Identifying and Monitoring Water Quality Needs for Puget Sound's Aquaculture Industry and Environmental Security Through the Design and Use of a Prototype Sensor Suite

Emily Hammermeister
University of Washington, Seattle, WA
School of Oceanography, Box 357940
Ocean Technology Program
Email: emhammer@uw.edu
May 19, 2019

Contextual Setting:

The impact of climate change on the environmental and economic security of marine resources has now entered the public and political discourse. The sectors of the marine industry that must make both short and long-term operational decisions under conditions of uncertainty regarding both the magnitude of ecological change and the response in governing policies, seek sources of trusted information which can guide their investments. The shellfish industry is a significant sector of the seafood industry, especially in the Puget Sound region of Washington State. This research identifies the water quality monitor needs of shellfish farmers and designs a prototype sensor suite at a favorable price point to provide both operational and long-term data for investments in light of uncertain localized climate change impacts.

Introduction

Background

As the nation's leading producer of farmed bivalves, (including clams, oysters, and mussels) Washington State's shellfish industry contributes significant economic value to the state. Altogether, the seafood industry in Washington state generates over 42,000 local jobs and contributes at least $1.7 billion to gross state product through profits and employment at restaurants, distributors, and retailers (U.S. Department of Commerce, National Atmospheric and Oceanographic Administration, 2011). In addition to their statewide economic importance, shellfish are a fundamental to tribal cultures and economies, as well as contributing to recreational

opportunities and tourism (Washington Sea Grant, 2015). Furthermore, shellfish are an integral component of marine ecosystems, providing habitat, increasing biodiversity, and helping to filter water.

As shellfish feed, they increase the clarity of the water, which allows more sunlight to penetrate deeper in the water column, promoting the healthy growth of habitat-critical organisms such as seagrasses. Seagrasses help to stabilize the seafloor from erosion, provide a food source, and maintain better water quality. When shellfish are harvested, their nutrient uptake is also removed from the marine environment, so they act as a nutrient sink, helping to compensate for potentially damaging excess nutrient runoff in coastal areas that cause harmful algae blooms and hypoxic zones (WSI, 2018).

Shellfish in the Northwest are at risk to the changing climate, specifically, the acidification of ocean water. Since the Industrial Revolution, atmospheric CO_2 levels have proliferated. A third of this CO_2 is absorbed by the ocean where it reacts with H_2O and protonates the water (Doney, 2008). As seen by the reaction: $CO_{2\,(aq)} + H_2O \leftrightarrow H_2\,CO_3 \leftrightarrow HCO_3^- + H^+$, there is an excess of hydrogen atoms yielding from anthropogenic CO_2, which then results in ocean acidification over time (Figure 1).

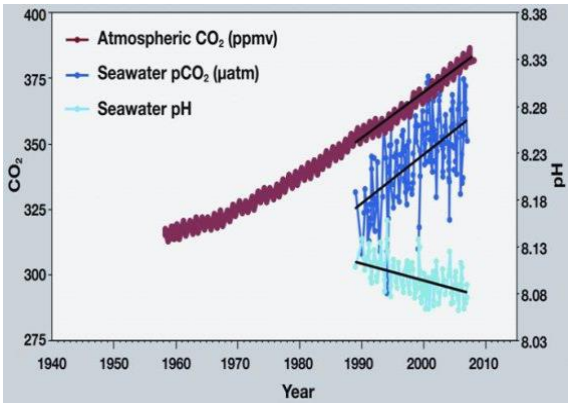


Figure 1. *As carbon dioxide rises in the atmosphere, some of it dissolves into seawater, increasing the CO2 concentration there as well. As CO2 dissolves in the ocean, it reacts with water to form hydrogen ions, which drive the pH down. (Picture rights to NOAA PMEL Carbon Program)*

In most normal ocean water conditions, formation of $CaCO_{3(S)}$ calcification), is thermodynamically favored by the abundance of the reactants: dissolved calcium ($[Ca^{2+}]$), and

carbonate ($[CO_3^{2-}]$) ions. However, as atmospheric CO_2 rises, ocean carbonate availability decreases, causing the energetic favorability of $CaCO_{3(s)}$ to lower. The carbonate is being consumed in the dissolved CO_2 reaction with water, and thus impedes calcification (Barton, A. et al., 2012).

Washington's marine waters are especially vulnerable to ocean acidification due to the natural physical processes of ocean circulation that intensify the acidifying effects of global atmospheric carbon dioxide. Coastal upwelling is one of the most important regional factors, because it brings offshore water from the deep ocean onto the continental shelf, and into the Puget Sound. This water is naturally rich in dissolved carbon dioxide with a low pH, and an abundance of nutrients (Washington State Blue Ribbon Panel on Ocean Acidification, 2012, Figure 2). In addition, nitrogen loading and other organic-matter inputs from local sources as well as upwelled waters fuel algae growth. In turn, this decreases oxygen levels and increases CO_2 levels in the water (WSDOE). Combined with the right set of other environmental conditions (e.g. light, temperature, circulation), the nutrient input is an initiating factor for a Harmful Algal Bloom (HAB), a common issue for fisheries in the Puget Sound (Northwest Fisheries Science Center). Due to their dependency on calcium carbonate to generate shells, skeletons, and other solid body parts, over 30 percent of Puget Sound's marine species are vulnerable to ocean acidification. This includes, but

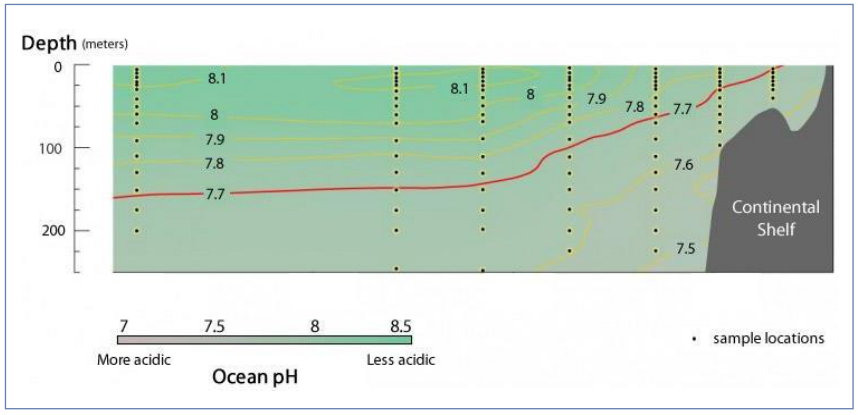


***Figure 2**. The pH of the water sampled at different depths along transect line 5 near the border of Oregon and California. The water upwelling onto the continental shelf recorded lower pH levels than water samples collected farther away from the shelf. The water below the red line was under saturated with respect to aragonite, a common type of calcium carbonate used in shell-building (Kennedy, 2009)*

is not limited to clams, oysters, mussels, and geoducks; all of which are central to Washington state's shellfish industry (Washington State Blue Ribbon Panel on Ocean Acidification, 2012).

Environmental Security

Environmental security comprises all the elements of the environment that provide for human survival and prosperity. Therefore, the relationship between the environment and the security of humans defines the importance of environmental security. The environment provides food, shelter, energy, transportation, and more for humans. If the environment's health is compromised, then so is the security of human quality of life. The increasingly complex and transnational drivers of environmental challenges can ultimately foster destabilization and geopolitical tension (Stimson). Over a billion people rely on the ocean for food (Stimson), and with the climate reaching unprecedented dangerous territory, life below water is being drastically altered. Changing marine ecosystems place increasing strain on aquaculture industries to output product. At this pace, the problems facing our environment threaten the economic and food security of communities and place a disproportionate burden on the overall security of coastal nations. This is, the issue of environmental security; and, therefore, the platform for which this paper hopes to facilitate public conversation through the use of data collection and marine industry involvement.

Objective

This project aims to prototype a low-cost **marine water monitoring instrument package** to be deployed at aquaculture facilities, and provide site-specific shareable information to assist in

both short and long-term operational decision making. The name of this sensor suite prototype is ***ARM*** *(Aquaculture Resource Monitoring):* Able to *arm* shellfish farmers with the tools they need for more effective and successful practices. The goal is to establish a long-term data chain between facilities with multiple site-deployed sensors. This provides an equal opportunity for small and large aquaculture facilities to get the useful location-based monitoring they need. Future operating decisions, as well as potential corrective action (i.e., mitigation, restoration) can be determined on a site-by-site basis, as needed. Therefore, a more detailed and accurate record of the Puget Sound and Washington coast water quality conditions can be measured, monitored, and modeled. Finally, a conversation between scientific communities and non-scientific stakeholders must be facilitated. The marine environment provides for food, economic benefits, transportation means, cultural tradition, and recreation. It's become a matter of Environmental Security: "One of the most important things we can do to meet our national security objectives and advance political stability, human health, economic development and peace around the world is to recognize — and act in ways reflecting this — that a healthy planet is a critical part of the policy equation." (Rodewald, A. D., 2018). The water monitoring sensor, ARM, will provide integral environmental data for use as an outreach tool that businesses, agencies, and organizations can use to acquire attention, and influence future policy making.

Methods

Needs Analysis

To provide for the most effective design of the sensor, a needs analysis was conducted with local shellfish farmers. Three separate shellfish growers based out of three separate basins of the Puget Sound were interviewed. The establishments and locations are as follows:

Rock Point Oyster Company: Quilcene, WA (Hood Canal Basin)

Penn Cove Shellfish: Coupeville, WA (Whidbey Basin)

Chelsea Farms: Olympia, WA and Shelton, WA (South Sound Basin)

The consensus among all three establishments were that such a sensor would be of great value and use to them. The current monitoring efforts, including the NANOOS Buoys, are not sufficient for their site-specific needs. Problems stem from frequent failure resulting in data blackouts for extended periods of time. The buoys are often in the middle of the sound, thus hard to reach and gather data when using physical pin plug connections. Furthermore, most monitored sites are not close enough to the embayments where the farms are located, meaning water quality and measurements could vary widely between bays and monitoring stations. In turn, this gives establishments less accurate data to base their operating decisions off *(Figure 3)*.

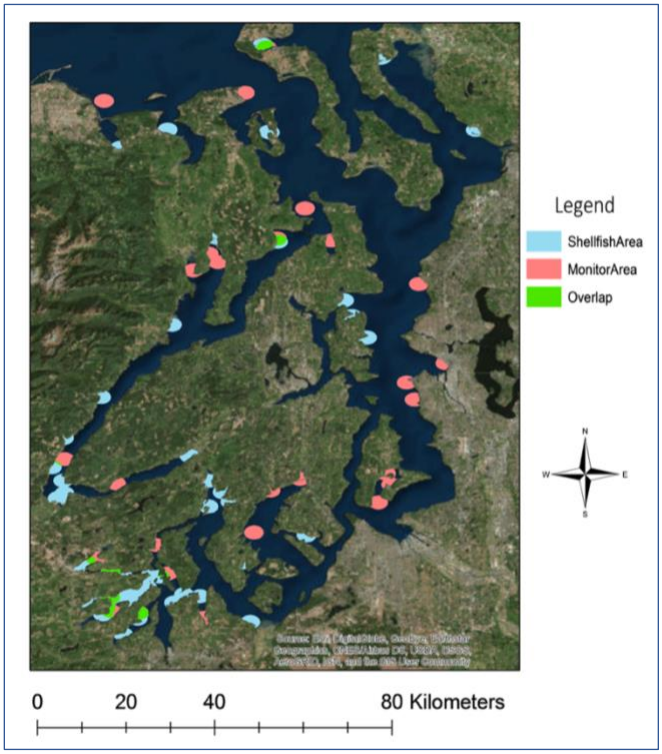


***Figure 3**. A map of the Puget Sound showing where shellfish growers are and where monitoring efforts are. A buffer of 1 mile was given to each point to show ranges of acceptable data that would show accuracy in that range. Shellfish farmers do not have a lot of access to monitoring buoys as represented by this map and the lack of overlap (green).*

After thorough discussion with the interviewed farms about the most beneficial components of water chemistry to measure, the sensor parameters were finalized (*Figure 4).* While turbidity and lux had the same amount of interest, because of logistics, this project ultimatly decided to go with lux. Lux can also be used as proxy for turbidity, since less light will attenuate in more turbid waters.

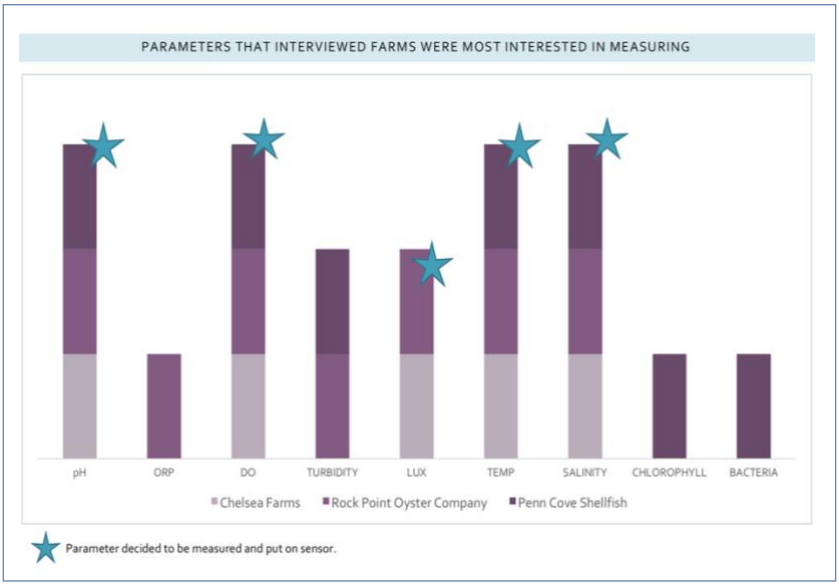


Figure 4. *Shown interest in specific water quality parameters by each interviewed shellfish farming*

The specific parameters for this sensor are as follows:

- pH
- Dissolved Oxygen (DO)
- Salinity (Conductivity)
- Lux (Light Proxy)
- Temperature

Design and Build

The housing for the underwater sensor is the 3" series clear acrylic 8.75" long tube from Blue Robotics. The tube is designed to be deployed vertically with one endcap towards the seafloor and the other towards the surface. The surface-facing clear acyclic endcap is 3", with an O-ring flange and one penetrating hole for a single cable to communicate to topside. The seafloor-facing

endcap is a 3” clear acrylic endcap with an O-ring flange and five penetrating holes for sensor exposure and measurement. The depth rating of this housing is 150 meters *(Figure 5)*.

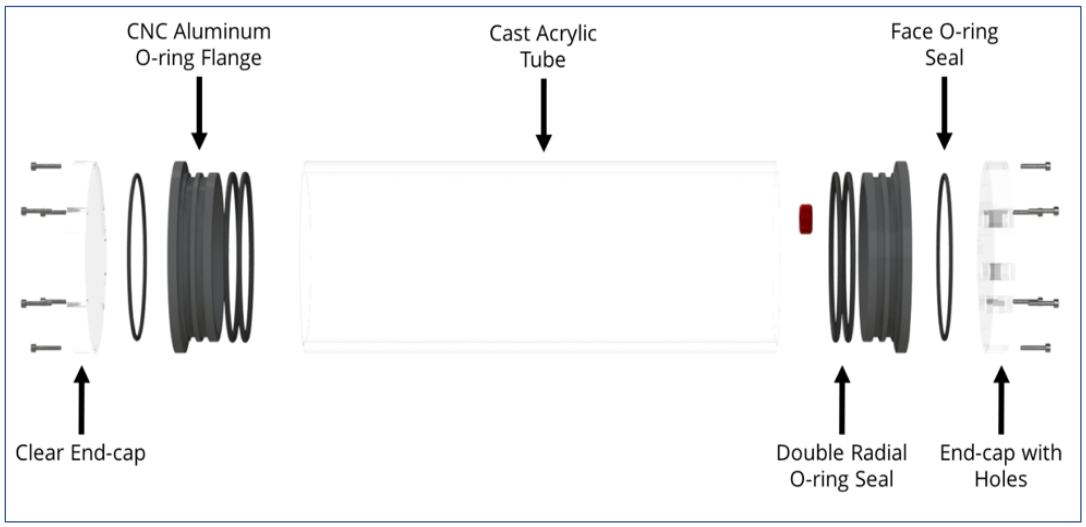


***Figure 5**. Blue Robotics 3” Series acrylic tube*

The dissolved oxygen, pH, conductivity, and temperature probes (*Appendix A)* are exposed externally from the sensor, with their cables penetrating the housing downward-facing endcap using Blue Robotics M10 Penetrator for 6mm cable *(Figure 6)*. Each penetrator is potted on both sides with epoxy and includes an o-ring to ensure no water passage.

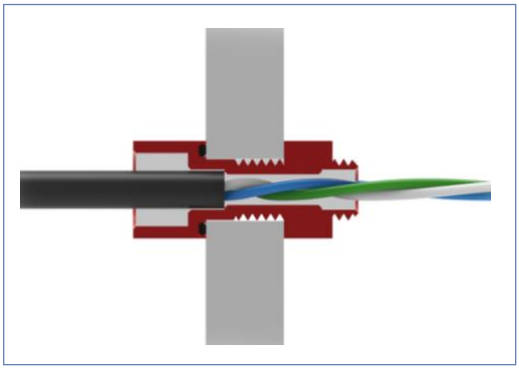


***Figure 6.** Blue Robotics M10 Penetrator for 6mm Cable*

The underwater enclosure houses each of the probe cables and respective breakout boards. The

dissolved oxygen, pH, and conductivity probes use BNC connectors that are connected within the underwater housing. The lux sensor is not exposed to the water and is secured to the topside-facing acrylic endcap to measure sunlight attentuation. Each probe's wires are connected to two busbars. One busbar is for power and the other is for data (*Figure 7).* The busbars connect the sensors to a 20/8 AWG 13ft shielded marine cable that penetrates out of the underwater housing and goes to a topside a junction box. The dockside weatherproof junction box will have a single penetrating hole with a cable gland for the cable coming from the underwater sensor manifold to connect to power and deliver data topside. The junction box houses the Arduino Mega 2560 micropocessor (*Appendix A)* that each of the wires coming from the underwater housing will connect to. Also connected to the Mega 2560 inside the junction box is a Real Time Clock (RTC) and an SD card reader (*Appendix A)*. The Mega 2560 is powered by two 5V recharchable power banks connected in series and charged by a SparkFun Sunny Buddy Solar Charger V13 and 6W Solar Panel (*Appendix A)*

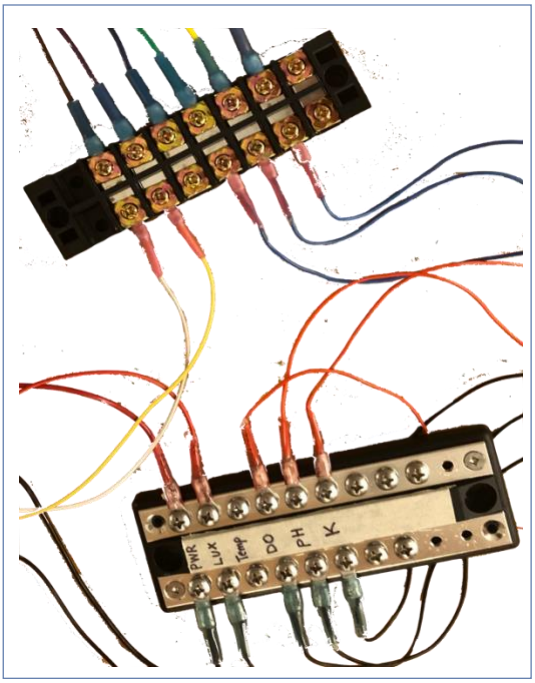


Figure 7. *Bus Bar Schematic. Data busbar (Top) Separated by parameter. Power busbar (Bottom) distributing power to all breakout boards from one cord coming from topside*

Software

The process of writing the software for the ARM Sensor involved writing each components code separately, then combining them afterwards. While each of the parameter sensors (pH, DO, Salinity, Lux, and Temp) have sample codes available, their code for ARM has been strongly modified to best adhere to its functionalities. The most involved component of the ARM sensor to

code for was the SD card module, because it required each other component to write to the SD card after each reading. Most individual sensors needed a function to run an actual reading, so then the function had to be embedded within the SD writing function code. Each of these packages were called void logging(). These logging functions all then had to be put into a loop so that the sensor would keep taking readings 3 minutes apart each. A complete copy of the code used to program the ARM sensor can be found in ***Appendix B***.

Deployment

For this research and testing of the ARM sensor prototype, it was deployed at Rock Point Oyster Company in Dabob Bay, Hood Canal, Washington State, from May 5th- May 11th, 2019. The ARM sensor was deployed in a holding tank on site, that continually pumped water directly from the bay. Since this is where Rock Point Oyster Company holds its juvenile (and most sensitive) oysters, it was the location they were most interested in knowing the water quality, and therfore the deployment site for the ARM sensor. The deployment method for the ARM sensor is to be dockside vertical suspension into the water, while the deck junction box stays topside. The submerged housing is positively buoyant with about 8 lbs of lift in water depending on the water's density. In order to keep the housing submerged, and not too negatively buoyant so not to put too much strain on the cable connecting topside, weights were suspended from the bottom of the housing. The weights were attached through a bridal line configuration secured to the housing by a hose clamp. Small rubber sheets were used as chafe protection between the hose clamp and the acryllic housing. With this setup, the ARM sensor was barely negatively bouyant and sat vertical in the water column.

Results

Build

Completed ARM sensor suite (*Figure 8,9,10)*. Cost of building totaling ~$500. Major components of sensor can be found in *Appendix A.*

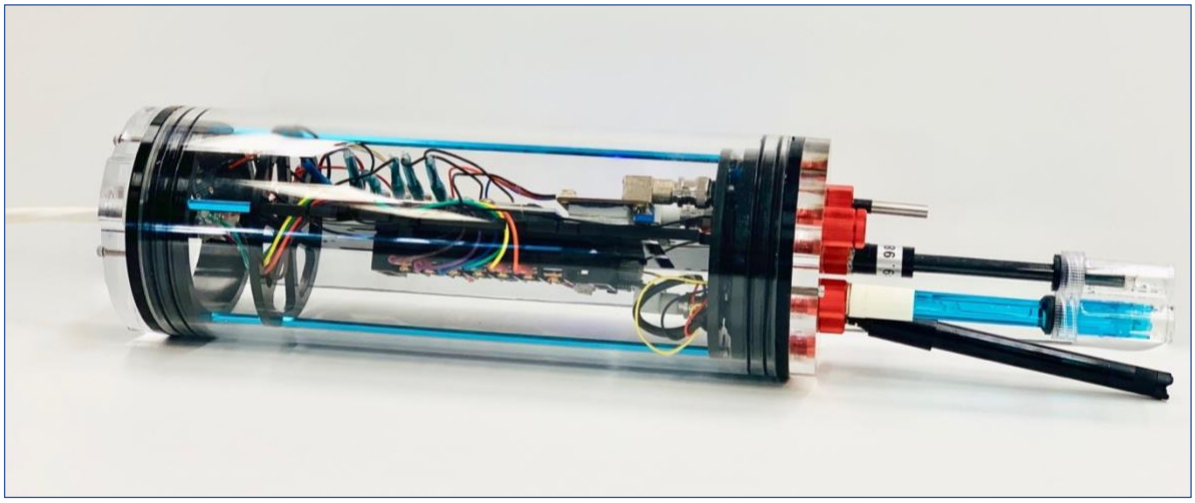


Figure 8. *Completed underwater portion of sensor. Electronics inside Blue Robotics Housing with protruding probes taking water quality measurements.*

Figure 9. *Inside wiring of junction box. Including Arduino Mega 2560, SD Card Module, the Real-Time Clock, and the power storage. The penetrating cable on the left has come from the underwater sensor and transfers data between the underwater sensor and the Arduino, as well as writing the data onto the SD card.*

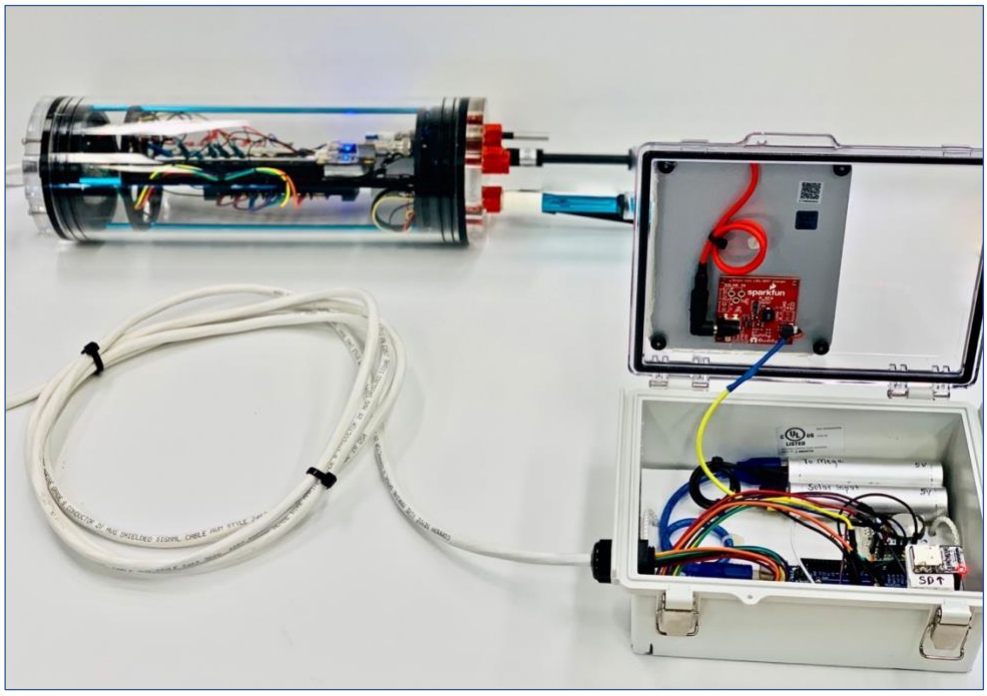


***Figure 10**. Completed ARM Sensor Suite with 13ft data cable connecting underwater housing with dockside junction box.*

Functionality

Completed ARM sensor suite was successfully deployed (*Figure 11)* for 6 days without water intrusion and without damage. However, the battery manifold lasted 14 hours, so there was no data collected after 14 hours into deployment. Each sensor collected data, however the Dissolved Oxygen Probe did not collect accurate data, and the luminosity sensor periodically shorted, invalidating certain data points. The biological growth on the housing after deployment for only 6 days was significant, as seen in comparison between *Figure 11* (before) and *Figure 12* (after).

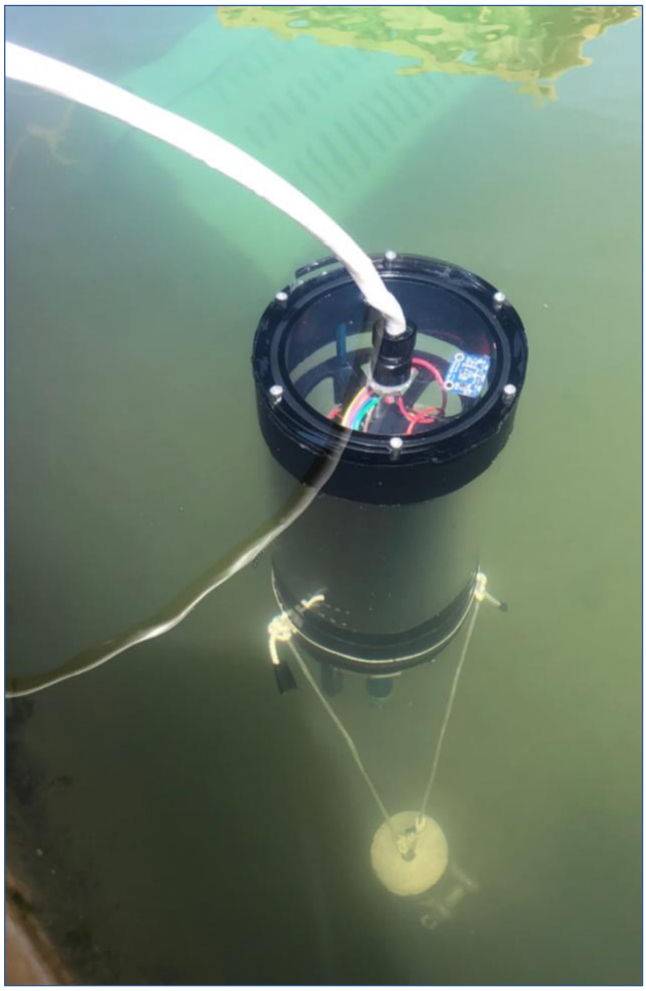


***Figure 11**. ARM Sensor deployment May 5th, 2019.*

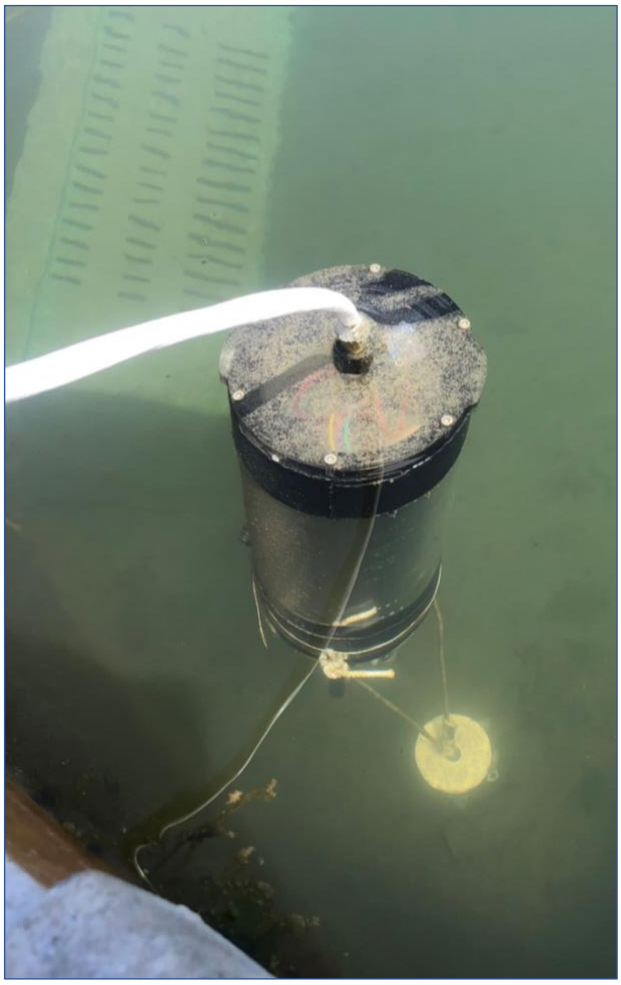


***Figure 12**. ARM sensor recovery on May 11th, 2019. Bio-fouling evident.*

Data

The 14 hours of data collected align with obvious trends such as daylight and temperature corresponding with time. The temperature dropped about 5°C during the night hours *(Figure 13)*. While this temperature change is significant and unlikely in open water, the sensor was deployed in a holding tank for oysters, so because of the limited volume, the water could heat and cool easier, therefore giving these results.

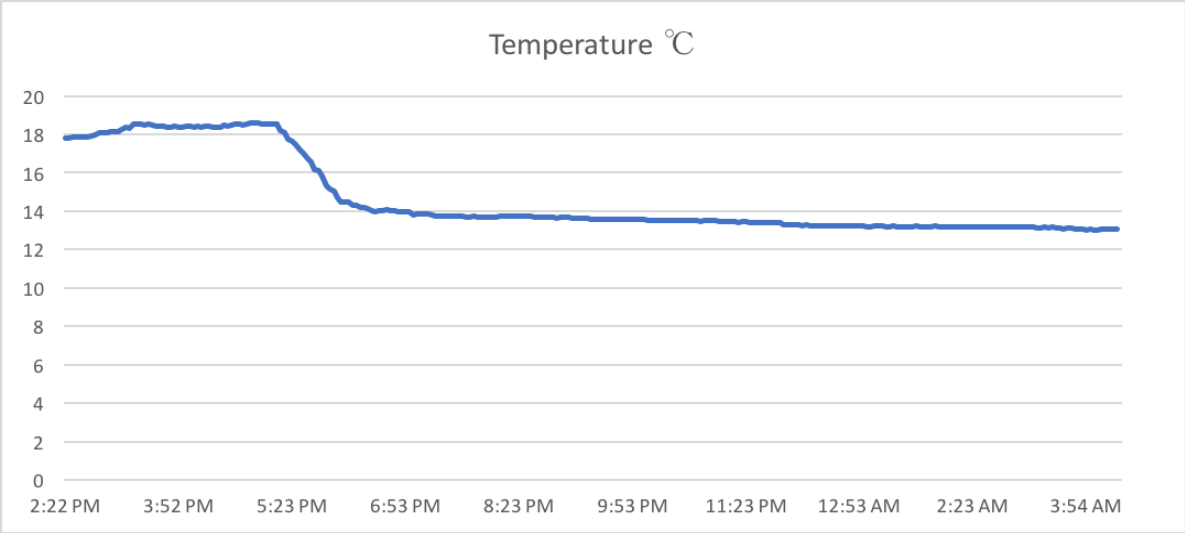


***Figure 13**. Temperature readings from the ARM sensor within the 14-hour lifespan of the sensor from May 5th to May 6th, 2019*

The luminosity sensor reads 65536 when it shorts. This happened several times within the 14 hours of data collection. Regardless of the spiking shorts, there is still an obvious trend that the sensor picked up sunset, and nighttime: measuring 0 lux during the night (*Figure 14).*

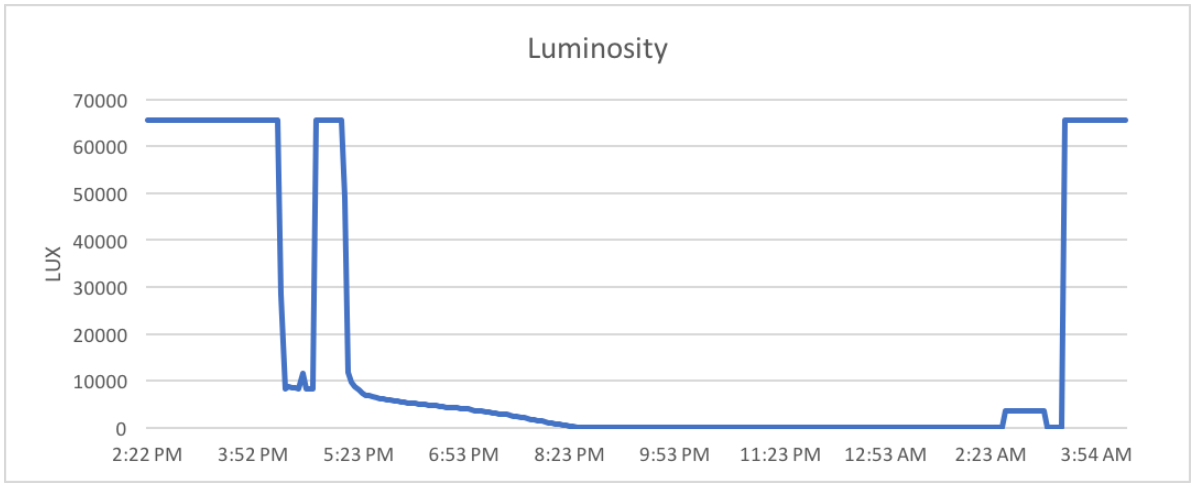


***Figure 14**. Luminosity (Lux) readings from the ARM sensor within the 14-hour deployment from May 5th to May 6th, 2019*

The pH probe took over an hour to acclimate and start reading accurate pH levels (represented by the climbing line during 3:00pm). As the night goes on the pH decreases to levels the farms should be concerned about (6.5-7.5) for their calcifying shellfish product *(Figure 15).*

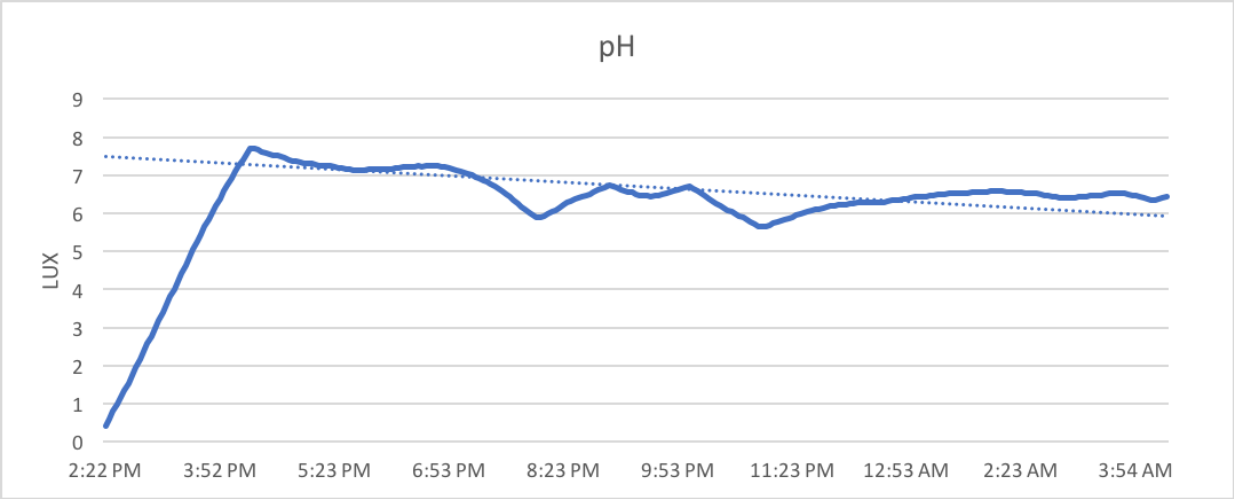


***Figure 15**. pH readings from the ARM sensor within the 14-hour deployment from May 5th to May 6th, 2019*

The salinity probe showed relatively jagged, yet consistent results. Mostly staying in the 40-60 ms/cm range, Dabob bay showed consistent conductivity with the normal seawater range of about 50ms/cm (*Figure 16).* It is also interesting to compare the salinity with the tides during that 14-hour range *(Figure 17),* to see that conductivity increased just after high tide at 6:53pm. This makes sense because the high tide brings in ocean water from the Pacific Ocean. Following the 6:53pm high tide, the conductivity stays relatively constant, considering the low tide wasn't very large, and there was no rainfall. This is an example as to why location-based data gathering is so important for these farms.

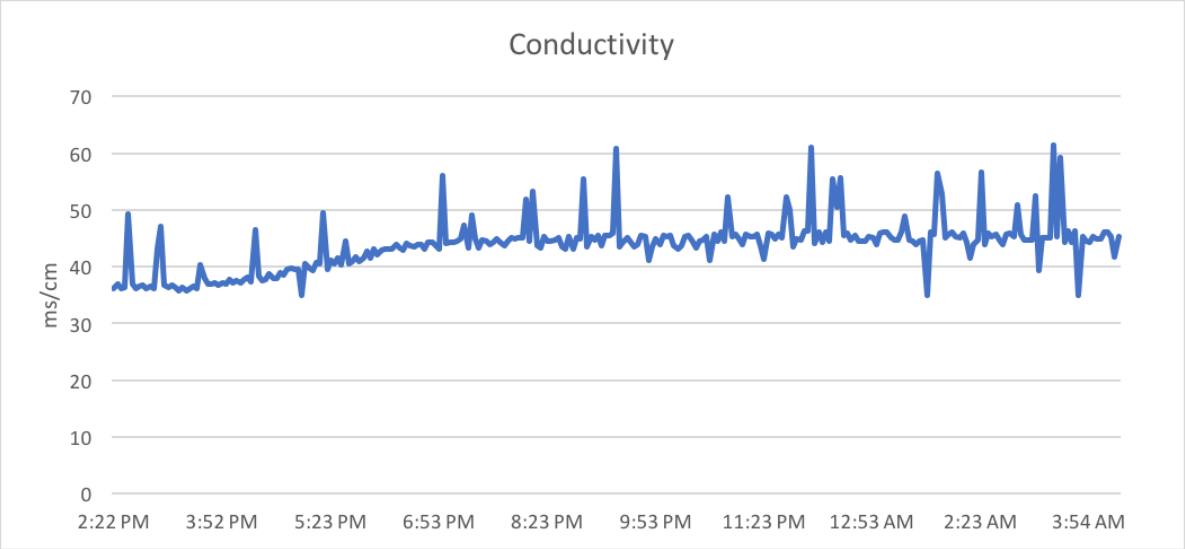


***Figure 16**. Conductivity, a proxy for salinity, readings from the ARM sensor within the 14-hour deployment from May 5th to May 6th, 2019*

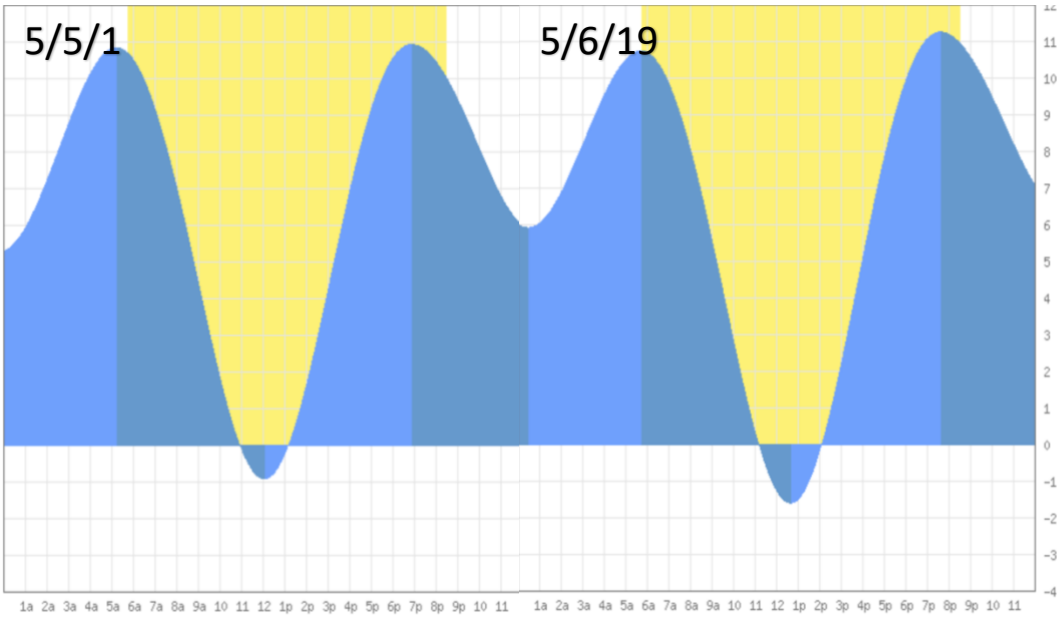


***Figure 17**. Dabob Bay Tide Chart for May 5th-6th 2019. May 5th High tide 6:53pm (10.91ft). May 6th Low tide 12:28am (5.90ft), high tide 5:41am (10.73ft)*

Unfortunately, the Dissolved Oxygen probe malfunctioned and did not get any viable readings during the ARM sensor prototype testing deployment.

Discussion

After the completion of the design, build, and deployment of the ARM sensor, and upon analysis of the results from doing so, there are some key takeaways. The ARM sensor, designed to "arm" aquaculture farms with the tools they need to effectively grow and monitor shellfish, has been ultimately deemed a successful prototype. The users at Rocky Point Oyster Company were enthusiastic about the product and expressed interest in future endeavors. They assisted in the launch and recovery of the sensor, and checked for leaks throughout the deployment week. The user approval and enthusiasm of the ARM sensor presents success to the main goal of this project: working with local growers to monitor and guide their daily operations. After being able to complete the needs analysis from local shellfish farms, and deliver a product of their liking, the ARM sensor can be regarded as a huge accomplishment in their favor. However, there are several technical components to ARM that can be improved for future implications:

Power & Battery System

The battery system implemented into the ARM sensor was designed for prototyping, so it wasn't designed to last an extended period. However, it still did not last as anticipated. The increased frequency of readings from every 30 minutes to every 3 minutes is likely the cause of the early battery depletion. Furthermore, the two 5V portable phone charging batteries was simply not enough power storage and the solar panel charging them was simply not large enough.

Furthermore, the Sunny Buddy microprocessor (*Appendix A)* that regulated the charging of the batteries from the solar panel was not powerful enough. It stepped the ~5V that the solar panel was receiving down to 3.3V which was not efficient enough to keep up with the batteries that powered the sensor. A larger Sunny Buddy microprocessor is needed

so that the voltage from the sun is not stepped down to 3.3V, but rather 5V, so to be able to charge the batteries proficiently.

The overall solution is to get a more powerful setup. The solar panel (*Appendix A)* must be larger so it has a greater voltage output. The Sunny Buddy microprocessor must be able to handle a larger voltage and current so that it can regulate more power and not step down to such a low voltage that wasn't enough to power the battery. Finally, the battery power storage needs to be larger. Ideally a 12V lithium battery would be enough to store power and receive enough input from a larger solar panel to keep the sensor running indefinitely.

Protruding Probes

While the probes that measure pH, salinity, dissolved oxygen, and temperature must be exposed to the elements of the water, the current design allowing each one to be fully protruding the housing may be flawed. The pH, salinity, and dissolved oxygen probes all were fully protruded, meaning only the cord was inside of the waterproof housing. Once epoxied, this allowed for any strain on the epoxy to be exaggerated due to the leverage provided from the entire protruded sensor tip. The epoxy on the pH probe cracked twice because of this and had to be repaired. This problem can be avoided one of two ways:

1. Only allow a small part of the probe to be exposed outside of the housing. This keeps leveraging at a minimum and provides for a much steadier epoxy hold. However, this solution requires larger penetrating holes into the housing, making water intrusion a higher possibility, as well as making it more difficult to find the correct size of effective penetrators.

2. Create a protective cage to cover the sensors so that they are still able to be emerged in water, but unable to be bent and break the epoxy seal. This is a good solution because it provides protection for the probes themselves as well as the epoxy potting. During deployment, and maintenance, the probes are at risk of getting damaged since they are so exposed, so a protective cover is the best solution overall.

Dissolved Oxygen Probe

The dissolved oxygen (DO) probe protruding from the sensor was defective during deployment. This most likely occurred because of the other sensor's code interfering with the DO code. During the bench test, the DO recorded correct readings of a known substance and calibrated correctly, but once the software was joined with other sensors, it did not give accurate readings. The solution to this problem is to develop a more advanced code for the probe or to use a more advanced DO probe. While this will slightly increase the cost of the sensor, it is a necessary solution, as dissolved oxygen is a vital indicator of the environment for shellfish.

Luminosity (lux) Sensor

The lux sensor residing on the inside of the surface-facing endcap of the waterproof housing shorted a few times during deployment. It is estimated that the reason for this was overexposure to light. In other words, the lux it was recording was outside of its dynamic range of (0 to 40,000). Seeing that direct sunlight can range from 30,000 to 100,000 lux, and the sensor was deployed during a clear 70ºF sunny day, it was most likely overexposed. There is a reading in the raw data of 49,143 lux (*Appendix A*) right before recording 65536 (indicating the sensor shorted) repeatedly, meaning it briefly recorded outside of its range

before shorting from the sunlight being too intense for the TSL2561 luminosity sensor. While this could be an oversight of the range of the sensor, the main reason this happened is because the sensor was not deployed very deep into the water column. It was neutrally buoyant, and the top of the sensor sat inches below the surface (*Figure 11*). For future deployments, this issue can be avoided by deeper deployment, and possibly a stronger luminosity sensor.

Furthermore, since the lux sensor was placed at the top of the sensor housing to face upwards towards the sunlight penetration (*Figure 11*), it may show a less accurate representation of the actual light penetration where the shellfish reside. The solution to this inconsistency is to have another light sensor in the topside dock junction box as well. With two light sensors, the ARM sensor can derive the light attenuation at the depth where the shellfish are. This is a dynamic solution since the shellfish depth will vary from site to site, and is not a major design change.

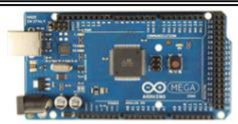
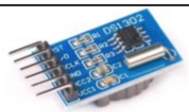
Conclusion

The future of the ARM sensor is promising. As mentioned earlier there is a clear need for its capabilities in the aquaculture industry, especially among the hundreds of small farms that are the backbone of the industry itself. With the improvements noted above, this sensor can remain low-cost and be of use to farms all over the Puget Sound. As an easy, informative, and portable tool, ARM can accompany farmers to more effective operational practices. More importantly, ARM can be used to facilitate the conversation of environmental security across multiple industries. With the information and data provided by ARM, the aquaculture industry- whom have a strong political, economic, and social influence- can create discourse on increasingly critical

subjects such as climate change and its harmful effects. The aquaculture industry provides a bridge from the science community to the rest of the world, and the ARM sensor provides the data that creates the stability for the bridge to be crossed. The world is changing, and the public must be educated on the implications of what's to come, and how to avoid irreversible damage. While not the solution, the ARM sensor is a stepping stone to a solution by directly representing the necessity for environmental security.

APPENDIX

Appendix A. Components of ARM Sensor

Product		Specifications
ARDUINO MEGA 2560 REV3		• 5V Operating Voltage • 54 Digital I/O Pins • 16 Analog Pins • 256 KB Flash Memory • 20 mA/ pin
DS1307 Real Time Clock Breakout Board		• Keep time up to 5 Years • 5V Module PWR • I2C 7-bit address 0x68
TSL2561 Luminosity Sensor		• Dynamic Lux Range: 0.1-40,000 • I2C interface • Temperature range: -30 to 80 ºC
DF Robot PH Meter		• BNC Connector • 0-14 PH Range, 5V Module PWR • Temperature Range 0-60 ºC • Response Time < 1 min
DFRobot Gravity Analog Dissolved Oxygen Sensor		• BNC Connector • Detection Range: 0~20 mg/L • Temperature Range: 0~40 °C • Response Time 90 sec
DFRobot Gravity Analog Electrical Conductivity Sensor / Meter(K=10)		• BNC Connector • Measurement Accuracy: ±5% F.S. • Temperature Range: 0~40°C • Supply Voltage: 3.0-5.0V • Support Detection: 10~100ms/cm
DS18B20 Digital Temperature Sensor		• -55ºC to 125 ºC ± 0.5 ºC • Waterproof • 3.0V to 5.5V • 1-Wire® interface
Adafruit MicroSD card breakout board+		• 3v or 5v systems • LED indicator • 2Gb+ of Storage • Push pup socket
Sparkfun Sunny Buddy Solar Charger V13		• maximum charge current of 450mA • maximum recommended input of 20V (minimum 6V) • LT3652 power tracking
Sparkfun 3.5W Solar Panel		• Output around 6V at 615mA • Waterproof urethane coating • 8.3in x 4.4in x 0.2in • Monocrystalline 19%eff. cells

Appendix B. ARM Sensor Code

```
//Emily Hammermeister
//FINAL THESIS CODE

#include <SPI.h> //for the SD card module
#include <SD.h> // for the SD card
#include <RTClib.h> // for the RTC
#include <OneWire.h>
#include "Wire.h"
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>
#include <EEPROM.h>
#include <avr/pgmspace.h>
#include "DFRobot_EC10.h"


////Definitions & Declarations////

/////pH/////
#define SensorPin A0            //pH meter Analog output to Arduino Analog Input 0
#define Offset 0.43            //deviation compensate
#define LED 13
#define samplingInterval 200
#define printInterval 60000
#define ArrayLenth  40    //times of collection
int pHArray[ArrayLenth];   //Store the average value of the sensor feedback
int pHArrayIndex = 0;

///DO////
#define DoSensorPin  A2    //dissolved oxygen sensor analog output pin to arduino
mainboard
#define VREF 5000    //for arduino uno, the ADC reference is the AVCC, that is
5000mV(TYP)
float doValue;      //current dissolved oxygen value, unit; mg/Lfloat temperature
= 25;
//default temperature is 25^C, you can use a temperature sensor to read it

#define EEPROM_write(address, p)
{
  int i = 0;
  byte *pp = (byte*) & (p);
  for (; i < sizeof(p); i++) EEPROM.write(address + i, pp[i]);
}
#define EEPROM_read(address, p)
{



 int i = 0;
 byte *pp = (byte*) & (p);
 for (; i < sizeof(p); i++) pp[i] = EEPROM.read(address + i);
}

#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength + 1];  // store the serial command
byte receivedBufferIndex = 0;

#define SCOUNT  30           // sum of sample point
int analogBuffer[SCOUNT];    //store the analog value in the array, readed from
ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0, copyIndex = 0;

#define SaturationDoVoltageAddress 12
#define SaturationDoTemperatureAddress 16
float SaturationDoVoltage, SaturationDoTemperature;
float averageVoltage;

const float SaturationValueTab[41] PROGMEM = {
  //saturation dissolved oxygen concentrations at various temperatures
  14.46, 14.22, 13.82, 13.44, 13.09,
  12.74, 12.42, 12.11, 11.81, 11.53,
  11.26, 11.01, 10.77, 10.53, 10.30,
  10.08, 9.86,  9.66,  9.46,  9.27,
  9.08,  8.90,  8.73,  8.57,  8.41,
  8.25,  8.11,  7.96,  7.82,  7.69,
  7.56,  7.43,  7.30,  7.18,  7.07,
  6.95,  6.84,  6.73,  6.63,  6.53,
  6.41,
};


/////Salinity/////
#define EC_PIN A1
float voltage, ecValue, temperature;
DFRobot_EC10 ec;

////Temp///
int DS18S20_Pin = 2; //DS18S20 Signal pin on digital 2
//Temperature chip i/o
OneWire ds(DS18S20_Pin);  // on digital pin 2
```

```
//SD
const int chipSelect = 53;


// Create a file to store the data
File myFile;

// RTC
RTC_DS1307 rtc;


//Lux
Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT,
12345);


/////SETUP////
void setup() {

  //Lux
  configureSensor();

  //initializing Serial monitor
  Serial.begin(9600);

  Serial.print('\n');

  //////pH//////
  pinMode(LED, OUTPUT);

  //DO///
  pinMode(DoSensorPin, INPUT);
  readDoCharacteristicValues();


  /////Salinity/////
  ec.begin();

  //Printing//

  Serial.println("Clock On");
  Serial.println("Temp On");
  Serial.println("Lux on");
  Serial.println("pH on");
  Serial.println("Salinity on");




  Serial.println("DO on");

  // setup for the RTC
  while (!Serial); // for Leonardo/Micro/Zero
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }
  else {
    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
  if (! rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
  }



  // setup for the SD card
  Serial.print("Initializing SD card...");
  pinMode(53, OUTPUT);
  if (!SD.begin(chipSelect)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");

  //open file
  myFile = SD.open("DATA.txt", FILE_WRITE);

  // if the file opened ok, write to it:
  if (myFile) {
    Serial.println("File opened ok");
    // print the headings for our data
    myFile.println("Date,Time,Temperature °C,Lux,pH,Salinity,DO");
  }
  myFile.close();
}

void loggingTime() {
  DateTime now = rtc.now();
  myFile = SD.open("DATA.txt", FILE_WRITE);
  if (myFile) {
    myFile.print(now.year(), DEC);
    myFile.print('/');
```

```
    myFile.print(now.month(), DEC);
    myFile.print('/');
    myFile.print(now.day(), DEC);
    myFile.print(',');
    myFile.print(now.hour(), DEC);
    myFile.print(':');
    myFile.print(now.minute(), DEC);
    myFile.print(':');
    myFile.print(now.second(), DEC);
    myFile.print(",");
  }
  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.println(now.day(), DEC);
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.println(now.second(), DEC);
  myFile.close();
  delay(60000); // (10 min=600000), 0.5 h = 1800000 ms
}

void loggingTemperature() {

  float temperature = getTemp();



  // Check if any reads failed and exit early (to try again).
  if  (isnan(temperature) /*|| isnan(f)*/) {
    Serial.println("Failed to read from temp sensor!");
    return;
  }

  //debugging purposes
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" C");
  //Serial.print(f);
  //Serial.println(" *F\t");

  myFile = SD.open("DATA.txt", FILE_WRITE);


  if (myFile) {
    Serial.println("open with success");
    myFile.print(temperature);
    myFile.print(",");
  }
  myFile.close();
}



void loggingLux()
{

  /* Get a new sensor event */
  sensors_event_t event;
  tsl.getEvent(&event);

  /* Display the results (light is measured in lux) */
  if (event.light)
  {
    Serial.print("Light: ");
    Serial.print(event.light); Serial.println(" lux");
  }
  else
  {
    /* If event.light = 0 lux the sensor is probably saturated
       and no reliable data could be generated! */
    Serial.println("Sensor overload");
  }

  delay(60000); //1 min //0.5 h = 1800000 ms
  myFile = SD.open("DATA.txt", FILE_WRITE);
  if (myFile) {
    Serial.println("open with success");
    myFile.print(event.light);
    myFile.print(",");
  }
  myFile.close();

}


void loggingpH()
{
  /////pH/////
```

```
  static unsigned long samplingTime = millis();
  static unsigned long printTime = millis();
  static float pHValue, voltage;
  if (millis() - samplingTime > samplingInterval)
  {
    pHArray[pHArrayIndex++] = analogRead(SensorPin);
    if (pHArrayIndex == ArrayLenth)pHArrayIndex = 0;
    voltage = avergearray(pHArray, ArrayLenth) * 5.0 / 1024;
    pHValue = 3.5 * voltage + Offset;
    samplingTime = millis();
  }
  if (millis() - printTime > printInterval)
  {

    //Printing
    Serial.print('\n');
    Serial.print("pH value: ");
    Serial.println(pHValue, 2);
    digitalWrite(LED, digitalRead(LED) ^ 1);
    printTime = millis();
  }

  myFile = SD.open("DATA.txt", FILE_WRITE);
  if (myFile) {
    Serial.println("open with success");
    myFile.print(pHValue, 2);
    myFile.print(",");
  }
  myFile.close();


}

void loggingSalinity()
{
  /////Salinity/////

  static unsigned long timepoint = millis();
  if (millis() - timepoint > 1000U) //time interval: 1s
  {
    timepoint = millis();
    voltage = analogRead(EC_PIN) / 1024.0 * 5000; // read the voltage
    temperature = getTemp();  // read your temperature sensor to execute
temperature compensation


    ecValue =  ec.readEC(voltage, temperature); // convert voltage to EC with
temperature compensation

    //Printing

    Serial.print('\n');
    Serial.print("Salinity EC: ");
    Serial.print(ecValue, 2);
    Serial.println(" ms/cm");
  }

  myFile = SD.open("DATA.txt", FILE_WRITE);
  if (myFile) {
    Serial.println("open with success");
    myFile.print(ecValue, 2);
    myFile.print(",");
  }
  myFile.close();

}


void loggingDO()
{

  static unsigned long analogSampleTimepoint = millis();
  if (millis() - analogSampleTimepoint > 30U)
    //every 30 milliseconds,read the analog value from the ADC
  {
    analogSampleTimepoint = millis();
    analogBuffer[analogBufferIndex] = analogRead(DoSensorPin);
    //read the analog value and store into the buffer
    analogBufferIndex++;
    if (analogBufferIndex == SCOUNT)
      analogBufferIndex = 0;
  }

  static unsigned long tempSampleTimepoint = millis();
  if (millis() - tempSampleTimepoint > 500U) // every 500 milliseconds, read the
temperature
  {
    tempSampleTimepoint = millis();
    //temperature = readTemperature();  // add your temperature codes here to
read the temperature, unit:^C
  }
```

```
  static unsigned long printTimepoint = millis();
  if (millis() - printTimepoint > 10000U) //printing 10 sec
  {
    printTimepoint = millis();
    for (copyIndex = 0; copyIndex < SCOUNT; copyIndex++)
    {
      analogBufferTemp[copyIndex] = analogBuffer[copyIndex];
    }
    averageVoltage = getMedianNum(analogBufferTemp, SCOUNT) * (float)VREF / 1024.
0;
    // read the value more stable by the median filtering algorithm
    doValue = pgm_read_float_near( &SaturationValueTab[0] +
(int)(SaturationDoTemperature + 0.5) )
              * averageVoltage / SaturationDoVoltage;
    //calculate the do value, doValue = Voltage / SaturationDoVoltage *
SaturationDoValue(with temperature compensation)

    //Printing
    Serial.print('\n');
    Serial.print(F("Dissolved Oxygen:"));
    Serial.print(doValue, 2);
    Serial.println(F("mg/L"));


  }


  myFile = SD.open("DATA.txt", FILE_WRITE);
  if (myFile) {
    Serial.println("open with success");
    myFile.print(doValue, 2);
    myFile.println(",");
  }
  myFile.close();

}

void loop() {
  loggingTime();
  loggingTemperature();
  loggingLux();
  loggingpH();
  loggingSalinity();
  loggingDO();


  delay(60000); //=10min & 0.5 h = 1800000 ms
}


//Temp Function//

float getTemp() {
  //returns the temperature from one DS18S20 in DEG Celsius

  byte data[12];
  byte addr[8];

  if ( !ds.search(addr)) {
    //no more sensors on chain, reset search
    ds.reset_search();
    return -1000;
  }

  if ( OneWire::crc8( addr, 7) != addr[7]) {
    Serial.println("CRC is not valid!");
    return -1000;
  }

  if ( addr[0] != 0x10 && addr[0] != 0x28) {
    Serial.print("Device is not recognized");
    return -1000;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1); // start conversion, with parasite power on at the end

  byte present = ds.reset();
  ds.select(addr);
  ds.write(0xBE); // Read Scratchpad


  for (int i = 0; i < 9; i++) { // we need 9 bytes
    data[i] = ds.read();
  }

  ds.reset_search();

  byte MSB = data[1];
  byte LSB = data[0];
```

```
  float tempRead = ((MSB << 8) | LSB); //using two's compliment
  float TemperatureSum = tempRead / 16;

  return TemperatureSum;

}


///Lux Functions///



void configureSensor(void)
{
  /* You can also manually set the gain or enable auto-gain support */
  // tsl.setGain(TSL2561_GAIN_1X);      /* No gain ... use in bright light to
avoid sensor saturation */
  // tsl.setGain(TSL2561_GAIN_16X);     /* 16x gain ... use in low light to boost
sensitivity */
  tsl.enableAutoRange(true);            /* Auto-gain ... switches automatically
between 1x and 16x */

  /* Changing the integration time gives you better sensor resolution (402ms =
16-bit data) */
  tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);      /* fast but low
resolution */
  // tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_101MS);  /* medium resolution
and speed   */
  // tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_402MS);  /* 16-bit data but
slowest conversions */


}
///pH FUNCTIONS///

double avergearray(int* arr, int number) {
  int i;
  int max, min;
  double avg;
  long amount = 0;
  if (number <= 0) {
    Serial.println("Error number for the array to avraging!/n");
    return 0;
  }


  if (number < 5) { //less than 5, calculated directly statistics
    for (i = 0; i < number; i++) {
      amount += arr[i];
    }
    avg = amount / number;
    return avg;
  } else {
    if (arr[0] < arr[1]) {
      min = arr[0]; max = arr[1];
    }
    else {
      min = arr[1]; max = arr[0];
    }
    for (i = 2; i < number; i++) {
      if (arr[i] < min) {
        amount += min;      //arr<min
        min = arr[i];
      } else {
        if (arr[i] > max) {
          amount += max;  //arr>max
          max = arr[i];
        } else {
          amount += arr[i]; //min<=arr<=max
        }
      }//if
    }//for
    avg = (double)amount / (number - 2);
  }//if
  return avg;
}


////DO FUNCTIONS///
void readDoCharacteristicValues(void)
{
  EEPROM_read(SaturationDoVoltageAddress, SaturationDoVoltage);
  EEPROM_read(SaturationDoTemperatureAddress, SaturationDoTemperature);
  if (EEPROM.read(SaturationDoVoltageAddress) == 0xFF && EEPROM.read
      (SaturationDoVoltageAddress + 1) == 0xFF && EEPROM.
read(SaturationDoVoltageAddress + 2)
      == 0xFF && EEPROM.read(SaturationDoVoltageAddress + 3) == 0xFF)
  {
    SaturationDoVoltage = 1127.6;   //default voltage:1127.6mv
    EEPROM_write(SaturationDoVoltageAddress, SaturationDoVoltage);
  }
```

```
  if (EEPROM.read(SaturationDoTemperatureAddress) == 0xFF && EEPROM.read
      (SaturationDoTemperatureAddress + 1) == 0xFF && EEPROM.
read(SaturationDoTemperatureAddress + 2)
      == 0xFF && EEPROM.read(SaturationDoTemperatureAddress + 3) == 0xFF)
  {
    SaturationDoTemperature = 25.0;   //default temperature is 25^C
    EEPROM_write(SaturationDoTemperatureAddress, SaturationDoTemperature);
  }
}

int getMedianNum(int bArray[], int iFilterLen)
{
  int bTab[iFilterLen];
  for (byte i = 0; i < iFilterLen; i++)
  {
    bTab[i] = bArray[i];
  }
  int i, j, bTemp;
  for (j = 0; j < iFilterLen - 1; j++)
  {
    for (i = 0; i < iFilterLen - j - 1; i++)
    {
      if (bTab[i] > bTab[i + 1])
      {
        bTemp = bTab[i];
        bTab[i] = bTab[i + 1];
        bTab[i + 1] = bTemp;
      }
    }
  }
  if ((iFilterLen & 1) > 0)
    bTemp = bTab[(iFilterLen - 1) / 2];
  else
    bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
  return bTemp;
}

//DO CALIBRATION
void doCalibration(byte mode)
{
  char *receivedBufferPtr;
  static boolean doCalibrationFinishFlag = 0, enterCalibrationFlag = 0;
  float voltageValueStore;
  switch (mode)
  {


    case 0:
      if (enterCalibrationFlag)
        Serial.println(F("Command Error"));
      break;

    case 1:
      enterCalibrationFlag = 1;
      doCalibrationFinishFlag = 0;
      Serial.println();
      Serial.println(F(">>>Enter Calibration Mode<<<"));
      Serial.println(F(">>>Please put the probe into the saturation oxygen water!
<<<"));
      Serial.println();
      break;

    case 2:
      if (enterCalibrationFlag)
      {
        Serial.println();
        Serial.println(F(">>>Saturation Calibration Finish!<<<"));
        Serial.println();
        EEPROM_write(SaturationDoVoltageAddress, averageVoltage);
        EEPROM_write(SaturationDoTemperatureAddress, temperature);
        SaturationDoVoltage = averageVoltage;
        SaturationDoTemperature = temperature;
        doCalibrationFinishFlag = 1;
      }
      break;

    case 3:
      if (enterCalibrationFlag)
      {
        Serial.println();
        if (doCalibrationFinishFlag)
          Serial.print(F(">>>Calibration Successful"));
        else
          Serial.print(F(">>>Calibration Failed"));
        Serial.println(F(",Exit Calibration Mode<<<"));
        Serial.println();
        doCalibrationFinishFlag = 0;
        enterCalibrationFlag = 0;
      }
      break;
  }
}
```

Sources Cited

Barton, A., Hales, B., Waldbusser, G. G., Langdon, C., & Feely, R. A. (2012). The Pacific oyster, *Crassostrea gigas*, shows negative correlation to naturally elevated carbon dioxide levels: Implications for near-term ocean acidification effects. Limnology and Oceanography,57(3), 698-710. doi:10.4319/lo.2012.57.3.0698

Casey, M. (2015, February 23). Could climate change take oysters off the menu? CBS News. Retrieved from https://www.cbsnews.com/news/ocean-acidification-spells-trouble-for-shellfish-industry/

Decker, K. 2015. Patterns in the Economic Contribution of Shellfish Aquaculture. In Shellfish Aquaculture in Washington State, Final Report to the Washington State Legislature. Seattle, WA.

Doney, Scott. 2008 "Faculty of 1000 evaluation for Evidence for upwelling of corrosive "acidified" water onto the continental shelf." F1000 - Post-Publication peer review of the biomedical literature, May 2008, doi:10.3410/f.1118986.575112.

Dumbauld, B.R., J.L. Ruesink and S.S. Rumrill. 2009. The Ecological Role of Bivalve Shellfish Aquaculture in the Estuarine Environment: A Review with Application to Oyster and Clam Culture in West Coast (USA) Estuaries. Aquaculture 290:196-223.

Feely, R. A., S. R. Alin, J. Newton, C. L. Sabine, M. Warner, A. Devol, C. Krembs, and C. Maloy. 2010. The combined effects of ocean acidification, mixing, and respiration on pH and carbonate saturation in an urbanized estuary. Estuarine, Coastal and Shelf Science 88: 442–449. doi: 10.1016/j.ecss.2010.05.004

Feely, R. A., and S. C. Doney. 2011. Ocean Acidification: The Other CO2 Problem. Limnology and Oceanography e-Lectures. doi: 10.4319/lol.2011.rfeely_sdoney.5

Feely, R.A., T. Klinger, J.A. Newton, and M. Chase (2012): *Scientific Summary of Ocean Acidification in Washington State Marine Waters*. NOAA OAR Special Report.

NANOOS: What is NANOOS Doing? (2012). Retrieved from http://www.nanoos.org/products/oa/ocean_acidification.php?section=nanoos_actions

Northern Economics, Inc. The Economic Impact of Shellfish Aquaculture in Washington, Oregon and California. Prepared for Pacific Shellfish Institute. April 2013.

Northwest Fisheries Science Center. (n.d.). Algal Bloom Dynamics. & NOAA Fisheries Retrieved November 18, 2018, from https://www.nwfsc.noaa.gov/research/divisions/efs/microbes/hab/habs_toxins/phytoplankton/algal_dynamics.cfm

OAP. NOAA Ocean Acidification Program. (2018). Retrieved November 18, 2018, from https://oceanacidification.noaa.gov/Home.aspx

PCSG. Pacific Coast Shellfish Growers Association (2018). Retrieved November 18, 2018, from https://pcsga.org/ecosystem-services/

Rodewald, A. D. (2015, February 18). National and environmental security, two sides of same coin. The Hill. Retrieved from https://thehill.com/blogs/pundits-blog/energy-environment/233060-national-security-and-environmental-security-are-two

Stimson, "Environmental Security." Stimson Center, 13 June 2017, www.stimson.org/programs/environmental-security.

U.S. Department of Commerce, National Atmospheric and Oceanographic Administration. (2011). Fisheries Economics of the U.S. 2009: Economics and Sociocultural Status and Trends Series. www.st.nmfs.noaa.gov/st5/publication/index.html

Washington Sea Grant (2015) Shellfish aquaculture in Washington State. Final report to the Washington State Legislature, 84 p.

Washington State Blue Ribbon Panel on Ocean Acidification (2012): Ocean Acidification: From Knowledge to Action, Washington State's Strategic Response. H. Adelsman and L. Whitely Binder (eds). Washington Department of Ecology, Olympia, Washington. Publication no. 12-01-015.

WHOI (2013). 20 Facts About Ocean Acidification. 20-FACTS Woods Hole Oceanographic Institution Retrieved November 18, 2018, from http://www.whoi.edu/fileserver.do?id=165564&pt=2&p=150429

WSI. Gov. Inslee's Washington Shellfish Initiative. (2018.). Retrieved November 18, 2018, from https://www.governor.wa.gov/issues/issues/energy-environment/shellfish

WSDOE. "Acidification in Puget Sound." *Washington State Department of Ecology - Acidification*, Washington State Department of Ecology, ecology.wa.gov/Water-Shorelines/Puget-Sound/Issues-problems/Acidification.