Master's thesis

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

Ali Aljumaili

Hybrid electronic tag platform for mapping of large-scale fish migration

Master's thesis in Cybernetics and Robotics
Supervisor: Jo Arve Alfredsen
June 2019

NTNU
Norwegian University of
Science and Technology

Ali Aljumaili

Hybrid electronic tag platform for mapping of large-scale fish migration

Master's thesis in Cybernetics and Robotics
Supervisor: Jo Arve Alfredsen
June 2019

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 (30 Stp.)

Name:	Ali Aljumaili
Program:	Engineering Cybernetics
Title:	Hybrid electronic tag platform for mapping of large-scale fish migration
Title (Norw.):	Hybrid elektronisk merkeplattform for storskala kartlegging av fiskemigrasjon

Project description

Sustainable management of fisheries resources requires deep understanding of fish behaviour and how they interact with the complex and dynamic environment they inhabit. By increasing knowledge of what fish populations do, and correlating this to the attributes of their habitat, we can start understanding why they migrate and how they distribute in space and time. Proper understanding of the mechanisms of ocean migration in the Atlantic salmon, one of the iconic species of the North Atlantic, has riddled researchers for centuries and technology has only recently allowed us to start pursue the validity of different migration hypotheses. Electronic fish tags have proved to be useful tools in this respect, and the rapid technological progress of microelectronics in terms of miniaturization, energy efficiency, MEMS and signal processing, together with increasing availability of data from large-scale earth observation systems (satellites), are currently pushing the boundaries of what is possible in fish migration studies.

The proposed project involves development of a miniature combined ATT (Acoustic Transmitter Tag) and DST (Data Storage Tag) specifically designed for studying the fjord and ocean migration of returning post-spawning adult Atlantic salmon. The combination will allow the tag to measure and store sensor data locally when migrating at the open ocean, while functioning as an acoustic transmitter in the more confined waters of the fjord and river. The transmitter function, and hence the possibility it gives to localize the tag/fish, is proposed as a means of mitigating low tag recovery numbers normally associated with DSTs, as well as a way of providing useful fjord/river migration data when the fish resides within the detection range of acoustic receiver arrays. The tag will feature sensors for magnetic bearing of swimming (magnetometer), swimming activity (accelerometer), inclination of geomagnetic field (magnetometer), water temperature (thermometer), and swimming depth (pressure sensor), and will be able to store a time record of these variables with sufficient sampling rate and resolution for a period of 18 months (one sea-winter). The project includes the following tasks:

- Revisit the requirements specification and prototype tag design developed during the preceding specialization project and provide an account of its current status
- Elaborate on the requirements with respect to the resolution and accuracy of the sensors, and the capacity of memories, battery, communication and processing power. Make a theoretical assessment of the current design's possibilities and limitations, and discuss possible improvements.
- Make a physical implementation of a tag prototype (PCB), write low-level drivers, and validate its design in the laboratory
- Design and write firmware that implements the tag's logger functionality and wireless interface for downloading of data
- Plan and carry out a field test with the tag, document its performance, and finally discuss the results achieved and possible improvements

Project start:	18th September 2018
Project due:	24th June 2019
Host institution:	Department of Engineering Cybernetics, NTNU
Supervisor:	Jo Arve Alfredsen, NTNU/DEC

Trondheim, 17th September 2018
Jo Arve Alfredsen

This page has been left intentionally blank.

Abstract

Acoustic transmitters tags (ATT), together with hydrophones, allow for real-time tracking of fish but have the downside of hydrophones not being widely available in the open ocean. Indeed, hydrophones need to be placed every 200 - 1,000 meters to receive the acoustic pulse from the ATT tags. Data storage tags (DST) provide another telemetry offline monitoring technique by using sensors to measure temperature, pressure, magnetic field and tilt, and then compare these recording with earth observation data in order to gain an estimate of the migration path of the fish/tag. DST tags are excellent for tracking fish migration paths but have the downside of low retrieval rates.

As of 2018, the Atlantic salmon population is declining. This is due to both human impacts and general large-scale reduction in survival at sea. Therefore, it is necessary to gain more insight into their migration journey and real-time tracking data. A possible solution for this problem is combining acoustic transmitters with data storage tags to create a hybrid system that has real-time tracking from ATT and hydrophones and offline tracking for long migration journeys.

The results of this thesis are a miniaturised PCB (40 mm x 14.5 mm x 1.65 mm) for a data storage tag with an acoustic transmitter expansion possibility. The design contains a top of the line temperature sensor with ± 0.1 °C accuracy, an MSP430 FRAM microcontroller, an inertial sensor for measuring acceleration, tilt and magnetic field strength, a pressure sensor to measure swimming depth and an NFC transponder.

Keywords: ATT, Acoustic Transmitter, DST, Data storage tags, Hybrid ATT/DST Atlantic Salmon Tacking, Navigation.

Sammendrag

(Norwegian translation of the abstract)

To kjente metoder for fisk lokalisering er bruk av ATT (Acoustic Transmitter Tag) og DST (Data Storage Tags). Akustisk transmitter tagger sender ut en lydbølge som kan mottas av hydrofoner plassert i sjøen. Dette gir en mulighet for sanntid sporing av fisk, men har ulempen av å være økonomisk dyrt da mange hydrophone trenger å bli plassert i fjern områder i saltvann for å kunne tilby sanntid sporing i saltvann. En annen telemetri metode for fisk lokalisering er bruk av DST (Data Storage Tags) som er små innebygde systemer med flere sensorer, en mikrokontroller og et minne for offline-lagring av data. DST er ofte plantet på innsiden av fisken eller festet eksternt, noe som betyr taggen må hentes fra fisken for å laste ned lagret data.

Denne masteroppgave forsker på mulighet av å kombinere ATT med DST til et hybrid ATT/DST løsning som har fordelene av begge verder, akkurat sanntid sporing mulighet av ATT og offline sporing av DST. Gjennomførbarhet av å kombinere ATT og DST er forsket, dette er avhengig på flere faktorer som pris, strømforbruk og størrelse. En miniatyrisert design for en DST er utviklet med alt nødvendig strømforbruk budsjetter, analyse av komponenter og alt som trenges for å bygge en plattform som kan videreutvikles.

Resultatet av denne masteroppgaven er en offentlig tilgjengelig design for en miniatyrisert DST med mulighet for en akustisk transmitter "addon".. The hybrid løsning har en størrelse på 40 mm lengde og 14.5 mm i bredde, en moderne temperatur sensor med nøyaktighet av ±0.1 °C, en MSP430 FRAM mikrokontroller, en bevegelse sensor for måling av akselerasjon i 3 akser og magnetisk felt i 3 akser, i tillegg til en trykksensor for å måle svømmedybde. Designet har også en NFC grensesnitt for å kunne konfigurere og laste ned data og energihøsting fra NFC felt som gir muligheten til å power taggen ved dødt batteri.

Stikkord: ATT, Acoustic Transmitter, DST, Data storage tags, Hybrid ATT/DST Atlantic Salmon tacking, navigation.

Preface

This thesis is the result of a two-year master's program in Cybernetics and Robotics with specialisation in Embedded Systems at the Norwegian University of Science and Technology. The thesis continues the work on the ATT/DST platform work started in the specialisation project in 2017.

All of the equipment used in this thesis is funded by the Norwegian University of Science and Technology. The work focuses on design research and establishment of an electronic hybrid tag platform that can be used for Atlantic salmon migration tracking and/or gaining more insight into fish behaviour.

This document is also meant to provide information useful to anyone maintaining, modifying, or improving the ATT/DST platform. It has valuable insight into the design choices and experience gained from developing miniaturised PCBs and modular firmware.

The reader is expected to have basic knowledge about electronics, software design and the C programming language.

Trondheim, 24th June 2019

Ali Aljumaili

Acknowledgments

First and foremost, I would like to express my most profound appreciation to Professor Jo Arve Alfredsen for his continuous support of my thesis research and implementation, motivation, enthusiasm, and immense patience. His guidance has helped me gain excellent working experience in the exciting field of embedded systems.

Besides my Supervisor, I would like to acknowledge all of my friends and others who, in one way or another, shared their support throughout my master's journey.

Last but not the least, I would like to thank my family: my dad, Tareq Aljumylee, for giving me incredible support and reading through multiple revisions of this thesis, my brother, my sister, and, most importantly, mother for supporting me throughout my life.

Table of Contents

List of Figures

List of Tables

This page has been left intentionally blank.

Acronym and Abbreviation List

ACLK Auxillary Clock

ADC Analog-to-Digital Converter

ATT Acoustic Transmitter Tag

BSL Boostrap Loader

CAD Computer-aided Design

CCS Composer Studio ™ Development Tool for MSP430

CPU Central Processing Unit

CTD Conductivity, Temperature, Depth (in situ ocean measurements)

DAC Digital-to-Analog Converter

DCO Digitally Controlled Oscillator

DNI Do Not Install

DST Data Storage Tag

EAGLE Easily Applicable Graphical Layout Editor

EEPROM Electrically Erasable and Programmable Read Only Memory

ESA European Space Agency

ET Energy Trace

eUSCI Enhanced Universal Serial Communication Interfaces

FET Flash Emulation Tool

FLL Frequency Locked Loop

GEO Geostationary Orbit

GHRSST Group for High Resolution SST

GIE General Interrupt Enable

GPIO General Purpose Input Output

GRACE Graphical Peripheral Configuration Tool

GUI Graphical User Interface

I/O Input/Output

I^2C Inter-Integrated Circuit 2-wire communication bus

IDE Integrated Development Environment

IEC International Electrotechnical commission

IGRF International Geomagnetic Reference Field

ISO International Organization for Standardization

ISR Interrupt Service Routine

JTAG Joint Test action Group

LDO Low Dropout Regulator

LED Light Emitting Diode

LP Low Power

LPM Low-Power Mode; also named PM for Power Mode

LSB Least-Signficant Bit

MCLK Master Clock

MCU Microcontroller Unit

MEMES Micro-Electro-Mechanical Systems

MSB Most-Signficant Digit

MSP Mixed Signal Processor

NASA National Aeronautics and Space Administration

NFC Near-Field Communication

NMI (Non)-Maskable Interrupt

NOAA National Oceanic and Atmospheric Administration

NTNU Norwegian University of Science and Technology
(Norwegian:Norgesteknisk-naturvitenskapelige universitet)

NVM Non-Volatile Memory

ODR Output Data Rate

OI Optimal Interpolation

OSR Oversampling Ratio

PC Program Counter

PCB Printed Circuit Board

POR Power-On Reset

PUC Power-Up Clear

Q-factor Quality Factor

RoHS Restriction of Hazardous Substancess Directive (Directive 2002/95/EC)

RTC Real-Time Clock

RX Receive

SBW Spy-Bi-Wire (2-wire JTAG protocol) communication

SMCLK Sub-System Master Clock

SPI Serial Peripheral Interface

SR Status Register

SST Sea Surface Temperature

SWCLK Serial Wire Clock

SWD Serial Wire Debug

TX Transmit

UART Universal Asynchonous reciever transmitter

ULP Ultra Low-Power

USI Universal Serial Interface

WDT Watchdog Timer

Table 1: SI Units

Quantity	Unit	Symbol
Capacitance	Farad	F
Charge	Columb	C
Electric Current	Ampere	A
Energy	Joule	J
Frequency	Hertz	Hz
Inductance	Henery	H
Length	Meter	m
Magnetic Flux Density	Tesla	T
Mass	Kilogram	kg
Power	Watt	W
Pressure	Bar	Bar
Pressure	Pascal	Pa
Resistance	Ohm	Ω
Temperature	Degree Celsius (relative to 273.15 K)	°C
Temperature	Kelvin	K
Time	Second	s
Voltage	Volt	V

Table 2: SI Prefixes

Prefix	Symbol	Power
atto	a	10^{-18}
femto	f	10^{-15}
pico	p	10^{-12}
nano	n	10^{-9}
micro	µ	10^{-6}
milli	m	10^{-3}
centi	c	10^{-2}
deci	d	10^{-1}
deka	da	10
kilo	k	10^{3}
mega	M	10^{6}
giga	G	10^{9}
tetra	T	10^{12}

1 Introduction

Contents

This chapter introduces the motive, theory of combining existing aquatic telemetry technologies to create an enhanced platform that can be used for both short- and long-term fish tracking.

1.1 Introduction

Aquatic telemetry technology and the information it provides will define how future global aquatic management practice must evolve. However, we still face significant challenges when it comes to monitoring aquatic life across the ocean [1]. These challenges are partly resolved by the use of telemetry monitoring systems [2], [3] such as Data Storage Tags (DSTs) and Acoustic Transmitter Tags (ATT) [4]. Both of these technologies have limitations that are discussed in this section.

Some of the benefits of using acoustic transmitter tags include the possibility of remote fish monitoring in both fresh and saltwater. These tags can be made very small in size and last for multiple years. Further, they are not often used for across the ocean tracking because many receivers (hydrophones) need to be placed in order to provide this remote tracking possibility. Data storage tags are more suited for long migration journeys, as they do not depend on any receivers. These tags record environmental data that can be used for providing migration path estimates. Yet, the downside of these tags is that they must be retrieved in order to access the recorded data.

The work conducted in this thesis focuses on harvesting the benefits of both telemetry technologies ATT and DST to create a hybrid electronic tag platform that can be used to provide a new telemetry aquatic monitoring method.

1.2 Geopositioning Techniques

This section introduces the two tracking techniques that will be utilised in the hybrid electronic tag platform to provide across the ocean monitoring. Both have strengths and weaknesses [5], but combining them can improve the geolocation estimates.

1.2.1 Sea Surface Temperature Matching

The motion of molecules at the top layer of the ocean radiates heat that can be measured using satellites [6] that collect sea surface temperature (SST) data, at any time of the day. This data is then made publicly available by providers such as NOAA, to provide a snapshot of the ocean's temperature at a particular time and date.

The SST matching method [7] requires making a time-stamped temperature measurement at the surface of the water. This local temperature measurement can then be used to constrain the search area by finding an appropriate SST

field and matching the local temperature with the SST field data. Figure 1.1 shows an example of such SST field map.

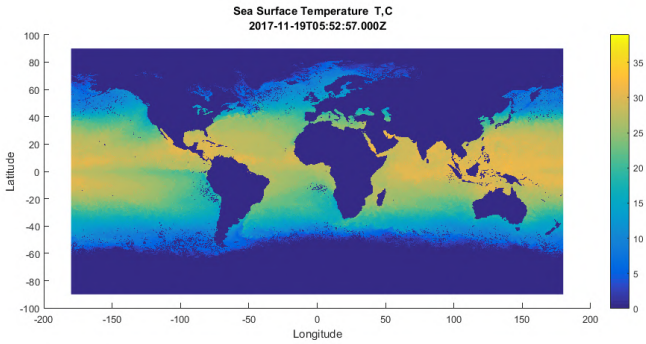


Figure 1.1: Sea Surface Temperature Map

1.2.2 Geomagnetic Field Matching

The earth's geomagnetic field is represented as a vector with three components (Figure 1.2) respecting the total magnetic field intensity and inclination angle. This vector field is modeled by the International Geomagnetic Reference Field (IGRF). More information can be read in Appendix D.

The approach to geopositioning using the earth's magnetic field intensity and inclination angle (the angle at which field lines intersect the earth's surface) is made by comparing the measured sensor data to a mathematical model, such as the IGRF, to find the magnetic signature of a particular area [5].

1.3 Background and Motive

There are currently many ongoing projects [8] conducted by IASRB[1] to study fish migration paths. These projects use different techniques to track the fish, but their exact migration path still has many unknown variables that we wish to gain an insight to. Studies using electronics tags, such as DSTs [9], for fish tracking [3], [4], [10]–[12] have a retrieval rate of under 50%. Further, combining a data logger with an acoustic transmitter can increase the retrieval rate, as well as give an accurate estimate of their past movement path and other vital information. This gives the motivation of developing an ultra-low power hybrid ATT/DST system that comes from using the new advancements in low-energy microcontrollers and sensors to create a system that takes as much space as possible while still providing increased flexibility to a well-established tracking technique ATT. Adding multiple sensors with a data logger functionality can provide more information on the fish migration path, especially in periods where no hydrophones (receivers) are nearby to receive the signal from the acoustic transmitter tags.

1.4 Previous Work

This master's thesis provides a follow-up to the work completed in my specialisation project [13] taken in autumn 2017 at NTNU, Trondheim. During the specialisation project, the necessary theoretical groundwork and practical PCB design experience were gained to create a prototype with temperature, motion, magnetic and pressure sensors. The assembly, testing, and verification of this prototype were unsuccessful due to time restraints and other design choices that caused assembly difficulties. The prototype was finally assembled thanks to the help of Elprolab at NTNU, yet many shortcomings were

[1] Atlantic Salmon Research Board

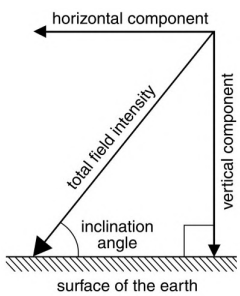


Figure 1.2: The Earth's Magnetic Field

discovered and great motivation was needed to create an enhanced prototype that solved those problems. Chapter 4 discusses these shortcomings in great detail.

This thesis project continues the work done in the specialisation project with an initiation to solve all of the difficulties, as well as to provide a platform for later development of the ATT/DST hybrid system. The timeline is provided in Table 1.1.

Table 1.1: ATT/DST Project Timeline

November 2017	Prototype Rev.1.0.0 designed and manufactured as a part of the specialisation project.
September- October 2018	Assembly, testing & verification for Rev.1.0.0 completed.
February 2019	Re-evaluation of all components.
March-April 2019	Release of Rev.2.0.0 manufacturing.
May - June 2019	Testing, coding, and verification of Rev.2.0.0

1.5 Objective and Scope

The objective of this thesis is to design and implement a hybrid electronic tag platform that combines the strength of ATT and DST. This platform will be designed to allow for future expansion and development.

The scope of this thesis is focused on creating a hybrid electronic tag solution; the acoustic transmitter part is provided by Thelma Biotel, while the data storage tag will be created from scratch. This tag can then be used commercially and for research to expand on existing technologies.

1.6 Thesis Outline

This thesis is divided into seven chapters that provide enough information to understand and learn from the development work of the hybrid ATT/DST project.

Chapter 1 provides an introduction and motive for developing an ultra-low power ATT/DST system.

Chapter 2 discusses the specifications for such a system.

Chapter 3 provides the components of choice together with the feasibility analysis based on the specifications set out in Chapter 2.

Chapter 4 describes the work completed to put together all of the devices into a PCB that is easy to assemble and test.

Chapter 5 introduces the firmware design of the hybrid electronic tag.

Chapter 6 provides the results from testing the data storage tag, as well as information for the feasibility concerning storage capacity and power consumption.

Chapter 7 provides the conclusion and reflection on lessons learned from creating the DST tag, in addition to whether or not it is a feasible solution.

Appendix A gives background information that can be necessary to understand the concepts.

Appendix B provides some of the calculations used in this thesis.

Appendix C shows the design schematics, PCB board layouts, and 3D renders of the assemble, as well a bill of materials and footprint documentation.

Appendix D, a getting started guide, continues an introduction to downloading the files and making progress with the work of the ATT/DST project. This guide has multiple small, informal sections that provide useful information for

customisation of the development environment, as well as known limitations which are often problems that were encountered when testing the prototypes and are now documented to avoid wasting time in the future.

1.7 License Information

The work presented in this thesis can be found in a Bitbucket repository together with all necessary documentation. To make getting started on this work even faster, a custom landing page with the documentation, hardware, software, and extras can be found by accessing the link below:

`https://dst-ntnu.bitbucket.io/`

This work can be copied and redistributed to any medium or format as long the license terms above are followed. If you make use of hybrid ATT/DST project in scientific publications, please cite it. The BibTeX reference is

```
@article{Aljumaili:2019,
author = {Ali Aljumaili},
title = {Hybrid ATT/DST platform for mapping of large-scale
    fish migration},
school = {Norwegian University of Science and Technology},
year = {2019},
month = June,
}
```

This page has been left intentionally blank.

2

System Requirements and Specifications

Contents

One research task for this thesis is to elaborate on the requirements and specifications of the ATT/DST hybrid system. These chapters set out the requirements and specifications for the hardware, software, functionality and product lifetime. The requirements for the required accuracy and resolution for the sensors is researched for existing data sources. The system components are then chosen to fulfil these requirements and presented in Chapter 3.

2.1 Time Accuracy Requirement *(R1)*

The Hybrid electronic tag platform is a real-time driven system. This makes it necessary to have an accurate pulse to calibrate the system.

The acoustic transmitter signal is usually modulated by encoding data in the time between the pulses, a technique called Differential Pulse Position Modulated (DPPM). On the other hand in Data Storage system, events such as measurement, need to be time and date stamped, which can be achieved by a Real-Time Clock (RTC). Both of these systems require an accurate reference frequency.

In the field of embedded systems the reference frequency is 32.768 kHz (2^{15}), a frequency that can be divided using binary dividers to provide a frequency of 1 Hz to be used in RTC logic, as well as any additional timers. The frequency can also be used to tune higher frequencies, which is something that will be discussed in Section 6.4.1. The time accuracy requirement is summarised in Table 2.1.

Table 2.1: Time Accuracy Requirement (***R1***)

Frequency	Accuracy
32.768 kHz	± 5 ppm

2.2 Swimming Depth Requirement (*R2*)

The swimming depth is an important parameter that is used to determine the depth or vertical displacement of the fish/tag in the open ocean. This parameter is especially crucial for the SST geolocalisation method introduced in Section 1.2.1, for the provided earth observations data is only valid for the top layer of the sea surface.

Pressure and depth have a directly proportional relationship, hence the requirement for a pressure sensor with parameters summarised in Table 2.2.

Table 2.2: Swimming Depth Requirement ***(R2)***

Parameters	Range of Values	Resolution	Accuracy
Depth	0.1 - 300 meters	0.01% of range	±0.1%

2.3 Temperature Measurement Requirement *(R3)*

Temperature plays a vital role in the reliability of many of the measurements. Pressure (***R1***), frequency (***R2***), and the SST geolocalisation method all depend on an accurate temperature measurement. The requirement for the temperature sensor is summarised in Table 2.3.

Table 2.3: Temperature Measurement Requirement ***(R3)***

Parameters	Range of Values	Resolution	Accuracy
Temperature	−1 °C - 40 °C	0.0078 °C	± 0.1 °C

2.4 Tilt Measurement Requirement (R4)

The tilt angle can be detected using a 3-axis accelerometer [14]. This is an essential parameter for providing the inclination angle needed by the geomagnetic geolocalisation method introduced in section 1.2. An accelerometer can further provide data about acceleration data used for tag heading.

Table 2.4: Tilt Measurement Requirement ***(R4)***

Parameters	Range of Values	Resolution	Accuracy
Movement (Tilt)	± 90° (180° span)	0.05°	± 3°

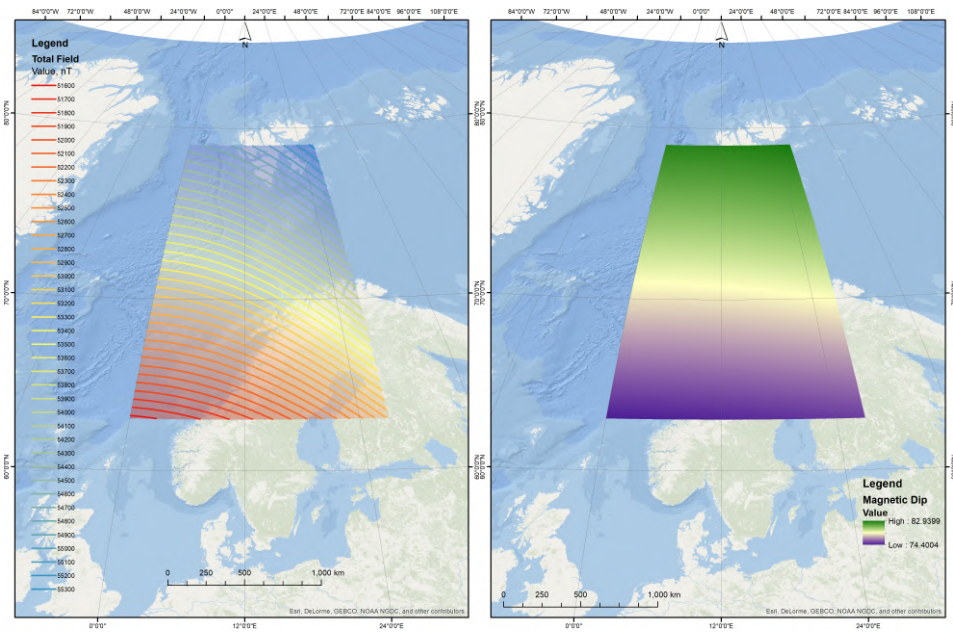
2.5 Geomagnetic Field Strength Requirement *(R5)*

The required parameters for the geomagnetic field strength are based on [5] research on using the geomagnetic field for geolocation estimates. These parameters are summarised in Table 2.5. The range of these values was also verified by simulating the IGRF model in a square area between Trondheim's fjord and Svalbard. This simulation started as part of the specialisation project and was completed in this thesis. The code for this can be found in Appendix D.

The simulation results are plotted in Figure 2.1a (Full field strength) and Figure 2.1b (inclination angle).

Table 2.5: Geomagnetic Field Strength Requirement ***(R5)***

Parameters	Range of Values	Resolution	Accuracy
Magnetic Field Strength	26 µT - 66 µT	100 nT	±300 nT



(a) IGRF Total Field Strength **(b)** IGRF Inclination

Figure 2.1: IGRF Simulation Result

2.6 Storage Capacity Requirement (R6)

A non-volatile (retains data when powered off) memory is required to store the data samples during the 18-month migration journey. Two types of modern, non-volatile memory technologies are researched with respect to data reorientation period, read/write times, and current consumption. The results of this comparison in Table 2.6 show that FRAM is a more modern non-volatile memory technology that uses much less current and has a faster read/write duration. One downside of using FRAM is that it does not come with large capacities, yet the storage capacity needs to be compared with respect to the required resolution of the sensor. The chosen memories must be drafted with respect to polling interval vs. capacity during the 18-month duration.

Table 2.6: Comparison FRAM, and Flash based on data from Fujitsu Semiconductor [15] and [16]

Item	FRAM	Flash Memory
Data retention	Non-volatile	Non-volatile
Period data retained	151 years	25 years
Page read time	120 ns	< 120 ns
Byte write time	125 ns	75 µs
Standby Current	3.5 µA	45 µA
Read current	4 mA	11 mA
Write current	3.2 mA	17 mA
Number of write cycles	Unlimited	100 000
Writing Method	Overwriting	Erase (sector) + write
Unified memory	Yes	No

2.7 Wireless Interface Requirement *(R7)*

The hybrid electronic tag components are to be sealed after testing inside a waterproof hermetic seal. This makes it inconvenient to penetrate the tag's enclosure to configure or download data from the electronic tag. Having a wireless interface allows added flexibility, such as remote configuration and data download. This interface is required to have as small a size as possible and consume zero power in standby, as it is only needed when configuring and downloading

data from the system.

2.8 Energy Harvesting Requirement *(R8)*

The energy harvesting requirements is motivated by the need to download stored tag data at the end of the mission when the battery is flat and not able to provide enough power. The energy harvesting circuit is required to work interdependently of the battery and power up the microcontroller, on-board memory, and wireless interface for the duration of the read cycle of the full memory.

2.9 Battery Lifetime Requirement (*R9*)

Salmon spends approximately 18 months in salt water where they achieve most of their growth before returning to fresh water to reproduce. Hence, the hybrid electronic system must have enough battery capacity for the duration of the journey. An additional sub-requirement for this is that we desire to monitor the battery lifetime by using either a hardware or software solution to estimate remaining battery life.

2.10 Hardware Size Requirement (*R10*)

Having electronic tags implanted into fish requires them to be as small as possible. This motivates researching different manufacturing techniques to make the final electronic tag as small as possible. The required size for the prototype is set to 40 mm (length) x 14.5 mm (width).

2.11 Firmware Requirements and Specifications

The upcoming subsections describe the required considerations when developing the firmware for the electronic hybrid tag. The firmware solution is specified to be built on a modular design.

2.11.1 Power Lost Behaviour Requirement *(R11)*

The electronic tag is a time driven system that depends on accurate timing. Upon the unforeseen event of power loss, the system needs to detect and register the time before safely shutting down the system.

Ideally, the system would keep the time running by using a redundant power source. A possible solution for this is required to be researched.

2.11.2 Ability to detect RFID field *(R12)*

This is required to be able to configure the system or download the data. A wireless interface must be created to fulfil this requirement.

2.11.3 Operating Modes Requirement *(R13)*

The tag will have three operating modes:

1. Data Logger Mode: sample the devices at a certain predefined polling interval and store the data on memory.

2. Transmitter Mode: transmit an acoustic pulse at a set interval without any storage capabilities.

3. Hybrid Data Storage and Transmitter mode: this mode combines the sampling of sensors and recording on memory and transmitting an acoustic pulse at a set interval.

The configuration settings of these modes are to be made wirelessly with the ability to choose different sampling and transmission periods. The wireless interface is also to provide an interface for downloading the stored data.

2.12 Documentation Requirements *(R14)*

The work should be well documented and easily accessible with previous revisions and extra material that can be used to improve the product. In practice,

these documents include:

- Bill of Materials.
- Board and Schematics Design Files.
- Assembly drawings.
- Solder Pads Dimensions
- Revision History.
- Gerber Files and PCB Stackup.
- Firmware Code.

3

System Realisation

Contents

All devices and components are reconsidered with respect to the requirements in Chapter 2 and the experience from testing the first version of the prototype. This chapter presents the chosen components with a short introduction of their function, in addition to some comparisons and calculations.

3.1 Introduction

The design of embedded systems is an iterative process shown in Figure 3.1. There is a constant race to choose components that are more accurate, smaller, and affordable. This makes it essential to keep up with the rapid development by re-evaluating design choices and upgrading components whenever applicable. The specialisation project [13] conducted some work on choosing suitable components for a data storage tag. These are now reevaluated as part of this thesis and tagged with version 1 (V1) or version 2 (V2) for the upgraded design choice. The reevaluation of these components is done with respect to the requirements (***R1 - R14***).

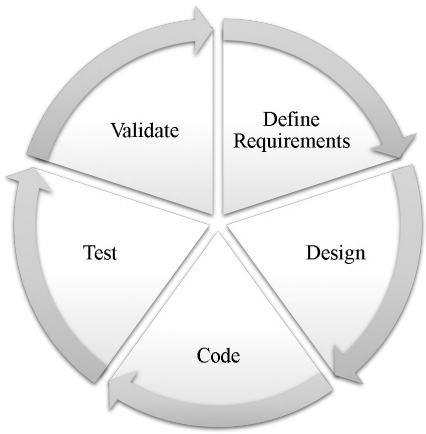


Figure 3.1: Design Process Cycle

3.2 Frequency Reference

The upcoming subsection provides the fulfilment for ***(R1)*** by choosing a low frequency (LF) 32.768 kHz quartz crystal oscillator that has a ± 5 ppm accuracy. This frequency will be used to provide an internal timestamp in the electronic tag and to give an accurate LF reference for all time-dependent sub-modules.

3.2.1 CM315D Crrystal (V2)

Figure 3.2: CM315D Package [17]

The selected turning fork crystal oscillator [18], [19] shown in Figure 3.3 works by using an inverted amplifier controlled by a microcontroller that provides a gain of 180° shift; an additional 180° is then provided by the load capacitors C_{L1} and C_{L2}, creating a closed-loop phase shift of 360°. Rf act as the feedback resistance, which is large enough to be neglected. The calculation for selecting these capacitors is shown in section B.2 and are selected as $C_1 = C_2 = 23pF$

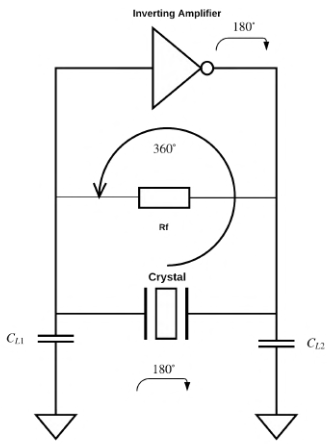


Figure 3.3: MOS Fundamental Crystal Oscillator Circuit

3.3 Swimming Depth Measurement

This section provides the fulfilment for the swimming depth measurement requirement ***(R2)*** by choosing a suitable pressure sensor.

3.3.1 MS5837-30BA - Pressure Sensor (V1)

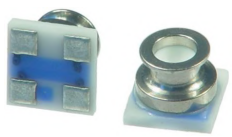


Figure 3.4: MS5837-30BA Package [20]

The MS5837-30BA pressure sensor comes with a built-in temperature sensor that can be used to provide a temperature compensated pressure measurement. The operating range for the sensor is 0 to 30 bar or approximately 0 to 300 metres with a maximum pressure rating of 50 bar or approximately 500. The sensor has an I^2C interface, as well to a built-in PROM memory where calibration words from the factory are stored. The specifications for the pressure sensor are summarised in Table 3.1

Table 3.1: MS5837-30BA Specifications

MS5837-30BA Pressure Sensor	
Manufacturer	TI Connectivity
Communication Interface	I^2C
Operating Temperature	−20 °C to 85 °C
Voltage Range	1.5V to 3.6V
Operating Range	0 - 30 Bar
Max Pressure	50 Bar (ISO 22810) [1]
Accuracy	± 400 mbar (full range)
Resolution	0.2 mbar [2]
Active Current	1.5 mA
Standby Current	0.1 µA
Package Size	3.3 mm x 3.3 mm x 2.75 mm
No. of pins	4

[1] The sensor would get damaged beyond the depth of approximately 500 meters, this can typically be detected in software and does not damage any of the other components.

[2] Resolution is dependent on the selected depth range.

3.4 Temperature Measurement

The first version of the prototype contained a SI7051 [21] temperature sensor that had a temperature accuracy of ± 0.25 °C for the required temperature range defined in ***(R3)***. The revised design chooses the TMP117 temperature sensor to fulfil the ***(R3)*** temperature measurement requirement.

3.4.1 TMP117 - Temperature Sensor (V2)

Figure 3.5: TMP117 WSON Package [22]

The TMP117 temperature sensor is a modern digital temperature sensor that came out in 2018 that has an accuracy of ± 0.1 °C and requires no calibration. The specifications for this sensor are summarised in Table 3.2.

Table 3.2: TMP117 Specifications

TMP117 Digital Temperature Sensor	
Manufacturer	Texas Instruments
Communication Interface	I^2C [1]
Operating Voltage	1.8 V to 5.5 V
Operating Temperature	−55 °C to 150 °C
Accuracy	± 0.1 °C [2]
Resolution	16-bit: 0.0078 °C
Active Current	0.22 mA
Standby Current	3.1 µA
Extra Features	EEPROM Memory
No. of Pins	6
Package	WSON [3] 2.00 mm x 2.00 mm x 0.8 mm

[1] I^2C interface has four possible slave addresses.
[2] Accuracy for the temperature range −20 °C to 50 °C.
[3] The TMP117 sensor is also available in a smaller DSBGA 1.53 mm x 1.00 mm x 0.5 mm package.

3.5 Geomagnetic Field and tilt Measurements

This section introduces the KMX62 Inertia sensor that fulfils the tilt measurement requirement ***(R4)*** and geomagnetic field strength measurement ***(R5)***.

3.5.1 KMX62-1031 Inertia Sensor (V1)

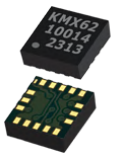


Figure 3.6: KMX62 LGA Package [23]

The KMX62-1031 inertia sensor can sense the magnetic field strength and acceleration in 3 axes. The acceleration data can then be processed to estimate the heading and tilt. The specifications for the sensor are summarised in Table 3.3.

Table 3.3: KMX62-1031 Specifications

KMX62-1031 6 DOF Intertial sensor	
Manufacturer	Kionix
Communication Interface	I^2C
Operating Temperature	−40 °C to 85 °C
Voltage Range	1.7V to 3.6V
Magnetic field range	± 1200 µT
Magnetic sensitivity	± 0.0366 µT
Magnetic field sensor accuracy	± 20%
Accelerometer Full Range	±16 g
Accelerometer Accuracy	±5%
Accelerometer Sensitivity	0.49 mg/LSB [1]
Temperature Sensor Accuracy	± 5 °C
Active Current	0.395 mA
Standby Current	5 µA
Extra Features	Interrupt
Package Size	LGA: 3.0 mm x 3.0 mm x 0.9 mm
No. of pins	14

[1] g is the acceleration due to gravity or 9.81 m/s^2.

3.6 Wireless Interface and Energy Harvesting

This section introduces the Near Field Communication (NFC) transponder to fulfil the wireless interface requirement ***(R7)*** in combination with an extra energy harvesting circuit ***(R8)***.

3.6.1 NFC Transponder (V2)

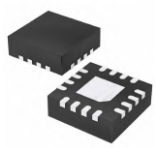


Figure 3.7: RF430CL331H VQFN Package [24]

The RF430Cl331H is a slightly upgraded NFC transponder compared to the one used in version 1. It supports large file transfers and can transmit and receive data wireless at a rate up to 848 Kbps. The chip requires an external antenna and a tuning capacitor to resonate at the NFC resonance frequency of 13.56 MHz. The selected antenna and the tuning circuit are shown in appendix B.3. The NFC transponder fulfils the requirement of zero power consumption when operated at 2V since no current will be drawn.Once an RF field is present the NFC transponder provides an interrupt to the microcontroller and can safely be operated at 2V. The summary of the NFC transponder chip specification is shown in Table 3.4.

Table 3.4: RF430CL331H Specifications (V2)

RF430CL331H Specifications	
Power supply range	2 V - 3.6 V
Communication Interfaces	SPI and I^2C
Operating Temperature	−40 °C to 85 °C
Radio Frequency	13.56 MHz ISO/IEC 14443B and NFC Tag Type 4B
Data rate	Up to 848 kbps
Package size	VQFN: 3 mm x 3 mm x 1 mm

3.6.2 Energy Harvesting (V2)

The energy harvesting circuit is activated upon detection of the 13.56 MHz resonance frequency of the NFC chip. The circuit utilises a half bridge rectifier and a 100 µF capacitor in parallel to the RF430CL331 antenna voltage limiter

to provide a maximum of 3.6V as input to the circuit. More information on this will be discussed in the Design Section 4.2.4.

3.7 Microcontroller

The Microcontroller (MCU) is a vital component of the electronic tag because it controls the entire system. The analysis of choosing a microcontroller depends on power consumption, storage need, required GPIOs, peripherals and package size. Multiple microcontrollers were researched including the Gecko Tiny from Silicon Labs and XLP PIC from Microchip. The results from this research proved that the MSP430 FRAM variant is the best for data storage applications due to the numerous benefits of FRAM [25], and TI's ecosystem.

3.7.1 MSP430FR5738 Microcontroller (V2)

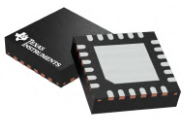


Figure 3.8: MSP430FR5738 Package [26]

The MSP430 is a series of ultra-low-power microcontrollers [27] that come in different models, peripherals and sizes. The first version of the prototype contained an MSP430F2272 , a Flash variant MSP430 microcontroller. This microcontroller was upgraded to the MSP430FR5738 [28]. The technical specifications are summarized in Table 3.5.

Table 3.5: MSP430FR5738 Specification

MSP430FR5738 Microcontroller	
Manufacturer	Texas Instruments
Memory size	FRAM 15.5 KB, SRAM 1KB
Voltage Range	2.0 V to 3.6 V
Operating Temperature	−40 °C to 85 °C
Power consumption	Active mode: 81.4 µA/MHz
	Standby Mode: 1.5 µA LPM3.5 with crystal;
Features	24 MHz CPU, SVS, I2C, SPI, UART (eUSCI) [29], RTC with Calender and Alarm Functions, Radiation Resistance
Package Size	VQFN: 4 mm x 4 mm
No. of Pins	28

3.8 Offline Memory Storage

This section presents the selected FLASH and FRAM memories to fulfill the storage capacity requirement ***(R6)***. These memories are also analyzed with respect to the polling interval in the results section 6.2.

A measurement is defined as the sampling of all sensors and recordings which requires 11 bytes depending on the sensors resolution as seen in Figure 3.9.

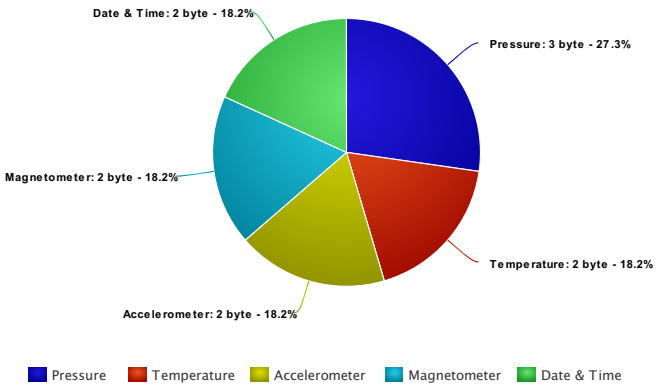


Figure 3.9: Required Memory

Table 3.6: Flash and FRAM Specifications

	CY15B104Q FRAM (V2)	AT45DB641E Flash (V1)
Manufacturer	Cypress Semiconductor Corp. [30]	Adesto Technologies [31]
Communication Interface	SPI	SPI
Memory Size (bit)	8 Mb	64Mb
Possible Mesaurements [1]	90 909	727 237
Clock Frequency	50 MHz	85 MHz
Operating Voltage Range	2.0 V - 3.6 V	1.7 V - 3.6 V
Operating Temperature	−40 °C - 85 °C	−40 °C - 85 °C
Deep Sleep mode current (max)	5 µA	25 µA
Active Write Current (max)	3 mA	22 mA
Package Size	GQFN: 3.22 mm x 3.28 mm x 0.55 mm	5 mm x 6 mm x 0.6 m
No. of pins	8	8

[1] Based on 11 bytes or 88 bit as visualized in Figure 3.9.

3.9 XC6504 Low Dropout Voltage Regulator (V2)

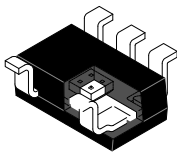


Figure 3.10: XC6504 SOT-25-5 Package[32]

The system has a peak current of ≈ 30 mA and a possible operating voltage range of 2V - 3.6V as summarised in Table 3.7.

Power dissipation $P = I \times V = I^2 \times R = \frac{V^2}{R}$ has an ohmic characteristic, and reducing the operating voltage, reduces the total power dissipation. As a direct result of this, a 2V LDO is chosen from Torex Semiconductor with the model number XC6504. This LDO is chosen for its 2V output and 1.5 µA max supplies current. It has a possible input voltage of 1.4 V - 6 V with an extra CE pin for on/off control.

Table 3.7: Analysis for choosing an LDO.

Component	Voltage Range	Peak Active Current (mA)
U1 - MCU (MSP430FR5738)	2.0 V - 3.6 V	1.95 mA[1]
U2 - Acc. and Mag. (KMX62-1031)	1.7 V - 3.6 V	0.40 mA[2]
U3 - Pressure (MS5837-30BA)	1.5 V - 3.6 V	1.25 mA
U4 - Temp. (TMP117)	1.5 V - 5.5 V	0.22 mA
U5 -Flash (AT45DB641E)	1.7 V - 3.6 V	22 mA
U6 - FRAM (CY15B108QN)	1.8 V - 3.6 V	3.2 mA
U7 -NFC (RF430CL331H)	[3]2.0 V - 3.6 V	0.25 mA
I2C resistors	2 V - 3.6	0.50 mA[4]
Transmitter	Battery Voltage	Battery[5]
Total (DST Only)	2 V - 3.6 V	29.77 mA

[1] Assuming 81.4 µA/MHz and running at 24 MHz.
[2] Operating (mac+accel)
[3] Supply voltage during program execution with RF field present. No RF field present required a minimum of 3 V.
[4] 2 8 kΩ resistors at 2 V $I = \frac{V}{R} = \frac{2 \cdot 2}{8 \cdot 10^3} = 0.5\,\text{mA}$.
[5] The transmitter draws as much current as it needs directly from the battery.

3.10 Battery Choice

This section describes the work carried out to fulfil ***(R9)*** requirements for choosing battery chemistry and capacity. The research on this requirement concluded that Lithium Thionyl Chloride [33] $LI - SOCl_2$ have the largest power densities and longest lifetime among the commercially available batteries.

3.10.1 TL-4902 1/2AA Battery (V2)

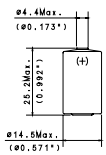


Figure 3.11:
1/2 Battery Size [33]

A 1/2 AA Lithium Thionyl Chloride [33] $LI - SOCl_2$ is chosen for their low self-discharge rate of 1% per year and long operating life depending on the application sampling rate lasting many years. The specifications for the chosen battery are shown in Table 3.8.

Table 3.8: Tadiran Lithum Battery Specifications

1/2AA TL-4902 Battery Specifications	
Nominal capacity @ 0.5 mA, to 2V	1.2 Ah
Rated Voltage	3.6 V
Maximum recommended continuous current	20 mA
Maximum 1 sec. pulse capability	50 mA
Operating temperature	−55 °C to 85 °C
Termination Style	Axial Leaded
Battery Chemistry	Lithium Thionyl Chloride
Diameter	14.5 mm
Height	25.2 mm
Weight	9.6 g

3.11 Overall System Specifications

A block diagram for the hybrid electronic tag can be seen in Figure 3.12. A summary of the overall specification of the data storage tag is seen in Table 3.9.

Table 3.9: Summary of Data Storage Tag Specifications

PARAMETER	SPECIFICATIONS / COMMENTS
Sensors	Temperature, Magn. field strength (3D), tilt (3D), temp., depth
Data Resolution	16 bits for temp/depth, 16 bits for magnetic field strength and acceleration
Magnetic Field Strength Range	26 to 66 uT
Magnetic Field Strength Resolution	100 nT
Magnetic Field Strength Accuracy	±300 nT
Accelerometer full scale range	±16g
Temperature Sensor Range	−1 °C to 40 °C
Temperature Sensor Resolution	0.0078 °C
Temperature Accuracy	± 0.1 °C
Memory Type	FLASH/FRAM
Memory Capacity	727 237 / 90 909 measurements
Data Retention	Flash: 20 years, FRAM: min. 160 years @ 60 °C
Sampling Interval	User specified in seconds, minutes or hours.
Clock	Real time clock. Accuracy ± 14 seconds/month (5ppm crystal).
Communication Interface	Wireless NFC Transponder for setting and retrieving data
Battery Life	2 years minimum
Operating Voltage	2.0 to 3.6 V
Operating Temperature Range	−40 °C to 85 °C
LDO	2.0 V output voltage 150 mA output current

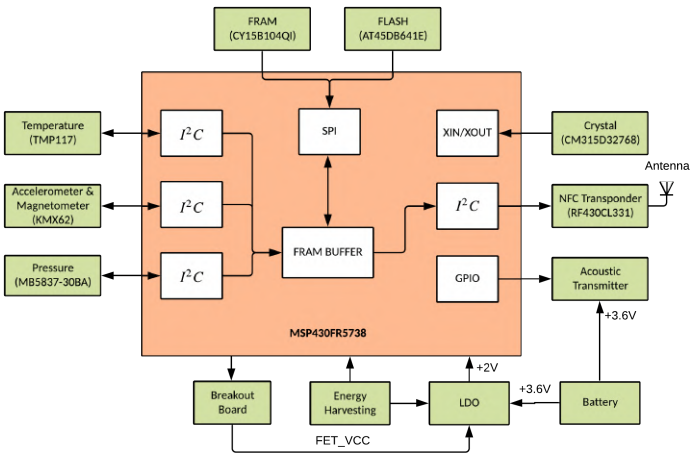


Figure 3.12: Block Diagram with All Devices

4

Prototyping and PCB Design

Contents

This chapter presents the design material for the second version of the printed circuit board (PCB). The chapter also presents the assembly and verification of the PCB Prototype.

4.1 Introduction

PCB design usually consists of drawing schematic diagrams that provide a clear overview of the logical connections between the different components. Today electronic design automation software (EDA or electronic CAD) is often used. This software usually integrates the schematics with the board containing all physical routes, providing a more automated design workflow.

The first version of the prototype was created using Autodesk Eagle with a student license. The software is simple to use and excellent for learning about PCB design and the terminology associated with it [34]. EAGLE software has been in constant development since its initial release in 1988. The speed of this development has dramatically increased since Autodesk Inc. acquired EAGLE in 2016. Autodesk Eagle can be customised with extra functionality using ULP [1] [35], although this customising is somewhat limited and not always version backwards compatible.

The second version of the prototype started as revision 1.1.0 in Autodesk Eagle as seen in the revision history in Figure C.6. In then migrated to Altium Designer 19 for being more effective than Autodesk Eagle, especially in the fields of 3D PCB, smart PDF schematics, assembly drawings and the cross probing between PCB and Schematics. The latest revision (2.0.0) consists of totally revised schematics and a PCB board that is a four layers board with components on both sides. The PCB is now designed with ease of assembly and debugging in mind, and it does not sacrifice the pros of functionality or increased size.

4.2 Schematics

This section briefly explains the prototype's design schematics that are found in Appendix C.

4.2.1 I^2C Sensors

The connection for the I^2C sensors is shown in Figure C.3. The value for decoupling capacitors is chosen as recommended in the datasheet. Two connections are left initially unconnected GPIO2 on the KMX62 chip which is a general interrupt pin. Only one interrupt is needed, and that is GPIO1 labelled as MAG-ACC-INT1. The second initially unconnected pin is the ALERT pin on

[1] User Language Programs

the TMP117 temperature sensor that can trigger an interrupt[2] upon a set, predefined temperature.

4.2.2 Memories

The SPI bus is used to communicate with the FLASH and FRAM memories shown in Figure C.3. The chips connect to the same SPI interface on the microcontroller with two unique chip enable connections controlled by the microcontroller.

4.2.3 Near Field Communication

The NFC Interface connects to the I^2C bus, as shown in Figure C.4. The RF430CL331H chip has the ability to connect to either SPI or I^2C. The I^2C bus is chosen to simultaneously retrieve data from the SPI interface from the memories and send it to the chips using I^2C interface. The chip has an external antenna together with C17 that tunes its frequency to 13.56 MHz. The chip triggers an interrupt to the microcontroller with the pin labelled NFC_INT0 because a radio field is present.

4.2.4 Power Sources and Energy Harvesting

The power source during debugging is the MSP-FET debugger tool, which has been chosen for its energy trace capabilities and convenience. While in production, the battery and energy harvesting are used as power inputs. In order to be able to connect and switch between these different power sources, multiple diodes are used to offer reverse polarity protection, as well as to the functionality of power switching. In addition to these diodes, a manual switch is added to provide switching between the MSP-FET and battery. The schematics for this are shown in the power source section in Figure C.2 and breakout board Debug Spy-Bi-Wire interface in Figure C.5.

4.2.5 Microcontroller

The schematics for the MSP430FR5738 is shown in Figure C.2. The microcontroller provides the control of the I^2C and SPI communication bus with 3 zero ohm resistors that provide the functionality of multiplexing the SPI interface with UART. The UART can be used by removing R4-R5 which disconnects the

[2]The interrupt pin can also provide data-ready interrupt. Connecting this pin is recommended in a later revision.

memories from the bus and UART RX, and TX pins can be used that are already connected to the debugger interface as seen in Section 4.2.6.

4.2.6 Programming Interface

The microcontroller can be programmed using the Spy-Bi-Wire (SBW) interface. This is a 2-wire JTAG debugging interface, instead of the 4-wire full JTAG interface. This can be seen in Figure 4.1 with a full description of the test points in Table 4.1.

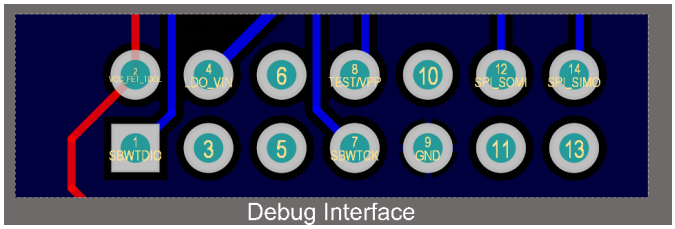


Figure 4.1: PCB: Debugging Interface

Table 4.1: JTAG interface connector pin configuration

Pin	Label	Direction	Description
1	SBWTDIO	IN,OUT	Spy-Bi-Wire data input and output
2	VCC-FET-TOOL	OUT	Power from FET tool
4	LDO-VIN	PWR IN	Feedback from LDO (see note)
7	SBWTCK	IN	Spy-Bi-Wire clock input
8	TEST/VPP	IN	HI-Z, pulled to VCC
9	GND		Ground supply
12	SPI-SOMI	OUT	UART TXD
14	SPI-SIMO	IN	UART RXD

Note: Pin 4 should be connected to the BAT+ and not LDO-VIN, as MSP-FET automatically turns off the power from the FET-tool (pin 2) if it detects any power. A workaround to receive power from FET is to remove pin 2, which will cut the feedback. In this case, the correct logic level needs to be configured if debugging on battery power.

4.3 Version 2 PCB Design Features

The second PCB version of the prototype solves the assembly issues caused by small margined solder pads on some of the components. All parts' sizes are now slightly extended compare to data from the datasheets, as shown in Appendix C.5. The extended pads make assembly easier by following the guidelines for designing solder pads for QFN packages from NXP [36] and taking considerations for manual rework from TI [37].

The main board size is now reduced by 50% as defined in Requirement ***R10***. This reduction makes it necessary to place components on both sides of the PCB and upgrade from two to four layers PCB board. More information about the layer stackup is given in section C.2. The physical PCB design is shown in Figure 4.4 and now consisting of three physical parts:

1. The main board consists of all components that are to be placed in the final production tag with a size of 40 x 14.5 mm.

2. A breakout board for the pressure sensor with a size of 7.24 mm x 6.91 mm. This includes headers for connecting more I^2C peripherals upon future expandability, such as adding a heart rate monitor sensor.

3. A breakout board for debugging, test points, and switch to change the power source, and reset the circuit with the size of 20 mm x 40 mm.

4.3.1 Design Considerations

Many considerations were taken while designing the second version of the prototype, and they are summarised below:

- High-current writing effects: high current, such as the current needed by the flash memory, is 20 mA. This can affect the readings on some of the sensors such as magnetic field measurement. Therefore, the component is placed slightly away (a few millimetres) from the Flash memory, which is the component that has the highest peak current consumption.

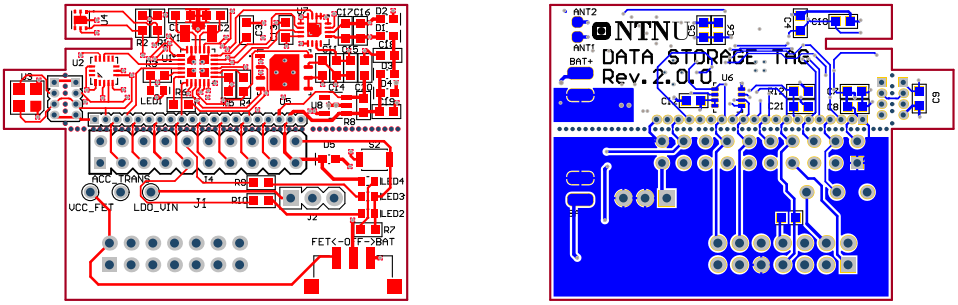
- Crystal Layout: the crystal depends on the load capacitance to operate, as explained in Section 3.2.1. As such, it is necessary to have this capacitance as accurate as possible. Having the crystal traces (XIN and XOUT) routed slightly away, while as short as possible, minimises this capacitance. This is a design choice to help maintain the correct frequency from the crystal.

- Temperature sensor isolation: the temperature sensor is isolated physically by making milling around it to minimise heat from the rest of the circuit. The sensor is also far from the LDO and Flash Memory. These are the two elements that produce the most heat.
- Pin 1 indication: in order to make assembly faster and correctly orient the components, a pin one mark is placed on the silk layer. In addition to the silk layer, the pad shape for the first pin is made to have a rectangular shape rather than an oval one.

4.3.2 Packages Choice and Size

The passive components, such as capacitors and resistors, are chosen to have the 0603-package size, which has the dimensions 1.6 mm x 0.8 mm. According to the datasheets, these components can be mounted using both solder reflow or manual soldering. There are, in total, 41 passive components, as summarised in Table 4.2. However, it is possible to reduce their size to 0402 (0.4 x 0.2 mm). This process would require reflow only, for the smallest packages are hard to manipulate by tweezers. All of the other components are chosen to have the smallest size, except the microcontroller, which is chosen as the 24 pins VQFN 4 x 4 mm package. There also exists a YQD package with half of the size 2 x 2 mm body size. More on the topic of miniaturisation is discussed in Section 7.2.

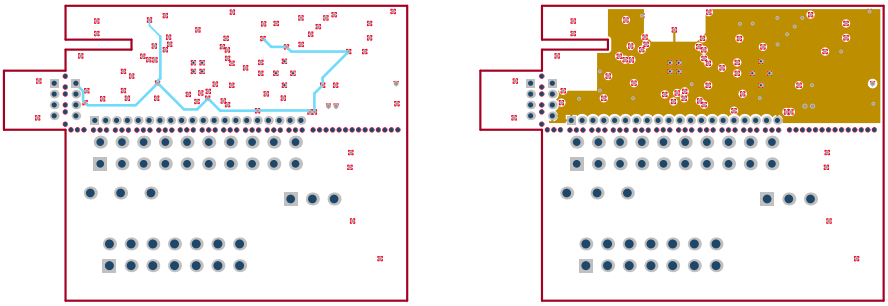
4.3.3 PCB Routing Layers



(a) Top Signal Layer **(b)** Bottom Signal Layer

Figure 4.2: PCB: Signal Routing Layers

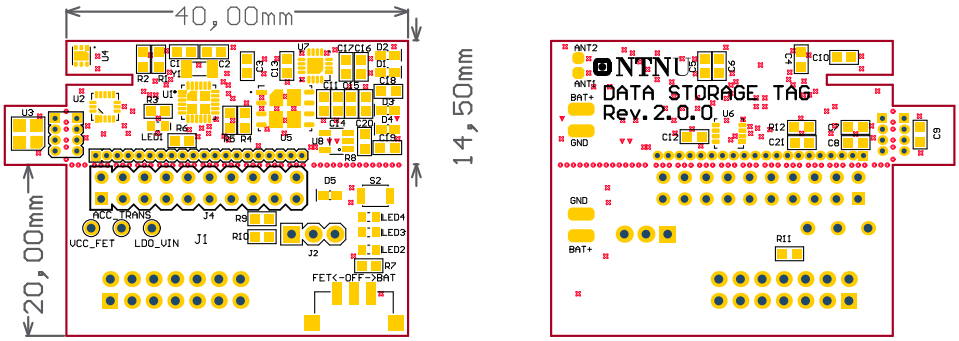
4.3.4 PCB Power Layers



(a) Power Layer **(b)** Ground Layer

Figure 4.3: PCB: Power and Ground Layers

4.3.5 PCB Silk Layers



(a) Top Silk Layer **(b)** Bottom Silk Layer

Figure 4.4: PCB: Top and Bottom Silk Layers

Table 4.2: PCB Information

Board Information	
Board Size	47.5mm x 34.8 mm
Layers	4 layers
Connections Routed	205
Vias	72 (plated)
Components on board	58
Component on top layer	46
Components on bottom layer	12
Components	
Capacitors	20
Chips	8
Crystal	1
Diodes	5
Headers	6
LEDs	4
Resistors	12
Switches	2

4.4 PCB Assembly Drawings

This section contains top and bottom assembly drawings where red indicates DNI (Do not Install) components. C18 is a tuning capacitor for NFC antenna, and the red headers are for adding external sensors to the I^2C bus.

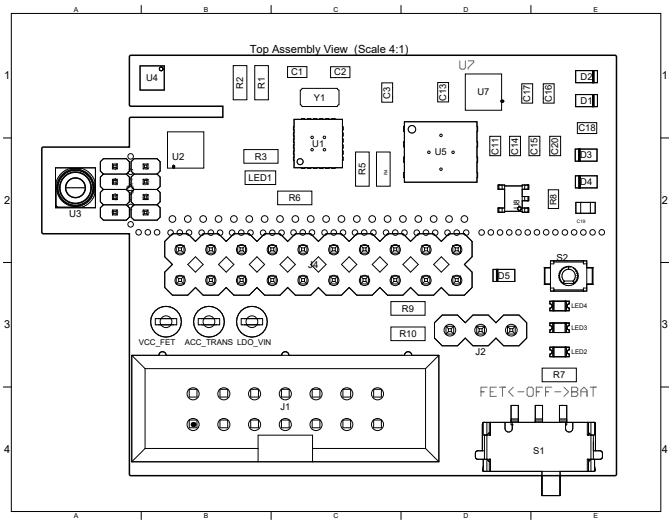


Figure 4.5: PCB: Top Assembly View

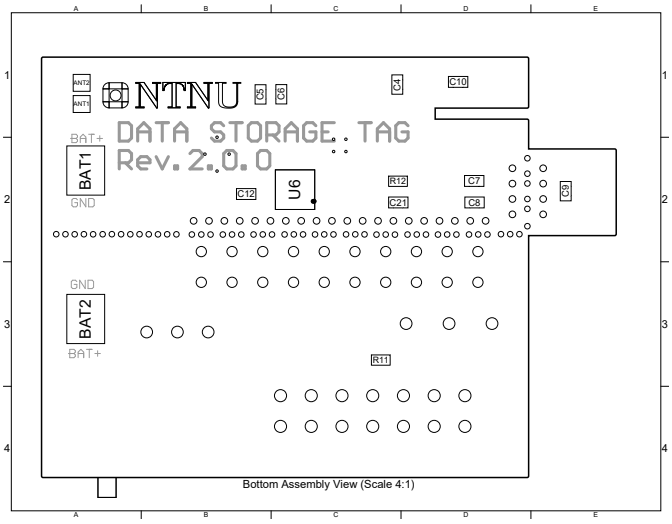


Figure 4.6: PCB: Bottom Assembly View

4.5 Bill of Materials

A full list of materials can be seen in Table 4.3, the components are ordered from Mouser and Digikey.

Quantity	Designator	Description	Qty	Manufacturer	Part Number	Package Reference	Body Size
1	Antenna	Antennas REC NFC ANT W/O Ferrite 15X25	1	Molex	146236-0001	NFC ANT	15 mm x 25 mm
2	VCC FET,VCC LDO	Red PC Test Point	2	Keystone Electronics	5000	Hole Diameter Mounting	1.02 mm
3	ACC Trans	Black PC Test Point	1	Keystone Electronics	5001	Hole Diameter Mounting	1.02 mm
4	C1,C2	25pF ±5% 50V Ceramic Capacitor C0G, NP0	2	Yageo	CC0603JRNPO9BN250	0603	1.6 mm x 0.8 mm
5	C3,C9,C13	0.47µF ±10% 16V Ceramic Capacitor X7R	3	Yageo	CC0603KRX7R7BB474	0603	1.6 mm x 0.8 mm
6	C4,C6,C7,C8,C10,C11,C12,C15	0.1µF ±10% 16V Ceramic Capacitor X7R	8	Yageo	CC0603KPX7R7BB104	0603	1.6 mm x 0.8 mm
7	C5	47µF ±20% 2.5V Ceramic Capacitor X6S	1	Yageo	GRM188C80E476ME05D	0603	1.6 mm x 0.8 mm
8	C17	30pF ±1% 100V Ceramic Capacitor C0G, NP0	1	AVX	06031A300FAT2A	0603	1.6 mm x 0.8 mm
9	C18	100µF ±20% 6.3V Ceramic Capacitor X5R	1	Murata	GRM21BR60J107ME15L	0805	2.0 mm x 1.25 mm
10	C21	2200pF ±10% 16V Ceramic Capacitor X7R	1	Yageo	06031A300FAT2A	0603	1.6 mm x 0.8 mm
11	D1,D2,D3,D4,D5	DIODE SCHOTTKY 30V 100MA 0603	5	Comchip	CDBU0130L	0603	1.6 mm x 0.8 mm
12	JTAG	14-pin connector, male	1	Assman	AWHW14G-0202-T-R	14 POS	2.54 mm pitch
13	J2	Connector Header Through Hole 3 position	1	Harwin Inc.	M22-2581005	20 POS	2.00 mm pitch
14	J4	Headers & Wire Housings 10+10 DIL VERTICAL SMT HEADER GOLD+TIN	1	Harwin	M20-8761042	20 POS	2.54 mm pitch
15	LED1	Red 630nm LED Indication - Discrete 1.8V	1	OSRAM	LS L29K-G1H2-1-Z	0603	1.6 mm x 0.8 mm
16	LED2	Orange 606nm LED Indication - Discrete 1.8V	1	OSRAM	LO L29K-H2K1-24-Z	0603	1.6 mm x 0.8 mm
17	LED3	Yellow 587nm LED Indication - Discrete 1.8V	1	OSRAM	LY L29K-J1K2-26-Z	0603	1.6 mm x 0.8 mm
18	LED4	Green 570nm LED Indication - Discrete 1.7V	1	OSRAM	LG L29K-F2J1-24-Z	0603	1.6 mm x 0.8 mm
19	R1,R2	RES SMD 7.87K OHM 1% 1/10W 0603	2	Yageo	RC0603FR-077K87L	0603	1.6 mm x 0.8 mm
20	R3,R7,R9,R10	RES SMD 50 OHM 1% 1/10W 0603	4	Vishay Dale	CRCW060350R0FKEA	0603	1.6 mm x 0.8 mm
21	R4,R5,R6	RES SMD 0 OHM JUMPER 1/10W 0603	3	Yageo	RC0603FR-070RL	0603	1.6 mm x 0.8 mm
22	R8,R11	RES SMD 330 OHM 1% 1/10W 0603	2	Yageo	RC0603FR-07330RL	0603	1.6 mm x 0.8 mm
23	R12	RES SMD 47K OHM 5% 1/10W 0603	1	Yageo	RC0603JR-0747KL	0603	1.6 mm x 0.8 mm
24	PCB	47.5 mm x 34.8 mm 4 layer board	1	OSH Park			
25	Rubber stand off	Bumpers	3	3M	SJ5302-R25		
26	S1	Slide Switch SPDT Surface Mount, Right Angle	1	NKK	SS312SAH4-R	SMD	Switch
27	S2	Tactile Switch SPST-NO Top Actuated Surface Mount	1	Omron	B3U-1000P	SMD	Switch
28	U1	CPUXV2 MSP430™ FRAM Microcontroller IC 16-Bit 24MHz 16KB (16K x 8) FRAM	1	Texas Instruments	MSP430FR5738IRGER	24-VQFN	4 mm x 4 mm
29	U2	Accelerometer, Magnetometer, 3 Axis Sensor I^2C Output	1	Kionix	KMX62-1031-SR	16-LGA	3 mm x 3 mm
30	U3	Pressure Sensor 435.1PSI (3000kPa) Absolute 24 b 4-SMD, No Lead	1	TE Connectivity	MS583730BA01-50	4-SMD	3.3 mm x 3.3 mm
31	U4	Temperature sensor	1	Texas Instruments	TMP117MAIDRVR	6-WSON	2 mm x 2 mm
32	U5	FLASH Memory IC 64Mb (264 Bytes x 32K pages) SPI 85MHz	1	Adesto Technologies	AT45DB641E-MHN-Y	UDFN	5 mm x 6 mm
33	U6	FRAM 4Mb Inrush CC FRAM 20 MHz SPI	1	Cypress Semiconductor	CY15B104QI-20LPXCES	GQFN-8	3.23 mm x 3.28 mm
34	U7	NFC Dynamic Transponder	1	Texas Instruments	RF430CL330HIRGTR	16-QFN	3 mm x 3 mm
35	U8	2.0V LDO	1	Torex Semiconductor	XC6504A201MR-G	SOT-25-5	LDO
36	Y1	32.768kHz ±5ppm Crystal 12.5pF 70 kOhms 2-SMD, No Lead	1	Citizen	CM315D32768HZFT	2-SMD	3.20 mm x 1.5 mm

Table 4.3: Bill of Materials

4.6 3D Render of PCB

This section contains realistic 3D renders of the board seen in Figure 4.7. An interactive model of the 3D model can be viewed using Adobe Acrobat and is found in the documentation website presented in Appendix D.

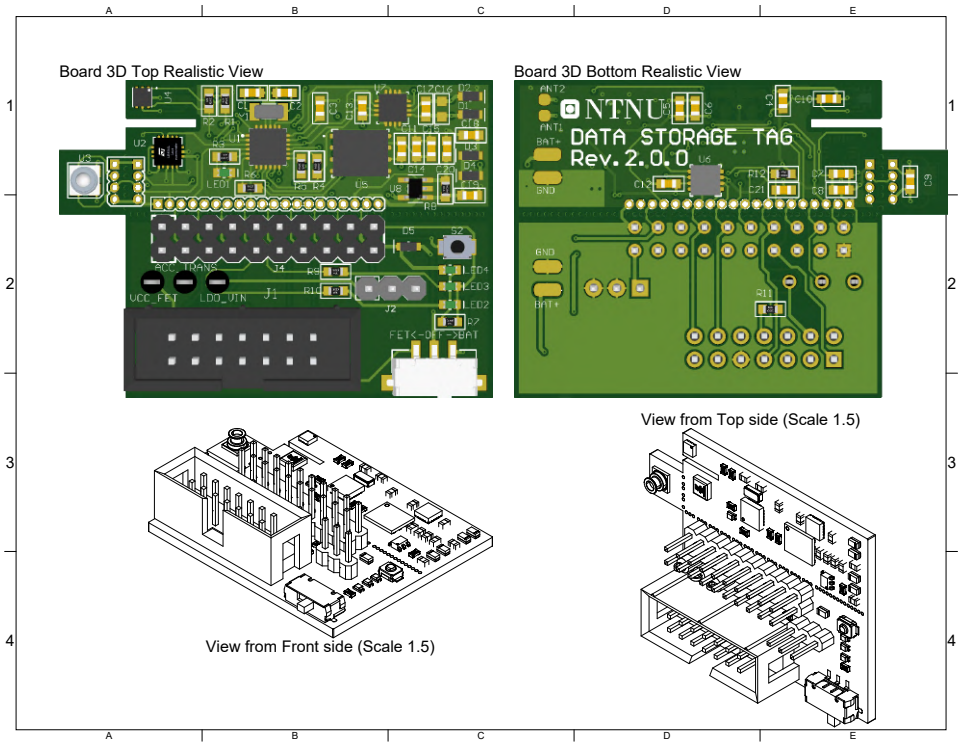


Figure 4.7: PCB: Realistic 3D Views

4.7 Breakout Board

The breakout board has test points that can be connected to an oscilloscope and a logic analyser to analyse the signals and verify the code during development. This breakout board has 26 test points shown in Figure 4.8.

Function	Label	Pin
I^2C Bus	SDA & SCL	2 and 3
SPI	SPI-SOMI, SPI-SIMO and SPI,CLK	12, 14 and 15
UART	SPI-SOMI and SPI-SIMO	14 and 14
SPI Chip Select	SPI-CS and SPI-CS2	11 and 13
SBW Interface	SBWTDIO,SBWTCLK	7 and 8
Detect RF Field	NFC-INTO	6
Acoustic Transmitter	ACOUSTIC-Trans	4
Power	LDO-VIN, GND	19 ,1 and 20
SMCLK and MCLK	LED2 and LED3	10 and 12
Flash Memory Reset	FLASH-RST	17
NFC Transponder Reset	NFC-RST	5

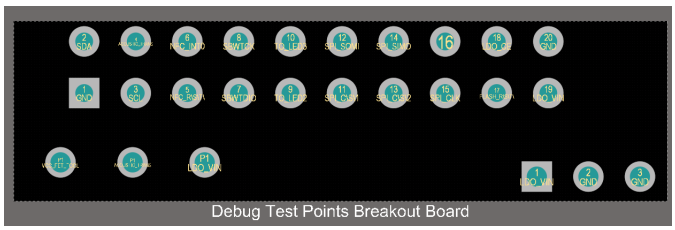


Figure 4.8: PCB Debug Test Points

Note: Please be careful breaking off the break-out sections. Due to the small form factor and small trace width, it is recommended to use a saw to remove the sections rather than breaking them away.

4.8 Prototype Assembly V2

The PCB assembly was done at NTNU cybernetics electronic laboratory using the tools in Table 4.4. A combination of reflow oven and manual soldering using a soldering iron and hot air was used. Four components from each component (bill of material in Table 4.3) then the following procedure is followed:

1. Wipe the workspace with a paper cloth and IPA if necessary. Turn on the Fume extractor while soldering.
2. Tape down the L shaped fixture to the workspace, place the PCB into the L-shape fixture. Place the second L-shaped fixture on the opposite corner and Tape down again. Align the stencil over PCB and Tape down the flat edge to the L-shaped bracket.
3. Apply solder paste on the edge of the stencil and squeegee into stencil holes to deposit the solder paste on the PCB.
4. Setup the reflow oven profile with three stages to match Figure C.8. Stage one (Pre-heat) 100 °C - 30 s, stage two (Pre-heat and soaking) 150 °C - 120 s, Stage three (Reflow) 235 °C - 60 s and finally cool down for 30 second.
5. Adjust the ovens PCB holder to match the size of the PCB, notice that this might move after opening the oven door due to friction (carefully open the door). Run the profile without the PCB inserted to verify the profile stages.
6. Bake the PCB using the reflow profile in Figure C.8 and let it cool down for an extra 30 seconds before picking it up.
7. Bring the PCB to the digital microscope to check for solder bridges if any (most likely will be), they can easily be removed by applying a drop of flux and quickly applying heat in swiping motion to remove the bridges. This technique is very effective on QFN packages and faster than using solder wick.
8. Check the top side for short circuits by setting the multimeter on continuity and use pointy needle probes to check all neighbours pins for short circuits as well to I^2C, SPI, VCC-GND, and LDO power.
9. Flip the board, apply solder paste and solder the components manually. Tweezers can be used to hold down the component while soldering, and this is especially necessary for passive components as they might flip (or tombstone) if not held down.

10. Check for short-circuits again by probing, following the same procedure in step 8.

11. Wear gloves and clean the board from flux residues using IPA and/or flux remover. The flux used is a no-clean although it is still recommended [38] to clean the board to clean off any flux residues.

The results of the assembly are shown in Figure C.17.

Table 4.4: PCB Assembly Tools

Main Tools	
• Digital Microscope • Hot air soldering station	•Solder Fume Extractor • Reflow Ovn
Soldering Tools	
• Solder paste (Sn63/Pb37), No-clean syringe • TS80 Soldering Iron with TS-D25 solder tip.	• Solder Wire Sn95,5Ag3,8Cu0,7 Liquid Flux (No-clean)
Handling	
• IC SMD vacuum sucking pen • Solder Tape	• Anti-magnetic, ESD fine tip tweezers
Cleaning	
• Gloves • Flux remover • Iron Tip Cleaner with Rosin Flux	• Iorpropyl Alcohol (IPA) • PCB rework ESD cleaning brush

4.8.1 Summary of PCB Limitation

The results from assembling and testing the PCB has been greatly successful, although there are a few discovered limitations that are discussed below for future reference.

- Zero ohm resistors and UART Routing: The MSP430FR5738 has hardware support for UART only by using eUSCI_A0, which is multiplexed with SPI and used by the FLASH and FRAM memories. To be able to use the eUSCI UART support the memories need to be physically disconnected, something that is achieved by adding 0 ohm resistors. A discovered routing mistake is that traces for RX and TX lines used by UART are placed after the 0 ohm resistor rather than before, this makes the 0 ohm resistors useless as they would disconnect both UART interface coming from the MSP-FET debugger as well to the memories MISO and MOSI lines. A future revision should fix this routing mistake by placing TX and RX routes on the MCU and 0 ohm resistor rather than 0 ohm and memories. Another solution is to implement bit banging to have a dedicated UART interface as multiplexing the SPI is not recommended according to the flash memory datasheet due to the possible introduction of signal reflection.

- MSP-FET and VCC-Target: This line is unnecessary to connect, future revision should consider leaving this line unconnected as connecting LDO-VIN is not correct as describes in the power results in section 6.1.1. The line can be connected to BAT+, but manual voltage adjustment in the MSP-FET target voltage would still be needed as it does not automatically regulate it down to 2V.

- TMP117 Alert Interrupt: Having this line unconnected adds an extra dependency on timers to save the MCU from waste power while waiting for the TMP117 to finish converting and averaging the temperature. A better solution is to route this interrupt line and depend on interrupts as it will reduce the code complexity.

This page has been left intentionally blank.

5

Firmware Development

Contents

This chapter starts by introducing the MSP430FR5738 microcontroller, specifically its hardware and their functionality, then discusses the coding techniques that are used to develop a firmware. A high-level image of the firmware is then introduced. The code for the developed firmware is attached in Appendix D.

5.1 Microcontroller Introduction

This section explains the functionality of the MSP430FR5738 block diagram (Figure 5.1). We will start by explaining the different modules that are provided by the microcontroller hardware and how they will be utilized in the electronic tag platform.

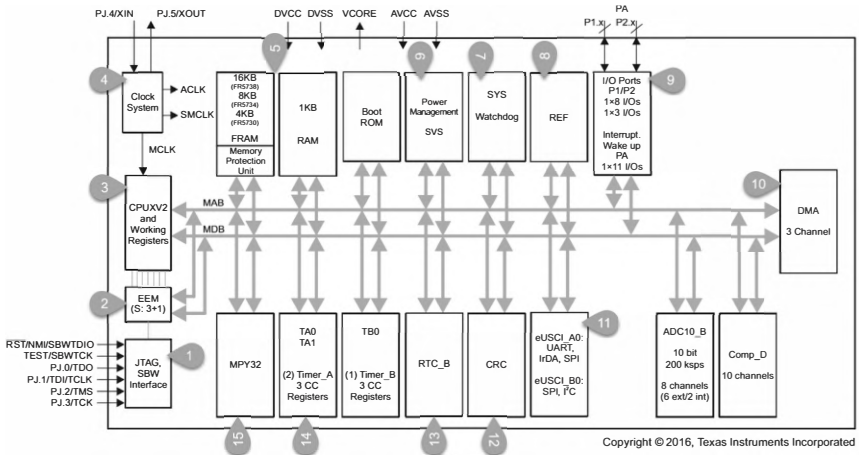


Figure 5.1: MSP430FR5738 Block Diagram

1. JTAG/SBW Interface: the SBW Interface is used for debugging and programming the microcontroller. This interface requires only two wires instead of four on regular JTAG. The two wires provide identical functionality: the 4-wire JTAG is slightly slower than the SBW, which is fine for this platform, for the speed of programming is not a deal breaker.

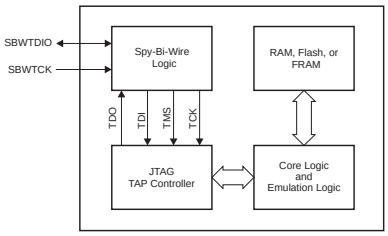


Figure 5.2: Spy-Bi-Wire Basic Concept [39]

2. Embedded Emulation Module (EEM): this module is controlled by the Spy-Bi-Wire interface and provides debugging capabilities when using an IDE, such as CCS. It provides real-time breakpoints, debug in LPM, clock control, and many more features. This module makes debugging and validating the firmware possible [40].

3. CPUXV2: the CPU is a 16-bit RISC CPU (CPUX) with 1MB memory access. The CPU contains a Program Counter (PC) that points to the next instruction to execute. The older MSP430 and MSP430X instruction used a slightly older technology with a smaller memory range. The CPUXV2 is fully compatible with the instruction set out in MSP430 and MSP430X. The CPU instruction sets are somewhat different, but nothing we are directly impacted by as firmware is written in C then converted to assembly instruction set using the C-compiler.

4. Clock System (CS): this is one of the most important modules, for the ATT/DST system is time-dependent and accurate frequencies are required. The module has five clock sources:

   - XT1CLK: this supports the low frequency 32.768 kHz Citizen crystal presented in Section 3.2.1.
   - VLOCK: this is a less accurate low-frequency oscillator with a frequency 10 kHz that is automatically activated upon the failure of XT1CLK.
   - DCOCLK: internal digitally controlled oscillator (DCO) has predefined[1] trimmed frequencies up to 24 MHz.
   - XT2CLK is an optimal external high-frequency oscillator in the range of 4 - 24 MHz. This is not used as the DCO is used to provide HF instead.

   The system depends on four system clock signals that are ACLK, which is sourced from the 32.768 kHz (XT1CLK) and later used to provide accurate timing to the RTC module. MCLK and SMCLK are sourced with a higher frequency from DCOCLK and are used by the CPU and communication peripherals, and lastly, MODCLK, which is sourced from MODOSC and utilised by various peripheral modules.

5. Memory Protection Unit (MPU): this module protects against accidental writes to designated read-only memory. It acts as a file system by allowing to divide the main memory into three segments logically. This module is not implemented in the design of the ATT/DST platform but is considered beneficial to protect constant variables from changes; for example, after the initial system start, the variables are protected from write operations.

6. Power Management Module (PMM) and Supply Voltage Supervisor (SVS): the PPM modules have an internal LDO that produces a secondary core

[1] This is a difference between the MSP430F2272 (formal MCU) and the MSP430FR5738 which does not allow to trim the DCO frequency. More on this is discussed in Section 6.4.1

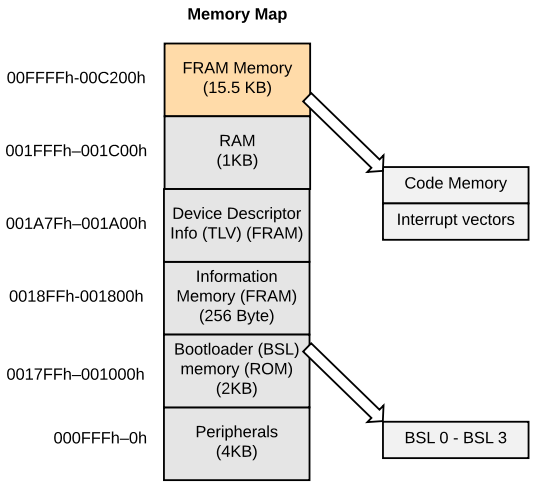


Figure 5.3: MSP430FR5738 Memory Organisation

voltage V_{Core} from the primary one. The module, together with the SVS, can provide interrupts upon power-failure that are accessible by the RST/NMI interrupt vector. The module was initially thought out to be used for detecting power failure, but, since it is fully turned off in LPM3.5 to save power, a different approach is suggested in Section 6.3.

7. System Control Module (SYS) and watchdog: this module talks to the SVS module and is responsible for the interaction between various modules throughout the system. The watchdog is a 32-bit timer that can be used to perform a controlled system starts upon a software problem error or firmware hang. The watchdog is automatically configured with approximately 32-ms reset interval using SMCLK. Therefore, the timer is initially halted to prevent reset upon initialisation of the firmware.

8. REF and ADC: these two modules can be used to create a voltage measurement system to monitor the input voltage and ensure power supply stability. This is usually done using a voltage divider with external components. The MSP430FR MCU has a unique property, as described in TI application note [41]. This method was considered as a method to monitor battery voltage. More information on this can be found in Section 6.3.2

9. This module has two channels: eUSCI_A and eUSCI_B. They provide native support for UART, I^2C, and SPI communication protocols. These modules have a built-in hardware state machine and interrupt capability that allows the MCU to be asleep for a portion of the communication

with automatic activation of the necessary clocks.

10. Cyclic Redundancy Check (CRC): this module provides a signature for a given data sequence to check for stored data validity. This is used in the CTPL library to check for a valid image before restoring the application to its saved state.

11. RTC_B: this module provides a Real Time Clock with clock counters, a calendar, and alarm functionality. The module is used to provide time and date stamp, as well as to periodically wake up the microcontroller using the alarm function.

12. TA0, TA1, and TB0 Timers: these timers can be configured to compare and trigger an interrupt upon a timer overflow. TA0 is used to put the system into LPM3.

13. MPY32: this module is a 32-bit hardware multiplier for multiplying signed, unsigned and fractional numbers. The module is not a part of the CPU.

5.2 Development Environment

The development environment for the firmware consisted of Code Composer Studio (CCS) IDE with MSP-FET [42] debugger to code and verify the system, in order to achieve these multiple development boards were used in this process as seen in Table 5.1.

Table 5.1: Development Setup

Item	Version	Description
Code Composer Studio	9.0.0 64bit	Development IDE (Software)
MSP-FET	v2.06	Stand-Alone Debug Probe (Hardware)
DLP-RF430BP	v1.2	NFC Transponder Booster Pack (Hardware)
DLP-7970ABP	V4.3	NFC Transceiver Booster Pack (Hardware)
MSP-EXP432P401R		Evaluation Board (Hardware)
MSP-TS430RHA40A		Development Board for MSP430FRxx FRAM MCUs (Hardware)
TMP 117EVM		Temperature Sensor Development Tool (Hardware)
DST Prototype	2.0.0	Fully Assembled Prototype

-

5.3 Firmware Introduction

The firmware is written with flexibility in mind to maximise portability, modularity and re-usability of the code. As a result of this design approach, the firmware is broken up into individual software modules. The suggested firmware design has a total of twenty-one modules, of which eleven are driver modules, eight are interface modules and one algorithm module. The suggested design for the firmware modules is illustrated in Figure 5.4. The firmware code is attached in Appendix D as well to a guide for getting started with the development

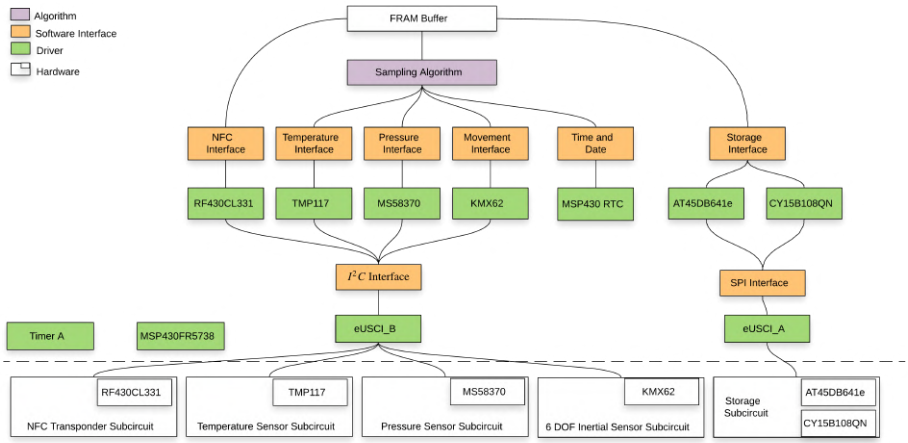


Figure 5.4: Firmware Module Blocks

Note: The firmware has many modules and not all of them are fully completed/tested at the time of delivery of this thesis, although it does provide the basis for an ATT/DST platform and designed for future expansion.

5.4 Main Program Flow

The most important design consideration when developing the firmware is that it consumes as little power as possible to extend the lifetime of the ATT/DST tag, as specified in chapter 2. In this thesis this is achieved by using a periodic timer interrupt that wakes up the microcontroller at a set interval,

samples all of the sensors then puts all of the sensors and microcontroller to sleep. This is the main program flow for the firmware and it is shown in Figure 5.5.

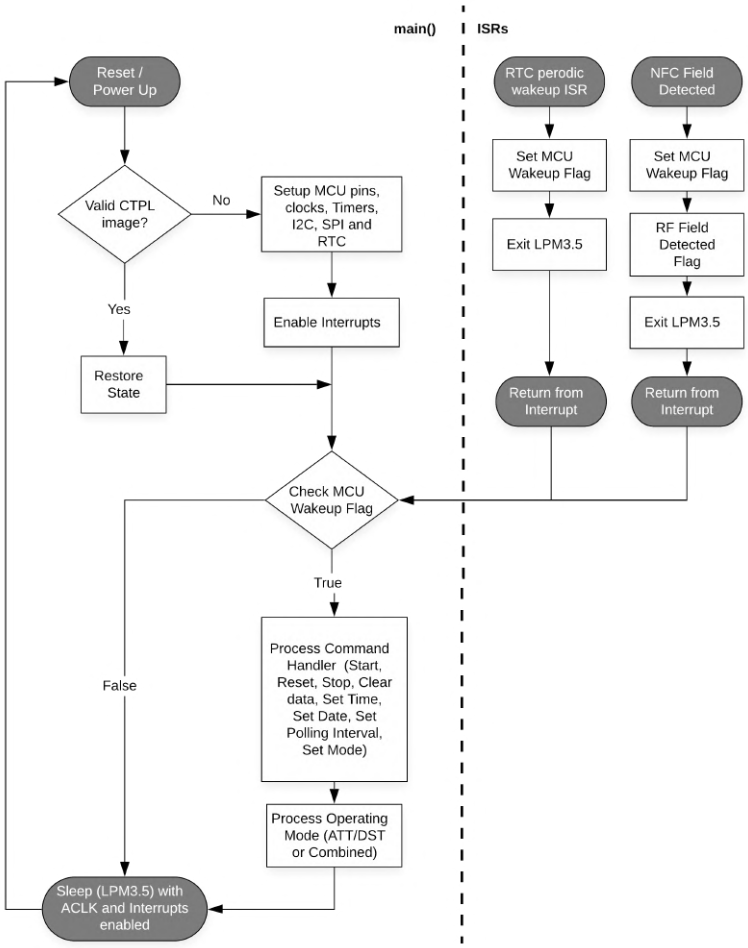


Figure 5.5: Main Program Flow

5.5 Firmware Size Consideration

The MSP430FR5738 has 15 kB FRAM memory, special considerations must be taken into account when declaring data types due to the small size of the memory. The use of fixed length integer can help manage variables and expected sizes, the C99 data types shown in Table 5.2 are used in all of the variable dec-

larations in the firmware.

Table 5.2: Data Types *intx_t and uintx_t defined in stdint.h

Data Type *	Size	Range of Values
int8_t	1 byte signed	-128 - +127
uint8_t	1 byte unsigned	0 - 255
int16_t	2 byte signed	-32768 - +32767
uint16_t	2 byte unsigned	0 - 65535
int32_t	4 byte signed	-2147483648 - +2147483647
uint32_t	4 byte unsigned	0 - 4294967295
int64_t	8 byte signed	-2^{63} to $+2^{63}-1$
uint64_t	8 byte unsigned	0 to $+2^{64}-1$

6 Results

Contents

This chapter presents the validation results from testing the prototype and some additional results that are related to the requirements set in Chapter 2.

6.1 System Validation Testing

Multiple tests are conducted to ensure that the prototype is operational and to verify the physical design. The equipment used in these tests are:

- MSP-FET as a debugger and power source.
- A Computer with CCS, NFC Cockpit, and Waveforms Installed.
- Analog Discovery 2 USB Oscilloscope and Logic Analyser.
- TENMA 72-10410 Multimeter.
- Fully assembled prototype.
- NFC Reader Frontend CLRC663 [43].

6.1.1 Voltage rails Power Test

Motivation: the goal of this test is to validate the power delivery rails using the input sources: VCC_FET and BAT+. These sources are expected to have an input voltage range of 3.3V to 3.6V, and an output of 2V stable output.

Method: connect MSP-FET Power and a 3.6V voltage source to BAT+ and measure the output voltages using a voltmeter.

Results: the results of this test were a constant 2V output when sourced from BAT+.

An unexpected result was that the regulated output voltage was measured to 3.3V when powered from MSP-FET. After researching the JTAG communication protocol [39] and testing the VCC-FET voltage rail without the MSP-FET debugger, it was found that this is not a fault of the physical design. Rather a shortcoming with how the MSP-FET debugger communicates with the microcontroller. The MSP-FET expects the debugging voltage to be the same as the one in the CCS configuration, then tries to boost up the voltage using the internal built-in LDO, creating a voltage conflict. This conflict can be solved by merely choosing the target output voltage to be 2.3V, to account for the voltage drop caused by diodes, and a stable 2V will be provided by the MSP-FET.

Workaround: CCS –> Run –> Debug Configuration –> Target –> MSP430 Flash Settings –> Target Voltage (mv) (set this to 2200).

6.1.2 Energy Harvesting Circuit Test

Motivation: this test validates the energy harvesting circuit and the feasibility of fully powering the system, requirement ***R8***, from NFC radio field, as well as in parallel with a battery.

Method: connect the CLRC663 NFC reader from NXP to a PC with NFC Cockpit software. An additional 12V power supply is connected to the development kit to provide maximum power. Enable the RF field from the NFC cockpit and bring the prototype to close proximity from the NFC reader and measure the voltage and maximum current consumption. Repeat the process at different distances from the NFC reader.

Results: the NFC reader successfully powered up the system with a measured current of 17 mA, with an input of 2.2V to the LDO, measured at a distance of less than half a cm. This is enough to power on the microcontroller and read data actively from the flash memory. The read distance is very short due to the bad coupling between the reader and the receiver. This distance can be significantly improved and provide even higher power transfers by optimising the coupling between the reader and the receiver.

This test was also conducted using a mobile NFC reader and Texas Instruments TRF7970A development kits. The results were no power up when using a mobile phone as the NFC field generated is very weak and had a maximum current output of 1.0 mA with a distance less than half a centimetre with the TI NFC reader.

6.1.3 Communication with sensors Test

Motivation: the system needs to have a method to test when sensors are connected and if they fail upon unforeseen circumstances. The I^2C protocol specification (Appendix A.5) determines that each sensor shall send an ACK bit when addressed by the master.

Method: the i2c driver will be used to initiate communication between the MCU and the I2C devices, then the SDA and SCL signals are monitored using a logic analyser. The communication is confirmed to be working correctly upon receiving an ACK bit.

Results: these tests show successful communication with all of the slaves.

6.2 Memory Capacity vs. Polling Interval

The motivation of this test is to fulfil requirement ***R6*** that the storage capacity must be analysed with respect to the polling interval. The results from this analysis, which are attached as an excel sheet in the Extra folder (see Appendix D. Figure 6.1 Show that 8Mbit of FRAM memory would last 18 months with a polling interval of 7 times per hour, while the flash memory can operate at a much higher polling interval of up to 56 times/hour for 18 months. This show high feasibility of both the 8 Mbit FRAM and 64 Mbit FLASH capacities.

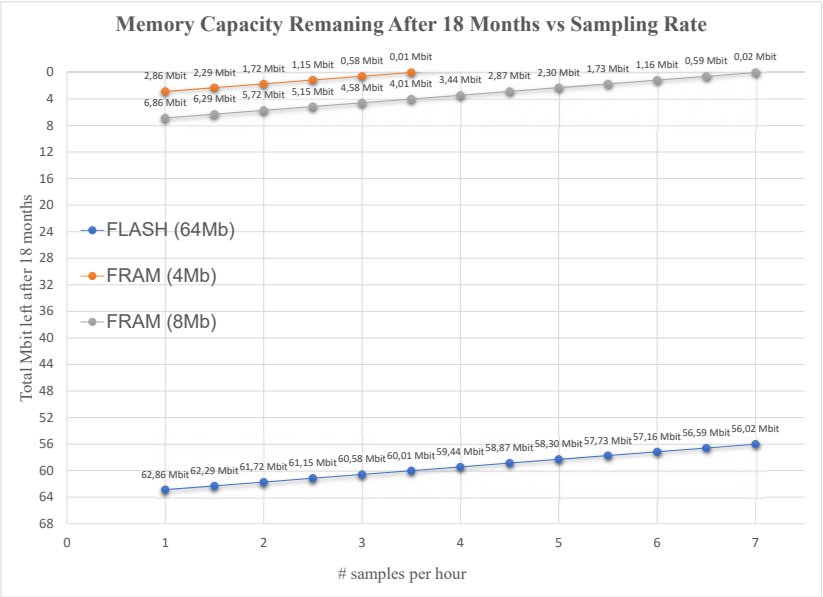


Figure 6.1: Polling Interval vs. Remaining Memory Capacity (18 months)

6.3 Power Analysis

This section discusses the research conducted on detecting power loss, requirement ***(R12)***, and battery life, requirement ***(R9)***.

6.3.1 Battery Life and Power Consumption Estimation

This section explains the work done to theoretically estimate the battery life vs. polling interval. The results are achieved by researching all datasheets for current consumption data and using equation 6.1 to calculate the expected worst-case battery life. The results are shown in Figure 6.1 show an expected battery life of about 12 years with a polling of 6 times an hour. It can also be shown that combining an acoustic transmitter to this budget with a power consumption of 0.45 mWh per day would result in halving the battery life. Hence the battery capacity 1200 mAh exceed the required lifetime of 18 months, and even a smaller battery can be used.

$$\text{Battery life (years)} = \frac{\text{Battery capacity}(mAh)}{\frac{I_{on}(mA)\times t_{on}(s)+I_{off}(\mu A)\times t_{off}(s)}{t_{on}(s)+t_{off}(s)}} \times \frac{1\text{ year}}{8760\text{ hour}} \times 85\% \tag{6.1}$$

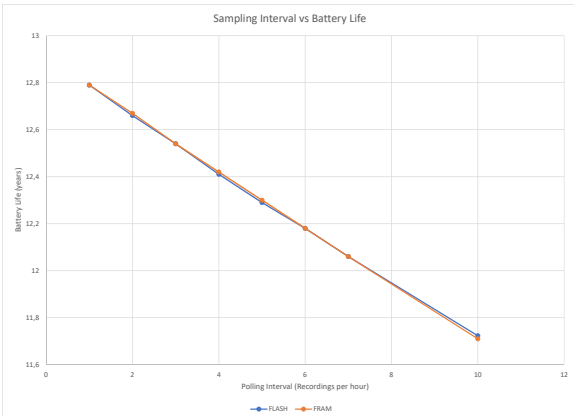


Figure 6.2: Battery Life vs. Sampling Period Estimation

Table 6.1: Power budget for DST system

Component	$I_{Activate}$ (mA)	T_{on}(ms)	$I_{standby}$ (μA)	t_{off}(s)
U1 - MCU (MSP430FR5738)	1.95 mA[1]	167	5 µA[3]	3599.83
U2 - Acc. & Mag. (KMX62-1031)	0.395 mA[3]	21 ms	5 µA	3599.98
U3 - Pressure (MS5837-30BA)	1.5 mA	18 ms	0.1 µA	3599.98
U4 - Temp. (TMP117)	0.22 mA	125 ms	3.1 µA	3599.88
U5 -Flash (AT45DB641E)	0.86 mA [4]	3 ms	0.9 µA	3600
U6 - FRAM (CY15B108QN)	0.075 mA [5]	50 µs	0.9 µA	3600
U7 -NFC (RF430CL331H) [6]	0 mA	0 ms	0 µA	0
I2C resistors	2 V - 3.6	0.50 mA[7]		
Transmitter	Battery Voltage	Battery[6]		
Total (DST Only)	2 V - 3.6 V	29.77 mA		

[1] Assuming 81.4 µA/MHz and running at 24 MHz.
[2] RTC is on, ACLK and interrupts enabled.
[3] Operating (mac+accel)
[4] One write cycle every 23 measurements (one page) $^{20\,\mathrm{mA}}/_{23}$
[5] 2 8 kΩ resistors at 2 V $I = \frac{V}{R} = \frac{2 \cdot 2}{8 \cdot 10^3} = 0.5\,\mathrm{mA}$.
[6] Does not use any power when operated by voltage under 3V.

6.3.2 Monitoring Battery Capacity

This section describes the work done on the matter of monitoring battery capacity for $Li - SOCI_2$ and providing a battery gauge functionality. The motivation for this work comes from complementing the research for the future work section and aiming to provide a smart way to operate the system by switching between the functionality of transmitter and storage tag, in addition to providing an end of service way.

Lithium Thionyl Chloride batteries have a characteristic flat discharge curve, as seen in Figure 6.3. This makes battery monitoring by voltage method and other parameters, such as temperature and impedance, need to be taken into consideration for creating an algorithm that gives the state-of-charge for the battery.

Conveniently, there exist integrated circuits (ICs) that provide battery fuel gauge with the possibility for coulomb accumulation (ACC) and end of service (EOS). These chips can typically measure current, temperature, voltage, and impudence to give an estimate of the battery charge status. Two of the suggested possible chips is the BQ35100 battery fuel gauge IC from TI [44] with a size of 5 mm x 4.2 mm. This is on the large size compared to the rest of the circuit size shown in Table 4.3.

Another approach to solving this problem for "free", without the use of any external components, is by adding a software module that calculates the remaining battery life based on elapsed time. In order to do this, a worst-case current consumption is presented in Section 6.3.1; this data can be used when developing the algorithm.

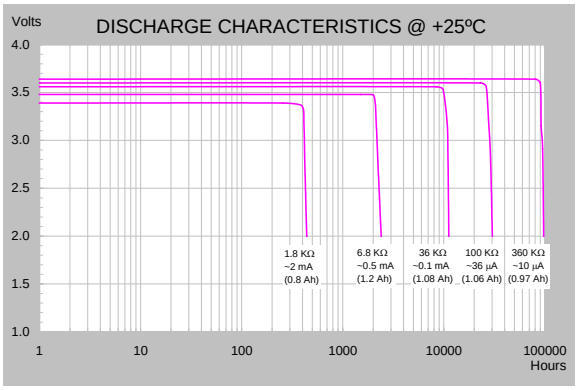


Figure 6.3: $Li - SOCI_2$ Battery Discharge Rate [33]

6.3.3 Methods for Detecting Power Loss

This section discusses the research done on monitoring and detecting power loss to satisfy requirement ***(R11)***.

The MSP430FR5738 has a built-in Supply-Voltage Supervisor (SVS) that can be used to detect a voltage drop on the power input and trigger an interrupt; this is set up using the NMI interrupt vector. The PMM module discussed in Section 5.1, allows for this functionality but is turned off in LPM 3.5. This means an external SVS or a higher LP mode such as LPM3 must be used to facilitate safely shutting down.

6.4 Other Researched Topics

This section briefly discusses some of the touched upon topics during the research in this thesis, and these topics are of interest for future development of the platform.

6.4.1 Tuning the Microcontroller

A part of the ***(R1)*** accurate timing requirement is to provide an accurate frequency for the acoustic transmitter. The ATT transmitters from Thelma Biotel operate at 69 kHz, requiring the use of DCO and dividers to achieve this frequency. The 32.768 kHz can be used to tune the DCO frequency, as it is accuracy is too poor: ± 3.5%. A limitation was found with the MSP430FR5738: it does not allow for software tuning of DCO frequency.

After further investigation, it was found that the MSP430FR2433 microcontroller does provide the possibility for software tuning using the code snippet in Section D.5. This microcontroller provides the same package sizes as the MSP430FR5738 at a slightly higher current consumption and lower maximum CPU speed.

6.4.2 Real Time Operating System

The hybrid electronic tag is a time driven system, and as the complexity of the system increases, it might be desirable to add an OS (operating system) to the microcontroller that manages, for example, wireless connectivity, multitasking, file management, event handling, semaphore, drivers, and diagnostics, such as logs.

In results of this research TI-RTOS [45], [46] with Code Composer Studio (CCS) provides all of the necessary tools and code to easily integrate advanced functionality without the need of starting from scratch, as TI has invested many resources towards creating a deterministic real-time kernel that can be easily integrated into TI microcontrollers free of charge. The kernel provides an SDK with easy to use APIs and training resources of videos and documents to quickly get started with the OS. This OS could be a good alternative when increasing the complexity of the system by, for example, creating a wireless interface (NFC Reader) that configures the system and receives the data and sends them to some remote server for analysing the data and creating a pathfinder software or integrating to the already existing framework from Thelma Biotel and their hydrophones.

6.4.3 Management Graphical Interface

GUI composer is a tool inside Code Composer Studio (CCS) that allows for creating a graphical interface that interacts with the MSP430 microcontrollers. This tool can be used to extend the functionality of the hybrid electronic tag in later revisions of this project by creating a graphical user interface. Such as the NFC Reader that configures the device using the NFC/RFID reader. The firmware also provides functionality for over-the-air programming and code examples for this are provided in the project repository Section A.1.

This page has been left intentionally blank.

7

Conclusion and Future work

Contents

This chapter presents the conclusion for the feasibility of the hybrid electronic tag, as well to provide suggestions for future work and miniaturisation.

7.1 Conclusion

The work conducted in this thesis made a large improvement on the original design created in the specialisation project. The lessons learned from the shortcomings of the first design made it possible to greatly improve and assemble a new electronic tag prototype.

A major priority of this work was to provide a well-documented base for an electronic hybrid platform. This is now done in a systematic way, where a list of requirements completed and the fulfilment of each requirement is discussed.

The new design proves to be highly feasible with respect to battery lifetime, miniaturisation possibilities and using it in commercial products. The added wireless interface and energy harvesting opened up for exiting development possibilities that will be discussed in the Future work section.

There is still a need for more firmware code to test the tag in open water and verify the feasibility of the SST and IGRF matching methods. The coding work should be simplified due to the work done in this thesis, especially the code examples for firmware that use the same components. This is a time consuming process, which can be overcome using the "getting started guide" that is presented in Appendix D.

7.2 Tag Miniaturisation

The hybrid solution had the requirement ***(R10)*** of the physical size to be as small as possible. The new prototype design fulfils ***(R10)*** and shows that it is possible to cut down the size by half from the first version of the prototype or Rev.1.0.0. Currently the new PCB design has over 200 routed connections, 72 vias and 58 components as summarised in Table 4.2. The design methodology followed in prototyping (Revision 2.0.0), is to place small through-hole connection points on the edges of the PCB to allow the tag to be build up vertically by having multiple small boards stacked up on top of each other and horizontally for testing and debugging purposes. This design technique can be optimised for the minimisation process by using a Rigid-Flex PCB where all boards are manufactured at the same time and connected using flex cables. Figure 7.1 shows a mock-up of this suggested design concept. Further, the size of the main board can be achieved by using the smallest package sizes as formally discussed in Section 4.3.2. This is especially relevant for passive components such as capacitors, resistors, and diodes. It is estimated that the length can be reduced by another 10-20 mm by doing this and utilising both PCB sides.

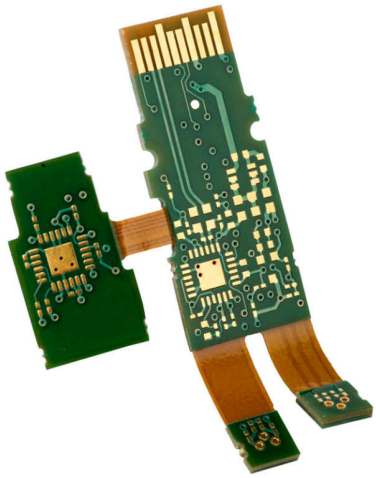


Figure 7.1: Rigid-Flex PCB Design Concept For Future Miniaturisation [47]

7.3 Future Work

This section provides some possible future work suggestions for the hybrid electronic tag platform.

1. Data streaming mode by utilising the NFC Transponder and energy har-

vesting. This is a software implementation inspired by a Texas Instruments reference design [48] where the sensor data are both continuously and in real-time transferred to the NFC reader. This mode gives the advantage of being able to reuse the tags after the battery possibility dies out and the components being in a hermetic sealed with epoxy. Providing such a mode in software is easy to do,for the firmware implementation from TI is based on the same hardware used in the DST prototype.

2. Possibility to use Pathfinder software to analyse the downloaded data. This could be done by infused with the management interface discussed in Section 6.4.3.

3. Over the air firmware upgrade using NFC. This would require an extra component because the RF430CL331 NFC transponder cannot operate without a microcontroller. A possible solution is to add an extra microcontroller dedicated to controlling the RF430CL331 and speaking to the bootloader [49] over UART or I^2C. The bootloader of the MSP430FR5738 supports only UART while MSP430FR2433 has both UART and I^2C bootloader interfaces. An example of achieving this is added in the repository extra folder explained in Section D.1.

4. Battery life estimation[1] algorithm: this parameter could be reported when connecting to the NFC reader by creating an algorithm that estimates the remaining battery based on the on/off time of the devices. The algorithm would accumulate charge, and discharge counts theoretically or by depending on an external circuit such as the one mentioned in Section 6.3.2.

[1] A dynamic excel sheet for battery life estimation is attached in the repository Extra/Research Budgets folder.

This page has been left intentionally blank.

References

[1] "Aquatic animal telemetry: A panoramic window into the underwater world", *REVIEW SUMMARY ECOLOGY*, pp. 1–11, DOI: 10.1126/science.1255642. [Online]. Available: http://science.sciencemag.org/.

[2] D. Kilfoyle and A. Baggeroer, "The state of the art in underwater acoustic telemetry", *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 4–27, Jan. 2000, ISSN: 0364-9059. DOI: 10.1109/48.820733. [Online]. Available: http://ieeexplore.ieee.org/document/820733/.

[3] E. Thorstad, A. H. Rikardsen, A. Alp, and F. Økland, "The Use of Electronic Tags in Fish Research–An Overview of Fish Telemetry Methods", *Turkish Journal of Fisheries and Aquatic Sciences*, vol. 13, pp. 881–896, 2013, ISSN: 1303-2712. [Online]. Available: www.trjfas.org.

[4] "A small long-life acoustic transmitter for studying the behavior of aquatic animals", *Review of Scientific Instruments*, vol. 87, no. 11, p. 114 902, Nov. 2016. [Online]. Available: http://aip.scitation.org/doi/10.1063/1.4967941.

[5] "The value of using measurements of geomagnetic field in addition to irradiance and sea surface temperature to estimate geolocations of tagged aquatic animals", vol. 5, pp. 1–13, 2017. [Online]. Available: https://animalbiotelemetry.biomedcentral.com/track/pdf/10.1186/s40317-017-0134-y?site=animalbiotelemetry.biomedcentral.com.

[6] "Satellite and In Situ Observations for Advancing Global Earth Surface Modelling: A Review", vol. 10, no. 12, p. 2038, Dec. 2018. [Online]. Available: http://www.mdpi.com/2072-4292/10/12/2038.

[7] C. H. Lam, A. Nielsen, and J. Sibert, "Incorporating sea-surface temperature to the light-based geolocation model trackit", *Marine Ecology Progress Series*, vol. 419, pp. 71–84, Nov. 2010.

[8] *Salmon at Sea - 2018 Ongoing & Completed Projects*. [Online]. Available: http://www.nasco.int/sas/research/party18.htm (Accessed: 3 Jan. 2019).

[9] E. B. Thorstad, A. H. Rikardsen, A. Alp, and F. Økland, "The use of eletronic tags in fish research - an overview of fish telemetry methods", *Turkish Journal of Fisheries and Aquatic Sciences*, vol. 13, no. January, pp. 87–94, 2013.

[10] J. D. Metcalfe and G. P. Arnold, "Tracking fish with electronic tags", *Nature*, vol. 387, no. 6634, pp. 665–666, Jun. 1997. [Online]. Available: http://www.nature.com/articles/42622.

[11] N. Jepsen, E. B. Thorstad, T. Havn, and M. C. Lucas, "The use of external electronic tags on fish: an evaluation of tag retention and tagging effects", vol. 3, no. 3, p. 49, 2015. [Online]. Available: https://animalbiotelemetry.biomedcentral.com/articles/10.1186/s40317-015-0086-z.

[12] M. C. Lucas and E. Baras, *Migration of freshwater fishes*. Blackwell Science, 2001, p. 420, ISBN: 9780470999653.

[13] A. Aljumaili, "Miniature combined ATT / DST platform for mapping of large-scale fish migration Supervisor : Jo Arve Alfredsen", Norwegian University of Science and Technology, Tech. Rep. December, 2017, pp. 4–10. [Online]. Available: https://dst-ntnu.bitbucket.io/download/aljumaili2017.pdf.

[14] B. Lingesan and R. Rajesh, "Tilt Angle Detector Using 3-Axis Accelerometer", vol. 4, pp. 2–8, 2018. [Online]. Available: https://www.researchgate.net/publication/324388303_Tilt_Angle_Detector_Using_3-Axis_Accelerometer.

[15] Fujitsu Semiconductor, "FRAM Guide Book", Tech. Rep., 2018, pp. 1–4. [Online]. Available: https://www.fujitsu.com/downloads/MICRO/fme/fram/fram-guide-book.pdf.

[16] D. Kleidermacher and M. Kleidermacher, "Introduction to Embedded Systems Security", in *Embedded Systems Security*, Elsevier, May 2012, pp. 1–24.

[17] Citizen Finedevice Co. LTD, "CM315D 32.768kHz Crystal Datasheet", pp. 1–2, 2018. [Online]. Available: http://cfd.citizen.co.jp/english/prod-tech/product/pdf/datasheet_TF/CM315D_E.pdf.

[18] *Handling Notes- CITIZEN FINEDEVICE CO.,LTD.* [Online]. Available: https://cfd.citizen.co.jp/english/prod-tech/product/cvo_manual.html (Accessed: 30 May 2019).

[19] Texas Instruments, "MSP430™ 32-kHz Crystal Oscillators Application Report MSP430™ 32-kHz Crystal Oscillators", Tech. Rep., 2006, pp. 1–22. [Online]. Available: https://www.ti.com/lit/an/slaa322d/slaa322d.pdf.

[20] TE Connectivity, "MS5837-30BAUltra Small Gel Filled Pressure Sensor Specifications Datasheet", pp. 1–16, 2015. [Online]. Available: http://www.mouser.com/ds/2/418/MS5837-30BA-736494.pdf.

[21] Silicon Labs, "Si7051 I2C TEMPERATURE SENSORS Datasheet", p. 7, 2018. [Online]. Available: https://www.silabs.com/documents/public/data-sheets/Si7050-1-3-4-5-A20.pdf.

[22] Texas Instruments, "TMP117 Diigital Temperature Sensor datasheet", pp. 1–51, 2018. [Online]. Available: https://www.ti.com/lit/ds/symlink/tmp117.pdf.

[23] Kionix, "KMX62-1031 Specifications Datasheet", pp. 1–70, 2017. [Online]. Available: http://kionixfs.kionix.com/en/document/KMX62-1031-Specifications-Rev-4.0.pdf.

[24] Texas Instruments, "RF430CL331H Dynamic NFC Interface Transponder Datasheet", pp. 1–50, 2014. [Online]. Available: https://www.ti.com/lit/gpn/rf430cl331h.

[25] "FRAM-New Generation of Non-Volatile Memory Key Advantages", Tech. Rep., 2009, pp. 1–2. [Online]. Available: www.ti.com/fram.

[26] Texas Instruments, "MSP430FR573x Mixed-Signal Microcontrollers 1 Device Overview", Tech. Rep., 2016, pp. 1–119. [Online]. Available: https://www.ti.com/lit/ds/symlink/msp430fr5738.pdf.

[27] J. Davies, *MSP430 Microcontroller Basics*. 2018, p. 668.

[28] T. Instruments, Tech. Rep., 2014, pp. 1–677. [Online]. Available: https://www.ti.com/lit/ug/slau272d/slau272d.pdf.

[29] Texas Instruments, "Solutions to Common eUSCI and USCI Serial Communication Issues on MSP430™ - Application Report", Tech. Rep., 2017. [Online]. Available: www.ti.com/lit/an/slaa734a/slaa734a.pdf.

[30] Cypress Semiconductor, "CY15B108QN F-RAM Datasheet", pp. 1–28, 2018. [Online]. Available: https://www.cypress.com/file/444186/download.

[31] Adesto Technologies, "AT45DB641E Flash Memory", pp. 1–72, 2013. [Online]. Available: https://www.adestotech.com/wp-content/uploads/DS-45DB641E-027.pdf.

[32] Torex Semiconductor, "XC6504 Small Voltage Regulator", pp. 1–32, 2012. [Online]. Available: https://www.torexsemi.com/file/xc6504/XC6504.pdf.

[33] Tadiran Batteries, "Technical Brochure LTC-Batteries", pp. 1–31, 2018. [Online]. Available: https://tadiranbatteries.de/pdf/Technical-Brochure-LTC-Batteries.pdf.

[34] A. Aljumaili, "NFC PCB with external Antenna Project report in TTK8", Tech. Rep., 2017, pp. 3–10. [Online]. Available: https://dst-ntnu.bitbucket.io/download/ttk8-nfc-report.pdf.

[35] Autodesk, *Autodesk Eagle ULP*. [Online]. Available: https://eagle.autodesk.com/eagle/ulp (Accessed: 18 Apr. 2019).

[36] NXP, "AN1902 Assembly guidelines for QFN (quad flat no-lead) and SON (small outline no-lead)", Tech. Rep., 2018, pp. 1–51. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN1902.pdf.

[37] T. Instruments, "QFN/SON PCB Attachment Application Report (Rev. A", Tech. Rep., 2018. [Online]. Available: http://www.ti.com/litv/pdf/slua271b.

[38] M. Bixenman and B. Tolla, "Why clean no-clean flux", Tech. Rep., 2016, pp. 1–6. [Online]. Available: https://www.kester.com/.

[39] Texas Instruments, "MSP430™ Programming With the JTAG Interface User's Guide MSP430™ Programming With the JTAG Interface", Tech. Rep., 2019, pp. 4–29. [Online]. Available: http://www.ti.com/lit/pdf/slau320.

[40] ——, "Advanced Debugging Using the Enhanced Emulation Module (EEM) With Code Composer Studio Version 6", Tech. Rep., 2008, pp. 1–17. [Online]. Available: https://www.ti.com/lit/an/slaa393f/slaa393f.pdf.

[41] ——, "Low-Power Battery Voltage Measurement With MSP430FR MCU On-Chip VREF and ADC Application Report Low-Power Battery Voltage Measurement With MSP430FR MCU On-Chip VREF and ADC", Tech. Rep., 2018, pp. 1–5. [Online]. Available: https://www.ti.com/lit/an/slaa828/slaa828.pdf.

[42] ——, "MSP Debuggers User's Guide", Tech. Rep. July, 2015, pp. 1–53. [Online]. Available: https://www.ti.com/lit/ug/slau647m/slau647m.pdf.

[43] NXP Semiconductors, "CLRC663 Evaluation board quick start guide Rev. 1.5-28 May 2018 205915 Application note COMPANY PUBLIC Document information Info Content", Tech. Rep., 2018, pp. 3–35. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN11022.pdf.

[44] Texas Instruments, "BQ35100 Lithium Primary Battery Fuel Gauge and End-Of-Service Monitor", pp. 1–9, 2019. [Online]. Available: https://www.ti.com/lit/gpn/bq35100.

[45] ——, "TI-RTOS 2.20 User's Guide", pp. 7–24, 2016. [Online]. Available: https://www.ti.com/lit/ug/spruhd4m/spruhd4m.pdf.

[46] ——, "TI-RTOS: A real-time operating system for TI devices", pp. 1–2, 2015. [Online]. Available: www.ti.com/tool/ti-rtos.

[47] Raypcb.com, *Rigid Flex PCB Design and Manufacturing Steps -RayMing*. [Online]. Available: https://www.raypcb.com/rigid-flex-pcb-design-and-manufacturing-steps/ (Accessed: 15 Jun. 2019).

[48] T. Instruments, "MultiParameter BioSignal Monitor Design Guide", Tech. Rep., 2015, pp. 15–31. [Online]. Available: https://www.ti.com/lit/pdf/tidu875.

[49] Texas Instruments, "MSP430FRBoot-Main Memory Bootloader and Over-the-Air Updates for MSP430™ FRAM Large Memory Model Devices", pp. 1–23, 2016. [Online]. Available: http://www.ti.com/lit/an/slaa721b/slaa721b.pdf.

[50] *Transmitters – Thelma Biotel.* [Online]. Available: https://www.thelmabiotel.com/transmitter/ (Accessed: 4 Feb. 2019).

[51] GHRSST Science Team, "The Recommended GHRSST Data Specification (GDS) 2.0, document revision 4, available from the GHRSST International Project Office", Tech. Rep., 2010, pp. 19–22. [Online]. Available: https://www.ghrsst.org/documents/q/category/gds-documents/operational/.

[52] A. Kaiser-Weiss and P. Minnett, "GHRSST Strategy an Implementation Plan (GDIP)", Tech. Rep., 2012, pp. 17–36. [Online]. Available: https://www.ghrsst.org/about-ghrsst/governance-documents/.

[53] C. Donlon and Robinson, "The Global Ocean Data Assimilation Experiment High-Resolution Sea Surface Temperature Pilot Project", *Bulletin of the American Meteorological Society*, vol. 88, no. 8, pp. 1197–1213, Aug. 2007.

[54] Miljødirektoratet, *Kjemikalier - Miljødirektoratet Regelverk og føringer*, 2019. [Online]. Available: https://www.miljodirektoratet.no/ansvarsomrader/kjemikalier/ (Accessed: 28 Apr. 2019).

[55] *2019 RoHS Compliance Guide: Regulations, 10 Substances, Exemptions*, 2019. [Online]. Available: https://www.rohsguide.com/ (Accessed: 28 Apr. 2019).

[56] COMMISSION DELEGATED DIRECTIVE (EU), "European Parliament and of the Council as regards the list of restricted substances", Tech. Rep., 2015, pp. 1–3. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32015L0863%7B%5C&%7Dfrom=EN.

[57] T. Instruments, "I2C Bus Pullup Resistor Calculation Application Report I2C Bus Pullup Resistor Calculation", Tech. Rep., 2015, pp. 1–5. [Online]. Available: http://www.ti.com/lit/an/slva689/slva689.pdf.

[58] Molex, "Molex (1462360001) NFC Antenna 15 mm by 25 mm", Tech. Rep., 2019, p. 1. [Online]. Available: http://www.mouser.com/ds/2/276/1462360001_ANTENNAS-1110719.pdf.

[59] A. Schweitzer, “PCB Layer Stack-up”, Tech. Rep., 2015. [Online]. Available: https://www.kbl-circuits.com/Layer%20stack%20up.pdf.

A

Background Materials

A.1 Acoustic Transmitters and Hydrophones

ATT functionality and hydrophones can give the possibility for triangulation[1] to study the fish migration path. These projects use different techniques to track the fish, but their exact migration path still has many unknown variables which we wish to gain an insight to. Studies using electronics tags such as DSTs [9] for fish tracking [3], [4], [10]–[12] have a retrieval rate of under 50% and combining a data logger with an acoustic transmitter can increase the retrieval rate and give an accurate estimate of their past movement path and other vital information.

The acoustic signal sent from the transmitter has a frequency of 69 kHz transmitted using a piezoelectric transducer. The signal is modelled by sending a Differential Pulse Position Modulated (DPPM) signal which is a signal that has data encoded in the time between the pulses and accurate timing for the transmitter is essential to decoding the acoustic signal correctly.

The acoustic transmitter tag for this system shall be provided by Thelma Biotel AS [50] as a separate module that can be operated using a general purpose pin from the microcontroller, this pin must be capable of providing the accurate timing interval with the acoustic pulse or "pings" need to be transmitted. An accurate "heartbeat" or frequency is vital for the hydrophone (receiver) to recognize the acoustic signal correctly. The hydrophone and acoustic transmitter can both been seen in Figure A.1, having multiple hydrophones spread in a special pattern also allows for triangulation of the signal and locating the fish/tag.

This gives the motivation of developing an ultra-low power hybrid ATT/DST system come from using the new advancements in low-energy microcontrollers

[1] Triangulation is the process by which the location of a sound signal is deterministic Salmon Research Board

and sensors to create a system that takes as much space as possible while still providing increased flexibility to a well established tracking technique ATT. Adding multiple sensors with a data logger functionality can provide more information on the fish migration path, especially in periods where no hydrophones (receivers) are nearby to receive the signal from the acoustic transmitter tags.

Figure A.1: Thelma Biotel AS Hydrophone and Transmitter [50]

A.2 Accuracy and Resolution

The term uncertainty to an engineer comes from different sources such as noise, resolution, accuracy or precision of a device. It is often desired to minimize uncertainty by having as accurate measurements to the real value to be measured as possible.

The accuracy of the sensors is defined as the maximum difference between the actual value and the outputted value of the sensor. True values are measurements using situ[2] observations[6] that provide and can be expressed as a percentage of the full scale or in absolute terms.

The resolution is the smallest detectable incremental change of input parameter (pressure, magnetic field strength, acceleration, temperature) that can be detected in the output signal. Resolution is usually expressed as a proportion of the reading or in absolute terms.

A.3 Geolocalization Underwater

The Global Positioning System (GPS) allows for a precise determination of the geo-position as well to the identification of accurate time. There are also several global positioning systems (Galileo, GLONASS, Bediodu) which offer global positioning anywhere on earth. Although these geolocation systems can provide accurate positions, this method of geolocation is insufficient for marine animals considering the animals are submerged out of the research of satellites

[2] a Latin phrase that translates literally to "on-site" or "in position".

for most of them. Hence, it is necessary to use other geolocation methods such as sea surface temperature matching and geomagnetic field strength matching find an estimate of the location (longitude and latitude).

A.4 Group of High-Resolution Sea Surface Temperature

Sea Surface Temperature (SST) measurements are collected using satellites [6] that have been increasing in accuracy thanks to technology advancements and higher requirements such as the ones set by the group of high-resolution sea surface temperature (GHRSST) which sets specifications [51]for the accuracy to be less than 0.4 °C and even higher accuracy is planned for the ultra-high-resolution SST for 2022 [52] specified in GHRST plan [53].

A.5 I^2C Bus

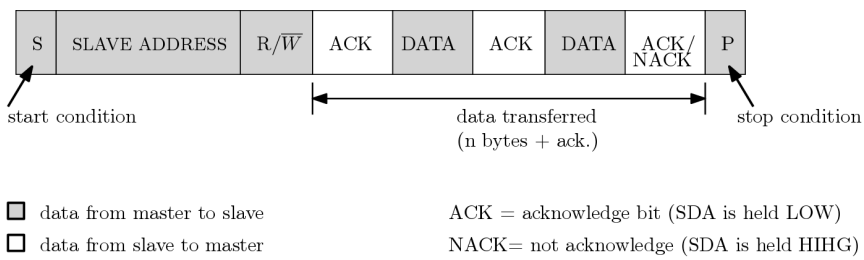


Figure A.2: I^2C Protocol Data Frame

All I2C transfers begin with a START condition and end with a STOP condition. A START condition is defined as a high-to-low transition on the SDA line while SCL is high, and a STOP condition is defined as a low-to-high transition on the SDA line while the SCL is high. When the START condition occurs on I^2C bus, the bus is considered busy and cannot be used by another master until a STOP condition is detected.

The START condition is always followed by the address and then by a data direction bit (R/W bit). If the R/W bit is 0, the master will write to the slave device. If the R/W bit is 1, the master will read from the slave device. After the R/W bit is sent, the master releases the bus and allows the slave to acknowledge (ACK) the request. The slave device that was addressed acknowledges to the master by holding SDA low for one clock cycle. Then the master or slave transmits data on the SDA line, depending on the R/W bit value. The SDA line can change only when SCL is low, and SDA must be stable when SCL is high. Data on the I2C bus is transferred in 8-bit packets (bytes). There is no limitation on the number of

bytes, but each byte must be followed by an ACK bit. If the slave device does not acknowledge the transfer, this means that there is no more data or the device is not ready for transfer. In this case the master device must generate either a STOP or a repeated START condition.

A.6 RoHS Compliance

This section explains why it is essential that all chosen components, including soldering paste, flux, and other required manufacturing materials, are RoHS compliant.

All of the products sold in Europe should have a CE-mark. This informs the purchaser that the manufacturer has complied with the European requirements maintained by the RoHS Directive 2011/65/EU. One of the 6 RoHS substances is lead (Pb), and it is often used as an additive material to solder. Lead is a toxic material proven to be [54] very harmful to life in water. There are also laws in place that prohibit the use of lead soldering in Norway. This is undoubtedly the case at NTNU, and any use should be recorded for 60 years. The RoHS directive [55] specifies the maximum level of lead (Pb) less than 1,000 ppm (0.1%)-to be compliant with the regulation (RoHS 3) that takes effect on July 22, 2019 [56].

B
Calculations

B.1 I^2C Pullup Resistance Calculation

The I^2C bus is an open-drain bus which means it needs pull-up resistors. The value of these resistors is important for proper operation and low power consumption. The bus can operate at two speeds 100 kbps or 400 kbps. The calculations for fast mode (400 kbps) is shown below using equation B.1 and B.2. Information needed for these calculations are shown in Table B.1.

Table B.1: Bus Capacitance and *I2C* Parameters

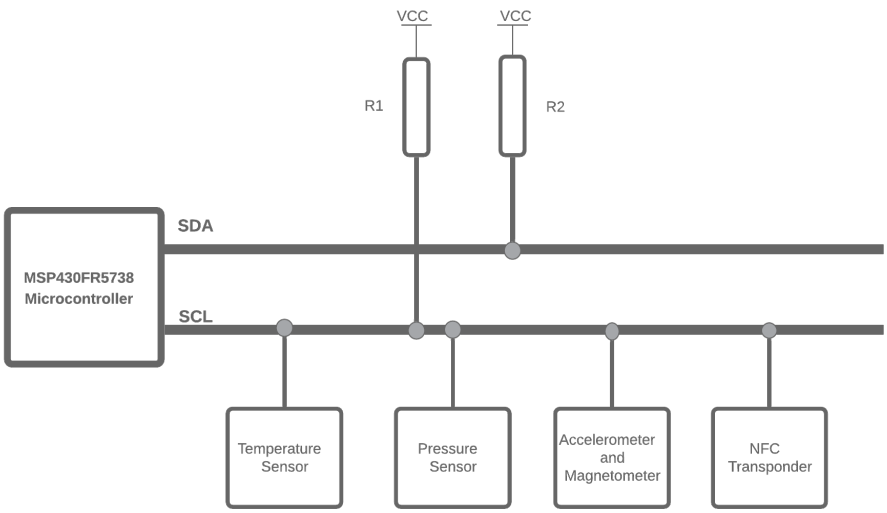
Bus Capacitance Information	
Device	Input Capacitance
TMP117	4 pF
MSP430FR5738	5 pF
KMX62	20 pF (max)
RF430CL331H	5 pF
MS5837-30BA	20 pF
PCB Trace Capacitance	7 pF
Total	45 pF

Parameter	Standard mode	Fast mode	Unit
t_r	1000	300	ns
V_{OL}	0.4	0.4	V

$$R_P(max) = \frac{t_r}{(0.8463 \times C_b)} = \frac{\left(300 \cdot 10^{-9}\right)}{\left(0.8463 \times 45 \cdot 10^{-12}\right)} = 7.8681\,\text{k}\Omega \tag{B.1}$$

$$R_p(min) = \frac{(V_{CC} - V_{OL}(max))}{I_{OL}} = \frac{(2 - 0.4)}{4 \cdot 10^{-3}} = 400\,\Omega \tag{B.2}$$

Therefore, a resistor value between 400 Ω and 7.9 kΩ. The value of the pull-up resistors can be selected based on the trade-off for power consumption and speed [57]. A small resistance value will give a higher speed due to the smaller RC delay, while a high resistance will give less power consumption. The design prioritizes low power consumption. Therefore the resistance value should be as close to the maximum as possible.



I^2C Bus Topology

Figure B.1: I^2C Bus Topology

B.2 Crystal Load Capacitors Calculation

This section provides the calculations required to determine the load capacitors C_{L1} and C_{L2} that provides the load capacitance of $C_L = 12.5$ pF. The capacitances can be determined by solving equation B.3.

$$C_{Load} = \frac{C_{L1} \times C_{L2}}{C_{L1} + C_{L2}} + C_{stray} \tag{B.3}$$

Setting $C_{L1} = C_{L2}$ and C_{stray} is estimated to approximately 2 pF [26]. we can simplify the equation to:

$$\begin{aligned} CL_1 = CL_2 &= 2 \cdot C_{Load} - C_{stray} \\ &= 2 \cdot 12.5 - 2 \rightarrow 23\text{pF} \end{aligned} \tag{B.4}$$

B.3 NFC Antenna Calculation

This section provides the calculations needed for tuning the external Molex antenna [58]. This antenna has an inductance of L_{ant} =2.11 µH which must be tuned to resonate at 13.56 MHz by adding a capacitor to create an LC circuit. The resonance capacitance $C_{Res} = C_{int} + C_{tune}$ can be calculated using Equation B.5, that results from Equation B.6 show the total required capacitance $C_{Res} = 65$ pF which is used to find the tuning capacitance by subtracting the internal RF430CL331 capacitance $C_{int} = 30$ pF, making the final required tuning capacitance $C_{Tune} = 65pF - 30pF = 35pF$.

$$f = \frac{1}{2\pi \cdot \sqrt{L_{coil} \cdot C_{Res}}} \rightarrow find\,C \tag{B.5}$$

$$C_{Res} = \frac{1}{(2\pi \cdot 13.56MHz)^2 \cdot L_{ant}} \approx 65pF \tag{B.6}$$

This page has been left intentionally blank.

C

Design and Fabrication Information

This appendix contains extra design information for chapter 4. The schematics, footprint documentation as well to fabrication and assembly information can be found in this appendix. More information on downloading the design files is found in Appendix D.

C.1 Schematics

Table C.1: PCB Designators

Designator	Explanation
ANT	Antenna
BAT	Battery
C	Capacitor
D	Diode
R	Resistor
S	Switch
U	Chip
Y	Crystal

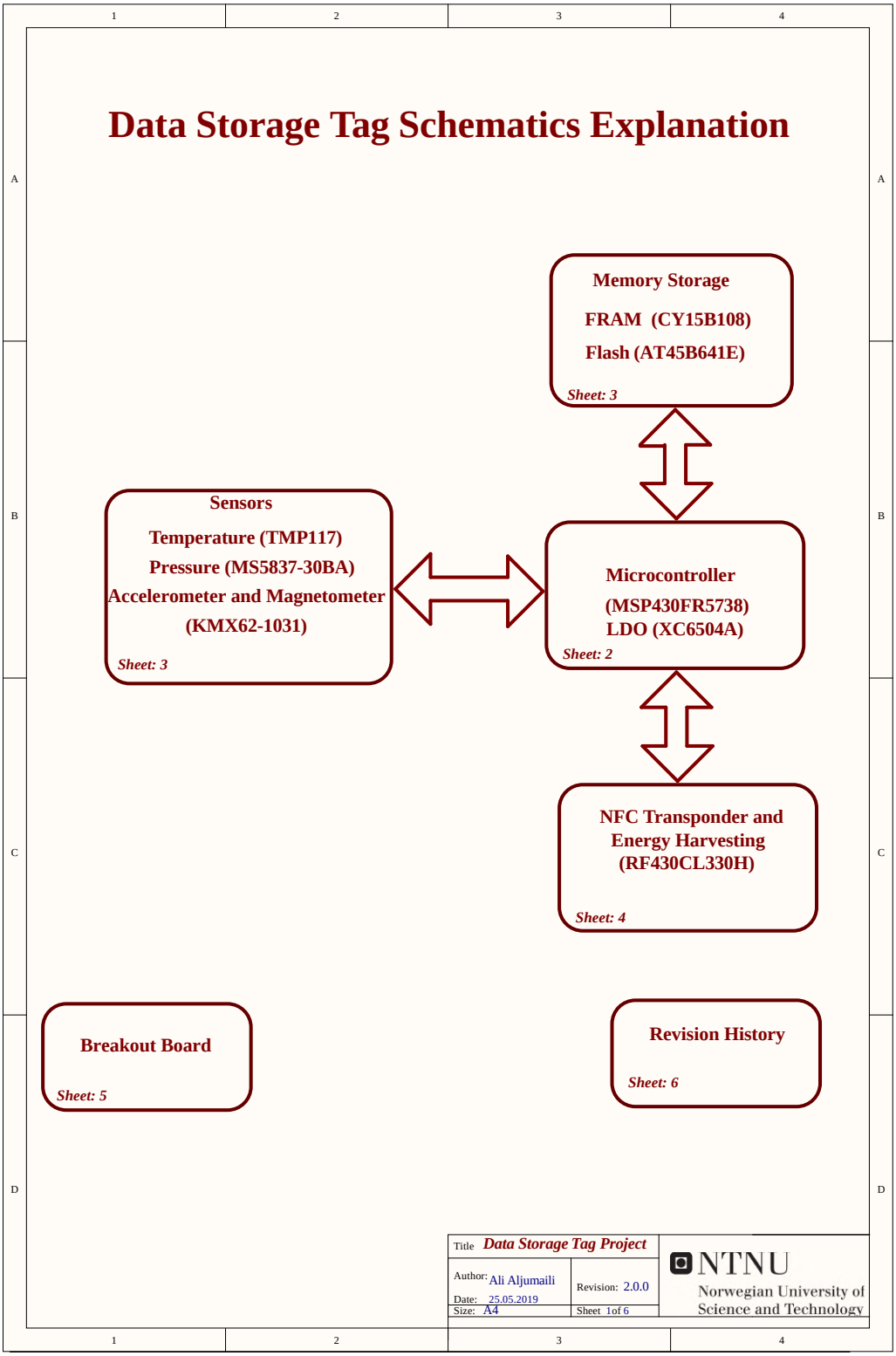


Figure C.1: Schematics: Front page 1/6

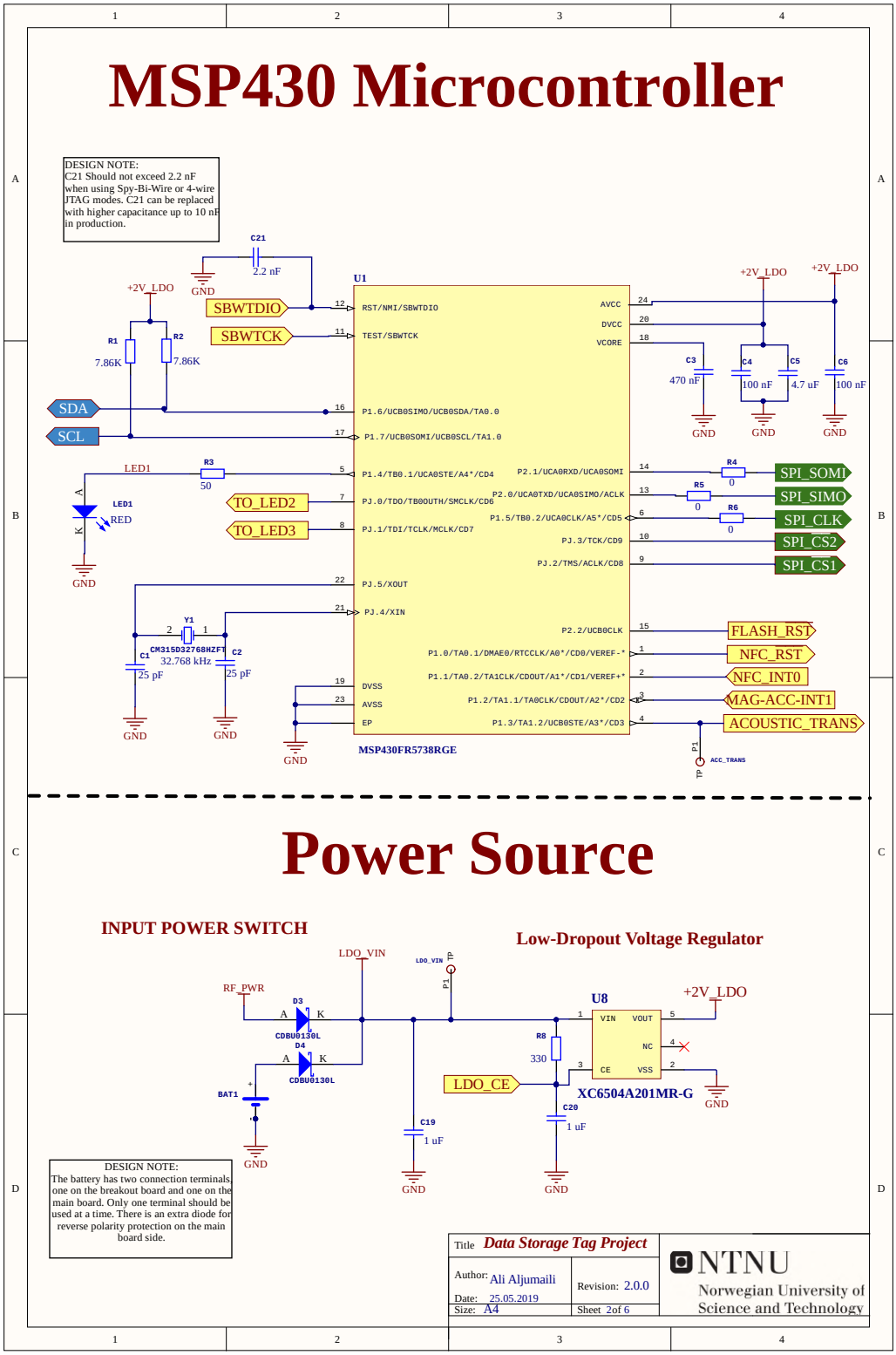


Figure C.2: Schematics: MSP430 MCU page 2/6

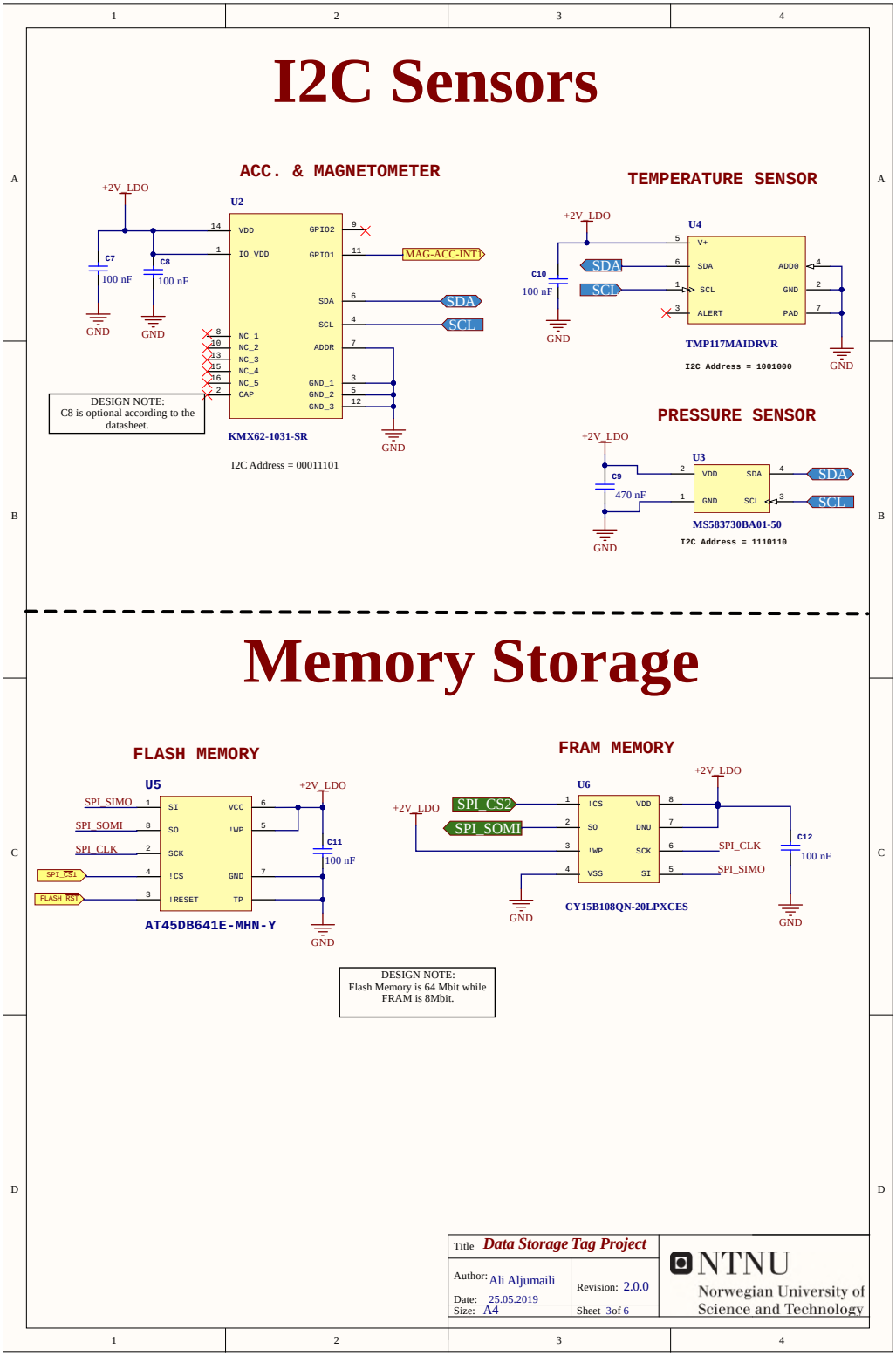


Figure C.3: Schematics: Sensors and Memories page 3/6

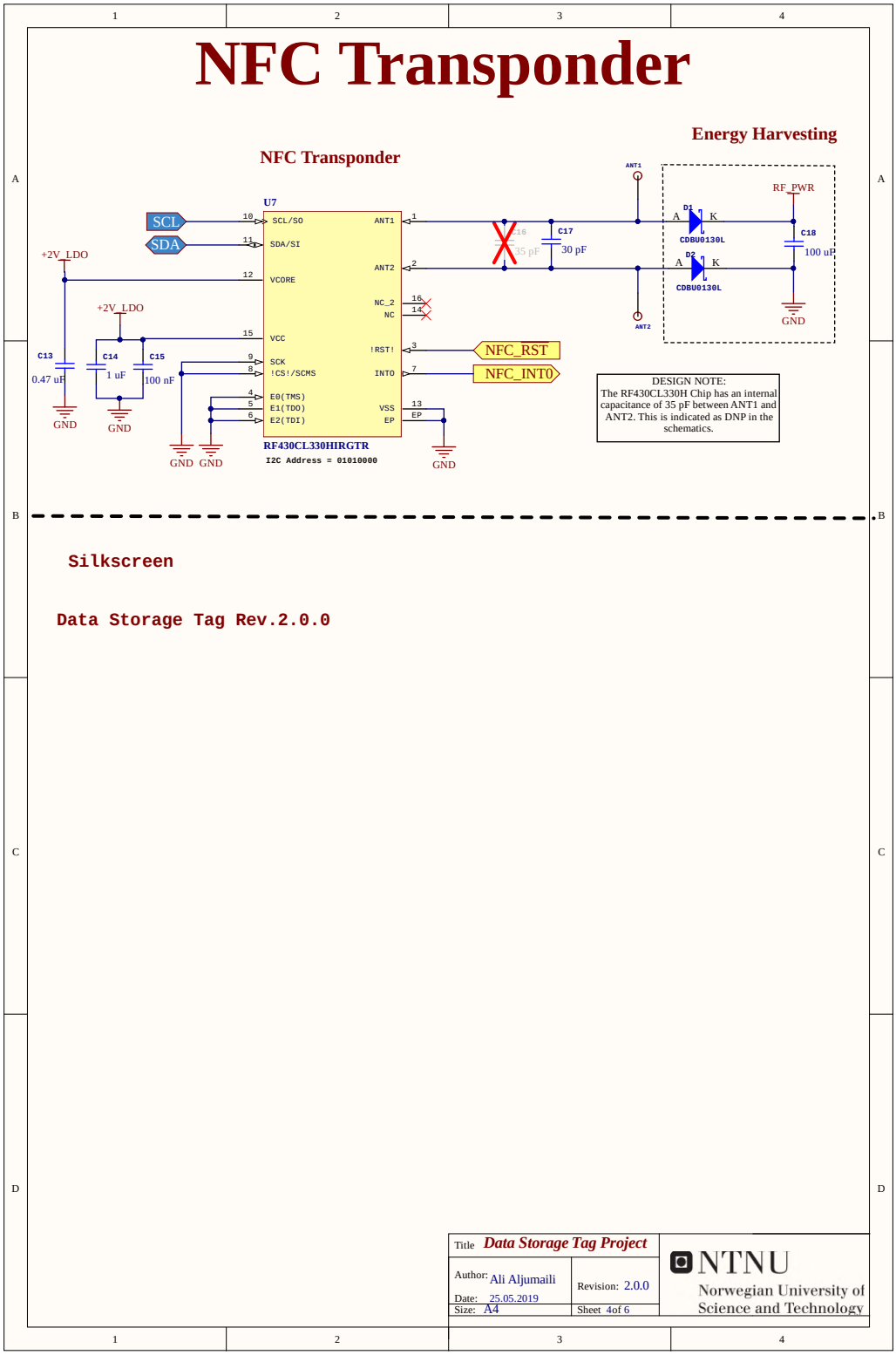


Figure C.4: Schematics: NFC and LDO page 4/6

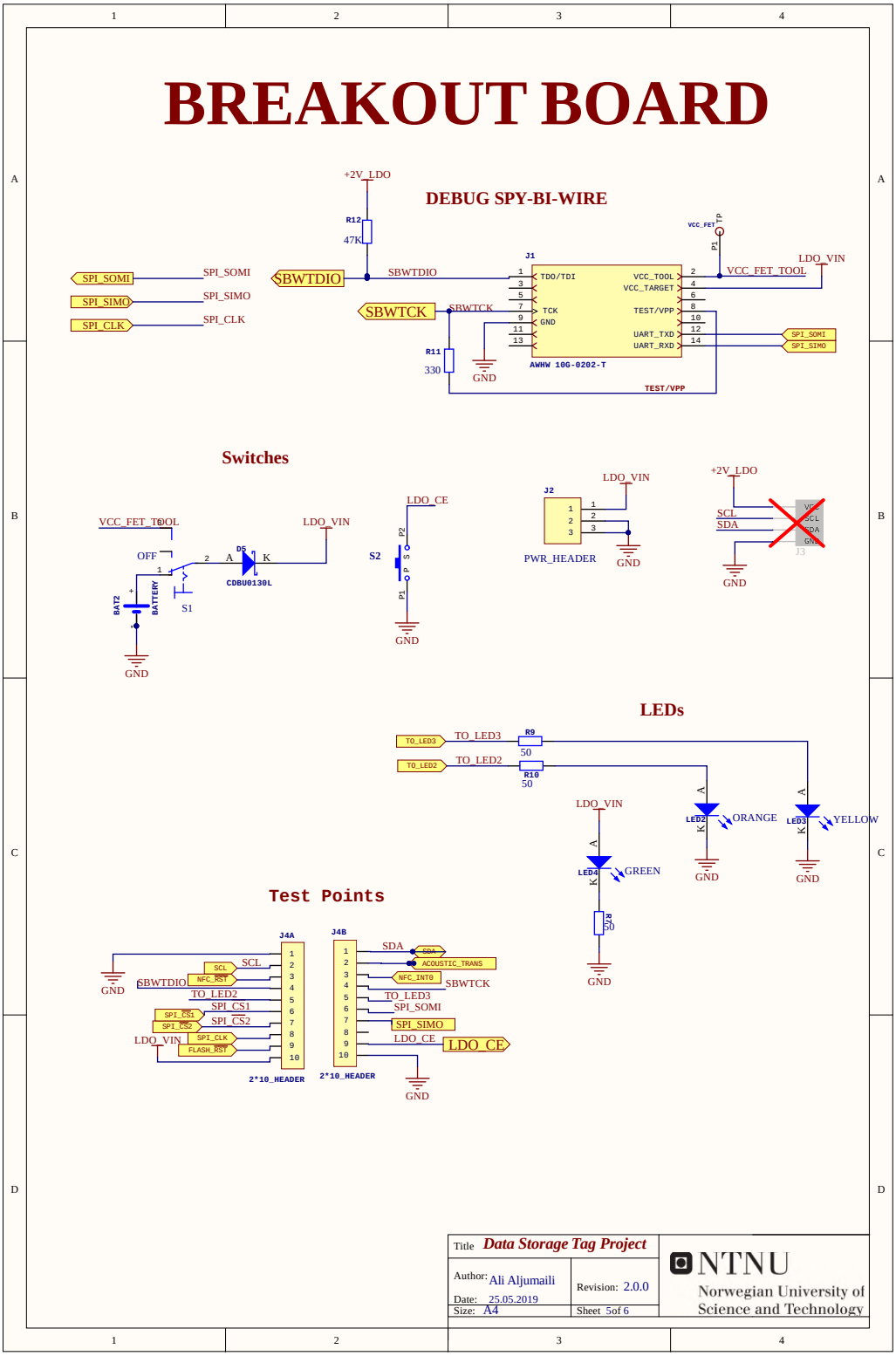


Figure C.5: Schematics: Breakout Board page 5/6

Revision History

Date	Version	Revision Summary	Author
December 2017	1.0.0	- First release	Ali Aljumaili
February 2019	1.1.0	- Extend SMD pads - Change MSP430F2272 to MSPFR2633. - Change debugging interface from JTAG to SBW. - Removed RTC. - Add 32 KHz crystal - Add energy harvesting circuit. - Add milling to insulate temp and mag-accel sensors. - Board size reduced to 40 mm x 13 mm	Ali Aljumaili
March 2019	1.2.0	- Change temperature sensor from SI7051 to TMP117. - Change LDO from 2.5V to 2.0V variant. - Change MCU from MSP430FR2633 to MSP430FR2633 to MSP430FR5738. - Add front page and revision history.	Ali Aljumaili
March 2019	1.2.1	- Add breakout board for pressure sensor. - Add breakout board for test points. - Add reset switch. - Add power switch. - Add FRAM chip. - Update silk layer	Ali Aljumaili
March 2019	2.0.0	- Move project to Altium Designer. - Update 3D models - Update the silk layer. - Add documentation for all footprints. - Add a draftsman document. - Export 3D PDF, SmartPDF Schematics and layer stackup. - Create Assembly drawings. - Create a product variation.	Ali Aljumaili

Title *Data Storage Tag Project*		NTNU Norwegian University of Science and Technology
Author: Ali Aljumaili Date: 25.05.2019	Revision: 2.0.0	
Size: A4	Sheet 6of 6	

Figure C.6: Schematics: Revision History page 6/6

C.1.1 Fabrication information

The PCB was ordered from OSH Park and Stencil from OSH Stencil.

Table C.2: PCB Design Information

Description	Value
Board Size	40 mm x 14 mm
Number of Layers:	4
Min. Track Width	8 Mil
Min. Clearance	8 Mil
Min. Via Pad Size	24 Mil
Min. Angular ring	0.05 mm (2mil)
Thermal Isolation	10 mil
Material	FR-4 High
Thickness	62 MIL (1.6 mm)
Tolerance	ANSI IPC-6012 Type 3 Class 2
Bow & Twist	ANSI IPC-6012 Type 3 Class 2
Cooper Thickness	1.4 MIL (1oz)
Drilling	Files attached
Surface Finish	
Surface Finish	Immersion Gold (ENIG)
Silk Screen:	Top and Bottom
Silkscreen color	White
Solder Resist Color	Green

C.2 Layer Stackup

The new layer stackup of version four consists of 4 layers where the top and bottom layers are used for signal routing. Meanwhile, the two middle layers act as ground and power planes, as seen in Figure C.7. The upgrade to a four-layer PCB provides more noise immunity [59] due to extra capacitance filtering from the two inner layers.

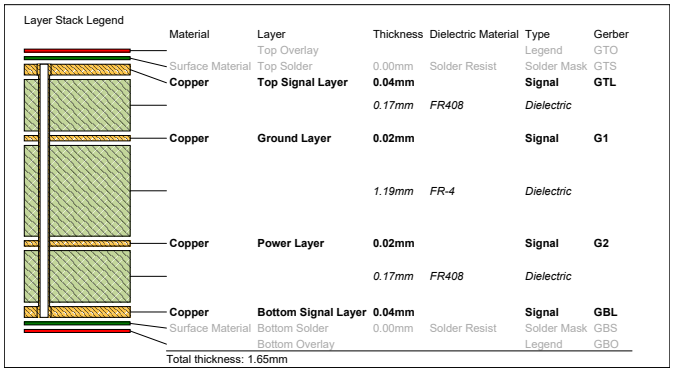


Figure C.7: PCB: Layers Stackup and Materials

C.3 Solder Paste Reflow Profile

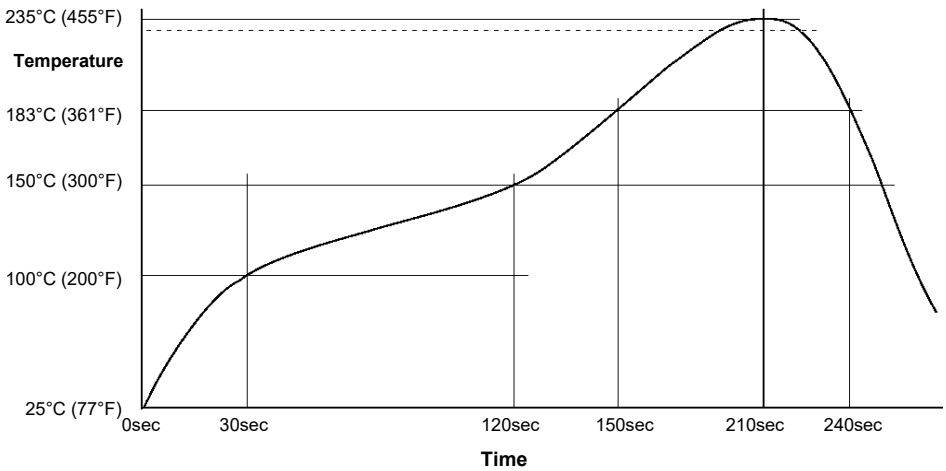


Figure C.8: Solder Paste Reflow Profile

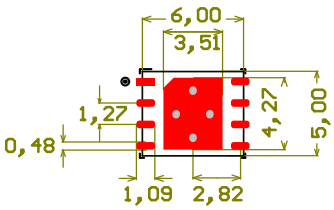
Note: The Solder Paste used during assembly contained a high percentage of lead, which makes it not RoHS compliant, as discussed in section A.6. Next revision should use a RoHS compliant solder paste, which often means higher peak reflow temperature.

C.4 Component Cost Summary

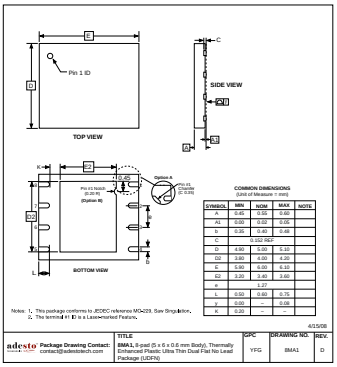
Table C.3: Component Cost Summary

Component	Cost (NOK)
MSP430FR5738	25 kr
CM315D32768HZFT	27 kr
KMX62-1031	129 kr
MS5837-30BA	204
TMP117	45 kr
AT45DB641E (Flash)	43 kr
CY15B108QN (FRAM)	187 kr
RF430CL331H	14 kr
XC6504A201MR-G	5 kr
Battery	55 kr
Antenna	7 kr
Passive Components	70 kr
PCB	85 kr
Total	**896 kr**

C.5 Footprint Documentation

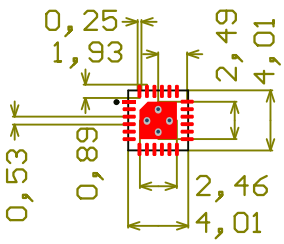


(a) Flash Board Footprint

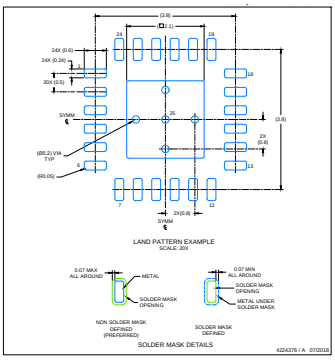


(b) Datasheet [31]

Figure C.9: PCB: Flash Memory Footprint Documentation

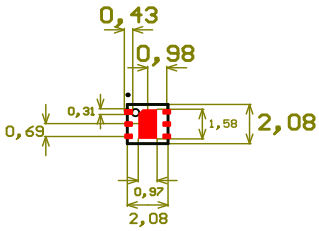


(a) MCU Package Footprint

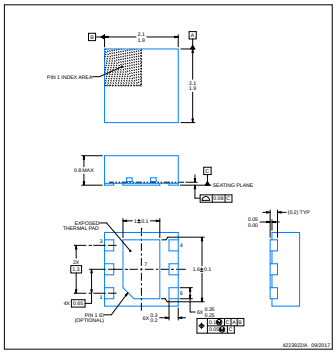


(b) MSP430FR5738 Datasheet [26]

Figure C.10: PCB: MSP430FR5378 Footprint Documentation

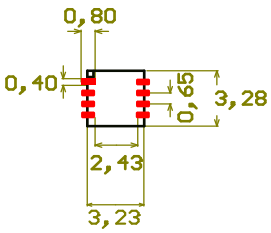


(a) TMP117 Footprint

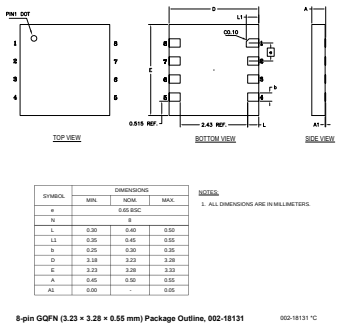


(b) TMP117 datasheet [22]

Figure C.11: PCB: TMP117 Footprint Documentation

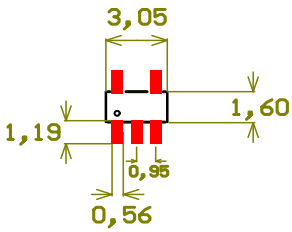


(a) Top Layer

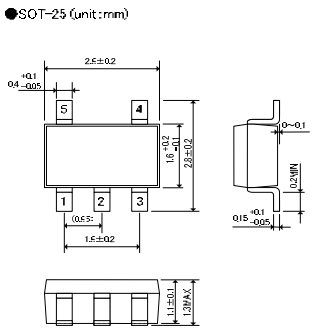


(b) FRAM Datasheet [30]

Figure C.12: PCB: F-RAM Footprint Documentation



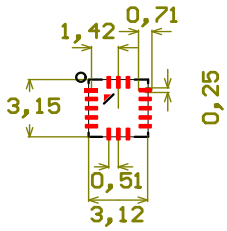
(a) LDO Footprint



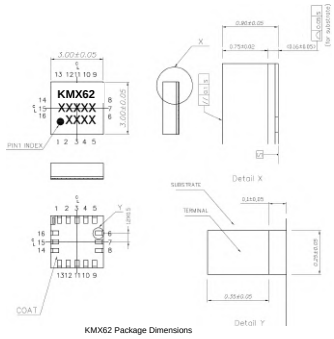
(b) Datasheet [32]

Figure C.13: PCB: LDO Footprint Documentation

C.6 Development Process Images

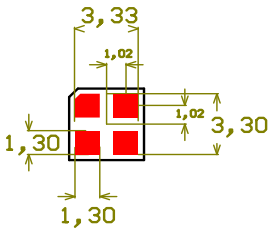


(a) KMX62 Footprint

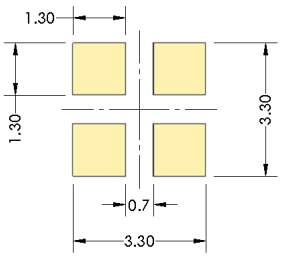


(b) KMX62 Datasheet [23]

Figure C.14: PCB: KMX62 Footprint Documentation

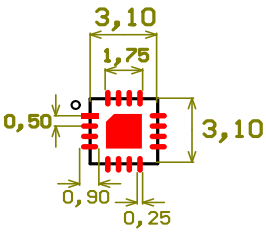


(a) Pressure Sensor Footprint

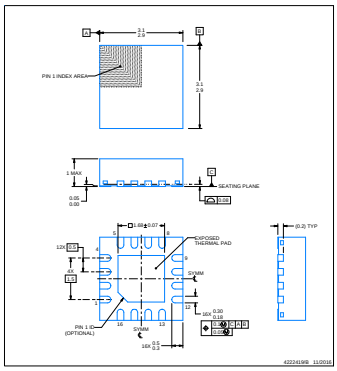


(b) Datasheet [20]

Figure C.15: PCB:Pressure Sensor Footprint Documentation



(a) NFC Footprint



(b) Datasheet [24]

Figure C.16: PCB: NFC Transponder Footprint Documentation

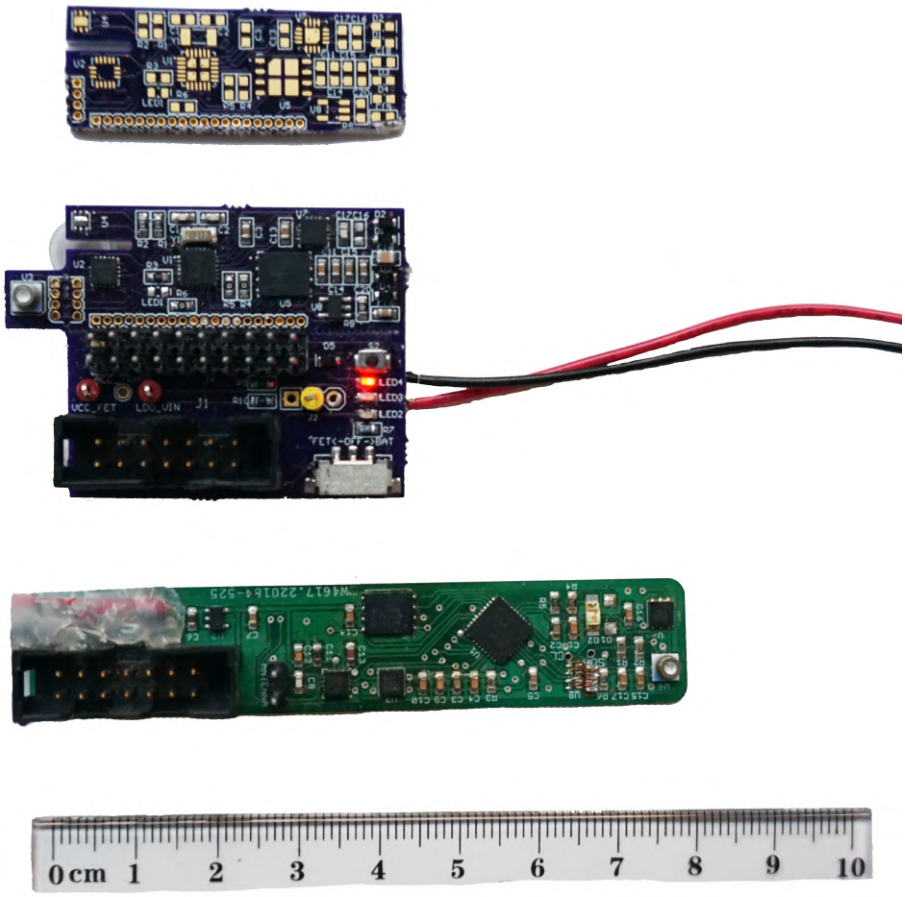


Figure C.17: Assembled Prototype V2 and V1 Size Comparison

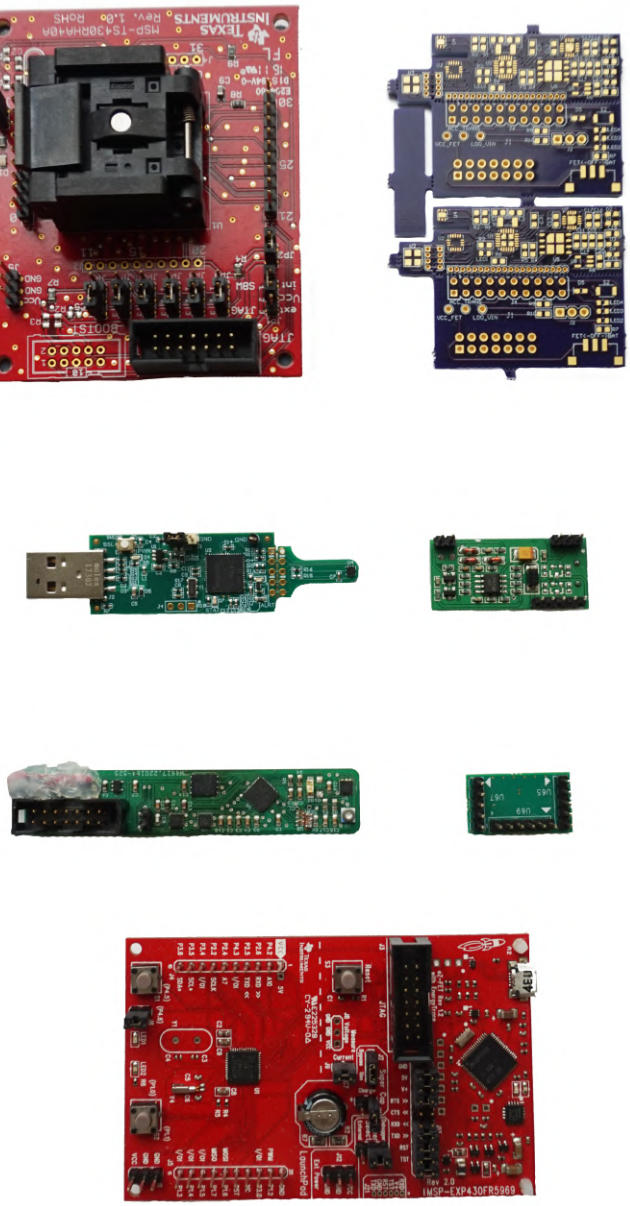


Figure C.18: Development Boards Used in Testing

Figure C.19: Main Prototype Test Tools

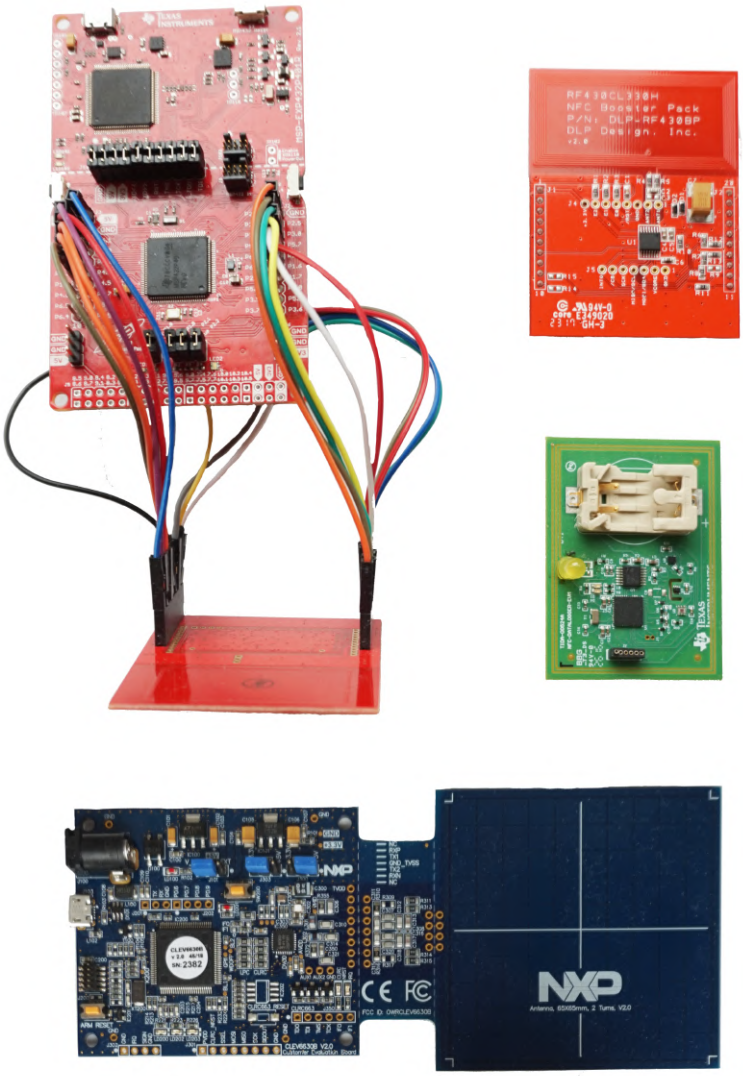


Figure C.20: NFC Development Boards and Readers

D
Getting Started Guide

This guide is created to make it easier to find all of the files and easily get started with the project. There is a number of code examples, research budgets and CAD files which are made available by a custom documentation website that is accessed on:

`https://dst-ntnu.bitbucket.io/`

Figure D.1 is tagged with a series of numbers that are explained below:

1. Link to the bitbucket git repository, the content is briefly introduced in section D.1.
2. Clicking on the history symbol transfers to a commit history with recent changes.
3. This link download a PDF of the current document or thesis publication.
4. Downloads a PDF version of the website user's guide.
5. Downloads a 3D PDF Model of the prototype, this file needs to be opened with Adobe Acrobat or another PDF reader that supports 3D rendering.
6. Downloads a smart PDF for the design schematics.
7. Link to gathered code examples from reference designs that are highly relevant to the design.
8. Link to excel research sheets for battery life estimate and memory budget.
9. Getting started guide for getting started with the repository.
10. Guides for installing and customizing the development tools.

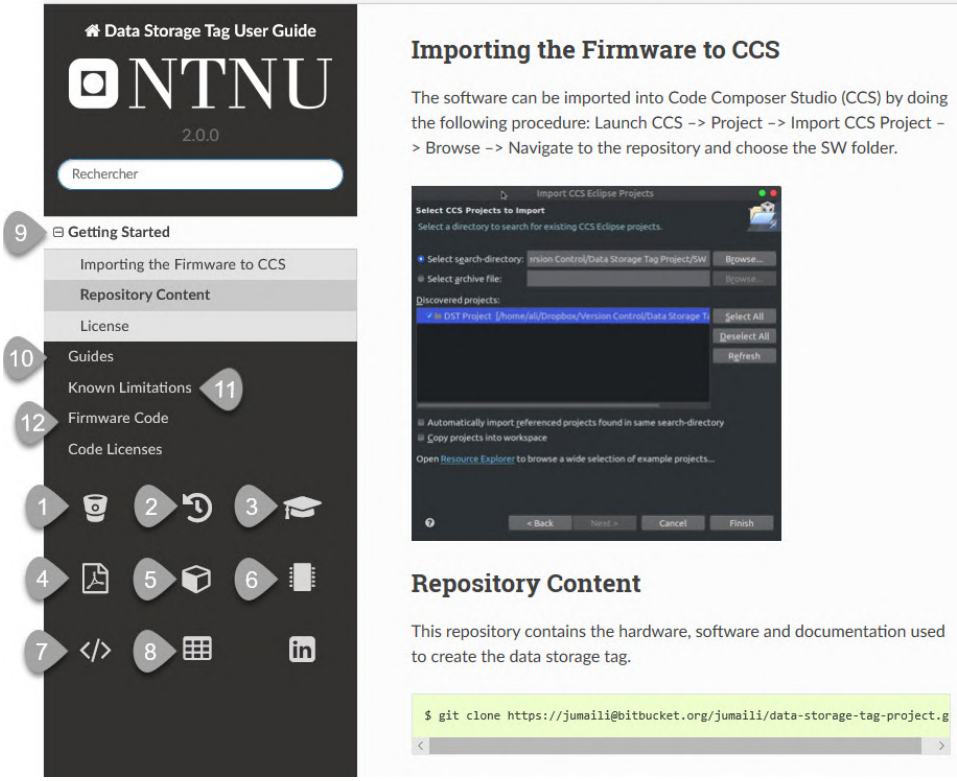


Figure D.1: Documentation Website dst-ntnu.bitbucket.io

11. List of limitations and their solution.
12. A copy paste friendly version of the firmware code.

D.1 Overview of Repository Content

- Gerbers
- Board and Schematics
- Bill of Materials
- 3D Model of PCB

} Hardware Folder

- Code for Mainboard (MSP430FR5738)
- Code for Devkit (MSP430FR5739)

} Software Folder

- Code Examples
- Datasheets
- Settings for CCS and WaveForms
- Research Budgets

} Extra Folder

- Sphinx Documentation for dst-ntnu.bitbucket.io

} Doc Folder

This page has been left intentionally blank.

NTNU

Norwegian University of Science and Technology

Data Storage Tag User Guide

Release 2.0.0

Ali Aljumaili

Jun 24, 2019

CONTENTS

Recent Changes

- **Appendices Edit, doc refactor** by *Ali Aljumaili* at *2019-06-21 04:45:59*
- **Edit Website Layout, open links in new tab, add NFC keyboard code** by *Ali Aljumaili* at *2019-06-18 20:15:48*
- **Add 3Dpdf and software examples used** by *Ali Aljumaili* at *2019-06-18 18:40:27*
- **Fix build issue, refactor structure to core, devices and app. layer** by *Ali Aljumaili* at *2019-06-10 17:39:22*

Keyword Index, Search Page

CHAPTER

ONE

GETTING STARTED

This guide is written for documentation of the data storage tag project firmware and hardware.

Contents

1.1 Importing the Firmware to CCS

The software can be imported into Code Composer Studio (CCS) by doing the following procedure: Launch CCS –> Project –> Import CCS Project –> Browse –> Navigate to the repository and choose the SW folder.

1.2 Repository Content

This repository contains the hardware, software and documentation used to create the data storage tag.

This documentation for this repository can be found under the the `doc` folder in the root folder.

Cloning this repository can be done by done with git:

```
$ git clone https://jumaili@bitbucket.org/jumaili/data-storage-tag-project.git
```

Folder Structure of Project

```
│ data storage tag project
│
├── Extra
│    ├── Custom Settings
│    │    ├── CCS-Darktheme.xml
│    │    ├── TexStudio_DarkTheme.txsprofile
│    │    └── waveforms-i2c-protocol.dwf3work
│
│    ├── Datasheets and Technical Reports
│    │    ├── Devices
│    │    │    ├── CM315D_E-annotated.pdf
│    │    │    ├── RF430CL330H-NFC -annotated.pdf
```

(continues on next page)

(continued from previous page)

```
│   │   │   .
│   │   │   .
│   │   │   .
│   │   │   ├── TMP117-annotated.pdf
│   │   │   └── XC6201_LDO.pdf
│   │   └── NFC
│   │       ├── DLP_DESIGN_RFID2_MODULE_Schematic.pdf
│   │       └── dlp-rf430bp.pdf
│   ├── Research Budgets
│   │   ├── Battery-Life-Worst-Case-Estimate.xlsx
│   │   ├── Memory Budget.pdf
│   │   └── Memory Budget.xlsx
│   ├── TI-Code-Examples
│   │   ├── MSP430FR57xx_Code_Examples
│   │   │   └── ...
│   │   ├── NFC Example Software/nfc-keyboard
│   │   │   └── ...
│   │   ├── Over The Air Upgrade using The Bootloader Examples/MSP430FRBoot_1_02_00_
↪00
│   │   │   └── ...
│   │   ├── Software Snapshot
│   │   │   └── ...
│   │   └── data-logger-software
│   │       └── ...
│   └── Elprolab Dokumentasjon.zip
├── HW
│   ├── CAD
│   │   ├── 3D-Step-Models-Files
│   │   │   ├── 0805-NO.step
│   │   │   ├── 1X03.step
│   │   │   .
│   │   │   .
│   │   │   .
│   │   └── Rev.2.0.0 - Altium Designer
│   │       ├── A4-NTNU.SchDoc
│   │       ├── DST-Rev.2.0.0.OutJob
│   │       .
│   │       .
│   │       .
│   │       └── Sheet5-DST-Rev.2.0.0.SchDoc
│   └── Production-PCB-V2
│       ├── Assembly.PDF
│       ├── DST-BOM-V2.xlsx
│       ├── Gerber V2.0.0.zip
│       ├── PCB-3D.pdf
│       └── PCB-Layers.pdf
```

(continues on next page)

(continued from previous page)

```
│
├── SW
│    ├── DST_DevKit_FR5739
│    │    ├── application_layer
│    │    │    └── ...
│    │    ├── driverlib/MSP430FR57xx
│    │    │    └── ...
│    │    .
│    │    .
│    │    .
│    │    └── main.c
│    │
│    └── DST_MainBoard_FR5738v
│         ├── application_layer
│         │    └── ...
│         ├── core
│         │    └── ...
│         .
│         .
│         .
│         └── main.c
│
├── docs
│    .
│    .
│    .
│    └── requirements.txt
│
├── .gitinorge
├── .readthedocs.yml
├── LICENSE
├── README.rst
```

In the folder structure above:

- `HW` is the folder with all of the design files (schematics, board layout, BOM, Gerber files, assembly drawings) for Altium Designer and Eagle.
- `doc` is the directory where the Sphinx documentation reside these are used to generate the Data Storage Tag User Guide Webs[1].
- `SW` is the directory with the firmware software for the main prototype (MSP430FR5738 - 28 pin MCU) as well to software for the development kit (MSP430FR5739 - 40 pin MCU).
- `extra` is a directory that contains extra content like CCS configuration settings, device datasheets and code examples for the MSP430FR57xx MCU.

Note:

`data-storage-tag-project` is referred to as the *Repository root*, while the folder `data-storage-tag-project/docs` will be referred to *Sphinx root* or equivalently *Documentation root* folder.

[1] https://dst-ntnu.bitbucket.io

1.3 License

[2] https://opensource.org/licenses/MIT

CHAPTER

TWO

GUIDES

Contents

2.1 Code Composer Studio (CCS) Setup

2.1.1 Installing Code Composer Studio

1. update system sudo apt update
2. install dependencies: sudo apt install libc6:i386 libusb-0.1-4 libgconf-2-4 build-essential
3. download CCS from www.ti.com/tool/CCSTUDIO
4. extract using tar xvzf CCS9.x.x.xxxxx_linux.tar.gz
5. Install CCS by running ./ccs_setup_9.x.x.xxxxx.bin. Note: Select both MSP430 and MSP432 families to install all drivers needed

Note: If CCS was installed as user then run the driver install script cd <CCS_INSTALL_DIR>/ccsv9/install_scripts && sudo ./install_drivers.sh

2.1.2 Customising CCS

Code composer studio is an integerated developement envireonment (IDE) that natively supports the MSP430 line with powerfull debugging capabilities. CCS is based on the eclipse software framework which has a wide range of avalaible packages. A dark theme with a custome color scheme is used. Dark theme have many benfits like improved readability of text, better contrast and reduced eye fatigue. Research from S Brown[3] from the University of Cambridge discusses the many benfits and negatives of a detailed research on using dark backgrounds on computer displays.

[3] http://people.ds.cam.ac.uk/ssb22/css/dark.html

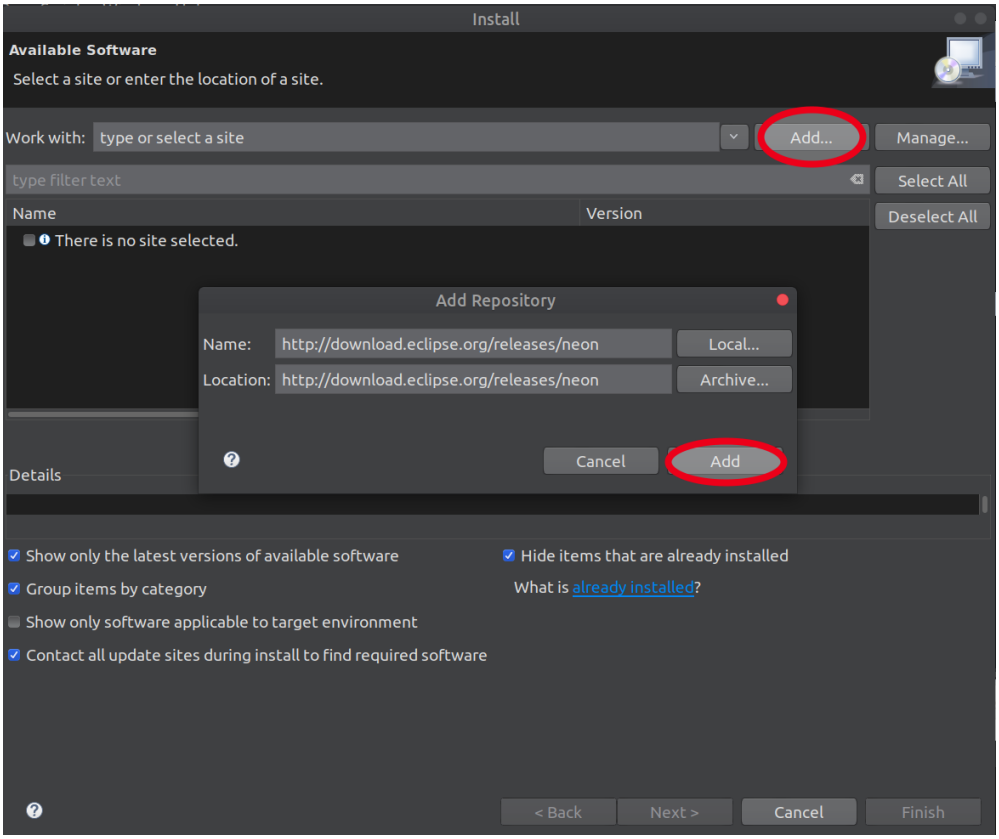
Eclipse has a popular theme called DevStyle theme, this theme can be seen on most of the screenshots and a special customizing for the colors can be found in the Extra folder in the project repository.

Installing this plugin can fail due to dependencies, it is therefor recommended to add

```
http://download.eclipse.org/releases/neon
```

to the

Go back to your IDE and click on "Install New Software..." in the "Help" menu.

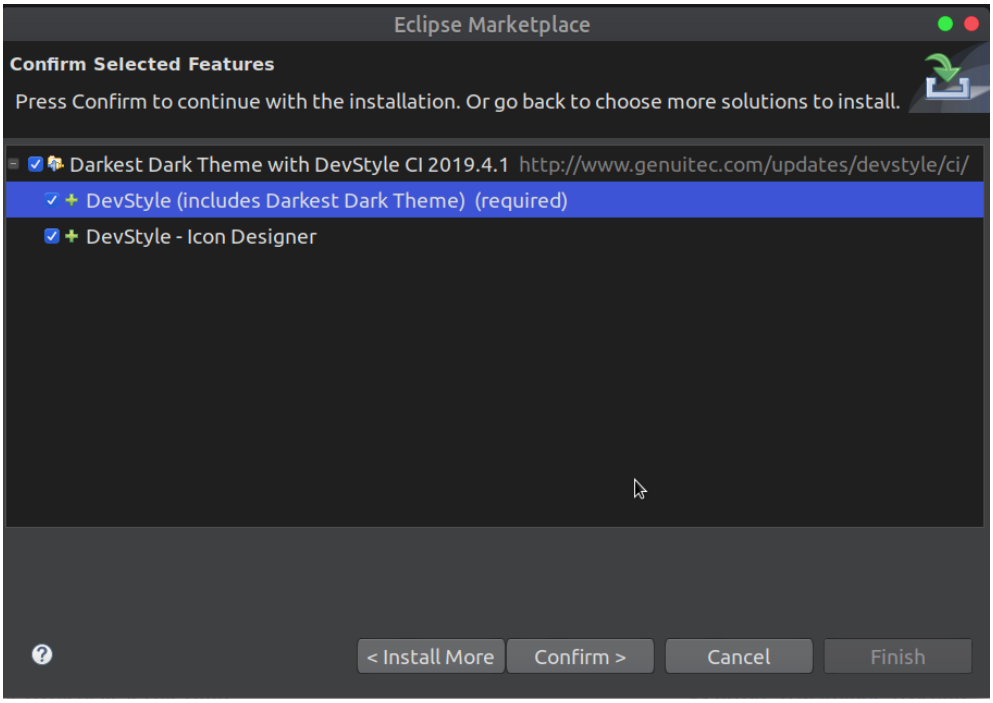


The theme can be found in the `extra/CCS Settings` folder and can be imported by going to Windows –> Preferences –> DevSTyle –> Color Themes –> Import and choosing the .xml theme in the `/extra/CCS Settings` directory.

2.2 Building the Documentation

This doucmentation is built using Sphinx. A python package that allows to build easy to maintain documentation in HTML, pub and latex/pdf format.

In order to use the Sphinx package, python needs to be installed which is preinstalled on many of the modern linux disutros.



The doc folder contains the makefile for both windows and linux. You can run read README.rst[4] for information on how to build these documentations.

- GNU/Linux Debian 7
- Git
- Sphinx 1.1.3 or higher
- Inkscape

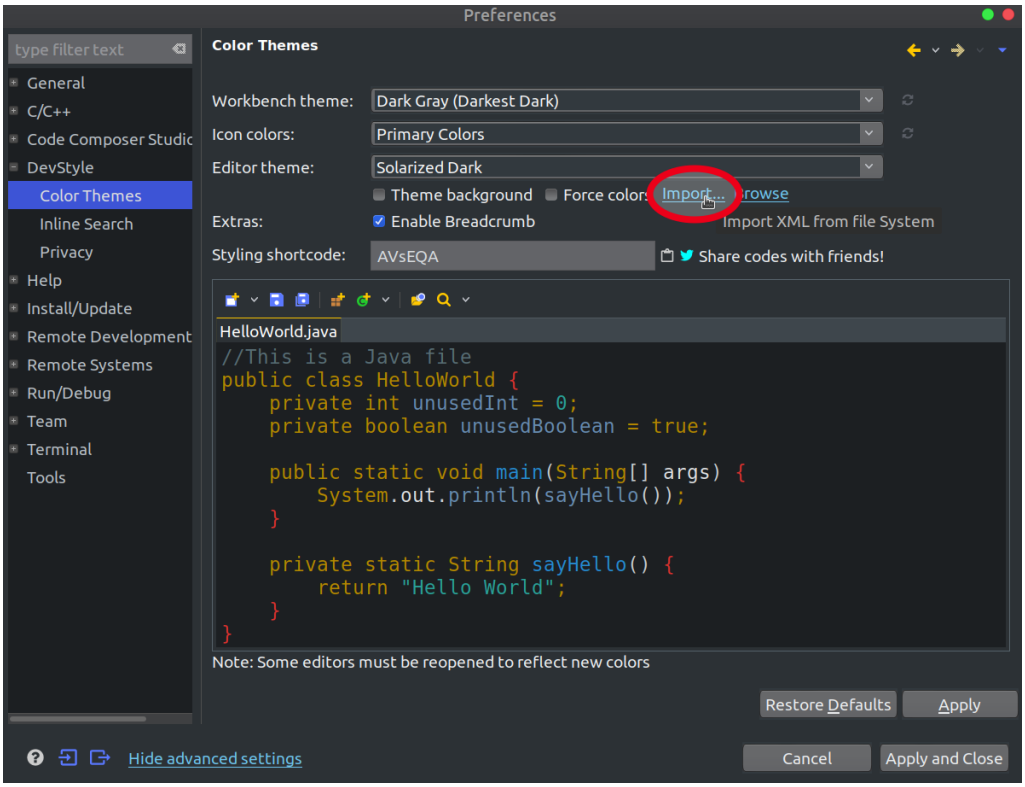
```
$ apt-get install git
$ apt-get install inkscape
$ apt-get install python-sphinx
$ apt-get install python3-sphinx-rtd-theme
$ pip install -r requirements.txt
```

If it not already present, this will install Python for you.

- texlive
- texlive-fonts-recommended
- texlive-latex-extra

```
$ git clone https://jumaili@bitbucket.org/jumaili/data-storage-tag-project.git
$ cd data-storage-tag-project/docs
$ make html
```

[4] https://bitbucket.org/jumaili/data-storage-tag-project/src/master/README.rst


Preferences
type filter text
General
C/C++
Code Composer Studio
DevStyle
Color Themes
Inline Search
Privacy
Help
Install/Update
Remote Development
Remote Systems
Run/Debug
Team
Terminal
Tools
Color Themes
Workbench theme:
Dark Gray (Darkest Dark)
Icon colors:
Primary Colors
Editor theme:
Solarized Dark
Theme background
Force colors
Import...
Browse
Extras:
Enable Breadcrumb
Import XML from file System
Styling shortcode:
AVsEQA
Share codes with friends!
HelloWorld.java
//This is a Java file
public class HelloWorld {
private int unusedInt = 0;
private boolean unusedBoolean = true;
public static void main(String[] args) {
System.out.println(sayHello());
}
private static String sayHello() {
return "Hello World";
}
}
Note: Some editors must be reopened to reflect new colors
Restore Defaults
Apply
Hide advanced settings
Cancel
Apply and Close

```
$ git clone https://jumaili@bitbucket.org/jumaili/data-storage-tag-project.git
$ cd data-storage-tag-project/docs
$ make latexpdf
```

2.3 Learning Resources

Embedded systems have a lot of new terminology it is highly recommended to read the TI glossary that explains many of the common terms, acronyms and definitions of embedded systems https://www.ti.com/lit/ml/slyz022j/slyz022j.pdf.

The table below shows a collection of useful resources that were either researched or use during the developement process.

1. **Simulators**
   - LTSpice: Spice Circuit simulator from Linear Technologies.
2. **CAD Software**
   - Eagle: One of the popular EDA packages due to it's (board size restricted) free version.
   - Altium circuit maker: free online PCB design tool powered by Altium technology. (Not recommended).
   - CircuitStudio: PCB deisgn tool by Altium Designer, cheaper in price with similiar capaabilitie, more functionalitythan Eagle.
   - Altium Designer: The best in terms of funcality, highly recommended. Steep learning curve.
3. **Gerber Viewers**
   - Tracespace (tracespace.io): Online Gerber viewer that lets you inspect the individual layers as well as the board preview.
   - Viewmate: Windows Gerber Viewer software http://www.pentalogix.com/viewmate.php(recommended).
4. **PCB Panelization**
   - Can be done using Altium Designer.
   - CAM 350: http://www.downstreamtech.com/CAM350XL.html
   - FAB 3000: http://www.numericalinnovations.com/fab3000.html
5. **Parts Search Engines**
   - findchips (https://www.findchips.com/)
   - parts.io component search engine designed towards discovering new parts.

CHAPTER

THREE

KNOWN LIMITATIONS

Contents

3.1 Problem 1 - Unable to connect to the target : Unknown device

This is a known problem with the Spy-Bi-Wire (2-wire JTAG) protocol on MSP-TS430RHA40A and MSP430FR5739 target board. The issue can occur when programming the newer version of the MSP-FET (V2.06) that has a CE sticker label on the case. The issue has been narrowed down to the RST/SBWTDIO signal. Specifically, the capacitance present on that bi-directional net which affects the rise and fall times during debug.

This issue is confirmed by TI and for the full discussion check this post thread:

MSP-TS430RHA40A: Problem using SBW with MSP-FET

- https://e2e.ti.com/support/microcontrollers/msp430/f/166/t/799408

3.2 Problem 2 - Outputting MCLK; SMCLK and ACKL to port pins not oscillating

Using the PxSELx register one can output the frequencies of the clocks and measure them using an oscilloscope. If there are other components placed there such as resistors, LEDS or other components, the load of the components would stop the oscillations of these components. A possible solution is to remove the components before testing or using drivelib get frequency functions to get a numerical value (based on the configured registers).

More info can be read this post: MSP430FR5738: MCLK, SMCLK and ACLK

- https://e2e.ti.com/support/microcontrollers/msp430/f/166/p/799846/2959558#2959558

3.3 Problem 3 - Measuring XIN/XOUT by probing crystal directly on pads

The probes add an additional capacitance depending on this capacitance this can stop the crystal from oscillation. For example the probes that come with the Analog Discovery 2 add 25 pF which is enough to stop the crystal from operating when both C1 and C2 are fitted.

3.4 Problem 4 - Building the project fails

A common problem that can occur is related to using non-alphanumeric characters in the directory name. This can cause build issues in CCS. For example having a username "C:UsersØyvind-SteinDST-Project" might case CCS to fail during project build. It is therefor to use due to the letter 'Ø'. It is therefor recommended to avoid such characters in project names, source/header files, CCS workspace folder names, System temp folder, etc. One exception is the underscore character '_' which is normally accepted.

Spaces in the path can also cause build errors in CCS. Creating a workspace in a new path which doesn't contain any spaces is highly recommended.

More information can be found here: Build errors in CCS

- http://software-dl.ti.com/ccs/esd/documents/sdto_ccs_build-errors.html#general

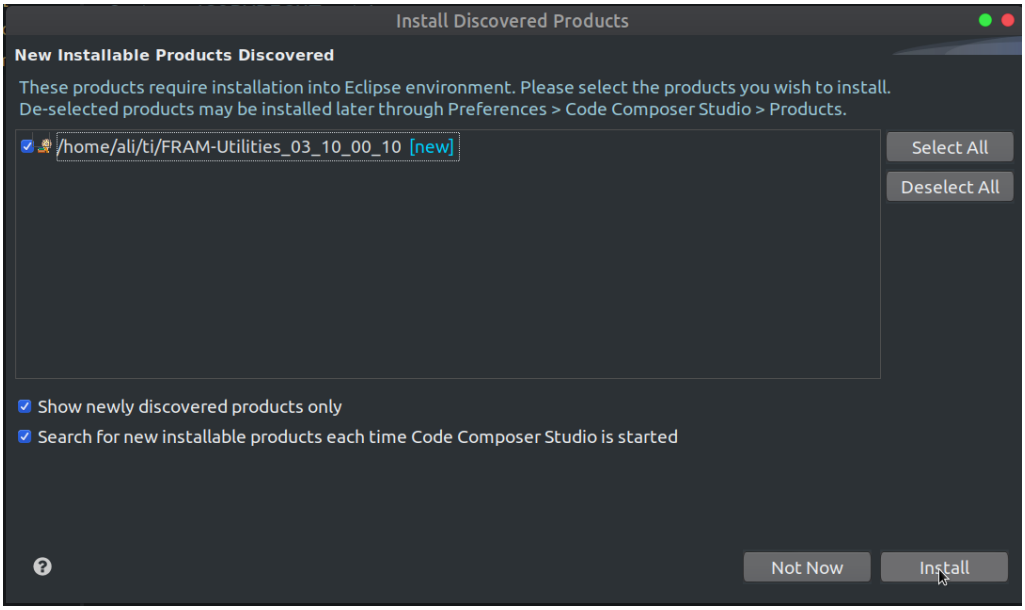
3.5 Problem 5 - Problems with building driverlib or FRAM-Utilities

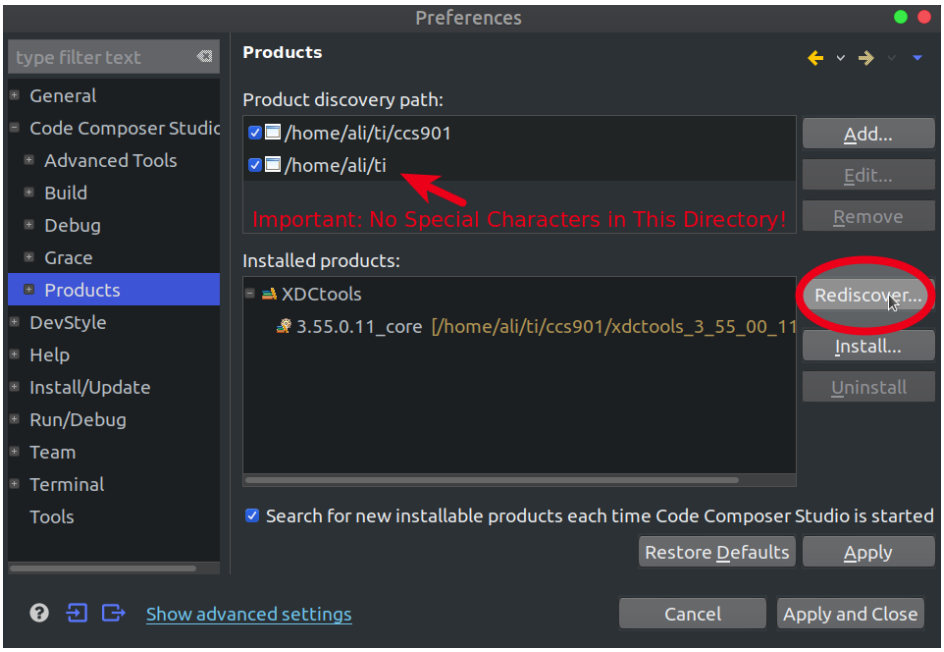
CCS can fail to detect MSP-FRAM-UTILITIES or driverlib. Although these files are now included in the project, problems can occur when changing code. It is therefor recommended to install these directly into CCS by going to: Window->Preferences->Code Composer Studio->Products, and click on Rediscover (note that the path for MSP-FRAM-UTILITES and driverlib should be in the discovery path)

3.6 Problem 6 - Handling System Reset Events Fails

The placement of the ISR for SYSRSTIV (System Reset Interrupt Vector Register) is important. For exact details on this issue read this e2e post. [FAQ] Handling MSP430 System Reset Events

- https://e2e.ti.com/support/microcontrollers/msp430/f/166/t/746272?tisearch=e2e-sitesearch&keymatch=faq:true


Install Discovered Products
New Installable Products Discovered
These products require installation into Eclipse environment. Please select the products you wish to install.
De-selected products may be installed later through Preferences > Code Composer Studio > Products.
/home/ali/ti/FRAM-Utilities_03_10_00_10 [new]
Select All
Deselect All
Show newly discovered products only
Search for new installable products each time Code Composer Studio is started
Not Now
Install


Preferences
type filter text
General
Code Composer Studio
Advanced Tools
Build
Debug
Grace
Products
DevStyle
Help
Install/Update
Run/Debug
Team
Terminal
Tools
Products
Product discovery path:
/home/ali/ti/ccs901
/home/ali/ti
Important: No Special Characters in This Directory!
Add...
Edit...
Remove
Installed products:
XDCtools
3.55.0.11_core [/home/ali/ti/ccs901/xdctools_3_55_00_11
Rediscover...
Install...
Uninstall
Search for new installable products each time Code Composer Studio is started
Restore Defaults
Apply
Show advanced settings
Cancel
Apply and Close

CHAPTER

FOUR

CODE EXAMPLES AND STYLE GUIDE

Contents

4.1 Code Examples

The `extra/TI-Code-Example` folder has many well-written code examples that can be read to learn about embedded system software architect.

4.2 Code Style Style

There are multiple style guides avaliable <https://www.maultech.com/chrislott/resources/cstyle/>.

This guide is intended to explain the coding conventions used for writing the firmware in C language.

The goal of this guide is *not* to be best way to write C code, but rather provide *consistency* in the firmware code.

When in doubt, use your best judgement. Look at other examples by e.g. using Texas Instruments refrence guides.

4.2.1 Function Declaration

The documentation should be written like this.

```
/*******************************************************
 * @fn          I2C_Init
 *
 * @brief       Initialize the I2C eUSCI Interface.
 *
 * @param       none
```

(continues on next page)

(continued from previous page)

```
 *
 * @return      nonenn
 ******************************************************
 void I2C_Init(void)
```

4.2.2 Indentation

Use 4 spaces per indentation level.

4.2.3 Maximum Line Length

Keeping lines under the PEP 8 recommendation[5] to a maximum of 79 (or 99) characters helps readers easily parse the code

Code Snippet for MSP430FR2433 DCO Tuning

```
/* --COPYRIGHT--,BSD_EX
 * Copyright (c) 2014, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************
 *
 *                        MSP430 CODE EXAMPLE DISCLAIMER
```

(continues on next page)

[5] https://www.python.org/dev/peps/pep-0008/#maximum-line-length

(continued from previous page)

```
 *
 * MSP430 code examples are self-contained low-level programs that typically
 * demonstrate a single peripheral function or device feature in a highly
 * concise manner. For this the code may rely on the device's power-on default
 * register values and settings such as the clock configuration and care must
 * be taken when combining code from several examples to avoid potential side
 * effects. Also see www.ti.com/grace for a GUI- and www.ti.com/msp430ware
 * for an API functional library-approach to peripheral configuration.
 *
 * --/COPYRIGHT--*/
//******************************************************************************
//  MSP430FR243x Demo - Configure MCLK for 8MHz and XT1 sourcing ACLK and
//                      FLLREF.
//
//  Description: Configure ACLK = 32768Hz,
//               MCLK = DCO + XT1CLK REF = 8MHz,
//               SMCLK = MCLK/2 = 4MHz.
//               Toggle LED to indicate that the program is running.
//
//           MSP430FR2433
//         ---------------
//     /|\|               |
//      | |               |
//      --|RST            |
//        |          P1.0 |---> LED
//        |          P1.3 |---> MCLK  = 8MHz
//        |          P1.7 |---> SMCLK = 4MHz
//        |          P2.2 |---> ACLK  = 32768Hz
//
//
//   Ling Zhu
//   Texas Instruments Inc.
//   Feb 2015
//   Built with IAR Embedded Workbench v6.20 & Code Composer Studio v6.0.1
//******************************************************************************
#include <msp430.h>

void Software_Trim();                       // Software Trim to get the best DCOFTRIM␣
↪value
#define MCLK_FREQ_MHZ 8                     // MCLK = 8MHz

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;               // Stop watchdog timer

    P2SEL0 |= BIT0 | BIT1;                  // set XT1 pin as second function
    do
    {
        CSCTL7 &= ~(XT1OFFG | DCOFFG);      // Clear XT1 and DCO fault flag
        SFRIFG1 &= ~OFIFG;
    } while (SFRIFG1 & OFIFG);              // Test oscillator fault flag

    __bis_SR_register(SCG0);                // disable FLL
    CSCTL3 |= SELREF__XT1CLK;               // Set XT1CLK as FLL reference source
    CSCTL1 = DCOFTRIMEN | DCOFTRIM0 | DCOFTRIM1 | DCORSEL_3;// DCOFTRIM=3, DCO Range␣
↪= 8MHz
    CSCTL2 = FLLD_0 + 243;                  // DCODIV = 8MHz
```

(continues on next page)

(continued from previous page)

```
    __delay_cycles(3);
    __bic_SR_register(SCG0);                // enable FLL
    Software_Trim();                         // Software Trim to get the best␣
↪DCOFTRIM value
    CSCTL4 = SELMS__DCOCLKDIV | SELA__XT1CLK;  // Set ACLK = XT1CLK = 32768Hz
                                            // DCOCLK = MCLK and SMCLK source
    CSCTL5 |= DIVM_0 | DIVS_1;              // MCLK = DCOCLK = 8MHZ,
                                            // SMCLK = MCLK/2 = 4MHz

    P1DIR |= BIT0 | BIT3 | BIT7;            // set MCLK SMCLK and LED pin as output
    P1SEL1 |= BIT3 | BIT7;                  // set MCLK and  SMCLK pin as second␣
↪function
    P2DIR |= BIT2;                          // set ACLK pin as output
    P2SEL1 |= BIT2;                         // set ACLK pin as second function

    PM5CTL0 &= ~LOCKLPM5;                   // Disable the GPIO power-on default high-
↪impedance mode
                                            // to activate previously configured port␣
↪settings

    while(1)
    {
        P1OUT ^= BIT0;                      // Toggle P1.0 using exclusive-OR
        __delay_cycles(10000000);           // Delay for 10000000*(1/MCLK)=1.25s
    }
}

void Software_Trim()
{
    unsigned int oldDcoTap = 0xffff;
    unsigned int newDcoTap = 0xffff;
    unsigned int newDcoDelta = 0xffff;
    unsigned int bestDcoDelta = 0xffff;
    unsigned int csCtl0Copy = 0;
    unsigned int csCtl1Copy = 0;
    unsigned int csCtl0Read = 0;
    unsigned int csCtl1Read = 0;
    unsigned int dcoFreqTrim = 3;
    unsigned char endLoop = 0;

    do
    {
        CSCTL0 = 0x100;                         // DCO Tap = 256
        do
        {
            CSCTL7 &= ~DCOFFG;                  // Clear DCO fault flag
        }while (CSCTL7 & DCOFFG);               // Test DCO fault flag

        __delay_cycles((unsigned int)3000 * MCLK_FREQ_MHZ);// Wait FLL lock status␣
↪(FLLUNLOCK) to be stable
                                                           // Suggest to wait 24␣
↪cycles of divided FLL reference clock
        while((CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1)) && ((CSCTL7 & DCOFFG) == 0));

        csCtl0Read = CSCTL0;                   // Read CSCTL0
        csCtl1Read = CSCTL1;                   // Read CSCTL1
```

(continues on next page)

(continued from previous page)

```
        oldDcoTap = newDcoTap;                  // Record DCOTAP value of last time
        newDcoTap = csCtl0Read & 0x01ff;        // Get DCOTAP value of this time
        dcoFreqTrim = (csCtl1Read & 0x0070)>>4;// Get DCOFTRIM value

        if(newDcoTap < 256)                     // DCOTAP < 256
        {
            newDcoDelta = 256 - newDcoTap;      // Delta value between DCPTAP and 256
            if((oldDcoTap != 0xffff) && (oldDcoTap >= 256)) // DCOTAP cross 256
                endLoop = 1;                    // Stop while loop
            else
            {
                dcoFreqTrim--;
                CSCTL1 = (csCtl1Read & (~(DCOFTRIM0+DCOFTRIM1+DCOFTRIM2))) |␣
↪(dcoFreqTrim<<4);
            }
        }
        else                                    // DCOTAP >= 256
        {
            newDcoDelta = newDcoTap - 256;      // Delta value between DCPTAP and 256
            if(oldDcoTap < 256)                 // DCOTAP cross 256
                endLoop = 1;                    // Stop while loop
            else
            {
                dcoFreqTrim++;
                CSCTL1 = (csCtl1Read & (~(DCOFTRIM0+DCOFTRIM1+DCOFTRIM2))) |␣
↪(dcoFreqTrim<<4);
            }
        }

        if(newDcoDelta < bestDcoDelta)          // Record DCOTAP closest to 256
        {
            csCtl0Copy = csCtl0Read;
            csCtl1Copy = csCtl1Read;
            bestDcoDelta = newDcoDelta;
        }

    }while(endLoop == 0);                       // Poll until endLoop == 1

    CSCTL0 = csCtl0Copy;                        // Reload locked DCOTAP
    CSCTL1 = csCtl1Copy;                        // Reload locked DCOFTRIM
    while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1)); // Poll until FLL is locked
}
```

More information on the DCO tuning can be read on the e2e forum:

- https://e2e.ti.com/support/microcontrollers/msp430/f/166/t/806342?MSP430FR5738-Setting-DCO-frequency

CHAPTER

FIVE

FIRMWARE CODE

Contents

5.1 main.c

```
/**
 * Use of this software is copyright Ali Aljumaili and licensed under
```

(continues on next page)

(continued from previous page)

```
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
 * File {main.c}
 *
 ******************************************************************************
 *                        DATA STORAGE TAG FIRMWARE Description
 * This is the main entry point for the DST firmware developed in 2019 as a
 * part of Ali Aljumaili Master Thesis. The firmware is divided into folders for
 * peripherals-drivers, sensor-drivers, storage-drivers and finally, application
 * layer folder where the datalogger and wireless NFC functionalities exist
 ******************************************************************************
//            DST Rev.2.0.0 Simplified Schematics
//
//                   MSP430FR5738
//             ------------------------
//         /|\|                   P1.1|<---NFC_INT0
//          | |                   P1.2|<---MAG-ACC_INT1
//          --|RST                    |
 *          |                       |
//      |---- |PJ.4/XIN    P1.3/TA1.2|--> Acoustic Transmitter
//      32 kHz|                   P2.2|--> FLASH_RST
//      |---- |PJ.5/XOUT          P1.0|--> NFC_RST
 *          |                       |
//     LED1<--|P1.4               P1.6|--> I2C-SDA
//     LED2<--|PJ.0               P1.7|--> I2C-SCL
//     LED3<--|PJ.1                   |
//            |                   P2.1|<-- SPI - SOMI
//  SPI-CS1<--|PJ.2               P2.0|--> SPI - SIMO
//  SPI-CS2<--|PJ.3               P1.5|--> SPI - CLK
//
 */

/* ------------------------------------------------------------------------------------------
 *                                                Includes
 * ------------------------------------------------------------------------------------------
 */
#include <core/mcu.h>
#include <core/rtc.h>
#include <fsm.h>

#include <settings.h>
#include <msp430.h>

#include <ctpl.h>
#include <devices/ms58370.h>

#include <stdint.h>


void main(void)
{
    MCU_Init();                     /* Initialize  WDT, GPIO's and Clocks */

    RTC_B_Init();                   /* Starts the Real Time Clock with a default polling Interval */
```

(continues on next page)

(continued from previous page)

```

    Timer_B0_Init();                /* Configures the timer to up mode. */

    I2C_Init();                     /* Enables the I2C Interface eUSCI_B0 */

    MS5837_Init();                  /* Resets the pressure sensor as per datasheet */

    TMP117_Init();

    __enable_interrupt();           /* Enable Global Interrupts for using ISR */


    while(1)
    {

       FSM_Init();                 /* Starts the finite-state-machine, program in␣
↪control of handlers from now on */
    }
}


/*
 * This function will be called before main and initialization of variables.
 * The ctpl_init() function must be called at the start to enable the compute
 * through power loss library.
 */
int _system_pre_init(void)
{
    /* Initialize ctpl library */
    ctpl_init();

    /* Insert application pre-init code here. */

    return 1;
}

// This code is commented, it was used to test power loss  SVSH and SVSL
//void Reset_ISR(void) {
//
//   switch (__even_in_range(SYSRSTIV, SYSRSTIV_PMMKEY))
//   {
//       case SYSRSTIV_NONE:                       // No Interrupt pending
//                                   __no_operation();
//                                   break;
//       case SYSRSTIV_BOR:                        // SYSRSTIV : BOR
//                                   __no_operation();
//                                   break;
//       case SYSRSTIV_RSTNMI:                     // SYSRSTIV : RST/NMI
//                                   __no_operation();
//                                   break;
//       case SYSRSTIV_DOBOR:                      // SYSRSTIV : Do BOR
//                                   __no_operation();
//                                   break;
//       case SYSRSTIV_LPM5WU:                     // SYSRSTIV : Port LPM5 Wake Up
//                                   __no_operation();
//                                   break;
//       case SYSRSTIV_SECYV:                      // SYSRSTIV : Security violation
```

(continues on next page)

(continued from previous page)

```
//                                  __no_operation();
//                                  break;
//        case SYSRSTIV_SVSLIFG:                          // SYSRSTIV : SVSL
//                                  __no_operation();
//                                  break;
//        case SYSRSTIV_SVSHIFG:                          // SYSRSTIV : SVSH
//                                  __no_operation();
//                                  break;
//
//        case SYSRSTIV_DOPOR:                        // SYSRSTIV : Do POR
//                                  __no_operation();
//                                  break;
//        case SYSRSTIV_WDTTO:                        // SYSRSTIV : WDT Time out
//                                  __no_operation();
//                                  break;
//        case SYSRSTIV_WDTKEY:                       // SYSRSTIV : WDTKEY violation
//                                  __no_operation();
//                                  break;
//
//        default: break;
//    }
//}
```

5.2 Core Layer

5.2.1 i2c.c

```
//##############################################################################

//
//! \file   i2c.c
//!
//! \brief  please read i2c.h
//!
//! (C) Copyright 2019, Ali Aljumaili
//
//##############################################################################



//*****************************************************************************
// the includes
//*****************************************************************************
#include <core/i2c.h>
#include <driverlib.h>
#include <settings.h> // for global buffer

void I2C_Init(void)
{
    EUSCI_B_I2C_initMasterParam Settings_I2C_Parm = {0};
    Settings_I2C_Parm.selectClockSource = EUSCI_B_I2C_CLOCKSOURCE_SMCLK;
    Settings_I2C_Parm.i2cClk = CS_getSMCLK();
    Settings_I2C_Parm.dataRate = EUSCI_B_I2C_SET_DATA_RATE_100KBPS;
```

(continues on next page)

(continued from previous page)

```
    Settings_I2C_Parm.byteCounterThreshold = 0;
    Settings_I2C_Parm.autoSTOPGeneration = EUSCI_B_I2C_NO_AUTO_STOP;
    EUSCI_B_I2C_initMaster(EUSCI_B0_BASE, &Settings_I2C_Parm); /* Sets the Master Configurations defined in settings.h*/

    EUSCI_B_I2C_enable(EUSCI_B0_BASE); /* Enable I2C by reseting the UCSWRST bit */
}

void I2C_Set_SlaveAddr(uint8_t slaveAddr)
{
    EUSCI_B_I2C_setSlaveAddress(EUSCI_B0_BASE,slaveAddr);
}


void I2C_Send_Single_Byte(uint8_t * txData)
{
    EUSCI_B_I2C_masterSendSingleByte(EUSCI_B0_BASE,&txData);
}

void I2C_Enable_Interrupt(void)
{
    //UCB0IE |= UCSTTIE; /* Enable Start Condition Interrupt*/
    //UCB0IE |= UCTXIE0; /* Enable Transmit Interrupt*/

    UCB0IE |= UCRXIE0;  /* Enable Receive Interrupt*/
}


#pragma vector = USCI_B0_VECTOR
__interrupt void USCIB0_ISR(void)
{
    switch(__even_in_range(UCB0IV,0x1E))
    {
    case 0x00: break; // Vector 0: No interrupts break;
    case 0x02: break; // Vector 2: ALIFG break;
    case 0x04: UCB0CTL1 |= UCTXSTT; // I2C start condition
    break; // Vector 4: NACKIFG break;
    case 0x06: break; // Vector 6: STTIFG break;
    case 0x08: break; // Vector 8: STPIFG break;
    case 0x0a:

        break; // Vector 10: RXIFG3 break;
    case 0x0c: break; // Vector 14: TXIFG3 break;
    case 0x0e:
        buffer = EUSCI_B_I2C_masterReceiveMultiByteNext(EUSCI_B0_BASE); // Reads the first byte of the buffer and store it into the buffer
        EUSCI_B_I2C_masterReceiveMultiByteFinish(EUSCI_B0_BASE); //stop and ignore the input.
        break; // Vector 16: RXIFG2 break;
    case 0x10: break; // Vector 18: TXIFG2 break;
    case 0x12:
        buffer = EUSCI_B_I2C_masterReceiveMultiByteNext(EUSCI_B0_BASE); // Reads the first byte of the buffer and store it into the buffer
        EUSCI_B_I2C_masterReceiveMultiByteFinish(EUSCI_B0_BASE); //stop and ignore the input.
        break; // Vector 20: RXIFG1 break;
    case 0x14: break; // Vector 22: TXIFG1 break;
```

(continues on next page)

(continued from previous page)

```
    case 0x16:
        buffer = EUSCI_B_I2C_masterReceiveMultiByteNext(EUSCI_B0_BASE); // Reads the first byte of the buffer and store it into the buffer
        EUSCI_B_I2C_masterReceiveMultiByteFinish(EUSCI_B0_BASE); //stop and ignore the input.
        __no_operation();
        break; // Vector 24: RXIFG0
    case 0x18: break; // Vector 26: TXIFG0 break;
    case 0x1a: break; // Vector 28: BCNTIFG break;
    case 0x1c: break; // Vector 30: clock low timeout break;
    case 0x1e: break; // Vector 32: 9th bit break;
    default: break;
    }
}
```

5.2.2 i2c.h

```
//##############################################################################

//
//! \file   i2c.h
//!
//! \brief  This is the i2c interface driver, it is dependent on driverlib for
//          low level register configurations.
//          This modules configures the eUSCIB0, transmit data over I2C bus by
//          the way of interrupt and read data from the I2C by the way of interrupt.
//          This module also configures the eUSCIB0 clock source, frequency,
//          peripheral address and initialize I2C ports.
// (C) Copyright 2019, Ali Aljumaili
//
//##############################################################################


#ifndef CORE_I2C_H_
#define CORE_I2C_H_

//*****************************************************************************
// the includes
//*****************************************************************************
#include <msp430.h>
#include <stdint.h>



//! \brief This defines the I/O ports needed by the USCIBO peripheral.
//  these can be edited in case of changing port or MCU.
#define I2C_PORT_I2C_OUT    P1OUT
#define I2C_PORT_I2C_DIR    P1DIR
#define I2C_PORT_I2C_SEL    P1SEL0
#define I2C_SDA_PIN         BIT6
#define I2C_SCL_PIN         BIT7


typedef unsigned char bool_t;

```

(continues on next page)

(continued from previous page)

```
//! \brief This defines the I/O ports needed by the USCIBO peripheral. It makes
//! sense to declare the port here at the top so in case they change, they are
//!edited only here.
#define PORT_I2C_OUT    P1OUT
#define PORT_I2C_DIR    P1DIR
#define PORT_I2C_SEL    P1SEL0
#define SDA BIT6
#define SCL BIT7

//! \brief These defines are needed to keep track of data during read and write␣
↪process.
#define START           1
#define CONTINUE        0
#define FAIL            0
#define PASS            1
#define SET             1
#define CLEAR           0
#define I2C_NACK_RCVD   2
#define I2C_TRANSMIT    3
#define LPM_MODE        LPM0_bits
#define I2C_ALLOW       (I2C_READY_IN & I2C_READY_PIN)

uint8_t RF430_I2C_State;
uint8_t RF430_I2C_Start;            //AK 10-23-2013



//! \brief This declares both the register address and data.
//!
typedef struct
{
    unsigned char configReg;
    unsigned char data;
} i2cCmd_t;


//*****************************************************************************
//
//! \brief  This function is called only once and it is called when the MSP430
//!         is initializing. Furthermore, the function is called indirectly by
//!         the I2C peripheral interface module when it itself is initializing.
//!         This function configures the USCIBO peripheral to act as a I2C peripheral,
//!         set to master transmitting mode, with SMCLK clock frequency
//!         with auto detection of frequency and set the daterate to 100 kbps with no
//!         automatic stop bit generation.
//!
//! \param  none
//
//! \return None
//
//*****************************************************************************

void I2C_Init(void);

//*****************************************************************************
//
//! \brief  This function is called only once and it is called when the MSP430
```

(continues on next page)

(continued from previous page)

```
//!         is initializing. This function select I2C from eUSCIB (P1.6 and P1.7 as)
//! \param  none
//
//! \return None
//
//*****************************************************************************
void I2C_Init_Ports(void);

//*****************************************************************************
//
//! \brief  This function sets the slave address for the eUSCB_0 module.
//!
//! \param slaveAddr   Destination address of the remote IC
//
//! \return None
//
//*****************************************************************************
void I2C_Set_SlaveAddr(uint8_t slaveAddr);


//*****************************************************************************
//
//! \brief  This function transmits a single byte on the I2C bus, the slave
//!         address must be configured before calling this function.
//! \param txData Pointer to a buffer which contains the data to be written
//
//! \return None
//
//*****************************************************************************
void I2C_Send_Single_Byte(uint8_t * txData);

#endif /* CORE_I2C_H_ */
```

5.2.3 spi.c

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
*****************************************************************************
                {spi.c} - Driver for the MSP430FR5738 SPI Interface
******************************************************************************
*/

/* ------------------------------------------------------------------------------------------
 *                                          Includes
 * ------------------------------------------------------------------------------------------
 */
#include <core/spi.h>
#include <driverlib.h>

void SPI_Init(void)
{
```

(continues on next page)

(continued from previous page)

```
    /* initialize eUSCI SPI master mode */
    EUSCI_A_SPI_masterInit(EUSCI_A0_BASE,
                           EUSCI_A_SPI_CLOCKSOURCE_SMCLK,
                           12000000, /* SMCLK Speed */
                           1000000,  /* SPI Bus Bitrate */
                           EUSCI_A_SPI_LSB_FIRST,
                           EUSCI_A_SPI_PHASE_DATA_CHANGED_ONFIRST_CAPTURED_ON_NEXT,
                           EUSCI_A_SPI_CLOCKPOLARITY_INACTIVITY_LOW,
                           EUSCI_A_SPI_3PIN);

    /* enable eUSCI SPI */
    EUSCI_A_SPI_enable(EUSCI_A0_BASE);
}


/*  ======== eUSCI_A0 Interrupt Service Routine ========
*/
#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR_HOOK(void)
{
   /* USER CODE START (section: USCI_A0_ISR_HOOK) */
   /* replace this comment with your code */
   /* USER CODE END (section: USCI_A0_ISR_HOOK) */
}
```

5.2.4 spi.h

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
*************************************************************************
               {spi.h} - Driver for the MSP430FR5738 SPI Interface
**************************************************************************
*/

#ifndef CORE_SPI_H_
#define CORE_SPI_H_
/* ------------------------------------------------------------------------------------------
 *                                          Includes
 * ------------------------------------------------------------------------------------------
 */

/* ------------------------------------------------------------------------------------------
 *                                          Enums
 * ------------------------------------------------------------------------------------------
 */

/* ------------------------------------------------------------------------------------------
 *                                          Defines
```

(continues on next page)

(continued from previous page)

```
 * ------------------------------------------------------------------------------------
 */

/* ------------------------------------------------------------------------------------
 *                                        Function Prototypes
 * ------------------------------------------------------------------------------------
 */
void SPI_Init(void);


#endif /* CORE_SPI_H_ */
```

5.2.5 mcu.c

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
*************************************************************************
                  {mcu.c} - Driver for the MSP430FR5738 Peripherals
**************************************************************************
*/

/* ------------------------------------------------------------------------------------
 *                                            Includes
 * ------------------------------------------------------------------------------------
 */
#include <core/mcu.h>
#include <driverlib/MSP430FR57xx/inc/hw_memmap.h>
#include <driverlib/MSP430FR57xx/cs.h>
#include <ctpl.h>

/* Sets up the MSP430FR738 GPIO, Watchdog and Clocks */
void MCU_Init(void){

    MCU_Stop_Watchdog();                    /* Stops the WDT*/
    MCU_Configure_GPIO();                   /* Setup GPIOs */
    MCU_Configure_Clocks();                 /* Configure DCO and ACLK */
}

/* Stops the watchdog timer to avoid timing out during start-up */
void MCU_Stop_Watchdog(void)
{
     WDTCTL = WDTPW | WDTHOLD;              /* Stop watchdog timer from
                                            timing out during initial start-up */
}

/* Initialize MSP430 General Purpose Input Output Ports
 *  The GPIO registers should be set in a specific order:
 *   PxOUT --> PxSEL or PxSELx -->  PxDIR -->  PxREN -->  PxIES --> PxIFG -->  PxIE
```

(continues on next page)

(continued from previous page)

```
 *   (see section 8.2.6 of the MSP430 User's manual )
*/
void MCU_Configure_GPIO(void)
{
/*================================ Port 1 GPIO Setup ================================*/
    P1OUT = MCU_NFC_RST_PIN;                        /* Pins Output Registers PORT1 */

    P1SEL1 = (MCU_I2C_SDA_PIN | MCU_I2C_SCL_PIN /* Port 1 Port Select Register */
             | MCU_SPI_CLK_PIN);                 /* Select I2C and SPI */

    P1DIR = (MCU_ACC_MAG_PIN|                    /* PORT 1 Pins Direction Registers */
            MCU_ACOUSTIC_TRANSMITTER_PIN);       /* Set pins as output */

    P1REN = MCU_NFC_RST_PIN;                     /* PORT 1 Enable Pullup for NFC_RST */

    P1IES = (MCU_NFC_INT_PIN |                   /* PORT1 Interrupt Edge Select Register */
            MCU_ACC_MAG_INT_PIN);

    P1IFG = 0;                                   /* Port 1 Interrupt Flag Register */

    P1IE = (MCU_NFC_INT_PIN |
            MCU_ACC_MAG_INT_PIN);                /* Port 1 Enable Interrupts for pins */

    /*================================ Port 2 GPIO Setup ================================*/
    P2OUT = MCU_FLASH_RST_PIN;                   /* Pins Output Registers PORT2 */

    P2SEL1 = (MCU_SPI_SIMO_PIN |
            MCU_SPI_SOMI_PIN);                   /* Port 2 Port Select Register */

    P2DIR = 0;                                   /* Set All PORT2 Pins as Input*/

    P2REN = MCU_FLASH_RST_PIN;                   /* Enable Pullup for FLASH */

    P2IES = 0;                                   /* Interrupt Edge Select Register */

    P2IFG = 0;                                   /* Interrupt Flag Register */

    /*================================ Port J GPIO Setup ================================*/

    PJSEL0  =  XIN_PIN | XOUT_PIN;               /* Set up XIN and XOUT for bypass crystal mode*/

    //PJSEL0 |= BIT0 | BIT1; /* Output MCLK and SMCLK to J0 and J1 (only for testing)*/

    PJDIR |= BIT2;                               /* Set pin as output for ACLK */
    PJSEL0 |= SELECT_ACLK_OUTPUT_PIN_2;          /* Set PJ.2 as ACLK for testing */
    PJSEL1= 0; // for test

    PJDIR   = LED2 |LED3;                        /* Set pins as output for LED2 and LED3 */

```

(continues on next page)

(continued from previous page)

```
    /* Disable the GPIO power-on default high-impedance mode to activate previously configured port settings */
    PM5CTL0 &= ~LOCKLPM5;
}


/* Sets up the DCO, SMCLK and  XT1 as a source for ACLK*/
void MCU_Configure_Clocks(void)
{
    CS_setExternalClockSource(32768,0);             /*   XT1 Frequency -  32768 Hz,
                                                     *   XT2 Frequency -  0 Hz     */

    CS_turnOnXT1WithTimeout(XT1DRIVE_3,100000);     /* Start XT1 crystal in low frequency mode */

    CS_setDCOFreq (CS_DCORSEL_1, CS_DCOFSEL_3);     /* Set DCO frequency to 24 MHz */
    //CS_setDCOFreq (CS_DCORSEL_1, CS_DCOFSEL_1);   /* Set DCO frequency to 20 MHz */
    //CS_setDCOFreq (CS_DCORSEL_0, CS_DCOFSEL_3);   /* Set DCO frequency to 8 MHz */

    CS_initClockSignal(CS_MCLK, CS_DCOCLK_SELECT,   //* Setting MCLK source from CS_DCOCLK_SELECT
                       CS_CLOCK_DIVIDER_1);         //* with the divider of CS_CLOCK_DIVIDER_1.

    CS_initClockSignal(CS_SMCLK, CS_DCOCLK_SELECT,  //* Setting SMCLK source from CS_DCOCLK_SELECT with the
                        CS_CLOCK_DIVIDER_2);        //* the divider of CS_CLOCK_DIVIDER_2 frequency 24(DCO)/2 = 12 MHz

    CS_initClockSignal(CS_ACLK, CS_XT1CLK_SELECT,  //* Setting ACLK source from CS_XT1CLK_SELECT
                         CS_CLOCK_DIVIDER_1);       //* with the divider of CS_CLOCK_DIVIDER_1.

    CS_clearAllOscFlagsWithTimeout(100000);        /* Clears all oscillator fault flags including
                                                    * global oscillator fault flag before switching clock sources. */
}


///* Calibrate DCO to a certain frequency */
//void MCU_Calibrate_DCO(uint16_t delta)
//{
//    uint16_t Compare, Oldcapture = 0;
//
//    /* Set Timer */
//  //  BCSCTL1 |= DIVA_3;                        // ACLK = LFXT1CLK/8 */
//  //  TACCTL0 = CM_1 + CCIS_1 + CAP;            /* CAPture, ACLK */
//  //  TACTL = TASSEL_2 + MC_2 + TACLR;          /* SMCLK, count-mode, clear */
//
//    while (1)
//    {
//        while (!(CCIFG & TACCTL0));             /* Wait until capture occured */
//        TACCTL0 &= ~CCIFG;                      /* Capture occured, clear flag */
//        Compare = TACCR0;                       /* Get current captured SMCLK */
//        Compare = Compare - Oldcapture;         /* SMCLK difference */
```

(continues on next page)

(continued from previous page)

```
//        Oldcapture = TACCR0;                    /* Save current captured SMCLK */
//
//        if (delta == Compare)
//            break;                              // If equal, leave "while(1)"
//        else if (delta < Compare)
//        {
//            DCOCTL--;                           /* DCO is too fast, slow it down */
//            if (DCOCTL == 0xFF)                 /* Did DCO roll under? */
//                if (BCSCTL1 & 0x0f)
//                    BCSCTL1--;                  /* Select lower RSEL */
//        }
//        else
//        {
//            DCOCTL++;                           /* DCO is too slow, speed it up */
//
//            if (DCOCTL == 0x00)                 /* Did DCO roll over? */
//                if ((BCSCTL1 & 0x0f) != 0x0f)
//                    BCSCTL1++;                  /* Select higher RSEL */
//        }
//    }
//    TACCTL0 = 0;                                /* Stop TACCR0 */
//    TACTL = 0;                                  /* Stop Timer_A */
//    BCSCTL1 &= ~DIVA_3;                         /* ACLK = LFXT1CLK */
//
//    for (i = 0; i < 0x4000; i++);               /* SW Delay */
//}
```

5.2.6 mcu.h

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
***************************************************************************
                  {mcu.h} - Driver for the MSP430FR5738 Peripherals
*****************************************************************************
*/

#ifndef CORE_MCU_H_
#define CORE_MCU_H_

/* ------------------------------------------------------------------------------------------
 *                                          Includes
 * ------------------------------------------------------------------------------------------
 */
#include <msp430.h>

/* ------------------------------------------------------------------------------------------
 *                                          Defines
 * ------------------------------------------------------------------------------------------
 */
```

(continues on next page)

(continued from previous page)

```
#define MCU_SPI_CLK_PIN BIT5
#define MCU_SPI_SOMI_PIN BIT1
#define MCU_SPI_SIMO_PIN BIT0

#define MCU_I2C_SDA_PIN BIT6
#define MCU_I2C_SCL_PIN BIT7

#define MCU_NFC_INT_PIN     BIT1
#define MCU_ACC_MAG_INT_PIN BIT2

#define MCU_NFC_RST_PIN BIT0
#define MCU_FLASH_RST_PIN BIT2

#define MCU_ACC_MAG_PIN BIT2
#define MCU_ACOUSTIC_TRANSMITTER_PIN BIT3

#define LED1 BIT4
#define LED2 BIT0
#define LED3 BIT1

#define LED1_OUT P1OUT
#define LED2_OUT PJOUT
#define LED3_OUT PJOUT

#define LED1_DIR P1DIR
#define LED2_DIR PJDIR
#define LED3_DIR PJDIR

#define LED1_POWER_ON   LED1_DIR |=LED1; LED1_OUT|=LED1;
#define LED1_POWER_OFF  LED1_OUT &= ~LED1;
#define LED2_POWER_ON   LED2_DIR |=LED2; LED2_OUT|=LED2;
#define LED2_POWER_OFF  LED2_OUT &= ~LED2;
#define LED3_POWER_ON   LED3_DIR |=LED3; LED3_OUT|=LED3;
#define LED3_POWER_OFF  LED3_OUT &= ~LED3;

#define XIN_PIN  BIT4
#define XOUT_PIN BIT5

#define SELECT_ACLK_OUTPUT_PIN_2 BIT2
/* ------------------------------------------------------------------------------------------
 *                                          Prototypes
 * ------------------------------------------------------------------------------------------
 */
void MCU_Stop_Watchdog(void);
void MCU_Configure_GPIO(void);
void MCU_Init(void);
void MCU_Configure_Clocks(void);
//void MCU_Calibrate_DCO(uint16_t delta);

#endif /* CORE_MCU_H_ */
```

5.2.7 timer.c

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
***************************************************************************
                {timer.h} - Driver for timer MSP430FR5738
****************************************************************************
*/

#ifndef CORE_TIMER_H_
#define CORE_TIMER_H_

/* ------------------------------------------------------------------------------------------
 *                                        Includes
 * ------------------------------------------------------------------------------------------
 */
#include <msp430.h>
#include <stdint.h>


#define TIMER_B_STOP_MODE                                                    MC_0
#define TIMER_B_UP_MODE                                                      MC_1
#define TIMER_B_CONTINUOUS_MODE                                              MC_2
#define TIMER_B_UPDOWN_MODE                                                  MC_3


/* ------------------------------------------------------------------------------------------
 *                                        Function Prototypes
 * ------------------------------------------------------------------------------------------
 */
void Low_Power_Delay_ms(uint32_t ms);
void Timer_B0_Init(void);

#endif /* CORE_TIMER_H_ */
```

5.2.8 timer.h

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
***************************************************************************
                {timer.h} - Driver for timer MSP430FR5738
****************************************************************************
*/

#ifndef CORE_TIMER_H_
#define CORE_TIMER_H_

```

(continues on next page)

(continued from previous page)

```
/* ------------------------------------------------------------------------------------------
 * Includes
 * ------------------------------------------------------------------------------------------
 */
#include <msp430.h>
#include <stdint.h>


#define TIMER_B_STOP_MODE                                                    MC_0
#define TIMER_B_UP_MODE                                                      MC_1
#define TIMER_B_CONTINUOUS_MODE                                              MC_2
#define TIMER_B_UPDOWN_MODE                                                  MC_3


/* ------------------------------------------------------------------------------------------
 * Function Prototypes
 * ------------------------------------------------------------------------------------------
 */
void Low_Power_Delay_ms(uint32_t ms);
void Timer_B0_Init(void);

#endif /* CORE_TIMER_H_ */
```

5.3 Application Layer

5.3.1 fsm.c

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
***************************************************************************
                  {fsm.c} - Driver for the Finite State Machine
*****************************************************************************
*/

/* ------------------------------------------------------------------------------------------
 * Includes
 * ------------------------------------------------------------------------------------------
 */
#include <fsm.h>
#include <settings.h>
#include <nfc_request_handler.h>
#include <ctpl.h>
#include <ms58370.h>
/*For Testing, shall be moved to an event handler */
```

(continues on next page)

(continued from previous page)

```
void FSM_Init(void)
{
    switch(CurrentState_t)
    {
    case Start:
        //MAKE LED BLINK

        I2C_Send_Single_Byte(MS5837_CONVERT_D1_8192);
        //__delay_cycles(24000*2); // wait 3 ms for the pressure sensor to get␣
↪restarted. This should be accomplished by timers now.

        MS5738_Convert_And_Read_ADC();

        //
        //   __delay_cycles(1000000);                // Delay between transmissions

        //   while (EUSCI_B_I2C_SENDING_STOP == EUSCI_B_I2C_masterIsStopSent(EUSCI_B0_
↪BASE));
        //   __delay_cycles(1000000);                // Delay between transmissions
        //        //

        /* Enter into LPM3.5 with restore on reset disabled. The RTC interrupt will␣
↪wake up the MCU */
        ctpl_enterLpm35(CTPL_DISABLE_RESTORE_ON_RESET);
        break;
    case Power_Lost:

        break;

    case Data_logger_Init:

        break;

    case Wait_For_Command:

        //check NFC
        break;

    case Process_command:

        /* Switch Depending On Received Command */

        switch(Command_Received_t)
        {
        case Start_CMD:

            break;

        case Stop_CMD:

            break;

        case Clear_Data_CMD:

            break;

        case Reset_CMD:
```

(continues on next page)

(continued from previous page)

```
            break;

        }

        break;
//
//    default:
//        CurrentState_t = Wait_For_Command;
//        break;
    }
}
```

5.3.2 fsm.h

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
***************************************************************************
                  {fsm.h} - Driver for the Finite State Machine
****************************************************************************
*/

#ifndef APPLICATION_LAYER_FSM_H_
#define APPLICATION_LAYER_FSM_H_

/* ------------------------------------------------------------------------------------
↪-------------
 *                                          Includes
 * ------------------------------------------------------------------------------------
↪-------------
 */
#include <msp430.h>

/* ------------------------------------------------------------------------------------
↪-------------
 *                                       Structs and Enums
 * ------------------------------------------------------------------------------------
↪-------------
 */
typedef enum {
    Start,
    Power_Lost,
    Data_logger_Init,
    Wait_For_Command,
    Process_command,
} States_t;

/* ------------------------------------------------------------------------------------
↪-------------
 *                                          Defines
 * ------------------------------------------------------------------------------------
↪-------------
```

(continues on next page)

(continued from previous page)

```
 */

/* ------------------------------------------------------------------------------------
↪-------------
 *                                                Prototypes
 * ------------------------------------------------------------------------------------
↪-------------
 */
void FSM_Init(void);
#endif /* APPLICATION_LAYER_FSM_H_ */
```

5.3.3 settings.c

```
/*
 * settings.c
 *
 *  Created on: May 16, 2019
 *      Author: ali
 */

#include <settings.h>


States_t CurrentState_t = Start;

Datalogger_Commands_t Command_Received_t;

uint32_t buffer;
```

5.3.4 setings.h

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
 ***************************************************************************
              {settings.h} - Application Settings (mostly global)
 ***************************************************************************
 */

#ifndef APPLICATION_LAYER_SETTINGS_H_
#define APPLICATION_LAYER_SETTINGS_H_
/* ------------------------------------------------------------------------------------
↪-------------
 *                                                Includes
 * ------------------------------------------------------------------------------------
↪-------------
 */
#include "application_layer/fsm.h"
#include "stdint.h"
#include "driverlib.h"
#include <nfc_request_handler.h>
```

(continues on next page)

(continued from previous page)

```
typedef struct Settings_Global_t {
    uint8_t ui8PollingInterval; /* Polling Interval used by RTC for alarm in Minutes␣
↪*/
}Settings_Global_t;


extern uint32_t buffer; //global variable for testing to store I2C data


/* ------------------------------------------------------------------------------------
↪-------------
 *                                          Global States
 * ------------------------------------------------------------------------------------
↪-------------
 */

extern States_t CurrentState_t;

/* Used in FSM */
#pragma NOINIT (Command_Received_t);
extern Datalogger_Commands_t Command_Received_t;

//States_t CurrentState_t = Start;
//DEBUGGER MODE STATE
//FLAGS


#endif /* APPLICATION_LAYER_SETTINGS_H_ */
```

5.4 Devices Layer

5.4.1 tmp117.c

```
/**
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
 * File {tmp117.c}
 */

/* ------------------------------------------------------------------------------------
↪-------------
 *                                            Includes
 * ------------------------------------------------------------------------------------
↪-------------
 */
#include <devices/tmp117.h>


void TMP117_Init(void)
{
    // set temperature offset
```

(continues on next page)

(continued from previous page)

```
    // set low temp limit
    // set high temp limit
}

//constant uint8_t TMP117_DeviceAddress=TMP117_DeviceID1;


//float uint2floatconvertdata(uint8_t HiByte, uint8_t LoByte) {
//    float temp;
//    temp = (float) (HiByte << 8 | LoByte);
//    temp *= 0.0078125;
//    return temp;
//}
```

5.4.2 tmp117.h

```
/**
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
 * File {tmp117.h}
 */

#ifndef DEVICES_TMP117_H_
#define DEVICES_TMP117_H_
/* ------------------------------------------------------------------------------------
 ↪-------------
 *                                          Includes
 * ------------------------------------------------------------------------------------
 ↪-------------
 */

/* ------------------------------------------------------------------------------------
 ↪-------------
 *                                          Defines
 * ------------------------------------------------------------------------------------
 ↪-------------
 */
/* TMP117 Addresses */
#define TMP117_ADDR                0x48            /*  ADDR connected to GND */
#define TMP117_DeviceID2           0x49            /*  Vcc */
#define TMP117_DeviceID3           0x4A            /*  SDA */
#define TMP117_DeviceID4           0x4B            /*  SCL */

/* TMP117 Registers */
#define TMP117_REG_TEMPERATURESULT 0x00
#define TMP117_REG_CONFIGURATION 0x01
#define TMP117_REG_ALERTHIGH 0x02
#define TMP117_REG_ALERTLOW 0x03
#define TMP117_REG_EEPROMUNLOCK 0x04
#define TMP117_REG_EEPROM1 0x05
#define TMP117_REG_EEPROM2 0x06
#define TMP117_REG_OFFSET 0x07
#define TMP117_REG_EEPROM3 0x08
#define TMP117_REG_DEVICEID 0x0F
```

(continues on next page)

(continued from previous page)

```
//from below is unverfied.
#define TMP117CONFIGREGISTERPOL 8
#define TMP117CONFIGREGISTERTnA 16
#define TMP117CONFIGREGISTERAVG0 32
#define TMP117CONFIGREGISTERAVG1 64
#define TMP117CONFIGREGISTERCONV0 128

#define TMP117CONFIGREGISTERCONV1 1
#define TMP117CONFIGREGISTERCONV2 2
#define TMP117CONFIGREGISTERMOD0 4
#define TMP117CONFIGREGISTERMOD1 8
#define TMP117CONFIGREGISTEREEPROMBUSY 16
#define TMP117CONFIGREGISTERDATAREADY 32
#define TMP117CONFIGREGISTERLOWALERT 64
#define TMP117CONFIGREGISTERHIGHALERT 128

#define TMP117SHUTDOWN 0x04
#define TMP117ONESHOT 0x06


/* ------------------------------------------------------------------------------------
↪-------------
 *                                        Function Prototypes
 * ------------------------------------------------------------------------------------
↪-------------
 */
void TMP117_Init(void);

#endif /* DEVICES_TMP117_H_ */
```

5.4.3 ms58370.c

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
 ***************************************************************************
             {ms5738.c} - Driver for the pressure sensor
 ***************************************************************************
/*

/* ------------------------------------------------------------------------------------
↪-------------
 *                                            Includes
 * ------------------------------------------------------------------------------------
↪-------------
 */
#include <core/i2c.h>
#include <devices/ms58370.h>
#include <driverlib.h> //TO BE MOVED to I2C
```

(continues on next page)

(continued from previous page)

```
/**
 * \brief Configures the I2C master to be used with the MS5837 device.
 */
void MS5837_Init(void)
{
    I2C_Set_SlaveAddr(MS5837_ADDR);      /* Set I2C Slave Address to MS5837 ADDR */

    I2C_Send_Single_Byte(MS5837_RESET); /* S-- > Device Address --> W --> A --> RESET␣
↪COMMAND --> A --> Stop */
}

//uint32_t *adcValue
void MS5837_Convert_And_Read_ADC(void)
{

    //I2C_Set_SlaveAddr(MS5837_ADDR);      /* Set I2C Slave Address to MS5837 ADDR */
   // I2C_Send_Single_Byte(MS5837_RESET); /* S-- > Device Address --> W --> A -->␣
↪RESET COMMAND --> A --> Stop */

    //__delay_cycles(48000); // wait 2 ms after reset.

    //(1/24MHz)*X=1ms,           MCLK= 24 Mhz (MCLK is the source for delay cycles)
    //X=24000 1 ms

    I2C_Send_Single_Byte(MS5837_CONVERT_D1_8192); //issue command to start ADC␣
↪pressure conversion D1

    __delay_cycles(432000); // delay for 18 ms

    I2C_Send_Single_Byte(MS5837_ADC_READ); //issue the command for ADC read

    __delay_cycles(432000); // delay for 18 ms

    EUSCI_B_I2C_masterReceiveStart(EUSCI_B0_BASE); //set I2C in Read Mode

    //read the 3 byte buffer.
    UCB0IE |=  UCRXIE0;                //receive interrupt enable
    // then check interrupt
    __no_operation();
}


void MS5837_Calculate()
{
    int32_t dT = 0;
    int64_t SENS = 0;
    int64_t OFF = 0;
    int32_t SENSi = 0;
    int32_t OFFi = 0;
    int32_t Ti = 0;
    int64_t OFF2 = 0;
    int64_t SENS2 = 0;


}
```

5.4.4 ms58370.h

```
/*
 * Use of this software is copyright Ali Aljumaili and licensed under
 * the MIT license found in the LICENSE file associated to this repository.
 * Copyright (c) 2019, Ali Aljumaili
 ******************************************************************************
            {ms5738.h} - Driver for the pressure sensor
 ******************************************************************************
*/

#ifndef DEVICES_MS58370_H_
#define DEVICES_MS58370_H_


/* ------------------------------------------------------------------------------------
 *                                          Constants
 * ------------------------------------------------------------------------------------
 */
//const float MS5837_PA = 100.0f;
//const float MS5837_BAR = 0.001f;
//const float MS5837_MBAR = 1.0f;



/* ------------------------------------------------------------------------------------
 *                                          Defines
 * ------------------------------------------------------------------------------------
 */
/* MS5737 Device Commands */
#define MS5837_ADDR                                                 0x76
#define MS5837_RESET                                                0x1E
#define MS5837_ADC_READ                                             0x00
#define MS5837_PROM_READ                                            0xA0
#define MS5837_START_PRESSURE_ADC_CONVERSION                        0x40
#define MS5837_START_TEMPERATURE_ADC_CONVERSION                     0x50


#define MS5837_CONVERT_D1_256                                       0x40
#define MS5837_CONVERT_D1_1024                                      0x44
#define MS5837_CONVERT_D1_2048                                      0x46
#define MS5837_CONVERT_D1_4096                                      0x48
#define MS5837_CONVERT_D1_8192                                      0x4A

#define MS5837_CONVERT_D2_256                                       0x50
#define MS5837_CONVERT_D2_1024                                      0x54
#define MS5837_CONVERT_D2_2048                                      0x56
#define MS5837_CONVERT_D2_4096                                      0x58
#define MS5837_CONVERT_D2_8192                                      0x5A

/* PROM READ ADDRESSES */
#define MS5837_PROM_ADDRESS_0                                       0xA0
#define MS5837_PROM_ADDRESS_1                                       0xA2
```

(continues on next page)

(continued from previous page)

```
#define MS5837_PROM_ADDRESS_2                                    0xA4
#define MS5837_PROM_ADDRESS_3                                    0xA6
#define MS5837_PROM_ADDRESS_4                                    0xA8
#define MS5837_PROM_ADDRESS_5                                    0xAA
#define MS5837_PROM_ADDRESS_6                                    0xAC
#define MS5837_PROM_ADDRESS_7                                    0xAE

/* Coefficients indexes for temperature and pressure computation */

#define MS5837_CRC_INDEX                                         0
#define MS5837_PRESSURE_SENSITIVITY_INDEX                        1
#define MS5837_PRESSURE_OFFSET_INDEX                             2
#define MS5837_TEMP_COEFF_OF_PRESSURE_SENSITIVITY_INDEX          3
#define MS5837_TEMP_COEFF_OF_PRESSURE_OFFSET_INDEX               4
#define MS5837_REFERENCE_TEMPERATURE_INDEX                       5
#define MS5837_TEMP_COEFF_OF_TEMPERATURE_INDEX                   6
#define MS5837_COEFFICIENT_NUMBERS                               7


/* ------------------------------------------------------------------------------------
 ↪-------------
 *                                    Function Prototypes
 * ------------------------------------------------------------------------------------
 ↪-------------
 */
void MS5837_Init(void);
void MS5837_Convert_And_Read_ADC(void);
#endif /* DEVICES_MS58370_H_ */
```

CHAPTER

SIX

MATLAB SIMULATION

Contents

6.1 Code for Matlab Simulation

```
        %% Generate magnetic field parameters using WMM and IGRF models as well to SST data
        % Made by Ali Aljumaili (Ali.jum@outlook.com) for data storage tag project

        clear all
        close all
        clc

        %% Make 'datadir' folder in Current Folder

        datadir = fullfile(pwd, 'datadir');
        if ~exist(datadir, 'dir')
           mkdir(datadir)
        end

        datafile = @(filename)fullfile(datadir, filename);


        %% set the latitude and longitude limits
        latlimits = [63 79];
        lonlimits = [0 27];


        %% NetCDF (SST DATA)
        % https://data.nodc.noaa.gov/cgi-bin/iso?id=gov.noaa.nodc:AVHRR_Pathfinder-NCEI-L3C-v5.3
        % ftp://ftp.nodc.noaa.gov/pub/data.nodc/pathfinder/Version5.3/L3C/
        % http://coastwatch.pfeg.noaa.gov/erddap/griddap/erdMH1sstdmday.html
        % the file example: A20172742017304.L3m_MO_SST_sst_4km.nc

```

(continues on next page)

(continued from previous page)

```
        [file,folder] = uigetfile('*.nc', 'Select the NetCDF file');
        filename = fullfile(folder,file)

        finfo = ncinfo(filename);
        %disp(finfo);

        AttrNames = {finfo.Attributes.Name};
        minMatch = strncmpi(AttrNames,'data_minimun',10);
        maxMatch = strncmpi(AttrNames,'data_maximun',10);
        dateMatch = strncmpi(AttrNames,'date_created',10);

        AttrVal = {finfo.Attributes.Value};
        minVal = cell2mat(AttrVal(find(minMatch,1)));
        maxVal = cell2mat(AttrVal(find(maxMatch,1)));
        dateVal = AttrVal(find(dateMatch,1));

        ncid = netcdf.open(filename);

        varname_sst = netcdf.inqVar(ncid,0);
        varname_lat = netcdf.inqVar(ncid,2);
        varname_lon = netcdf.inqVar(ncid,3);
        varname_pal = netcdf.inqVar(ncid,4);

        varid_sst = netcdf.inqVarID(ncid,varname_sst);
        varid_lat = netcdf.inqVarID(ncid,varname_lat);
        varid_lon = netcdf.inqVarID(ncid,varname_lon);
        varid_pal = netcdf.inqVarID(ncid,varname_pal);

        sst = netcdf.getVar(ncid,varid_sst);
        sst(sst<=-32767 | sst>=32767)=nan;
        sst = double(sst);
        sstC = sst*(maxVal-minVal)/(max(sst(:))-min(sst(:)));
        lat = netcdf.getVar(ncid,varid_lat);
        lon = netcdf.getVar(ncid,varid_lon);


        % plot full lat-lon range
        figure(1)
        [latgrid, longrid] = meshgrid(lat,lon);
        geoshow(latgrid,longrid,uint8(sstC),'DisplayType', 'texturemap')
        hcb = colorbar('eastoutside');
        title(['Sea Surface Temperature  T,C   ', dateVal])
        xlabel('Longitude')
        ylabel('Latitude')


        % select lat-lon range
        rangelat = find(lat>latlimits(1) & lat<latlimits(2));
        rangelon = find(lon>lonlimits(1) & lon<lonlimits(2));
        sstC_clip = sstC(rangelon,rangelat);
        lat_clip = double(lat(rangelat));
        lon_clip = double(lon(rangelon));


        % plot selected lat-lon range
        figure(2)
        [latgrid, longrid] = meshgrid(lat_clip,lon_clip);
```

(continues on next page)

(continued from previous page)

```
        geoshow(latgrid,longrid,uint8(sstC_clip),'DisplayType', 'texturemap')
        hcb = colorbar('eastoutside');
        title(['Sea Surface Temperature  T,C   ', dateVal])
        xlabel('Longitude')
        ylabel('Latitude')


        R = georefcells([lat_clip(end) lat_clip(1)],[lon_clip(1) lon_clip(end)],...
            size(sstC_clip'),'ColumnsStartFrom','north');

        filename_geotiff = datafile('sst.tif');
        geotiffwrite(filename_geotiff,sstC_clip',R)



        %% WMM (World Magnetic model)

        listlat = lat_clip';
        listlon = lon_clip';

        %% Magneticfield parameters
        %declination, inclination, horizontal component, north component,
        %east component, vertical component, and total field
        xyz = zeros(length(listlat), length(listlon), 3);
        h = zeros(length(listlat), length(listlon));
        dec = zeros(length(listlat), length(listlon));
        dip = zeros(length(listlat), length(listlon));
        f = zeros(length(listlat), length(listlon));

        for i = 1:length(listlat)
            for j = 1:length(listlon)
                [xyz(i,j,1:3), h(i,j), dec(i,j), dip(i,j), f(i,j)] = ...
                    wrldmagm(0, listlat(i), listlon(j), decyear(2017,12,5), '2015');
            end
        end


        R = georefcells([listlat(end) listlat(1)],[listlon(1) listlon(end)],...
            size(f),'ColumnsStartFrom','north');

        filename_geotiff = datafile('WMM_F.tif');
        geotiffwrite(filename_geotiff,f,R)

        filename_geotiff = datafile('WMM_H.tif');
        geotiffwrite(filename_geotiff,h,R)

        filename_geotiff = datafile('WMM_Dec.tif');
        geotiffwrite(filename_geotiff,dec,R)

        filename_geotiff = datafile('WMM_Dip.tif');
        geotiffwrite(filename_geotiff,dip,R)


        %% IGRF (International Geomagnetic Reference Field)

        listlat = lat_clip';
        listlon = lon_clip';
```

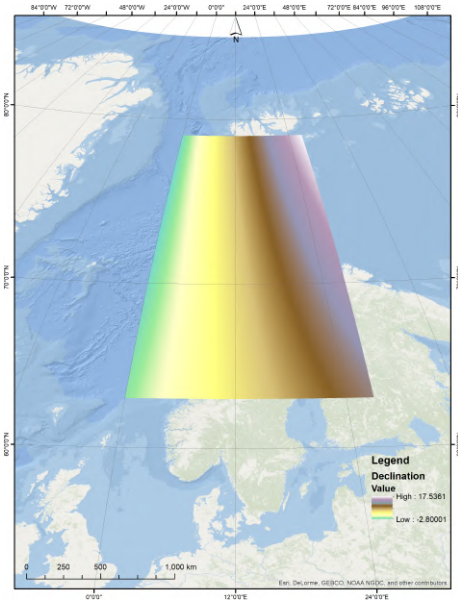
(continues on next page)

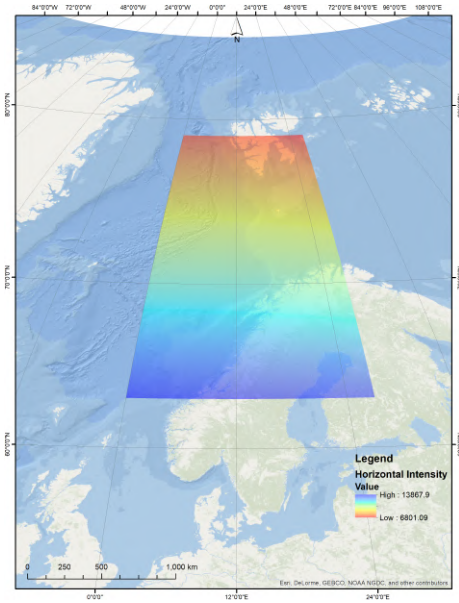
(continued from previous page)

```
        %% Magneticfield parameters
        %declination, inclination, horizontal component, north component,
        %east component, vertical component, and total field
        %xyz = zeros(length(listlat), length(listlon), 3);
        h = zeros(length(listlat), length(listlon));
        dec = zeros(length(listlat), length(listlon));
        dip = zeros(length(listlat), length(listlon));
        f = zeros(length(listlat), length(listlon));

        for i = 1:length(listlat)
            for j = 1:length(listlon)
                [~,h(i,j),dec(i,j),dip(i,j),f(i,j)] ...
                    = igrfmagm(0,listlat(i), listlon(j), decyear(2017,12,5),12);
            end
        end


        R = georefcells([listlat(end) listlat(1)],[listlon(1) listlon(end)],...
            size(f),'ColumnsStartFrom','north');

        filename_geotiff = datafile('IGRF_F.tif');
        geotiffwrite(filename_geotiff,f,R)

        filename_geotiff = datafile('IGRF_H.tif');
        geotiffwrite(filename_geotiff,h,R)

        filename_geotiff = datafile('IGRF_Dec.tif');
        geotiffwrite(filename_geotiff,dec,R)

        filename_geotiff = datafile('IGRF_Dip.tif');
        geotiffwrite(filename_geotiff,dip,R)
```

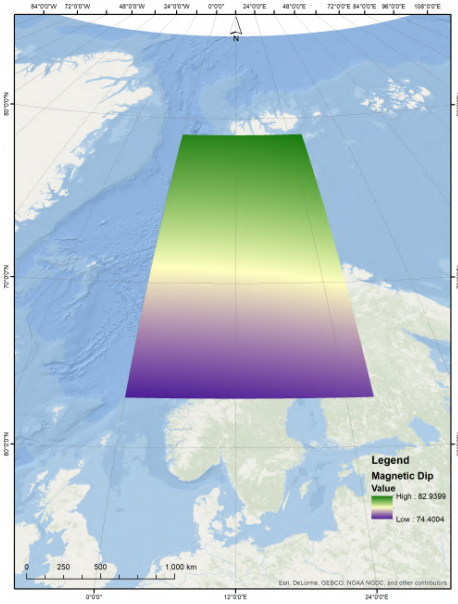
6.1.1 Simulation Results

IGRF Simulation Results

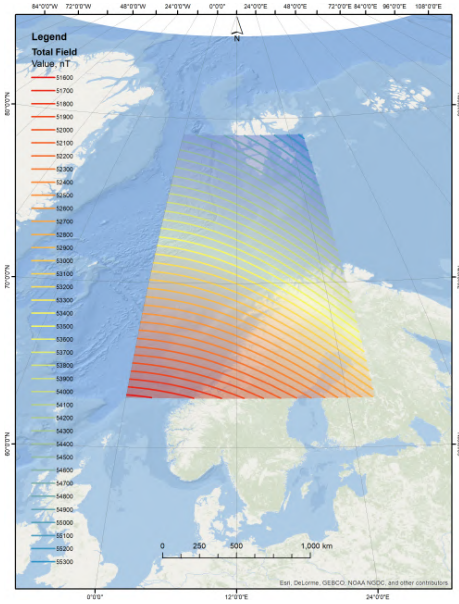

Legend
Declination
Value
High : 17.5361
Low : -2.80001
0 250 500 1,000 km
Esri, DeLorme, GEBCO, NOAA NGDC, and other contributors


Legend
Horizontal Intensity
Value
High : 13867.9
Low : 6801.09
0 250 500 1,000 km
Esri, DeLorme, GEBCO, NOAA NGDC, and other contributors

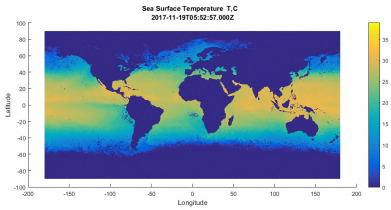
and


Legend
Magnetic Dip
Value
High : 82.9399
Low : 74.4004
0 250 500 1,000 km
Esri, DeLorme, GEBCO, NOAA NGDC, and other contributors

and


Legend
Total Field
Value, nT
0 250 500 1,000 km
Esri, DeLorme, GEBCO, NOAA NGDC, and other contributors

6.1.2 Results from SST Simulation



CHAPTER

SEVEN

CODE LICENSES

7.1 Firmware License for DST Project

Copyright 2019 Ali Aljumaili:

```
/; Permission is hereby granted, free of charge, to any person obtaining a copy of␣
↪this software and associated documentation files (the "Software"), to deal in the␣
↪Software without restriction,
; including without limitation the rights to use, copy, modify, merge, publish,␣
↪distribute, sublicense, and/or sell copies of the Software, and to permit persons␣
↪to whom the Software is furnished to do ; so, subject to the following conditions:
;
; The above copyright notice and this permission notice shall be included in all␣
↪copies or substantial portions of the Software.
;
; THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,␣
↪INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A␣
↪PARTICULAR PURPOSE AND
; NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR␣
↪ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR␣
↪OTHERWISE, ARISING FROM, OUT OF OR
; IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

Copyright Notice for Code Examples from Texas Instruments:

```
/; --COPYRIGHT--,BSD_EX
; Copyright (c) 2012, Texas Instruments Incorporated
; All rights reserved.
;
; Redistribution and use in source and binary forms, with or without
; modification, are permitted provided that the following conditions
; are met:
;
; *  Redistributions of source code must retain the above copyright
;    notice, this list of conditions and the following disclaimer.
;
; *  Redistributions in binary form must reproduce the above copyright
;    notice, this list of conditions and the following disclaimer in the
;    documentation and/or other materials provided with the distribution.
;
; *  Neither the name of Texas Instruments Incorporated nor the names of
;    its contributors may be used to endorse or promote products derived
;    from this software without specific prior written permission.
;
```

(continues on next page)

(continued from previous page)

```
; THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
; AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
; THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
; PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
; CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
; EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
; PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
; WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
; OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
; EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
MSP430 CODE EXAMPLE DISCLAIMER
; ; MSP430 code examples are self -contained low -level programs that typically ;␣
↪demonstrate a single peripheral function or device feature in a highly ; concise␣
↪manner. For this the code may rely on the device 's power -on default ; register␣
↪values and settings such as the clock configuration and care must ; be taken when␣
↪combining code from several examples to avoid potential side ; effects. Also see␣
↪www.ti.com/grace for a GUI - and www.ti.com/msp430ware ; for an API functional␣
↪library -approach to peripheral configuration. ;
;
; --/COPYRIGHT --
```

There are no more guides. You are now guideless. Good luck.

Ali Aljumaili

NTNU
Norwegian University of
Science and Technology