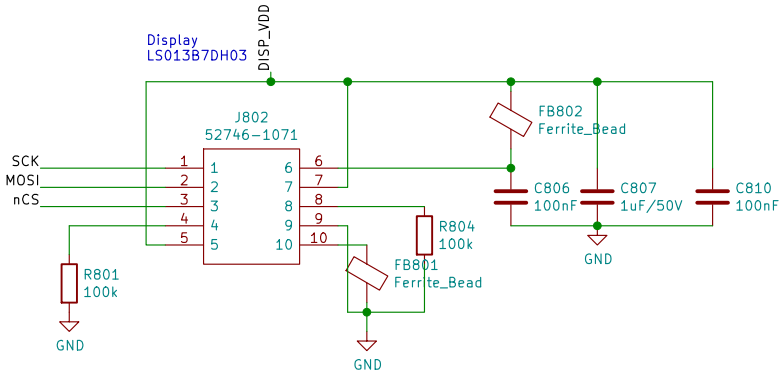


Eivind Jølsgard		
NTNU		
Sheet: /cSLIM uSD/ File: cSLIM_uSD.sch		
Title: cSLIM uSD		
Size: A4	Date: 2021-03-31	**Rev: v1**
KiCad E.D.A. kicad (5.1.4)-1		Id: 7/11



ERROR: Display CS is active high
remove !OE connection on IC801 and solder this pin to GND
(must be done before soldering components).

As display is always on, the LLC should be removed for future designs.
Skyldes at display har aktiv høy cs



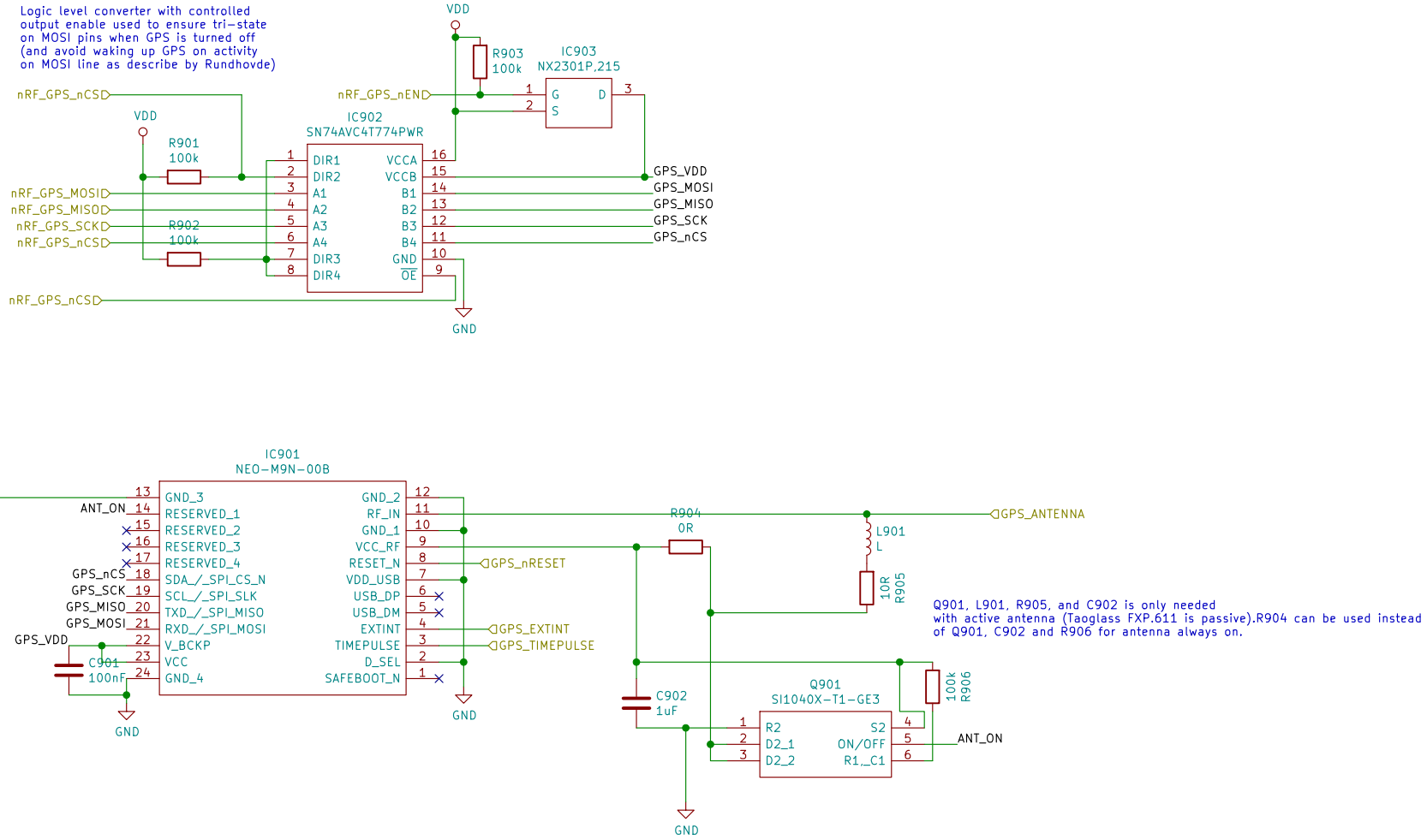
Eivind Jølsgard

NTNU

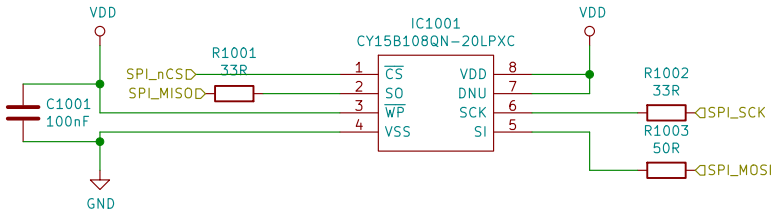
Sheet: /cSLIM Display/
File: cSLIM_display.sch

Title: cSLIM Display

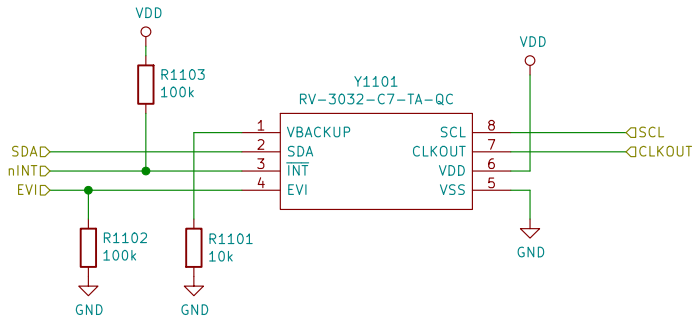
Size: A4	Date: 2021-06-30	**Rev: v1.1**
KiCad E.D.A. kicad (5.1.4)-1		Id: 8/11



Eivind Jølsgard		
NTNU		
Sheet: /cSLIM GPS External/		
File: cSLIM_GPS.sch		
Title: cSLIM External GPS		
Size: A4	Date: 2021-03-31	**Rev: v1**
KiCad E.D.A. kicad (5.1.4)-1		Id: 9/11

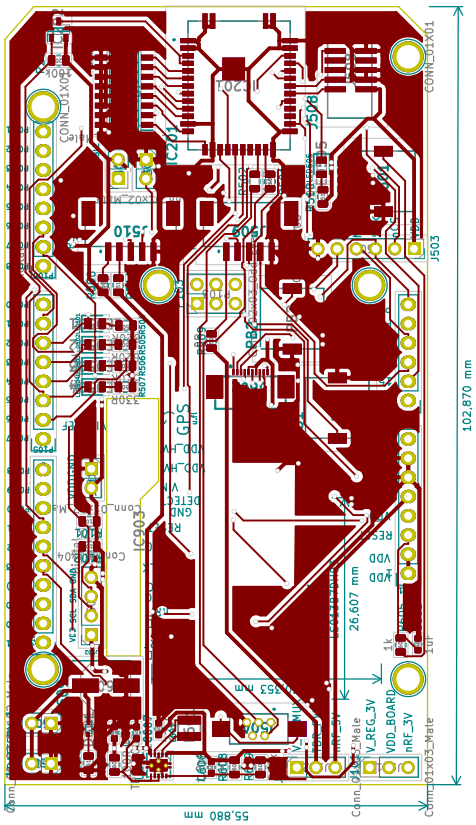


Eivind Jølsgard		
NTNU		
Sheet: /cSLIM FRAM/ File: cSLIM_FRAM.sch		
Title: cSLIM FRAM		
Size: A4	Date: 2021-03-31	**Rev: v1**
KiCad E.D.A. kicad (5.1.4)-1		Id: 10/11

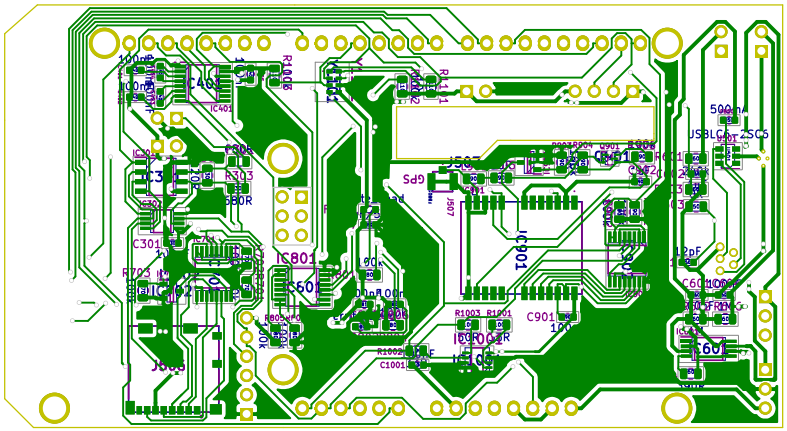


Eivind Jølsgard		
NTNU		
Sheet: /cSLIM RTC/ File: cSLIM_RTC.sch		
Title: cSLIM RTC		
Size: A4	Date: 2021-03-31	**Rev: v1**
KiCad E.D.A. kicad (5.1.4)-1		Id: 11/11

E Shield PCB layout



Eivind Jølsgard	NNU	Sheet:	
		File: _autosave-cSLIM-shield.kicad_pcb	
		Title: cSLIM Shield PCB	
		Size: A4	Date: lun. 30 mars 2015
		KiCad E.D.A. pcbnew (5.1.4)-1	Id: 1/1



Eivind Jølsgard		
NNU		
Sheet: File: _autosave-cSLIM-shield.kicad_pcb		
Title: cSLIM Shield PCB		
Size: A4	Date: lun. 30 mars 2015	Rev: v1
KiCad E.D.A. pcbnew (5.1.4)-1		Id: 1/1

Bibliography

[1] 3GPPrelease 13. https://www.3gpp.org/release-13. last accessed: 2021-07-07.

[2] Altibox gjør norge smartere. https://www.altibox.no/iot/. last accessed: 2021-08-07.

[3] Atsamr34_lorawan_rn_parser. https://github.com/MicrochipTech/atsamr34_lorawan_rn_parser. last accessed: 2021-06-27.

[4] Hilal Bello, Xin Jian, Yixiao Wei, and Min Chen. Energy-Delay Evaluation and Optimization for NB-IoT PSM With Periodic Uplink Reporting. *IEEE Access*, 2018.

[5] Marco Calabretta, Riccardo Pecori, Massimo Vecchio, and Luca Veltri. Mqtt-auth: a token-based solution to endow mqtt with authentication and authorization capabilities. *Journal of Communications Software and Systems*, 14, 12 2018.

[6] Marco Chiani and Ahmed Elzanaty. On the lora modulation for iot: Waveform properties and spectral analysis. *IEEE Internet of Things Journal*, 6(5):8463–8470, 2019.

[7] Coap. https://coap.technology/. Last accessed: 2021-07-26.

[8] Cy15b108qn datasheet. https://www.cypress.com/file/444186/download. Last accessed: 2021-08-07.

[9] Doxygen documenting the code. https://www.doxygen.nl/manual/docblocks.html. last accessed: 2021-08-03.

[10] Dune: Unified navigation environment. https://github.com/LSTS/dune. Last accessed: 2021-07-17.

[11] FAO. *The State of World Fisheries and Aquaculture 2020. Sustainability in action.* Food and Agriculture Organization of the United nations, Rome, 2019.

[12] GSMAmobile iot deployment map. https://www.gsma.com/iot/deployment-map/. last accessed: 2021-07-07.

[13] What is the difference in data throughput between lte-m/nb-iot and 3g or 4g? https://www.gsma.com/iot/resources/what-is-the-difference-in-data-throughput-between-lte-m-nb-iot-and-3g-or-4g/. last accessed: 2021-08-07.

[14] W. Hassan, M. Føre, H.A. Urke, T. Kristensen, J.B. Ulvlund, and J.A. Alfredsen. *System for Real-time positioning and Monitoring of Fish in Commercial Marine Farms Based On Acoustic Telemetry and internet of Fish (IoF).* NTNU, Trondheim, 2019.

[15] Eric Hockersmith and John Beeman. *A History of Telemetry in Fishery Research*, pages 7–19. American Fisheries Society, 01 2012.

[16] Stokland I. Disse karenes undersøkelser kan få følger for E6-utbyggingen. (Norwegian) [these guys' research can have consequences for the e6 develompent]. *Gudbrandsdølen Dagningen*, 2020.

[17] Ingenu back to basics: The shannon-hartley theorem. https://www.ingenu.com/2016/07/back-to-basics-the-shannon-hartley-theorem/. last accessed: 2021-07-07.

[18] E. Jølsgard. *TTK8Shield for Embedded nRF52840-DK robot.* NTNU, Trondheim, 2020.

[19] Device specification for lcd module ls013b7dh03. https://media.digikey.com/pdf/Data%20Sheets/Sharp%20PDFs/LS013B7DH03_Spec.pdf. Last accessed: 2021-08-07.

[20] Laboratório de sistemas e tecnologia subaquática - underwater systems and technology laboratory. https://lsts.fe.up.pt/.

[21] Foivos Michelinakis, Anas Saeed Al-Selwi, Martina Capuzzo, Andrea Zanella, Kashif Mahmood, and Ahmed Elmokashfi. Dissecting energy consumption of nb-iot devices empirically. *IEEE Internet of Things Journal*, 8(2):1224–1242, 2021.

[22] Vivek Mohan. 10 things about lorawan & nb-iot. `https://blog.semtech.com/title-10-things-about-lorawan-nb-iot`. last accessed: 2021-07-30.

[23] Mqtt. `https://mqtt.org/`. Last accessed: 2021-06-08.

[24] Mqtt and coap, iot protocols. `https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php`. Last accessed: 2021-06-08.

[25] Beat Naef-Daenzer, Daniel Frueh, M Stalder, P Wetli, and E Weise. Miniaturization (0.2 g) and evaluation of attachment techniques of telemetry transmitters. *The Journal of experimental biology*, 208:4063–8, 12 2005.

[26] nrf91 series. `https://infocenter.nordicsemi.com/index.jsp?topic=%2Fstruct_nrf91%2Fstruct%2Fnrf91.html`. Last accessed: 2021-07-17.

[27] nrf9160-dk board controlelr. `https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/zephyr/boards/arm/nrf9160dk_nrf52840/doc/index.html#nrf9160dk-board-controller-firmware`. Last accessed: 2021-08-07.

[28] nrf9160 product brief. `nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF9160-SiP-product-brief.pdf?sc_trk={Download%20product%20brief}&la=en&hash=5C889507F72CE933BE712F9748D6FE19600C6746`. Last accessed: 2021-08-07.

[29] Kjelsvik P.A. *Internet of Fish*. NTNU, Trondheim, 2019.

[30] Agustin Pelaez. Lorawan vs nb-iot: A comparison between iot trend-setters. *Ubidots*, 2020.

[31] ABI Research. Lorawan and nb-iot: Competitors or complementary? *LoRaWAN Alliance*, 2019.

[32] Brecht Reynders and Sofie Pollin. Chirp spread spectrum as a modulation technique for long range communication. In *2016 Symposium on Communications and Vehicular Technologies (SCVT)*, pages 1–5, 2016.

[33] M. Rundhovde. *Optimization of LoRa surface bouy for underwater acoustic receiver*. NTNU, Trondheim, 2020.

[34] Rv3032-c7 datasheet. `https://www.microcrystal.com/fileadmin/Media/Products/RTC/Datasheet/RV-3032-C7.pdf`. Last accessed: 2021-08-07.

[35] Bill Schweber. Understanding the advantages and disadvantages of linear regulators. *Digi-Key*, 2017.

[36] S. Santiago Sebastian. Energy efficiency in internet of things: An overview. *International Journal of Recent Trends in Engineering & Research (IJRTER)*, 02:475–482, 06 2016.

[37] Chipquick smd291snl data sheet rev.1.2. `https://www.chipquik.com/datasheets/SMD291SNL10.pdf`. last accessed: 2021-08-03.

[38] Chipquick smdltlfp data sheet rev.1.3. `https://www.chipquik.com/datasheets/SMDLTLFP.pdf`. last accessed: 2021-08-03.

[39] Tbr live datasheet. `https://www.thelmabiotel.com/wp-content/uploads/tb-live-datasheet-1.pdf`. Last accessed: 2021-08-07.

[40] Telenor dekningskart. `https://www.telenor.no/system/coverage/client/?iot=true/`. last accessed: 2021-07-07.

[41] Telia dekningskart. `https://www.telia.no/nett/dekning/`. last accessed: 2021-07-07.

[42] The things network coverage map. https://www.thethingsnetwork.org/map. last accessed: 2021-07-27.

[43] Tps63000 voltage regulator datasheet. http://www.ti.com/general/docs/suppproductinfo.tsp?distId=26&gotoUrl=http%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Ftps63000-q1. Last accessed: 2021-08-07.

[44] P.S. Trefethen. Sonic equipment for tracking individual fish. *U.S. Fish & Wildlife Service, Special Scientific Rpt.*, 1956.

[45] The Things Network duty cycle. https://www.thethingsnetwork.org/docs/lorawan/duty-cycle/. last accessed: 2021-08-08.

[46] Ubloxl neo m8n datasheet. https://www.u-blox.com/sites/default/files/NEO-M8-FW3_DataSheet_UBX-15031086.pdf. last accessed: 2021-08-10.

[47] Ugui. https://github.com/achimdoebler/UGUI. Last accessed: 2021-06-16.

[48] Wlr089u0 product page. https://www.microchip.com/wwwproducts/en/WLR089U0. Last accessed: 2021-08-07.

[49] Alexander M. Wyglinski, Maziar Nekovee, and Y. Thomas Hou. Dedication. In *Cognitive Radio Communications and Networks*, page v. Academic Press, Oxford, 2010.

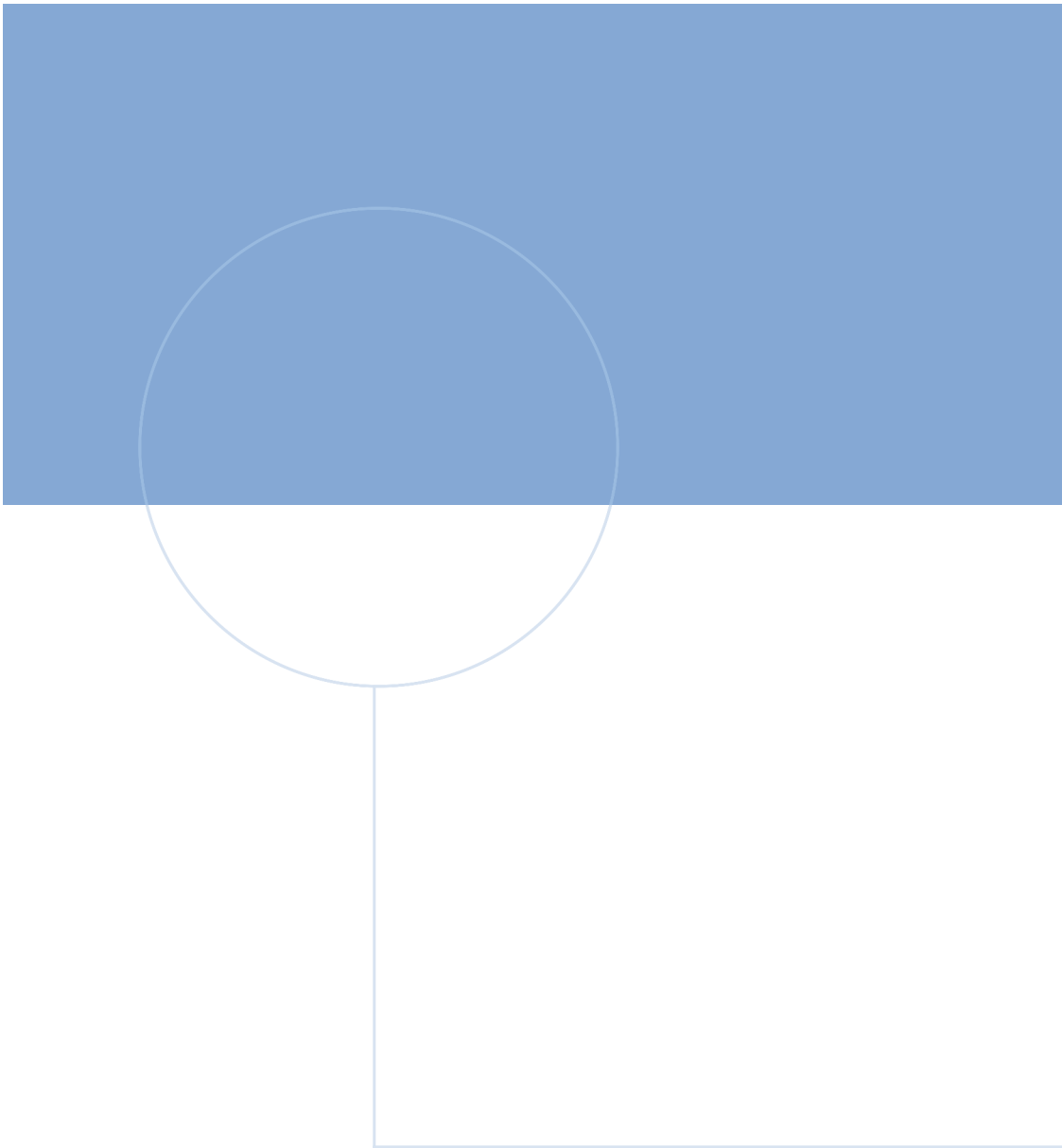
[50] Deliang Yang, Xianghui Zhang, Xuan Huang, Liqian Shen, Jun Huang, Xiangmao Chang, and Guoliang Xing. Understanding power consumption of nb-iot in the wild: Tool and large-scale measurement. *MobiCom '20*, 2020.

[51] E. Zanaj, G. Caso, L. De Nardis, A. Mohammadpour, Ö. Alay, and M.-G Di Benedetto. Energy Efficiency in Short and Wide-Area IoT Technologies—A Survey. *MDIP*, 2021.

[52] Zephyr supported boards. https://docs.zephyrproject.org/latest/boards/index.html. Last accessed: 2021-07-17.

[53] How to build drivers for zephyr. https://interrupt.memfault.com/blog/building-drivers-on-zephyr. Last accessed: 2021-06-16.

[54] Zephyr project documentation. https://docs.zephyrproject.org/latest/. Last accessed: 2021-07-17.



NTNU

Norwegian University of
Science and Technology