Spring 2013

Wireless Water Flow Meter Network in the Great Bay

Braden S. Blanchette
University of New Hampshire - Main Campus, bsz28@wildcats.unh.edu

Wireless Water Flow Meter Network in the Great Bay

ECE 792H Final Report/Senior Thesis

Author: Braden Blanchette

Faculty Advisor: Professor Nicholas Kirsch

Courses Involved: CS 410, ECE 651, ECE 649, ECE 796, ECE 791H, ECE 792H

Final Report Submission: Monday, May 20, 2013

Estimated Project Completion Date: Fall, 2013

Abstract:

The Oyster Restoration Program alongside the New Hampshire chapter of the Nature Conservancy is working towards developing new oyster beds throughout the Great Bay. Sedimentation is proving to be a vast problem by covering up the beds before they have a chance to grow to a healthy level. The many rivers entering the Great Bay are bringing the sediments from all over the region and limiting the ability of the program to develop the new beds. They need a way to measure the sedimentation rate, by measuring the flow rate of the rivers over a single tidal cycle in various locations throughout the bay. This is done simply by the design of a wireless water flow meter network. Using a Price Meter as the measurement tool and an Arduino UNO to organize the data, the Oyster Restoration Program can monitor the characteristics of the locations to gain a better understanding of the location as a potential site for a new oyster bed. The design of an self contained system to extract and store the data to be collected is essential to speed up the process of monitoring these locations, which the device developed here will do.

Contents

I. Overview

I a. Background

The Great Bay is roughly 7,300 acres of open water in southeastern New Hampshire connecting many of the state's rivers and tributaries to the Atlantic Ocean. This body of water east of the University of New Hampshire campus was home to Oysters that were an essential part of the ecosystem of the Great Bay for hundreds maybe even thousands of years. These prehistoric creatures have the ability to filter the water, taking out nitrogen based pollutants, and providing healthy living for other organisms sharing the habitat. They were a food source for the Native American and European settlers in the early years of our country and were abundant until commercial fishing came close to wiping them out in the late 1800's. When regulations were placed on the harvesting of the oysters, the population grew back to healthy levels until the protozoa MSX and Dermo came close to wiping them all out in the mid 90's. This creates a serious problem for the ecosystem as the oysters are essential for other life to exist. Here at the University of New Hampshire, there is an organization that is dedicated to promoting the growth of a healthy, disease resistant population of the oysters. This program, known as the Oyster Restoration Program[1], is headed by Professor Ray Grizzle. By recycling old oyster shells from local beds and restaurants, the program works closely with the non-profit Nature Conservancy to reintroduce new oyster beds throughout the Great Bay.

I b. Problem Definition

Water in the Great Bay is a combination of salt water from the Atlantic and fresh water from rivers flowing from all over New Hampshire. Along with this inflow of water from the rivers and tributaries, comes an inflow of sediments and runoff from all over the state. Damming of the rivers and logging operations occurring even way up in the White Mountains leads to silt and pollutants making their way down the river into the bay. This proves to be a problem for new oyster beds as the sediment can cover the base of the beds before they are able to really start growing. An important step to reintroducing the oysters is to track the water flow and other useful information in various potential oyster bed locations.

I c. Goal of Project

The purpose of this project was to design a network of water flow meters that can track the changing current over one tidal cycle. With additional assistance, other sensors that can give more information about the water column will be included. The official location(s) where the measurement within the Great Bay will be taken has yet to be determined, but will most likely be at the mouths of various rivers. The frequency at which measurements are taken will be determined based on the request of the Oyster Restoration Program to meet their requirements.

I d. Specific Design Objectives

For the scope of this senior design project, two sensors in the network were designed, built, tested and will be used as a working prototype for the continuation of the project. Part of the project will be working with members of IEEE and St. Thomas Aquinas High School through IEEE's Engineering Projects In Community Service (EPICS) program to build a network of the meters and include other sensors such as temperature and salinity of the water to better understand the location to determine whether or not it would be suitable for an oyster bed.

II. Design Methods

II a. Design Overview

The device is designed to use a water flow meter to measure the current speed in the water over a predefined period of time. To do this, a water flow meter design was selected to meet the requirements of the oyster restoration program and to work in the waters of the Great Bay. The design of a sensor circuit to go with a microcontroller to calculate the speed of the water flow was designed to best calibrate the sensor. The overall design is shown in Figure 1. Using an Arduino microcontroller to take an input from the water flow meter in the water column and the sensor circuit and storing it on an SD card, along with processing GPS information to allow accurate timing of the measurements. The flow meters can be purchased with professionally designed data loggers. However, designing the data logger with an Arduino allows other sensors to be added and calibrated to do what the user wants in a simple way. The flow meter was chosen based on budget availability to be able to provide the most accurate results. The buoy is designed to keep the electronics dry and keep the meter at a fixed depth below the surface to provide the flow information at the correct portion of the water column.

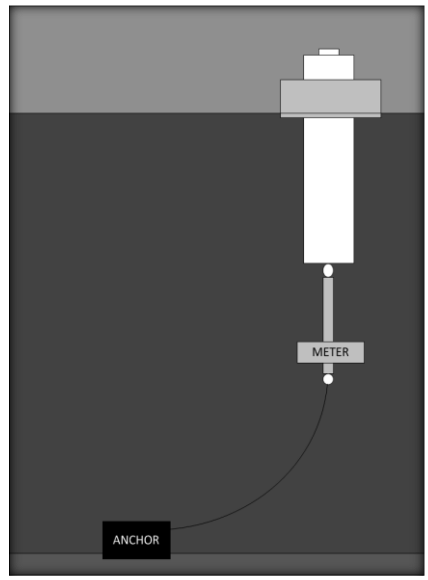

Figure 1: Overall Device Design

II b. Electronics System

This electrical system, shown in Figure 2, will be contained underwater, so it needs to be in a watertight housing. After meeting with Andy McLeod who has worked on similar project designs at the University of New Hampshire's Center for Coastal and Ocean Mapping (CCOM), a common design using some PVC pipe, was decided to be the best option to fit within the budget. This enables the battery and electronic systems to be enclosed safely underwater and minimize the wiring that will go through the water. Since the device will be in a relatively cold body of water, insulation will have to be considered to be sure no condensation could short the system. The PVC pipe will provide the insolation necessary, and silica gel packets are inserted into the case to absorb the moisture in the air, to prevent condensation, and

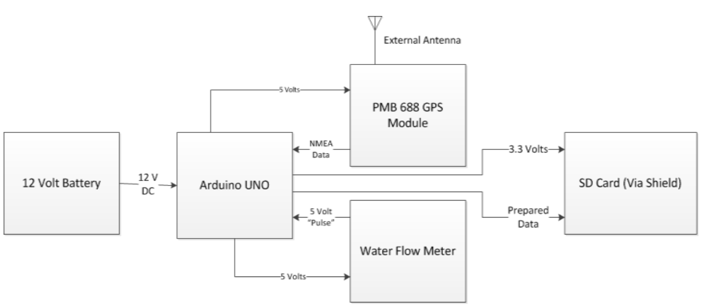

Figure 2: Electronics Block Diagram

electronics failure.

The sensor network is designed based on the water flow meter used and how the microcontroller accepts the input information. The information from the flow sensor and the GPS synched time and location data, is stored in an SD card for easy extraction. Saving the data in a comma-separated values (.csv) file will enable simple post processing to be done by the Oyster Restoration Program and the Nature Conservancy to better understand the location. The system specifications are that it needs to run for a minimum of one tidal cycle. Since the user may not be able retrieve the device right away, the system can keep running, collecting data to provide more useful information outside of the requirements of the user. To ensure that the device will keep running, the battery was selected to power the electronics for a minimum of 30 hours. This will allow the device to keep collecting data that could prove useful for the restoration of the oyster beds.

III. Description of Individual Components

III a. Microcontroller and Data Storage

(i) Arduino UNO R3

The third generation Arduino UNO is used to control the system. There are a number of Arduino microcontrollers available. Each have similar designs but do contain various differences. The Arduino UNO is the standard board, the Arduino Mega is a larger version and the Arduino Leonardo is a new version that utilizes USB connection within the central processor allowing it to be interfaced with a computer in the same way a mouse or keyboard would be. The UNO was enough to control this device and the lower cost, and compatibility with many different attachable shields made it the right choice for this application.

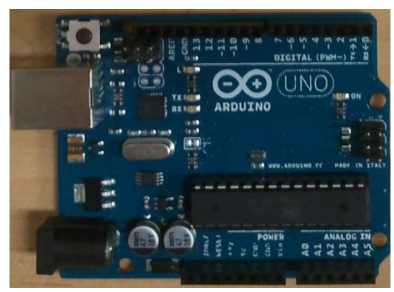

Figure 3: Arduino UNO R3

The Arduino microcontroller is an open source device, meaning anyone can use code that is posted by other users for their own design. The website for the device also has a large reference section containing a description of available libraries, program syntax, and a help forum. In addition to being simple to use, the Arduino is a powerful microcontroller for the price. The Arduino UNO uses the ATmega328[2] processor and places the pin connections on the outside of the board to be easily accessed and enabling stackable shields to be used. The ATmega328 has 14 digital pins which can be used as an input or output pin. Six of the digital pins are capable of providing Pulse Width Modulation (PWM) which can be used for motor control, LED fading etc. Additionally, there are six analog input pins. They are connected to a 10 bit A/D converter which can measure an input of 0-5 volts at integer values between 0 and 1023 (2^{10} levels). This enables the user to set a threshold level between the high and low adjusting for noise or loss in the system. Power can be drawn from a USB computer connection, or from the DC jack or V_{IN} pin. The DC jack and V_{IN} pin are connected to an onboard regulator allowing a safe input voltage from 6V to 20V to provide the regulated 5 Volts to power the ATmega chip. In addition, the chip can also provide a 3.3 V level with the 5V source.

(ii) Adafruit Data Logging Shield

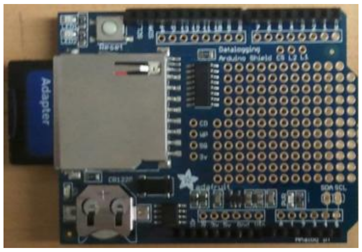

Figure 4: Adafruit Data Logging Shield

The Arduino UNO R3 has only minimal data storage capabilities, with 32KB of Flash, 2KB of SRAM and 1KB of EEPROM. For large data collection requirements, interfacing with external storage is essential for long term logging. The simplest and most convenient way is through the use of an SD card. SD cards range in size and are a low cost form of removable memory. For interfacing with the Arduino, the Adafruit Data Logging Shield is used. This shield stacks on top of the Arduino UNO allowing all pins to be accessed in the same way they would on the Arduino board itself. The SD card requires use of the serial communication pins 10, 11, 12, and 13[3]. The other 16 pins are open for use by other sensors. Onboard the shield, there are two LEDs with series resistors included, one red and one green. This is used for convenient field notifications of the functionality of the device. For this system, when the green LED is on, the device is measuring. When the green LED is flashing, the system is resting. When the red LED is on, there has been an error in the system, indicating that the system is not running, and conserving power. These LEDs can be used in any way the user may want.

Along with the SD card, the shield also includes an on board prototyping area for more confined systems. This area was not utilized, to allow for easier component replacement. Also included is a real time clock (RTC) for applying a time stamp to the logs. There are simple Arduino libraries that can operate this functionality, but the time stamp is being taken from the GPS unit for this system.

III b. Sensors

(i) Water Flow Meter

The sensor used for measuring the flow of the water coming in from the rivers is a Price Type analog meter from Gurley Precision Instruments.[4] This device uses a bucket wheel with 6 conical buckets that rotate with the flow of water. The rotation of the bucket wheel rotates a shaft inside the device, which makes contact with the upper contact wire once a revolution and the lower contact wire once every 5 revolutions as seen in Figure 6, completing a circuit from the binding post. Measuring along the chassis allows the user to detect when the circuit is completed, and therefore, when a rotation of the bucket wheel has been made. These rotations can then be converted to flow velocity by Equation 1.

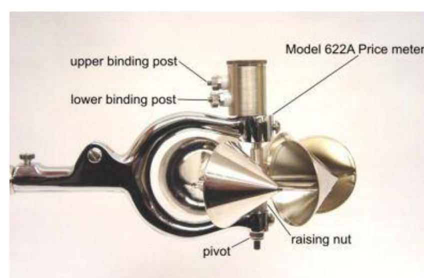

Figure 5: Price Meter

The Price Meter is rated to read velocities ranging from 0.2 – 25 feet per second with 2% accuracy. With the analog meter, this is typically done with a headphones attachment. The headphones have an internal 1.5 volt battery that sends a voltage down to the binding post of the meter. The second lead is attached somewhere along the chassis to then detect when the circuit is completed. An audible clicking sound is heard in the headphones, as the user counts how many clicks over a sample period, a calibrated table[1], or Equation 1, is used to convert counted revolutions to velocity. With the Arduino, using an analog input pin will act as the listening device, counting when the revolutions are made. Connecting the sensor directly to the Arduino causes some erroneous data from the

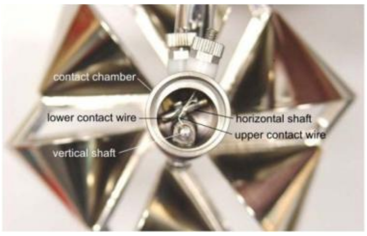

Figure 6: Bucket Wheel of Price Meter

(ii) Global Positioning System

To better track when and where the measurements are being made, using a Global Positioning System module is required. This system uses a PMB 688 GPS module with external antenna to track the location and time of the measurements. This module has the ability to track satellites on 20 parallel channels. The GPS uses TTL communication lines to send NMEA sentences to the Arduino. These sentences contain the date and time in Universal Time, and location information used with this device, along with other validation information like the number of satellites and checksum verification data. The Arduino then uses the TinyGPS library[6] to parse the data from the NMEA sentences into strings to be saved to the SD card along with the flow velocity data.

Figure 7: PMB 688 GPS Module

III c. Power

To power the device, a single 12 volt battery can be used. The Arduino's DC regulator can safely accept voltages between 6V and 20V to provide a steady 5V power source for the ATmega chip. The low current draw of the Arduino UNO, SD card, GPS module, and flow meter allows the battery capacity to be kept low and still have long life for the device. Measuring the current draw of the devices resulted in a current of 110mA when the device was running. To find the battery capacity needed to operate for 30 hours to account for the device being left in the field longer than intended, discharge rate and temperature considerations should be taken. Assuming a minimum average low of -11°C[7] ,even though the buoy will insulate the battery and never allow the battery to reach this temperature, and using the graphs for finding the maximum depth of discharge (MDOD) and discharge rate capacity adjustments for the low temperature[2] the needed battery capacity can be calculated. For this, a standard discharge rate of C/20 is assumed for best performance. From Figure D1, the MDOD is found to be ~80%. Figure D2 gives a capacity adjustment of ~75%. Equation 2 results in a required battery capacity of 5.5Ah.

[1] See Appendix C
[2] See Appendix D

This means a 12V, 5.5Ah battery is acceptable. Lower range batteries like this are very similar price, so to account for any changes in current draw and other potential physical characteristics degrading the battery capacity, a 12V 10Ah battery was selected. This will also help extend the overall battery life from avoiding excessive charging and discharging levels as well as accounting for longer durations of deployment in the future.

III d. Buoy Design

To keep the electronics safe from the water, and mark the location of the flow meter, a buoy is needed. The buoy needs to be completely sealed but still able to be opened to insert and remove the electronics as needed. A common design practice is to use some sort of waterproof cylinder, sealing on both ends to hold the electronics and act as the buoy. Material for the buoy could be from various plastics. Polyvinyl Chloride is a common plastic used for piping and can be easily purchased and cut to size. This buoy uses a two foot length of 6" Schedule 40 PVC pipe. Various end caps are available for sealing. Sealing is done with PVC cement which is a special bonding agent that forms a permanent bond, welding the cap to the pipe. This end cap has a hole for the anchor line bolt. The bolt is sealed with adhesive sealant intended for through hull applications on boats. This polyurethane sealant is stronger than silicon and if applied properly, creates a permanent seal.

Adding an adapter to one end enables the pipe to be closed using a screw on plug. This end will be best suited above the waterline since it is not a permanent seal. To ensure a proper seal, thread tape is added to the threads of the plug, and removable silicon is applied around the outside. This will be sufficient to keep rain or splash water from entering the buoy. Keeping water out is absolutely essential to keep the electronics in operation.

The Price Meter needs to have a wired connection to the Arduino which requires an opening in the buoy. For this, cable glands were purchased to create a tight seal around the cable at the entry point to the buoy. The cable gland is a gasket with a cap that tightens that gasket around the wire creating a removable seal. To ensure that this seal is completely waterproof, a layer of electrical tape can be applied as an added precaution. The gland is placed in the screw on plug and sealed with silicon to limit the possibility of error that having the opening submerged could cause over time.

A foam collar is added to the buoy both to provide insurance of the extra flotation, and to help keep the buoy level in rougher waters. The large diameter and height of the pipe supplies efficient buoyancy for the sensor, battery and other electronics, but adding the collar will protect the buoy from being pulled under if the flow meter gets clogged with vegetation, adding weight to the device. The foam used is simply dense sheets of insulation foam, cut to the shape of the buoy, and wrapped in duct tape to prolong the life of the foam in the brackish water of the bay.

The buoy is anchored to the floor of the bay with a 12 lb river anchor and some nylon braded rope. The use of the lighter rope as opposed to chains both brings down the cost of the project and helps allow the flow meter, which is attached to a stainless steel bar in line with the anchor, to rotate with the changing currents. A flag will be added to buoy as an additional marker showing the involvement of UNH IEEE and St. Thomas Aquinas High School.

IV. Testing Methods

Due to the delay in funding for the project, testing of the design is still in process. The Arduino and GPS were purchased first and code was compiled to open communications between the microcontroller and the GPS. With the Arduino being open source, this code was derived from the various examples available from the Arduino website, and what was included with the TinyGPS library. The GPS module only works in direct line of sight of the sky. When placed in a window, the GPS would only occasionally make a lock with the GPS satellites. For the GPS data to be seen as accurate, a minimum of three satellites need to be locked to the device. When this doesn't happen, the code doesn't shut off, but just prints to the SD card that the GPS data is Invalid. This way, if the GPS cuts out at some point, knowing the timing of the device, the user can still infer when the data was collected.

Before the flow meters were ordered, the design was implemented using a function generator outputting a 1% duty cycle square wave at various frequencies. Using pin 2 on the Arduino as an interrupt monitor, the rising edge of the pulse was detected, incrementing the counter. This is a low power setup that only runs the code when the impulse is detected. When the funding was approved and the meters were purchased, testing showed that this interrupt method was detecting more than just a single pulse for each revolution from the meter pin bouncing. To overcome this error, implementation as an analog input was used. This way, a threshold voltage could be set for the counter so little jumps in voltage wouldn't trigger the counter. With this set up, the readings were much more accurate, but there was still some error in the readings due to the speed of the Arduino measuring the voltage at the pin. Initially, changing the read time was helping to correct the errors, but it didn't completely remove the cause of error completely. This error, known as bouncing, is common with setups of switches or pushbutton. Since the meter is essentially a mechanical switch that is turned on and off quickly, there is some inevitable bounce in the voltage. This is overcome by inserting a pull down resistor into the circuit. The pull down resistor connects the analog pin to ground so that when the meter pin is not in contact with the shaft, the voltage remaining on the line from the previous contact, is pulled to ground faster, recreating a true impulse as closely as possible. This design produced much cleaner results in the counter.

After the bench-top testing was complete, underwater testing was needed. The first of this testing was done in a sink filled with water. When the binding post was left open to the water with the 5 volts on the post, there was a short measured due to the water conducting the electricity. This post was sealed by wrapping electrical tape around it and up the wire. This proved efficient enough to measure the voltage on the chassis at the analog input pin.

Next was the build and testing of the buoy. After the bottom cap was sealed on, a simple float test was done measuring the weight handling capabilities of the pipe. In a barrel of water, sand was added to the pipe to calculate how much weight the pipe would need as a ballast to keep the buoy floating upright and not on its side. It was found to be a weight of roughly 15-20lbs. The batteries weigh roughly 7 pounds so at least another 10 pounds is needed. To add this weight, different measures could be taken. Sealing sand in the bottom is one method, while another is lead or concrete weight. This idea will be approached more when the sealing of the buoy is completed.

When holding electronics underwater, the single most important aspect of the design is making sure everything is waterproof. The PVC cap and adapter connections are all sealed with PVC cement creating a watertight bond trusted by plumbers and tested under pressure to ensure that there is not leakage. However for the buoy, there is a hole for the anchor bolt and holes for the wire to the meter and the GPS antenna. Polyurethane adhesive sealant is used on the bolt hole to provide a permanent tight seal. After initial testing in the pools of CCOM, the sealant was found to have a leak. More sealant was applied and additional epoxy will be applied as well as foam sealant on the inner parts of the buoy to help protect this seal. A lot of strain is put on the seal from the weight of the meter and anchor line in the water flow as well as the battery resting on top of the bolt. Additional sealant is necessary to ensure that the seal won't break over time.

The delays in the funding have limited the time available to test the device in its entirety. Following the tests to ensure that the buoy is sealed and the meter, GPS and Arduino are all functioning properly; the meter will be deployed into the bay to start collecting data. The following testing plan will be implemented:

1. Finish device set up; add applicable protection for electronics in the buoy
2. Test the functionality of the sensor in the bay: Allow some readings to be taken and check to see that they are similar, valid results
3. Coordinate with the Oyster Restoration Program to set up deployment using their boats to bring the two made devices to the locations in question.
4. Assist with initial deployment and data collection
5. Verify that the data collected is valid and everything functioned as planned; attempt to correct any errors or faults in the system.
6. Once the design is verified to work, turn over the devices to the Oyster Restoration Program to use as they please.

Following the preceding testing plan will enable a hands on approach for all involved from beginning to end of the design.

V. Results

The design of this device has been delayed enough from the lack of funding to not produce results that where envisioned at the start of the project. The term "wireless" in the title of the project has not been realized, aside from being able to remove the data from the Arduino via the memory card. The project is going to be turned over to the IEEE UNH group to improve and also to assist the Oyster Restoration Program with future deployments.

The design of the buoy went through a few changes after further research and testing was done showing the initial design was insufficient. Originally, the flow meter was to be purchased from the

Rickly Hydrological Company. They stocked meters similar to the flow meter purchased, and at lower cost. However, after further discussion with some of the technicians from the company, it was found that the meters do not work with an Arduino. The company was in process of making circuitry to work with the Arduino, but would not disclose any more information as they were intending on selling it. Not willing to make a large financial gamble, the Gurley flow meter was more likely to work, so this was chosen for the project. Initial testing has shown that this device should work properly, as long as appropriate sealing techniques are taken.

Finding the final design followed a more circular path. The initial design more closely matched the final result. Throughout the design process, different concepts were examined. A design with two anchors to hold the buoy more stationary was discussed, but Andy McLeod of CCOM stated that in his experience, one anchor is always best to avoid lines tangling. This led the design back to the initial one anchor line as seen in Figure 1.

Initial testing of the buoy showed that the sealant used for the anchor bolt hole was insufficient after one application. The stress caused by the weight of the meter and the battery on the bolt caused the seal to come undone, allowing water to enter the buoy. Adding additional sealant is necessary to prevent this from occurring again. To attach the bolt, a simple nut and washer was used. This however could be improved upon by the use of a rubber gasket in addition to the steel washer to provide a tight seal against the PVC cap. Since the cap is not completely flat, there is little surface area that the washer covers on the cap. A rubber gasket would form to the shape of the surface, creating a tighter seal. Since there is a thick coat of sealant on the gasket, the rubber washer will be used on future variations of the design.

V c. Future Improvements

The design of the device is complete, but there is definitely room for improvement. What exists now meet the requirements set by the initial design request from the Oyster Restoration Program, but certain aspects of the design could be improved. The design of the buoy has not been adjusted much from the original prototype. There are sure to be improvements that could be made both in the way it is sealed, like the addition of the rubber gasket, and in the overall design to better suit the possible additions of other devices.

(i) Thermistor Flow Meter

The possibility of improvement of the device starts with the most important component. The device is made to measure the water flow entering the bay from the surrounding rivers. The flow meters used in this device are high cost devices due to the importance of the calibration of the bucket wheel rotations. Another alternative that has been researched is the use of a thermistor as a flow meter. The thermistor is a heat variable resistor, meaning the resistance changes as the temperature changes.[8] This idea can be used to measure water flow. As the water flows across a calibrated thermistor, it will pull heat from the component. The power needed to maintain the voltage on the thermistor can be directly related to the velocity of the flow of the water based on heat diffusion. This idea is often used as anemometers to measure air flow[9] and can be modeled similarly in water flow analysis. An early model of the thermistor flow meter utilizes a Wheatstone bridge[3] to measure the flow with resistors, the sensing thermistor and a calibrated temperature thermistor to match. When the water flow pulls the heat off the sensing thermistor, the bridge will pull more power to balance all legs. This power can be measured to relate to the flow rate. The calibration is done by comparison based on still water and moving water to analyze the difference and calculate the heat-exchange coefficient of the water to calibrate the relationship. This heat exchange is affected by heat conduction, convection and radiation which all determine the coefficient.[10]

This design has been done in the past but requires a lot of testing to ensure proper calibration. For this device, there needed to be a guaranteed working meter, so the mechanical meter was selected. However, if a thermistor flow meter can be developed; there are numerous advantages to their use:

1. The thermistor is a small, low cost component and would require minimal power to run.
2. There are no moving parts, minimizing the possibility of vegetation interfering with the device
3. The simplicity of design and small size could enable multiple meters throughout the water column, or even in a grid design which could give more inclusive flow analysis

The use of thermistor flow meters in the future would make extending the device network a simpler and cheaper task.

(ii) Additional Sensors

While the need for flow rate data is most important for the Oyster Restoration program's use, including other information of the deployment area could prove to be valuable. Some of the possibilities include temperature sensors and salinity or pH sensors.

Sensors to test salinity and pH are readily available for interfacing with an Arduino. A simple salinity[4] sensor is done by measuring the conductivity of the water. One way of doing this is to measure the voltage drop in a voltage divider with on leg of the divider in the water.[11] This is done simply in an Arduino program by measuring the voltage as an analog input and seeing how it differs from the calibrated value of distilled water. From this, conductivity can be determined, and using conversion equations, the salinity can be found. This process could be done all in the Arduino code, or to save processing time, done after the initial readings are taken and saved.

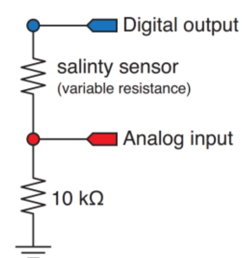

Figure 8: Salinity Sensor in a Voltage Divider

A pH sensor is typically a more involved process. pH[5] probes are available for purchase, with some companies like Atlas Scientific, making breakout boards to simplify the measurement process. There is a higher cost involved with these sensors, but if there is a need for the pH measurement to

[3] A circuit used to measure an unknown resistance of one of its legs
[4] Salinity is essentially the amount of salt in the water
[5] pH is a measurement of the acidity of the water based on the excess or deficiency of Hydronium ions

better understand the oyster bed location, the simplicity of interfacing in a single device would prove useful.

In order to fully understand the location of the proposed oyster bed, knowing how the temperature fluctuates throughout the tidal cycle could prove useful. Oysters are robust organisms and are present in the environment of the Great Bay to begin with, so the temperature is mostly a non-issue. However, it is simple to measure the temperature, and give a better picture of the location. Measuring the temperature can be done using a single thermistor in a voltage divider set up similar to the salinity sensor. By measuring the change in resistance across the thermistor, the temperature can be calculated from the calibration of the thermistor. Some thermistors can be very sensitive, changing as much as 100Ω with each degree fluctuation. This resistance change can be converted to temperature using the Steinhart-Hart equation, and more simply, the simplified B parameter equation (Equation 3)

— — — —

Where T= temperature, T_0= Room Temperature (25°C), B = Thermistor Coefficent, R=resistance measured across thermistor, and R_0 = resistance of thermistor at room temperature.[13] The low cost of the thermistor and the simplicity of design makes the temperature sensor a simple and logical improvement for the device.

(iii) Wireless Data Transmission

In order to make the devices truly wireless, a method of wireless data transmission needs to be utilized. There are multiple options available for doing this, and one of the possibilities is Bluetooth. Bluetooth communication refers to wireless communications over short distances in a similar manner to WiFi. The frequency is centered around 2.4GHz and can communicate between to Bluetooth modems. Connecting one modem to the Arduino[6], and another to a computer would enable the electronics to stay in the water and have the data extracted remotely. This can be done one of two ways. One way is done by setting an interrupt on the Arduino to be triggered to send the collected data to the user when the user is within range, allowing long term storage on board, and a single, longer transmission. A second possibility would be to set a single base station within range of all devices in the network and have the data be transmitted to the station from all of the devices, and stored on one single device, making data extraction easier. This idea would require a limited range however as Bluetooth communications is limited in the transmission distance. Other possibilities can be approached for this, but more in depth research is needed before a decision can be made. Allowing for wireless data transmission would enable the device to be left in the water and operating for longer periods of time.

(iv)Solar Panels

When leaving the device in the water for long periods of time, the main consideration needed is providing the device with enough power to operate long periods of time. This can be done with larger battery banks, but this would require a larger buoy. A simple way to extend battery life is to add Solar

[6] Sparkfun sells a Bluetooth modem compatible with the Arduino for ~$65: https://www.sparkfun.com/products/10268

panels. The panels can be designed to recharge the batteries in a short period of time to enable more permanent instillation of the device, or simply provide the batteries with a little extra life to enable the device to be left unattended for longer than a couple of days. The latter design would not be permanent as the battery would eventually discharge completely, but if the intent is to extend the life to a week instead of a couple of days, this design would be simpler and cheaper to implement. Either design would require the use of a charge controller. When charging a sealed lead acid battery, or any battery for that matter, the biggest concern is overcharging. Overcharging essentially begins to cook the battery as power is being put into the battery, but has nowhere to go, as the capacity of the battery has reached full power. This not only effects the life of the battery by destroying some of the chemical used to hold the charge, but could cause dangerous explosions from heating of the acid and forming hydrogen gas. To prevent this, the charge controller is placed in between the charging source and the battery. A charge controller is essentially a switch that either turns off the current flowing into the battery once it is full, or turns the current down low enough to help maintain the full charge of the battery, without damaging it. Typically in low power systems like this, a charge controller can be manufactured with only a few components. One example is a Low Dropout Voltage charge controller[14] which uses a differential amplifier, consisting of BJT transistors, with a MOSFET transistor to regulate the voltage and current. Diodes and resistors are used to maintain the voltage along with the MOSFET. This simple circuit would help keep the battery charged for longer periods of time. However, when charging batteries, it should never be done in enclosed spaces like the buoy designed here, so some ventilation would be needed to prevent the buildup of hydrogen gas in the buoy, preventing explosions.

(v) Additional Devices

The original request from the Oyster Restoration Program was to have six devices to be deployed at a time. Due to the high cost of the flow meters, the budget was cut down to build four devices. Since not all of the budget has been approved, only two devices have been built. As the devices come to completion, the money saving choices that were made along the way has left enough money in the budget to begin and possibly even complete the build of a third device. Since the design was initially still being finalized when purchasing began, only two devices were completed to be sure that everything for them could be purchased. The remaining funds could be used to make a third device or could be used for the other advancements discussed.

VI. Discussion

VI a. Constraints

There were a few constraints to the design aside from the requested specifications of the oyster restoration program. One of the major constraints on the design was an economic constraint. The water flow meter used in this device is a finely tuned instrument. There are not many different options readily available, resulting in high cost. When the design hoped to be for a network of these devices, the budget increased rapidly. Due to the high cost of the system, a significant investment is needed from some source of funding. Doing this project with the IEEE EPICs program enabled a lot of the funding necessary to be obtained, but not as much as was originally designed for. Because of this, the design had to be adjusted, and the process became held up significantly, resulting in major delays in the final product being deployed. To overcome this, other outlets will need to be explored for additional funding to improve the device. Doing so will benefit the students working on the project, giving more hands on learning and development with different components, and provide the nonprofit Oyster Restoration Program with a device that will better explore the Great Bay and determine the best locations for installation of oyster beds.

A second constraint on the system was the environment of the device deployment. A lot of consideration was needed to protect the electronics. Purchasing marine grade wiring and sealants is important to help ensure the proper operation of the system, without it degrading overtime. Sealing the buoy became the most crucial part of the design. Initial attempts were unsuccessful and if the device were to leak in the bay, the result would be electronics failure, making the device worthless. Fortunately, the largest cost of the system, the flow meter, is designed to be used in the water so loss of this device is unlikely. However, losing the electronics would still introduce unnecessary cost to the system. The approach to sealing the device may not have been the most elegant solution, so taking on a mechanical engineer, who has more practice in this sort of design could help overcome this constraint.

VI b. Professional Standards

There are many professional standards that need to be considered when designing a device like this. These standards create an acceptable guideline for various devices. Though no standards were needed for the design of this specific device, standards were used for the design of the various components. Some of these include GPS Standards, the RoHS standard, and as on all electronics the UL standard.

(i) National Marine Electronics Association Standards

The GPS standard of most concern here are the National Marine Electronics Association (NMEA) standards. These standards dictate the acceptable use of the GPS information. Typically, this is concerned with the information received from the GPS module, which is in the form of NMEA sentences. These various sentences contain the location information typically expected, but also information on the satellites used for data acquisition. The PMB 688 GPS uses the NMEA0183 v2.2 protocol in its data acquisition. The minimum required number of satellites for longitude and latitude information is three. If the user also wants altitude information, a forth is needed. This standard helps ensure accurate information is received by the GPS module.

(ii) Restriction of Hazardous Substances Directive

A lot of the devices researched for use in this system contained the RoHS stamp. This refers to a standard for the European Union known as the Restriction of Hazardous Substances Directive. This standard states what types of metals are allowed for use in the fabrication of devices. This is especially applicable when the devices are in an environment like the Great Bay, where the hazardous materials could harm the ecosystem it is introduced to. The restriction is commonly for lead and mercury, but other substances are included in the restriction. This standard hasn't been adopted by the US, but similar restrictions are in place to limit the use of lead and mercury from getting into ecosystems like the

Great Bay. This was not a consideration needed in this design, but the devices like the GPS module are designed to meet the standard.

(iii) Underwriters Laboratory Standards

The most common standard around electrical devices is the Underwriters Laboratory (UL) standard. This standard is widely recognized for its testing of devices for performance, environment, health, stability and safety standard. The marine wiring used for this device along with the jumper wires used meet the UL safety standards for the safety and acceptable operation of the wiring. This is important to follow when developing electronics to ensure proper and safe operation of the devices. Since the wiring has gone through rigorous testing, it is safe to assume the operation will occur as expected, resulting in little concern for device failure and safety issues.

VI c. Promoting STEM Education and Conservation Issues

The funding of this project was provided by the IEEE EPICS program under the agreement that it works on a conservation issue with a nonprofit while working with high school students to promote STEM education. Working with the high school students involved setting up tutorials along the way on various topics from the basics of the Arduino, to voltage regulation and simple voltage divider circuits to the development of a PCB for the device. Trying to get the students interested in technology in other STEM fields will help the growth of these fields in the United States. These constantly evolving fields needs constant growth of the education and providing interest in the pre college years is essential to see this growth continue.

Conservation has always been a hot topic, but never as much as now. Environmental decline is rampant from the human impact on the world. Trying to reverse this process is difficult, but every little bit helps. Helping the Oyster Restoration Program with their attempts to replant the oysters helps the process. By reintroducing the oysters that have been killed off from overharvesting a diseases, other organisms in the Bay will be given a fighting chance. Oysters have an unparalleled ability to filter pollution out of the water. This pollution from runoff from all over the state is harming the ecosystem of the Great Bay. Reintroducing the oysters will help reverse this process. The conservation of these oysters is essential, and promoting the education of such topics will hopefully get more people involved with this program either by volunteering with the program directly, or working on projects like this one to help their cause.

VIII. Special Thanks

- Professor Nicholas Kirsch, the project advisor, guided the design from concept to completion
- St. Thomas Aquinas High School Students and their coordinator, Bill Dinan, who assisted in the build and deployment of the devices
- Andy McLeod of the University of New Hampshire Center for Costal and Ocean Mapping for guidance in the buoy design and testing
- Ray Grizzle, head of the Oyster Restoration Program, for guidance in the design of the device
- Institute of Electrical and Electronics Engineers EPICS program and the University of New Hampshire Electrical and Computer Engineering Department for funding of the project

- Travis Coon of Gurley Precision Instruments for selling the meters at a discount and providing device insight along the way

References

[1] University of New Hampshire, "Oyster Restoration Program," 2009. [Online]. Available: http://oysters.unh.edu/. [Accessed 16 10 2012].

[2] "Arduino," October 2012. [Online]. Available: http://arduino.cc/. [Accessed 16 October 2012].

[3] B. Earl, "Adafruit Data Logger Shield," 12 April 2013. [Online]. Available: http://learn.adafruit.com/downloads/pdf/adafruit-data-logger-shield.pdf. [Accessed 22 April 2013].

[4] "GPI/Hydrological Equipment," Gurley Precision Instruments, 2004. [Online]. Available: http://www.gpi-hydro.com/. [Accessed 20 February 2013].

[5] G. M. Masters, Renewable and Efficient Electric Power Systems, Hoboken, New Jersey: John Wiley & Sons, 2004.

[6] M. Hart, "Arduiniana," 2013. [Online]. Available: http://arduiniana.org/libraries/tinygps/. [Accessed 15 March 2013].

[7] "Intellacast.com:The Authority in Expert Weather," WSI Corporation, 2013. [Online]. Available: http://www.intellicast.com/Local/History.aspx?unit=C&location=USNH0057. [Accessed 2013].

[8] M. LaBarbera and S. Vogel, "An inexpensive thermistor flowmeter for aquatic biology," 1976. [Online]. Available: http://aslo.org/lo/toc/vol_21/issue_5/0750.pdf. [Accessed December 2012].

[9] H. Fujita, T. Ohhashi, M. Asakura, M. Yamada and K. Watanabe, "A Thermistor Anemometer for Low-Flow Rate Measurements," *IEEE Transactions on Instrumentation and Measurement,* vol. 44, no. 3, pp. 779-782, 1995.

[10] H. Forstner and K. Rutzler, "Two Temperature-compensated Thermistor Current Meters for Use in Marine Ecology," *Journal of Marine Research,* vol. 27, no. 2, pp. 263-271, 1969.

[11] G. Recktenwald, "Arduino Programming Part 5: Functions Redux and Intro to Arrays," 2013. [Online]. Available: http://web.cecs.pdx.edu/~eas199/B/howto/arduino/Arduino_programming_part5_slides.pdf. [Accessed 30 April 2013].

[12] Atlas Scientific LLC, "Atlas Scientific: Sensors," 2013. [Online]. Available: https://www.atlas-

scientific.com/sensors.html. [Accessed 6 May 2013].

[13] LadyAda, "Adafruit Learning System: Thermistor," Adafruit, 02 July 2012. [Online]. Available: http://learn.adafruit.com/thermistor/using-a-thermistor. [Accessed 18 February 2013].

[14] J. Keith, "12V LDO Solar Charge Controller," Electro, [Online]. Available: http://www.electroschematics.com/6899/12v-ldo-solar-charge-control/. [Accessed 6 May 2013].

[15] D. DePriest, "NMEA Data," [Online]. Available: http://www.gpsinformation.org/dale/nmea.htm. [Accessed April 2013].

[16] Wikipedia, "Restriction of Hazardous Substances Directive," 8 May 2013. [Online]. Available: http://en.wikipedia.org/wiki/ROHS. [Accessed 8 May 2013].

[17] Wikipedia, "UL (safety organization)," 3 May 2013. [Online]. Available: http://en.wikipedia.org/wiki/Underwriters_Laboratories#UL_Standards. [Accessed 8 May 2013].

[18] J. Blum, "Tutorial 15 for Arduino: GPS Tracking," 16 July 2012. [Online]. Available: http://www.jeremyblum.com/2012/07/16/tutorial-15-for-arduino-gps-tracking/. [Accessed May 2013].

Appendix A: Updated Timeline

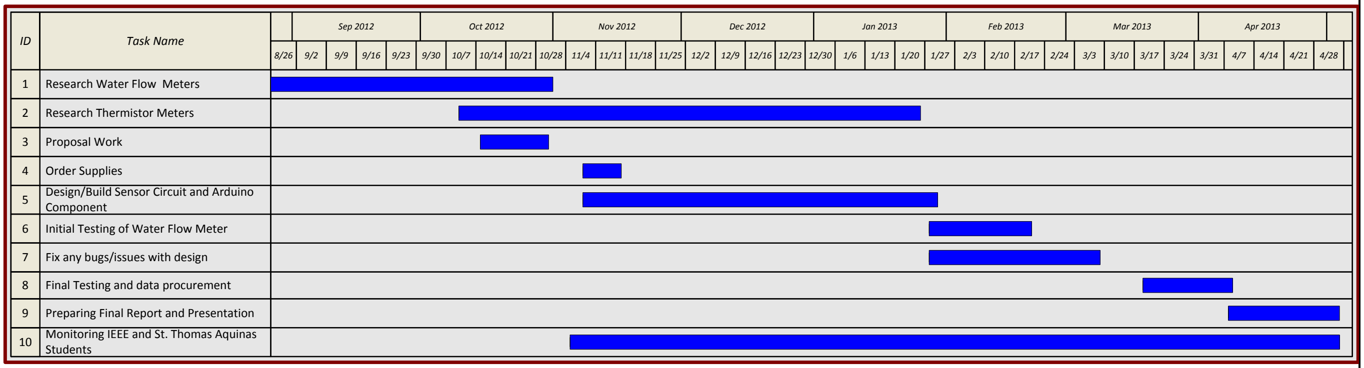

Figure A1: Original Estimated Project Timeline

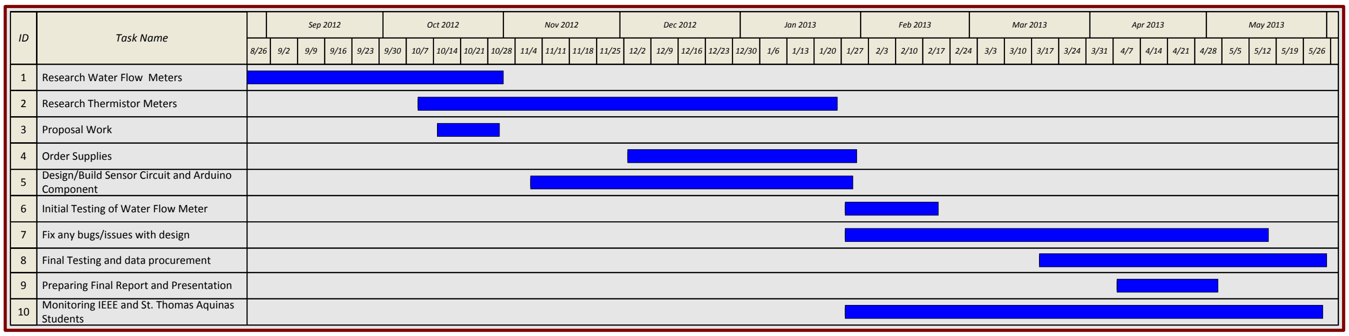

Figure A2: Updated Timeline

Appendix B: Budget Breakdown

Item	Description of Need	Estimated Cost($)	Actual Cost To Date($)
Arduino UNO Board	Microcontroller that organizes and stores all of the necessary information	25.00	21.95
SD Card Logger and SD Card	Storage of information taken from flow meter and GPS	30.00	37.26
Water Flow Meter	Water Flow Meter	1000.00	1007.50
Sensor Circuitry	Build of sensor circuitry and components	50.00	0.00
Batteries	Rechargeable Battery power necessary fro keeping devices running out on the water and battery charger	50.00	69.23
Floats	Need to keep electrronics afloat	150.00	52.33
Anchors and Chains/Rope	Need to anchor bouy and hold everything together	130.00	55.83
GPS/Antennas	Will give accurate timing and position information	60.00	74.69
Solar Panels(sets)	help keep the batteries charged for long term use	200.00	0.00
Wireless Networking	for setting the device in the water and being able to extract data without removing device	100.00	0.00
Other Expenses	Miscellaneous materials, waterproof housing transportation to testing sites and boat rentals	200.00	68.33
TOTAL COST PER DEVICE		1995.00	1387.11
TOTAL NUMBER OF DEVICES		4	2
TOTAL PROJECT COST		7980.00	2774.21
Source of Money		Total Funds Received	Funds Remaining
IEEE EPICS		100	0
UNH ECE Department		3990	1315.79

Appendix C: Meter Conversion Tables

STANDARD RATING TABLE No. 2 FOR AA CURRENT METERS (6/99)
EQUATION: V = 2.2048R + 0.0178 (R = revolutions per second)

Seconds	VELOCITY IN FEET PER SECOND Revolutions									Seconds
	3	5	7	10	15	20	25	30	40	
40	0.183	0.293	0.404	0.569	0.845	1.12	1.40	1.67	2.22	40
41	0.179	0.287	0.394	0.556	0.824	1.09	1.36	1.63	2.17	41
42	0.175	0.280	0.385	0.543	0.805	1.07	1.33	1.59	2.12	42
43	0.172	0.274	0.377	0.531	0.787	1.04	1.30	1.56	2.07	43
44	0.168	0.268	0.369	0.519	0.769	1.02	1.27	1.52	2.02	44
45	0.165	0.263	0.361	0.508	0.753	0.998	1.24	1.49	1.98	45
46	0.162	0.257	0.353	0.497	0.737	0.976	1.22	1.46	1.94	46
47	0.159	0.252	0.346	0.487	0.721	0.956	1.19	1.43	1.89	47
48	0.156	0.247	0.339	0.477	0.707	0.936	1.17	1.40	1.86	48
49	0.153	0.243	0.333	0.468	0.693	0.918	1.14	1.37	1.82	49
50	0.150	0.238	0.326	0.459	0.679	0.900	1.12	1.34	1.78	50
51	0.147	0.234	0.320	0.450	0.666	0.882	1.10	1.31	1.75	51
52	0.145	0.230	0.315	0.442	0.654	0.866	1.08	1.29	1.71	52
53	0.143	0.226	0.309	0.434	0.642	0.850	1.06	1.27	1.68	53
54	0.140	0.222	0.304	0.426	0.630	0.834	1.04	1.24	1.65	54
55	0.138	0.218	0.298	0.419	0.619	0.820	1.02	1.22	1.62	55
56	0.136	0.215	0.293	0.412	0.608	0.805	1.00	1.20	1.59	56
57	0.134	0.211	0.289	0.405	0.598	0.791	0.985	1.18	1.57	57
58	0.132	0.208	0.284	0.398	0.588	0.778	0.968	1.16	1.54	58
59	0.130	0.205	0.279	0.391	0.578	0.765	0.952	1.14	1.51	59
60	0.128	0.202	0.275	0.385	0.569	0.753	0.936	1.12	1.49	60
61	0.126	0.199	0.271	0.379	0.560	0.741	0.921	1.10	1.46	61
62	0.124	0.196	0.267	0.373	0.551	0.729	0.907	1.08	1.44	62
63	0.123	0.193	0.263	0.368	0.543	0.718	0.893	1.07	1.42	63
64	0.121	0.190	0.259	0.362	0.535	0.707	0.879	1.05	1.40	64
65	0.120	0.187	0.255	0.357	0.527	0.696	0.866	1.04	1.37	65
66	0.118	0.185	0.252	0.352	0.519	0.686	0.853	1.02	1.35	66
67	0.117	0.182	0.248	0.347	0.511	0.676	0.840	1.01	1.33	67
68	0.115	0.180	0.245	0.342	0.504	0.666	0.828	0.991	1.31	68
69	0.114	0.178	0.241	0.337	0.497	0.657	0.817	0.976	1.30	69
70	0.112	0.175	0.238	0.333	0.490	0.648	0.805	0.963	1.28	70
	3	5	7	10	15	20	25	30	40	

STANDARD RATING TABLE No. 2 FOR AA CURRENT METERS (6/99)
EQUATION: V = 2.2048R + 0.0178 (R = revolutions per second)

Seconds	VELOCITY IN FEET PER SECOND Revolutions									Seconds
	50	60	80	100	150	200	250	300	350	
40	2.77	3.33	4.43	5.53	8.29	11.04	13.80	16.55	19.31	40
41	2.71	3.24	4.32	5.40	8.08	10.77	13.46	16.15	18.84	41
42	2.64	3.17	4.22	5.27	7.89	10.52	13.14	15.77	18.39	42
43	2.58	3.09	4.12	5.15	7.71	10.27	12.84	15.40	17.96	43
44	2.52	3.02	4.03	5.03	7.53	10.04	12.55	15.05	17.56	44
45	2.47	2.96	3.94	4.92	7.37	9.82	12.27	14.72	17.17	45
46	2.41	2.89	3.85	4.81	7.21	9.60	12.00	14.40	16.79	46
47	2.36	2.83	3.77	4.71	7.05	9.40	11.75	14.09	16.44	47
48	2.31	2.77	3.69	4.61	6.91	9.20	11.50	13.80	16.09	48
49	2.27	2.72	3.62	4.52	6.77	9.02	11.27	13.52	15.77	49
50	2.22	2.66	3.55	4.43	6.63	8.84	11.04	13.25	15.45	50
51	2.18	2.61	3.48	4.34	6.50	8.66	10.83	12.99	15.15	51
52	2.14	2.56	3.41	4.26	6.38	8.50	10.62	12.74	14.86	52
53	2.10	2.51	3.35	4.18	6.26	8.34	10.42	12.50	14.58	53
54	2.06	2.47	3.28	4.10	6.14	8.18	10.23	12.27	14.31	54
55	2.02	2.42	3.22	4.03	6.03	8.04	10.04	12.04	14.05	55
56	1.99	2.38	3.17	3.95	5.92	7.89	9.86	11.83	13.80	56
57	1.95	2.34	3.11	3.89	5.82	7.75	9.69	11.62	13.56	57
58	1.92	2.30	3.06	3.82	5.72	7.62	9.52	11.42	13.32	58
59	1.89	2.26	3.01	3.75	5.62	7.49	9.36	11.23	13.10	59
60	1.86	2.22	2.96	3.69	5.53	7.37	9.20	11.04	12.88	60
61	1.83	2.19	2.91	3.63	5.44	7.25	9.05	10.86	12.67	61
62	1.80	2.15	2.86	3.57	5.35	7.13	8.91	10.69	12.46	62
63	1.77	2.12	2.82	3.52	5.27	7.02	8.77	10.52	12.27	63
64	1.74	2.08	2.77	3.46	5.19	6.91	8.63	10.35	12.08	64
65	1.71	2.05	2.73	3.41	5.11	6.80	8.50	10.19	11.89	65
66	1.69	2.02	2.69	3.36	5.03	6.70	8.37	10.04	11.71	66
67	1.66	1.99	2.65	3.31	4.95	6.60	8.24	9.89	11.54	67
68	1.64	1.96	2.61	3.26	4.88	6.50	8.12	9.74	11.37	68
69	1.62	1.94	2.57	3.21	4.81	6.41	8.01	9.60	11.20	69
70	1.59	1.91	2.54	3.17	4.74	6.32	7.89	9.47	11.04	70
	50	60	80	100	150	200	250	300	350	

Appendix D: Battery Capacity Graphs

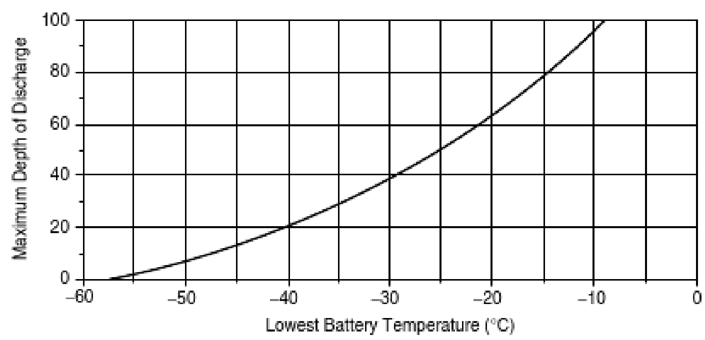

Figure D1: Maximum Depth of Discharge allowed accounting for possible freezing of battery[5]

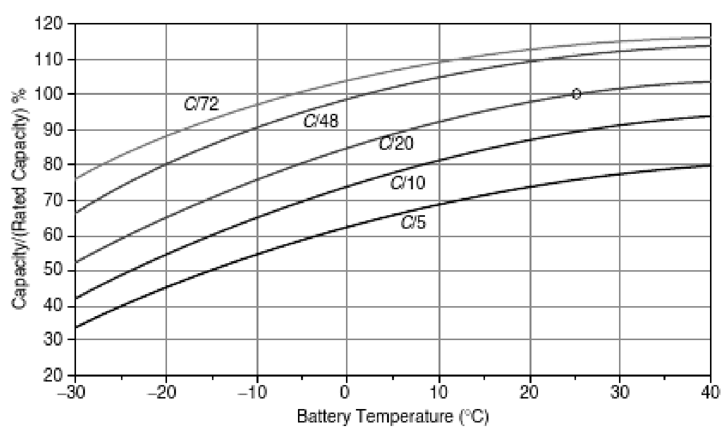

Figure D2: Capacity temperature adjustment for various discharge rate[5]

Appendix E: Arduino Code

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>
#include <SD.h>
#include <stdlib.h>

/*
   -Uses PMB-688 GPS device hooked up on pins 3(Rx)YELLOW and 4(Tx)BLUE
     -Code adapted from the TinyGPS example: test_with_gps_device
        http://arduiniana.org/libraries/tinygps/
     -Saving to an SD card follows examples from Jeremy Blum's Arduino Tutorial Series
        -Episode 15 - GPS Tracking: http://www.jeremyblum.com/2012/07/16/tutorial-15-for-arduino-gps-
tracking/
   -Flow Sensor Connected to pin 2
     -Price Meter attached using equation relating revolutions per second to velocity (ft/s):
          Velocity = 2.2048*(Revoltions) + 0.0178
          With 5 Second counts, this means an acceptably range of 0.46 feet per second to the rated 25
feet per second
   -SD Shield from adafruit.com
     -CS on pin 10
   -Notification LEDs on Pins 8 and 7 uses LEDs onboard the SD Shield (L2 = Red LED L1 = Green LED)
     -Red LED on, Green off means there was an SD Error
     -Both on, means the GPS didn't get a lock, reading and saving still being done
     -Green On, Red Off means functioning normally
     -Both off, means delay until next reading

*/
TinyGPS gps;
SoftwareSerial nss(3, 4);
static char dtostrfbuffer[20];
int CS = 10;
int LED_G = 8;
int LED_R = 7;
//Flowmeter Variables
unsigned int flowcount = 0;      //Revolutions counter
float revs = 0.0;                //revolutions per second
float velocity = 0.0;            //Calculated velocity in feet per second
unsigned long startflow = 0;     //Loop counter
unsigned int flow = HIGH;        //Sets the flow reader to HIGH to avoid incorrect readings
int duration = 12000;            //Run counter for 12 seconds
int duration_seconds = 0;        //Convert from ms to seconds

//Define String
String SD_date_time = "invalid";
String SD_lat = "invalid";
String SD_lon = "invalid";
String dataString ="";
```

```
static void gpsdump(TinyGPS &gps);
static bool feedgps();
static void print_float(float val, float invalid, int len, int prec, int SD_val);
static void print_date(TinyGPS &gps);
/***********************************************************************************
*********************************************************************/
//Error Printing/Stopping the program to conserve energy
void error(char *str)
{
 Serial.print("error: ");
 Serial.println(str);

 // red LED indicates error
 digitalWrite(LED_R, HIGH);

 while(1); //halts the program
}
/***********************************************************************************
*********************************************************************/
void setup()
{
 pinMode(CS, OUTPUT);  //Chip Select Pin for the SD Card
 pinMode(LED_G, OUTPUT);
 digitalWrite(LED_G, LOW);
 pinMode(LED_R, OUTPUT);
 digitalWrite(LED_R, LOW);
 //Prepare Counter and delay timers
 duration_seconds = (duration / 1000);
 //Serial interfaces
 Serial.begin(115200);
 nss.begin(4800);

 //Connect to the SD Card
 if(!SD.begin(CS))
 {
  error("Card Failure");
 }
}
/***********************************************************************************
*********************************************************************/
void loop()
{
 for(int f = 0; f < 6; f++)
 {
  bool newdata = false;
  unsigned long start = millis();
```

```
// Every second we print an update
while (millis() - start < 2000)
{
   if (feedgps())
   newdata = true;
}
gpsdump(gps);

//Write the newest information to the SD Card
dataString = SD_date_time + "," + SD_lat + "," + SD_lon;
if(SD_date_time != "invalid")
{
  digitalWrite(LED_R, LOW);
  digitalWrite(LED_G, HIGH);
}
else
{
  digitalWrite(LED_G, HIGH);
  digitalWrite(LED_R, HIGH);
}

//Open the Data CSV File
File dataFile = SD.open("LOG.csv", FILE_WRITE);
if (dataFile)
{
  dataFile.print(dataString);
  dataFile.print(",");
  dataFile.close();
}
//Read the meter
startflow = millis();
while (millis() - startflow < duration)
{
  // read the input on analog pin 0:
   int sensorValueNew = analogRead(A0);
   // Software Debouncing
   if (sensorValueNew >= 200)
   {
    delay(50);
    sensorValueNew = analogRead(A0);
    if(sensorValueNew <= 200)
    {
       flowcount++;
    }
   }
}
if(flowcount == 0)
```

```
  {
   revs = 0;
   velocity = 0;
  }
  else
  {
   revs = (float)flowcount / (float)duration_seconds;
   velocity = 2.2048 * (revs) + 0.0178;
  }
  //Save Velocity to the SD Card
  dataFile = SD.open("LOG.csv", FILE_WRITE);
  if (dataFile)
  {
   dataFile.println(velocity);
   dataFile.close();
  }
  flowcount = 0;
 }
 digitalWrite(LED_R, LOW);
 digitalWrite(LED_G, LOW);
 delay(300000);
}
/************************************************************************************
********************************************************************/
static void gpsdump(TinyGPS &gps)
{
 float flat, flon;
 unsigned long age, date, time, chars = 0;
 unsigned short sentences = 0, failed = 0;
 static const float LONDON_LAT = 51.508131, LONDON_LON = -0.128002;

 gps.f_get_position(&flat, &flon, &age);
 print_float(flat, TinyGPS::GPS_INVALID_F_ANGLE, 9, 5, 1); //LATITUDE
 print_float(flon, TinyGPS::GPS_INVALID_F_ANGLE, 10, 5, 2); //LONGITUDE

 print_date(gps); //DATE AND TIME
}
/************************************************************************************
********************************************************************/
static void print_float(float val, float invalid, int len, int prec, int SD_val)
{
 char sz[32];
 if (val == invalid)
 {
  strcpy(sz, "*******");
  sz[len] = 0;
    if (len > 0)
     sz[len-1] = ' ';
```

```
  for (int i=7; i<len; ++i)
    sz[i] = ' ';
  if(SD_val == 1) SD_lat = sz;
  else if(SD_val == 2) SD_lon = sz;
 }
 else
 {
  if (SD_val == 1) SD_lat = dtostrf(val,10,5,dtostrfbuffer);
  else if (SD_val == 2) SD_lon = dtostrf(val,10,5,dtostrfbuffer);
 }
 feedgps();
}
/*******************************************************************************************
*********************************************************************/
static void print_date(TinyGPS &gps)
{
 int year;
 byte month, day, hour, minute, second, hundredths;
 unsigned long age;
 gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);
 if (age == TinyGPS::GPS_INVALID_AGE)
 {
  SD_date_time = "invalid";
 }
 else
 {
  char sz[32];
  sprintf(sz, "%02d/%02d/%02d %02d:%02d:%02d   ",
    month, day, year, hour, minute, second);
  SD_date_time = sz;
 }
 feedgps();
}
/*******************************************************************************************
*********************************************************************/
static bool feedgps()
{
 while (nss.available())
 {
  if (gps.encode(nss.read()))
   return true;
 }
 return false;
}
/*******************************************************************************************
*********************************************************************/
```