Wave Detection for Ocean Going Robots

Samuel Barton, Miles Drake, Malachi Fisher, Grayson Lewis

Table of Contents

Section 1: Overview

1.1: Executive Summary

The purpose of this project is to create a system to allow the Slocum G3 Glider to detect wave conditions while on the ocean surface, and to make it available to the communications modules inside the glider so that it may also be transmitted back to shore. This was accomplished using a low power microcontroller, memory storage, and accelerometer on a printed circuit board (PCB) mounted in the glider's science bay, and in communication with the glider's central computer. It can detect wave amplitude, direction and period, while using minimal power and maintaining a small form factor.

The Slocum Glider is a type of Autonomous Underwater Vehicle (AUV) designed to monitor underwater conditions such as ocean salinity, temperature, depth, etc. It operates by using ballast tanks and short wings on its sides to propel itself through the water. While in operation it dives and resurfaces, and that process causes water to move over its wings, propelling it forwards. After a predetermined number of glide cycles, the glider resurfaces to transmit the collected data back to shore for analysis. The end-user group that this project is focused on is the College of Earth, Ocean, and Atmospheric Sciences here at Oregon State University which uses these gliders around the world for a number of research projects.

The project is currently in the construction stage. The ECE team has developed the hardware that will be installed on the AUV, the interface between the hardware and the AUV, as well as the firmware that will collect, store and process data. The CS team has developed software to visualize the data as it is transmitted from the AUV, and perform more in-depth data processing when the AUV returns from its mission and the raw data can be extracted from the device.

The implementation of this project uses an STM microcontroller connected to an accelerometer. The microcontroller gathers data from the accelerometer and stores it in an SD card, as well as performing the fourier transform on the data to send it to the main computer onboard the glider. The glider provides the board with a 17V power supply. A buck regulator is used on our board to convert the 17V down to 3.3V.

1.2: Team Contacts, Communication Protocols, and Standards

Table 1.1: Team Member Contact Information

Member Name	**Contact Information**	**Role**	**Responsibilities**
Pat Welch	pat@mousebrains.com	Project Partner	ECE Team Mentor
Kipp Shearman	shearmar@oregonstate.edu	Project Partner	CS Team Mentor
Kai-Fu Chang	chankaif@oregonstate.edu	Project Partner	CS Team Mentor
Grayson Lewis	lewigray@oregonstate.edu	Team Member	Accelerometer Block, PC App Block, Serial interface Firmware Block, XSENS Firmware Block
Miles Drake	drakemil@oregonstate.edu	Team Member	Buck Regulator Block, Main Firmware Block, Power Management Block
Samuel Barton	bartonsa@oregonstate.edu	Team Member	Data Analysis Firmware Block, Main Board PCB, Microcontroller Block
Malachi Fisher	fishemal@oregonstate.edu	Team Member	MicroSD Card Block, MicroSD Card PCB, SD Interface Firmware Block

Table 1.2: Team Member Communication Protocols and Standards

Topic	**Protocol**	**Standard**
Internal Communication	The team will use Discord as the primary means for internal communication and remote meetings.	Communications within the discord will be kept in their appropriate channels. All members are expected to check the discord at least once per day, and participate as necessary.
Partner Communication	Project partner communication will take place primarily via email, but the project partner will also have access to the team Discord channel.	Project partner communication should be kept professional and should be discussed and agreed upon prior to sending. It is not critical who in the group sends the message, as long as everyone has agreed on it.
Task Management	Tasks will be assigned after team discussion and will be tracked via a spreadsheet kept in the team google drive.	Team members are expected to give an accurate assessment of their task project and update the task tracking spreadsheet accordingly.
File Sharing	Shared Files will be kept in a team google drive as well as a box drive, and github	Documentation and shared class assignments will be stored in a google drive for easy access and collaboration. Design files and files provided by the project partner will be stored in a box drive and a github provided by the project partner.
Quality of work	Assignments will be submitted to Canvas.	Work that is submitted as a draft should be submitted on time, and at least 80% complete. Work that is submitted as a final draft should be submitted on time, 100% complete, and the quality of work that one would show to a potential employer.
Time Tracking	Time spent working will be tracked with a spreadsheet kept in the team google drive.	It is expected that team members will update the spreadsheet with the time that they spend working on the project.
Time Management	Time spent working will be focused and on task.	It is expected that team members do not waste each other's or project partners time.

1.2.1 Communication Analysis:

- The Project Partners (Pat Welch, Kai-Fu Chang, and Kipp Shearman) have agreed to join a communal Discord server between the two CS Teams and the ECE Team as a primary form of communication.
- The Project Partners would like to meet weekly for progress updates from each of the teams. Though Discord is the primary form of communication, the weekly meetings will be held on Zoom for ease of use for each of the parties.

1.3. Gap Analysis

1.3.1 Current Limitations and our Solution

As it stands the Slocum ocean gliders do not have nine degrees-of-freedom inertial measurement units that contain accelerometers, magnetometers, and gyroscopes to detect wave conditions. Instead, researchers around the world rely on fixed buoys to monitor wave conditions, which are expensive, immobile, and are often spaced many miles apart, creating a very low resolution idea of current wave conditions in a given area. By creating a way to retrofit the gliders with nine degrees-of-freedom inertial measurement unit that contains accelerometers, magnetometers, and gyroscopes, the College of Earth, Ocean, and Atmospheric Sciences will be able to quickly deploy wave monitors to whatever areas they wish to ascertain swell height, period, and direction and immediately view the results.

1.3.2 Expected End User

This project will be used by researchers and deployed on a specialized vehicle. Our end user is someone who works in a research facility and already has some Slocum gliders laying around but has no way to use them to measure waves. With our project, they can retrofit their glider with an accelerometer board and be well on their way to capturing wave data. Because this project will be installed on the inside of a glider, it can be delivered as a bare printed-circuit board. We can assume that the end user will have a technical background, and can safely handle the device to prevent damage due to electrostatic discharge, and that we will not be responsible for preventing damage due to water or particle ingress. Ocean gliders are expensive to operate, and are typically deployed for up to three months, so it is important to the stakeholders that the system operates reliably during that period.

1.3.3 User Story

The end user will receive the completed system and access to the project wiki. The user will then install the system in a SLOCUM G3 ocean glider. When the glider is deployed in the ocean, the user will command the glider to surface several times per day. When it is at the surface, the glider’s computer will power the system on and float at the surface for 20 minutes while the system collects and processes wave data. When the system has

completed a data collection cycle, it will transmit the calculated wave parameters to the glider's computer and indicate that it has completed the data analysis. The glider can then send the wave parameters to shore wirelessly. Once the glider has completed its deployment, it is retrieved from the ocean, and can be opened to retrieve the system's SD card which contains all raw data that was collected. The user can then take this data and perform a more thorough analysis.

1.4: Project Timeline

			Fall Term 1 2 3 4 5 6 7 8 9 10 F	Winter Term 1 2 3 4 5 6 7 8 9 10 F	Spring Term 1 2 3 4 5 6 7 8 9 10 F	
Tasks:	**Task / Responsible Party**					**REQUIRED RESCOURCE**
Design	1) Specification of Requirements	GL				NONE
	2) Component Selection	MD				SYSTEM REQUIREMENTS
	3) PCB Schematic	MF				WORKING DESIGN
	4) Windows App for Testing	SB				NONE
	5) PCB Layout	GL				FINISHED SCHEM, SIZE REQ.
	6) PCB Fabrication and Construction	MD				PCB DESIGN AND LAYOUT
	7) Firmware	MF				SYSTEM REQUIREMENTS
	8) Creation of Test Glider	MF				WORKING PCB
Testing	9) Benchtop Testing	SB				WORKING PCB
	10) Document Wiki	MD				NEAR FINISHED SYSTEM
And	11) Buoyancy Tank Testing	GL				WORKING TEST GLIDER
Revision	12) Wave Lab Testing	MD				WORKING TEST GLIDER
	13) Small Wave Marine Enviroment Testing	MF				WORKING TEST GLIDER
	14) Near Shore Ocean Enviroment Testing	SB	-> Critical Path			NEAR FINISHED SYSTEM
	15) Near Buoy Ocean Enviroment Testing	GL				NEAR FINISHED SYSTEM

Figure 1.1: Project Timeline

1.5: References and File Links

1.5.1 References

1.5.2 File Links

Project Timeline:
https://docs.google.com/spreadsheets/d/1Lr86H4AFnyPC4HZCUiYSTr-dyl6JOHirsw4xZEpULac/edit?usp=sharing

Xsens MTi-3 Product Sheet:
https://www.xsens.com/hubfs/Downloads/Leaflets/MTi-3.pdf?hsCtaTracking=8b4e849f-c2cb-4ed7-847a-82a9c3524e46%7Cd8830f31-7f54-48f5-99ec-74cf5dc160dc

1.6: Revision Table

Table 1.3: Section 1 Revision Table

Date:	**Action:**
10/20/21	Malachi drafted an executive summary
10/21/21	Malachi updated Team Communication Protocols and Standards; added project timeline
10/21/21	Grayson added state of project to executive summary, other minor edits.
10/29/21	Sam updated Section 1 to be better in line with the instructor feedback on Section 1.
10/29/21	Miles updated Section 1 to be better in line with the instructor feedback on Section 1.
10/29/21	Grayson updated Section 1 to be better in line with the instructor feedback on Section 1.
10/29/21	Malachi updated Section 1 to be better in line with the instructor feedback on Section 1.
11/10/21	Sam updated formatting, project timeline and executive summary per peer review feedback.
11/10/21	Malachi updated formatting, project timeline and executive summary per peer review feedback.

11/10/21	Miles updated formatting, project timeline and executive summary per peer review feedback.
11/10/21	Grayson updated formatting, project timeline and executive summary per peer review feedback.
11/19/21	Malachi Updated Timeline, Responsibilities
11/19/21	Sam Updated Responsibilities
12/3/2021	Miles updated section 1 based on peer feedback

Section 2: Requirements, Impacts and Risks

2.1 Requirements

2.1.1 Data Collection and Analysis

- *Project Partner Requirement (PPR)*: Build a system that analyzes swell data for 20 minutes and sends it to the main science computer in the glider.
- *Engineering Requirement (ER)*: The system shall sample data at a maximum frequency of 10 Hz (though lower frequencies may be desired) for up to 20 minutes, and analyze the data collected to calculate the following parameters:
    a) Hs, significant wave height in meters
    b) Dom_period, dominant wave period in seconds
    c) wave_dir, wave direction, magnetic
    d) Hmax, maximum wave height in meters
    e) Hmax2, second highest wave height
    f) Pmax, maximum period
    g) A1, spectral parameters
    h) B1, spectral parameter
    i) A2, spectral parameter
    j) B2, spectral parameter index,
    k) Wave component number

  Note: The project partners may decide that some of these parameters are unnecessary and not require their calculation for the final system.

- *Verification Process (Black Box)*:
    1) Power system and hook up to PC Application built for testing.
    2) Run the system for 20 minutes.
    3) Save the parameters calculated by the system
    4) Ensure that there are at least 12,000 data samples measured and the last time stamp ensures that 20 minutes have elapsed.

5) Remove SD card from system and connect to PC
6) Use a script that has been approved by the project partners to manually calculate the parameters from the raw data that was collected.
7) Compare the parameters calculated by the system, and the parameters calculated by the script. This requirement will be met if the two calculations differ by no more than 5% for each parameter.

2.1.2 PCB Size

- *PPR*: Board must fit inside the science bay of the glider.
- *ER*: The microcontroller PCB must be smaller than 7 inches by 1.5 inches (and 2 inches tall) to fit onto the back of the acoustic modem board in the science bay of the glider.
- *Verification Process (Black Box)*:
    1) Measure length of PCB.
    2) Measure width of PCB.
    3) Measure height of PCB.
    4) Ensure that measurements are smaller than requirements.

2.1.3 Removable Data Storage

- *PPR*: Collected data must be available in removable memory.
- *ER*: Data must be stored to a micro SD card.
- *Verification Process (White Box)*:
    1) Collect data for 10 minutes
    2) Remove memory card
    3) Check if the memory card has 6,000 samples.

2.1.4 Glider Battery Life Impacts

- *PPR*: Board must not reduce gliders battery life by more than 1 percent.
- *ER*: For a 20 minute collection cycle, the system must consume less than 230 Joules
- *Verification Process (Black Box)*:
    1) Run a full 20 minute data collection cycle.
    2) Measure the average voltage and current on the input of the module.
    3) Use this data to calculate average power over the 20 minutes.
    4) Integrate it to get the number of Joules.
    5) The number of Joules should be less than 230.

2.1.5 Data Ease of Access

- *PPR*: Removable Memory must be easily accessible
- ER: SD card and associated PCB will have at least 0.5 inches of clearance from nearby military style screw connector and 0.25 inches of clearance from nearby white snap type connectors.
- *Verification Process (Black Box)*:
    1) Install SD card and associated PCB into the glider see Figure 2.1 below.

2) Measure the distance to the connectors.
3) This will be successful if the measurements are less than or equal to 0.5 inches from the military style screw connector and 0.25 inches from the white snap type connectors. .

2.1.6 Project Overview Wiki Page

- *PPR*: The project must be thoroughly documented
- ER: A wiki page shall be created that includes: - A user guide - Firmware - Altium Designer files - Electrical Schematic(s) - PCB Fabrication files (gerbers) - Electrical Bill of Materials.
- *Verification Process (White Box)*:
    1) Write the initial draft of the Wiki Page and submit to the Project Partners by the Friday of Week 9 Winter 2022
    2) Receive feedback from the Project Partners by the Friday of Week 10 Winter 2022
    3) Write the second draft of the Wiki Page and submit to the Project Partners by the Friday of Week 1 Spring 2022
    4) Receive feedback from the Project Partners by the Friday of Week 2 Spring 2022
    5) Write the final draft of the Wiki Page and submit to the Project Partners by the Friday of Week 3 Spring 2022
    6) If the project Partners find our Wiki Page sufficient, have each of them sign the final copy and return the document by Friday of Week 4 Spring 2022

2.1.7 System Longevity

- *PPR*: There should be no hardware or firmware bugs that cause the system to stop logging data while in use.
- *ER*: System must work properly for at least 90 days of continual use.
- *Verification Process (White box)*:
  During a 90 day period, the system will be cycled 1080 times, a cycle consisting of the system being powered on, collecting data for 20 minutes, processing/storing the collected data, and then being powered off. To demonstrate the system will not encounter any errors in this 90 day period, we will cycle it 206 times, with a reduced data collection time.
    1) For a 24 hour window, cycle 5 minutes of data capture and 2 minutes with the system powered off.
    2) The system shall not encounter any hardware malfunctions or firmware errors.
    3) All data collected will be stored on the SD card, and shall not contain any errors.

2.1.8 System Security

- *PPR*: System must be rugged to withstand the forces associated with being put into and taken out of the ocean.

- ER: The system shall function properly after a 10 foot drop. During the drop test the system shall be mounted in a test enclosure, and the system shall be powered off during the drop test.
- *Verification Process (Black Box)*:
    1) Enclose the system in the test enclosure.
    2) Drop enclosure from 10 ft above the surface of a water tank (specific tank TBD).
    3) Remove system from test enclosure.
    4) Connect system to power source.
    5) Run the system for 5 minutes to ensure all 13 parameters listed above are being transmitted and there are at least 3000 data points measured and stored on the SD card.

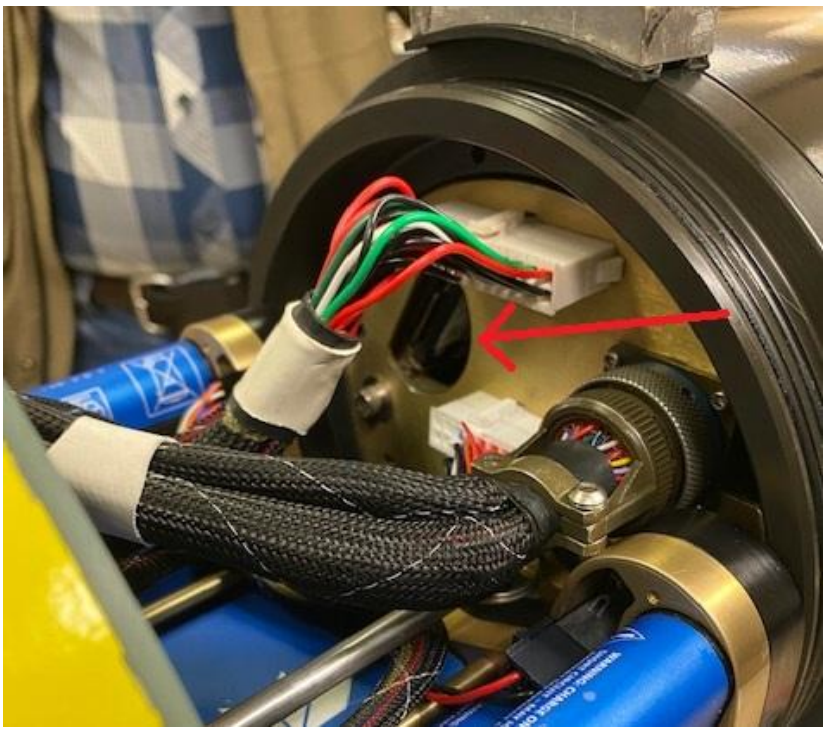

Figure 2.1: SD card retrieval hole in glider science bay.

2.2 Design Impact Statement

This section is intentionally left blank.

2.3 Risks

Table 2.1: Risk Assessment and Action Plans

Risk ID	Risk Description	Risk Category	Risk Prob-ability	Risk Impact	Performance Indicator	Respon sible Party	Action Plan
R1	Don't receive gliders back from Teledyne due to shipping issues	Timeline	10%	L	Shipping delays due to COVID etc	Samuel Barton	Retain
R2	Component shortages	Timeline	100%	H	Available stock of desired components	Grayson Lewis	Reduce
R3	ICs destroyed by static in delivery	Technical	5%	H	Chips don't work	Miles Drake	Avoid
R4	PCB flaws in manufacturing	Technical	25%	M	PCBs are visually different than designed	Malachi Fisher	Avoid
R5	Teammate Sickness (COVID, etc.)	Timeline	80%	H	Recent exposure to COVID positive individuals	Samuel Barton	Reduce
R6	Test enclosure breaks during testing (drop test, buoyancy test)	Timeline	5%	H	Water damage to board	Miles Drake	Retain
R7	Glider Sinks During testing; board is lost	Timeline	1%	H	Glider is not recovered.	Malachi Fisher	Retain
R8	Computer with firmware/code is lost/destroyed	Timeline	2%	H	Can't turn on or find the computer that has firmware.	Grayson Lewis	Avoid

2.4 References and File Links

2.4.1 References

2.4.2 File Links

2.5 Revision Table

Table 2.2: Section 2 Revision Table

Date	**Action**
4/21/22	Updated wording of engineering requirements 2.1.6 and 2.1.8 for clarity and feasibility, with project partner's permission.
3/13/2022	Updated Data Collection and Analysis requirement at the suggestion of our project partners.
10/13/2022	Updated data ease of access requirement with the approval of our project partners.
10/29/2021	Grayson worked on the initial draft for section 2.1, 2.2, and 2.3 of the project document.
10/29/2021	Malachi worked on the initial draft for section 2.1, 2.2, and 2.3 of the project document.
10/29/2021	Sam worked on the initial draft for section 2.1, 2.2, and 2.3 of the project document.
10/29/2021	Miles worked on the initial draft for section 2.1, 2.2, and 2.3 of the project document.
10/11/2021	Grayson, minor revisions
10/11/2021	Malachi, General reformatting. Revised project risks to be more in line with instructor feedback.
10/11/2021	Sam revised the project requirements to reflect feedback given by peers/instructors
10/11/2021	Miles revised the action plans for the risk table.
10/12/2021	Grayson finalized Requirements after team meeting
11/19/2021	Sam revised the project requirements and risks to reflect feedback given by instructors
11/19/2021	Miles revised the risks and project requirements according to instructor

	feedback
12/3/2021	Miles revised requirement 2.1.3
12/3/2021	Sam revised requirement 2.1.6
12/3/2021	Grayson revised requirement 2.1.1

Section 3: Top Level Architecture

3.1 Block Diagram

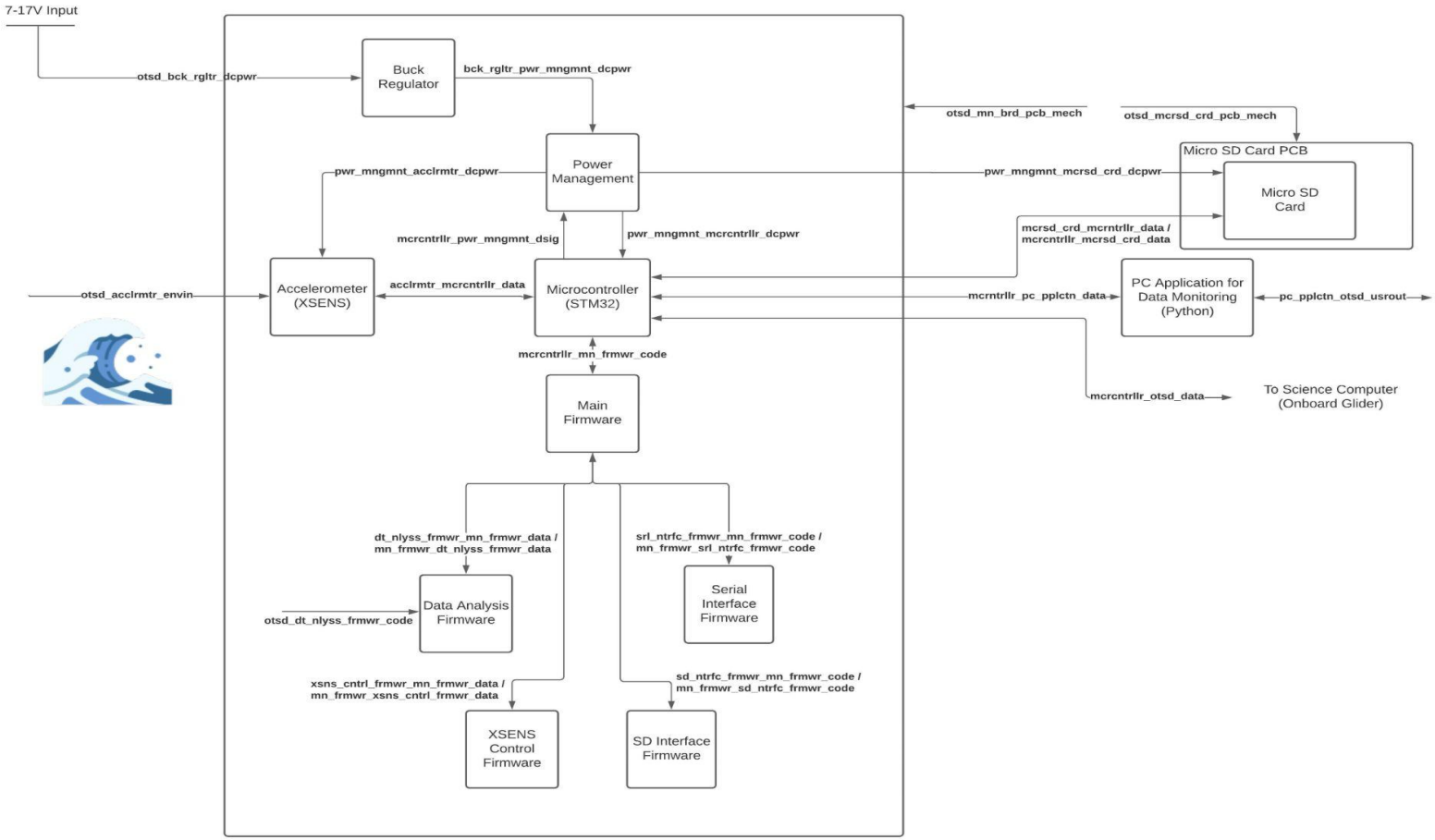

Figure 3.1: Draft Block Diagram for our System

3.2 Block Descriptions

Table 3.1: Block Descriptions

Name	Description
Buck Regulator Champion: Miles Drake	This block includes the buck regulator power supply for the main board PCB. The buck regulator block is a necessary part of the system. The Slocum glider is able to supply between 7 and 17 volts to our board, but the microcontroller that we are using needs a 3.3V supply. We will be designing a buck regulator power supply for our board using the TPS54232DR. Since this is the same chip that is used for the tech demo, we'll use the tech demo board to prove its functionality before implementing it in the final PCB.
Power Management Champion: Miles Drake	This block includes a power management circuit to cut power to the SD card and the accelerometer. Since the SD card uses a considerable amount of power while it is in sleep mode, we would like to be able to cut its power line while it is not being written to or read from. For this we will need a simple MOSFET circuit that uses the output of a GPIO pin on the microcontroller to control the flow of power to the SD card. We will use an identical circuit to cut the power to the accelerometer when we aren't using it (while performing the fourier transform). We plan to model the circuit in LTSpice before implementing it in the final PCB.
Accelerometer Champion: Grayson Lewis	This block encompasses the code that will be on the STM32 microcontroller used to interface with the XSENS Accelerometer chip. It will handle the initial handshake between the accelerometer and the microcontroller, and handle the transfer of commands from the microcontroller to the accelerometer as well as the transfer of data from the accelerometer to the microcontroller. This block is necessary because without it we would be unable to retrieve the data from our accelerometer.
Microcontroller Champion: Sam Barton	This block includes the schematic for the STM32 microcontroller. We need to create a symbol and footprint for the microcontroller. Then we need to place the symbol in the schematic and put down the clock crystal and all of the necessary capacitors and resistors. We plan to use the datasheet for the microcontroller to find out the recommended implementation.
Main Firmware Champion: Miles Drake	This block includes the code that will run on the STM32 microcontroller to perform the memory management functions. This block is necessary because there are memory management tasks that do not fit into one of the other firmware blocks.

XSENS Control Firmware Champion: Grayson Lewis	This block encompasses the code that will be on the STM32 microcontroller used to interface with the XSENS Accelerometer chip. It will handle the initial handshake between the accelerometer and the microcontroller, and handle the transfer of commands from the microcontroller to the accelerometer as well as the transfer of data from the accelerometer to the microcontroller. This block is necessary because without it we would be unable to retrieve the data from our accelerometer.
SD Interface Firmware Champion: Malachi Fisher	This block will take raw data from the microcontroller and convert it to a format that can be sent to the SD card and stored. Without it, we would be unable to store data to the SD card. Additionally, this block will facilitate reading from the SD card and converting it into data usable by the Microcontroller.
Serial Interface Firmware Champion: Grayson Lewis	This block configures, and uses the hardware on the STM32 microcontroller for serial communication with the ocean glider, and serial communication with the PC application. It is expected that this will configure one or more USART modules, convert information from the main firmware into serial packets, and convert received serial packets into data to return to the main firmware.
PC Application Champion: Grayson Lewis	This is a python based PC application that will be used to communicate with the system serially, and show the user the status of the system. This block is designed to aid development by allowing the system to be connected to a PC, and display information on the PC, such as the orientation of the accelerometer, and overall status of the system.
MicroSD Card Champion: Malachi Fisher	The SD card will be the removable memory device for the system. It will be used to store the data from the accelerometer until it can be processed using a fourier transform. The SD card can also be removed and the data can be accessed with a PC program for further analysis.
MicroSD Card PCB Champion: Malachi Fisher	This block is the PCB that the SD card will be mounted to. Without it, the SD card will not be able to be properly secured to the inside of the glider. It will need to be small enough to fit between the connectors on the wall of the glider bay without compromising the ability to remove the connectors from their sockets.
Main Board PCB Champion: Sam Barton	This block encompasses the main PCB we will mount the system onto. This will then be mounted onto the back of the acoustic modem in the science bay of the glider. This block is necessary because without it the system would not be secure in the glider and would not be able to be mounted properly. The PBC will also have connectors that will be used to supply the system with power, to communicate with the science computer

	onboard the glider, and send data out to the external SD card.
Data Analysis Firmware Champion: Sam Barton	This block entails analyzing swell data from the accelerometer including glider heading, significant wave height (meters), wave period (seconds), wave direction, maximum wave height, second highest wave height, maximum period, and spectral parameters for the Fourier Transform.

3.3 Interface Definitions

Table 3.2: Interface Definitions

Name	Properties
otsd_bck_rgltr_dcpwr	● **Inominal:** 30mA ● **Ipeak:** 80mA ● **Vmax:** 17V ● **Vmin:** 3.5 ● **Vnominal:** 7V
otsd_acclrmtr_envin	● **Electromagnetic:** Earth's Magnetic Field X, Y, Z ● **Other:** Acceleration Nominal: +/- 1G in any axis ● **Other:** Roll, Pitch, Yaw ● **Other:** Acceleration: X, Y, Z
otsd_mcrsd_crd_pcb_mech	● **Fasteners:** 10 pin ribbon cable connector **Fasteners:** Mounting Hardware, metal screws with standoff ● **Fasteners:** SD Card Socket
otsd_mn_brd_pcb_mech	● **Fasteners:** Four screws on the four corners of the PCB. ● **Fasteners:** Mounting screws to attach PCB to the metal plates in the payload bay of the glider. ● **Pulling Force:** The PCB must stay connected in the payload bay under a dropping force of 5G's
otsd_dt_nlyss_frmwr_code	● **Other:** Multiple functions on the MCU to calculate

	the significant wave height (m), dominant period (s), wave direction (degrees magnetic), maximum wave height (m), second highest wave height (m), maximum period (s), four spectral parameters, and a wave component number (if needed). ● **Other:** Language: C/C++
bck_rgltr_pwr_mngmnt_dcpwr	● **Inominal:** 50mA ● **Ipeak:** 140mA ● **Vmax:** 3.4V ● **Vmin:** 3.2V ● **Vnominal:** 3.3V
pwr_mngmnt_acclrmtr_dcpwr	● **Inominal:** 20mA ● **Ipeak:** 25mA ● **Vmax:** 3.4V ● **Vmin:** 3.2V ● **Vnominal:** 3.3V
pwr_mngmnt_mcrcntrllr_dcpwr	● **Inominal:** 11mA ● **Ipeak:** 14mA ● **Vmax:** 3.4V ● **Vmin:** 3.2V ● **Vnominal:** 3.3V
pwr_mngmnt_mcrsd_crd_dcpwr	● **Inominal:** 20mA ● **Ipeak:** 100mA ● **Vmax:** 3.4V ● **Vmin:** 3.2V ● **Vnominal:** 3.3V
acclrmtr_mcrcntrllr_data	● **Messages:** Cartesian (XYZ) Acceleration, 3x 32-bit floating point numbers ● **Messages:** Cartesian Magnetometer Measurement: 3x 32-bit floating point numbers ● **Messages:** Rotation in Quaternions: 4x 32-bit floating point numbers ● **Protocol:** SPI

mcrcntrllr_otsd_data	● **Datarate:** 115200 Baud ● **Messages:** Significant wave height (m), dominant period (s), wave direction (degrees magnetic), maximum wave height (m), second highest wave height (m), maximum period (s), four spectral parameters, a wave component number (if needed), and a timestamp (UTC). ● **Protocol:** RS232 to science computer using level shifters
mcrcntrllr_pwr_mngmnt_dsig	● **Logic-Level:** 3.3V ● **Other:** Active High
mcrcntrllr_mn_frmwr_code	● **Other:** Purpose: Data Management ● **Other:** Language: C
mcrcntrllr_pc_pplctn__data	● **Datarate:** 115200 Baud ● **Datarate:** System Status: (what information will comprise 'status' is tbd) ● **Messages:** Cartesian (XYZ) Acceleration ● **Messages:** Roll, Pitch, Yaw orientation in quaternions ● **Protocol:** USB
mcrcntrllr_mcrsd_crd_data	● **Datarate:** 25MB/s ● **Protocol:** SD protocol
mn_frmwr_xsns_cntrl_frmwr_data	● **Messages:** Reset XSENS ● **Messages:** Configure XSENS ● **Messages:** Get Measurements
mn_frmwr_sd_ntrfc_frmwr_code	● **Other:** Data Transfer: To SD Card ● **Other:** Data Format: Array
mn_frmwr_srl_ntrfc_frmwr_code	● **Other:** UART Selection: Which UART interface should be used to transmit.

	• **Other:** UART Parameters: baud rate, word length, parity bits, stop bits. Or other necessary to correctly configure UART module. • **Other:** Data: Numbers and strings to be transmitted via a UART Module
mn_frmwr_dt_nlyss_frmwr_data	• **Messages:** Significant wave height (m), dominant period (s), wave direction (degrees magnetic), maximum wave height (m), second highest wave height (m), maximum period (s), four spectral parameters, a wave component number (if needed), and a timestamp (UTC). • **Other:** This data transfer is internal to the firmware inside the STM32 MCU. The data will be transferred between the MCU's program memory (RAM) and this firmware block, and then sent back to the MCU's RAM.
xsns_cntrl_frmwr_mn_frmwr_data	• **Messages:** Rotation in Quaternions: 4x 32-bit floating point numbers • **Messages:** Cartesian (XYZ) Acceleration: 3x 32-bit floating point numbers • **Messages:** Cartesian Magnetometer Measurement: 3x 32-bit floating point numbers
sd_ntrfc_frmwr_mn_frmwr_code	• **Other:** Integers from SD card storage to be used for Fourier Transform, Radio to shore. • **Other:** Data Format: Array
srl_ntrfc_frmwr_mn_frmwr_code	• **Other:** Data: Numbers and strings received from UART modules • **Other:** Connection Status: Which UART devices are connected to external devices
pc_pplctn__otsd_usrout	• **Type:** 3D visualization of accelerometer orientation • **Type:** Text display of system status • **Usability:** Any of the team members must be able to connect the system to a PC, open the application, and determine the system status.

mcrsd_crd_mcrcntrllr_data	• **Datarate:** 25MB/s • **Protocol:** SD protocol
dt_nlyss_frmwr_mn_frmwr_data	• **Messages:** Significant wave height (m), dominant period (s), wave direction (degrees magnetic), maximum wave height (m), second highest wave height (m), maximum period (s), four spectral parameters, a wave component number (if needed), and a timestamp (UTC). • **Other:** This data transfer is internal to the firmware inside the STM32 MCU. The data will be transferred between the MCU's program memory (RAM) and this firmware block, and then sent back to the MCU's RAM.

3.4 References and File Links

3.4.1 References

XSENS MTI-3 Sensor Specifications
[1] XSENS, “MTI-1 Series Sensor Specifications.” [Online] Available: https://mtidocs.xsens.com/sensor-specifications

TI-TPS54232DR Switching Regulator Datasheet
[2] Texas Instruments, “TPS54232 2-A, 28-V, 1 MHz, Step-Down Converter with Eco Mode” [Online] Available: https://www.ti.com/store/ti/en/p/product/?p=TPS54232DR

3.4.2 File Links

LucidChart Block Diagram
https://lucid.app/lucidchart/ead89e61-e7d5-48b2-860d-9d0f5be6ddcd/edit?from_docslist=true&invitationId=inv_aa010d0b-06fa-4156-a60e-a370e72956ad&page=0_0#

3.5 Revision Table

Table 3.3: Revision Table for Section 3

Date	Action
11/19/2021	Miles created the section headings and the revision table and wrote up some block descriptions.
11/19/2021	Malachi Inserted Block and interface tables from the block diagram tool
11/19/2021	Sam created blocks and interface definitions for the Microcontroller, main PCB, and Data Analysis Firmware blocks.
12/2/2021	Malachi updated the interface definitions related to communicating with the SD Card, updated table.
12/3/2021	Grayson updated interface definitions for Accelerometer, PC Application, XSENS Control Firmware, and Serial Interface Firmware blocks.
12/3/2021	Miles updated interface definitions for his blocks (Power management, Buck Regulator, Main firmware).
12/3/2021	Sam updated interface definitions for the Microcontroller block, Main board PCB block, and Data Analysis Firmware block

Section 4: Block Validations

4.1 MicroSD Card PCB

4.1.1 Description

This block is the PCB that the microSD card will be mounted to. Without it, the microSD card will not be able to be properly secured to the inside of the glider. It will need to be small enough to be securely mounted between the connectors on the wall of the glider bay without compromising the ability to remove the connectors from their sockets. The microSD card must be held firmly in place in a microSD card socket, preferably a hinged type, so that microSD card will remain secure throughout the deployment cycle of the glider. Additionally, the board will need a cable with a connector to interface with the main PCB located elsewhere in the glider.

4.1.2 Design

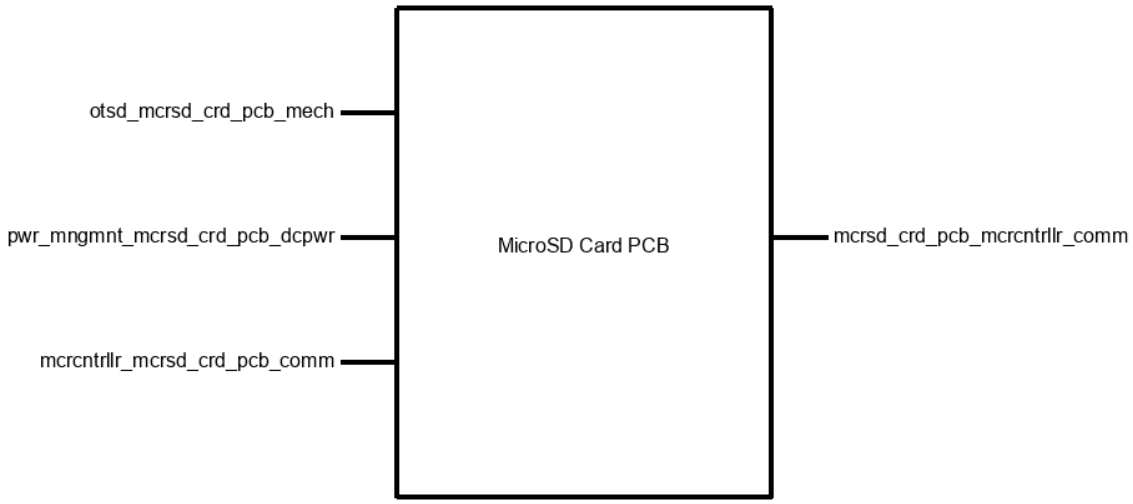

Figure 4.1.2.1: Black Box Diagram of MicroSD Card PCB

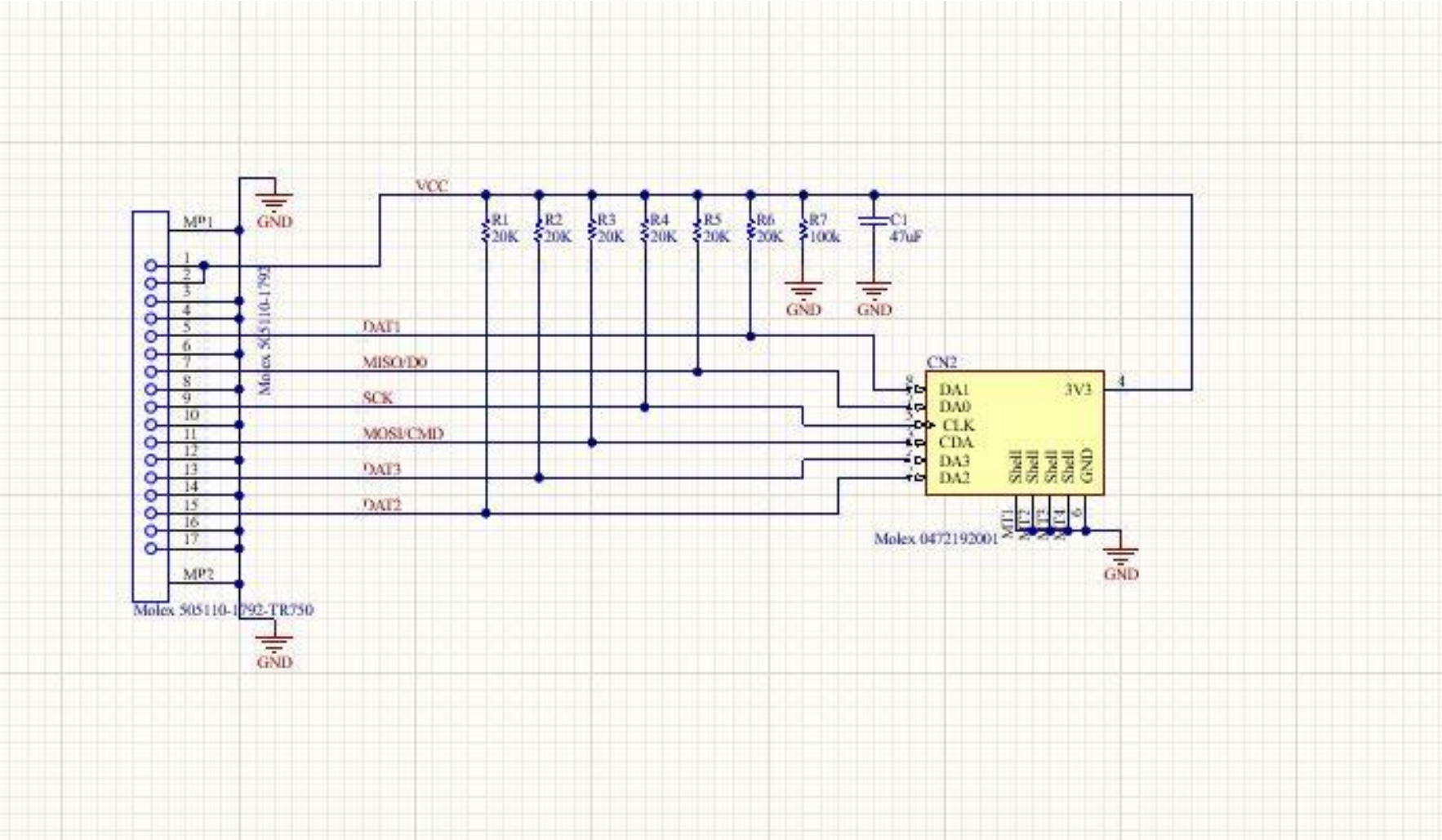

Figure 4.1.2.2: Schematic for MicroSD Card PCB

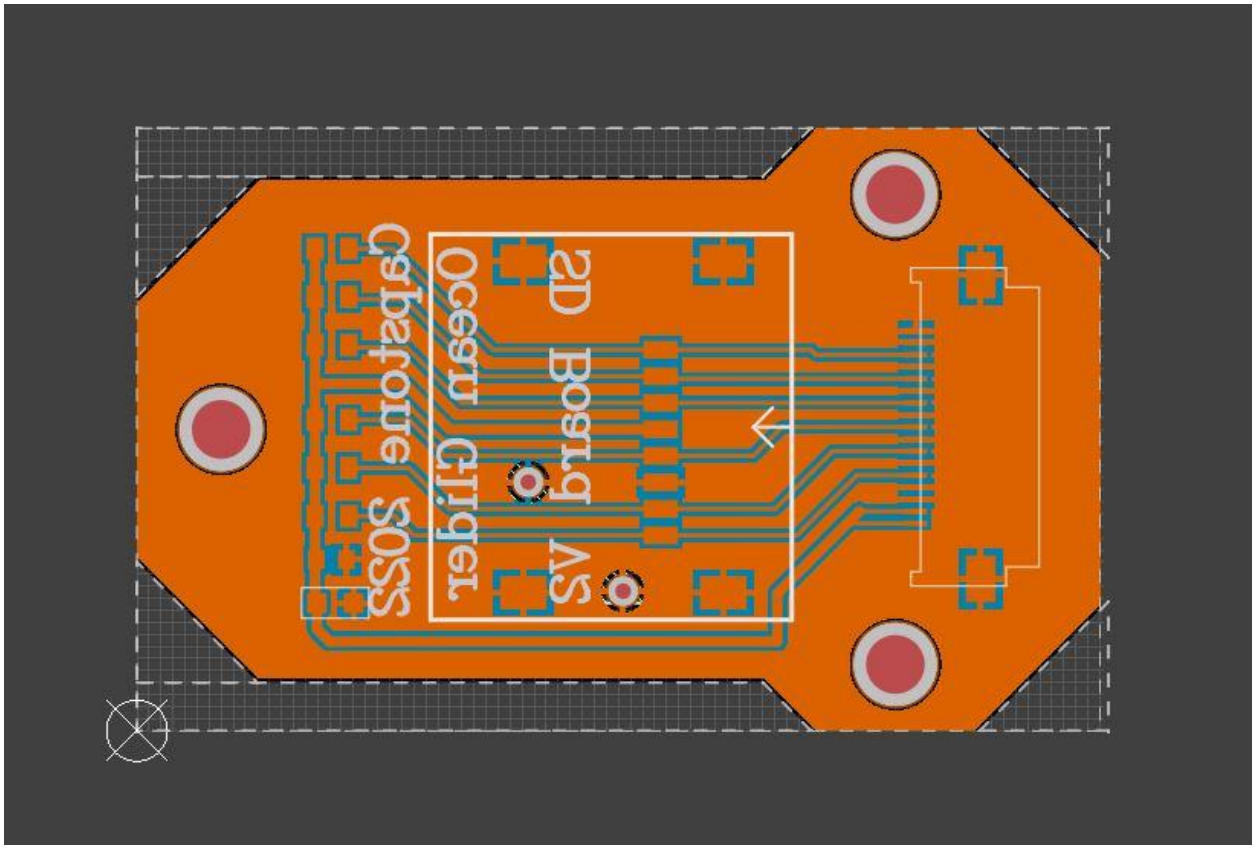

Figure 4.1.2.3: Important Dimensions for the MicroSD Card PCB

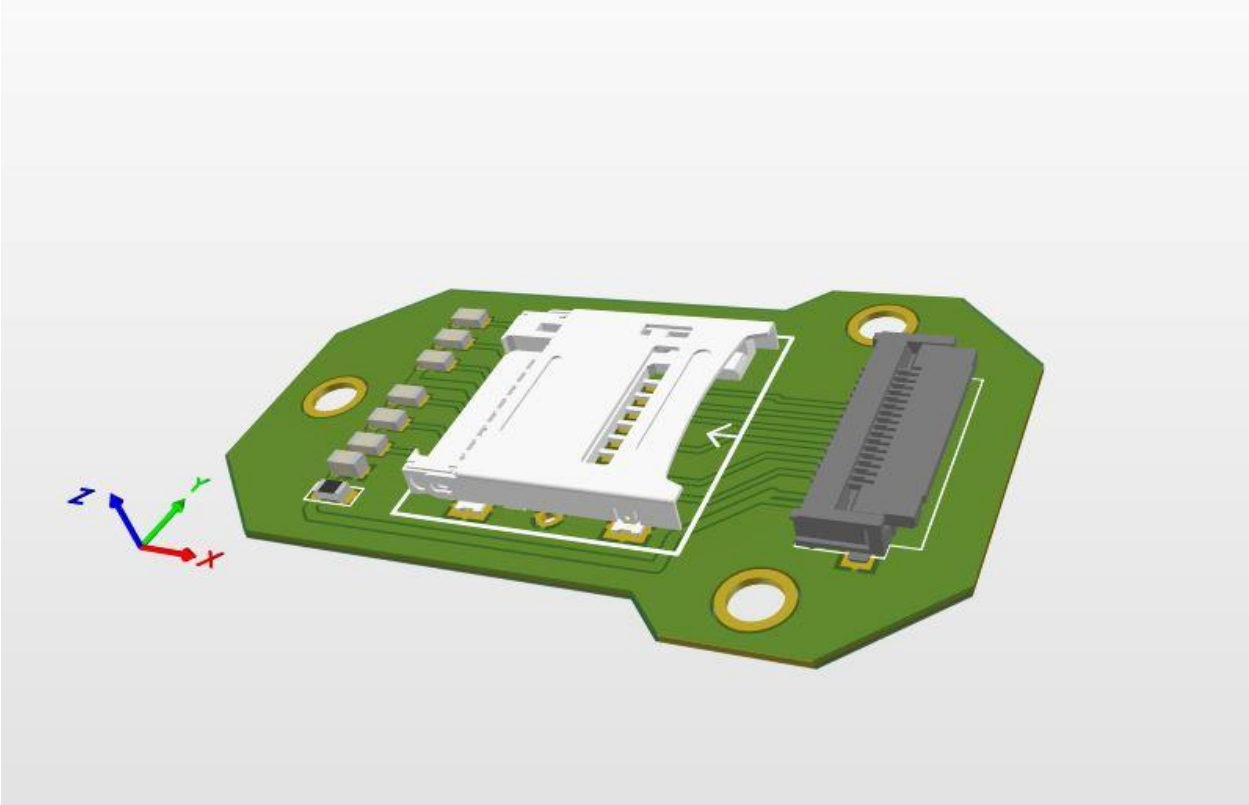

Figure 4.1.2.4: 3D Mockup of finished microSD card PCB

4.1.3 General Validation

The point of this block is to provide a way to access certain parts of the system (namely the microSD card) without completely disassembling the glider. It must be securely mounted to the internals of the glider and must securely hold the microSD card and the attached cable. It must also provide the additional circuitry required to properly operate the microSD card (pull-up resistors, capacitors, etc.).

As a backup and for the purposes of testing this board I have developed a second board that accepts the same style of connector used to connect the microSD card PCB to the main PCB and is equipped with a set of female headers such that it will be possible to connect an STM32 Nucleo board to it which can then be connected to the microSD board via the ribbon cable that will be used in the final assembly. However, if a problem arises with the design of the microSD board, this board can be used as a stand-in. the female headers also will accept a premade microSD card module which could be used in place of the primary microSD card PCB.

4.1.4 Interface Validation

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?

otsd_mcrsd_crd_pcb_mech : Input

Fasteners: 17 pin ribbon cable connector	A typical microSD card has 8 pins and 8 discrete signals. By using a 17 pin ribbon cable/connector to connect the microSD card PCB to the main PCB, we are sure to have enough signals to interface with the card with enough conductors to provide shielding between the signal wires.	8 signals are needed to communicate with a microSD card using the 4-bit bus, so a 17 conductor cable will be more than adequate. [1]
Fasteners: Mounting Hardware, metal screws with standoff	Metal screws and standoffs will be provided to ensure solid mounting of the PCB. A minimum of 3 mounting points will be required to ensure that the PCB will remain on a specified plane relative to the internal wall of the glider where the board will be mounted.	You need a minimum of 3 points to define a plane, so to ensure that our board is mounted on a specified plane 3 mounting points are required. [2]
Fasteners: SD Card Socket	A microSD card was required by the end user as the means to store the recorded Data, so to mount the card we will need an SD card socket.	microSD card sockets are specifically made to securely hold microSD cards.

pwr_mngmnt_mcrsd_crd_pcb_dcpwr : Input

Inominal: 20mA	This will be more than enough current to power the microSD card in standby state.	A typical microSD card consumes less than a milliamp of current in standby mode [1]. Having 20mA available will be

		more than sufficient to operate the microSD in standby mode.
Ipeak: 100mA	This should be enough current to operate the microSD during area/write operation.	While the microSD card specification allows for a card to consume 200mA and above during a read or write operation, a typical microSD card operating at less than 25Mhz will generally consume less than 100mA [3].
Vmax: 3.4	This is the highest voltage that the power management block will be providing.	This is well within the allowable operating voltage or a microSD card as defined by the microSD card specification, of 2.7-3.6V [3].
Vmin: 3.2	This is the lowest voltage that the power management block will be supplying.	This is well within the allowable operating voltage or a microSD card as defined by the microSD card specification, of 2.7-3.6V [3].
Vnominal: 3.3	This is the nominal voltage that the power management block will be supplying.	This is normal operating voltage of a microSD card [3].

mcrcntrllr_mcrsd_crd_pcb_comm : Input

Other: Pullup Resistors on Comm Lines	This is necessary to pull the data line into a high state when not being driven low by the microSD or the microcontroller.	According to the microSD card Specification, this is the typical way to prevent floating pins on the SD bus [3].
Protocol: SD Protocol	SD protocol is faster to write to the microSD than SPI.	The microSD card specification explicitly states that using the microSD card in SD mode comes at the cost of performance [3].

Vmax: 3.4V	This is the maximum voltage that will be supplied by the power management block, and therefore the highest voltage our signal wire can be at.	This is well within the allowable operating voltage or a microSD card as defined by the microSD card specification, of 2.7-3.6V [3].
Vnominal: 3.3V	This is the nominal voltage that will be supplied by the power management block and will therefore be the nominal ‘high’ voltage of the microSD data bus.	This is normal operating voltage of a microSD card [3].

mcrsd_crd_pcb_mcrcntrllr_comm : Output

Other: Pullup Resistors on Comm Lines	This is necessary to pull the data line into a high state when not being driven low by the microSD or the microcontroller.	According to the microSD card Specification, this is the typical way to prevent floating pins on the SD bus [3].
Protocol: SD Protocol	SD protocol is faster to write to the microSD than SPI.	The microSD card specification explicitly states that using the microSD card in SD mode comes at the cost of performance [3].
Vmax: 3.4V	This is the maximum voltage that will be supplied by the power management block, and therefore the highest voltage our signal wire can be at.	This is well within the allowable operating voltage or a microSD card as defined by the microSD card specification, of 2.7-3.6V [3].
Vnominal: 3.3V	This is the nominal voltage that will be supplied by the power management block and will therefore be the nominal ‘high’ voltage of the microSD data bus.	This is normal operating voltage of a microSD card [3].

4.1.5 Verification Process

otsd_mcrsd_crd_pcb_mech

1. **17 pin ribbon cable and fasteners**

    1.1. Connect the ribbon cable and fasteners

    1.2. Ensure that the connectors are firmly connected by tugging on them gently.

2. **Mounting Hardware, Metal screws with standoffs**

    2.1. Attatch the metal screws with and standoffs through the mounting holes of the PCB to ensure proper hole sizing.

    2.2. Mount the standoffs to a metal plate to simulate the interior wall of the glider, ensure that the board is solidly mounted.

3. **MicroSD card Socket**

    3.1. Ensure that the microSD card is secure when inserted into the microSD card socket.

pwr_mngmnt_mcrsd_crd_pcb_dcpwr : Input

4. **Inominal: 20mA**

    4.1. Connect the PCB with the microSD card to 3.3V DC through an ammeter.

    4.2. Initialize the SD card using a test program on a STM32 Nucleo Board.

    4.3. Ensure that after initialization, while idle the current on the DC power bus does not exceed 20mA.

5. **Ipeak 100mA**

    5.1. With the PCB still connected to 3.3V DC through an ammeter and still initialized by the STM32 Nucleo board, preform a write operation and a read operation to the microSD card.

    5.2. Ensure that at no point does the current to the microSD exceed 100mA.

6. **Vnominal: 3.3**

    6.1. Ensure that the above tests were successful at reading and writing to the micro

    SD card at nominal voltage.

7. **Vmax: 3.4**

7.1. Adjust the DC power into the PCB to 3.4V.

7.2. Perform an initialization, read, and write operation to the microSD card using the STM32 Nucleo board running a test program.

7.3. Ensure that these operations were successful at 3.4V.

8. Vmin: 3.2

8.1. Adjust the DC power into the PCB to 3.2V.

8.2. Perform an initialization, read, and write operation to the microSD card using the STM32 Nucleo board running a test program.

8.3. Ensure that these operations were successful at 3.2V.

mcrcntrllr_mcrsd_crd_pcb_comm : Input

9. Other: Pullup Resistors on Comm Lines

9.1. With the microSD PCB disconnected from DC power, test the resistance between each data bus line and the DC bus to ensure that each pullup resistor is in place.

9.2. Connect the microSD PCB to 3.3V DC and ensure that each data line is pulled to 3.3V.

10. Protocol: SD Protocol

10.1 Connect the microSD PCB DC power into the PCB to 3.3V.

10.2 Perform an initialization, read, and write operation to the microSD card using the STM32 Nucleo board running a test program.

10.3 Using an oscilloscope, ensure that these communications happen using SD protocol.

11. Vmax: 3.4

11.1 Adjust the DC power into the PCB to 3.4V.

11.2 Perform an initialization, read, and write operation to the microSD card using the STM32 Nucleo board running a test program.

11.3 Ensure that these operations were successful at 3.4V.

11.4 Because the high level of the data bus is determined by the voltage level of the power bus, the above tests will ascertain whether the databus can operate at 3.4V.

12. Vnominal: 3.3

12.1 Adjust the DC power into the PCB to 3.4V.

12.2 Perform an initialization, read, and write operation to the microSD card using the STM32 Nucleo board running a test program.

12.3 Ensure that these operations were successful at 3.4V.

Because the high level of the data bus is determined by the voltage level of the power bus, the above tests will ascertain whether the data bus can operate at 3.4V.

mcrsd_crd_pcb_mcrcntrllr_comm : Output

13. Other: Pullup Resistors on Comm Lines

13.1 With the microSD PCB disconnected from DC power, test the resistance between each data bus line and the DC bus to ensure that each pullup resistor is in place.

13.2 Connect the microSD PCB to 3.3V DC and ensure that each data line is pulled to 3.3V.

14. Protocol: SD Protocol

14.1 Connect the microSD PCB DC power into the PCB to 3.3V.

14.2 Perform an initialization, read, and write operation to the microSD card using the STM32 Nucleo board running a test program.

14.3 Using an oscilloscope, ensure that these communications happen using SD protocol.

15. Vmax: 3.4

15.1 Adjust the DC power into the PCB to 3.4V.

15.2 Perform an initialization, read, and write operation to the microSD card using the STM32 Nucleo board running a test program.

15.3 Ensure that these operations were successful at 3.4V.

15.4. Because the high level of the data bus is determined by the voltage level of the power bus, the above tests will ascertain whether the databus can operate at 3.4V.

16. **Vnominal: 3.3**

16.1 Adjust the DC power into the PCB to 3.4V.

16.2 Perform an initialization, read, and write operation to the microSD card using the STM32 Nucleo board running a test program.

16.3 Ensure that these operations were successful at 3.4V.

16.4 Because the high level of the data bus is determined by the voltage level of the power bus, the above tests will ascertain whether the data bus can operate at 3.4V.

4.1.6 References and File Links

[1] “MicroSDXC Memory Card Features - Kingston Technology.” [Online]. Available: https://www.kingston.com/datasheets/SDCIT-specsheet-64gb_us.pdf. [Accessed: 08-Jan-2022].

[2] “Plane (geometry),” *Wikipedia*, 25-Oct-2021. [Online]. Available: https://en.wikipedia.org/wiki/Plane_(geometry). [Accessed: 08-Jan-2022].

[3] “Simplified specifications: SD association,” *SD Association | The SD Association*, 23-Dec-2020. [Online]. Available: https://www.sdcard.org/downloads/pls/pdf/?p=Part1_Physical_Layer_Simplified_Specification_Ver8.00.jpg&f=Part1_Physical_Layer_Simplified_Specification_Ver8.00.pdf&e=EN_SS1_8. [Accessed: 22-Jan-2022].

4.1.7 Revision Table

Date	**Action**
3/5/22	Malachi added this section to the main document
1/21/22	Malachi completed Validation section.

1/20/22	Malachi updated Design Documentation
1/7/21	Malachi finished draft of section 1, 2, 3, 4, 5, 6 and 7.
1/6/21	Malachi finished Initial section creation. Created section 1.

4.2 Microcontroller Block

4.2.1 Description

The purpose of the Microcontroller block (championed by Samuel Barton) is to run all firmware blocks including data analysis, data transfer, serial communication firmware, etc., as well as receive and transmit data between the Inertial Measurement Unit (IMU), SD Card, and the science computer found inside the payload bay of the glider. This block will be the primary means of processing the measurement data collected by the XSENS MTI-3 IMU. With that in mind, the design for this block will be centered around an STM32 L4R5 series microcontroller that offers 2 Mbytes of Flash memory and 640 Kbytes of RAM memory to ensure ample memory to hold all data in memory before being transmitted over the satellite link to shore.

4.2.3 Design

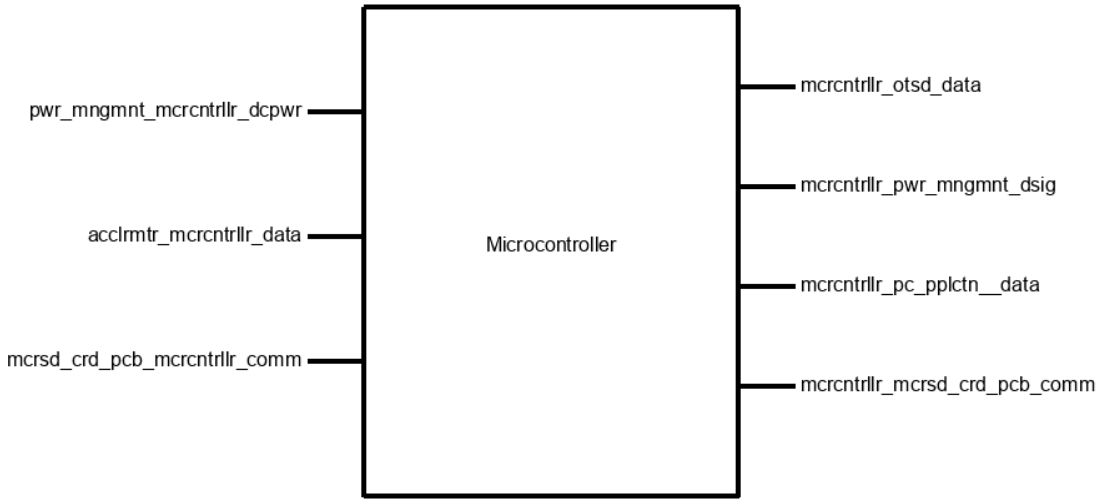

Figure 4.2.3.1: Black-box diagram for the “Microcontroller” block.

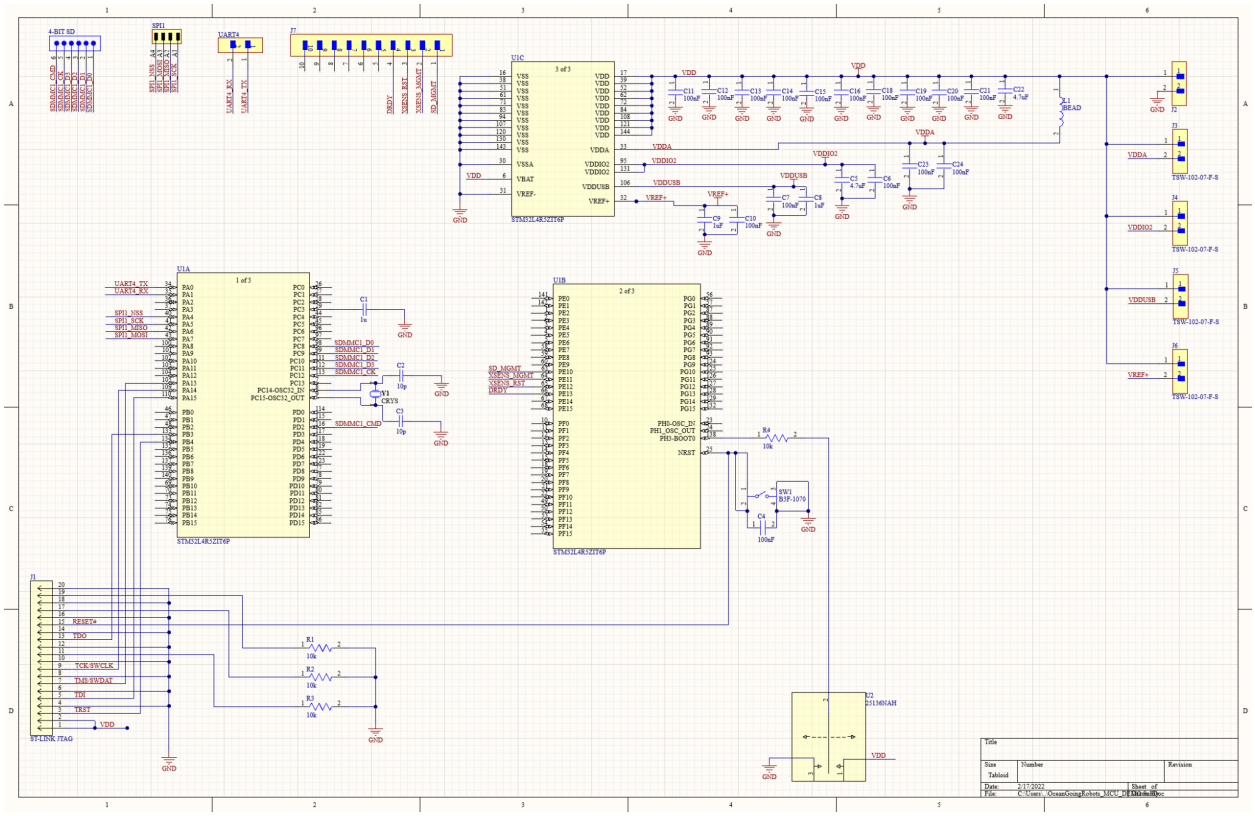

Figure 4.2.3.2: Designed schematic for the STM32 microcontroller and header pins to all peripherals.

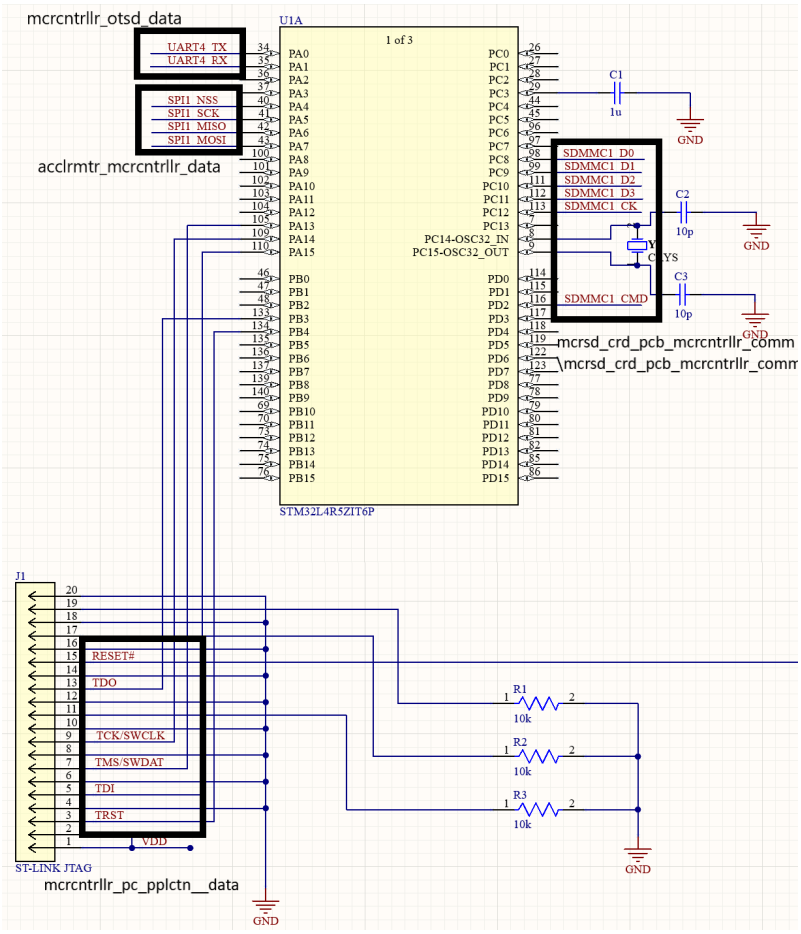

Figure 4.2.3.3: JTAG connection (used for programming/debugging using ST-Link) connections to GPIO pins.

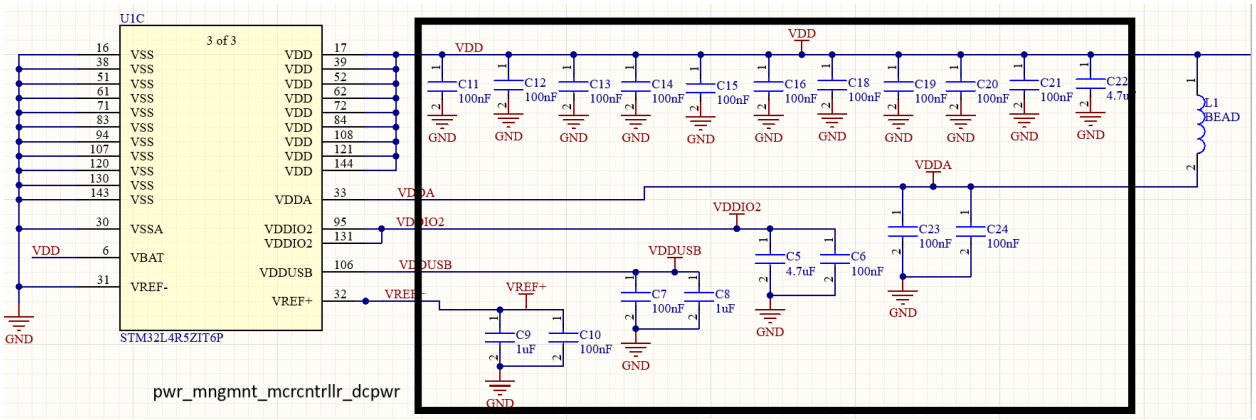

Figure 4.2.3.4: Bypass capacitor circuitry to STM32 VDD/VSS pairs with included ferrite bead.

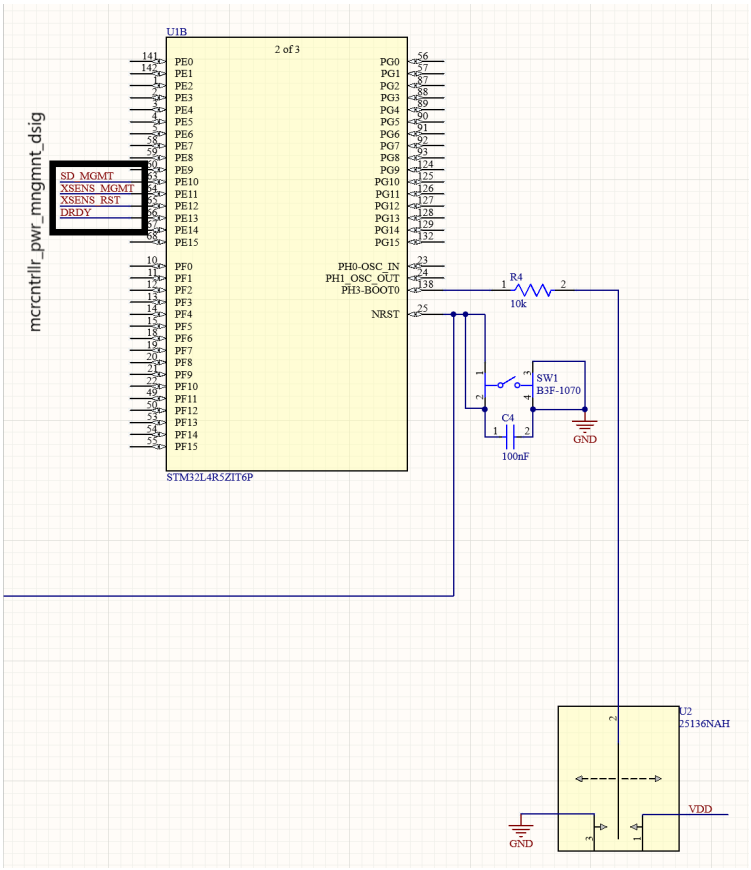

Figure 4.2.3.5: Connections to STM32 GPIO pins to power management block, as well as reset button and boot switch included.

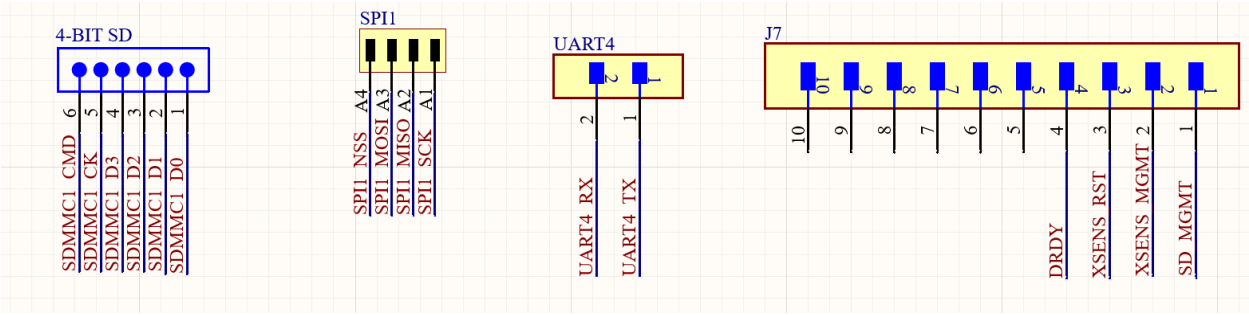

Figure 4.2.3.6: Header pins used for debugging SD, SPI, and UART protocols as well as monitoring the various GPIO pins to be used.

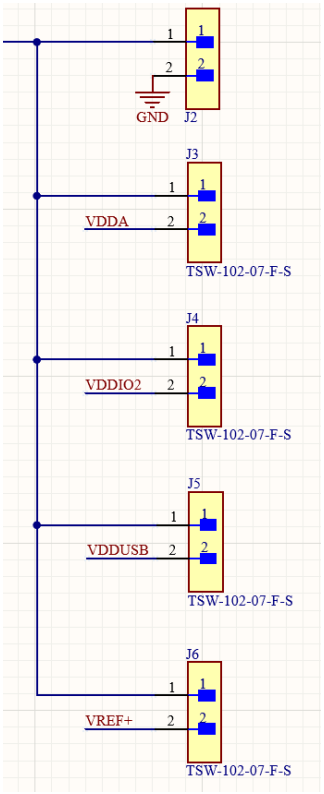

Figure 4.2.3.7: Header pins included to connect the extraneous input voltage pins (VDDA, VDDIO2, VDDUSB, VREF+) to VDD if not needed.

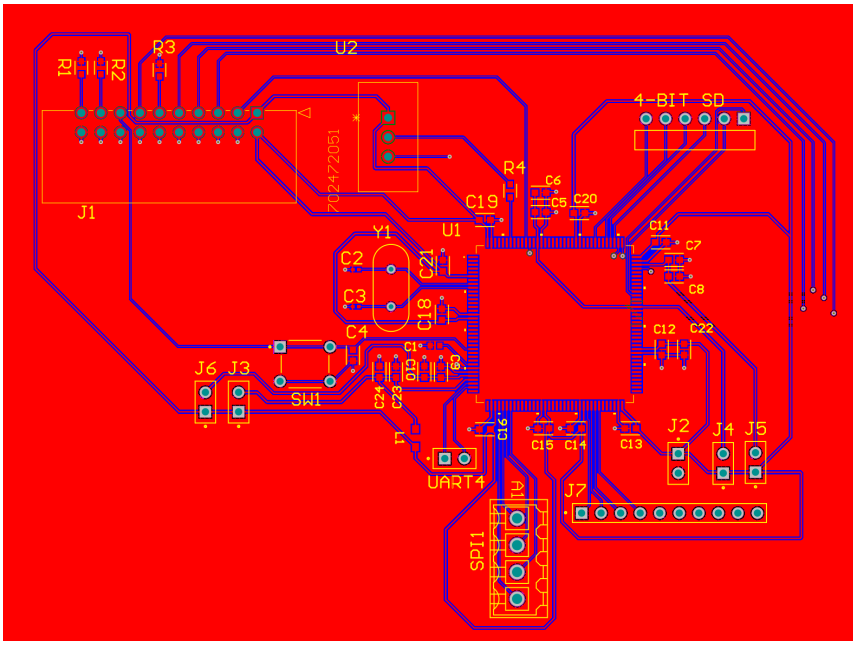

Figure 4.2.3.8: Designed PCB layout used for testing.

4.2.4 General Validation

As stated earlier, the two main purposes of this block are to hold the firmware to allow for communication between the various modules in the system and analyze the incoming data from the IMU. As the IMU outputs upwards of 7 different types of measurement data including acceleration, orientation, and a time stamp, the microcontroller chosen must have large amounts of RAM to ensure all measurements are stored and accounted for. In the block description statement above, it was noted that the STM32L4R5 series MCU has 640 Kbytes of RAM storage on board which should be plenty to hold the data being accumulated for the 20 minutes the Slocum G3 glider is at the surface.

As apparent in the Black-box diagram of the microcontroller block (Figure 1 above), there are 7 different interfaces that are associated with the microcontroller. Each of these interfaces will require physical connections to either the power management block, the accelerometer/inertial measurement unit (IMU), the SD card/serial interface, or our on-shore PCs as each interface will require at least one of the input/output pins to fulfill its purpose. The STM32 will be using 3 different communication protocols depending on the desired peripheral. SPI will be used between the STM32 and the XSENS IMU, UART (which will then be translated into RS232) between the microcontroller and the science computer on-board the gider, and the SD protocol for transferring data to/from the SD card. Furthermore, general GPIO pins will be used for toggling the desired voltage levels used for the power management digital signals. For debugging the communication protocols and digital signals, header pins have been established to allow for more simplicity in debugging (shown in Figure 6).

Alongside the I/O pins necessary to support the rest of the system, there are over 20 different pins out of the 144 total pins found on the STM32 that are dedicated to powering this module. As found in Figure 4 above, there are many instances around the perimeter of the block labeled VDD/VSS, in which VDD will be connected to the output of the voltage regulator, and VSS will be connected to the ground plane of the PCB. Between each of these sets of pins, a “decoupling” capacitor with a value between the 10’s of nanofarads and 1 microfarad will be placed between VDD and the via to the ground plane on the PCB in order to minimize some of the high-frequency noise that is typically common on the output of a switching voltage regulator, acting as a form of a low-pass filter.

The pins VDDUSB, VDDIO2, VDDA, VBAT, and VREF+ (found in Figure 4) will have the ability to be shorted to VDD using jumpers from header pins (Figure 7) if the extra functionality these pins offer will not be used. According to the STM32L4R5ZIP datasheet [1], this is the suggested configuration if the extra supply pins will not be used. Considering the timeline of the project and the requirement to be able to change designs as further blocks develop, including these headers for shorting will allow for more freedom and will prevent extraneous limitations being placed on the system.

For programmability, a JTAG ribbon connected to the STMicroelectronics ST-Link [2] seemed to be the most appropriate for the given design. The STM32 also allows for Serial Wire Debug (SWD) programmability, however, the STM NUCLEO boards our team has been using for

developing firmware and test scripts includes settings for using the ST-Link, making it a more suitable option as the settings currently in place for programming will not need to be changed.

Within the JTAG header shown in Figure 3, there includes pins RESET#, TDO, TCK/SWCLK, TMS/SWDAT, TDI, and TRST. These are all declared in the STM32L4R5 datasheet as the following and are stated in Table 1.

Table 4.2.4.1: Pin declaration for JTAG port

Pin Name	**JTAG Debug Port**	**SW Debug Port**	**Pin Assignment (STM)**
TMS/SWDAT	JTAG test mode selection	Serial wire data input/output	PA13
TCK/SWCLK	JTAG test clock	Serial wire clock	PA14
TDI	JTAG test data input	-	PA15
TDO	JTAG test data output	-	PB3
RESET#	JTAG test nReset	-	PB4

As shown above, the JTAG pins listed also include the ability to work as a Serial Wire (SW) debug configuration depending on the desired functionality of the connections (set in the STM Cube IDE). The choice to use the JTAG port will allow the team to use either JTAG for serial debugging/programming as well as SW debugging/programming which allows for the opportunity to further optimize the system to achieve our low-power goal.

4.2.5 Interface Validation

Table 4.2.5.1: Interface validation table for Microcontroller block.

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
pwr_mngmnt_mcrcntrllr_dcpwr : Input		
Inominal: 11mA	Given from the STM32 Data sheet for the given MCU. [1]	The voltage source is in the correct range given by that data sheet so the MCU won't have to pull more current for the same input power. Also, the MCU includes a low power mode that drastically decreases current

		draw that may be used for our project.
Ipeak: 14mA	Given from the STM32 Data sheet for the given MCU. [1]	After writing test scripts through the STM32 Cube IDE which gives an estimate of power consumption with the given firmware uploaded to the MCU, the power estimate showed a value for current below this value.
Vmax: 3.4V	Given from the STM32 Data sheet for the given MCU. [1]	The output of the buck regulator being designed will have an output voltage of 3.3V. All VDD pins will be connected directly to the output of the regulator module and it can be assumed that there ground plane used to reference this voltage is at 0V.
Vmin: 1.71V	Given from the STM32 Data sheet for the given MCU. [1]	Assuming the connections to the buck regulator block are low resistance within the PCB or any various protoboard arrangements, there should be no extraneous voltage drop along the signal line.
Vnominal: 3.3V	Given from the STM32 Data sheet for the given MCU. [1]	The output of the buck regulator being designed will have an output voltage of 3.3V. All VDD pins will be connected directly to the output of the regulator module.

acclrmtr_mcrcntrllr_data : Input

Messages: Cartesian (XYZ) Acceleration, 3x 32-bit floating point numbers	This is the defined data type from the accelerometer as given in the XSENS MTI-3 Data Sheet. [3]	The STM32L4R5 series MCUs have a 32-bit ARM Cortex M-4 core which allows for 32-bit float data transfer. [1]

Messages: Cartesian Magnetometer Measurement: 3x 32-bit floating point numbers (Note: This is a unitless measurement normalized to 1)	This is the defined data type from the accelerometer as given in the XSENS MTI-3 Data Sheet. [3]	The STM32L4R5 series MCUs have a 32-bit ARM Cortex M-4 core which allows for 32-bit float data transfer. [1]
Messages: Rotation in Quaternions: 4x 32-bit floating point numbers	This is the defined data type from the accelerometer as given in the XSENS MTI-3 Data Sheet. [3]	The STM32L4R5 series MCUs have a 32-bit ARM Cortex M-4 core which allows for 32-bit float data transfer. [1]
Protocol: SPI	The XSENS module has many examples and guides for implementing SPI communication which will allow for improved implementation for the team.	The STM32 Cube IDE allows for the designation of certain pins for various tasks (including SPI) and generates base code to help jump-start the programming process.

mcrcntrllr_otsd_data : Output

Datarate: 9600 Baud	Defined by the acceptable baud rates of the science computer currently installed on the glider.	The STM32 can handle this baud rate depending on the firmware and system configurations set using the STM32 Cube IDE.
Messages: Significant wave height (m), dominant period (s), wave direction (degrees magnetic), maximum wave height (m), second highest wave height (m), maximum period (s), four spectral parameters, a wave component number (if needed), and a timestamp (UTC).	Given to us by the project partners as “what they need.”	This is determined by firmware, there should be no physical constraints on the transfer of this data besides physical connections. Assuming the Data Analysis Firmware Block is able to calculate these values, this is simply a matter of writing these values to the I/O pins used for serial communication to the science computer.

Protocol: UART	This is what was suggested by the project partner and is still being researched. Will require a level shifter or some sort of custom module.	As this protocol will require a level shifter to generate the +/- 12V signals used in RS232, assuming the level shifter designed/purchased is suitable for the 3V voltage levels created by the I/O pins, this is obtainable.

mcrcntrllr_pwr_mngmnt_dsig : Output

Logic-Level: 3.3V	This is typical voltage level output by the MCU.	As stated by the STM32 data sheet, the GPIO pins on this board have a nominal “high” voltage level of 3.3V [1].
Other: Max Current Draw: 500uA	As the pins used for the dsig will be connected to the gate of a MOSFET, there should be very little leakage current.	The STM32 MCU can toggle its GPIO pins high or low based off of the firmware written in the Cube IDE.
Other: Active High	This is based on the design of the power management block for our system.	The GPIO pins on the STM32 allow for a maximum current draw of 20mA [1]. As the accompanying design will consist of FETs, there should be no more than 500nA of current drawn which will easily be supplied by the accompanying IO pins.

mcrcntrllr_pc_pplctn__data : Output

Datarate: 115200 Baud	This baud rate is common for serial communication and supported by both devices.	The STM32 can handle this baud rate depending on the firmware and system configurations set using the STM32 Cube IDE.
Messages: Cartesian (XYZ) Acceleration	Needs to mimic the entire system working with the on-board science computer in the glider.	This is determined by firmware, there should be no physical constraints on the transfer of this data besides physical connections.

Messages: Roll, Pitch, Yaw orientation in quaternions	Needs to mimic the entire system working with the on-board science computer in the glider.	This is determined by firmware, there should be no physical constraints on the transfer of this data besides physical connections.
Protocol: UART	UART is a common communication protocol for serial communication and can be implemented rather easily using the STM Cube IDE	There will be a Micro USB connection in series with a USB-Serial module to ensure serial communication through between the two devices.

mcrcntrllr_mcrsd_crd_pcb_comm : Output

Protocol: SD Protocol	This protocol is standard in practice when creating data transfer between a device and SD card.	This function is supported by the STM32 Cube IDE set up in which the properties of the protocol can be set to match whatever is currently written in firmware.
Vmax: 3.4V	The data transfer will be carried out with the GPIO pins, therefore the voltage levels should be within the specs of these pins.	This is below the threshold for maximum voltage the GPIO pins can output, therefore ensuring no damage will be done to either device. [1]
Vnominal: 3.3V	The data transfer will be carried out with the GPIO pins, therefore the voltage levels should be within the specs of these pins.	This is the nominal output voltage for the GPIO pins, therefore this voltage level should be suitable throughout the lifetime of the MCU without causing any damage to the device. [1]

mcrsd_crd_pcb_mcrcntrllr_comm : Input

Protocol: SD Protocol	This protocol is standard in practice when creating data transfer between a device and SD card.	This function is supported by the STM32 Cube IDE set up in which the properties of the protocol can be set to match whatever is currently written in firmware.
Vmax: 3.4	The data transfer will be carried out with the GPIO pins, therefore the voltage levels	This is below the threshold for maximum voltage the GPIO pins can receive, therefore ensuring

	should be within the specs of these pins.	no damage will be done to either device. [1]
Vnominal: 3.3	The data transfer will be carried out with the GPIO pins, therefore the voltage levels should be within the specs of these pins.	This is below the threshold for maximum voltage the GPIO pins can receive, therefore ensuring no damage will be done to either device. [1]

4.2.6 Verification Plan

1) Initial Set up:
    a) Connect VDD pin to power supply set to 3.3V and GND pin to negative terminal of power supply.
    b) Connect ST-Link to PC and JTAG ribbon to JTAG connector found on PCB.
    c) Flash test script to the STM32 MCU used for testing the system.
2) Verifying pwr_mngmnt_mcrcntrllr_dcpwr input interface:
    a) Measure current through the VDD connection using DMM.
        i) Target current: Less than 14mA
    b) Range supply voltage from 3.4V to 1.71V.
        i) Verify circuit is still functioning within this range. The MCU should still be discoverable through the STM32Cube IDE.
3) Verifying acclrmtr_mcrcntrllr_data input interface:
    a) Connect an oscilloscope to pins used for communication using SPI.
        i) Verify toggling clock (CLK).
        ii) Verify more data being sent through the MOSI (Master Output, Slave Input) connection in comparison to MISO (Master Input, Slave Output).
        iii) Verify the Slave Select Line becomes active before communication.
4) Verifying mcrcntrllr_otsd_data output interface:
    a) Connect the Tx pins to be used for serial communication to oscilloscope.
        i) Find the shortest time where the Tx pin is low.
            (1) Baud rate can be approximated by taking the inverse of the shortest period (f = 1/T) [4].
        ii) Connect UART, VDD, and GND to RS232 to USB module
        iii) Verify all parameters have been sent:
            (1) Significant wave height (m)
            (2) Dominant period (s)
            (3) Wave direction (degrees magnetic)
            (4) Maximum wave height (m)
            (5) Second highest wave height (m)
            (6) Maximum period (s)
            (7) Four spectral parameters
            (8) Wave component number
            (9) Timestamp (UTC)
5) Verifying mcrcntrllr_pwr_mngmnt_dsig output interface:
    a) Verify all pins used for power management (mcrcntrllr_pwr_mngmnt_dsig) toggle within noted acceptable voltage ranges using DMM.
        i) Target Voltage: 3.3V

ii) Maximum Output Voltage: 3.4V
iii) Imax: > 500nA

6) Verifying mcrcntrllr_pc_pplctn__data output interface:
   a) Connect the Tx pins to be used for serial communication to oscilloscope.
      i) Find the shortest time where the Tx pin is low.
         (1) Baud rate can be approximated by taking the inverse of the shortest period (f = 1/T) [4].
      ii) Connect UART, VDD, and GND to RS232 to USB module
      iii) Verify all parameters have been sent:
         (1) Cartesian (XYZ) Acceleration
         (2) Roll, Pitch, Yaw orientation in quaternions
7) Verifying mcrcntrllr_mcrsd_crd_pcb_comm/mcrsd_crd_pcb_mcrcntrllr_comm interface:
   a) Connect SDMMC1_D0-D3, SDMMC1_CK, SDMMC1_CMD, VDD, and GND pins to SD card PCB.
   b) Run test script transferring data to SD card from STM32.
   c) Transfer SD card to PC and check to make sure a write has been made.

4.2.7 References and File Links

[1] STMicroelectronics. *STM32L496R5ZI Datasheet.* (2021). Accessed: February 4, 2022. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32l4r5zi.html

[2] STMicroelectronics. *ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32.* (n.d.). Accessed: February 18, 2022. [Online]. Available: https://www.st.com/en/development-tools/st-link-v2.html

[3] XSENS. *MTI-1 series Datasheet.* Accessed: February 2, 2022. [Online]. Available: https://www.xsens.com/hubfs/Downloads/Manuals/MTi-1-series-datasheet.pdf

[4] Kumari.net. *Determining Unknown Baud Rate.* (n.d.). Accessed: January 7, 2022. [Online]. Available: https://www.kumari.net/index.php/random/37-determing-unknown-baud-rate

4.2.8 Revision Table

Table 4.2.8: Revision table for Microcontroller Block.

01/07/2022	Samuel Barton: Created first draft of document.
02/04/2022	Samuel Barton: Updated block description, general validation, interface validation, and verification plan sections.
02/16/2022	Samuel Barton: Updated design section to show designed configuration (schematic and PCB layout).
02/18/2022	Samuel Barton: Updated general validation section to highlight finalized design, updated

	interface validation to reflect scope changes.

4.3 Data Analysis Firmware

4.3.1 Description

The purpose of the Data Analysis Firmware block (championed by Samuel Barton) is to analyze the swell data taken by the accelerometer using the STM32 microcontroller (MCU). This data includes glider heading, significant wave height, wave period, wave direction, maximum wave height, second highest wave height, maximum period, the spectral parameters of the waves being analyzed, and a time-stamp from when each measurement was taken.

The primary functions of this block are to rotate the measurements taken by the inertial measurement system (IMU) to reflect the current position of the glider, take a Fourier Transform of each of the sensor's measurements in reference to the Earth's coordinate system, calculate three Power Spectral Densities (PSD) and three Cross Spectral Densities (CSD), calculate the five spectral coefficients and spectral moments, and finally calculate the eleven parameters requested by the project partners.

4.3.2 Design

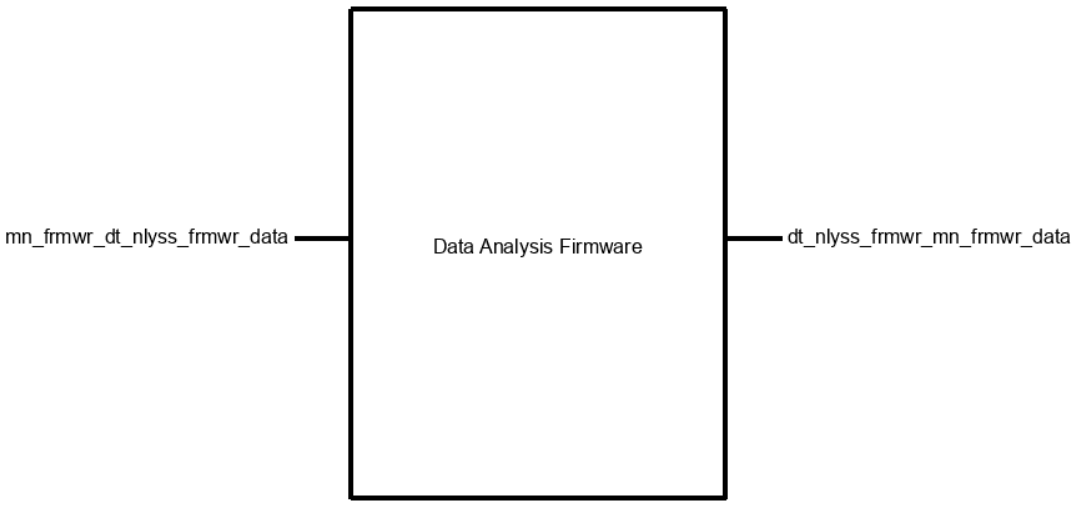

Figure 4.3.2.1: Black-box diagram of the Data Analysis Firmware block.

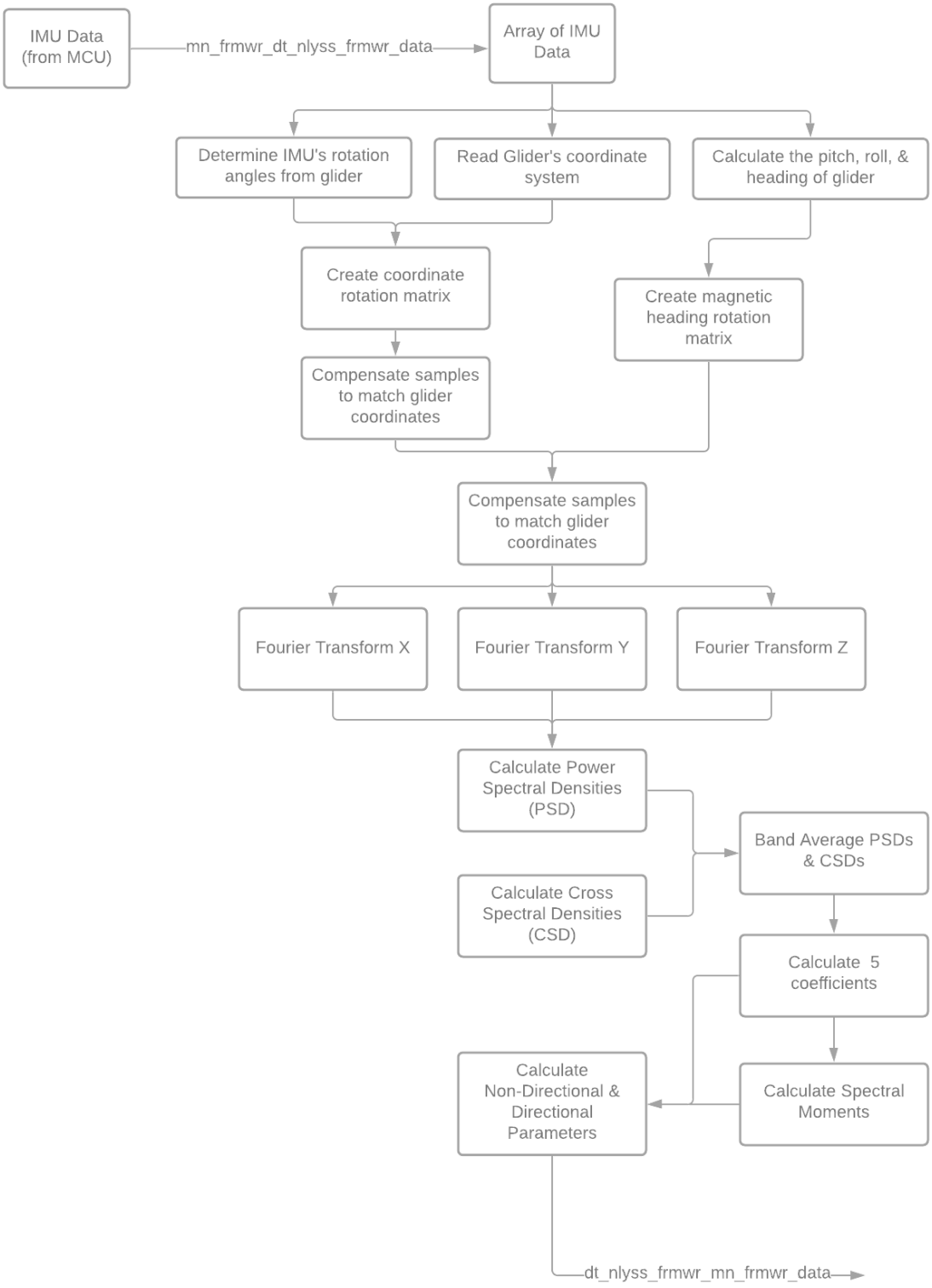

Figure 4.3.2.2: Higher Level code flow required for completion of the Data Analysis Firmware block.

4.3.3 General Validation

The sole purpose of this project is to be able to document swell conditions while the Slocum G3 gliders are out at sea. On the shore-side of this project, the CS team is focusing on creating a visual representation of the analyzed data that the IMU system is capturing while the glider is on the surface. This data will then be transmitted over a satellite link currently present on the glider. Not only will the data be transmitted via satellite communications, the Glider team requests analyzed data to be stored on the SD card to be mounted within the payload bay of the glider.

As the STM32 MCU has ample processing power to accomplish the calculations necessary parameters to be sent over the satellite link and stored on the SD card, these calculations will be carried out using the built-in functionalities of the microcontroller. The STM32 MCUs are built using ARM cortex M4 processors which include Common Microcontroller Software Interface Standard (CMSIS) tools including a specific digital signal processor (DSP) library [1].

The key function within the Data Analysis Firmware block is the Fourier Transform that is later used to find the Power Spectral Densities and Cross Spectral Densities of the swell conditions being monitored. The CMSIS DSP library includes many transform functions which can be used on-board the STM32, including multiple variations of Fast Fourier Transforms (FFTs). Therefore, designing this block around the CMSIS libraries currently included on the chosen MCU will allow for the block to accomplish all necessary tasks for the completion of this project.

4.3.4 Interface Validation

Table 4.3.4.1: Interface validation table for Data Analysis Firmware block.

mn_frmwr_dt_nlyss_frmwr_data : Input

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Messages: Acceleration Data in N-Direction (32-bit floats): Acceleration North-South	The data analysis firmware must be able to analyze acceleration in the North-to-South Direction to allow for complete wave-data analysis.	The data measured by the IMU will be sent into RAM on the MCU un-touched. This data will be read directly from the MCU's memory and analyzed from there.

Messages: Acceleration Data in U-Direction (32-bit floats): Vertical acceleration data	The data analysis firmware must be able to analyze vertical acceleration to allow for complete wave-data analysis.	These measurements taken by the IMU will also be stored in RAM on the MCU. The STM32 is a 32-bit microcontroller, and can handle 32-bit values [2].
Messages: Acceleration Data in E-Direction (32-bit floats): Acceleration East-West	The data analysis firmware must be able to analyze acceleration in the East-to-West Direction to allow for complete wave-data analysis.	Within RAM, the acceleration data coming from the XSENS accelerometer will be stored in three different arrays for the N, E, and U acceleration. Therefore, when the data analysis begins, it will be able to draw from the separate data stored in the IMU.

dt_nlyss_frmwr_mn_frmwr_data : Output

Messages: Dominant Period in E Direction (s)	This parameter is standard for most swell analyses. This was required per the project partners' requests.	The dominant period will be calculated by using the parameter a0 from the output of the "Calculate the 5 Coefficients" block as found in the NOAA document [3].
Messages: Dominant Period in U Direction (s)	This parameter is standard for most swell analyses. This was required per the project partners' requests.	The dominant period will be calculated by using the parameter a0 from the output of the "Calculate the 5 Coefficients" block as found in the NOAA document [3].
Messages: FFT Results in Upward (U) Direction (Real and Complex Values)	The results of the FFT will be used further to calculate the rest of the 11 parameters required per the project partners' requests.	Using the CMSIS RFFT library, the FFT data in the "U" direction will be calculated within the "Fourier Transform Z" block in Figure 2 above.
Messages: FFT Results in Northward (N) Direction (Real and Complex Values)	The results of the FFT will be used further to calculate the rest of the 11 parameters required per the project partners' requests.	As the CMSIS RFFT library is being used to calculate all subsequent parameters, the FFT results in the "N" direction will be calculated in the "Fourier Transform X" block in Figure 2 above.

Messages: FFT Results in Eastward (E) Direction (Real and Complex Values)	The results of the FFT will be used further to calculate the rest of the 11 parameters required per the project partners' requests.	Using the CMSIS RFFT library, the FFT data in the "U" direction will be calculated within the "Fourier Transform Z" block in Figure 2 above.
Messages: Dominant Period in N Direction (s)	This parameter is standard for most swell analyses. This was required per the project partners' requests.	The dominant period will be calculated by using the parameter a0 from the output of the "Calculate the 5 Coefficients" block as found in the NOAA document [3].

4.3.5 Verification Plan

In order to test the functionality of the Data Analysis Firmware block, test data will be hard-coded into the STM32 firmware via the STMCube IDE. From there, data from the Fourier Transform as well as the outputs from the calculations to find the Power and Cross Spectral Densities, five coefficients, spectral moments, and finally the directional and non-directional parameters will be transmitted to the test pc terminal for validation of completion.

To begin:

1) Download the STMCube IDE script to the STMCube.
2) Connect STM32 L4R5 nucleo board to the pc through the COM ports.
3) Open the terminal with specification to the COM port the nucleo board is connected to.
4) Build the STM32 Project.
5) Run the STM32 Debugger on the main.c file which includes all calculations to be completed.

To verify the mn_frmwr_dt_nlyss_frmwr_data interface:

1) Verify through the STM32 Cube IDE that acceleration data in N, E, U directions have been established using the memory allocation tool.

To verify the RFFT has been taken:

1) Verify data has been saved in memory showing multiple frequency bands and their respective energies to the terminal.

To verify the dt_nlyss_frmwr_mn_frmwr_data interface:

1) Verify data from calculations for the desired outputs include:
    a) Maximum Wave Height
    b) Second Highest Wave Height

c) Wave Component Number
d) Timestamp
e) Wave Direction
f) Significant Wave Height
g) Maximum Period
h) Four Spectral Parameters (a1, b1, a2, b2)
i) Dominant Period

4.3.6 References and File Links

[1] Common Microcontroller Software Interface Standard. *CMSIS DSP Library.* (n.d.). Accessed January 17, 2022. [Online]. Available: https://www.keil.com/pack/doc/CMSIS/DSP/html/index.html

[2] STMicroelectronics. *Floating point unit demonstration on STM32 microcontrollers.* (2016). Accessed: January 12, 2022. [Online]. Available: https://www.st.com/resource/en/application_note/dm00047230-floating-point-unit-demonstration-on-stm32-microcontrollers-stmicroelectronics.pdf

[3] National Data Buoy Center. *Nondirectional and Directional Wave Data Analysis Procedures.* (1996). Accessed: January 13, 2022. [Online]. Available: https://www.ndbc.noaa.gov/wavemeas.pdf

4.3.7 Revision Table

Table 4.3.7: Revision table for Data Analysis Firmware block.

Date:	**Action:**
01/07/2022	Samuel Barton: Created first draft of document.
01/21/2022	Samuel Barton: Changed the block to be validated from the MCU block to the Data Analysis Firmware block and re-worked each section accordingly.
03/05/2022	Samuel Barton: Changed the interface properties to show what was checked off for the Block 1 Checkoff.

4.4 SD Interface Firmware

4.4.1 Description

This project will be mounted to the inside of an ocean glider and will use an accelerometer to capture wave motion data and save it to an SD card for later examination. This block will be a function library that will take an array of floats from the accelerometer board and converts it into data formatted to be saved to a file on the microSD card to be analyzed on shore. The library will also have functions to perform the reverse of that operation- open a file on the microSD card and return an array of floats, so that the glider can process some of the data and radio it back to shore.

4.4.2 Design

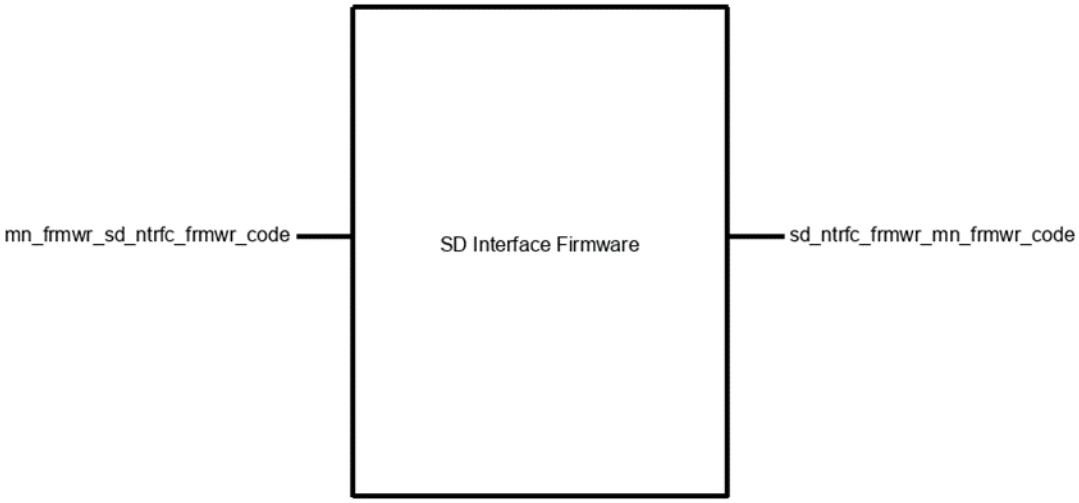

Figure 4.4.2.1: Black Box Diagram of the SD Software Block

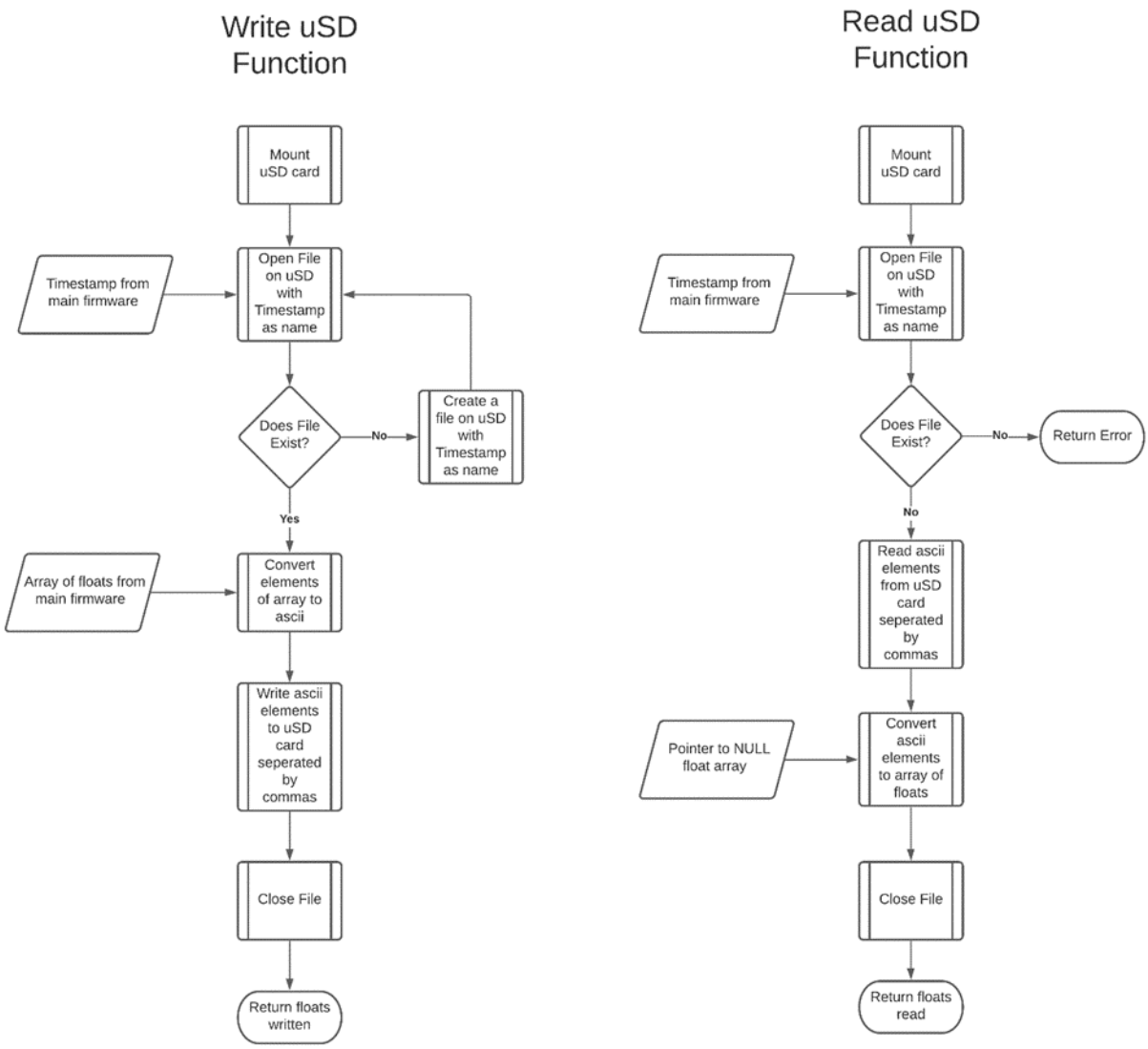

Figure 4.4.2.2: Flowchart of the software library

4.4.3 General Validation

This design allows for the main firmware to call functions that will take care of the reading and writing of the microSD card. Due to its design, all of the file creation, float conversion, SD initialization, etc. will be abstracted into simple read and write function calls. This will reduce the complexity of the main firmware, increase readability, and add versatility to the main firmware.

4.4.4 Interface Validation

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
mn_frmwr_sd_ntrfc_frmwr_code : Input		
Other: Data: Populated array of floats	This is the array that will be written to a .csv file on the SD card.	The Xsens accelerometer board firmware samples data and converts them to floats, which will be stored to an array. This array will be passed to the write function to be saved to the SD card.
Other: Data: Null float array pointer	This is a null pointer which will be populated by information from a SD card read.	The data analysis firmware will be expecting an array of floats to perform the Fourier transform on. By putting the data from the SD card into an array pointed to by this pointer, the data analysis firmware will be able to use that data easily [1].
Other: Data: Timestamp	The timestamp is a unique identifier which can be used to name files.	By using the timestamp given to our system by the science computer, we can ensure that each time the glider surfaces it will have a unique name to give the files where the accelerometer data will be stored, even after multiple reboot cycles of the system.
sd_ntrfc_frmwr_mn_frmwr_code : Output		
Other: Data: Populated array of floats from file on uSD	This is the array containing the data read from the microSD card.	The data analysis firmware will be expecting an array of floats to perform the Fourier transform on. By putting the data from the SD card into an array, the data

		analysis firmware will be able to use that data easily.
Other: Return Value: Number of floats read	This is the return value of the read function.	By returning the number of floats read from the microSD card, we can do some rudimentary error checking to ensure the operation was successful.
Other: Return Value: Number of floats written	This is the return value of the write function.	By returning the number of floats written to the microSD card, we can do some rudimentary error checking to ensure the operation was successful.
Other: Return Value: Error Code	This returns integer error codes in the case that a read or write operation fails.	Integer error codes are standard practice and are a useful debugging tool.

4.4.5 Verification Process

1. Create a test .csv file on the microSD card with a known name and populate it with floating point numbers.

2. Insert the microSD card into the test jig and begin running test code.

3. In the test code, attempt to use the name of the file as the timestamp input for the read function. If the file opens (Does not return an error code), the timestamp property is confirmed to work.

4. In the test code, pass the read function a null pointer. If after the read is complete, the pointer points to an array of floats read from the file, both the null float array property and the return value populated array of floats property will be confirmed to have worked.

5. Examine the integer value returned by the read function. If it matches the number of floating-point values in the uSD card file, this property is confirmed to work.

6. Using a different value for the timestamp as used previously and the array of floats read in the previous steps, attempt to call the write function to the uSD card.

7. If after the write operation is complete, there is a new file in the uSD card with the same floating point values as the other file, the floating-point array input is confirmed to work.

8. If after the write operation is complete, the write function returns the number of floating point numbers as were in the array, the return value: number of floats written property is confirmed to work.

9. Attempt to read a file by passing a non-existent filename as the timestamp value. If the read function returns a negative integer, the error code property is confirmed to work.

4.4.6 References and File Links

[1] S. Barton, “Data Analysis Firmware Block Validation.” .

4.4.7 Revision Table

Date	**Action**
3/5/22	Malachi added this to the main project document
2/18/22	Malachi updated sections to accommodate changes suggested by peers.
2/4/2022	Malachi completed the draft for sections 1-7 of this section.

4.5 Buck Regulator Block

4.5.1 Description

This block includes the buck regulator power supply for the main board PCB. The Slocum glider is able to supply between 7 and 17 volts to our board, but the microcontroller that we are using needs a 3.3V supply. We will be designing a buck regulator power supply for our board using the TPS54233D. Since this is the same chip that is used for the tech demo, we’ll use the tech demo board to prove its functionality before implementing it in the final PCB.

4.5.2 Design

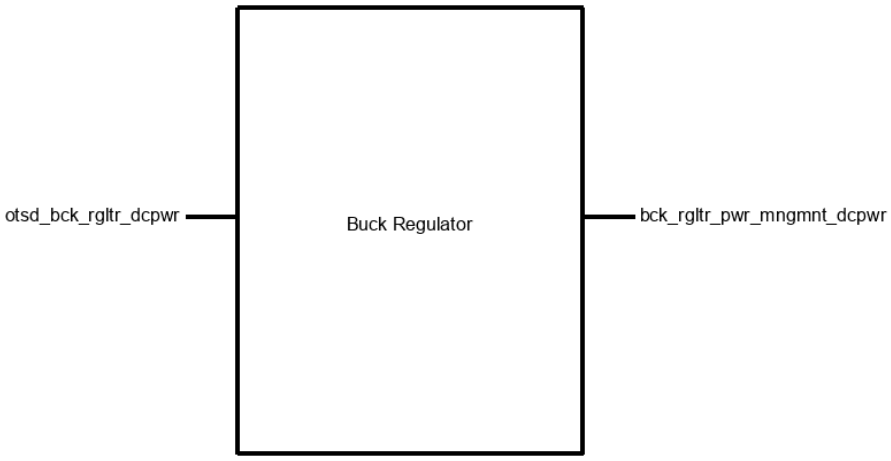

Figure 4.5.2.1: Black Box Diagram for the Buck Regulator

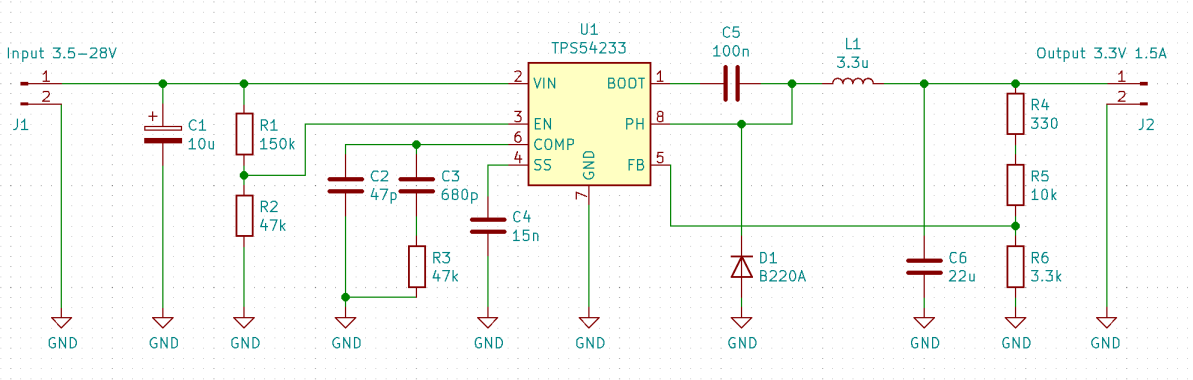

Figure 4.5.2.2: Schematic for the Buck Regulator

Note: otsd_bck_rgltr_dcpwr = Input 3.5-28V and bck_rgltr_pwr_mngmnt_dcpwr = Output 3.3V.

4.5.3 General Validation

The slocum glider provides our board with either a 7V or 17V DC power line. We need a way to turn this into a 3.3V supply for the microcontroller, accelerometer, and SD card. Whatever solution is used, it needs to be high efficiency because the power budget on our project is extremely tight.

The TPS54233 is a good choice because it has upwards of 85% efficiency with V_{IN} at 7-8V. This is among the highest of the regulator chips considered for this design. Additionally, this chip is readily available at the TekBots store, as well as being in stock on Mouser and DigiKey.

The circuit used for this block is known to be correct because it is taken almost entirely from the reference design on the TPS54233 datasheet. The only changes that were made were the feedback resistors. These were altered to change the output voltage from 5V to 3.3V.

4.5.4 Interface Validation

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
otsd_bck_rgltr_dcpwr : Input		
Inominal: 8mA	This was derived by working back from the average current needed for the microcontroller, accelerometer and SD card. It takes into account efficiency and voltage level changes.	$I_{IN} = (I_{OUT}*V_{OUT})/(efficiency*V_{IN})$. $I_{IN} = (0.05*3.3)/(0.85*17)$ ~= 8mA
Ipeak: 25mA	This was derived by working back from the maximum current needed for the microcontroller, accelerometer and SD card. It takes into account efficiency and voltage level changes.	$I_{IN} = (I_{OUT}*V_{OUT})/(efficiency*V_{IN})$. $I_{IN} = (0.14*3.3)/(0.85*17)$ ~= 25mA
Vmax: 17V	This is the highest possible supply voltage from the Slocum Glider	VIN of the TPS54233 can range from 3.5 to 28 Volts (from the TPS54233 Datasheet)
Vmin: 7	This is the lowest possible supply voltage from the Slocum Glider	VIN of the TPS54233 can range from 3.5 to 28 Volts (from the TPS54233 Datasheet)
bck_rgltr_pwr_mngmnt_dcpwr : Output		
Inominal: 50mA	This is the average current needed for the microcontroller, accelerometer and SD card.	The TPS54233 can supply up to 1.5A of current. (from the TPS54233 Datasheet)
Ipeak: 140mA	This is the average current needed for the microcontroller,	The TPS54233 can supply up to 1.5A of current. (from the

	accelerometer and SD card.	TPS54233 Datasheet)
Vmax: 3.35V	Calculated from the worst case of resistance tolerance issues.	With 1% tolerance resistors, V_{OUT} could be as high as 0.8*(10.433/3.267 + 1) = 3.35V (equation from the TPS54233 Datasheet)
Vmin: 3.1V	Calculated from the worst case of resistance tolerance issues.	Tested prototype PCB and found the minimum output voltage to be 3.1V

4.5.5 Verification Process

To test the TPS54233 Buck Regulator block, we will be using the PCB designed for the tech demo. The tech demo board is set up for 5V output but the resistances can be changed to allow for a 3.3V output.

Testing Procedure:

1. Set a power supply to 7V and then turn it off. Attach the ground lead from the power supply to the negative input pin on the board. Attach the positive lead from the power supply through a digital multimeter (for reading input current) to the positive input pin on the board.
2. Attach the output pins on the board to the leads of an electronic load. Set the load to draw 0mA of current.
3. Turn on the power supply and the load.
4. For each increment of 5mA up to 200mA, record the output voltage and the input current.
5. Repeat the process with the power supply at 17V.

Analysis:

1. Go through the data and ensure that the voltage never goes above 3.35V or below 3.25V.
2. Check that the input current is ~30mA when the output current is 50mA.
3. Check that the input current is ~80mA when the output current is 140mA.
4. Calculate the efficiency of the power supply at 50mA output current and verify that it is above 80%

4.5.6 References and File Links

TPS54233 Datasheet

4.5.7 Revision Table

Table 4.5.7.1: Section 4.5 Revision Table

Date:	**Action:**
1/6/22	Document Created by Miles
1/21/22	Fixed resistor value in schematic
1/21/22	Made changes to reflect peer feedback
3/5/22	Miles added section to project document

4.6 Power Management Block

4.6.1 Description

This block includes a power management circuit to cut power to the SD card and the accelerometer. Since the SD card uses a non negligible amount of power while it is in sleep mode, we would like to be able to cut its power line while it is not being written to or read from. For this we will need a simple MOSFET circuit that uses the output of a GPIO pin on the microcontroller to control the flow of power to the SD card. We will use an identical circuit to cut the power to the accelerometer when we aren't using it (while performing the fourier transform). We plan to model the circuit in LTSpice before implementing it in the final PCB. The block champion is Miles Drake.

4.6.3 Design

The image below shows the black box diagram for the power management block. The inputs are the 3.3V generated by the buck regulator and a digital signal from the microcontroller that tells the block which sections of the circuit to give power to. The outputs of the block are the power to each section of the circuit, the microcontroller, the SD card, and the accelerometer.

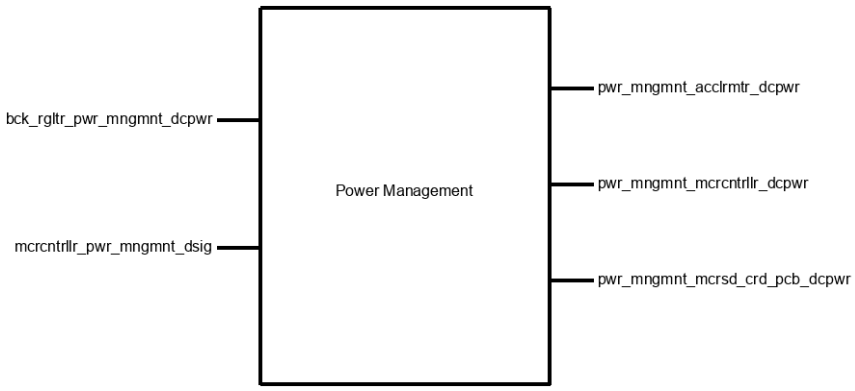

Figure 4.6.3.1: Black Box Diagram

Keep in mind that the output to the microcontroller is essentially just a pass through, since we always want to have it powered. This is shown in the top section of the circuit diagram below. The 3V3 line from the buck regulator passes directly to the microcontroller (Through 0 ohm resistors to keep net labels organized). This also provides the power for the power management circuit.

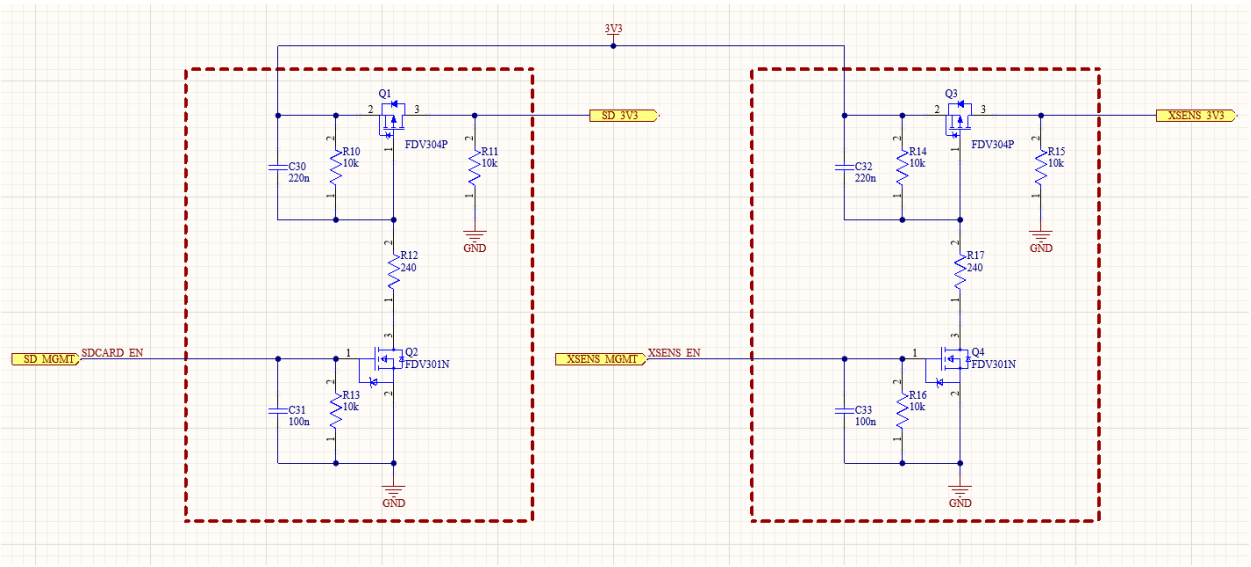

Figure 4.6.3.2: Schematic for Block Implementation

Table 4.6.3.1: Black Box and Schematic Corresponding Interfaces

Black Box Interface	**Corresponding Schematic Interface**
mcrcntrllr_pwr_mngmnt_dsig	SDCARD_EN, XSENS_EN
pwr_mngmnt_mcrsd_crd_pcb_dcpwr	SD_3V3
pwr_mngmnt_acclrmtr_dcpwr	XSENS_3V3

The rest of the schematic shows two identical power switching blocks, one for the SD card and one for the accelerometer. In each, the NMOS on the bottom is held closed by the pull down resistor. This sets the gate voltage of the PMOS to 3.3V, closing the PMOS and not letting any current flow to the SD Card/XSENS. When the digital logic enable signal is asserted high, it opens the NMOS, setting the gate of the PMOS to GND. This opens the PMOS and allows current to flow to the SD Card/XSENS.

The specific parts for the MOSFETS will be an FDV301N NMOS and an FDV304P PMOS.

4.6.3 General Validation

The design of this block is all about simplicity and low power. There are load switch ICs that you can buy off the shelf, but this circuit only takes two MOSFETs and a few passive components. Depending on the resistors that are used in the circuit (And how slow of a turn on time you can deal with), the current draw while on can be reduced to less than 8uA.

I chose MOSFETS over BJTs specifically because they do not need much gate current over time, just enough to get them open. This fits well with the low power requirements for our project.

I also chose to make the design active high so that when the power is coming up and the microcontroller is turning on, the SD card and the XSENS will be held in reset (no power).

4.6.4 Interface Validation

bck_rgltr_pwr_mngmnt_dcpwr : Input

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Inominal: 31mA	This is the expected nominal current draw of the whole system from the power supply. (Accelerometer, Microcontroller, and SD card)	The current will flow through the PMOS transistor FDV304P. According to the FDV304P Datasheet, it can handle up to 4.5A.
Ipeak: 97mA	This is the expected peak current draw of the whole system from the power supply. (Accelerometer, Microcontroller, and SD card)	The current will flow through the PMOS transistor FDV304P. According to the FDV304P Datasheet, it can handle up to 4.5A.
Vmax: 3.35V	This is the maximum output	According to the FDV304P

	voltage of the buck regulator based on resistor tolerances.	Datasheet, this PMOS can handle up to 8V gate source voltage.
Vmin: 3.1V	This is the minimum output voltage of the buck regulator based on resistor tolerances.	According to the FDV304P Datasheet, this PMOS can handle up to 8V gate source voltage.
Vnominal: 3.3V	This is the nominal output voltage of the buck regulator based on equations from the datasheet.	According to the FDV304P Datasheet, this PMOS has approximately 50mOhms on resistance with a Vgs of 3.3V.

pwr_mngmnt_acclrmtr_dcpwr : Output

Inominal: 15mA	This is the expected nominal current draw of the XSENS accelerometer. (Measured with prototype)	The current will flow through the PMOS transistor FDV304P. According to the FDV304P Datasheet, it can handle up to 4.5A.
Ipeak: 33mA	This is the expected peak current draw of the XSENS accelerometer. (Measured with prototype)	The current will flow through the PMOS transistor FDV304P. According to the FDV304P Datasheet, it can handle up to 4.5A.
Vmax: 3.4V	This is a safe range of operating supply for the XSENS. (Taken from the XSENS Datasheet)	This circuit will not increase the voltage, only relay it from the buck regulator, which has already been proven to not give voltage higher than 3.35V
Vmin: 2.9V	This is a safe range of operating supply for the XSENS. (Taken from the XSENS Datasheet)	This circuit will not decrease the voltage by very much, only relay it from the buck regulator, which has already been proven to not give voltage lower than 2.9

pwr_mngmnt_mcrcntrllr_dcpwr : Output

Inominal: 11mA	This is the expected nominal current draw for the microcontroller (Taken from the	The current will flow through the PMOS transistor FDV304P. According to the FDV304P

	STM32 Datasheet)	Datasheet, it can handle up to 4.5A.
Ipeak: 14mA	This is the expected peak current draw for the microcontroller (Taken from the STM32 Datasheet)	The current will flow through the PMOS transistor FDV304P. According to the FDV304P Datasheet, it can handle up to 4.5A.
Vmax: 3.4V	This is a safe range of operating supply for the microcontroller (Taken from the STM32 Datasheet)	This circuit will not increase the voltage, only relay it from the buck regulator, which has already been proven to not give voltage higher than 3.35V
Vmin: 1.71V	This is a safe range of operating supply for the microcontroller (Taken from the STM32 Datasheet)	This circuit will not decrease the voltage by very much, only relay it from the buck regulator, which has already been proven to not give voltage lower than 3.25

pwr_mngmnt_mcrsd_crd_pcb_dcpwr : Output

Inominal: 5mA	This is the expected nominal current of the SD card board. (Measured with prototype)	The current will flow through the PMOS transistor FDV304P. According to the FDV304P Datasheet, it can handle up to 4.5A.
Ipeak: 50mA	This is the expected peak current of the SD card board. (Measured with prototype)	The current will flow through the PMOS transistor FDV304P. According to the FDV304P Datasheet, it can handle up to 4.5A.
Vmax: 3.4	This is a safe range of operating supply voltage for the SD Card	This circuit will not increase the voltage, only relay it from the buck regulator, which has already been proven to not give voltage higher than 3.35V
Vmin: 2.9	This is a safe range of operating supply voltage for the SD Card	This circuit will not decrease the voltage by very much, only relay it from the buck regulator, which has already been proven to not give

		voltage lower than 2.9

mcrcntrllr_pwr_mngmnt_dsig : Input

Logic-Level: 3.3V	The microcontroller has a logic level of 3.3V, so it makes sense to use that here as well.	The LTSPICE simulation of the circuit has been proven to work with logic level 3.3V
Other: Max Current Draw: 500uA	This current is set very low because we don't want extra power to be used to turn off the SD Card or XSENS power	According to the FDV301N Datasheet, the current draw on the gate of the NMOS is a maximum of 100nA.
Other: Active High	This is active high so that the accelerometer and SD card will be held in reset (no power) before the microcontroller has booted up.	The LTSPICE simulation of the circuit has shown that no power is delivered to the SD card/XSENS when the logic signal is not asserted or asserted low.

4.6.5 Verification Process

To verify the power management block, we will be breadboarding the circuit using SOT23 breakout boards for the MOSFETS. The prototype buck regulator circuit will be used to power this block. The verification plan is as follows:

1. Set a power supply to 17V and then turn it off. Attach the ground lead from the power supply to the negative input pin on the buck regulator board. Attach the positive lead from the power supply to the positive input pin on the board.
2. Attach the output of the power supply to the 3V3_BUCK input of the power management circuit.
3. Attach the 3V3_SD to an electronic load and set the current draw to 50mA.
4. Attach the 3V3_XSENS to another electronic load and set the current draw to 33mA.
5. Turn on the power supply. Neither of the 3V3 lines for SD or XSENS should have any voltage. Verify that 3V3_MICRO has 3.3 volts on it using a DMM.
6. Take a jumper cable from the 3V3_BUCK line and attach it to the SDCARD_EN. Verify that 3V3_SD is in fact at 3.3 volts (by looking at the voltage level on the electronic load) and that nothing has exploded. Remove the jumper.
7. Put the jumper cable instead from 3V3_BUCK to the XSENS_EN. Verify that 3V3_XSENS is indeed 3.3 volts and that nothing has exploded.

8. Now attach two jumpers from 3V3_BUCK, one to each of the enable lines. Verify that both 3V3_SD and 3V3_XSENS have 3.3 volts on them.

4.6.6 References and File Links

FDV304P Datasheet
STM32 Datasheet
XSENS Datasheet

4.6.7 Revision Table

Table 4.6.7.1: Section 4.6 Revision Table

Date:	**Action:**
2/4/22	Document Created by Miles
2/17/22	Miles added information suggested by peer reviews
2/17/22	Miles Fixed schematic to show MOSFET Values
2/17/22	Miles Fixed interfaces to match new values on portal
3/5/22	Miles added section to project document

4.7 Accelerometer Block Validation

4.7.1 Description

This block includes the electrical circuitry necessary to power, configure and communicate with the XSENS Accelerometer. It includes the necessary passives to support the accelerometer, such as a decoupling capacitor, and resistors to configure the communication mode of the modules. It interfaces to the microcontroller via several digital signals, which allow it to transfer data to and from the microcontroller.

Our system will be mounted in an ocean glider, where it will use measurements from the accelerometer to measure the properties of waves. This block is necessary because it will allow our system to collect acceleration, and orientation data, and transfer that data to the host microcontroller. This data is critical to the operation of our system, and therefore this block is necessary.

4.7.3 Design

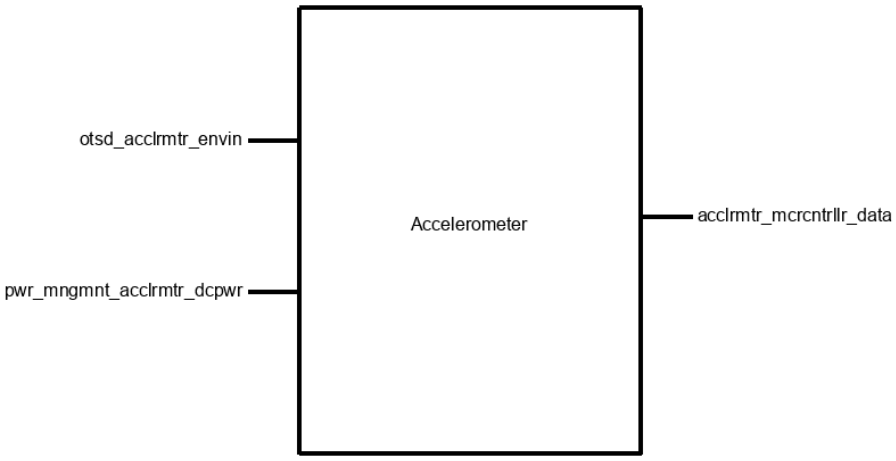

Figure 1: Accelerometer black box diagram.

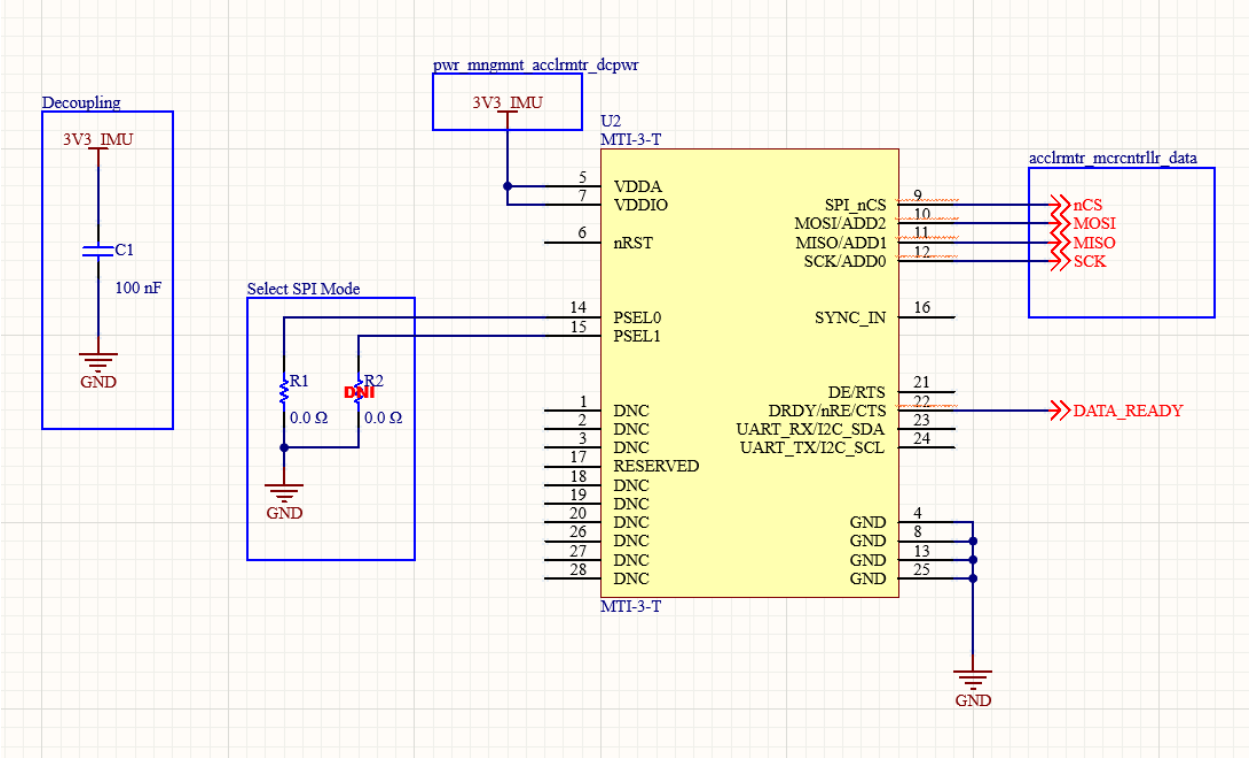

Figure 2: Accelerometer electrical schematic. Notes: (1) the blue boxes are labeled either with a note, or with the name of the interface they pertain to. (2) R2 is labelled DNI (Do not install), it is in the schematic so that the communication mode of the sensor could be changed without significant PCB rework if necessary later in the project.

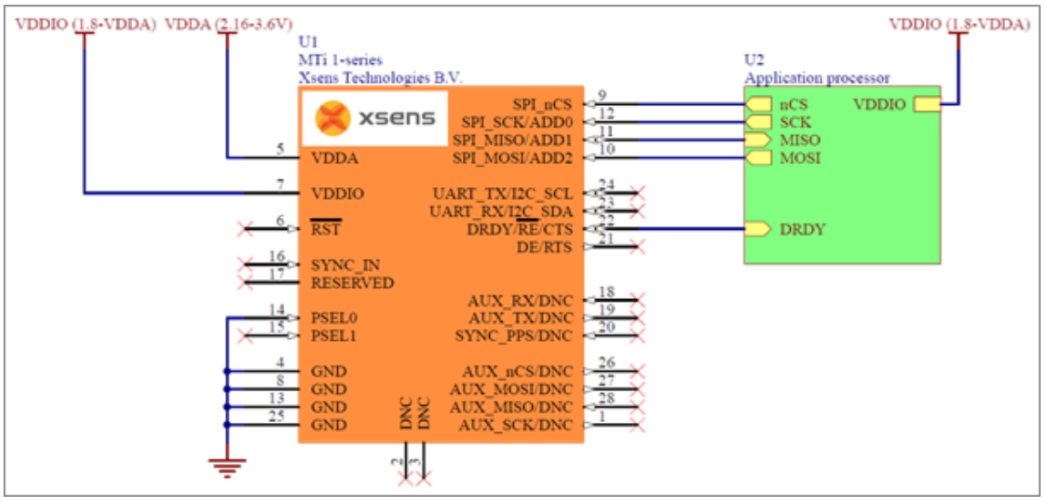

Figure 3: Typical application schematic for SPI mode communication. [1]

4.7.3 General Validation

This design was chosen because it implemented a sensor that will collect the data required by the system and easily interface with the microcontroller selected for the project. It is based on the typical schematic for the XSENS MTI-3 operating in SPI mode. The system needs to be able to measure acceleration, orientation, and heading in order to meet the system requirements, and this sensor is capable of collecting all of those measurements.

A significant driving force for the design of this block is that the previous year's group, as well as the project partners had already identified this sensor as being sufficient for the system in terms of accuracy, price, and power consumption. While other sensors could have been chosen, and implemented, this was the most obvious choice.

4.7.4 Interface Validation

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
otsd_acclrmtr_envin : Input		
Electromagnetic: Earth's Magnetic Field X, Y, Z	The system will be operating in locations where the earth's magnetic field will be strong enough to be detected.	The sensor can measure the Earth's magnetic field in the X, Y and Z directions [2]
Other: Roll, Pitch, Yaw	The system will have some orientation within 3D space.	The sensor can measure its orientation in 3D space [2]
Other: Acceleration: X, Y, Z	The system will experience acceleration throughout its operation.	The sensor can measure its acceleration in the X, Y and Z directions [2]
Other: Acceleration Nominal: +/- 1G in any axis	The system will be operating in a vehicle in the ocean and the primary source of acceleration will be from waves. Because of this it is unlikely that the system will experience more than +/-1G during normal operation.	The sensor can measure up to +/- 16G in any axis [2]
pwr_mngmnt_acclrmtr_dcpwr : Input		
Inominal: 15mA	The only part that consumes power in this block is the sensor, which has nominal current draw of 15mA	The product website lists power consumption of 44mW at 3.0v, giving 15mA of current consumption. [2]
Ipeak: 33mA	The sensor has a maximum current draw of 33mA	The datasheet for the sensor specifies a maximum power consumption of 100mW at 3.0v,

		giving 33mA peak current consumption. [3]
Vmax: 3.4V	This is the maximum voltage that the power supply block will output as per its design	The normal operating range of the sensor (not absolute maximums) specifies a maximum input voltage of 3.6V [3]
Vmin: 3.2V	This is the minimum voltage that the power supply block will output as per its design	The normal operating range of the sensor (not absolute maximums) specifies a minimum input voltage of 2.16V [3]
Vnominal: 3.3V	This is the nominal voltage that the power supply block will output as per its design	3.3V is between the minimum and maximum operating voltages of the sensor.

acclrmtr_mcrcntrllr_data : Output

Messages: Rotation in Quaternions: 4x 32-bit floating point numbers	We want to transfer the acceleration measurement from the sensor to the microcontroller, and quaternions make it computationally easy to perform vector rotations.	The sensor can be configured to output its orientation as a quaternion consisting of four 32-bit floating point numbers. [5]
Messages: Cartesian (XYZ) Acceleration, 3x 32-bit floating point numbers	We want to transfer the acceleration measurement from the sensor to the microcontroller.	The sensor can be configured to output its acceleration in X, Y, Z, as three 32-bit floating point numbers. [5]
Messages: Cartesian Magnetometer Measurement: 3x 32-bit floating point numbers (Note: This is a unitless measurement normalized to 1)	We want to transfer the magnetometer measurement from the sensor to the microcontroller so that the magnetic heading can be determined.	The sensor can be configured to output its magnetic measurement in X, Y, Z, as three 32-bit floating point numbers. [5]
Protocol: SPI	SPI is sufficient for our application and is supported by the microcontroller we are using.	The sensor has an SPI interface. [1]

4.7.5 Verification Process

1. The circuit will be connected to a variable power supply with the power supply turned OFF
2. The data connections will be connected to an STM32 Nucleo development board, and the ground of the circuit will be connected to the ground of the development board.
3. Set the variable power supply to 3.2V This will verify that the block works properly at Vmin from the pwr_mngmnt_acclrmtr_dcpwr interface.
4. Run a test program to collect floating-point data from the sensor. The test program should output the data collected (Orientation, Acceleration, Magnetometer) to a serial port. This will demonstrate both the acclrmtr_mcrcntrllr_data, and the otsd_acclrmtr_envin interfaces.
5. To verify acceleration measurements, the sensor shall be rotated to 6 orientations, corresponding to the sensor pointing in the +/-X, +/-Y, and +/-Z directions. When in each of these orientations, +/- 1G will be measured on each of the axes. During this test, the orientation and magnetometer measurements shall change with orientation as well. This will demonstrate the Rotation, Acceleration, Magnetometer, and SPI properties of the acclrmtr_mcrcntrllr_data interface.
    a. Orientation data can be verified using a visualization program.
    b. Acceleration data can be observed to change numerically, or it can be plotted.
    c. Magnetometer data can be observed numerically, or plotted. It can be verified using a cell phone compass. Note: magnetometer data is unitless and normalized to 1.0.
    d. If necessary, an oscilloscope or logic analyzer can be used to verify the operation of the SPI bus.
6. Monitor supplied current. During execution of the test program the supplied current shall be within +30%/-100% of Inominal, and shall never exceed Ipeak. This will demonstrate the Inominal, and Ipeak property of the pwr_mngmnt_acclrmtr_dcpwr interface.
7. Repeat steps 4-6 for power supply voltage of 3.4V This will demonstrate the block operates at Vmax from the pwr_mngmnt_acclrmtr_dcpwr interface.

4.7.6 References and File Links

[1] https://mtidocs.xsens.com/interfaces$spi

[2] https://www.xsens.com/mti-3

[3] https://www.xsens.com/hubfs/Downloads/Manuals/MTi-1-series-datasheet.pdf

[4] https://mtidocs.xsens.com/functional-description$pin-descriptions

[5] https://mtidocs.xsens.com/messages$data-related-messages

4.7.7 Revision Table

1/08/2022	Grayson Lewis: Initial Draft
1/21/2022	Implemented feedback, improved testing procedure, added typical application schematic from documentation, changed “2021” to “2022” in line 1 of the revision table.
2/1/2022	Removed DATA_READY property (not a strictly necessary property, difficult to test)

4.8 Serial Interface Firmware Block

4.8.1 Description

This block configures, and uses the hardware on the STM32 microcontroller for serial communication with the ocean glider, and serial communication with the PC application. This block is a firmware library that the main firmware block will be able to call in order to send and receive data from either the ocean glider or the PC application. The main firmware will tell this block how to (and which) UART modules to configure. The main firmware will be able to pass this library data, as well as information on how to format that data, and which UART should transmit, and this library will configure the data correctly, and transmit it. Conversely, the main code will be able to request data from either of the UART modules, and the module should return the data in the order it is received, as well as the status of the UART modules, and whether data has been received but not passed to the main firmware yet.

4.8.3 Design

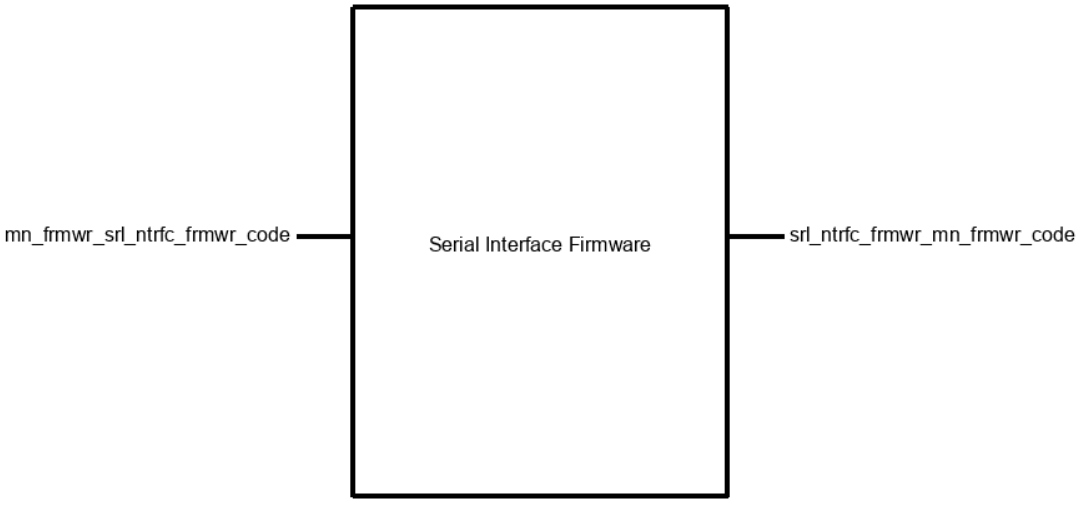

Figure 4.8.3.1: Black box diagram

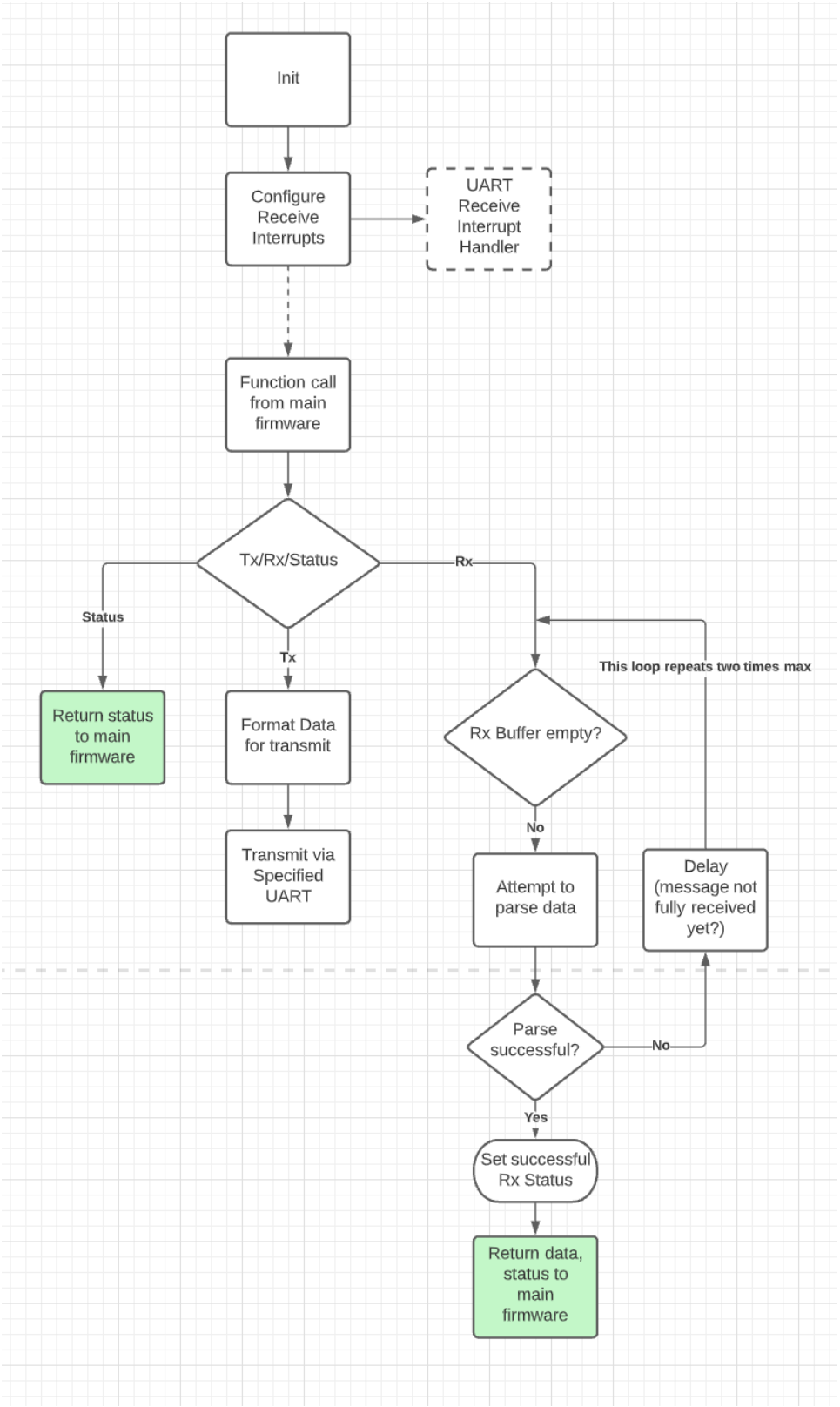

Figure 4.8.3.2: Main code flowchart

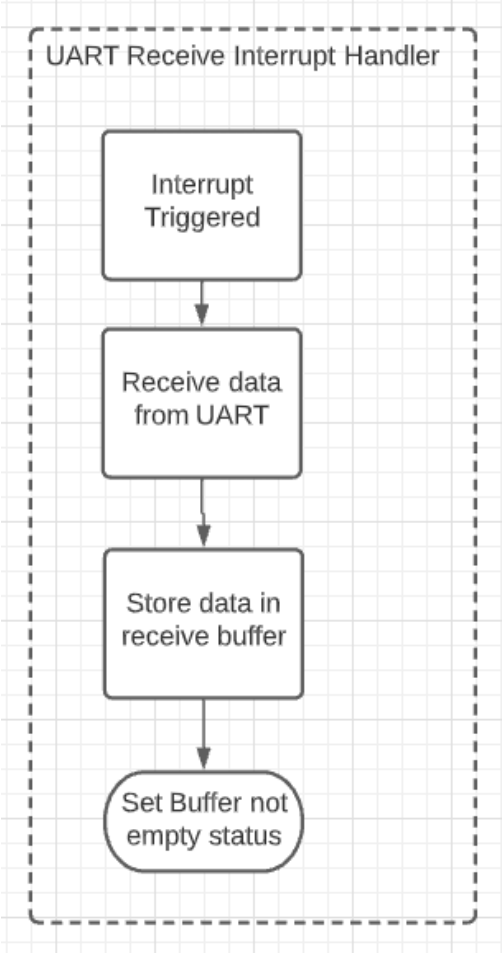

Figure 4.8.3.3: UART Receive Interrupt Handler

The first thing that this library will do is be initialized. The main firmware will call the init() function, which will configure which UART modules to use, set up the receive buffers, and enable the receive interrupts.

When the main firmware calls this library, it will have three options:

1. Transmit data:

   The main firmware will provide the data, the format the data should be transmitted in, as well as its selection of which UART module it should be transmitted from. The library will take this data, generate an array based on the format specified, and transmit it. The library will update its status and then return.

2. Receive data:

The main firmware will request to read from a specific UART module, if the receive buffer for that module is empty, a null pointer, and the status will be returned. If the receive buffer is not empty, the data in that buffer will be parsed. If the parse is successful, a pointer to the parsed data will be returned, as well as the status, and the data will be removed from the receive buffer. If the parse is unsuccessful, the code will delay for a period of time in the hopes that more data will be received which will allow the parse to be successful. If the parse is still unsuccessful, the data will be discarded, and the status will be set to an error state.

3. Check status:

   This will simply return the current status. This will probably include how much data is in the receive buffers, as well as whether any errors have occurred.

4.8.3 General Validation

This design was chosen for several reasons. The first being flexibility, as many of the decisions about what kind of data we want to send to/from the PC application are undecided, we want to leave room to add different kinds of data and methods of parsing that data. This design also handles the encoding and decoding of the data from the UART, which will make the UART aspects of the project much easier when we want to do system integration.

Another key design decision was to use interrupts instead of polling for receiving data. If polling is used, it can result in dropped bytes, as data may be received while the program is doing something else, and thus an incomplete or corrupted message may be received [1]. By using interrupts, this library will be able to receive data and hold it, even while the program is in the middle of executing other operations, reducing the risk of a corrupted message.

4.8.4 Interface Validation

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
mn_frmwr_srl_ntrfc_frmwr_code : Input		
Other: UART Selection: Which UART interface should be used to transmit/receive	This library will be responsible for managing communications with multiple UART modules, so we should be able to select which module to use	The microcontroller we are using has multiple UART modules, and is capable of interfacing with them through firmware. [2]

Other: Baud Rate: Speed the UART should operate at.	The baud rate depends on the hardware configurations of the other devices connected to the serial ports of the micro, so it should be configurable in order to accommodate those devices	The baud rate of the various UART modules is configurable. [2]
Other: Data: Numbers and strings to be transmitted via a UART Module	The main firmware will need to be able to pass this library data to be transmitted.	The microcontroller is capable of manipulating data and passing it to UART modules.

srl_ntrfc_frmwr_mn_frmwr_code : Output

Other: Status: Errors, and/or whether there is data in the receive buffers of the UARTs	The main firmware should know when an error has occurred, and should know how much data has been received	The firmware flowchart defines different states in which different statuses will be indicated.
Other: Data: Numbers and strings received from UART modules	The library should be able to return the data it received from the UART modules.	The microcontroller is capable of manipulating data and passing it to UART modules.
Other: Data Identifier: The type of data that is being received	The main firmware should not have to guess at what kind of data it is receiving.	By defining how data is parsed, metadata can be generated that describes the type of data being returned.

4.8.5 Verification Process

1. Load test program into microcontroller that interacts with the library.
2. Send data of various types and formats to both of the different UART modules. They can be received using a computer, or another microcontroller. Verify that the data was received and encoded correctly based on the specified format.
3. Using a microcontroller or computer, send data to the various uart modules. View the data that was returned to the main program and ensure that it is correct.
    a. Incorrect and incomplete data should be transmitted as well, the appropriate status messages should be returned.

4.8.6 References and File Links

[1] http://www.simplyembedded.org/tutorials/interrupt-free-ring-buffer/

[2] https://www.st.com/resource/en/datasheet/stm32l4r5vi.pdf

4.8.7 Revision Table

2/4/2022	Grayson Lewis: Initial version
2/18/2022	Grayson Lewis: Updated testing procedures

4.9 Revision Table

Table 4.9: Revision Table for Section 4.

Date	Action
03/05/2022	Samuel created the Microcontroller Block and Data Analysis Firmware Block sections.

Section 5: System Verification Evidence

5.1 Universal Constraints

5.1.1 The system may not contain a breadboard

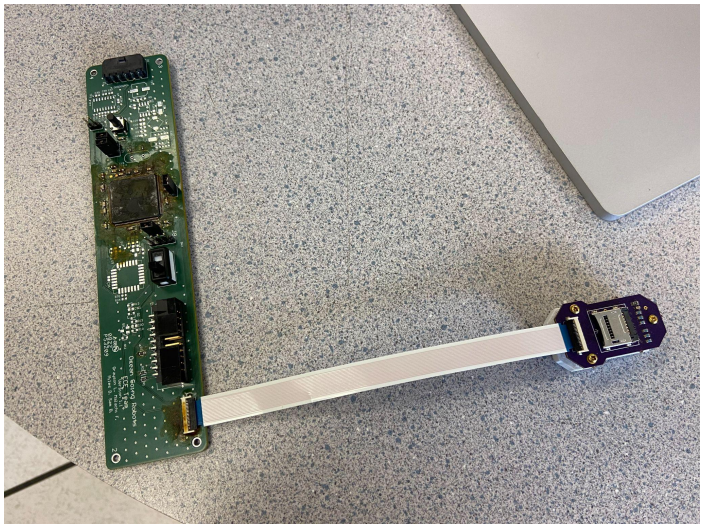

Figure 5.1.1.1: Fully assembled system does not contain a breadboard

5.1.2 The final system must contain both of the following: a student designed PCB and a custom Android/PC/Cloud application.

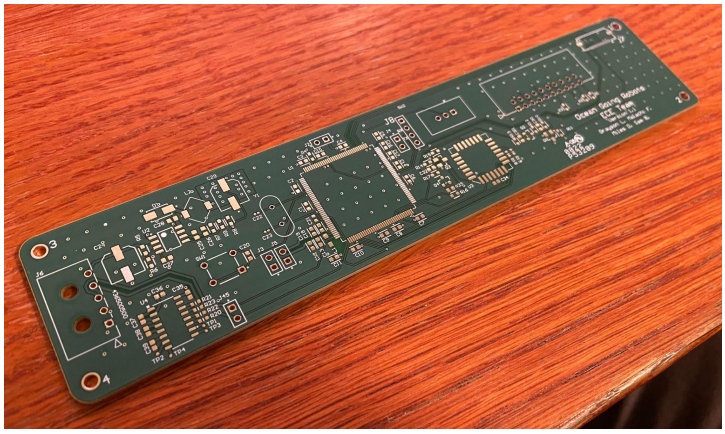

Figure 5.1.2.1: Student Designed PCB

A PC application was written to receive data from the system microcontroller, and store that data in a CSV file for analysis.

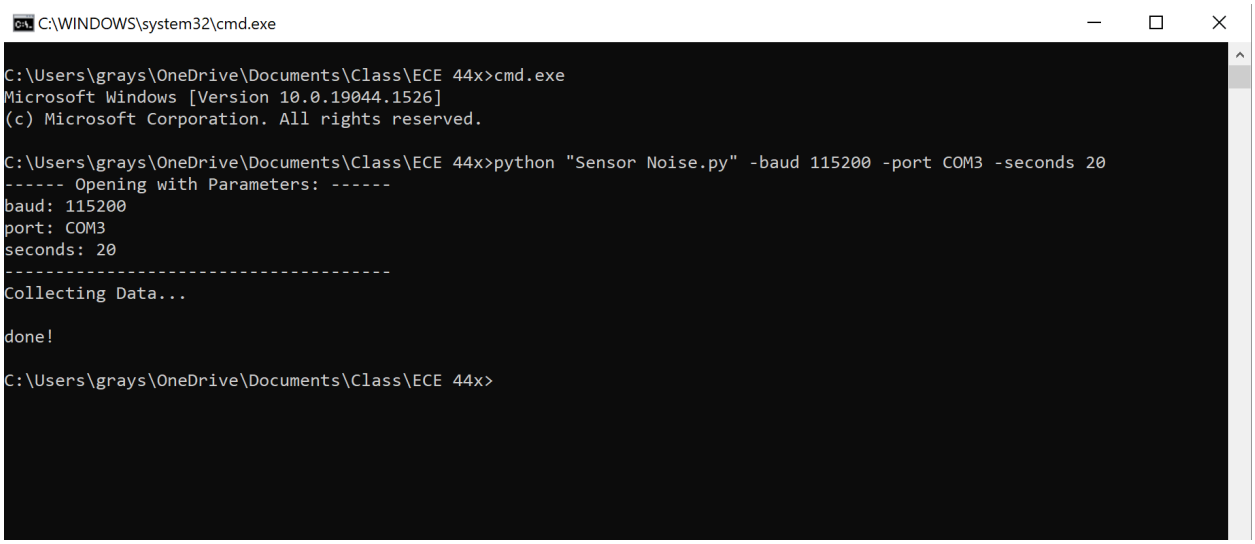

Figure 5.1.2.2: Command Line interface for PC application

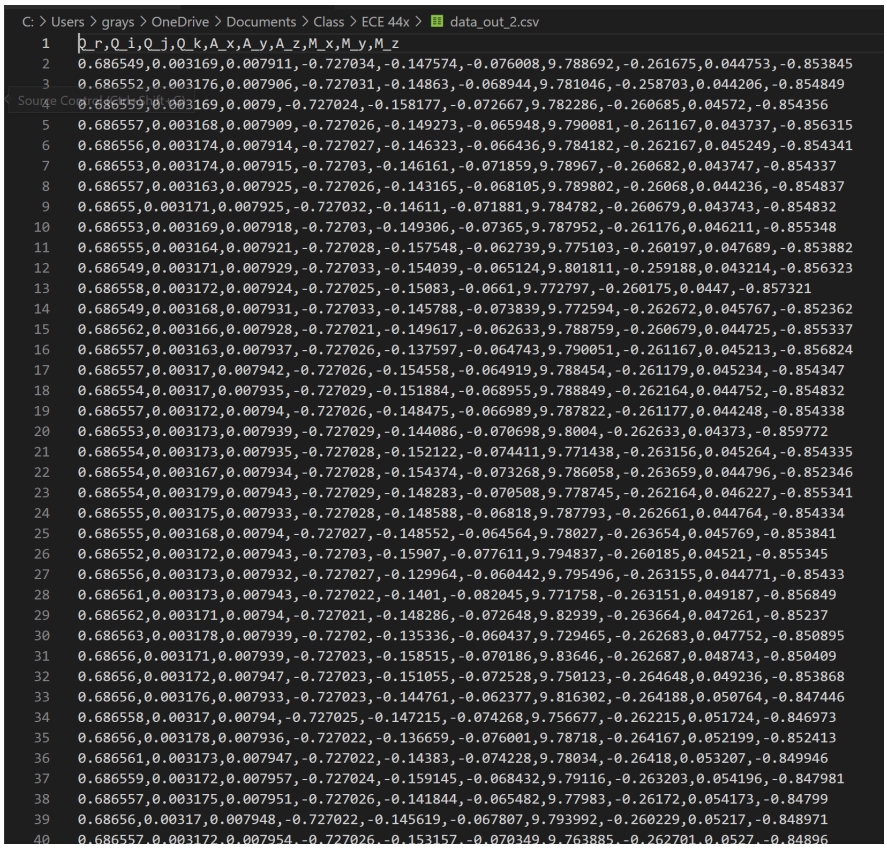

Figure 5.2.1.3: CSV file output of PC application

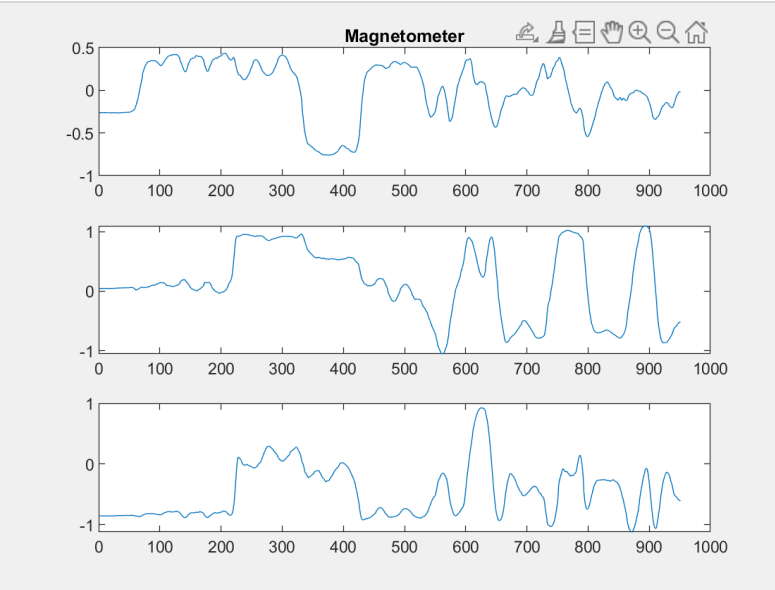

Figure 5.2.1.4: Magnetometer Plot from CSV file

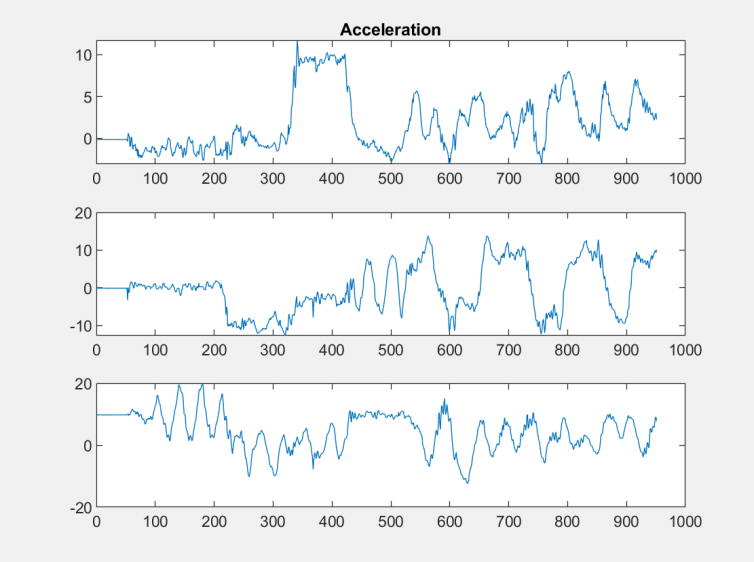

Figure 5.2.1.5: Acceleration Plot from CSV file

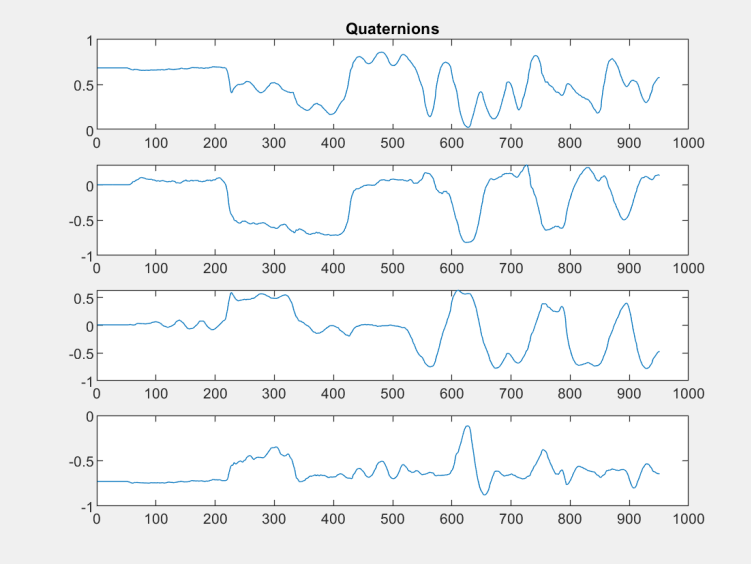

Figure 5.2.1.6: Quaternion (orientation) Plot from CSV file

Link to video demonstration of PC Application: https://youtu.be/6PUx6oeFvUs

5.1.3 If an enclosure is present, the contents must be ruggedly enclosed/mounted as evaluated by the course instructor.

The system will be directly mounted to the interior of the ocean glider, which is a waterproof environment that contains other bare PCBs to operate the glider. Because the glider itself will ruggedly protect our system, no enclosure is necessary.

5.1.4 If present, all wire connections to PCBs and going through an enclosure (entering or leaving) must use connectors.

See Figure 5.1.1.1 to view that all connections to the PCB are made with connectors.

5.1.5 All power supplies in the system must be at least 65% efficient.

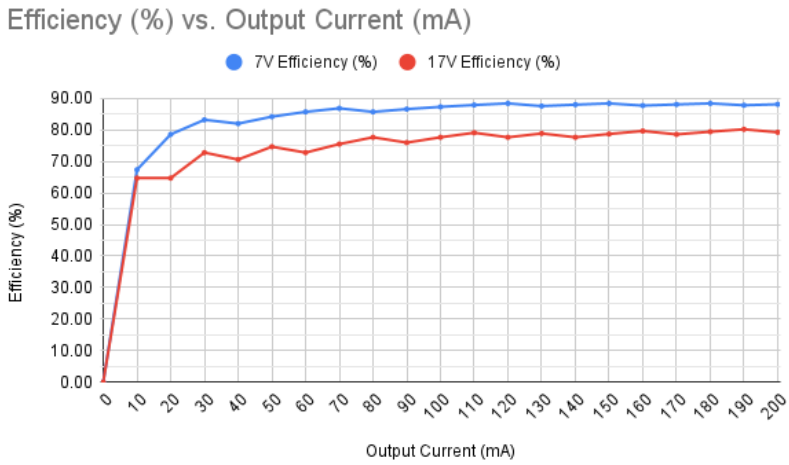

Figure 5.1.5.1: Buck converter efficiency plot.

This figure shows the tested efficiency of our power supply under different loads. In the relevant operating range (31mA nominal current), its efficiency is above 65% for both the 7V and 17V supply (~82% with a supply voltage of 7V and ~72% with a supply voltage of 17V).

5.1.6 The system may be no more than 50% pre-purchased modules.

The system only contains one pre-purchased module, the Xsens MTI-3 Accelerometer, which is soldered onto the system PCB.

5.2 PCB Size

5.2.1 Requirement

- *PPR*: Board must fit inside the science bay of the glider.
- *ER*: The microcontroller PCB must be smaller than 7 inches by 1.5 inches (and 2 inches tall) to fit onto the back of the acoustic modem board in the science bay of the glider.

5.2.2 Testing Process

Verification Process (Black Box):

5) Measure length of PCB.
6) Measure width of PCB.
7) Measure height of PCB.
8) Ensure that measurements are smaller than requirements (7 inches long, 1.5 inches wide, and 2 inches tall).

5.2.3 Testing Evidence

This Is a placeholder for the evidence that will be present in the next draft of this document.

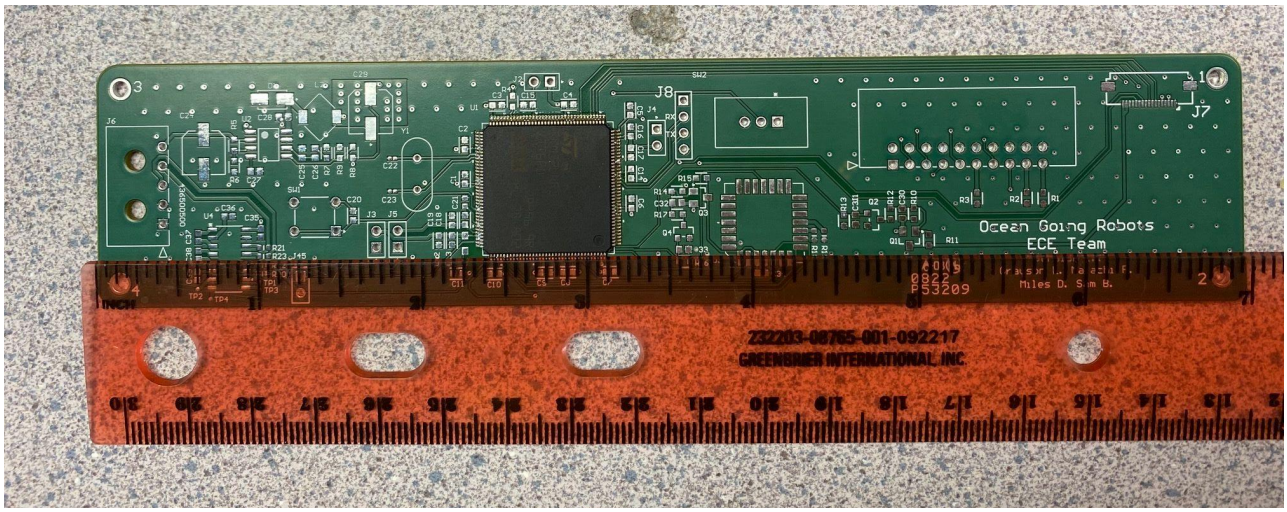

Figure 5.2.1: Measurement of PCB length (less than 7 inches).

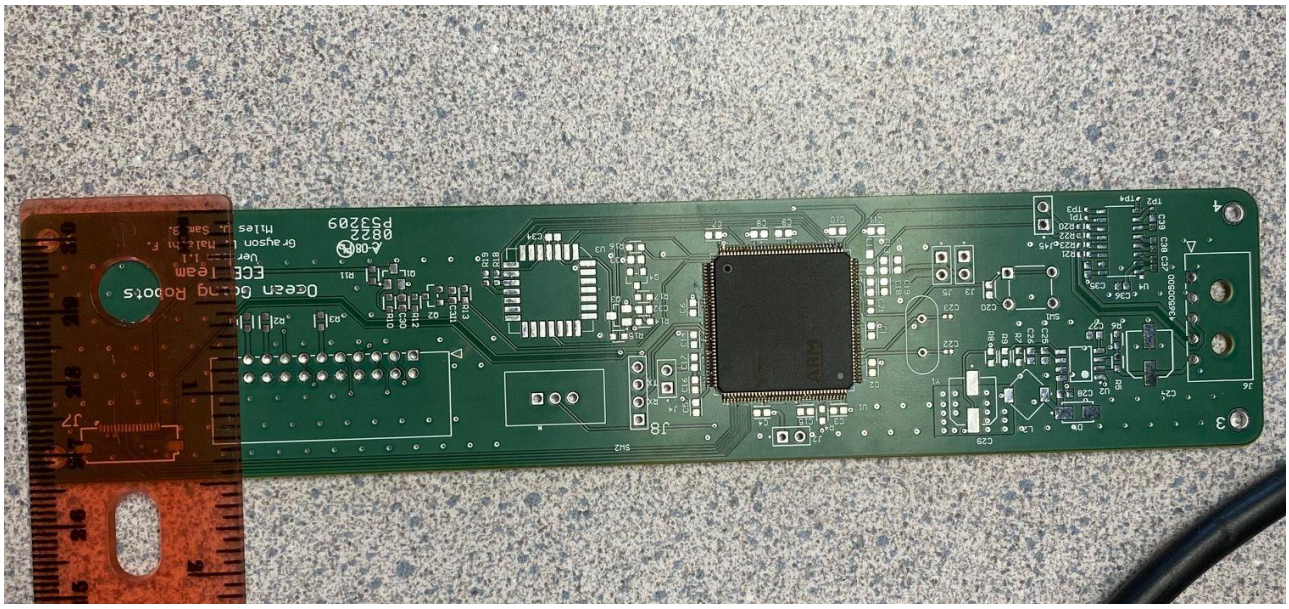

Figure 5.2.2: Measurement of the PCB width (less than 1.5 inches).

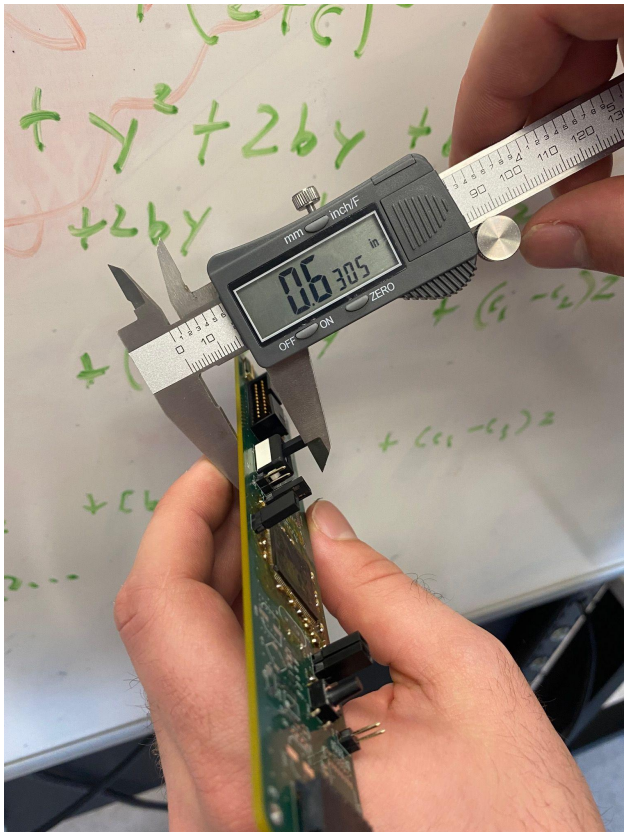

Figure 5.2.3: Measurement of the PCB height (less than two inches).

5.3 Data Collection and Analysis

5.3.1 Requirement

- PPR: Build a system that analyzes swell data for 20 minutes and sends it to the main science computer in the glider.
- ER: The system shall sample data at a maximum frequency of 10 Hz (though lower frequencies may be desired) for up to 20 minutes, and analyze the data collected to calculate the following parameters:
    - Hs, significant wave height in meters
    - Dom_period, dominant wave period in seconds
    - wave_dir, wave direction, magnetic
    - Hmax, maximum wave height in meters
    - Hmax2, second highest wave height
    - Pmax, maximum period
    - A1, spectral parameters
    - B1, spectral parameter
    - A2, spectral parameter
    - B2, spectral parameter index,
    - Wave component number

Note: The project partners may decide that some of these parameters are unnecessary and not require their calculation for the final system.

5.3.2 Testing Process

1. Power system and hook up to PC Application built for testing.
2. Run the system for 20 minutes.
3. Save the parameters calculated by the system
4. Ensure that there are at least 12,000 data samples measured and the last time stamp ensures that 20 minutes have elapsed.
5. Remove SD card from system and connect to PC
6. Use a script that has been approved by the project partners to manually calculate the parameters from the raw data that was collected.
7. Compare the parameters calculated by the system, and the parameters calculated by the script. This requirement will be met if the two calculations differ by no more than 5% for each parameter.

5.3.3 Testing Evidence

This Is a placeholder for the evidence that will be present in the next draft of this document.

5.4 Removable Data Storage

5.4.1 Requirement

- PPR: Collected data must be available in removable memory.
- ER: Data must be stored to a micro SD card.

5.4.2 Testing Process

1. Collect data for 10 minutes
2. Remove memory card
3. Check if the memory card has 6,000 samples.

5.4.3 Testing Evidence

This Is a placeholder for the evidence that will be present in the next draft of this document.

5.5 Glider Battery Life Impacts

5.5.1 Requirement

- PPR: Board must not reduce gliders battery life by more than 1 percent.
- ER: For a 20 minute collection cycle, the system must consume less than 230 Joules

5.5.2 Testing Process

1. Run a full 20 minute data collection cycle.
2. Measure the average voltage and current on the input of the module.
3. Use this data to calculate average power over the 20 minutes.
4. Integrate it to get the number of Joules.
5. The number of Joules should be less than 230.

5.5.3 Testing Evidence

This Is a placeholder for the evidence that will be present in the next draft of this document.

5.6 Data Ease of Access

5.6.1 Requirement

- PPR: Removable Memory must be easily accessible
- ER: SD card and associated PCB will have at least 0.5 inches of clearance from nearby military style screw connector and 0.25 inches of clearance from nearby white connectors.

5.6.2 Testing Process

1. Install SD card and associated PCB into the glider see Figure 2.1.
2. Measure the distance to the connectors.
3. This will be successful if the measurements are less than or equal to 0.5 inches from the connector with the screw on collar and 0.25 inches from the white plastic connectors.

5.6.3 Testing Evidence

The SD board was affixed to a 3D printed bracket which was installed into the glider with double sided tape to measure the connector clearances as was suggested by our project partners. In addition to directly measuring the clearances with a set of calipers, we also created a template out of cardstock the width of our minimum clearance dimensions that fits around the

bracket. By showing that the template fits between the connectors with the bracket and SD board mounted, our clearances can be easily verified visually.

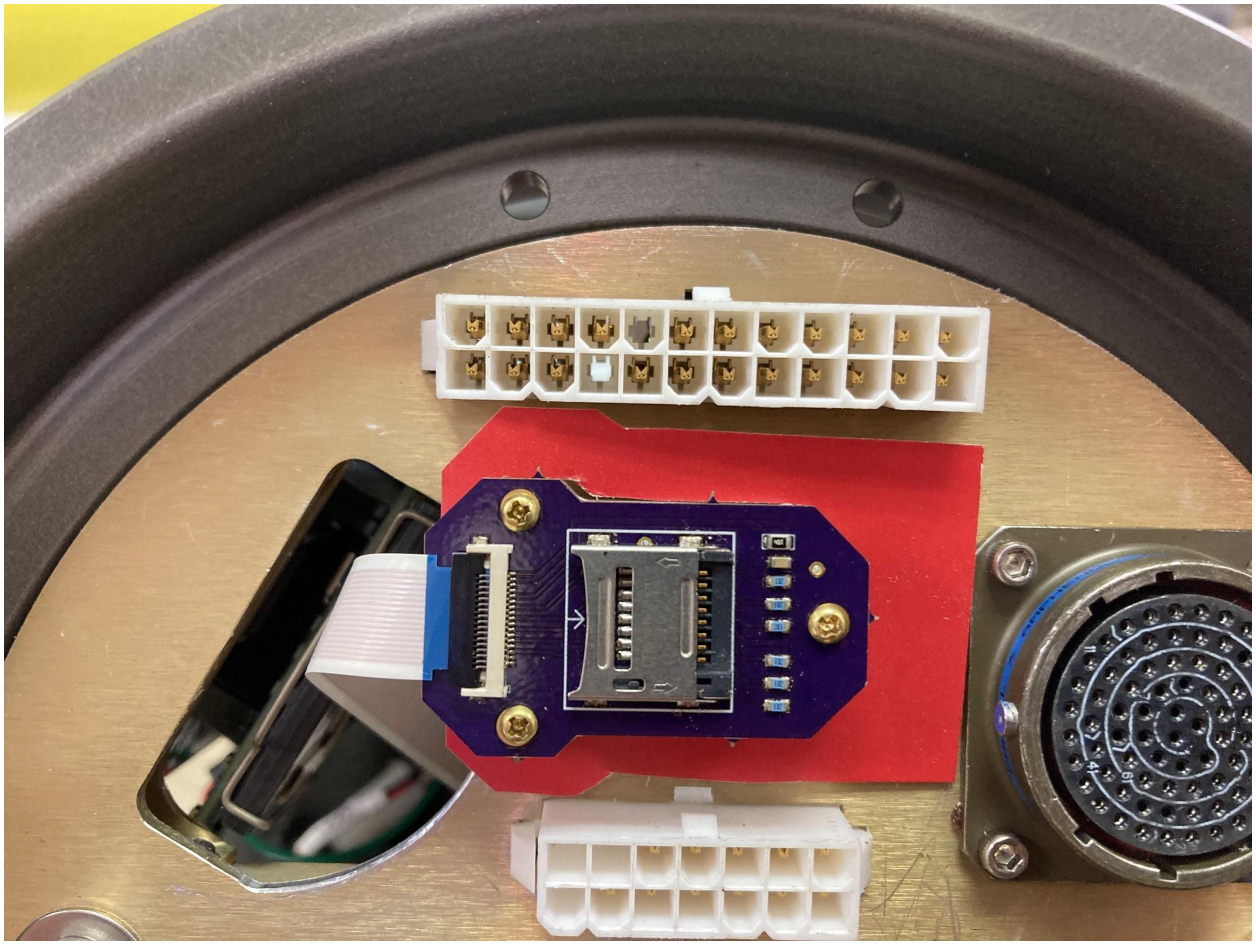

Figure 5.6.3.1: The SD card board mounted to the 3D printed bracket attached to the wall of the glider between the power and data connectors, with the red cardstock clearance template in place.

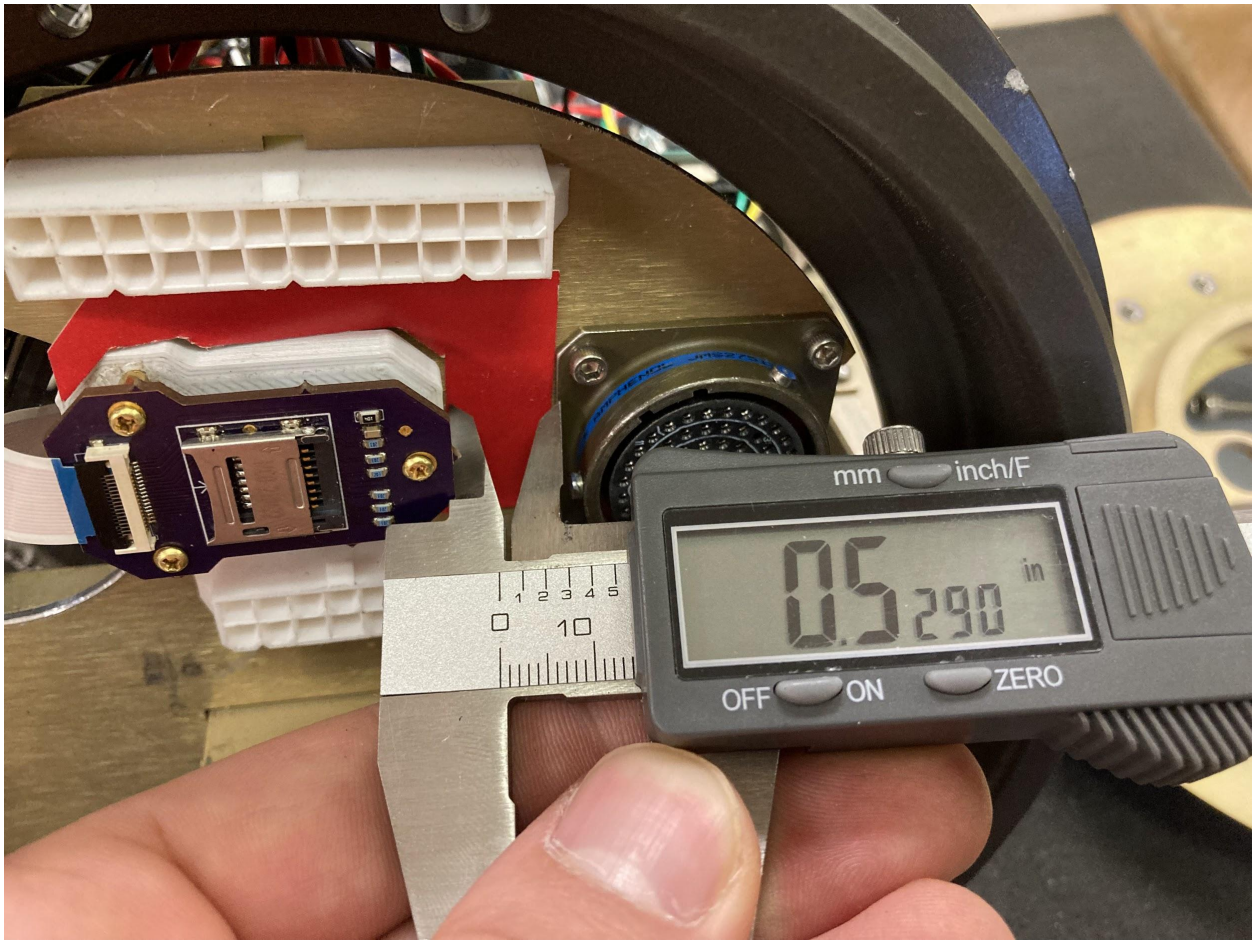

Figure 5.6.3.2: Demonstrating that the clearance requirement between the military style screw collared connector and the 3D printed bracket is greater than the required minimum of 0.5 inches.

Figure 5.6.3.3: Demonstrating that the clearance requirement between the top white snap connector and the 3D printed bracket is greater than the required minimum of 0.25 inches.

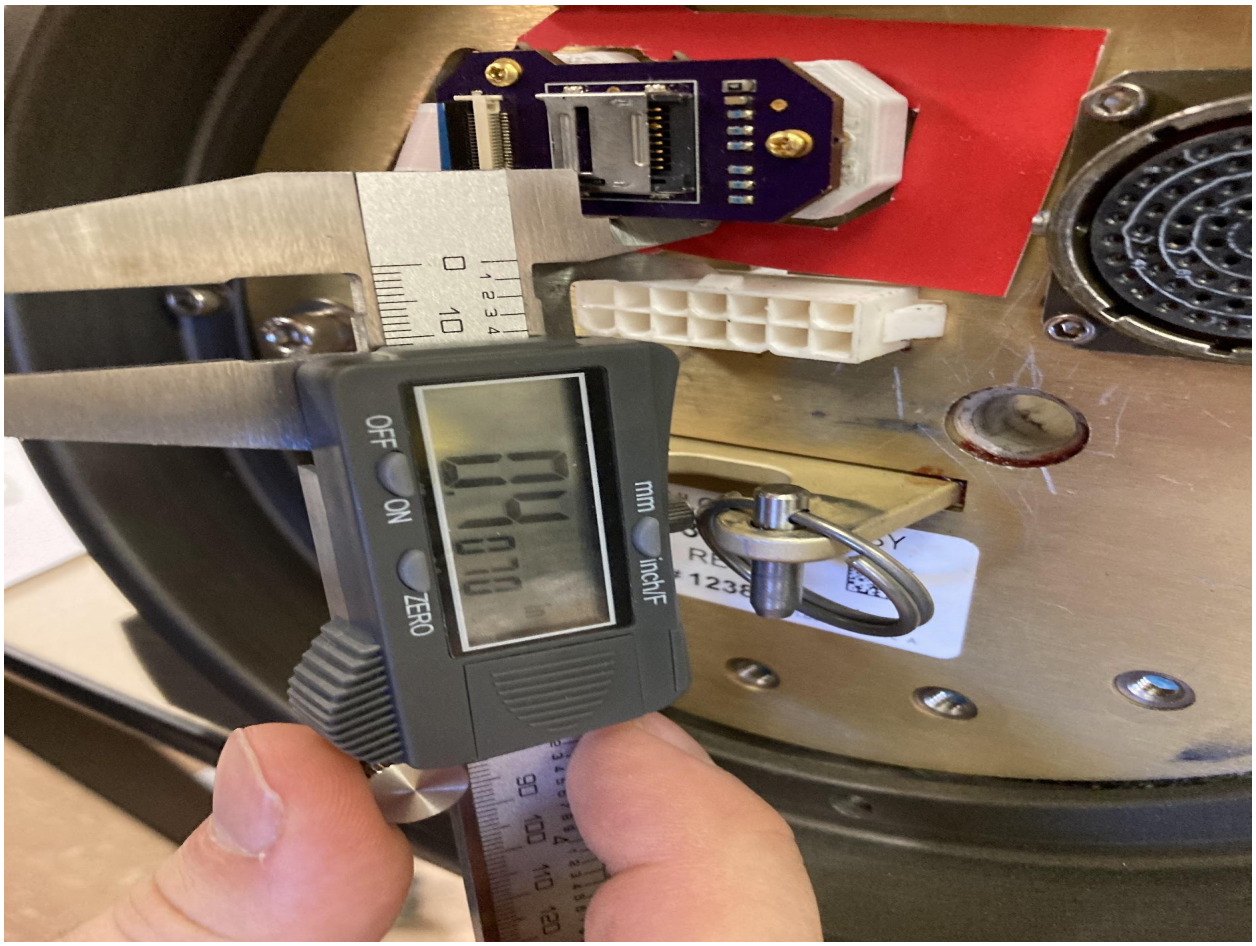

Figure 5.6.3.4: Demonstrating that the clearance requirement between the bottom white snap connector and the 3D printed bracket is greater than the required minimum of 0.25 inches.

Figure 5.6.3.5: Demonstrating that the width of the cardstock template is larger than minimum required tolerance between bracket and the connector, proving that the template can be used to visually verify the clearance requirement.

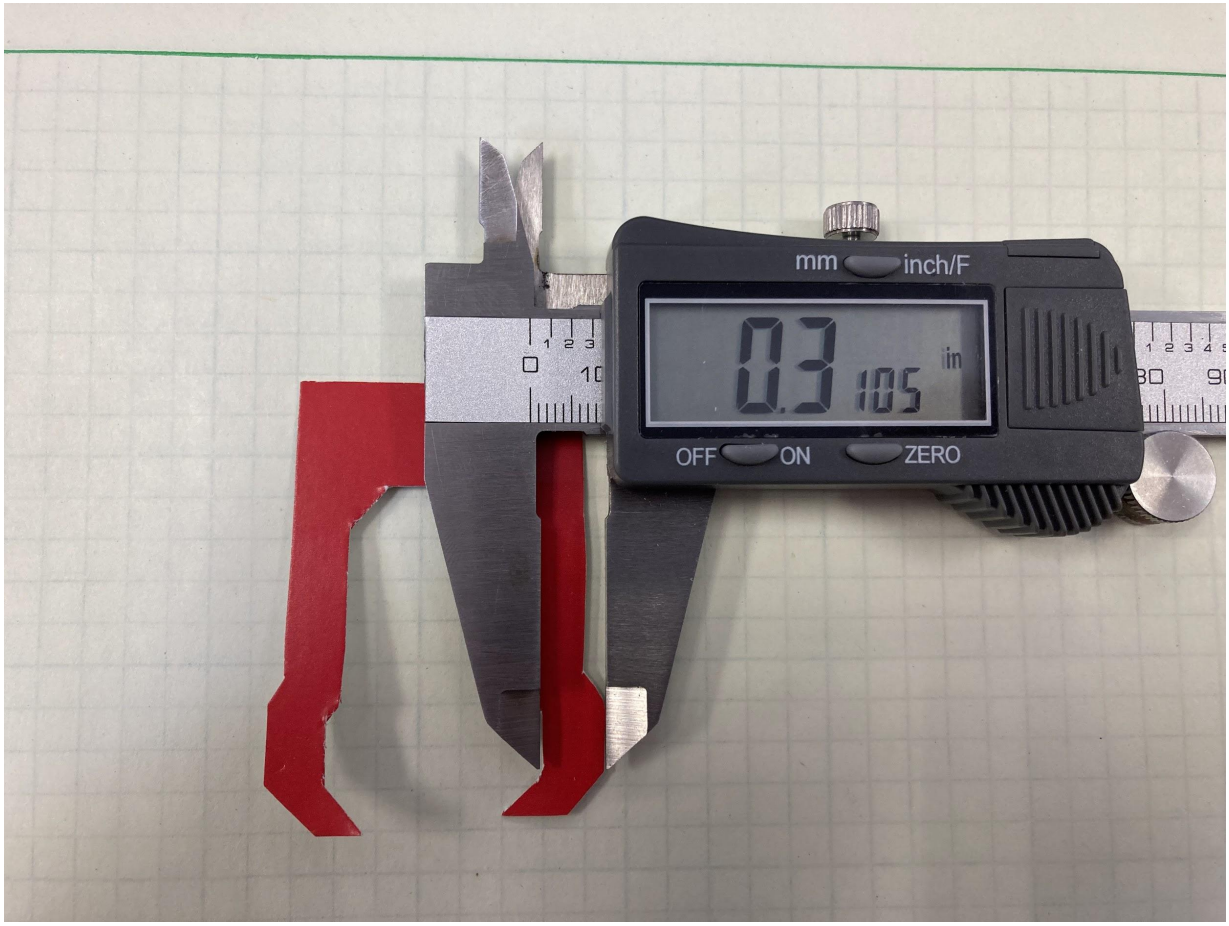

Figure 5.6.3.6: Demonstrating that the width of the cardstock template is larger than minimum required tolerance between bracket and the connector, proving that the template can be used to visually verify the clearance requirement.

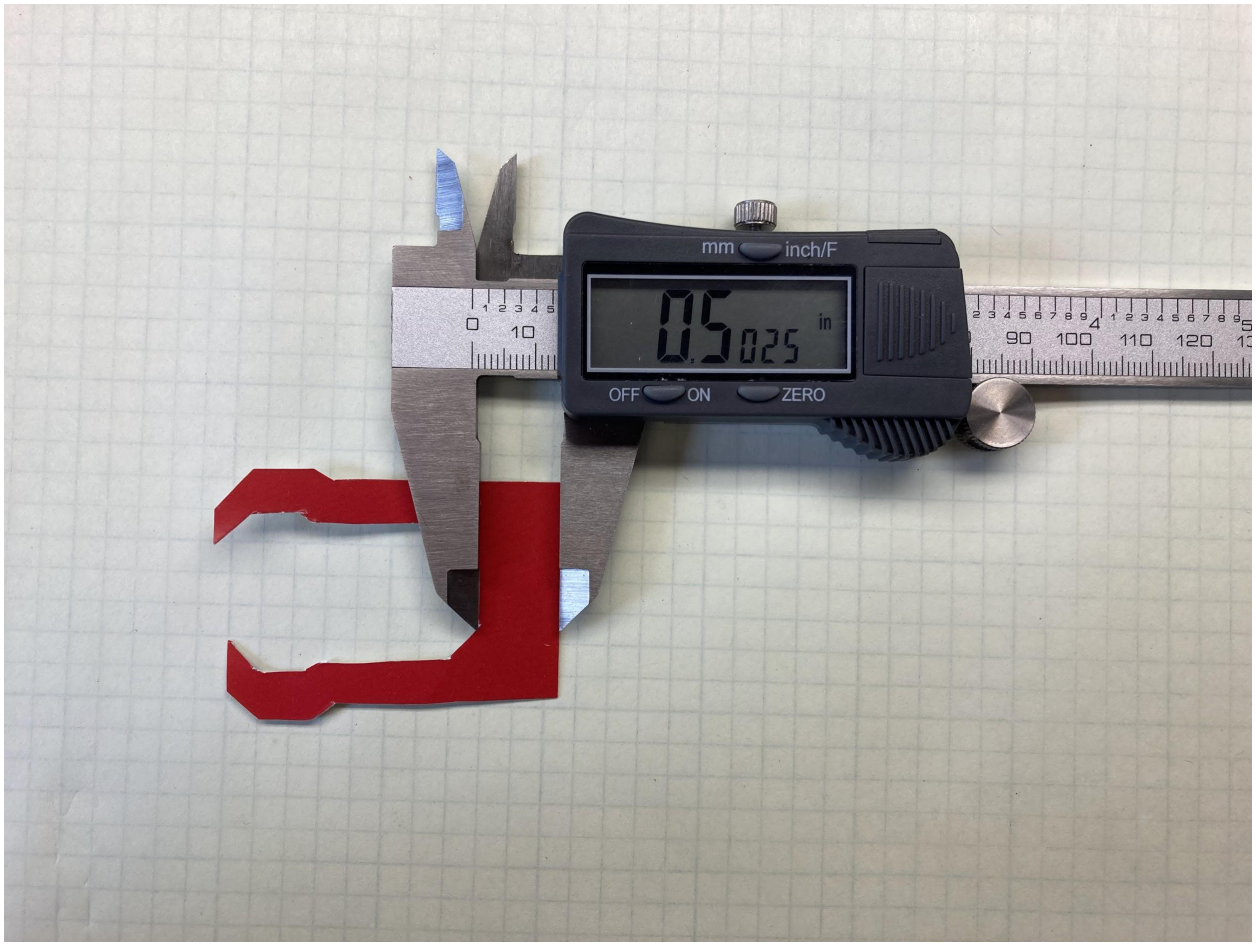

Figure 5.6.3.7: Demonstrating that the width of the cardstock template is larger than minimum required tolerance between bracket and the connector, proving that the template can be used to visually verify the clearance requirement.

5.7 Project Overview Wiki Page

5.7.1 Requirement

- PPR: The project must be thoroughly documented
- ER: A wiki page shall be created that includes:
    - A user guide
    - Firmware
    - Altium Designer files
    - Electrical Schematic(s)
    - PCB Fabrication files (gerbers)
    - Electrical Bill of Materials.

5.7.2 Testing Process

1. Write the initial draft of the Wiki Page and submit to the Project Partners by the Friday of Week 9 Winter 2022
2. Receive feedback from the Project Partners by the Friday of Week 10 Winter 2022
3. Write the second draft of the Wiki Page and submit to the Project Partners by the Friday of Week 1 Spring 2022
4. Receive feedback from the Project Partners by the Friday of Week 2 Spring 2022
5. Write the final draft of the Wiki Page and submit to the Project Partners by the Friday of Week 3 Spring 2022
6. If the project Partners find our Wiki Page sufficient, have each of them sign the final copy and return the document by Friday of Week 4 Spring 2022

5.7.3 Testing Evidence

This Is a placeholder for the evidence that will be present in the next draft of this document.
Ocean Going Robots (oregonstate.edu)

5.8 System Longevity

5.8.1 Requirement

- PPR: There should be no hardware or firmware bugs that cause the system to stop logging data while in use.
- ER: System must work properly for at least 90 days of continual use.

5.8.2 Testing Process

1. During a 90 day period, the system will be cycled 1080 times, a cycle consisting of the system being powered on, collecting data for 20 minutes, processing/storing the collected data, and then being powered off. To demonstrate the system will not encounter any errors in this 90 day period, we will cycle it 206 times, with a reduced data collection time.
2. For a 24 hour window, cycle 5 minutes of data capture and 2 minutes with the system powered off.
3. The system shall not encounter any hardware malfunctions or firmware errors.
4. All data collected will be stored on the SD card, and shall not contain any errors.

5.8.3 Testing Evidence

This Is a placeholder for the evidence that will be present in the next draft of this document.

5.9 System Security

5.9.1 Requirement

- PPR: System must be rugged to withstand the forces associated with being put into and taken out of the ocean.
- ER: The system shall function properly after a 10 foot drop. During the drop test the system shall be mounted in a test enclosure, and the system shall be powered off during the drop test.

5.9.2 Testing Process

1. Enclose the system in the test enclosure.
2. Drop enclosure from 10 ft above the surface of a water tank (specific tank TBD).
3. Remove system from test enclosure.
4. Connect system to power source.
5. Run the system for 5 minutes to ensure all 13 parameters listed above are being transmitted and there are at least 3000 data points measured and stored on the SD card.

5.9.3 Testing Evidence

This Is a placeholder for the evidence that will be present in the next draft of this document.

5.10 References and File Links

More links will be added as engineering requirements are verified.

5.11 Revision Table

Date	Revision

4/21/22	Updated wording of engineering requirements 5.1.6 and 5.1.8 for clarity and feasibility, with project partner's permission.
3/13/22	Malachi updated the phrasing of engineering requirements\ for section 5.3 at the suggestion of our project partners.
3/13/22	Updated engineering requirement for section 5.6 after getting approval from our project partners.
3/5/22	Initial creation, Malachi

Section 6: Project Closing

[To be completed at the end of the project]

Appendix A: