6/7/2016

Gabrielle Glynn and Demy Loulias  
TCES 482: Spring 2016

University of Washington - Tacoma

Spirit Lake Sensor Array

Final Report

Table of Contents

[2 Background Information 3](#_Toc452977200)

[2.1 Motivation For Project 3](#_Toc452977201)

[2.2 Identify Your Customer 4](#_Toc452977202)

[2.3 Devices Currently Available 4](#_Toc452977203)

[3 Problem Statement 5](#_Toc452977204)

[4 Customer Requirements 5](#_Toc452977205)

[5 Translating Customer Requirements 7](#_Toc452977206)

[5.1 Objective Analysis 7](#_Toc452977207)

[5.1.1 Objective: Battery Life 7](#_Toc452977208)

[5.1.2 Objective: Light Blocked By Device 7](#_Toc452977209)

[5.1.3 Objective: Quick Data Collection 8](#_Toc452977210)

[5.1.4 Objective: Quick Setup and Disassembly 8](#_Toc452977211)

[5.1.5 Objective: Easy Maintenance 9](#_Toc452977212)

[5.1.6 Objective: Lightweight 9](#_Toc452977213)

[5.1.7 Objective: Low Cost 9](#_Toc452977214)

[5.2 Constraint Analysis 10](#_Toc452977215)

[5.2.1 Constraint: Total Cost Cannot Exceed $2,000.00 10](#_Toc452977216)

[5.2.2 Constraint: Device Must Be Environmentally Friendly 10](#_Toc452977217)

[5.2.3 Constraint: Device Must Fit/Attach to Existing Platform 10](#_Toc452977218)

[5.3 Functional Analysis 10](#_Toc452977219)

[5.3.1 List of Specifications 11](#_Toc452977220)

[5.3.2 List of Means 12](#_Toc452977221)

[6 The Design Space 13](#_Toc452977222)

[6.1 Morph Chart 13](#_Toc452977223)

[6.2 Candidate Designs 14](#_Toc452977224)

[6.2.1 Candidate Design 1: Single Motor With SD Card 14](#_Toc452977225)

[6.2.2 Candidate Design 2: Multiple Stationary Sensors 14](#_Toc452977226)

[6.2.3 Candidate Design 3: Wireless Communication 15](#_Toc452977227)

[6.2.4 Candidate Design 4: Single Motor With Counterweight 16](#_Toc452977228)

[7 Evaluation of Design Alternatives 17](#_Toc452977229)

[7.1 Details and Drawings of Each Alternative 17](#_Toc452977230)

[7.1.1 Design 1: Single Motor With SD Card 17](#_Toc452977231)

[7.1.2 Design 2: Multiple Stationary Sensors 18](#_Toc452977232)

[7.1.3 Design 3: Wireless Communication 19](#_Toc452977233)

[7.1.4 Design 4: Single Motor With Counterweight 20](#_Toc452977234)

[7.2 Verify Constraints are Met 21](#_Toc452977235)

[7.3 How Well Do Designs Meet Objectives? 22](#_Toc452977236)

[7.4 How Well Do Designs Meet Functional Requirements? 22](#_Toc452977237)

[7.5 Method for Choosing Among Designs 23](#_Toc452977238)

[8 Preliminary Design Choice 23](#_Toc452977239)

[9 Testing 23](#_Toc452977240)

[9.1 How Will We Test the Device? 23](#_Toc452977241)

[9.2 How Do We Define Success? 23](#_Toc452977242)

[10 Social and Ethical Considerations 23](#_Toc452977243)

[11 Project Management 24](#_Toc452977244)

[11.1 Work Breakdown Schedule 24](#_Toc452977245)

[11.2 Schedule 27](#_Toc452977246)

[11.3 Budget 30](#_Toc452977247)

[12 Final Results 31](#_Toc452977248)

[12.1 Team Dynamics and Schedule 33](#_Toc452977249)

[12.2 Project Goals 34](#_Toc452977250)

[12.3 Updated Budget 34](#_Toc452977251)

[12.4 Testing 35](#_Toc452977252)

[12.5 What We Learned 37](#_Toc452977253)

[13 Appendix A: Research 38](#_Toc452977254)

[13.1 Library Search Results 38](#_Toc452977255)

[13.2 Other Search Results 38](#_Toc452977256)

# Background Information

## Motivation For Project

Dr. James Gawel, Associate Professor of Environmental Chemistry and Engineering at the University of Washington Tacoma, provided us with the background information necessary to understand the motivation for our project. From the project and budget proposal submitted to School of Interdisciplinary Arts and Sciences (SIAS) by Dr. Gawel on November 15, 2015:

The 1980 eruption of Mount St. Helens and the resulting massive landslide deposited much of the surrounding forest into Spirit Lake [located north of Mt. St. Helens in Washington State], where it has formed a floating log mat that covers close to one-fifth of the lake surface even 35 years after the blast. [Large woody debris (LWD)] decomposes slowly when submerged in water because many of the organisms that break down wood, such as fungi, require an oxygen-rich environment to function. This log mat provides an enormous surface area for the growth of biofilm, a complex mix of fungi, algae and bacteria. The constant grinding movement of the log mat results in the shedding of this biofilm and the subsequent input of carbon, nitrogen and phosphorus to the nutrient starved limnetic (open-water) areas of the lake.

Woody debris in stream ecosystems has garnered significant and increasing attention in the last few decades. Studies have shown that [LWD] such as logs, stumps, and large branches, are an important ecological component of streams (Richmond 1995). Although significant funding has been targeted at replacing LWD in stream ecosystems in the U.S. and elsewhere in response to the strength of evidence from research on its importance in the ecosystem, there has been little such research on the role of LWD in lakes. This may be due to the fact that humans have historically removed almost all LWD from accessible lakes for fuel and timber use and to improve boating conditions, potentially significantly reducing lake productivity and biodiversity. Therefore, understanding the role of LWD in lakes is of great importance for lake ecosystem management, especially in nutrient-poor systems including the montane areas so prevalent along the West coast of the U.S. and Canada.

This collaboration will hopefully lead to unmanned data collection in Spirit Lake under the log mat. With support from SIAS this past summer [2015], we created two experimental floating log mats tethered in Spirit Lake (in Duck Bay and Donnybrook Bay). We began data collection later in the summer. Because of the time and money required to make a trip to Spirit Lake, our data is limited in temporal scope. In order to better understand the ecological function of the logs, remote data collection and storage, to the extent possible, is desirable.

The deployment package [an autonomous sensor array lasting with enough power for about 30 days] will allow us to collect multiple vertical water column data profiles each day (depth, temperature, dissolved oxygen, specific conductivity, pH, light intensity, fish presence/numbers, and possibly others) using our existing multi-parameter water quality probe and other sensors along with a mechanical yoyo system, store that data on-site, and charge the package for over a month of work. We usually visit Spirit Lake once a month, and so can switch out components as needed at that time, download data and clean sensors.

In summary, the development of an unmanned sensor array will allow for more meaningful data collection of the Spirit Lake log boom that will hopefully lead to a greater understanding of the ecological impact of the logs.

## Identify Your Customer

Our “client” is Dr. Jim Gawel (as stated earlier), an Associate Professor of Environmental Chemistry and Engineering at the University of Washington Tacoma. We are working collaboratively with Dr. Gawel and the School of Interdisciplinary Arts and Sciences (SIAS) in order to build an autonomous sensor to fit the needs of his research at Spirit Lake. The concept came about through several conversations between Dr. Gawel and Bob Landowski, Laborator Support Engineer for the University of Washington Tacoma Institute of Technology. Our project is being funded by SIAS.

## Devices Currently Available

The current way Dr. Gawel and his team collects data on the water quality of Spirit Lake is by hiking to the lake about once per month and manually lowering a sensor to different depths and recording the information off the log platforms they built (see Figure 1). The sensor used is a Hydrolab Quanta Water Quality Monitoring System that includes a SDI-12 (Serial Digital Interface at 1200 baud) transmission system for the sensor and a display. This sensor is able to be deployed for long-term monitoring and can measure temperature, pH, dissolved oxygen (DO), specific conductance (SpC), depth, oxidation-reduction potential (ORP), turbidity, salinity, and total dissolved solids (TDS). It is currently discontinued, and although data-loggers are available that are compatible with the sensor they are expensive and do not include the ability to take readings at multiple depths.

The end of this past year (2015) Dr. Gawel was able to purchase another water quality sensor, the smarTROLL Multiparameter Handheld system, which collects a greater range of water quality parameters than the Quanta, is lighter (a consideration when trying to conserve power lowering and raising the sensor autonomously to different depths), and can send data wirelessly to an app for Android or IOS (there is no other way to capture the measurements).

Other autonomous long-term water quality data loggers are available for purchase but are generally very expensive, and are not as customizable. We hope to build a sensor array that can fulfill the exact requirements even if they change over the next few years. For example, Dr. Gawel has expressed the desire for a future project in collaboration with the Institute that will included the ability of the sensor array more mobility so that it can take measurements while staying with the movement of the log boom.



Figure 1: Picture of one of the log platforms constructed on Spirit Lake

# Problem Statement

Dr. Gawel, an Environmental Chemistry Professor at UWT, wants to further study the ecological effects of the log boom in Spirit Lake that resulted from the 1980 eruption of Mt. St. Helens. Currently the remote site is only accessed once a month from June to October, and sensor readings can only be taken then. Dr. Gawel is seeking a sensor array device that can be attached to a floating log mat to obtain daily readings without an increase in visits to the site. The sensor will still need to be calibrated every 30-35 days, so data can be stored and collected at the same time as the calibration. The system’s power needs to last 30-35 days, and the battery should be light enough to carry on a 3-mile hike, so ideally the system will be able to recharge the battery using renewable means to prolong the battery life. The system needs to move the sensor to take readings every 5 meters from 1 to 3 times per day. The system will use a water quality sensor that the professor already owns.

# Customer Requirements

Through several interviews and communications with Dr. Gawel from October 2015 through the present (March 2016), we were able to develop a chart that captures the objectives, functions, and constraints of the proposed sensor platform. See the chart below for a summary. The chart has been updated since our first presentation in January 2016.

Table 1: Objectives, Functions, and Constraints Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Characteristics** | **Objective** | **Constraint** | **Function** |
| The device should have a battery life of greater than 50 days. | + |  |  |
| The device should block as little light as possible. | + |  |  |
| Collecting data from the device should be fast and convenient. | + |  |  |
| The one-time setup and disassembly should be fast and convenient. | + |  |  |
| The device should be convenient to maintain. | + |  |  |
| Whole device should be lightweight and portable. | + |  |  |
| Total cost should be less than $2000 (excludes sensor with microprocessor). | + |  |  |
| Total cost cannot exceed $2000 (not including the sensor with microprocessor) |  | + |  |
| Device must be in compliance with environmental regulations. |  | + |  |
| The device must fit onto existing platform. |  | + |  |
| The device will take water quality readings (temperature, pH, dissolved oxygen (DO), specific conductance (SpC), depth, oxidation-reduction potential (ORP), turbidity, salinity, chlorophyll and total dissolved solids (TDS)). |  |  | + |
| Device will time-stamp all readings. |  |  | + |
| Device will be user-programmable. |  |  | + |
| Device will take readings at different depths. |  |  | + |
| The sensor array will attempt to prolong battery life. |  |  | + |
| The device will save data persistently in case of power failure so readings are not lost. |  |  | + |
| Device will be protected from water. |  |  | + |
| Device will protect the sensor transmission cable from bending. |  |  | + |
| Device will provide power for up to 35 days at a time. |  |  | + |
| Device will take readings autonomously. |  |  | + |

# Translating Customer Requirements

## Objective Analysis



Figure 2: Objectives Diagram

### Objective: Battery Life

While our ultimate goal is to construct the sensor array to last a period of 35 days with readings taken 3 times per day, a device that lasts for part of that time would still be acceptable given the current limitations of the research team to consistently take readings. Therefore, even a partial battery life would be acceptable. See the table below.

Table 2: Battery Life Metrics

|  |  |  |
| --- | --- | --- |
| **Time** | **Perceived Value** | **Points Awarded** |
| < 5 days | Unsatisfactory | 1 |
| 5-10 days | Tolerable | 3 |
| 10–20 days | Adequate | 5 |
| 20-30 days | Good | 7 |
| > 30 days | Ideal | 10 |

### Objective: Light Blocked By Device

If the device blocks too much of the light that would naturally fall through the log platform, then some of the readings (specifically the reading for chlorophyll) would not be as representative of the lake’s natural condition. When constructing the device we must be cognizant of its impact on the ecology of the lake and the water underneath the log boom. See the table below.

Table 3: Light Blocked Metric

|  |  |  |
| --- | --- | --- |
| **Percent under instrument** | **Perceived Value** | **Points Awarded** |
| 100 – 80% | Unsatisfactory | 1 |
| 80 – 60% | Tolerable | 3 |
| 60 – 40% | Adequate | 5 |
| 40 – 20% | Good | 7 |
| 20 – 0% | Ideal | 10 |

### Objective: Quick Data Collection

Our goal is to make collecting the saved data from the device as easy as possible since so much energy will be spent just to get to the lake. Collecting the data must be easy and clearly marked. Access to the platform also requires the use of a boat so eliminating any unnecessary difficulty is ideal. See table below for our time estimates.

Table 4: Data Collection Metric

|  |  |  |
| --- | --- | --- |
| **Time to Access Data** | **Perceived Value** | **Points Awarded** |
| > 30 minutes | Unsatisfactory | 1 |
| 30 – 20 minutes | Tolerable | 3 |
| 20 – 10 minutes | Adequate | 5 |
| 10 – 5 minutes | Good | 7 |
| < 5 minutes | Ideal | 10 |

### Objective: Quick Setup and Disassembly

Similar to our reasons for making data collection as easy and quick as possible, the setup and disassembly time for the sensor array should be as convenient as possible. Getting to this remote location is already time consuming. However, setup up time is expected to be somewhat longer since it will require initial calibration of the sensors and attaching/detaching the device to the log platform.

Table 5: Setup Time Metric

|  |  |  |
| --- | --- | --- |
| **Time for Setup** | **Perceived Value** | **Points Awarded** |
| > 240 minutes | Unsatisfactory | 1 |
| 240 – 120 minutes | Tolerable | 3 |
| 120 – 60 minutes | Adequate | 5 |
| 60 – 30 minutes | Good | 7 |
| < 30 minutes | Ideal | 10 |

### Objective: Easy Maintenance

About once a month the research team will be coming during the summer to collect the data and service the sensor array. The sensors will be recalibrated, batteries checked and changed out if necessary, the mechanical components checked for integrity…making this process as easy and quick is possible will help make our device more user friendly and convenient.

Table 6: Maintenance Time Metric

|  |  |  |
| --- | --- | --- |
| **Time for Maintenance** | **Perceived Value** | **Points Awarded** |
| > 120 minutes | Unsatisfactory | 1 |
| 120 – 90 minutes | Tolerable | 3 |
| 90 – 60 minutes | Adequate | 5 |
| 60 – 30 minutes | Good | 7 |
| < 30 minutes | Ideal | 10 |

### Objective: Lightweight

Since Spirit Lake is accessed by a three mile hike, we would like to make the whole of the device as light as possible. The heavier it is the less convenient and manageable it will be.

Table 7: Weight Metric

|  |  |  |
| --- | --- | --- |
| **Weight of System** | **Perceived Value** | **Points Awarded** |
| > 50 lbs. | Unsatisfactory | 1 |
| 50 – 40 lbs. | Tolerable | 3 |
| 40 – 20 lbs. | Adequate | 5 |
| 20 – 10 lbs. | Good | 7 |
| < 10 lbs. | Ideal | 10 |

### Objective: Low Cost

The prototype we are attempting to construct may be the first of several devices desired for Spirit Lake. There are already two log platforms constructed, so making the first prototype as cost effective as possible will make multiple devices more possible. At this time we have been approved for about $2000 (not including the sensors already owned by UW). The deadline has passed to request more money.

Table 8: Cost Metric

|  |  |  |
| --- | --- | --- |
| **Cost of System** | **Perceived Value** | **Points Awarded** |
| > $2,000 | Unsatisfactory | 1 |
| $2,000 - $1,500 | Tolerable | 3 |
| $1,500 - $1,000 | Adequate | 5 |
| $1,000 - $500 | Good | 7 |
| < $500 | Ideal | 10 |

## Constraint Analysis

### Constraint: Total Cost Cannot Exceed $2,000.00

We did not submit a request for additional funds since the $2000.00 our project was originally rewarded, according to our budget, will be enough to cover our expenses for the design we have chosen (more details in the following sections). We must keep the total cost for new components under this amount.

### Constraint: Device Must Be Environmentally Friendly

Spirit Lake is protected and its natural ecosystem cannot be disturbed by our sensor array. We must construct our device to be sturdy to prevent any debris from being released into the lake, as well as make sure the batteries we use will not leak and contaminate the lake.

### Constraint: Device Must Fit/Attach to Existing Platform

The first prototype must be attached to one of the already constructed log platforms. It will be acceptable to drill into the platform to attach our array.

## Functional Analysis

Below is the black box diagram we have constructed to frame the way we think about the inputs and outputs of our device. Going into our controller is power, user specifications regarding the readings that will be taken (how many, when, at what depths, etc.) and the readings from the sensor. Our controller will take these inputs and output the data in some form (we have chosen a CSV file after confirming this form would be acceptable to our client) and the movement of the sensor to different depths.

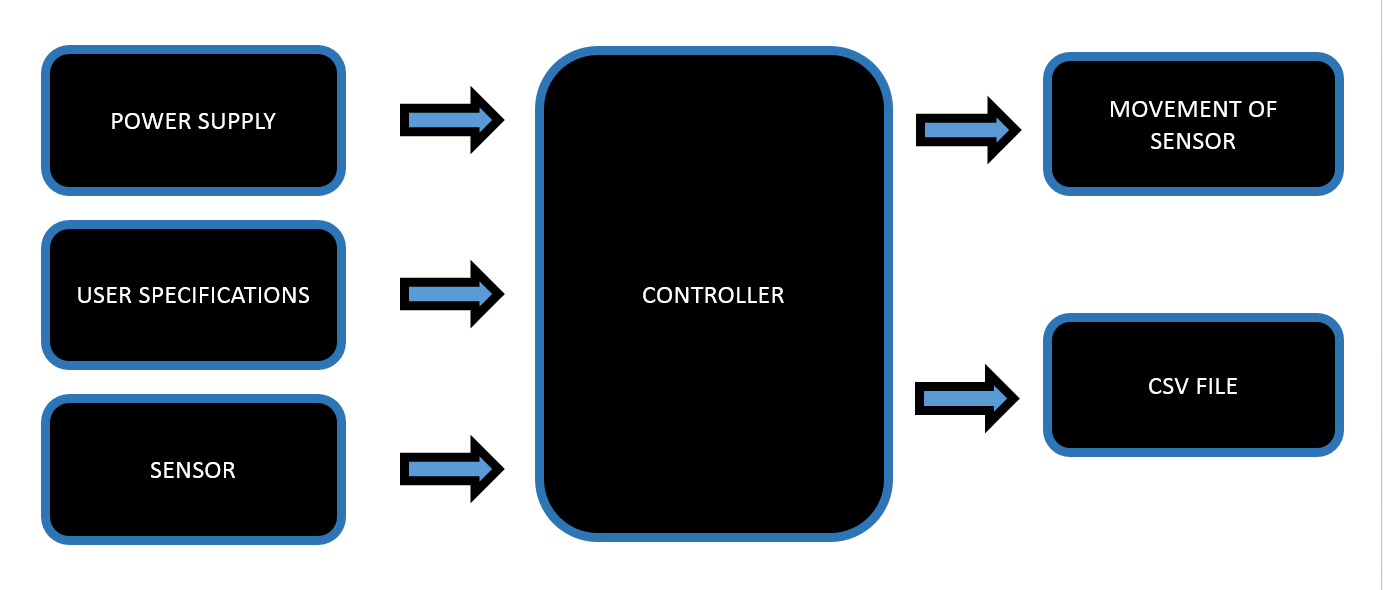


Figure 3: Black Box

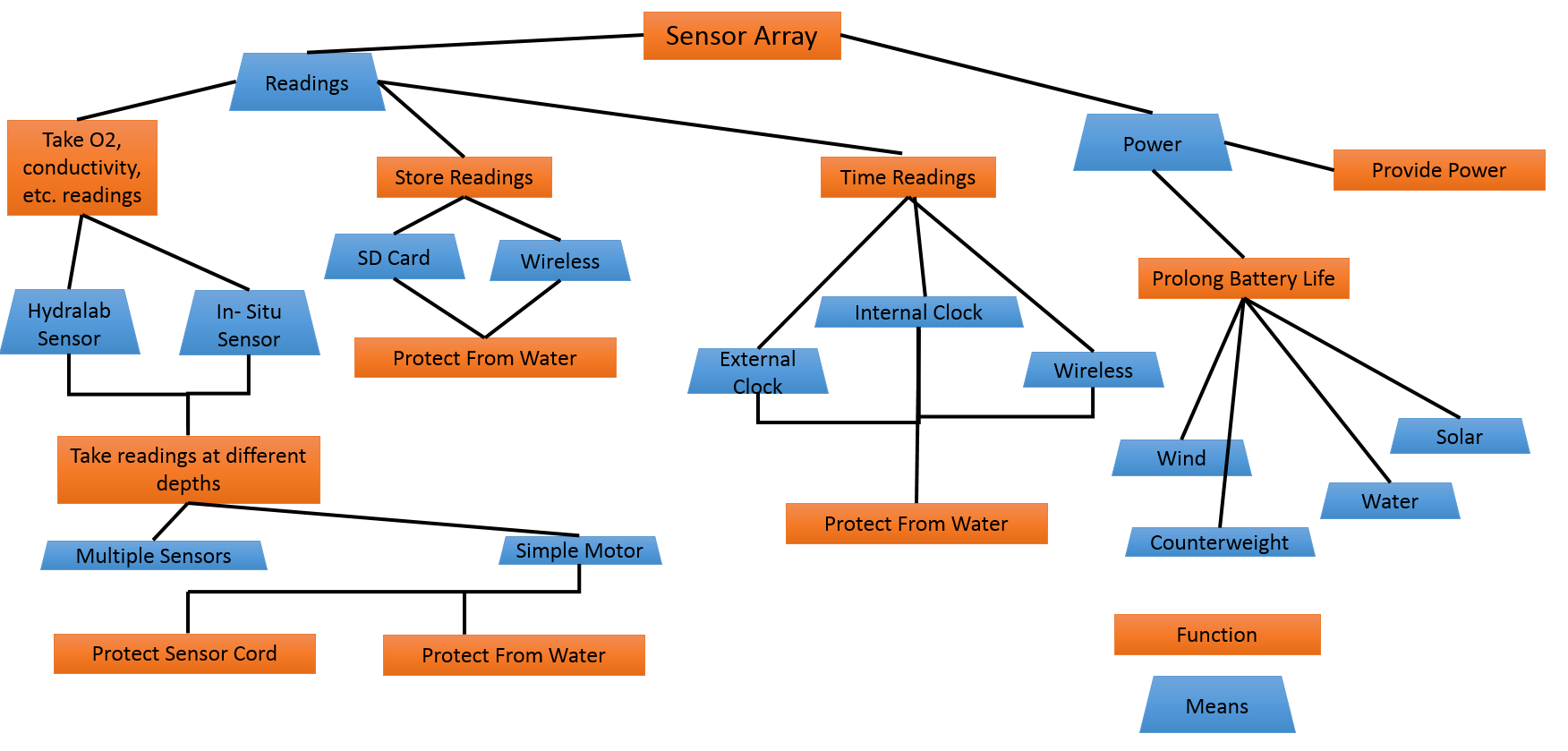


Figure 4: Function Means Tree

### List of Specifications

See table below for list of performance specifications for each function.

Table 9: Performance Specifications

|  |  |
| --- | --- |
| **Function** | **Performance Specifications** |
| The device will take water quality readings (temperature, pH, dissolved oxygen (DO), specific conductance (SpC), depth, oxidation-reduction potential (ORP), turbidity, salinity, chlorophyll and total dissolved solids (TDS)). | This should be achieved using one of the two sensors provided by the sponsor. The Quanta uses SDI-12 communication protocol and requires an SDI-12 transmission adapter cable to interface with a microcontroller or processor. The smarTROLL sensor requires an Android or IOS app downloadable from the company in order to display sensor data. |
| Device will time-stamp all readings. | Readings need to be time-stamped with the date and time (PST) for when they are taken. |
| Device will be user-programmable. | The system should allow the user to set the number of readings and the times they want them taken. |
| Device will take readings at different depths. | Readings should be taken every 5 meters. |
| The sensor array will attempt to prolong battery life. | The battery should be able to extend battery power using either solar, wind, water power or a counterweight. |
| The device will save data persistently in case of (cont.) power failure so readings are not lost. | Data needs to persist. It also needs to persist in the event of a power failure. |
| Device will be protected from water. | The system needs to be protected from water so that is will function as intended. |
| Device will protect the sensor transmission cable from bending. | The sensor cord needs to be protected when it is not fully extended rather than dangling in the water. |
| Device will provide its own power. | Device should be powered for up to 35 days. |
| Device will be autonomous. | The whole sensor array should be able to communicate with the sensor to take readings at prearranged times and depths, record them, keep time, and regulate power usage without human supervision regular intervals. Readings should begin to be taken at the user-programmed times which will be approximately the same each day. Exact time matching is not required. |

### List of Means

1. Take water quality readings
2. Hydrolab Quanta sensor
3. smarTROLL Multiparameter Handheld sensor
4. Time-stamp all readings
   1. External, battery-powered, clock with shield
   2. Internal microcontroller clock
   3. Wireless sync
      1. Ham radio / satellite
      2. XBees
5. User-programmable
   1. LCD Screen and buttons
   2. Wireless app
   3. Physical connection to phone or computer
6. Take readings at different depths
   1. DC Motor
   2. Multiple sensors
7. Prolong battery life
   1. Solar
   2. Wind
   3. Water current
   4. Counterweight to sensor
8. Data Persistence
   1. SD card
   2. Non-volatile memory
   3. Wireless sync
      1. Ham radio / satellite
      2. XBees
9. Protect from water
   1. Plastic casing
   2. Metal casing
   3. Wooden casing
10. Protect the sensor transmission cable
    1. Self-coiling cord casing
    2. Spring loaded hose-reel
    3. Motor-driven reel
11. Provide power
    1. One rechargeable LiPo battery
    2. Multiple rechargeable LiPo batteries
12. Autonomous take readings
    1. MSP430
    2. Arduino Mega 2560
    3. Intel Edison
    4. Raspberry Pi

# The Design Space

## Morph Chart

Below is the morphological chart that provides the framework for our design space.

Table 10: Morphological Chart

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function/ Means** | **1** | **2** | **3** | **4** |
| Take readings | Hydrolab Quanta Sensor | In-Situ smarTROLL Sensor |  |  |
| Time-stamped readings | External clock | Clock on microprocessor | Wireless (Ham/Satellite) | Wireless (Line of Sight, XBees) |
| User programmable | Knobs, buttons LCD screen | Wireless app | Physically connect to computer |  |
| Take readings at different depths | Motor and gearing | Multiple sensors |  |  |
| Prolong battery life | Wind | Solar | Water | Counterweight |
| Data persistence | SD card | Wireless (Ham/Satellite) | Wireless (Line of Sight, XBees) | Non-volatile memory |
| Water protection | Plastic casing | Metal casing | Wood casing |  |
| Protect sensor cord | Self-coiling casing | Spring-loaded hose-reel | Motor-driven reel |  |
| Provide power | 1 rechargeable battery (LiPo) | multiple rechargeable batteries (LiPo) |  |  |
| Autonomously take readings at specified times | MSP-430 | Intel Edison | Raspberry Pi | Arduino Mega 2560 |

## Candidate Designs

Below there are four copies of our morphological chart with the means we have chosen for each particular design highlighted. Green highlight indicates the means choice for the function.

### Candidate Design 1: Single Motor With SD Card

Table 11: Candidate Design 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function/ Means** | **1** | **2** | **3** | **4** |
| Take readings | Hydrolab Quanta Sensor | In-Situ smarTROLL Sensor |  |  |
| Time-stamped readings | External clock | Clock on microprocessor | Wireless (Ham/Satellite) | Wireless (Line of Sight, XBees) |
| User programmable | Knobs, buttons LCD screen | Wireless app | Physically connect to computer |  |
| Take readings at different depths | Motor and gearing | Multiple sensors |  |  |
| Prolong battery life | Wind | Solar | Water | Counterweight |
| Data persistence | SD card | Wireless (Ham/Satellite) | Wireless (Line of Sight, XBees) | Non-volatile memory |
| Water protection | Plastic casing | Metal casing | Wood casing |  |
| Protect sensor cord | Self-coiling casing | Spring-loaded hose-reel | Motor-driven reel |  |
| Provide power | 1 rechargeable battery (LiPo) | multiple rechargeable batteries (LiPo) |  |  |
| Autonomously take readings at specified times | MSP-430 | Intel Edison | Raspberry Pi | Arduino Mega 2560 |

### Candidate Design 2: Multiple Stationary Sensors

Table 12: Candidate Design 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function/ Means** | **1** | **2** | **3** | **4** |
| Take readings | Hydrolab Quanta Sensor | In-Situ smarTROLL Sensor |  |  |
| Time-stamped readings | External clock | Clock on microprocessor | Wireless (Ham/Satellite) | Wireless (Line of Sight, XBees) |
| User programmable | Knobs, buttons LCD screen | Wireless app | Physically connect to computer |  |
| Take readings at different depths | Motor and gearing | Multiple sensors |  |  |
| Prolong battery life | Wind | Solar | Water | Counterweight |
| Data persistence | SD card | Wireless (Ham/Satellite) | Wireless (Line of Sight, XBees) | Non-volatile memory |
| Water protection | Plastic casing | Metal casing | Wood casing |  |
| Protect sensor cord | Self-coiling casing | Spring-loaded hose-reel | Motor-driven reel |  |
| Provide power | 1 rechargeable battery (LiPo) | multiple rechargeable batteries (LiPo) |  |  |
| Autonomously take readings at specified times | MSP-430 | Intel Edison | Raspberry Pi | Arduino Mega 2560 |

### Candidate Design 3: Wireless Communication

Table 13: Candidate Design 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function/ Means** | **1** | **2** | **3** | **4** |
| Take readings | Hydrolab Quanta Sensor | In-Situ smarTROLL Sensor |  |  |
| Time-stamped readings | External clock | Clock on microprocessor | Wireless (Ham/Satellite) | Wireless (Line of Sight, XBees) |
| User programmable | Knobs, buttons LCD screen | Wireless app | Physically connect to computer |  |
| Take readings at different depths | Motor and gearing | Multiple sensors |  |  |
| Prolong battery life | Wind | Solar | Water | Counterweight |
| Data persistence | SD card | Wireless (Ham/Satellite) | Wireless (Line of Sight, XBees) | Non-volatile memory |
| Water protection | Plastic casing | Metal casing | Wood casing |  |
| Protect sensor cord | Self-coiling casing | Spring-loaded hose-reel | Motor-driven reel |  |
| Provide power | 1 rechargeable battery (LiPo) | multiple rechargeable batteries (LiPo) |  |  |
| Autonomously take readings at specified times | MSP-430 | Intel Edison | Raspberry Pi | Arduino Mega 2560 |

### Candidate Design 4: Single Motor With Counterweight

Table 14: Candidate Design 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function/ Means** | **1** | **2** | **3** | **4** |
| Take readings | Hydrolab Quanta Sensor | In-Situ smarTROLL Sensor |  |  |
| Time-stamped readings | External clock | Clock on microprocessor | Wireless (Ham/Satellite) | Wireless (Line of Sight, XBees) |
| User programmable | Knobs, buttons LCD screen | Wireless app | Physically connect to computer |  |
| Take readings at different depths | Motor and gearing | Multiple sensors |  |  |
| Prolong battery life | Wind | Solar | Water | Counterweight |
| Data persistence | SD card | Wireless (Ham/Satellite) | Wireless (Line of Sight, XBees) | Non-volatile memory |
| Water protection | Plastic casing | Metal casing | Wood casing |  |
| Protect sensor cord | Self-coiling casing | Spring-loaded hose-reel | Motor-driven reel |  |
| Provide power | 1 rechargeable battery (LiPo) | multiple rechargeable batteries (LiPo) |  |  |
| Autonomously take readings at specified times | MSP-430 | Intel Edison | Raspberry Pi | Arduino Mega 2560 |

# Evaluation of Design Alternatives

## Details and Drawings of Each Alternative

### Design 1: Single Motor With SD Card

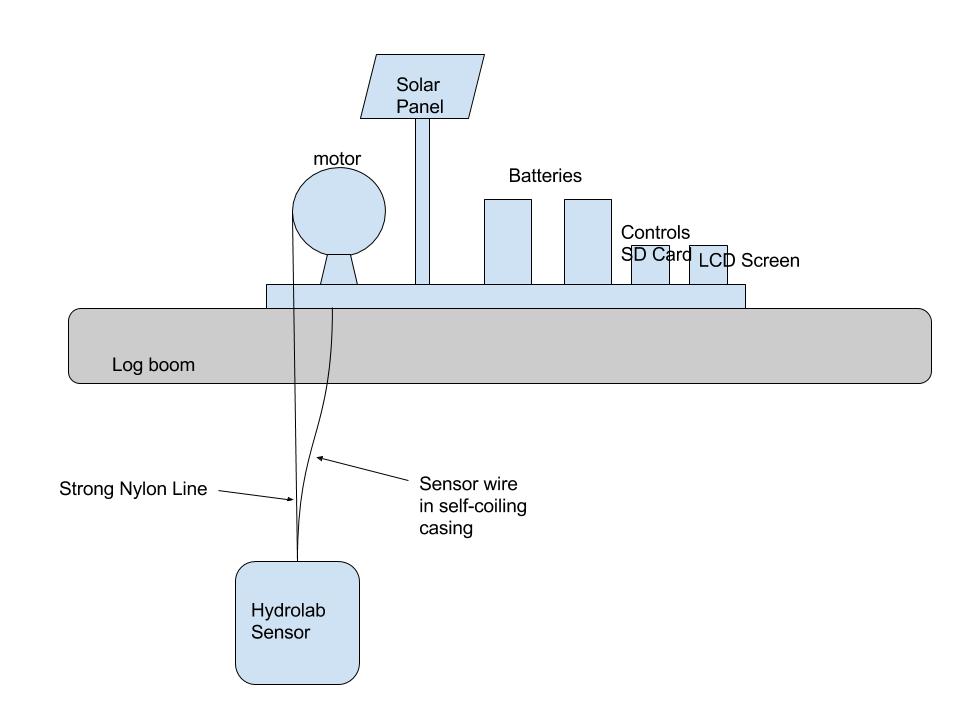


Figure 5: Design 1

Design one has a single DC motor that lowers and lifts the sensor using the feedback from the sensor to find the correct depth to take each reading. At this time the sensor will be in a stationary position (this applies to all designs) and the total distance to the bottom of the lake will be known; we will not need a sonar type sensor to find the bottom.

We have chosen to use an external clock with a shield to ensure that time is kept correctly over the max 35 day period. The clock module can run on its own battery power for several years and will keep the device accurate even if the microcontroller loses power and needs to be restarted.

We will be using the Quanta sensor instead of the smarTROLL sensor because the smarTROLL is not suitable for long-term deployment and cannot interface with a microcontroller; it can only send measurements to a company phone app where the user must physically be within a short range of the sensor to receive the measurements. The Quanta sensor uses SDI-12 communication protocol and can be programmed with a microcontroller to take readings without a person there. SDI-12 communication (two way serial communication with a 1200 baud rate and a 10 bit frame to send Ascii values) is designed for low-powered environmental sensors with microprocessors (like the Quanta). There are a plethora of resources in using the communication protocol, but we are currently having some trouble interfacing with the sensor and are working with Don McClane to solve the issue; we are also in contact with the Hydrolab company. We are currently looking at and pricing small one-reading sensors in case this issue persists.

We would like the user to be able to program the device using an LCD interface with buttons to choose times for taking readings. We feel this would be the most convenient and intuitive option. We have chosen to use an SD card for similar reasons. They are easy to program, keep data persistently in case of power failure, are very affordable, and can interface with a variety of microcontrollers.

We have chosen to use solar panel to prolong battery life as the lake does not get much wind, and also has very little current. The counterweight would decrease power usage but also makes the design less adaptable for future projects with Dr. Gawel.

Plastic casing to protect the device from water is a cheap, light option that is more durable than using wood. We plan to use the Fablab to help us construct a custom waterproof casing.

The self-coiling casing to protect the sensor cord from bending is a cheap and low tech alternative that requires less complicated design and less power than the reel alternatives.

Finally, we are choosing to have more than one LiPo (a more environmentally friendly type of battery than alternatives) in order to prevent power loss through heat. One battery will power the motor and sensor, and another the microcontroller. This way we can avoid using a regulator.

The MSP430 microcontroller is a low-powered device that can interface with SDI-12 and the various shields we have in mind (SD card, external clock). Our back up is the Arduino Mega 2560 because of our familiarity, its power specifications, and the large amount of I/O is can support.

### Design 2: Multiple Stationary Sensors

The main difference of design 2 is that the motor is replaced with multiple stationary sensors. This would reduce the power usage and make the overall device less mechanically complicated, but it is also not a design meant for the eventual mobile array that Dr. Gawel has in mind in future years.

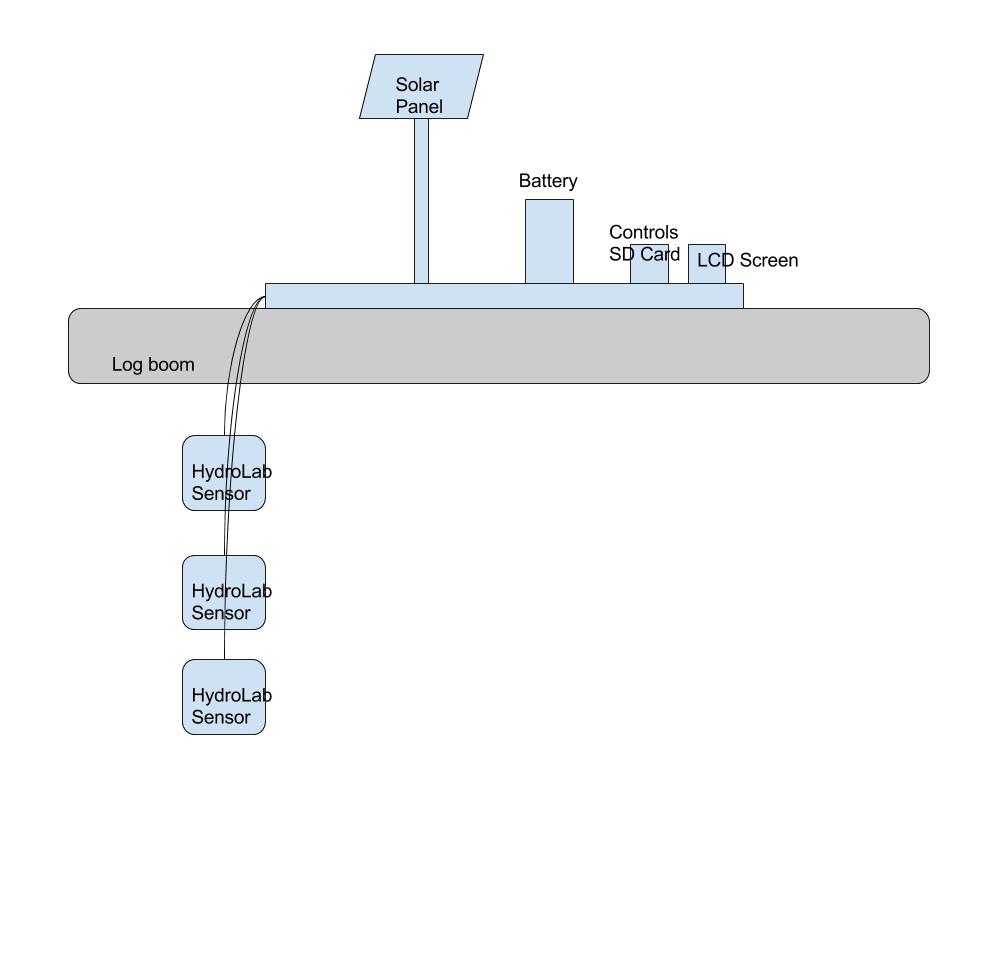


Figure 6: Design 2

### Design 3: Wireless Communication

The main difference for design 3 is that we are trying to create a wireless line of sight system that transmits data from the device, over land through a series of XBees, to a base station and then to a server where Dr. Gawel can access the data almost as soon as it is recorded. We also have a wireless design that involves ham radio, but this would require getting a license where using XBees would not.

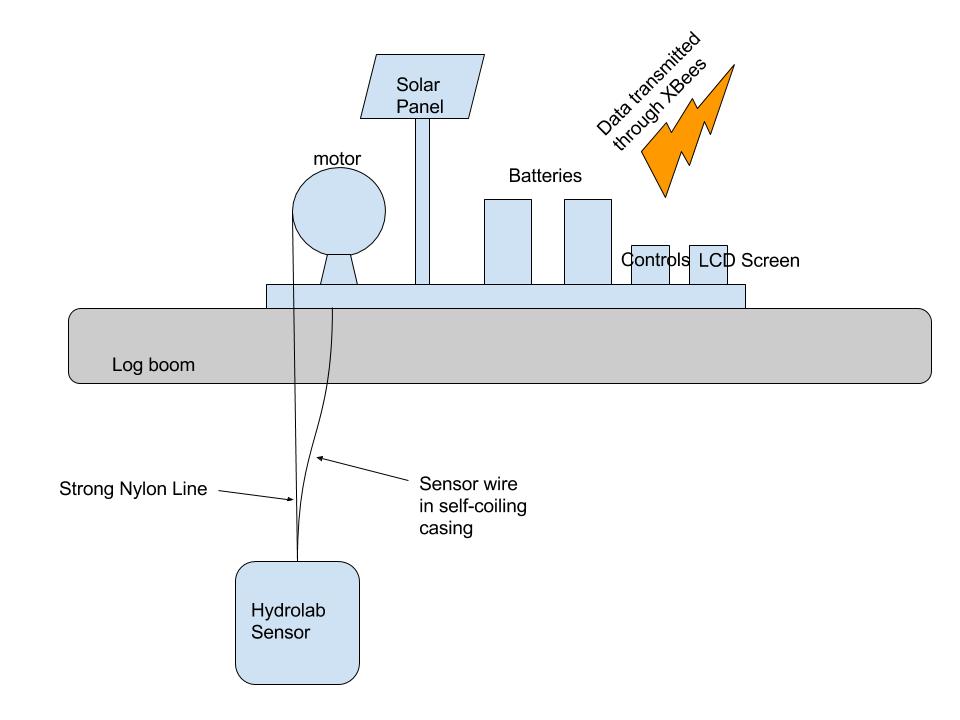


Figure 7: Design 3

### Design 4: Single Motor With Counterweight

This design is similar to design 1, but replaces the solar array with a counterweight to save power. See figure below.

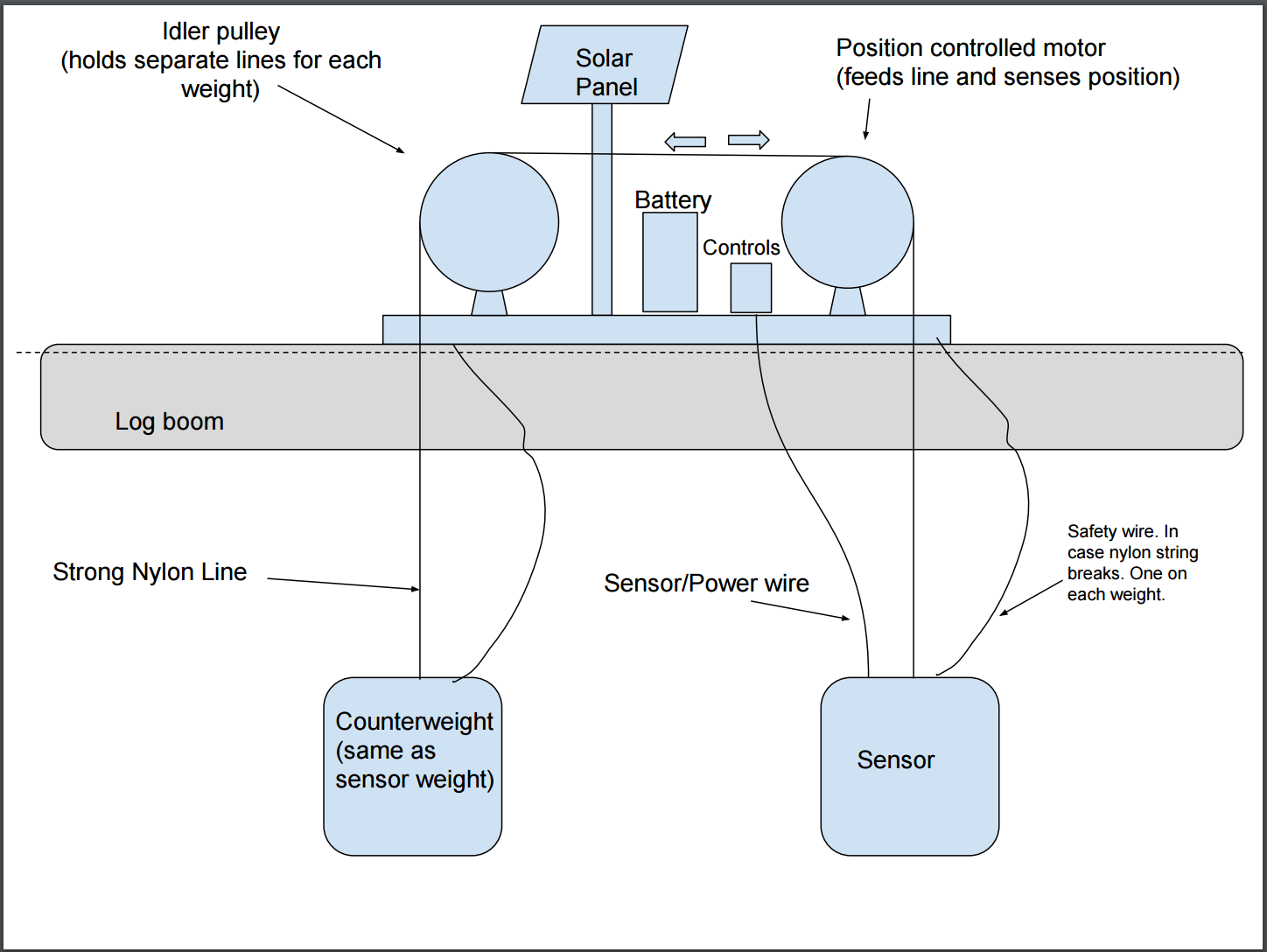


Figure 8: Design 4

## Verify Constraints are Met

**Meets Constraints**

**Does Not Meet Constraints**

Table 15: Do Designs Meet Constraints?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Constraint | Design 1: DC Motor and SD Card | Design 2: Multiple Stationary Sensor | Design 3: Wireless | Design 4: Counterweight |
| Total Cost Cannot Exceed $5000 |  |  |  |  |
| Must Fit Onto Existing Platform |  |  |  |  |
| Environmental Compliance |  |  |  |  |

The water quality sensors meant for long-term deployment similar to the Quanta and smarTROLL sensors are several thousand dollars each it would be not feasible to buy more to use them at different levels (design 2). The extra equipment required for the line of sight wireless setup would also potentially cost more than the funds we currently have available.

All designs could meet the other two constraints. We must make sure the batteries are protected from leaking into the environment, and we must confine the sensor to the platform. None of the devices would make meeting these constraints impossible as those components are flexible in design.

## How Well Do Designs Meet Objectives?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Objectives** | **Design 1 (Motor, One Sensor, SD)** | **Design 2 (Multiple Sensors)** | **Design 3 (Wireless)** | **Design 4**  **(Counter-weight)** | **Objectives** |
| **Battery Life/10** | 7 | 10 | 3 | 10 | **Battery Life/10** |
| **Light Blocked/10** | 7 | 7 | 7 | 7 | **Light Blocked/10** |
| **Data Collection/10** | 7 | 7 | 10 | 7 | **Data Collection/10** |
| **Weight/10** | 5 | 5 | 3 | 5 | **Weight/10** |
| **Maintenance/10** | 10 | 7 | 1 | 7 | **Maintenance/10** |
| **Setup/10** | 10 | 7 | 1 | 7 | **Setup/10** |
| **Cost/10** | 7 | 1 | 5 | 10 | **Cost/10** |
|  | 53 | 42 | 30 | 52 |  |

Design 3 fairs the worst with the objectives; it is both expensive and time consuming in all its aspects. It does have the undeniable benefit of data collection but the payoffs at this time are not worth the trouble. Design two would be a better choice if it did not violate the cost constraint. It is possible to use just the sensor we have and not take reading at different depths, but this would not be as useful for the Spirit Lake research (although we are keeping this in mind if we run into difficulties maintaining battery level for the full month). Designs 1 and 4 achieve a similar score but with different strengths. Design 1 would be the easiest to deal with maintenance and setup-wise but costs slightly more.

## How Well Do Designs Meet Functional Requirements?

All of our devices could meet the functional requirements at a bare minimum. Design 3 would require the most work and upkeep to make sure the system functioned properly, and may need the most maintenance and backups. It would allow more immediate data access, but since a person would still need to visit the site monthly it does not seem to offer any significant advantages.

## Method for Choosing Among Designs

Due to violating of the cost constraint we can discard designs 2 and 3 immediately, even though they each have significant advantages in certain areas as previously discussed. Our real choice is between the motor with solar power (design 1) and the motor with a counterweight (design 4). Their objective scores are very comparable, and they both meet the functional requirements as far as we can predict. In this case we must also consider the usefulness of our design in the future. Eventually Dr. Gawel would like to platform to move and drift with the greater log boom in the lake. Having the counterweight in the design would mean there would always be part of the design dangling, increasing the chances of damaging the device if it were to get tangled. Considering this, our preliminary design choice is design 1.

# Preliminary Design Choice

We have chosen our candidate design 1, the single motor with SD card for saving data. We have discussed these reasons in section 7.5. Designs 2 and 3 violate our cost constraint and the wireless aspect of design 3 would create more maintenance for those using it. The counterweight doesn't allow for the platform to move later on, which is what our client wanted. Design 1 is still fairly energy-efficient without having more dangling parts than necessary.

# Testing

## How Will We Test the Device?

Our main testing of the sensor device will take place in early May 2016 on one of the local lakes where Dr. Gawel and his team have platforms similar to the Spirit Lake log platforms. This will allow us to see how our device works in real world conditions. We will need to make sure the mechanical functions and software perform as expected (taking the correct number of readings, going to the correct depths, battery lasting the full length of time). We will also do testing in the lab as we create the device to make sure these things are happening as well.

## How Do We Define Success?

We define success as having something that Dr. Gawel can use this upcoming summer to collect data on the conditions of Spirit Lake without having to be there all the time.

# Social and Ethical Considerations

We have a responsibility to our client, Dr. Gawel, and our sponsor (SIAS) to deliver a prototype at the end of May 2016 that they will be able to use. By accepting this project and sponsorship we have implicitly agreed to put forth our best effort, keep open an honest line of communication, and provide updates if we run into any issues that may prevent us from continuing as envisioned.

We hope to continue to engage with Dr. Gawel so that he is satisfied with the end product, and to keep clear records and instructions to allow him to operate and maintain the device for the months and years to come. We hope the quality of our work will keep the sensor array functioning long after we have received a grade, and that our project is high enough quality that students coming after us can build on what we have done.

In addition, we have an obligation to keep in mind the potential damage our device could do to the ecosystem of Spirit Lake. We must carefully consider all components in our device and how they work together to ensure they have a neutral impact. This means we will need to thoroughly test our device before actually deploying it to a lake.

# Project Management

## Work Breakdown Schedule

\*Strikethroughs indicate complete

1. Design
   1. ~~Preliminary Work~~
      1. ~~Interview Dr. Gawel~~
      2. ~~Draft problem statement~~
      3. ~~Research~~
         1. ~~Research available sensor devices~~
            1. ~~Quanta~~

~~SDI-12 Communication Protocol~~

* + - * 1. ~~smarTROLL~~
      1. ~~Batteries~~
      2. ~~Water Data Loggers~~
      3. ~~Motors~~
      4. ~~Solar Panels~~
      5. ~~Microcontroller~~
  1. ~~Zoning In~~
     1. ~~Revise problem statement~~
     2. ~~OCF~~
     3. ~~Metrics~~
  2. ~~Brainstorming Alternative Designs~~
     1. ~~Morph Chart~~
     2. ~~Sketches~~
  3. ~~Choosing a Design~~
     1. ~~Applying metrics~~
     2. Present preliminary design to Dr. Gawel

1. Building Prototype/Deliverable
   1. Determine tools needed to build
   2. Determine place to build and store
   3. Order/Gather Parts
      1. Online and tracking
      2. From UW
      3. Local Stores
      4. Fablab
   4. Write Code for MSP430
      1. Sensor Module
      2. Clock Module
      3. SD Module
      4. Motor Module
   5. Interface with each module separately
   6. Assemble
      1. Casing
      2. Power
   7. Debug
2. Testing
   1. Develop Protocol
      1. Lab testing
      2. Monitored local lake testing
      3. Unmonitored local lake testing
   2. Conduct Tests
      1. Lab testing
      2. Monitored local lake testing
      3. Unmonitored local lake testing
   3. Evaluate Results
      1. Lab testing
      2. Monitored local lake testing
      3. Unmonitored local lake testing
3. Documents
   1. ~~Presentation 1~~
   2. ~~Presentation 2~~
   3. ~~Library Search Results~~
   4. ~~Preliminary Design Document~~
   5. User Manuel Draft
   6. User Manuel Final
   7. Final Document
   8. Final Spring 2016 Presentation
4. Management and Control
   1. Weekly Meetings
   2. Weekly Classes
   3. Schedule
   4. Weekly Updates to Dr. Gawel
5. Deployment
   1. Setup at Spirit Lake
   2. Monitoring and Maintenance

## Schedule

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MARCH | | | | | | | | | | | | |
| Sun |  | Mon |  | Tue |  | Wed |  | Thu |  | Fri |  | Sat | |
|  |  |  |  | 1 |  | 2 |  | 3 |  | 4 |  | 5 | |
|  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 6 |  | 7 |  | 8 |  | 9 |  | 10 |  | 11 |  | 12 | |
|  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 13 |  | 14 |  | 15 |  | 16 |  | 17  PPD Due Midnight |  | 18 |  | 19 | |
|  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 20 |  | 21  6:30AM – 10AM: Setup Quanta, call OTT  10AM-3PM:  Work logic analyzer |  | 22  Send email update to Dr. Gawel – sensor progress? Prelim design update |  | 23  Team meeting 10AM-4PM: create proposal for all components still to be purchased |  | 24 |  | 25  Send purchase requests for batteries, solar collectors, shields, and motor to Dr, Gawel |  | 26 | |
|  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 27 |  | 28 |  | 29  482: 3:05PM-5:10PM  Divide programming of modules |  | 30 |  | 31  482: 3:05PM-5:10PM: send updates to Dr. Gawel |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| APRIL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Sun |  | Mon | |  | | | Tue | | |  | | | Wed | |  | | Thu | | | |  | | Fri | | | |  | | Sat | | | | |
|  |  |  | |  | | |  | | |  | | |  | |  | |  | | | |  | | 1  10AM-4PM: Team meeting | | | |  | | 2 | | | | |
|  |  |  | |  | | |  | | |  | | |  | |  | |  | | | |  | |  | | | |  | |  | | | | |
| 3 |  | 4  purchase order for smaller sensors within budget? | |  | | | 5  482: 3:05PM-5:10PM | | |  | | | 6 | |  | | 7  482: 3:05PM-5:10PM:  send updates to Dr. Gawel | | | |  | | 8  10AM-4PM: Team meeting; if Begin draft of OS/SDI-12 | | | |  | | 9 | | | | |
|  |  |  | |  | | |  | | |  | | |  | |  | |  | | | |  | |  | | | |  | |  | | | | |
| 10 |  | 11 | |  | | | 12  482: 3:05PM-5:10PM | | |  | | | 13  SDI-12 OS first draft | |  | | 14  482: 3:05PM-5:10PM:  send updates to Dr. Gawel | | | |  | | 15  10AM-4PM: Team meeting  Begin integration and lab testing | | | |  | | 16 | | | | |
|  |  |  | |  | | |  | | |  | | |  | |  | |  | | | |  | |  | | | |  | |  | | | | |
| 17 |  | 18 | |  | | | 19  482: 3:05PM-5:10PM | | |  | | | 20  Begin to assemble and integrate protoype | |  | | 21  482: 3:05PM-5:10PM:  send updates to Dr. Gawel | | | |  | | 22  10AM-4PM: Team meeting | | | |  | | 23 | | | | |
|  |  |  | |  | | |  | | |  | | |  | |  | |  | | | |  | |  | | | |  | |  | | | | |
| 24 |  | 25 | |  | | | 26  482: 3:05PM-5:10PM  Debug | | |  | | | 27 | |  | | 28  482: 3:05PM-5:10PM:  updates to Dr. Gawel | | | |  | | 29  10AM-4PM: Team meeting  Debug | | | |  | | 30 | | | | |
|  |  |  | |  | | |  | | |  | | |  | |  | |  | | | |  | |  | | | |  | |  | | | | |
| MAY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Sun |  | Mon | | | |  | | Tue | | | |  | | Wed | |  | | Thu | | | |  | | Fri | | | |  | | Sat | | | | | |
| 1 |  | 2 | | | |  | | 3  482: 3:05PM-5:10PM  Debug | | | |  | | 4 | |  | | 5  482: 3:05PM-5:10PM:  send updates to Dr. Gawel | | | |  | | 6  10AM-4PM: Team meeting  Cont debug | | | |  | | 7 | | | | | |
|  |  |  | | | |  | |  | | | |  | |  | |  | |  | | | |  | |  | | | |  | |  | | | | | |
| 8 |  | 9 | | | |  | | 10  482: 3:05PM-5:10PM:  draft testing protocol | | | |  | | 11  COMPLETE PROTOTYPE and user manual first draft | |  | | 12  482: 3:05PM-5:10PM:  send updates to Dr. Gawel | | | |  | | 13  10AM-4PM: Team meeting;  Begin testing in local lakes | | | |  | | 14 | | | | | |
|  |  |  | | | |  | |  | | | |  | |  | |  | |  | | | |  | |  | | | |  | |  | | | | | |
| 15 |  | 16 | | | |  | | 17  482: 3:05PM-5:10PM;  Monitor testing | | | |  | | 18 | |  | | 19  482: 3:05PM-5:10PM:  send updates to Dr. Gawel | | | |  | | 20  10AM-4PM: Team meeting | | | |  | | 21 | | | | | |
|  |  |  | | | |  | |  | | | |  | |  | |  | |  | | | |  | |  | | | |  | |  | | | | | |
| 22 |  | 23 | | | |  | | 24  482: 3:05PM-5:10PM  Monitor testing | | | |  | | 25 | |  | | 26  482: 3:05PM-5:10PM:  send updates to Dr. Gawel | | | |  | | 27  10AM-4PM: Team meeting;  Discuss testing results | | | |  | | 28 | | | | | |
|  |  |  | | | |  | |  | | | |  | |  | |  | |  | | | |  | |  | | | |  | |  | | | | | |
| 29 |  | 30  MEMORIAL DAY (HOLIDAY) | | | |  | | 31  482: 3:05PM-5:10PM | | | |  | |  | |  | |  | | | |  | |  | | | |  | |  | | | | | |
|  |  |  | | | |  | |  | | | |  | |  | |  | |  | | | |  | |  | | | |  | |  | | | | | |
| JUNE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Sun |  | Mon |  | | Tue | | | |  | | Wed | | | | | | | |  | Thu | | | | |  | Fri | | | | |  | Sat | | | | | |
|  |  |  |  | |  | | | |  | | 1  Deadline for submitting reimbursement to SIAS | | | | | | | |  | 2  482: 3:05PM-5:10PM:  send updates to Dr. Gawel | | | | |  | 3  10AM-4PM: Team meeting  Complete user manual | | | | |  | 4 | | | | | |
|  |  |  |  | |  | | | |  | |  | | | | | | | |  |  | | | | |  |  | | | | |  |  | | | | | |
| 5 |  | 6 |  | | 7 | | | |  | | 8 | | | | | | | |  | 9 | | | | |  | 10 | | | | |  | 11 | | | | | |
|  |  |  |  | |  | | | |  | |  | | | | | | | |  |  | | | | |  |  | | | | |  |  | | | | | |
| 12 |  | 13 |  | | 14 | | | |  | | 15 | | | | | | | |  | 16 | | | | |  | 17 | | | | |  | 18 | | | | | |
|  |  |  |  | |  | | | |  | |  | | | | | | | |  |  | | | | |  |  | | | | |  |  | | | | | |
| 19 |  | 20 |  | | 21 | | | |  | | 22 | | | | | | | |  | 23 | | | | |  | 24 | | | | |  | 25 | | | | | |
|  |  |  |  | |  | | | |  | |  | | | | | | | |  |  | | | | |  |  | | | | |  |  | | | | | |
| 26 |  | 27 |  | | 28 | | | |  | | 29 | | | | | | | |  | 30 | | | | |  |  | | | | |  |  | | | | | |
|  |  |  |  | |  | | | |  | |  | | | | | | | |  |  | | | | |  |  | | | | |  |  | | | | | |

\*Note: not all tasks placed on schedule due to unknown due dates; also need to discuss dates for deployment to Spirit Lake with Dr. Gawel.

## Budget

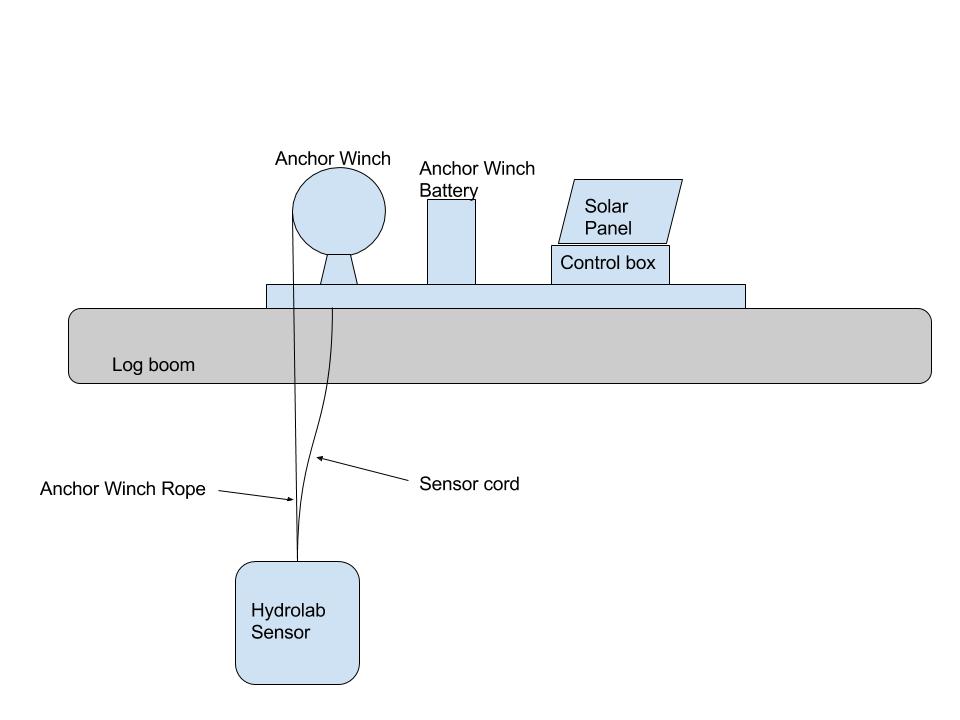
Table 16: Proposed Materials Budget

|  |  |  |
| --- | --- | --- |
| **Item** | **Qty.** | **Price (Total)** |
| **Hardware:** |  |  |
| 12-volt NiMH battery | 2 | $200 |
| 4.8-volt NiMH battery | 2 | $50 |
| Brushless DC Motor | 1 | $20 |
| Aluminum grate | 1 | $50 |
| Plastic casing | 1 | Free from FabLab |
| Solar Panel | 1 | $25 |
| SDI-12 Cable | 1 | $95 |
| Recoiling Hose | 1 | $15 |
| **Possible Expenses\*:** |  |  |
| PH Sensor | 1 | $86 |
| Oxygen Sensor | 1 | $100 |
| Pressure Sensor | 1 | $100 |
| ORP Sensor | 1 | $40 |
| Salinity Sensor | 1 | $75 |
| Temperature Sensor | 1 | $20 |
| BNC Connector | 6 | $12 |
| **Total (Budget is $2,000)** |  | $888 |

\* We are having trouble getting the provided sensor to work, so if we can’t get it working, we need to buy our own sensors from <http://www.reefangel.com/> or <http://atlas-scientific.com/>.

# Final Results

In the end, we were able to get the sensor working. This allowed us to stay in our budget and use the extra money to purchase an anchor winch to raise and lower the sensor instead of building our own mechanism.

Figure 9: The final design

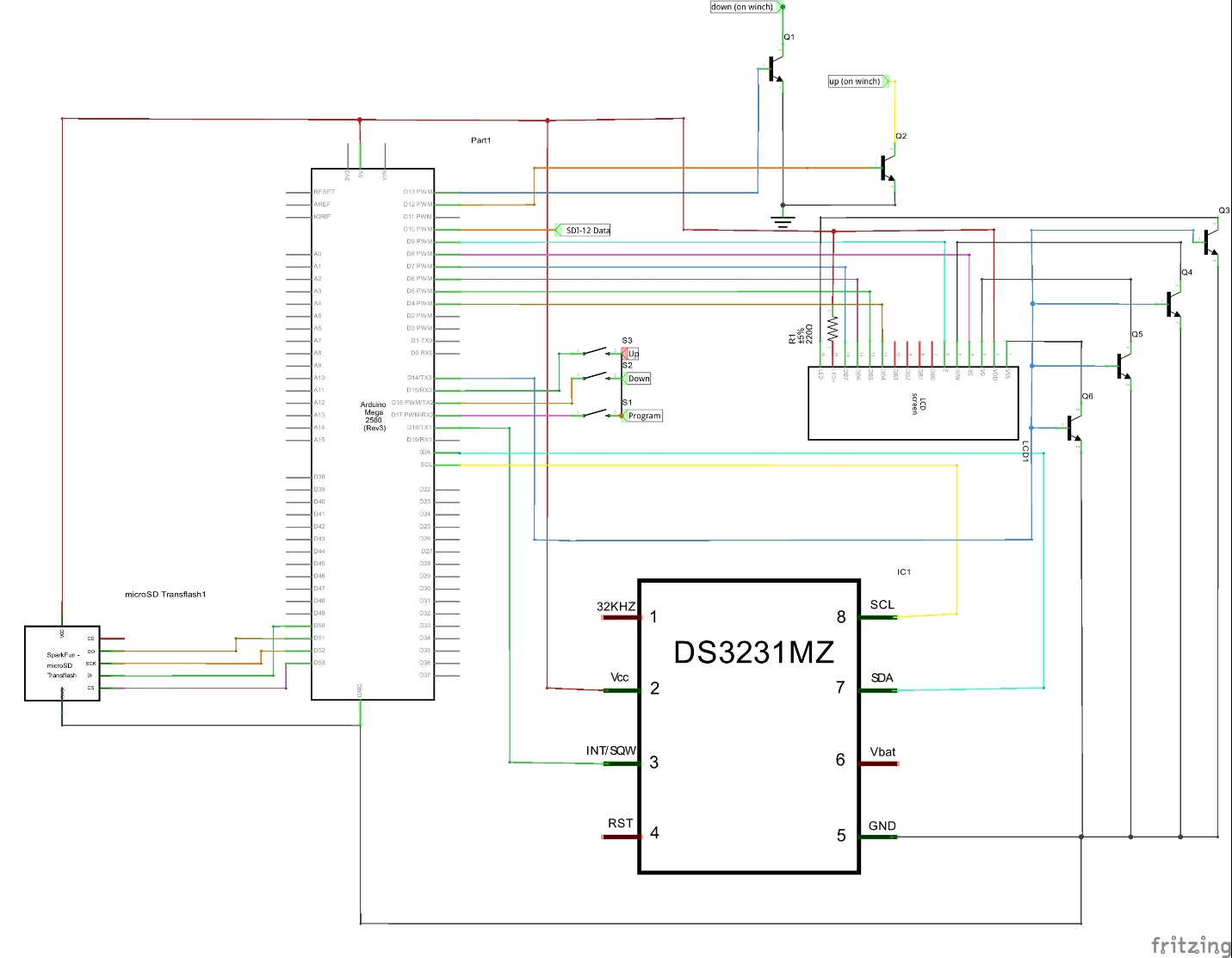
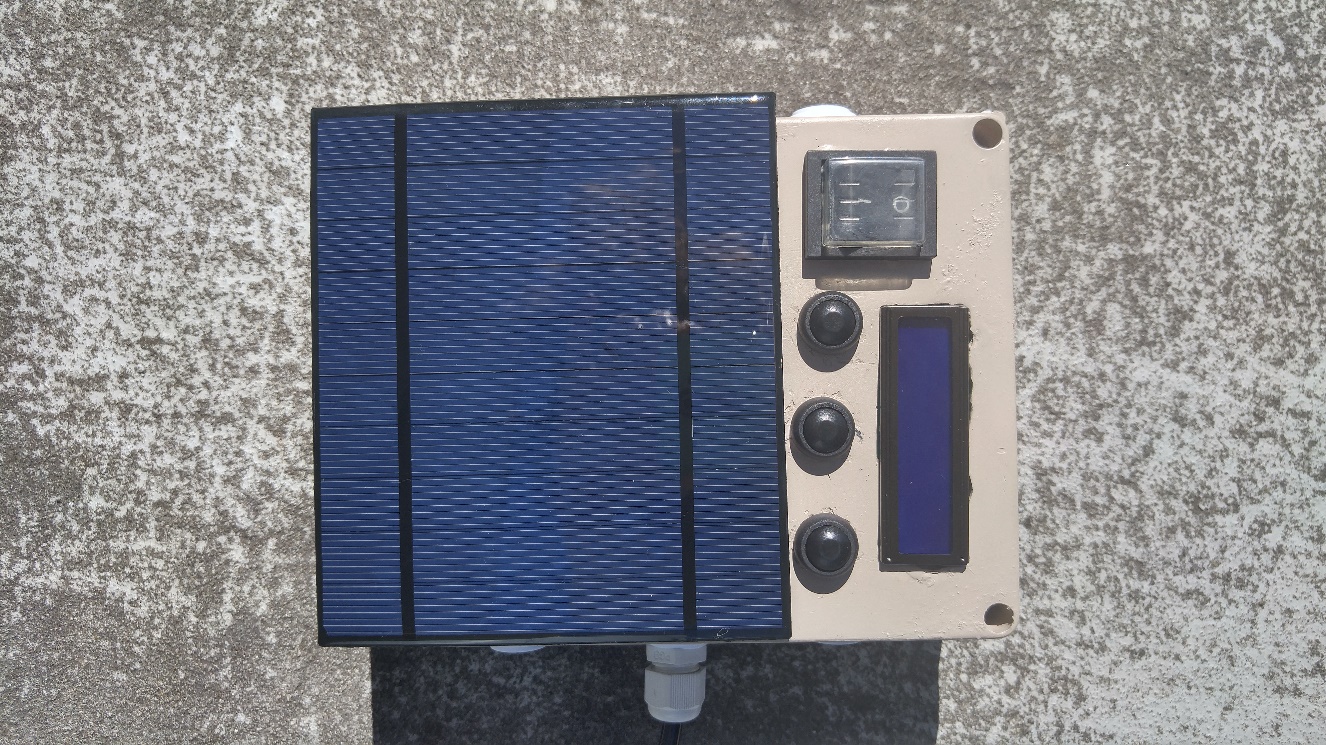
Figure 10: The final schematic for our control box.

Figure 11: Final set up.

Figure 12: Control box.

## Team Dynamics and Schedule

Since there were only two of us working on this project, we both worked on every aspect of the project. Gabrielle did all of the soldering for our final design. Demy designed the case layout and added the interface hardware (LCD, buttons, switch) to the lid of the case. Demy and Gabrielle both worked on final construction and waterproofing. Demy and Gabrielle both worked on the final report.

We ended up behind schedule because we were focusing too much on making something to raise and lower the sensor. In the end, Gabrielle found a reasonably priced anchor winch that we could hack into using transistors to move the sensor. Luckily we had a few weeks set aside for testing so we were able to shorten our testing period to allow more room for programming and soldering. We were able to complete our project for demonstration day, but we are going to do more testing on a lake before we hike it up with Dr.Gawel.

## Project Goals

Looking at our OCF chart (Table 1, Section 4), we have achieved the majority of our goals. The two goals that we have not completely met are having the battery last at least 30 days and making the system lightweight.

The battery life issue is due to 2 things: the microprocessor and the anchor winch. We are using an Arduino Mega 2560 for this project. This is not ideal because it takes a lot of power. Ideally, we would have used an MSP430, but we were having issues with the sensor and the Arduino that we didn’t have time to port over to the MSP430. The MSP430 has a logic level of 3.3V, but 5V is needed for SDI-12 communication (the protocol the sensor uses), so we would have had to construct and test a circuit to translate between the two. We made sure that we are putting the Arduino to the deepest sleep it can go in between readings, and we attempted to extend the battery life by using a solar charging circuit to power the Arduino. The other battery hog is the anchor winch. The anchor winch runs off of a 12V deep-cycle battery. We cannot optimize the power consumption of the motor, so we made sure that if the battery runs out on the motor that the sensor will still continue to take readings at scheduled times, just not at different depths.

The weight issue is brought about by the anchor winch as well. The battery used to power the anchor winch is about 25 pounds, which isn’t the best for taking on a 3-mile hike. After thinking about it, we realized it would probably be impossible to make it lighter because of the electrical power (and thus, battery size) and mechanical power (and thus, anchor winch size) needed to lift the weight of the sensor and the cord.

## Updated Budget

Table 17: Final Budget

|  |  |  |
| --- | --- | --- |
| **Item** | **Qty.** | **Price (total)** |
| 12V 2000mAh Battery | 1 | $24 |
| Battery Charger | 1 | $17 |
| Anchor Winch Battery | 1 | $81 |
| Anchor Winch | 1 | $100 |
| 3.7V Battery | 2 | $17 |
| SD Card Reader | 1 | $7 |
| Solar Panel + Components | 1 | $15 |
| IRLB3034PBF (Heavy Duty Transistors) | 2 | $5 |
| Solderable Breadboard | 3 | $15 |
| SDI-12 Cable | 1 | $100 |
| Casing + Buttons + LCD Screen | 1 | $20 |
| Miscellaneous (wires, connectors etc.) | 1 | $30 |
| Anchor Winch Battery Charger | 1 | $100 |
| Plywood | 1 | $13 |
| **Total:** |  | **$544** |

## Testing

Due to us falling behind schedule, we had to put off some of the testing. Originally we wanted to test on a local lake, but we were only able to test at school in a tank. Our tank was located on the roof of the Science building.

Our original test plan was to leave the sensor on all day and night and measure the battery usage. However, we were not able to leave it out overnight so we were only able to test for 3 days on the roof during the day. We measured the battery voltage at the beginning of the day, and then again at the end of the day. On the first and last days, the battery went down by about .1V, on the second day, the battery managed to get charged by the solar panel during the day. More testing overnight will be needed before deploying it in the field. We are also going to test for a day on a lake to check the performance there as well.

We didn’t do a lot of testing using the anchor winch battery (we mostly used a power supply), but according to our calculations the anchor winch should be able to be used for 2 hours total. This means that since the anchor winch will only be used for about 5 minutes maximum per day, the battery should last almost the entire month.

Figure 13: Testing location.

Figure 14: Testing set up.

Figure 15: Testing in progress

## What We Learned

From this project, we learned how to combine many components to create a working, usable product. We learned about battery power calculations, power efficiency, the practical use of transistors, and building our own projects. Before this project, neither of us had done much electronics tinkering or built anything of this magnitude. Now, we both feel comfortable building a project and finding the correct parts to do it. Classes that helped us accomplish this task are:

* **TCES 312 Electronic and Analog Systems** 
  + Understanding transistors and how they work.
* **TCES 430 Microprocessor System Design**
  + Understanding timers, interrupts, and microprocessor basics.
* **TCES 465 Embedded Real-Time Systems**
  + Understanding how to design and build an embedded system.
* **TCES 372 Machine Organization and Architecture**
  + Understanding how SDI-12 communication works and what a baud rate is.
* **TCSS 360 Software Development and Quality Assurance**
  + How to use GitHub.
* **TCES 279 Modern Fabrication**
  + Soldering and FabLab skills.

We think that a class in power would be helpful, as in how to calculate power usage and estimate battery needs. A class in project building would also be helpful, specifically how to look for parts, design casing, and basically make a final product. We think that this could possibly be added to the Modern Fabrication class.

# Appendix A: Research

## Library Search Results

[1] F. Quiles, M. Ortiz, A. Gersnoviez, M. Brox, A. Olivares and P. Glosekotter, "Development of a Wireless Low Power Datalogger with High Performance Converter", Elektronika ir Elektrotechnika, vol. 21, no. 3, 2015.

[2] S. S. Panahi et al., "A Low-Power Datalogger Based on CompactFlash Memory for Ocean Bottom Seismometers," in IEEE Transactions on Instrumentation and Measurement, vol. 57, no. 10, pp. 2297- 2303, Oct. 2008. doi: 10.1109/TIM.2008.919005Other Search Results

## Other Search Results

[1] Hydrolab Quanta Manual

<http://www.ott.com/download/hydrolab-quanta-manual/>

[2] SDI-12 Specifications

<http://www.sdi-12.org/current%20specification/SDI-12_version1_3%20January%2028,%202016.pdf>

[3] How to Handle LiPo Batteries

<http://thedronegirl.com/2015/02/07/15-things-every-lipo-battery-user-should-know/>

[4] Hobby King: Pricing for motors, batteries, solar

<http://www.hobbyking.com/hobbyking/store/index.asp>

Appendix B: Arduino Code

## Arduino Code

Here is the Arduino sketch we used to create our Datalogger.

It can also be found at: <https://github.com/dloulias/SpiritLake>

/\*

\* Spirit Lake Software

\* Version 1.5

\* Demetra Loulias and Gabrielle Glynn

\*

\* Takes readings at set alarm times from preferences CSV.

\*

\* Still to be done:

\* - Get depth when program is started

\* - Make sure that the sensor is actually going down

\* - Match depth reading to varible during readings

\*/

// Struct to hold hour and minute of alarms

typedef struct {

int hour;

int minute;

} AlarmTime;

#include <LowPower.h>

#include <SPI.h>

#include <SD.h>

#include <ctype.h>

#include <DS3232RTC.h> //http://github.com/JChristensen/DS3232RTC

#include <Time.h> //http://playground.arduino.cc/Code/Time

#include <Wire.h> //http://arduino.cc/en/Reference/Wire

#include <SDI12.h>

#include "LiquidCrystal.h"

#define D4 4 // LCD D4

#define D5 5 // LCD D5

#define D6 6 // LCD D6

#define D7 7 // LCD D7

#define RS 8 // LCD RS

#define E 9 // LCD E

#define DATAPIN 10 // SDI-12 pin

#define ALARM\_PIN 18 // Interrupt pin from RTC

#define UP 12 // Up pin (for transistor)

#define DOWN 13 // Down pin (for transistor)

#define LCDON 14 // Turns on transistors for LCD

#define TIMEUP 15 // Button for time up

#define TIMEDOWN 16 // Button for time down

#define PROGRAM 17 // Button for program

#define CHIPSELECT 53 // Chip select for SD card

#define MAX\_READINGS 10

#define MAX\_DEPTH 460 // The maximum depth the sensor will go, in meters

AlarmTime alarms[MAX\_READINGS]; // Holds all reading times

int numAlarms; // Total number of alarms that are set. (Max of 5)

int alarmCount; // The current alarm the program is on.

float lowest; // The lowest point in the water the sensor goes.

float highest; // The highest point in the water the sensor goes.

boolean motorOn; // Boolean value for whether or not the motor should be used

boolean goUp; // Boolean for whether the motor should take readings going up or down

SDI12 mySDI12(DATAPIN); // Attach SDI-12 communication to data pin.

LiquidCrystal lcd(RS,E,D4,D5,D6,D7); // LCD screen

void setup() {

mySDI12.begin(); // Begin SDI-12 communication

// Setup interface buttons

pinMode(UP, OUTPUT);

pinMode(DOWN, OUTPUT);

pinMode(TIMEUP, INPUT\_PULLUP);

digitalWrite(TIMEUP, HIGH);

pinMode(TIMEDOWN, INPUT\_PULLUP);

digitalWrite(TIMEDOWN, HIGH);

pinMode(PROGRAM, INPUT\_PULLUP);

digitalWrite(PROGRAM, HIGH);

pinMode(LCDON, OUTPUT);

// turn on LCD

digitalWrite(LCDON, HIGH);

lcd.begin(16,2);

// Wait for SD card to be inserted.

while(!SD.begin(CHIPSELECT)){

lcdPrintMessage(" CARD NOT", " INSERTED ");

}

lcdPrintMessage(" Card Inserted", "");

delay(1000);

// display current time

time\_t currtime = RTC.get();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Current Time:");

lcd.setCursor(0,1);

lcd.print(month(currtime));

lcd.print("/");

lcd.print(day(currtime));

lcd.print("/");

lcd.print(year(currtime));

lcd.print(" ");

lcd.print(hour(currtime));

lcd.print(":");

lcd.print(minute(currtime));

delay(3000);

// wait for user input for 10 seconds (if they want to program)

int counter = 10;

while(true){

if(digitalRead(PROGRAM) == LOW){

programClock();

programDepths();

break;

}

if(counter == -1) {

break;

}

lcdPrintMessage("Starting in...", String(counter));

counter--;

delay(1000);

}

lcdPrintMessage(" Loading", " preferences...");

delay(1000);

// Read in CSV file containing alarms

if(SD.exists("config.csv")){

if(!getTimes()) {

setDefaultAlarms(); // Set to defaults if incorrect format.

lcdPrintMessage("Preferences file", " not found.");

delay(2000);

lcdPrintMessage(" Default times", " loaded.");

}

else

lcdPrintMessage("Preferences file", " found.");

}

else{

setDefaultAlarms(); // Set to defaults if file does not exist

lcdPrintMessage("Preferences file", " not found.");

delay(2000);

lcdPrintMessage(" Default times", " loaded.");

}

delay(2000);

lcdPrintMessage("Loading", " depths...");

delay(2000);

// Check if depth range file is acceptable and move to start

if(getHeights()){

lcdPrintMessage("Moving to start", " position...");

bool startReached = false;

int timeDelay = 1000;

float diff;

float prevDiff;

float depth;

float prevDepth = 0;

while(true) {

depth = takeMeasurement(true);

prevDiff = abs(depth - prevDepth);

diff = lowest - depth;

if(prevDepth != 0) {

if(prevDiff < 0.05 || prevDiff == 0.05) {

motorOn = false;

lcdPrintMessage("Motor error", "");

delay(2000);

lcdPrintMessage("Operating","without motor.");

delay(2000);

break;

}

}

else if(diff < 0.1) {

goUp = true;

motorOn = true;

break;

}

else if(diff < 0.2) {

timeDelay = 500;

}

else if(diff < 0 || diff == 0) {

motorOn = false;

}

prevDepth = depth;

digitalWrite(DOWN, HIGH);

delay(timeDelay);

digitalWrite(DOWN, LOW);

}

}

// if no depth file, don't use motor for readings

else {

motorOn = false;

}

// Disable default square wave of pin

RTC.squareWave(SQWAVE\_NONE);

// Set pinmode to input

pinMode(ALARM\_PIN, INPUT);

lcdPrintMessage("Ready to", "start!");

digitalWrite(LCDON, LOW);

// set first alarm

setAlarmTime();

delay(5000);

}

/\*\*

\* Function that allows a user to set the range that the motor will go.

\* Saves the values to an SD card.

\*/

void programDepths() {

float depthHighest;

float depthLowest;

File depthFile;

// prompt user for highest position

lcdPrintMessage("Move sensor to", "highest position.");

delay(2000);

while(true) {

if(digitalRead(PROGRAM) == LOW){

lcdPrintMessage("Saving...","");

float depth1 = takeMeasurement(true);

float depth2 = takeMeasurement(true);

float depth3 = takeMeasurement(true);

// take average of 3 depth readings

depthHighest = (depth1 + depth2 + depth3)/3;

break;

}

}

// prompt user for lowest position

lcdPrintMessage("Move sensor to", "lowest position.");

delay(2000);

while(true) {

if(digitalRead(PROGRAM) == LOW){

lcdPrintMessage("Saving...","");

float depth4 = takeMeasurement(true);

float depth5 = takeMeasurement(true);

float depth6 = takeMeasurement(true);

// take average of 3 depth readings

depthLowest = (depth4 + depth5 + depth6)/3;

break;

}

}

if(depthHighest > depthLowest) {

float temp = depthHighest;

depthHighest = depthLowest;

depthLowest = temp;

}

// write over file if it exists

if(SD.exists("range.txt")) {

SD.remove("range.txt");

}

// open file and save values

File myFile = SD.open("range.txt", FILE\_WRITE);

if(myFile){

myFile.println(depthHighest);

myFile.println(depthLowest);

myFile.close();

lcdPrintMessage("Saved","");

}

else {

lcdPrintMessage("Error","saving.");

}

delay(2000);

}

/\*\*

\* Reads in the previously saved depth range for the sensor.

\*

\* @return whether or not the file is the correct format.

\*/

boolean getHeights() {

File depthFile = SD.open("range.txt");

int lineNum = 1;

String line = "";

float depth;

if(depthFile) {

while(depthFile.available() != 0) {

line = depthFile.readStringUntil('\n');

depth = line.toFloat(); // returns 0 if not actually a float

if(depth == 0) {

lcdPrintMessage("Incorrect", "format.");

delay(2000);

return false;

}

else if(lineNum == 1) {

highest = depth;

lineNum++;

}

else if(lineNum == 2) {

lowest = depth;

lineNum++;

}

else {

lcdPrintMessage("Incorrect", "format.");

delay(2000);

return false;

}

}

if(lineNum < 3){

lcdPrintMessage("Incorrect", "format.");

delay(2000);

return false;

}

else

return true;

}

else{

lcdPrintMessage("Depth file", "not found.");

delay(2000);

return false;

}

}

void lcdPrintMessage(String line1, String line2) {

lcd.clear();

lcd.setCursor(0,0);

lcd.print(line1);

lcd.setCursor(0,1);

lcd.print(line2);

}

/\*\*

\* Function to prompt user for clock programming.

\*/

void programClock() {

digitalWrite(LCDON, HIGH);

lcd.begin(16,2);

lcd.setCursor(0,0);

lcd.print("Programming");

lcd.setCursor(0,1);

lcd.print("Mode");

delay(1500);

tmElements\_t tm;

time\_t timeSet;

time\_t currtime = RTC.get();

int currMonth = month(currtime);

int currDay = day(currtime);

int currYear = year(currtime);

int currHour = hour(currtime);

int currMinute = minute(currtime);

delay(2000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Set month:");

lcd.setCursor(0,1);

currMonth = getEntry(12, 1, currMonth, "Set Month:");

currDay = getEntry(31, 1, currDay, "Set Day:");

currYear = getEntry(2030, 2016, currYear, "Set year:");

currHour = getEntry(23, 0, currHour, "Set Hour:");

currMinute = getEntry(59, 0, currMinute, "Set Minute");

tm.Month = currMonth;

tm.Day = currDay;

tm.Year = CalendarYrToTm(currYear);

tm.Hour = currHour;

tm.Minute = currMinute;

tm.Second = 0;

timeSet = makeTime(tm);

RTC.set(timeSet);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Time set!");

delay(1000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Current Time:");

lcd.setCursor(0,1);

lcd.print(month(timeSet));

lcd.print("/");

lcd.print(day(timeSet));

lcd.print("/");

lcd.print(year(timeSet));

lcd.print(" ");

lcd.print(hour(timeSet));

lcd.print(":");

lcd.print(minute(timeSet));

delay(3000);

}

/\*\*

\* Prompts user for a number between two values.

\*/

int getEntry(int high, int low, int initialVal, String prompt) {

int returnVal = initialVal;

lcd.clear();

lcd.setCursor(0,0);

lcd.print(prompt);

lcd.setCursor(0,1);

lcd.print(returnVal);

while(true) {

if(digitalRead(TIMEUP) == LOW) {

if(returnVal < high) {

returnVal++;

}

else {

returnVal = low;

}

lcd.clear();

lcd.setCursor(0,0);

lcd.print(prompt);

lcd.setCursor(0,1);

lcd.print(returnVal);

delay(500);

}

if(digitalRead(TIMEDOWN) == LOW) {

if(returnVal > low) {

returnVal--;

}

else {

returnVal = high;

}

lcd.clear();

lcd.setCursor(0,0);

lcd.print(prompt);

lcd.setCursor(0,1);

lcd.print(returnVal);

delay(500);

}

if(digitalRead(PROGRAM) == LOW) {

delay(300);

lcd.clear();

lcd.setCursor(0,0);

lcd.print(prompt);

lcd.setCursor(0,1);

lcd.print(returnVal);

break;

}

}

return returnVal;

}

/\*\*

\* Sets default alarm times if there is an issue with the config file.

\*/

void setDefaultAlarms() {

alarms[0].hour = 0;

alarms[0].minute = 0;

alarms[1].hour = 4;

alarms[1].minute = 0;

alarms[2].hour = 8;

alarms[2].minute = 0;

alarms[3].hour = 12;

alarms[3].minute = 0;

alarms[4].hour = 16;

alarms[4].minute = 0;

alarms[5].hour = 20;

alarms[5].minute = 0;

numAlarms = 5;

}

/\*\*

\* Parses the config file on the SD card and stores the alarm times.

\*

\* @return whether or not the file is the appropriate format.

\*/

bool getTimes() {

// buffer for sorting

AlarmTime buffer;

// counters

byte i,j;

// open config file

File timeFile = SD.open("config.csv", FILE\_READ);

// counter for number of alarms

int alarmCounter = 0;

// the character being looked at

char c;

// the time to be added

String timeAdd = "";

// return boolean

bool returnBool = true;

while(timeFile.available() && alarmCounter < MAX\_READINGS) {

c = timeFile.read();

// if colon, the previous numbers were hours

if(c == ':') {

alarms[alarmCounter].hour = timeAdd.toInt();

timeAdd = "";

}

// if digit, add to string to be added

else if(isdigit(c)){

timeAdd += c;

}

// if new line, add the remaining number

else if(c == '\n') {

alarms[alarmCounter].minute = timeAdd.toInt();

timeAdd = "";

alarmCounter++;

}

// if any other characters are contained in the file, it is not valid

else {

if(c != '\r') {

returnBool = false;

break;

}

}

}

// add the last character

if(isdigit(c)) {

alarms[alarmCounter].minute = timeAdd.toInt();

timeAdd = "";

alarmCounter++;

}

else {

if(c != '\r' && c != '\n' && c != 10) {

returnBool = false;

}

}

// set number of alarms to number of alarms counted

numAlarms = alarmCounter;

// organizes array into ascending order

for (i=0; i<(alarmCounter); i++) {

for (j=i+1; j<(alarmCounter); j++) {

if (alarms[i].hour > alarms[j].hour) {

buffer = alarms[i];

alarms[i] = alarms[j];

alarms[j] = buffer;

} else if (alarms[i].hour == alarms[j].hour) {

if (alarms[i].minute > alarms[j].minute) {

buffer = alarms[i];

alarms[i] = alarms[j];

alarms[j] = buffer;

} else if (alarms[i].minute == alarms[j].minute) {

//need to add code to get rid of previous equal

//maybe set to 99:99

alarms[j].hour = 25;

alarms[j].minute = 25;

numAlarms--;

}

}

}

}

// finds soonest alarm time to set

time\_t currtime = RTC.get();

int currHour = hour(currtime);

int currMinute = minute(currtime);

int nearAlarm;

for(i = 0; i < numAlarms; i++) {

if(alarms[i].hour > currHour || (alarms[i].hour == currHour && alarms[i].minute > currMinute)) {

nearAlarm = i;

break;

}

}

alarmCount = nearAlarm;

return returnBool;

}

/\*\*

\* Sets the alarm to the time held in alarmCount.

\*/

void setAlarmTime() {

int minuteAlarm = alarms[alarmCount].minute;

int hourAlarm = alarms[alarmCount].hour;

// set Alarm 1 time

RTC.setAlarm(ALM1\_MATCH\_HOURS, 0, minuteAlarm, hourAlarm, 1);

//RTC.setAlarm(ALM1\_MATCH\_HOURS, 0, 59, 15, 1);

// clear RTC flag

RTC.alarm(ALARM\_1);

// allow interrupt

RTC.alarmInterrupt(ALARM\_1, true);

}

/\*\*

\* ISR for when Alarm1 goes off.

\*/

void alarmISR() {

}

/\*\*

\* Takes measurement with motor controls.

\*/

void takeMeasurementWithMotor() {

int timeDelay;

float depth;

float prevDepth = 0;

float diff;

float prevDiff;

if(goUp) {

timeDelay = 1500;

while(true) {

depth = takeMeasurement(false);

diff = depth - highest;

prevDiff = abs(prevDepth - depth);

// turn off motor control if the sensor hasn't moved

if(prevDepth != 0) {

if(prevDepth < 0.05 || prevDiff == 0.05) {

motorOn = false;

break;

}

}

else if(diff < 0.1) {

goUp = false;

break;

}

else if(diff < 0.2) {

timeDelay = 700;

}

// if the sensor is going up instead of down, turn off

// this has happened when the rope gets tangled

else if(diff < 0 || diff == 0) {

motorOn = false;

break;

}

prevDepth = depth;

digitalWrite(UP, HIGH);

delay(timeDelay);

digitalWrite(UP, LOW);

}

}

else {

timeDelay = 700;

while(true) {

depth = takeMeasurement(false);

diff = lowest - depth;

prevDiff = abs(depth - prevDepth);

if(prevDepth != 0) {

if(prevDiff < 0.05 || prevDiff == 0.05) {

motorOn = false;

break;

}

}

if(diff < 0.1) {

goUp = true;

break;

}

else if(diff < 0.2) {

timeDelay = 500;

}

else if(diff < 0 || diff == 0) {

motorOn = false;

}

digitalWrite(DOWN, HIGH);

delay(timeDelay);

digitalWrite(DOWN, LOW);

}

}

}

/\*\*

\* Takes measurement from the sensor and stores it to an SD card.

\*/

float takeMeasurement(bool setUp) {

turnOnSensor();

requestMeasurement();

String data = requestData();

float depth = 0;

if(!setUp) {

recordData(data);

}

depth = getDepth(data);

turnOffSensor();

return depth;

}

/\*\*

\* Parses the string of data given by the sensor.

\*

\* @param the data to be parsed.

\* @return the CSV line to be added to the logger file.

\*/

String parseData(String data) {

// the line to be added to

String line = "";

// add timestamp to the line

time\_t currtime = RTC.get();

line += month(currtime);

line += "/";

line += day(currtime);

line += "/";

line += year(currtime);

line += ",";

line += hour(currtime);

line += ":";

if(minute(currtime) < 10)

line += "0";

line += minute(currtime);

line += ",";

// the string to be added to the line

String toAdd = "";

char c;

// check for negative sign of the first reading

if(data.charAt(1) == '-')

line += "-";

// loop through characters in data;

for(int i = 2; i < data.length(); i++){

c = data.charAt(i);

if(c == '+') {

line += toAdd;

line += ",";

toAdd = "";

}

else if(c == '-') {

line += toAdd;

line += ",-";

toAdd = "";

}

else if(i == data.length() - 1) {

toAdd += c;

line += toAdd;

toAdd = "";

}

// N is a character added between the 2 data readings from the sensor to differentiate

// the two bits of data.

else if(c == 'N') {

i++;

}

else {

toAdd += c;

}

}

return line;

}

float getDepth(String data) {

String depth = "";

char c;

int measureCount = 1;

bool hasStarted = false;

for(int i = 0; i < data.length(); i++) {

c = data.charAt(i);

if(measureCount == 9) {

if(c == '+' || c == '-') {

if(!hasStarted){

depth += c;

hasStarted = true;

}

else {

break;

}

}

else {

depth += c;

}

}

else {

if(c == '+' || c == '-'){

measureCount++;

}

}

}

return depth.toFloat();

}

/\*\*

\* Adds the given line of data to the logger CSV file.

\*

\* @param the data to be added.

\*/

void recordData(String data) {

// parse the data and make it into the formatted line to be added

String line = parseData(data);

// the logger file to be written to;

File logfile;

// the filename of the logger file

char filename[] = "logger.csv";

// if it doesn't exist, create the file and it's header

if(!SD.exists(filename)){

logfile = SD.open(filename, FILE\_WRITE);

logfile.println("Date,Time,Temperature,pH,Specific Conductance,Salinity,DO %Saturation,DO mg/L,ORP,Depth,Battery");

}

// if it exists, open the file

else {

logfile = SD.open(filename, FILE\_WRITE);

}

if(!logfile) {

}

else {

// add line to file and close the file

logfile.println(line);

logfile.close();

}

}

/\*\*

\* Requests the measured data from the sensor.

\*

\* @return the String of data.

\*/

String requestData() {

// boolean to wait for response

bool responded = false;

// the sensor response

String response = "";

// the response to be returned

String returnResponse = "";

// get second line of data

// wait for valid response

while(!responded) {

// send "report data" command for first line of data

mySDI12.sendCommand("0D0!");

// wait a bit

delay(30);

// get response

response = decodeResponse();

// valid response

if(!response.equals("0+0.00+0.00+0.000+0.00+0.0") && !response.equals(""))

{

responded = true;

}

// a response of '0' means that the sensor needs to be restarted

else if (response.equals("0")){

turnOnSensor();

requestMeasurement();

}

}

// add response to return string

returnResponse = response;

// reset for second line of data

responded = false;

response = "";

// get second line of data

// wait for valid response

while(!responded) {

// send "report data" command for second line of data

mySDI12.sendCommand("0D1!");

// wait a bit

delay(30);

// get response

response = decodeResponse();

// valid response

if(!response.equals("0+0.00+0+0.000+0.0") && !response.equals(""))

{

responded = true;

}

}

// add 'N' in between first and second lines to make parsing easier

returnResponse += "N";

// add second response to return string

returnResponse += response;

return returnResponse;

}

/\*\*

\* Commands the sensor to take a measurement.

\*/

void requestMeasurement() {

// boolean to wait for response

bool responded = false;

// string for response

String response = "";

// wait for valid response

while(!responded) {

// send measurement command

mySDI12.sendCommand("0M!");

// wait a bit

delay(30);

// get response

response = decodeResponse();

// if response contains characters, the sensor responded

if(response.length() > 0)

{

responded = true;

}

}

// parse response to figure out how long to wait

String waitFor = "";

for(int i = 1; i < 4; i++) {

waitFor += response.charAt(i);

}

unsigned int timed = waitFor.toInt();

// delay for a period of time

timed = timed \* 1000;

delay(timed);

}

/\*\*

\* Turns on the sensor.

\*/

void turnOnSensor() {

// boolean to wait for response

bool responded = false;

// wait for valid response

while(!responded) {

// send sensors on command

mySDI12.sendCommand("0X1!");

// wait a bit

delay(30);

// get response

String response = decodeResponse();

// check if response is valid

if(response.equals("0X1")) {

responded = true;

}

}

}

/\*\*

\* Turns off sensor.

\* Note: I have tried making the on/off functions the same, and for some reason the

\* sensor doesn't like it.

\*/

void turnOffSensor() {

// boolean to wait for response

bool responded = false;

// wait for valid response

while(!responded) {

// send sensors off command

mySDI12.sendCommand("0X0!");

// wait a bit

delay(30);

// get response

String response = decodeResponse();

// check if response is valid

if(response.equals("0X0")) {

responded = true;

}

}

}

/\*\*

\* Gets the response from the sensor by getting all of the available bits from

\* the data line.

\*

\* @return the response of the sensor.

\*/

String decodeResponse() {

// string to hold response

String sdiResponse = "";

// loop through available data bits

while(mySDI12.available()) {

// read character

char c = mySDI12.read();

if ((c != '\n') && (c != '\r')) {

sdiResponse += c;

delay(5);

}

}

// flush the data line

mySDI12.flush();

return sdiResponse;

}

void loop() {

attachInterrupt(5, alarmISR, FALLING);

LowPower.powerDown(SLEEP\_FOREVER, ADC\_OFF, BOD\_OFF);

detachInterrupt(5);

// set to next alarm

if(alarmCount == numAlarms-1){

alarmCount = 0;

}

else {

alarmCount++;

}

// clear alarm flag

RTC.alarm(ALARM\_1);

if(motorOn) {

takeMeasurementWithMotor();

}

else {

takeMeasurement(false);

}

// set to next alarm

setAlarmTime();

delay(5000);

}