“Potamoi:” A Device and Data-Driven Mobile App Revolutionizing the Measurement, Storage, and Presentation of Water Quality Metrics

Project ID: MER106

Pittsburgh Regional
Science and Engineering
Fair 2021

Abstract

As climate change gradually devastates life on this planet, poor water quality is particularly on the rise. According to a new report from the United Nations Children's Fund (UNICEF) and the World Health Organization (WHO), some 2.2 billion people around the world do not have safely-managed drinking water, while 4.2 billion go without safe sanitation services and three billion lack basic hand-washing facilities. Adding to this problem is the fact that individuals have neither convenient nor efficient ways of determining the quality of their water, especially since most water quality devices are expensive, hard to use, and possess complex user-interfaces.

In order to create an effective alternative to current water quality measuring devices, I designed and engineered a simple, all-in-one multi-module device, titled "**Potamoi**," to measure, store, and display real-time water quality data for public use.

I designed "**Potamoi**" based on a three-component design principle:

1) *Controller:* Sensor-based Arduino water quality testing device with 20x4 LCD Display-Module
2) *Storage and Transmission:* HC-06 Bluetooth Module which transmits data real-time to MyH_2O, a mobile app interface I created, and Data-Logging Shield Module which stores data for future retrieval
3) *View:* Mobile app interface, "MyH_2O," to display data

I measured and recorded the quality of water in the Allegheny, Monongahela, and Ohio Rivers, three rivers in Pittsburgh, my hometown. My component-based Arduino-device consists of pH, temperature, TDS (Total Dissolved Solids), and turbidity (Cloudiness of the water) sensors for different points-of-testing.

I sampled 5 locations in each of the three rivers. I tested each sample 200 times with four sensors each time. I validated the accuracy of my device by conducting a comparative analysis: I compared water quality data produced by my device with that generated by an industrial-grade water quality meter.

Based on the results of my statistical analysis, my device invention relatively accurately tests water quality. The mobile app I created, called "MyH_2O," displays water quality measurements in real-time on a user-friendly interface accessible to anyone, anytime, anywhere. "Potamoi" can successfully help individuals around the world access and view much-needed water quality measurements.

Please click HERE to view a brief video of my device, "Potamoi", and my app, "MyH_2O", in action prior to reading this document

Table of Contents

1. Introduction

1.1 Water

Lack of access to clean water is a global issue. We even see this in the most developed country in the world, the United States. Living in Pittsburgh, Pennsylvania all of my life and seeing how the Allegheny, Monongahela, and Ohio rivers are such staples of the city, I was motivated to investigate current commercially available devices for measuring water quality.

With Pennsylvania being the third largest producer of coal in the United States, it is not surprising that 4,010 of the state's public water systems violated the Environmental Protection Agency's (EPA's) water quality standards in 2019. As I found through my research, poor mining practices not only affect Pennsylvania's water quality, but that of many other states as well. For example, after years of poor mining exercises in Ohio have left many Ohio communities with water containing unsafe levels of iron, sulfate, and other mineral compounds. Overall, unregulated mining practices cause pollution in the form of rust-colored streams.

The Pittsburgh Water and Sewer Authority (PGH_2O) measures and reports the water quality of four drinking water reservoirs as well as other small rivers and streams in Pittsburgh. Not only is the reporting of PGH_2O not robust, but this characterizes other countries' water quality reporting as well. For example, the HSBC Water Program in India uses only colored test trips to determine the pH level of water samples. Given that the latest water quality report PGH_2O has conducted has only been in 2019 and that countries around the world are still struggling to advance their water quality metric reporting, the significance of the tool I have created in this project cannot be understated.

Water Pollution in Africa

	According to https://thelastwell.org/, Every hour, an estimated 115 people die in Africa from diseases linked to improper hygiene, poor sanitation, and contaminated water, according to the United Nations Department of Economic and Social Affairs (UNDESA). The main causes of this are Agriculture, Mining, Deforestation, Urbanization, and Poor Sanitation.



REF : HSBC Bank-sponsored Water Program in India

Featured linkedin post by Sathya Narayana Kaliprasad

Head of CRE IT in HTI | Doctor of Philosophy @ LPU in RE Analytics Mgmt using DS



2. Engineering Problem

My introductory research led me to believe that we have a key problem to solve: Current methods of testing water quality are not robust and do not meet the needs of individuals who are unaware they are drinking unsafe water.

To counter this prevalent issue, I researched commercially available water quality measuring devices currently on the market. I discovered that many of these devices contain key parameters including temperature, TDS, pH, and turbidity. The importance of these parameters cannot be understated—through these, users can access data on water cloudiness, acidity, level of solids contained, and temperature. There are many beneficial aspects of commercially available water quality devices. For example, the pocket pH and TDS meters made by HANNA Instruments are known for their portability and speed of testing. The Hach 2100Q Portable Turbidimeter allows users to accurately measure water samples. Algorithms calculate turbidity based on a series of automatic readings.

While there are many beneficial aspects of currently-available devices, I noted three major flaws in these devices as well:

1) Many do not possess the ability to record and/or access historical water quality data specific to users,

2) Interfaces commonly lack simplicity, and

3) Average costs are significantly high for all-in-one devices with temperature, pH, TDS and Turbidity (approximately $120 to $4000).

2.1 A Sample of Devices Currently on the Market

Product Name	Parameters measured	COST (US$)	Drawback
90XL Dialysis Technician Meter	pH, Temperature, Electrical Conductivity, Pressure	$1700.00	Not for household use
Thermo Scientific VSTAR94 – Orion Versa Star Pro Meter Kit (CONTROL FOR PH)	pH, Electrical Conductivity, TDS, Salinity	$3724.00	Very expensive, not for household use
CAT. NO. 44740-88 , PORTABLE TURBIDIMETER Model 2100P ISO by Hach (CONTROL FOR TURBIDITY)	Turbidity	$1396.00	Can only conduct one reading at a time
Oven; 2.3 cu. Ft., General Protocol, Gravity Convection, Heratherm, Coated Steel, 120 V (CONTROL FOR TDS")	TDS	$1658.00	Not for household use
YSI ProQuatro Multiparameter Meter	Dissolved Oxygen, Electrical Conductivity, Salinity, Resistivity, TDS, Temperature	$2784.00	Very expensive
Bluelab Guardian Monitor	Electrical conductivity/temperature/pH	$411	Many reviews on Amazon.com say that the probes do not work

PC : Amazon.com

The following are some devices currently on the market that I researched:

<table>
<tr>
<td>  </td>
<td>  </td>
<td>  </td>
</tr>
<tr>
<td>  </td>
<td>  </td>
<td>  </td>
</tr>
</table>

2.2 Engineering Goal with Cost Benefit

My <u>engineering goal</u> was to tackle the three aforementioned challenges while also incorporating the beneficial aspects of currently-available devices. Thus, I created both a device ("Potamoi") and an app ("MyH_2O") that measures water quality and, unlike currently-available devices, not only records historical data, but presents this data in an easily accessible and budget friendly manner. To prove the accuracy of "Potamoi's" results, I used a set of entries from the Industry grade device for the same water samples as my control. I chose the name "Potamoi" for my device because this is the name of the gods of the rivers and streams in Greek mythology.

Item	Cost
• Arduino Uno	$20
• DS18B20 Temperature Sensor	$6
• pH Sensor	$15
• TDS Sensor	$12
• Turbidity Sensor	$8
• Resistors	$2
• Jumper Wires	$2
• Breadboard	$7
• External Battery Pack	$2
• 20x4 LCD Display	$10
• 10k Potentiometer	$2
• Hc-06 Bluetooth Module	$4
Total Cost of my Device	**$90**

The cost of my device is under $100 in total:

2.3 Industrial-grade device used as control

Turbidimeter

$1396.00

TDS Meter

$1658.00

These are the devices used by the lab I sent my water samples to. I received the industrial-grade device measurements from there, and this was my control , which means I will compare my device readings (Experimental Group) to this device readings (Control Group)

pH Meter

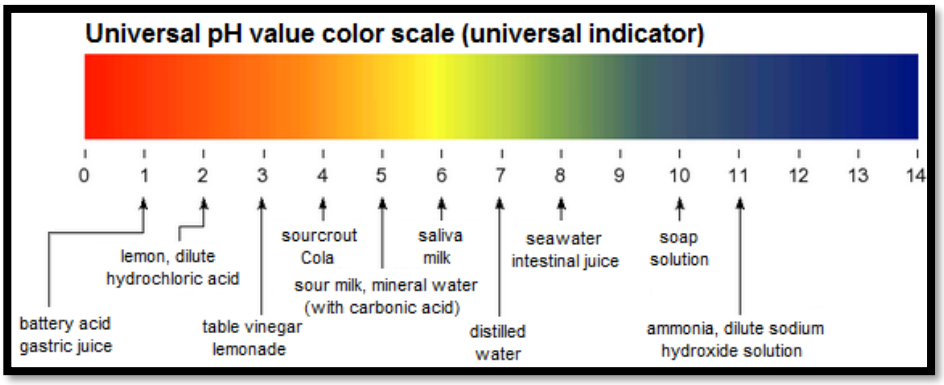
$3724.00

3. Engineering Approach

3.1 Water Quality Parameters Measured

Temperature – Warmer waters hold less dissolved oxygen than cooler waters, and may not contain enough dissolved oxygen for the survival of different aquatic species. Some compounds are also toxic to aquatic life at higher temperatures. Therefore, I selected temperature as one of my parameters.

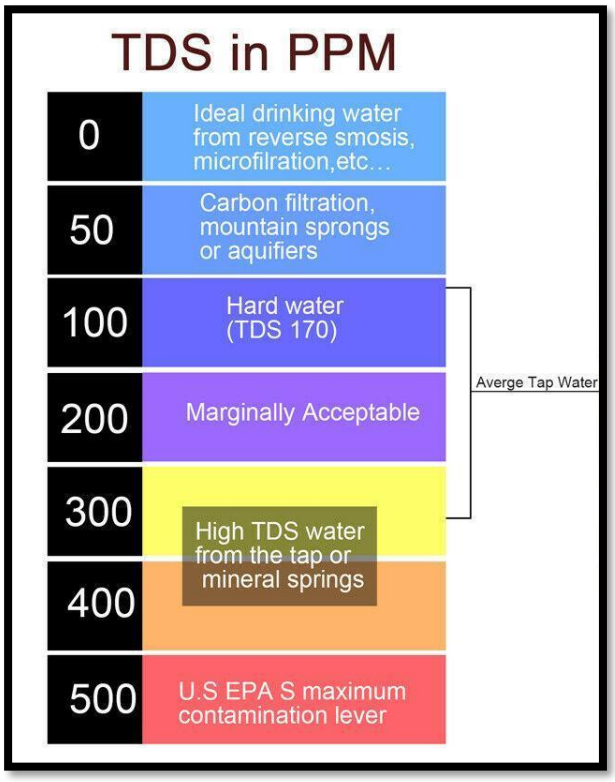
Potential of hydrogen (pH) - pH is critical to the vegetation and plant life of bodies of water. Contaminates from other sources, such as mine drainage, acid rain, or chemical spills can also cause drastic changes in pH levels. Thus, I used pH as one of my parameters.



PC : usgs.org

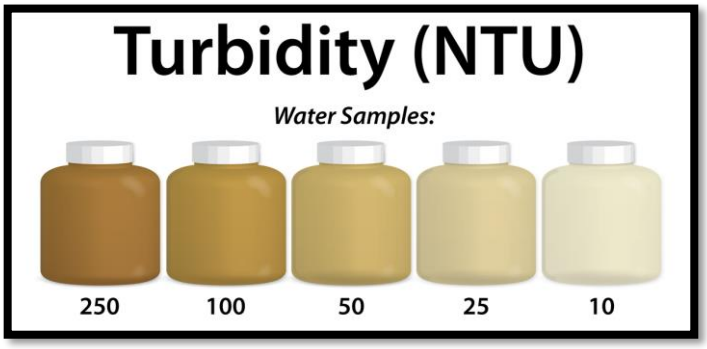
- **Conductivity/Total Dissolved Solids (TDS)** – TDS stands for total dissolved solids. It represents the concentration of dissolved substances in water. High amounts of TDS in water can corrode pipes, stain household fixtures, and cause water to have a metallic taste. The lower the TDS level, the better the quality of the water. For these reasons, I chose TDS as one of my parameters for testing. The main unit of measurement for TDS is PPM (Parts Per Million)

PC : usgs.org



Level of TDS (milligrams per litre)	Rating
Less than 300	Excellent
300 - 600	Good
600 - 900	Fair
900 - 1,200	Poor
Above 1,200	Unacceptable

- **Turbidity/suspended solids** - Turbidity is a measure of the degree to which the water loses its transparency due to the presence of suspended particles. There are many causes of higher turbidity, including sediments from erosion, water discharge, and algae growth. The more suspended amount of solids that exist in the water, the murkier the water and the higher its turbidity. The main unit of measurement for turbidity is NTU (Nephelometric Turbidity unit)



PC : usgs.org

3.2 Procedural Plan

Prior to conducting the experiment, I tested four different types of tap water with my device, "Potamoi," in order to ensure that the sensors were functioning properly.

Next, I selected 5 different locations in each of three rivers: the Ohio, Allegheny, and Monongahela rivers. I tested samples at various depths and with all four parameters.

Ultimately, I collected 200 data points from each location of each river using "Potamoi." It took "Potamoi" 10 minutes to report 200 data points for all 4 sensors at the same time.

ALL THE PICTURES WERE TAKEN BY THE STUDENT

	Collecting water from different locations
	

	I transferred each water sample into a sterilized container.
	*Ohio River* – Five samples each from a different location *Allegheny River* – Five samples each from a different location *Monongahela River* – Five samples each from a different location

3.3. Control Group and Parameters

River 1 (variable)	Control Group	Experimental Group
Location 1	Industry-Grade Water Quality readings	Water Quality readings from "Potamoi"
Location 2		
Location 3		
Location 4		
Location 5		

Independent Variable: Location and 2 different groups
Dependent Variable: water quality readings from "Potamoi"

3.4 Hardware setup

"Potamoi" stores data received from the four sensors onto an SD card. This feature, which does not exist in many currently-available devices, enables users to access their data whenever they desire. I incorporated four parameters, including temperature, pH, TDS, and turbidity, which all can be found on currently-available devices.

Image	Description
	Temperature Sensor - Purchased from Amazon. Uses 3 pins: Ground, VCC, and Data. Ground pin connects to the negative power side on the breadboard. VCC connects to the positive power side on the breadboard. The Data pin connects to any Analog pin on the Arduino Uno to be able to send data from the Arduino to a computer.
	pH Sensor - Purchased from Amazon. Uses 3 pins: Ground, VCC, and Data. Ground pin connects to the negative power side on the breadboard. VCC connects to the positive power side on the breadboard. The Data pin connects to any Analog pin on the Arduino Uno to be able to send data from the Arduino to a computer. Also includes a calibration tool to be able to calibrate the sensor before the first time using it.

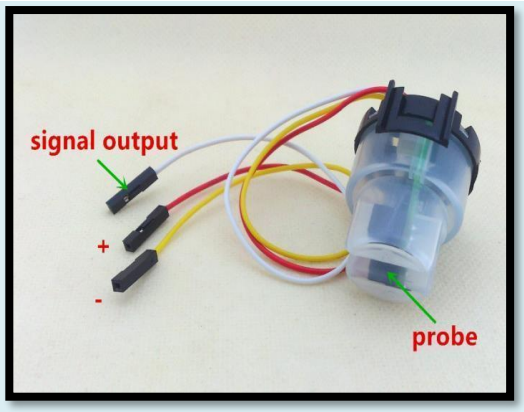
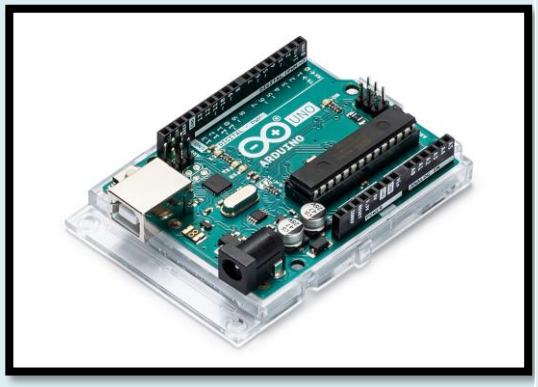
Image	Description
	TDS Sensor - Purchased from Amazon. Uses 3 pins: Ground, VCC, and Data. Ground pin connects to the negative power side on the breadboard. VCC connects to the positive power side on the breadboard. The Data pin connects to any Analog pin on the Arduino Uno to be able to send data from the Arduino to a computer. Also includes a calibration tool to be able to calibrate the sensor before the first time using it.
	Turbidity Sensor - Purchased from Amazon. Uses 3 pins: Ground, VCC, and Data. Ground pin connects to the negative power side on the breadboard. VCC connects to the positive power side on the breadboard. The Data pin connects to any Analog pin on the Arduino Uno to be able to send data from the Arduino to a computer. Includes a Digital to Analog switch to be able to switch between Digital (Only measures two values: HIGH and LOW) and Analog (Can measure any number values) modes. Also includes a calibration tool to be able to calibrate the sensor before the first time using it.
	**Arduino Uno Microcontroller -** • Includes 5 Analog pins, 13 Digital pins, a 5v pin (positive power side), 3 GND pins (negative power side), and a VIN pin (for external power supplies), and a Reset pin. • Can have any type of sensor shields put on top of it (for example, a data-logging shield).

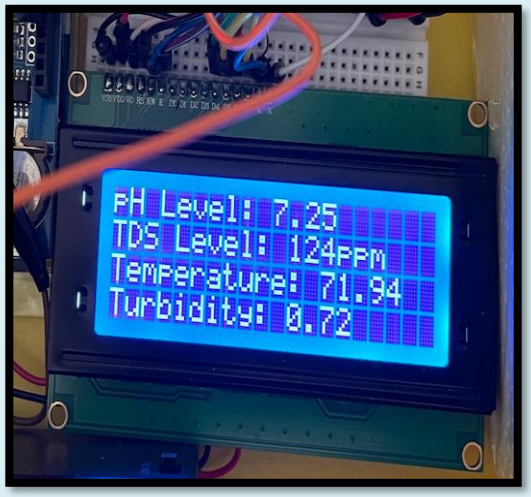
Image	Description
	**Programming Language - Arduino IDE** • This programming language is easy to use and is similar to C and C++ • Different sensors can be integrated in this programming using various libraries found on the web • Libraries are files written in C to provide more functionality to a program.
	**Potentiometer** • A potentiometer provides varying amounts of resistance through a circuit when the knob is turned • In my case, the potentiometer is used to adjust the contrast of the LCD screen
	**Breakdown of LCD Monitor** • The LCD Screen has 16 pins, but only 12 need to be connected to the microcontroller to function • The 1st, 2nd, 15th, and 16th pins are power pins that feed electricity to the LCD • Pin 3 is Vo which gives contrast to the color of the LCD • Pins 4, 5, and 6 (RS, R/W, E) enable letters to be written on the LCD to form words • Pins 11 thru 14 are digital pins that allow the LCD to communicate with the Arduino to form letters

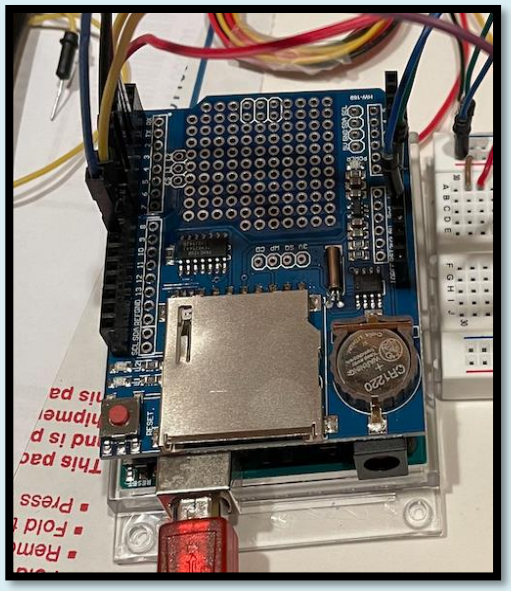
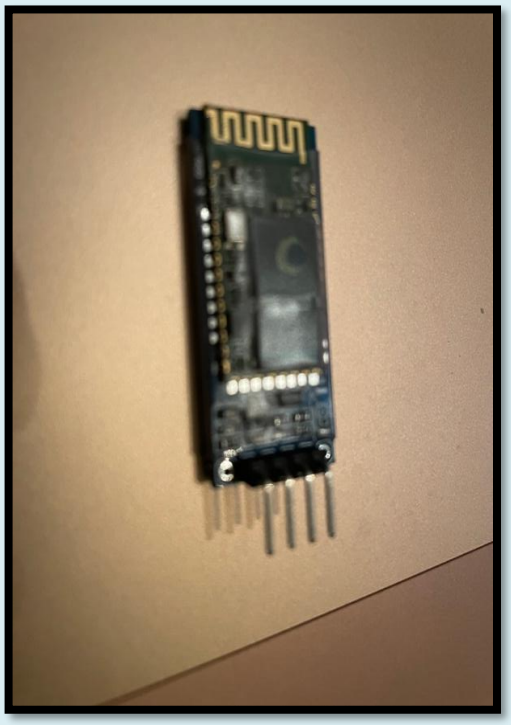
Image	Description
	**Data Logging Shield** • An Arduino-compatible shield that was bought from Amazon and was used to log the data to an SD Card • For the shield to function, I needed to undertake an extensive amount of extra coding and include extra libraries • Uses a RTC Clock Chip to log the date and time down to the second
	**HC-06 Bluetooth Module** • A module that allows a user to connect an Arduino to any interfacing app via Bluetooth • Has 4 pins: GND (negative power side), VCC (positive power side), TX (Transmit Data), and RX (Receive Data) • Works in conjunction with the Blynk app to control the microcontroller wirelessly from far away **Resistors** • Resistors are electrical components that limit the amount of current that is being run through a circuit • If too much current is running through a circuit than a sensor can handle, the sensor will get severely damaged and won't function properly

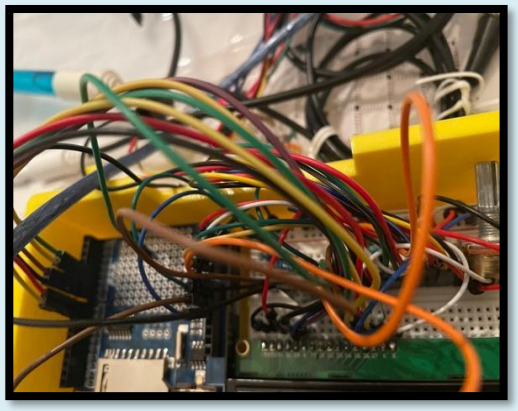
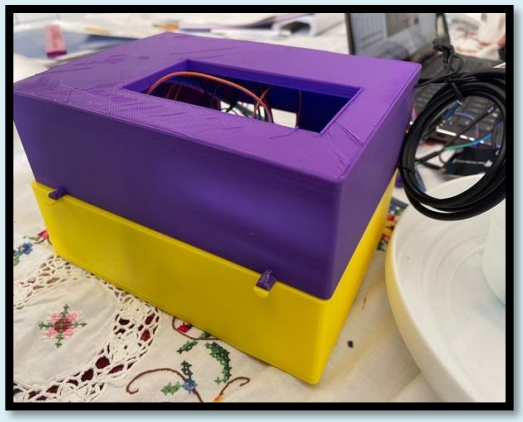
Image	Description
	**Jumper wires** • Jumper wires are used to connect two electrical points together and allow electricity to flow through a circuit and perform a function • Without jumper wires, there can be no circuit
#include <EEPROM.h> **#include "GravityTDS.h"** **#include <DallasTemperature.h>** **#include <OneWire.h>** **#include <LiquidCrystal.h>** **#include "RTClib.h"** **#include <Wire.h>** **#include <SD.h>**	**Libraries Used** • RTCLib and Wire – Used to communicate with the RTC Clock on the Data Logging Shield • SD – Used to communicate with the SD Card Reader on the Data Logging Shield • OneWire and DallasTemperature – Used to communicate with DS18B20 Temperature Sensor • GravityTDS – Used to communicate with the TDS Sensor • EEPROM – Used to safely transfer data from Arduino to any external sources • LiquidCrystal – Used to communicate with the LCD Screen

Image	Description
	3-D printed case to hold my device, Potamoi

3.5 Software

"MyH_2O", the mobile app I created, revolutionizes the water quality data storage and display industry using *Figma*, a prototyping app designing tool. I also learned and used *Bravo Studio*, which turns prototyped apps into functioning, interactive apps. As this is an engineering project, I intentionally chose to use *Figma* and *Bravo Studio* because they do not require any code, and thus I was able to engineer my app both accurately and efficiently.

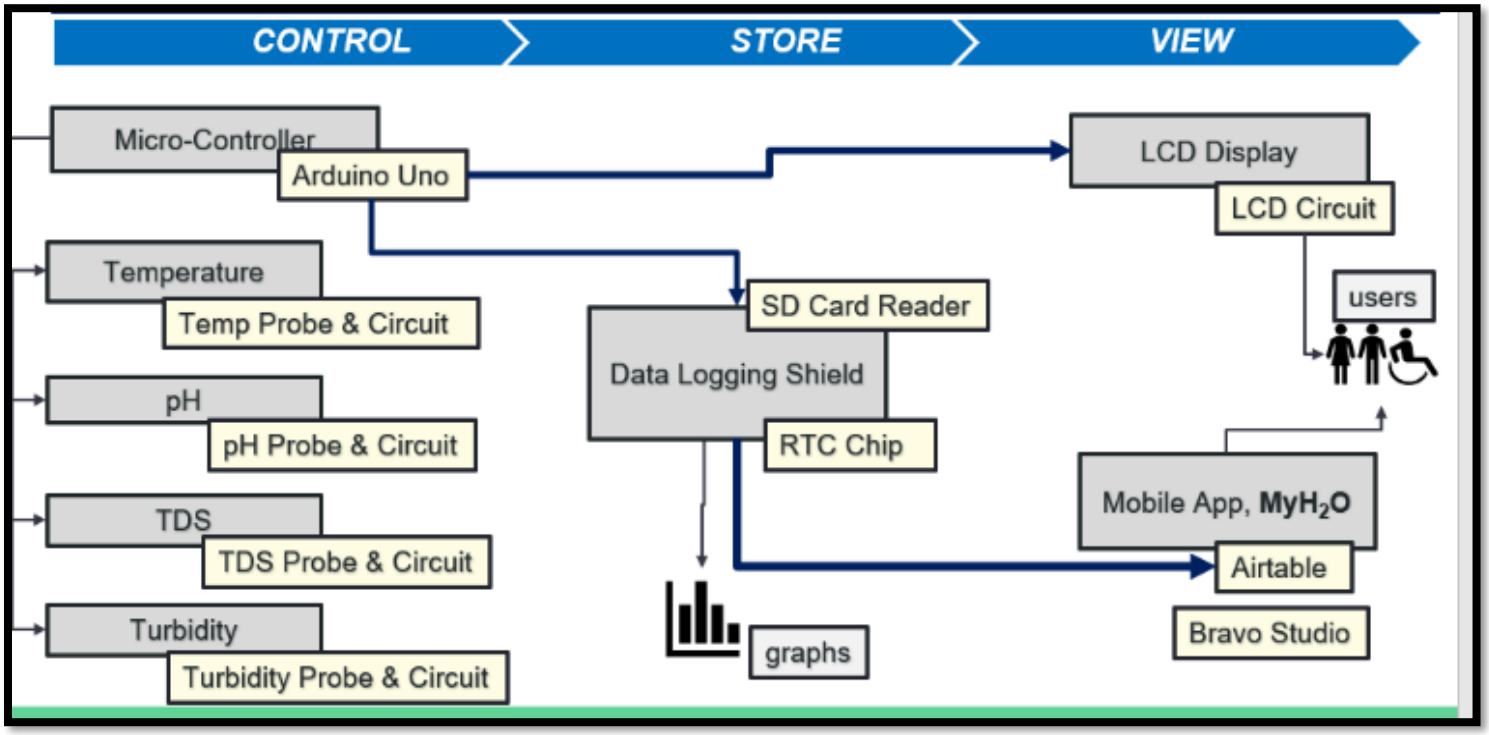
Software	Usage
	• MyH_2O was made by first creating the design of all of the screens in Figma • Figma is a service that allows users to create wireframes that are used as a proof-of-concept to make any app (mobile or browser) • Figma uses prototyping, which is how a user can interact with buttons, sliders, drop down menus, or even custom loading screens

Software	Usage
	• Airtable is a spreadsheet database similar to Excel but it can communicate with different app-building interfaces to send data using an API • The data that is being collected from “Potamoi” feeds into an Airtable spreadsheet, which is linked to Bravo Studio to upload data real-time into MyH$_2$O, my mobile app.
	• Bravo Studio is paired with Figma to make a fully functioning app that you can deploy to the iOS app store or Google Play Store. • Bravo Studio allows the user to bind and collect data from any external source like Airtable or Google Sheets, and integrate them into the app. • This allows the user to be able to update the data of an app in real-time when connected to a table in Airtable. • In conjunction with Bravo Studio, the user can use Bravo Vision to preview their app to see what it would like from an outsider’s view.

4. Engineering Design and Procedure

4.1 Components of “Potamoi” Device

Three Design principles applied.



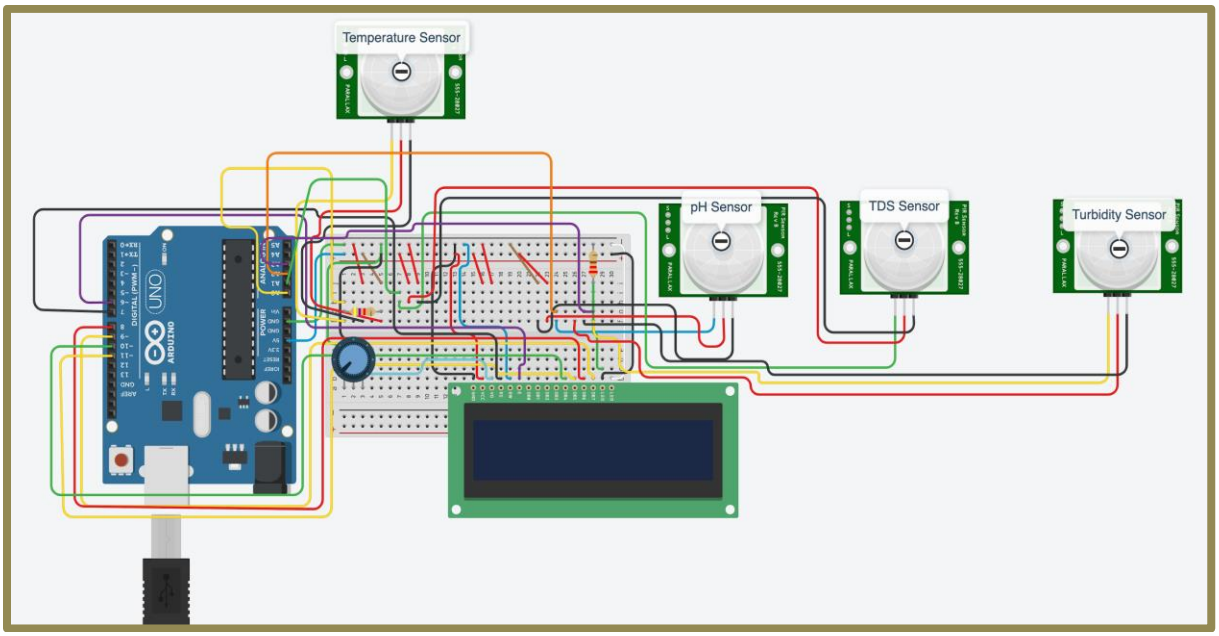
I downloaded the Arduino IDE software which is similar to C and C++ and I programmed the Arduino microcontroller.

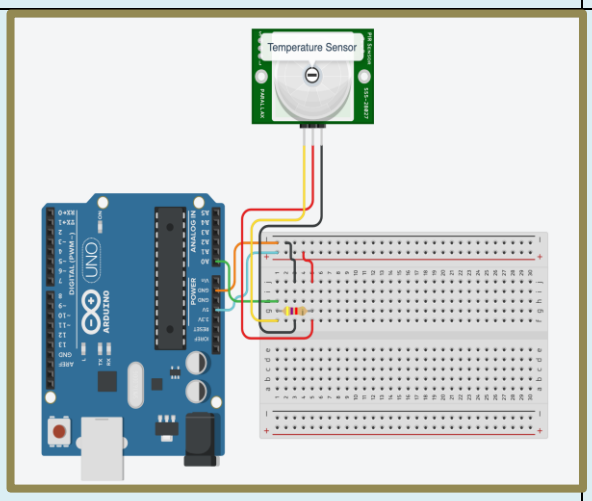
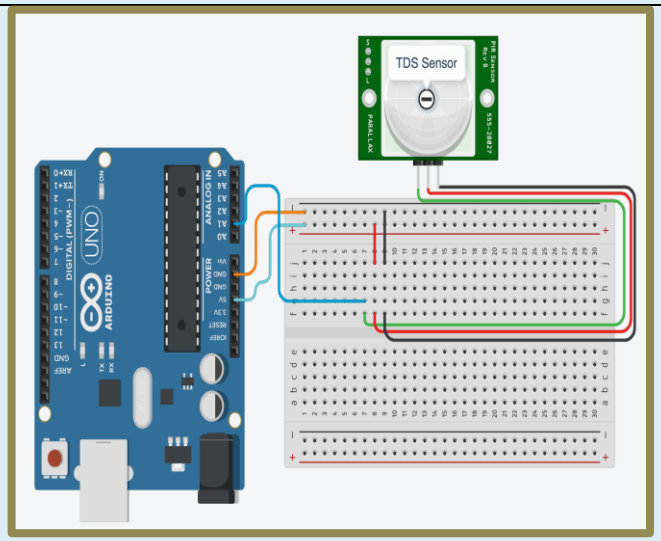
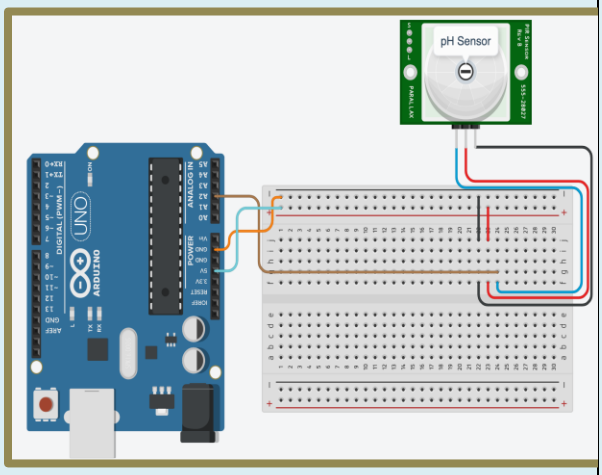
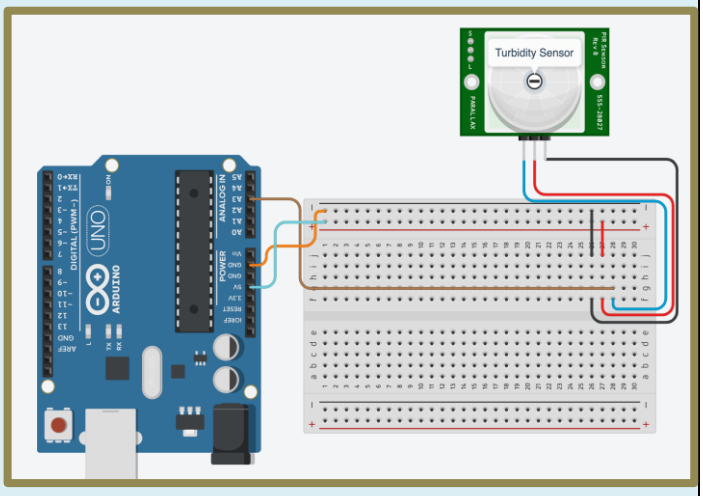
- I purchased and integrated the four sensors (pH, TDS, turbidity, and temperature) into the program using various libraries found on the web
- Libraries are files written in C to provide more functionality to a program
- I purchased an Arduino-compatible data-logging shield and used it to log the data onto an SD Card
- The shield took an extensive amount of extra coding and many different libraries needed to be included
- Shield uses an RTC Clock Chip to log the date and time
- I used the Bluetooth Module to send data directly to the Blynk App, which is an app available free for Arduino programmers like me. For this research though, I decided that Blynk did not display the data in a robust way. Therefore, I decided to build **my own app which I titled "MyH$_2$O".**

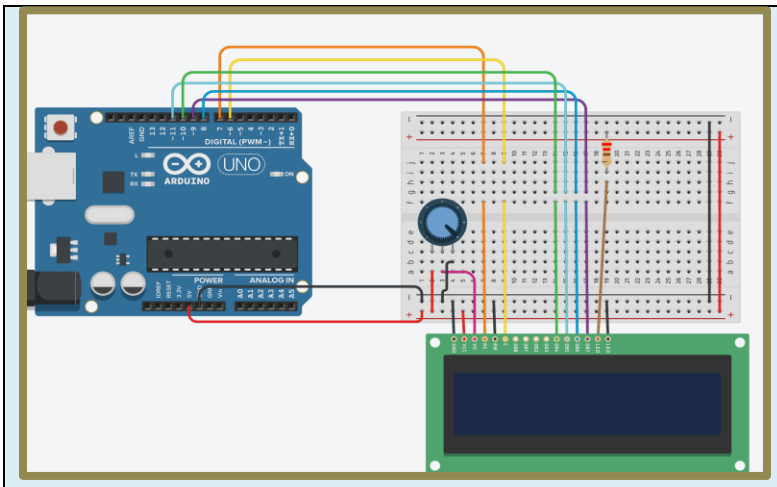
4.2 Potamoi Build Procedure

- I sketched the design for my device on TinkerCad, a design software
- I created the device using four sensors, an LCD display, various jumper wires, resistors, a breadboard, and an Arduino Uno
- I coded in Arduino IDE software and ran the code with the device to ensure reliability
- I collected five water samples from each of the three main rivers of Pittsburgh by using a prong-shaped tool connected to a bucket and pouring the water into plastic containers for testing. Altogether, I collected 15 different water samples.
- I placed the sensor probes into each of the water samples, and I collected 200 data points in 10 minutes and stored these on a computer through the Arduino Serial Monitor and an SD Card. The data was displayed on the computer screen using PuTTY, a software that transfers data onto a file
- I saved the data onto a text file transmitted to Airtable (a free, browser-based tool I learned how to use during the past three months)
- I created a user-friendly app, called "MyH_2O" to display the data, including averages, date of testing, and testing locations.

Full Circuit Diagram – Arduino Uno connects to the 4 sensors (photo credit: Student using TinkerCad to design the circuit)

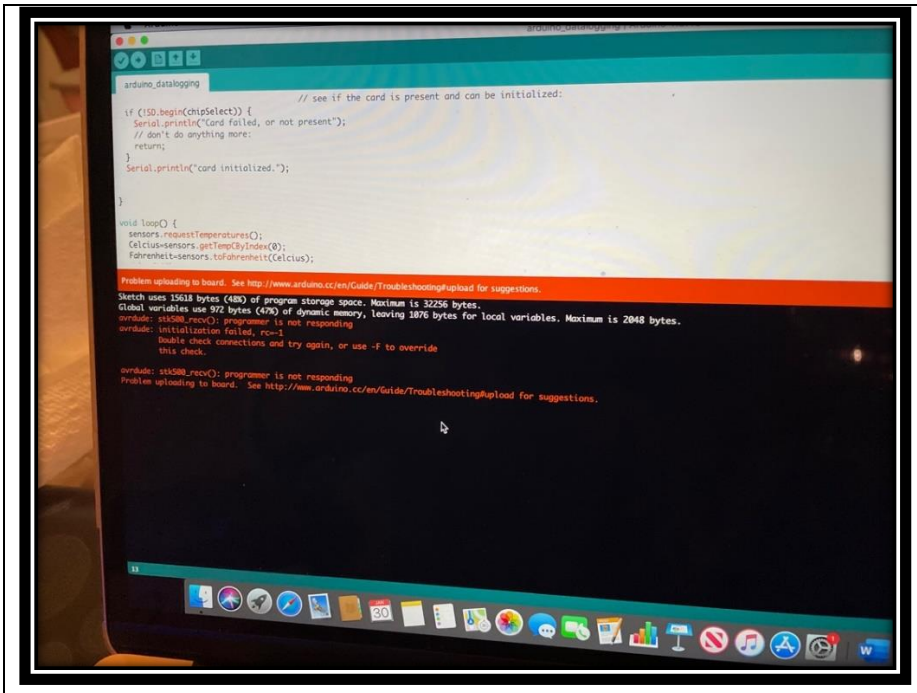


Temperature Sensor Circuit (photo credit: Student using TinkerCad to design the circuit)	**TDS Sensor Circuit** (photo credit: Student using TinkerCad to design the circuit)
	
pH Sensor Circuit (photo credit: Student using TinkerCad to design the circuit)	**Turbidity Sensor Circuit** (photo credit: Student using TinkerCad to design the circuit)
	
Even though the full circuit diagram looks very complicated, each individual circuit only has 3 components, and is very self-explanatory to wire.	



LCD Display and Potentiometer Circuit (photo credit: Student using TinkerCad to design the circuit)

Errors and Debugging Efforts during Arduino Programming



This is one of many errors I encountered throughout the Arduino programming. There were many instances, where when I fixed an error a new issue generated and program stopped uploading to Arduino

Below are pictures of how I made my Device from scratch (Photo Credit: Student)

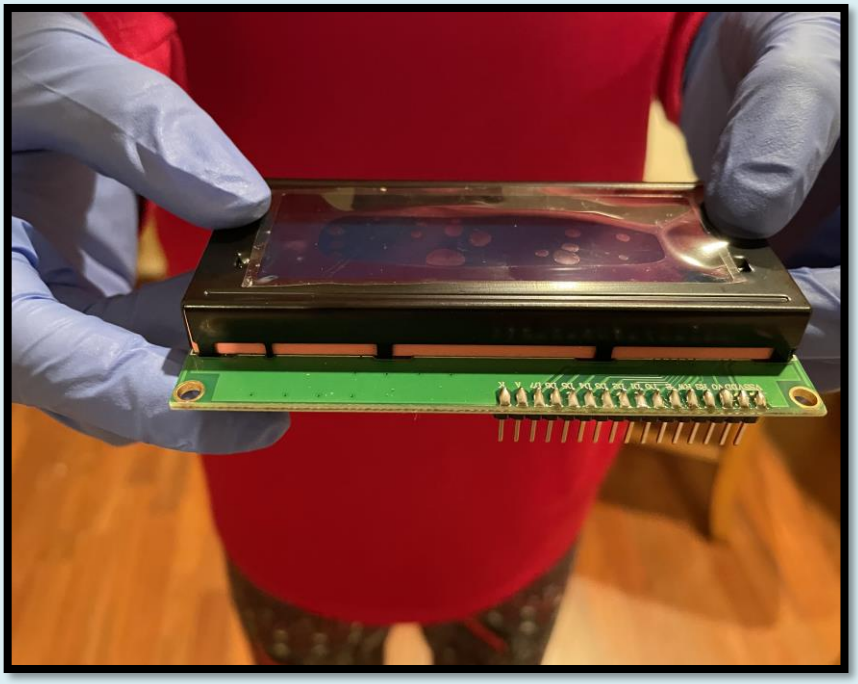
Image	Description
	Unpacking the LCD
	Getting ready with the Soldering Iron for my first step

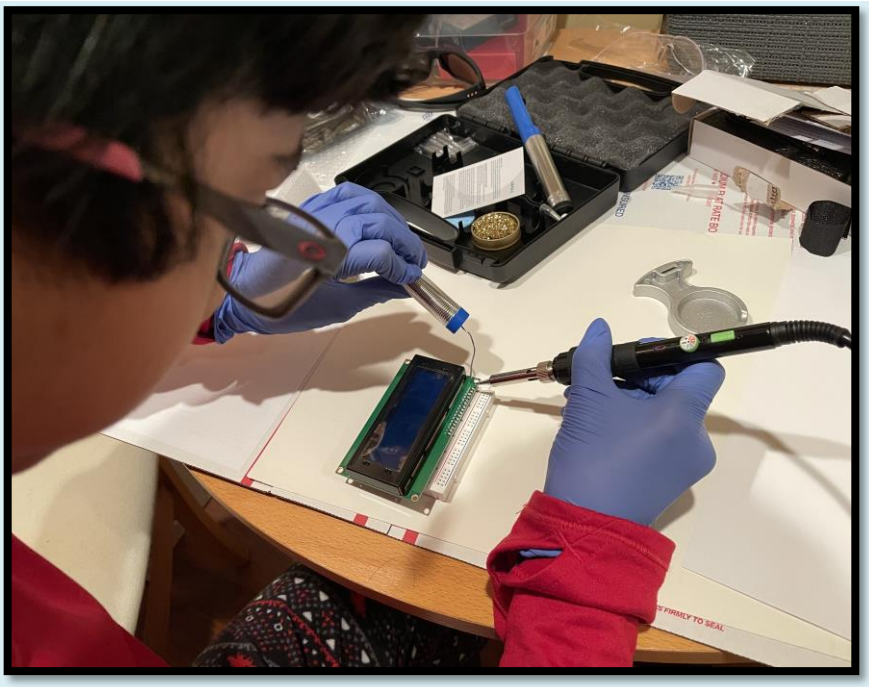
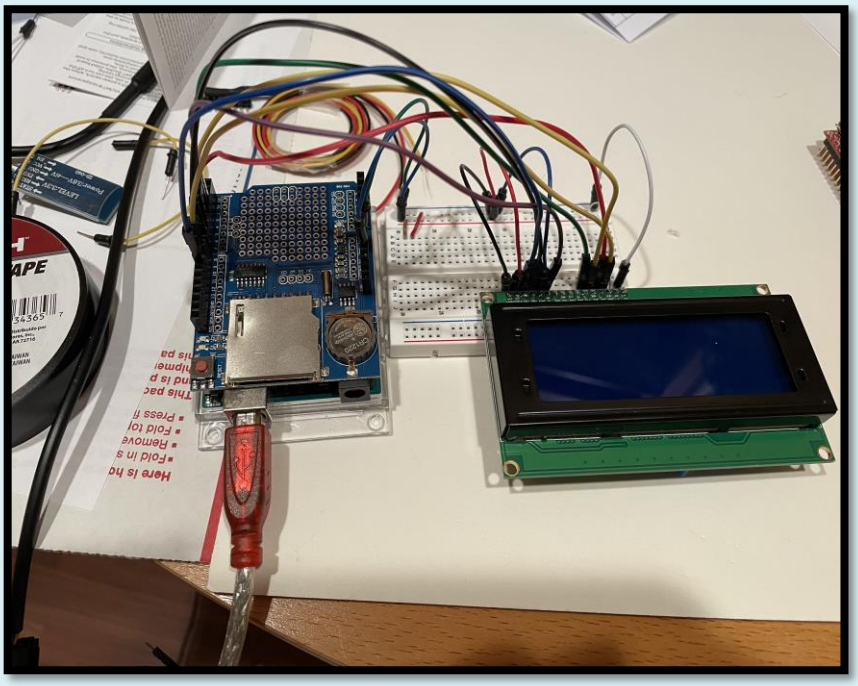
Image	Description
	Soldering the LCD
	• LCD (RIGHT) Soldering completed • Jumper wires connected for the Data Logging Shield to function (LEFT)

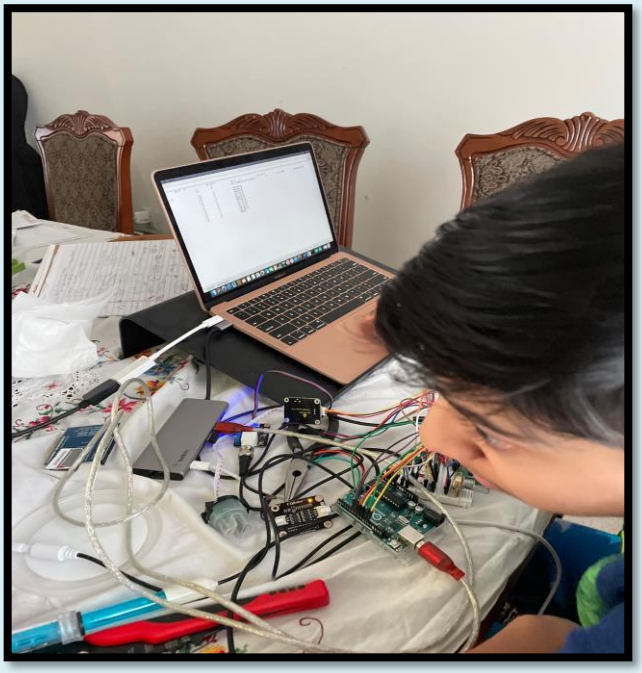
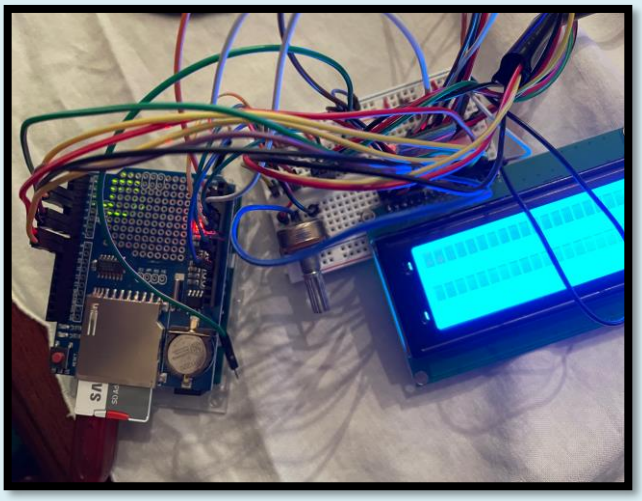
Image	Description
	When I started getting all the sensors ready
	Connecting Data-logging Shield Notice the SD Card

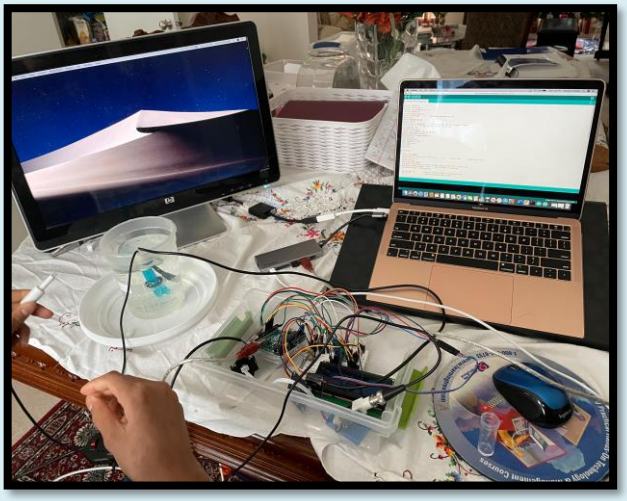
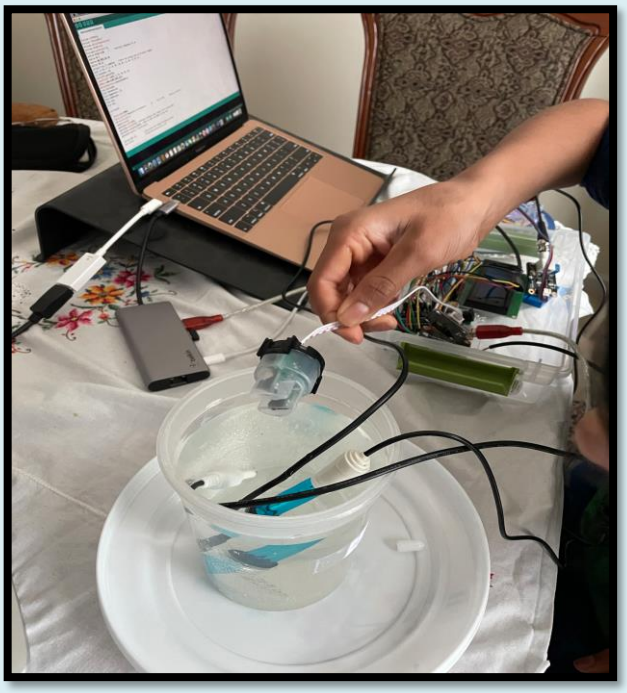
Image	Description
	Now I am connecting the Temperature sensor circuit
	Here is the Turbidity Sensor I am connecting to the Arduino. Ready, set go, testing!

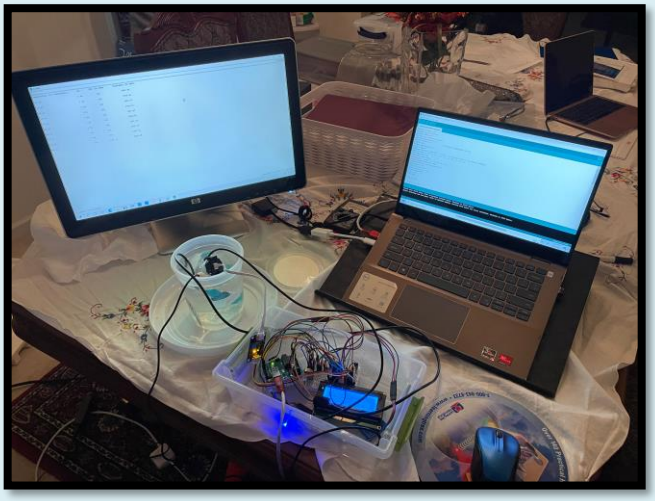
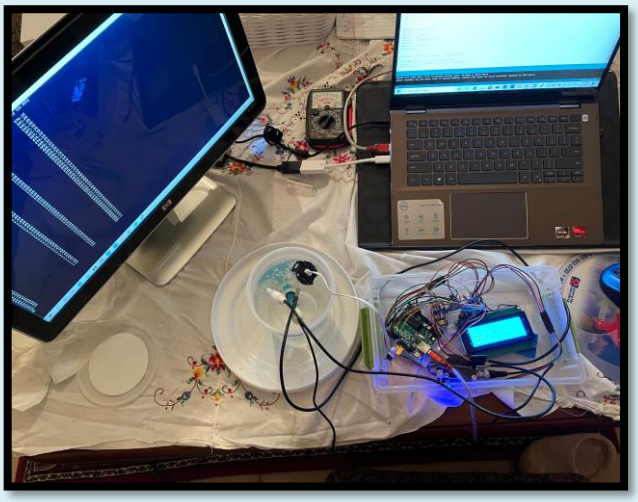
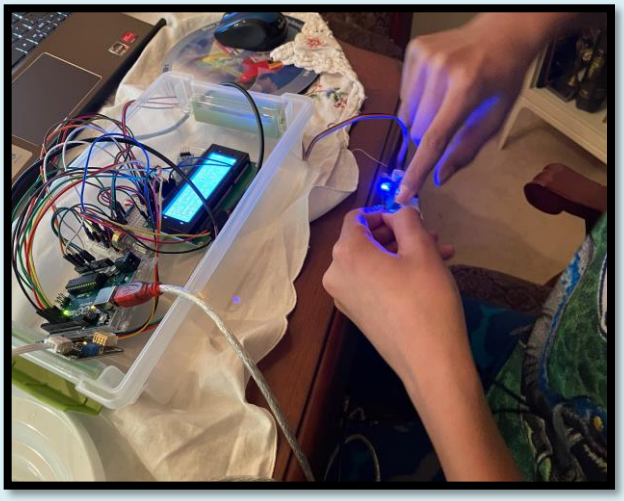
Image	Description
	All four sensors are ready to roll
	PuTty is showing the data on the screen for me to verify
	I had to re-calibrate the Turbidity sensor as I tested the 15 data samples each with 200 trials to preserve the accuracy and the performance of the sensor

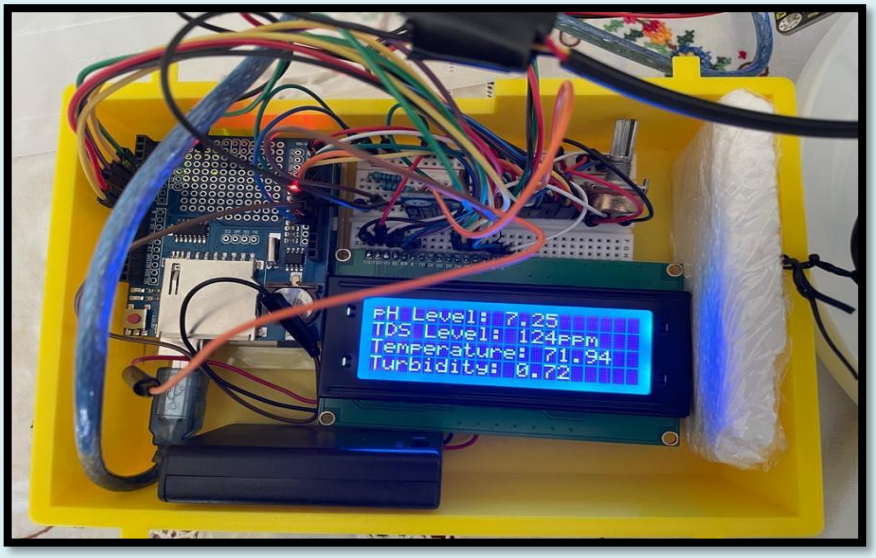
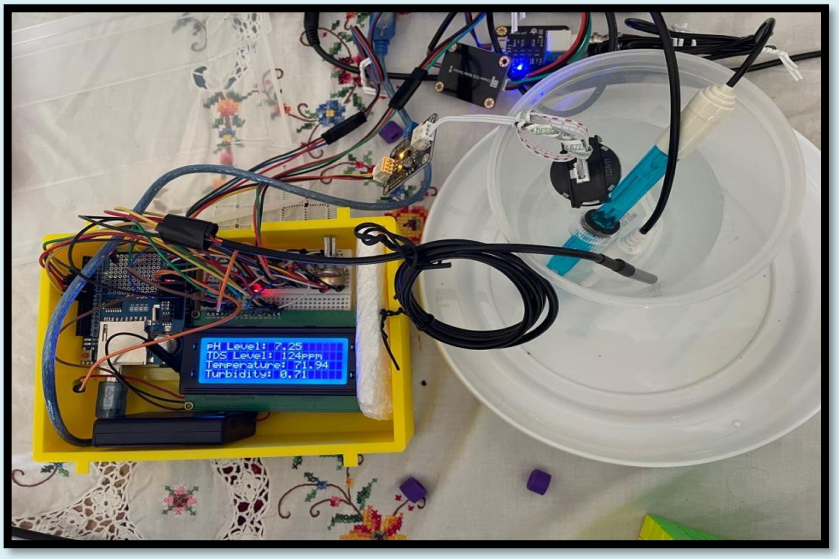
Image	Description
	Wiring the Circuit board
	Four sensors connected to Arduino ready to test Notice each trial displayed on the LCD display

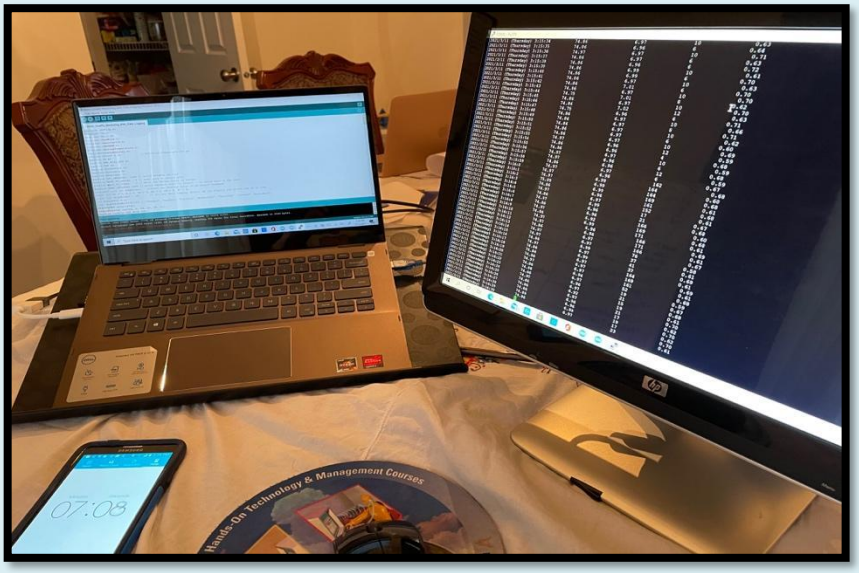
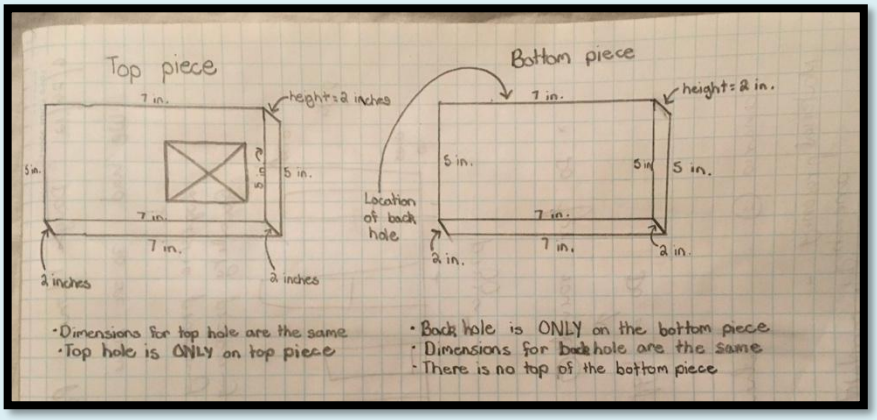
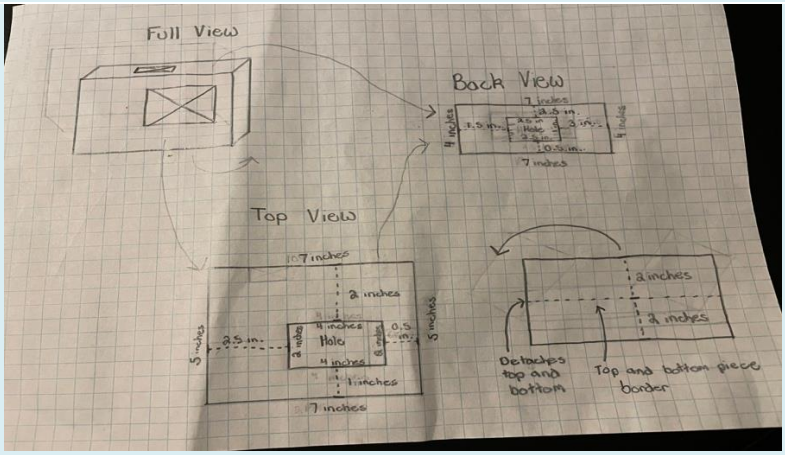
Image	Description
	Trials are sent on to PuTty (in the right-hand side of the screen)
	Blueprint of my 3D Box for "Potamoi" I sent to one of the high school teachers
	Blueprint of my 3D Box for "Potamoi" I sent to one of the high school teachers

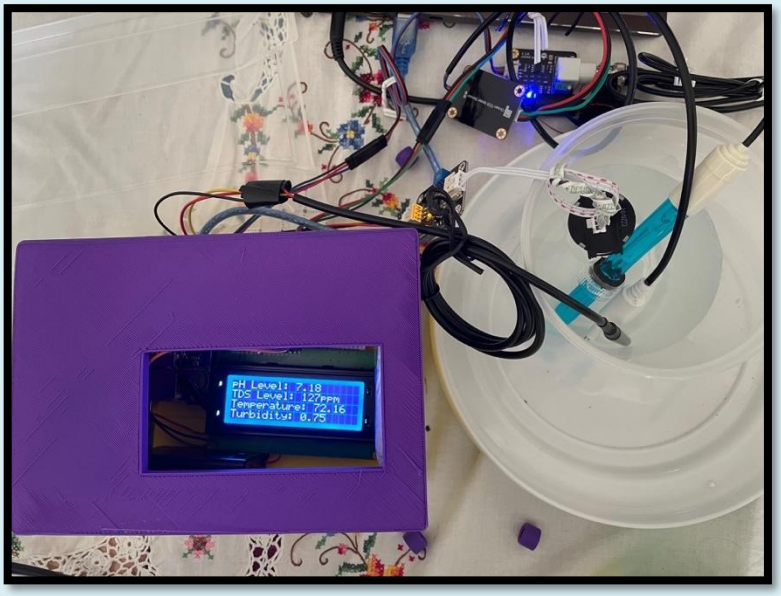
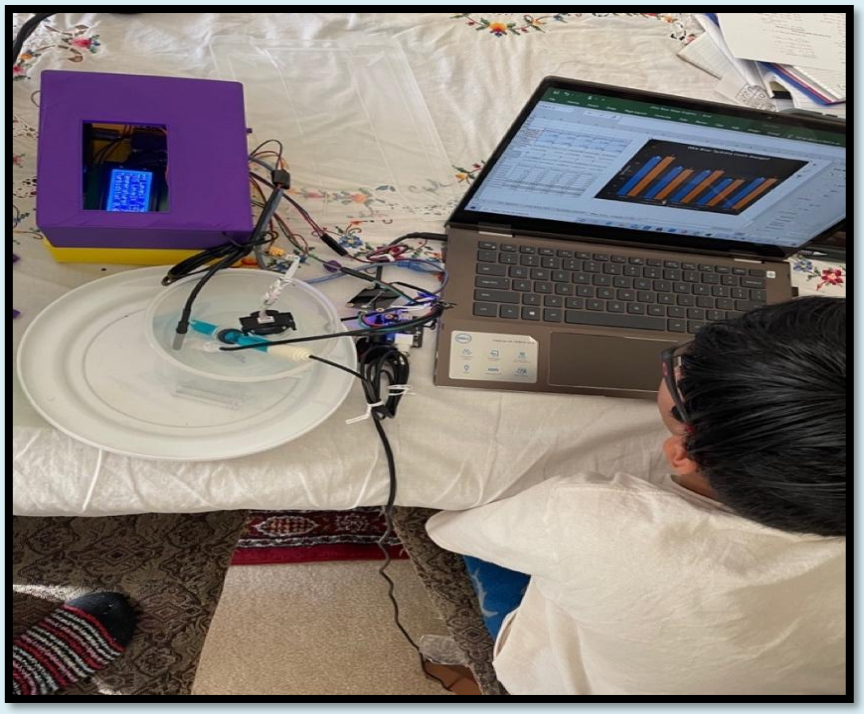
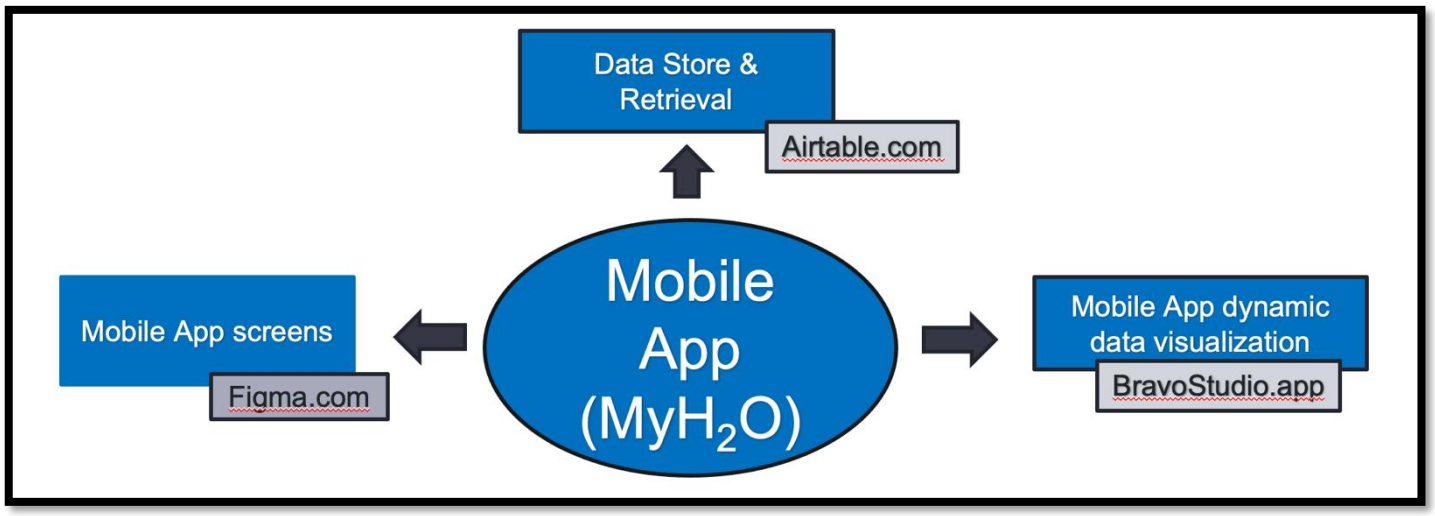
Image	Description
	3-D-printed box for “Potamoi” I had the box 3-D printed with a specific opening for my device, “Potamoi.”
	Box for my device, “Potamoi”

Image	Description
	I interpreted data in the form of graphs for each location in the three rivers.

4.3 Mobile App Development Steps

- I designed and prototyped the app in Figma
- I usedthe interactions tool in Figma to make the app more engaging as buttons became clickable
- I imported the data from the device into Airtable to communicate with Bravo Vision
- I bound the data to the app; the app's data could be updated real time from the table in Airtable
- I made the app become fully functional and usable to any user anywhere; the app can also be put onto the App Store or the Google Play Store

4.4 Mobile App Component Diagram



4.5 Step by step approach to Mobile App Development

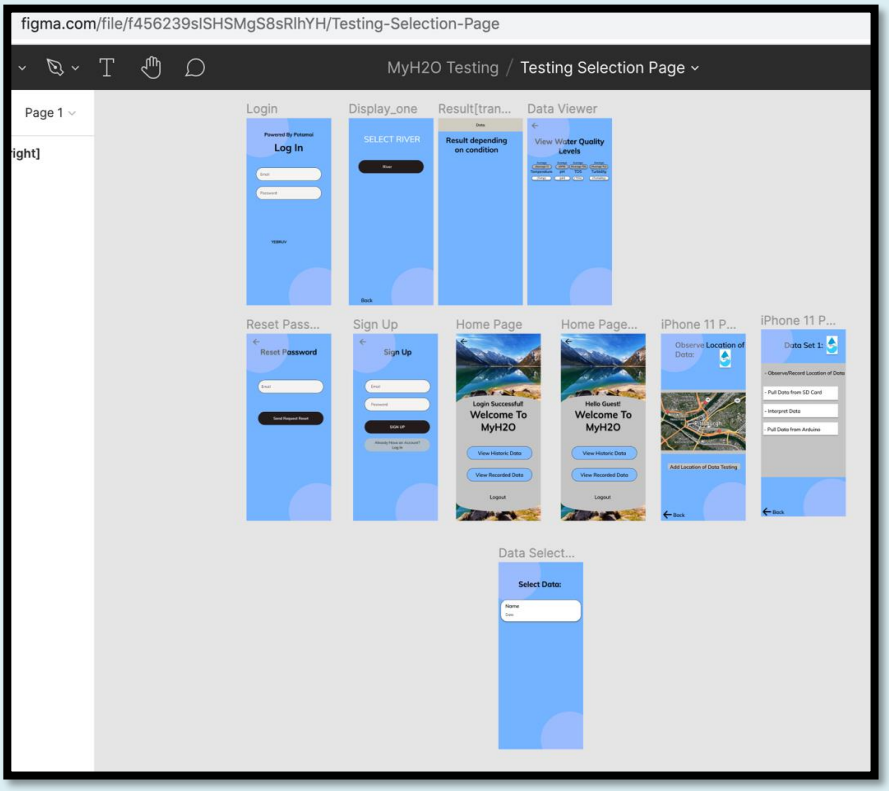
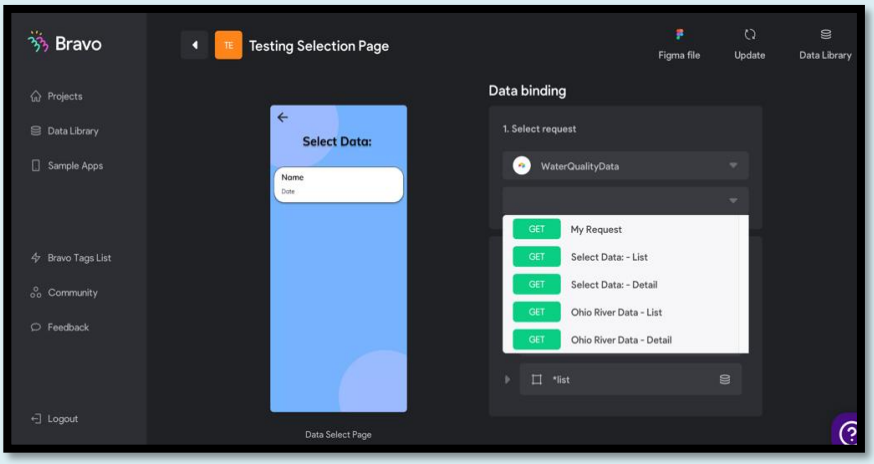
Image	Description
	**Figma design** The design was all done on browser-based tool called "Figma" • Here are the screens I designed to bind data from Airtable
	**Bravorizing my static Figma pages to get the River names** - On the static page I have only two variables called "Name" and "Date" but the data will populate when I run it on BravoStudio (I will show the demo Fair day)

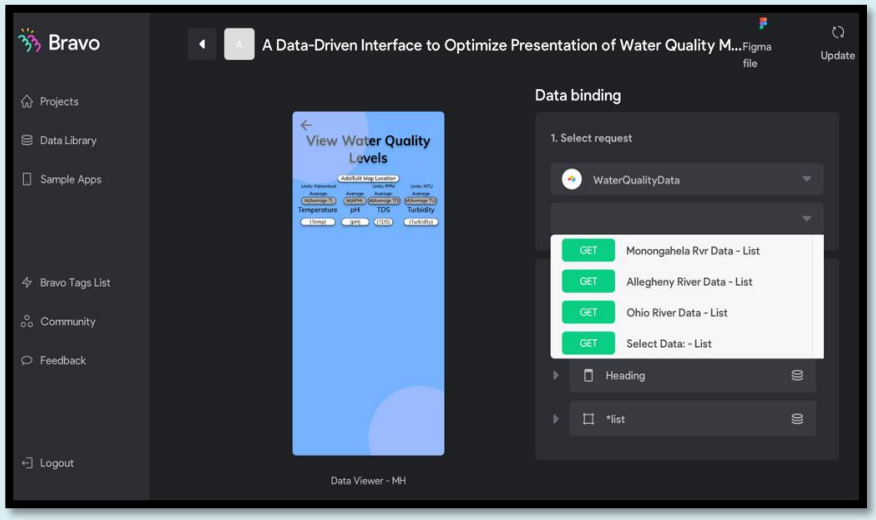
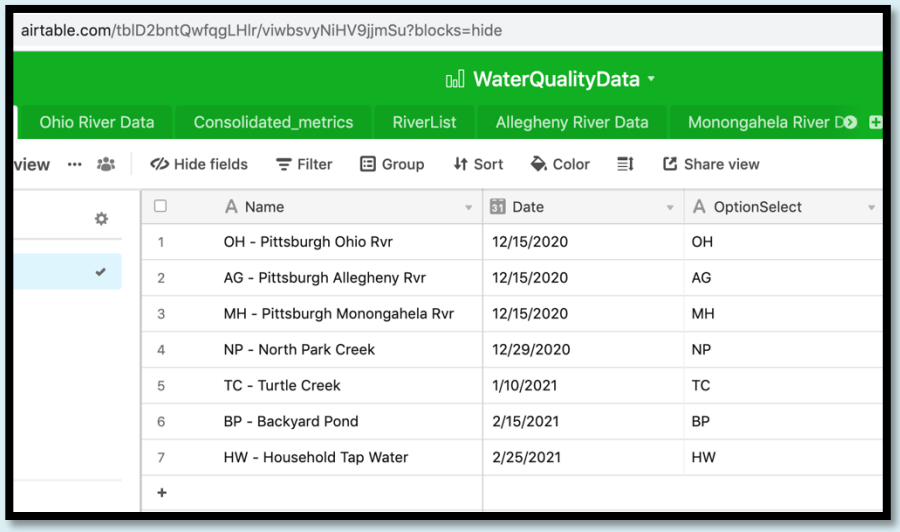
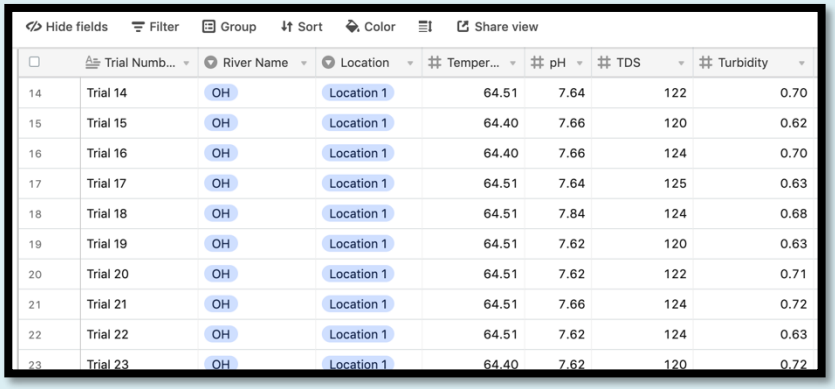
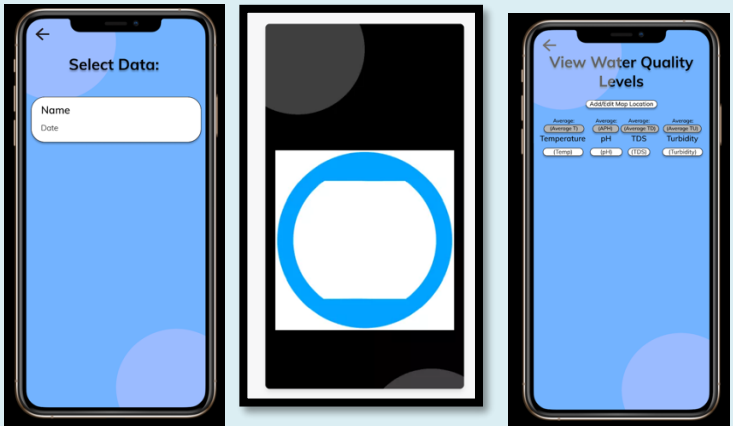
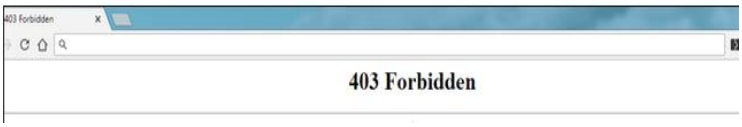
Image	Description
	**Bravorizing my static Figma pages with Arduino Data (Potamoi data)** The only drawback of this tool is that it does not allow you to display graphs - You can see how I "bound" the real data to the static Figma page using Bravo via the Get APIs listed - You will see all my 200 data trials on my mobile app (I will show the demo Fair day)
	Here is the Airtable where I display the River/Location name.
	This is the Airtable where my Water Quality Data gets transmitted from SD Card

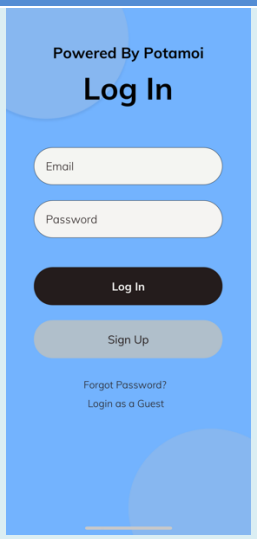
Image	Description
	The middle screen is the loading screen I have implemented into the design when the app is trying to populate the data (The white background will not be there when using the app) (I will show you on the Fair Day)

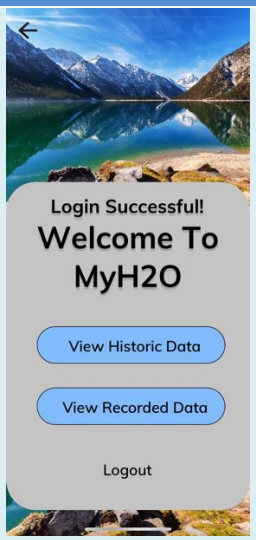
4.6 Errors and Debugging during the Mobile App development

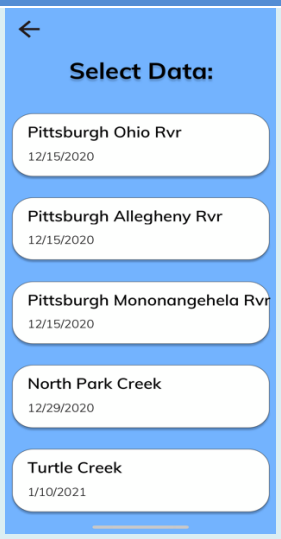
	While trying to update the data in Bravo Studio, I encountered an error like the one on the left that said I didn't have permission to access my project

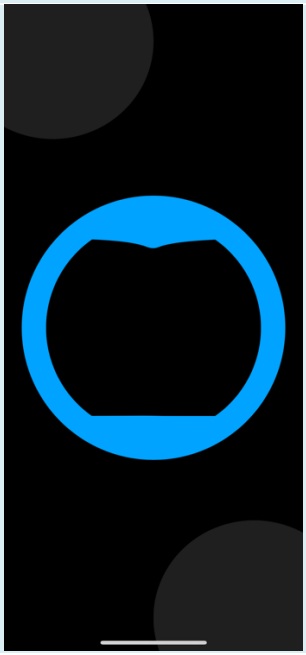
Mobile App during execution

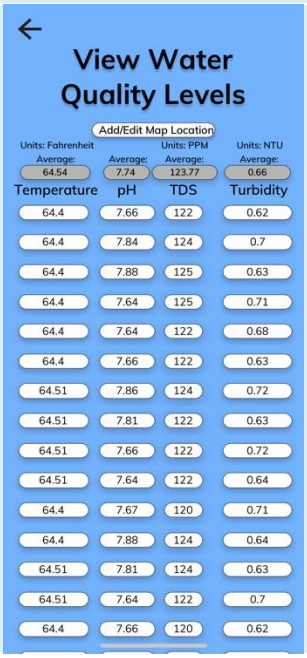
This is how the prior static pages displayed when I bound the data in my Airtable

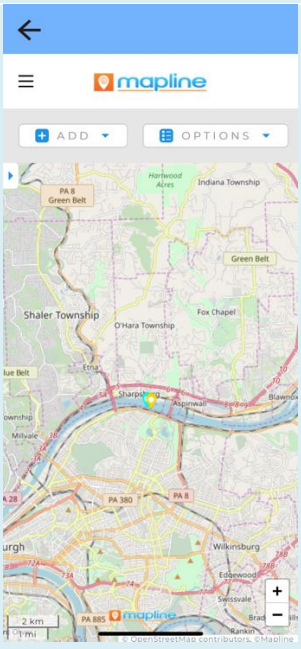






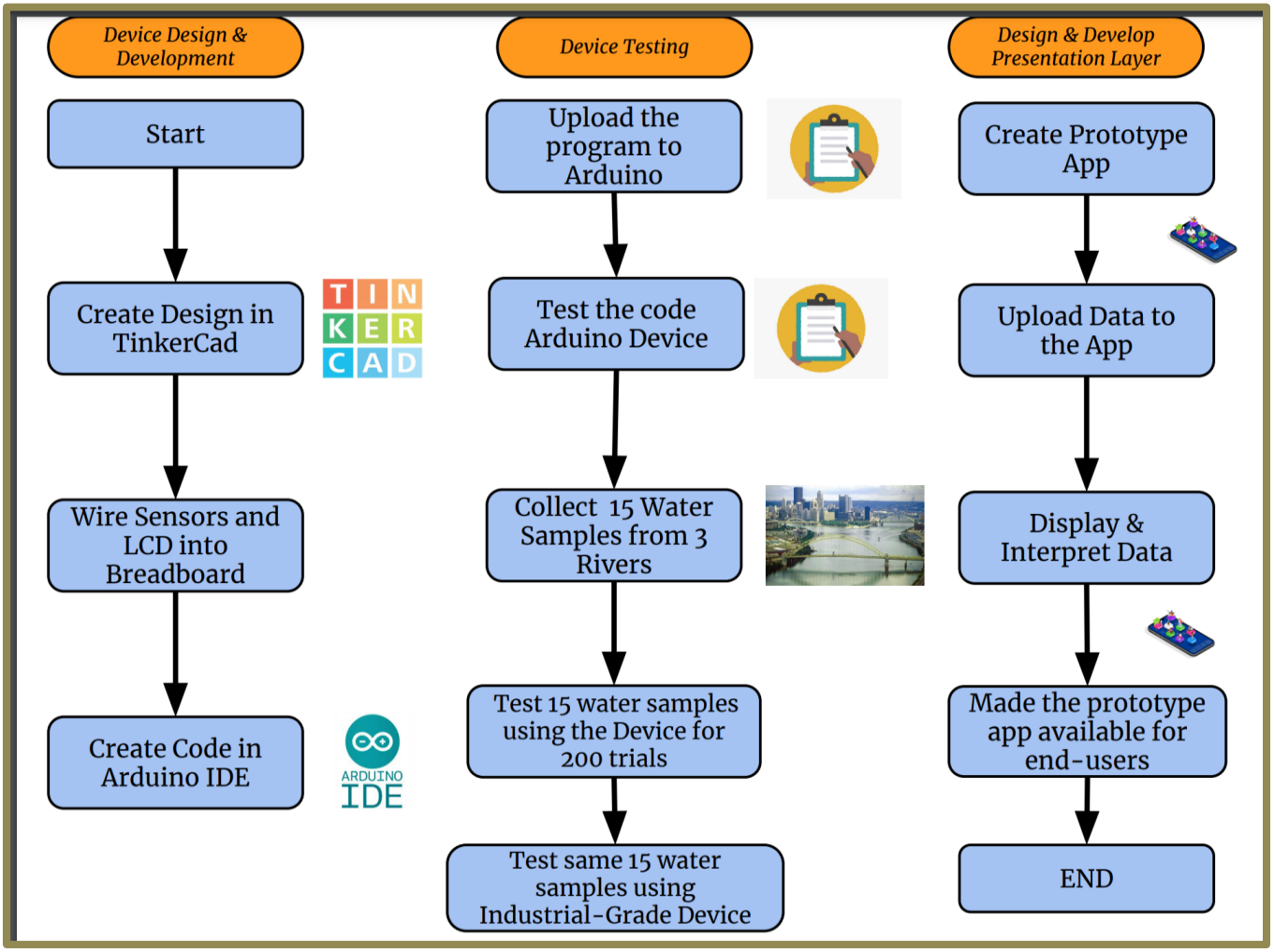






4.4. Flow Chart

Here is a Flow chart of my device design, test and presentation on mobile app



5. Data

5.1 Data Stream from Potamoi

Column 1 – Date and Time

Column 2 – Temperature reading (F)

Column 3 – pH

Column 4 – TDS (in PPM)

Column 5 – Turbidity (in NTU)

=~=~=~=~=~=~=~=~=~=~=~=~=~= PuTTY log 2021.03.11 18:20:26 =~=~=~=~=~=~=~=~=~=~=~=~=~=~=
Initializing SD card...

Date & Time	Temp (In Fahrenheit)	pH	TDS (In PPM)	Turbidity (In NTU)
2021/3/11 (Thursday) 6:20:2	74.86	6.94	4	0.72
2021/3/11 (Thursday) 6:20:3	74.86	6.92	10	0.72
2021/3/11 (Thursday) 6:20:4	74.75	6.96	4	0.66
2021/3/11 (Thursday) 6:20:5	74.86	6.94	10	0.72
2021/3/11 (Thursday) 6:20:6	74.75	6.94	10	0.67
2021/3/11 (Thursday) 6:20:7	74.75	6.92	4	0.73
2021/3/11 (Thursday) 6:20:8	74.75	6.96	10	0.66
2021/3/11 (Thursday) 6:20:9	74.75	6.96	4	0.73
2021/3/11 (Thursday) 6:20:10	74.75	6.94	10	0.68
2021/3/11 (Thursday) 6:20:11	74.75	6.96	6	0.68
2021/3/11 (Thursday) 6:20:12	74.86	6.94	12	0.74
2021/3/11 (Thursday) 6:20:13	74.75	6.94	4	0.68
2021/3/11 (Thursday) 6:20:14	74.86	6.92	4	0.72
2021/3/11 (Thursday) 6:20:15	74.86	6.94	10	0.66
2021/3/11 (Thursday) 6:20:16	74.86	6.92	4	0.72
2021/3/11 (Thursday) 6:20:17	74.86	6.92	10	0.66
2021/3/11 (Thursday) 6:20:18	74.86	6.94	4	0.72
2021/3/11 (Thursday) 6:20:19	74.86	6.92	10	0.74
2021/3/11 (Thursday) 6:20:20	74.86	6.94	6	0.68
2021/3/11 (Thursday) 6:20:21	74.86	6.94	8	0.73
2021/3/11 (Thursday) 6:20:22	74.86	6.94	10	0.68
2021/3/11 (Thursday) 6:20:23	74.86	6.94	4	0.74
2021/3/11 (Thursday) 6:20:24	74.97	6.92	10	0.68
2021/3/11 (Thursday) 6:20:25	74.86	6.94	4	0.74
2021/3/11 (Thursday) 6:20:26	74.86	6.94	10	0.66
2021/3/11 (Thursday) 6:20:27	74.86	6.96	6	0.66

5.2 Calculated Averages from Potamoi Data

I conducted 200 trials for each sample of water. Then, as the industrial-grade device had only 5 samples for each location, I took the average of every 40 samples (1-40, 41-80, 81-120, 121-160, 161-200) to have five samples even though I conducted 200 trials.

Sample ID	pH							Turbidity (NTU)						
	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Average	Std. Dev.	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Average	Std. Dev.
OH1	6.91	6.93	6.93	6.96	6.96	6.938	0.021679	0.98	0.99	1.03	0.96	0.98	0.988	0.02588
OH2	7.02	7.02	7.01	7.11	6.91	7.014	0.070922	0.83	0.83	0.82	0.87	0.77	0.824	0.035777
OH3	7.03	6.99	6.92	6.99	6.9	6.966	0.054129	0.85	0.86	0.84	0.81	0.79	0.83	0.029155
OH4	6.91	6.91	6.99	6.99	7.01	6.962	0.048166	0.72	0.71	0.7	0.69	0.72	0.708	0.01304
OH5	7	7.03	6.95	7	6.98	6.992	0.029496	0.81	0.78	0.83	0.85	0.87	0.828	0.034928
MH1	7.07	7.07	7.08	7.08	7.08	7.076	0.005477	0.74	0.71	0.73	0.73	0.76	0.734	0.0181659
MH2	6.8	6.8	6.75	6.75	6.74	6.768	0.029496	0.6	0.5	0.53	0.54	0.56	0.546	0.0371484
MH3	6.94	7	6.9	6.91	6.87	6.936	0.60249	0.4	0.41	0.42	0.45	0.38	0.412	0.0258844
MH4	6.98	6.99	6.9	6.97	6.98	6.968	0.039623	0.4	0.4	0.52	0.43	0.41	0.432	0.0506952
MH5	7	6.9	6.98	7.03	7.02	6.966	0.063087	0.51	0.6	0.52	0.51	0.53	0.534	0.037815
AH1	7.05	7.05	7.01	7.01	7.03	7.03	0.02	0.98	0.97	1.02	0.99	0.96	0.984	0.023022
AH2	7.05	7.01	7.04	7.03	7.03	7.032	0.014832	1.07	1.01	0.9	1.05	1.03	1.012	0.066483
AH3	7.04	7.04	7.06	7.05	7.05	7.048	0.008367	0.71	0.75	0.65	0.63	0.71	0.69	0.04899
AH4	7.12	7.07	7.07	7.08	7.08	7.084	0.020736	0.8	0.71	0.72	0.69	0.61	0.706	0.068044
AH5	7.1	7.1	7.14	7.13	7.13	7.12	0.018708	0.6	0.67	0.68	0.59	0.58	0.624	0.047223

Sample ID	TDS (PPM)							Temperature (Fahrenheit)						
	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Average	Std. Dev.	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Average	Std. Dev.
OH1	122.75	123.075	124	124.525	124.6216	123.7943	0.846855	64.44	64.5	64.53	64.62	64.64	64.54528	0.08273
OH2	135.75	135.95	136	135.75	135.946	135.888	0.11984	66.13	66.23	66.31	66.42	66.5	66.31784	0.149023
OH3	132.525	132.575	132.7	132.925	123.568	130.877	4.07869	64.48	64.63	64.73	64.84	65.14	64.76224	0.250888
OH4	122.7	123.1	123.05	125.65	125.73	124.03	1.50884	64.97	65.03	65.16	65.2	65.27	65.12749	0.125168
OH5	120.5	123.05	120.15	120.703	120.703	120.409	0.20726	66.96	67.07	67.13	67.2	67.2	67.11165	0.098877
MH1	160.95	161.4	161.75	161.55	161.846	161.499	0.3526	66.95	67.07	67.18	67.27	67.38	67.16978	0.165751
MH2	171.1	166.45	166.625	166.85	167.281	167.661	1.94734	67.44	67.61	67.64	67.72	67.75	67.63175	0.123072
MH3	183.25	182.95	182.925	183.05	183.077	183.05	0.12879	70.38	70.4	70.43	70.46	70.49	70.43512	0.043628
MH4	166.8	166.525	167.15	167.175	167.654	167.061	0.42656	68.9	68.99	69.08	69.19	69.23	69.07835	0.136749
MH5	187.25	186.875	186.75	186.775	186.846	186.899	0.20257	70.35	70.38	70.43	70.45	70.52	70.42723	0.062462
AH1	119.35	117.9	117.7	117.5	117.6757	118.0251	0.754077	66.2865	66.4055	66.52625	66.573	66.63778	66.48576	0.140183
AH2	126.025	126.65	126.725	126.975	127.7	126.815	0.605857	66.13125	66.16425	66.29475	66.36975	66.42586	66.27717	0.127508
AH3	129.5	130.7	131.625	131.675	131.5135	131.0027	0.928697	67.1715	67.2485	67.2595	67.31475	67.35222	67.26929	0.068984
AH4	135.25	132.65	132.5	132.6	132.2	133.04	1.247698	69.10375	69.18025	69.273	69.31975	56.68	66.71135	5.608316
AH5	121.45	117.05	117.05	117.05	117.1351	117.947	1.958568	67.32125	67.383	67.45	67.517	67.58422	67.45109	0.10436

5.3 Industry Grade Device Data for Comparison

Sample Id	pH					Turbidity (NTU)					TDS (mg/L)				
	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
OH1	6.95	6.96	6.96	6.96	6.96	1	0.82	1.02	1.25	1.08	137.6	133.6	123.6	118.4	105.6
OH2	6.99	7	7	7	7.01	1.1	1.01	0.75	0.49	0.69	136.8	81.6	124.0	120.0	113.2
OH3	6.99	6.99	6.99	6.99	7	1	0.76	0.97	0.78	0.87	139.6	117.6	119.2	118.4	120.0
OH4	6.96	6.97	6.97	6.98	6.98	0.8	0.6	0.75	0.69	0.84	114.0	118.4	89.6	116.0	124.4
OH5	6.99	6.99	6.99	6.99	7	0.9	0.85	0.86	0.75	0.94	121.2	114.4	100.4	106.0	117.2
MH1	7.13	7.1	7.07	7.05	7.06	0.77	0.7	0.81	0.73	0.7	228.0	204.0	211.2	214.8	190.8
MH2	6.75	6.77	6.78	6.78	6.8	0.51	0.59	0.54	0.53	0.52	168.0	191.6	204.4	184.8	191.2
MH3	6.82	6.84	6.92	6.93	6.94	0.47	0.42	0.44	0.49	0.44	194.4	201.6	192.8	198.8	208.4
MH4	6.94	6.96	6.96	6.98	6.98	0.45	0.45	0.43	0.38	0.38	214.0	188.0	188.8	188.0	208.4
MH5	6.98	6.99	6.99	7	7	0.47	0.46	0.65	0.54	0.46	201.2	203.2	208.8	194.8	186.0
AH1	7.02	7.02	7.02	7.02	7.03	1.2	1.06	1.09	1.12	1.11	169.2	117.6	183.2	187.2	133.2
AH2	7.03	7.04	7.05	7.05	7.06	1	1	1	1.04	1.06	128.0	138.4	133.6	138.4	132.4
AH3	7.03	7.04	7.05	7.05	7.05	0.73	0.6	0.63	0.74	0.71	154.4	117.2	102.8	104.8	137.6
AH4	7.06	7.07	7.07	7.08	7.09	0.61	0.71	0.61	0.72	0.57	161.6	98.0	139.2	141.6	153.2
AH5	7.08	7.09	7.09	7.1	7.1	0.57	0.61	0.62	0.59	0.63	138.0	130.4	140.8	133.2	127.6

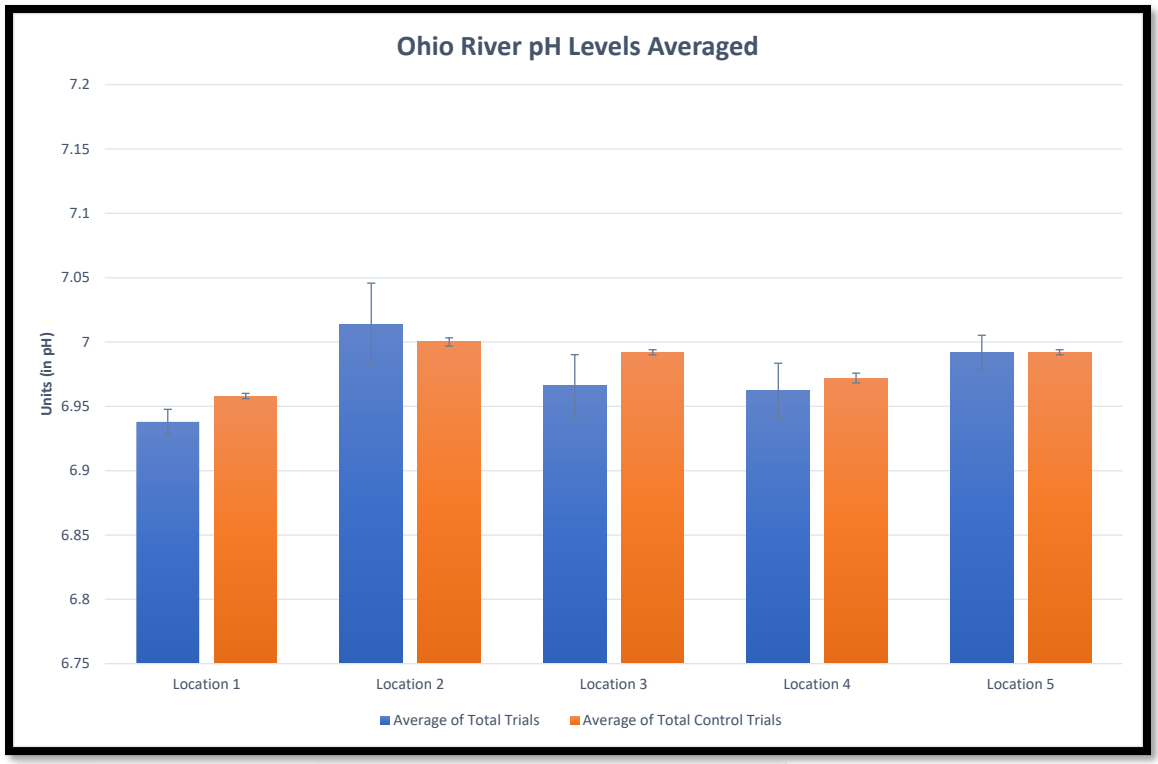
6. Statistical Analysis

6.1 Explanation of Two Factor ANOVA and Standard Error Bars

- In this project, I used a Two Factor ANOVA and Standard Error bars to show if there was variation between my device and the industrial grade device
- A Two Factor ANOVA compares the mean difference between two or more groups (in my case, my device vs. industrial grade device) that are split between two different independent variables (in my case, the location and the type of group)
- The 3 main values that an ANOVA calculates are the p-value, the f-value, and the f critical value
- If the p-value is less than 0.05, and the f-value is greater than the f critical, then there was significant variation between the groups
- Standard Error bars are another way of showing significant variation on a graph
- If two error bars from each group overlap, then there was significant variation between those two groups

6.2 Ohio River Water Quality Metrics

Two- Factor ANOVA and Graph for OHIO River pH Levels



Anova: Two-Factor With Replication

SUMMARY	Location 1	Location 2	Location 3	Location 4	Location 5	Total
Control						
Count	5	5	5	5	5	25
Sum	34.79	35	34.96	34.86	34.96	174.57
Average	6.958	7	6.992	6.972	6.992	6.9828
Variance	2E-05	5E-05	2E-05	7E-05	2E-05	0.000279
Treatment						
Count	5	5	5	5	5	25
Sum	34.69	35.07	34.83	34.81	34.96	174.36
Average	6.938	7.014	6.966	6.962	6.992	6.9744
Variance	0.00047	0.00503	0.00293	0.00232	0.00087	0.002651
Total						
Count	10	10	10	10	10	
Sum	69.48	70.07	69.79	69.67	69.92	
Average	6.948	7.007	6.979	6.967	6.992	
Variance	0.000329	0.002312	0.001499	0.00109	0.000396	
ANOVA						
ce of Varia	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	0.000882	1	0.000882	0.747458	0.392438	4.084746
Columns	0.020572	4	0.005143	4.358475	0.005097	2.605975
Interaction	0.002548	4	0.000637	0.539831	0.70732	2.605975
Within	0.0472	40	0.00118			
Total	0.071202	49				

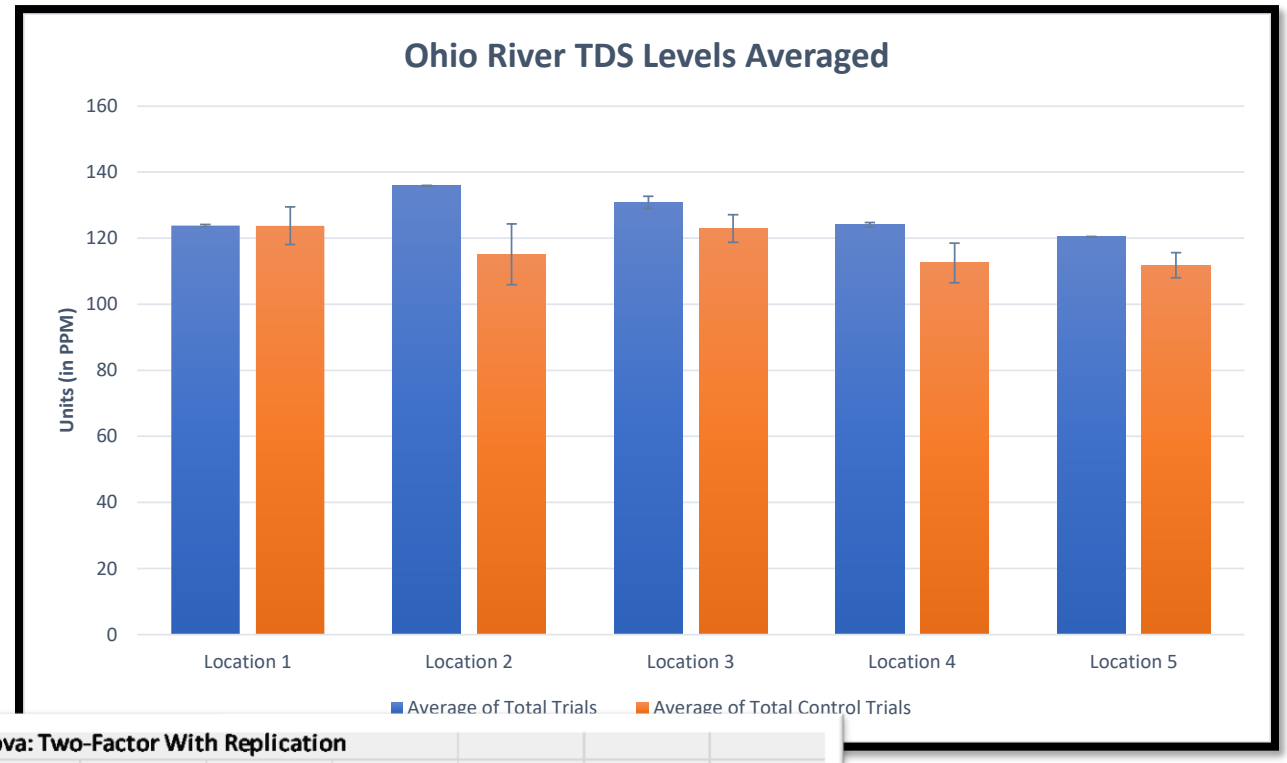
BLUE bar - Using "Potamoi" my Arduino device

ORANGE bar - Using Industrial-Grade (Control)

- The x-axis represents each location (there are two bars for each location because I tested my device and the industrial grade device) and the y-axis represents the units of measurement for each parameter

- Since the p-value is greater than 0.05 and the f-crit is greater than the f-value, there was not significant variation between my device's and the industrial grade device's data

Two Factor ANOVA and Graph for Ohio River TDS Levels



Anova: Two-Factor With Replication

SUMMARY	Location 1	Location 2	Location 3	Location 4	Location 5	Total
Control						
Count	5	5	5	5	5	25
Sum	618.8	575.6	614.8	562.4	559.2	2930.8
Average	123.76	115.12	122.96	112.48	111.84	117.232
Variance	161.728	424.992	87.328	178.832	71.968	181.5456
Treatment						
Count	5	5	5	5	5	25
Sum	618.9716	679.3959	654.2926	620.2297	602.2777	3175.168
Average	123.7943	135.8792	130.8585	124.0459	120.4555	127.0067
Variance	0.717163	0.014361	16.63571	2.276602	0.042958	35.6896
Total						
Count	10	10	10	10	10	
Sum	1237.772	1254.996	1269.093	1182.63	1161.478	
Average	123.7772	125.4996	126.9093	118.263	116.1478	
Variance	72.19818	308.5984	63.53568	117.6514	52.62363	

ANOVA

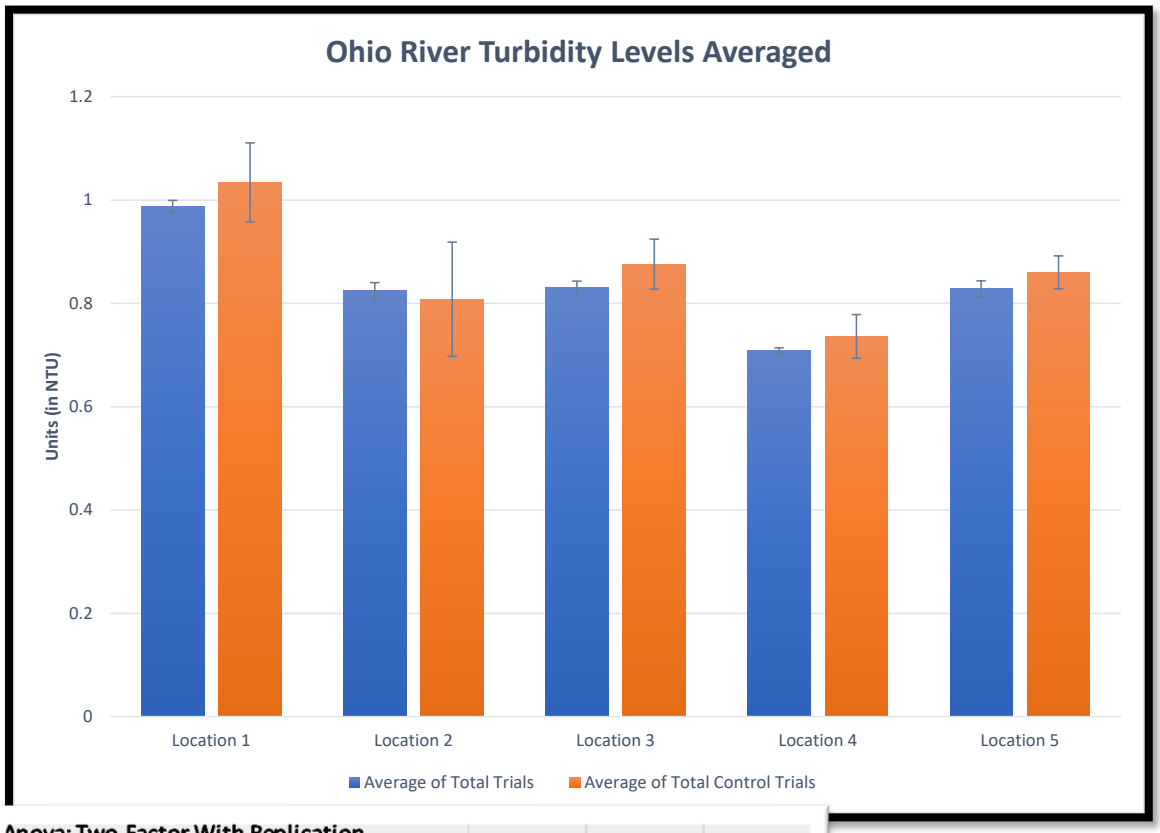
ce of Varia	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	1194.31	1	1194.31	12.64443	0.000986	4.084746
Columns	876.4901	4	219.1225	2.319899	0.073455	2.605975
Interaction	559.0155	4	139.7539	1.479605	0.226504	2.605975
Within	3778.139	40	94.45348			
Total	6407.955	49				

BLUE bar - Using "Potamoi" my Arduino device
ORANGE bar - Using Industrial-Grade (Control)

The x-axis represents each location (there are two bars for each location because I tested my device and the industrial grade device) and the y-axis represents the units of measurement for each parameter

- Since the p-value is less than 0.05 and the f-value is greater than the f-crit, there was variation between my device's and the industrial grade device's data

Two Factor ANOVA and Graph for Ohio River Turbidity Levels



Anova: Two-Factor With Replication

SUMMARY	Location 1	Location 2	Location 3	Location 4	Location 5	Total
Control						
Count	5	5	5	5	5	25
Sum	5.35	4.04	4.38	3.68	4.3	21.75
Average	1.07	0.808	0.876	0.736	0.86	0.87
Variance	0.0292	0.06112	0.01173	0.00893	0.00505	0.032242
Treatment						
Count	5	5	5	5	5	25
Sum	4.94	4.12	4.15	3.54	4.14	20.89
Average	0.988	0.824	0.83	0.708	0.828	0.8356
Variance	0.00067	0.00128	0.00085	0.00017	0.00122	0.008976
Total						
Count	10	10	10	10	10	
Sum	10.29	8.16	8.53	7.22	8.44	
Average	1.029	0.816	0.853	0.722	0.844	
Variance	0.015143	0.027804	0.006179	0.004262	0.003071	
ANOVA						
ce of Varia	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	0.014792	1	0.014792	1.230411	0.273952	4.084746
Columns	0.495868	4	0.123967	10.31168	7.81E-06	2.605975
Interaction	0.012468	4	0.003117	0.259275	0.90226	2.605975
Within	0.48088	40	0.012022			
Total	1.004008	49				

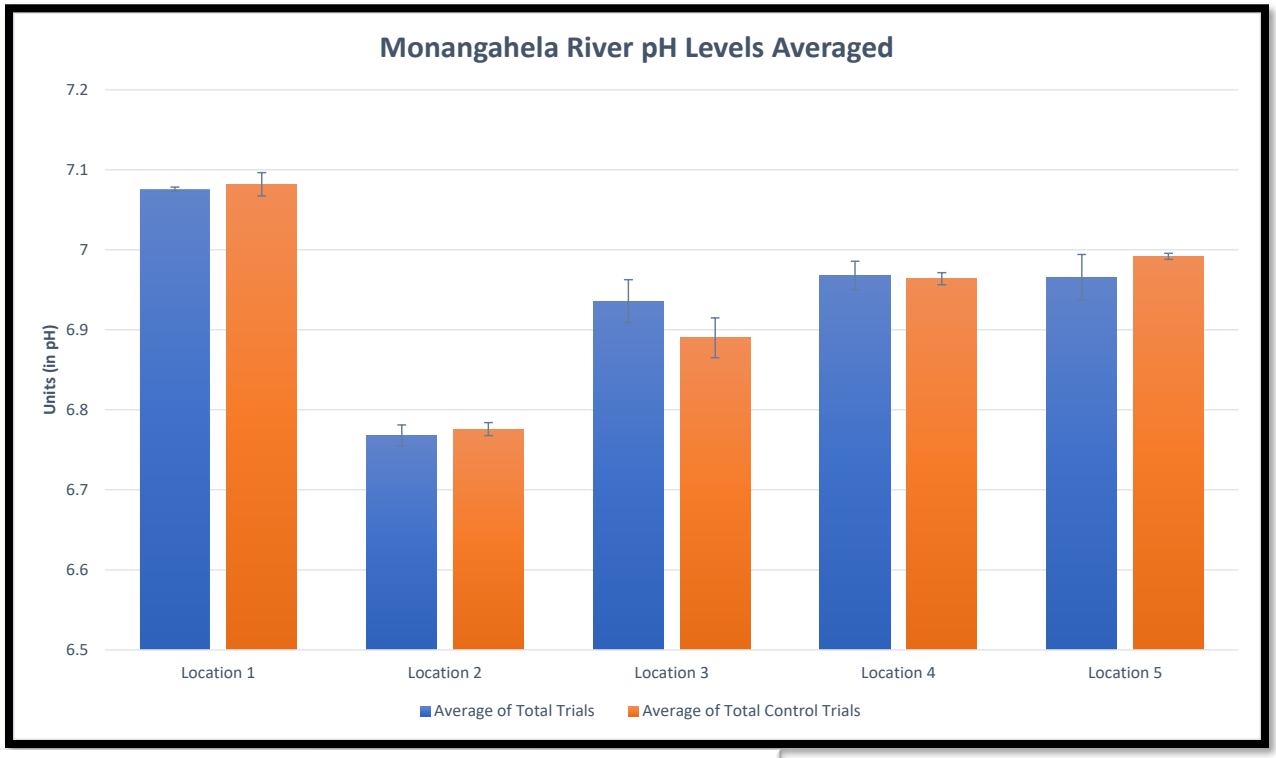
BLUE bar - Using "Potamoi" my Arduino device
ORANGE bar - Using Industrial-Grade (Control)

- The x-axis represents each location (there are two bars for each location because I tested my device and the industrial grade device) and the y-axis represents the units of measurement for each parameter

- Since the p-value is greater than 0.05 and the f-crit is greater than the f-value, there was not significant variation between my device's and the industrial grade device's data

6.3 Monongahela River Water Quality Metrics

Two- Factor ANOVA for Monongahela River pH Levels



Anova: Two-Factor With Replication

SUMMARY	Location 1	Location 2	Location 3	Location 4	Location 5	Total
Control						
Count	5	5	5	5	5	25
Sum	35.41	33.88	34.45	34.82	34.96	173.52
Average	7.082	6.776	6.89	6.964	6.992	6.9408
Variance	0.00107	0.00033	0.0031	0.00028	7E-05	0.011816
Treatment						
Count	5	5	5	5	5	25
Sum	35.38	33.84	34.68	34.84	34.83	173.57
Average	7.076	6.768	6.936	6.968	6.966	6.9428
Variance	3E-05	0.00087	0.00363	0.00157	0.00398	0.011996
Total						
Count	10	10	10	10	10	
Sum	70.79	67.72	69.13	69.66	69.79	
Average	7.079	6.772	6.913	6.966	6.979	
Variance	0.000499	0.000551	0.003579	0.000827	0.001988	

ANOVA						
ce of Varia	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	5E-05	1	5E-05	0.03349	0.855721	4.084746
Columns	0.504548	4	0.126137	84.4856	5.87E-19	2.605975
Interaction	0.00722	4	0.001805	1.208975	0.32209	2.605975
Within	0.05972	40	0.001493			
Total	0.571538	49				

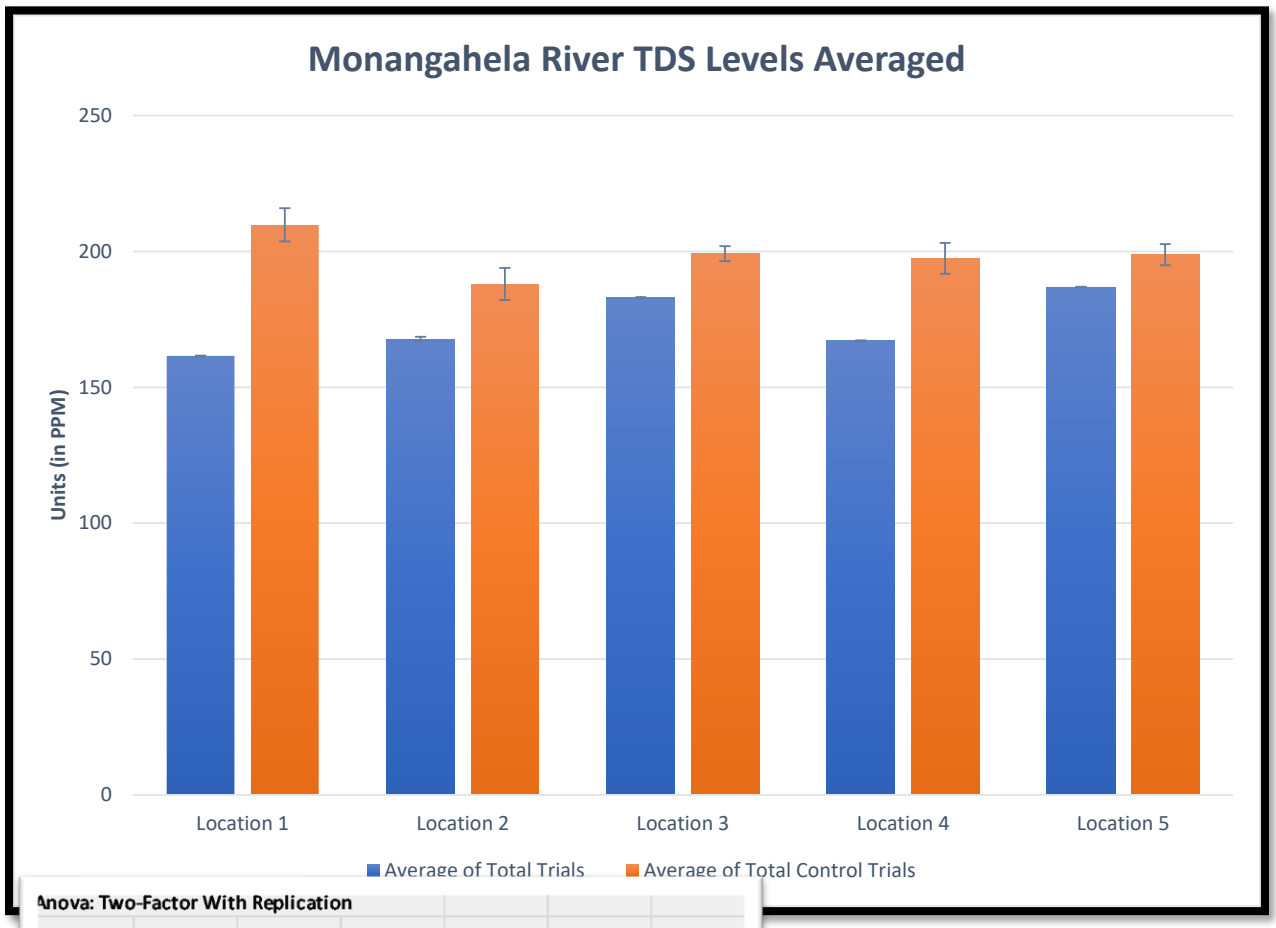
BLUE bar - Using "Potamoi" my Arduino device

ORANGE bar - Using Industrial-Grade (Control)

- The x-axis represents each location (there are two bars for each location because I tested my device and the industrial grade device) and the y-axis represents the units of measurement for each parameter

- Since the p-value is greater than 0.05 and the f-crit is greater than the f-value, there was not significant variation between my device's and the industrial grade device's data

Two- Factor ANOVA for Monongahela River TDS Levels



Anova: Two-Factor With Replication

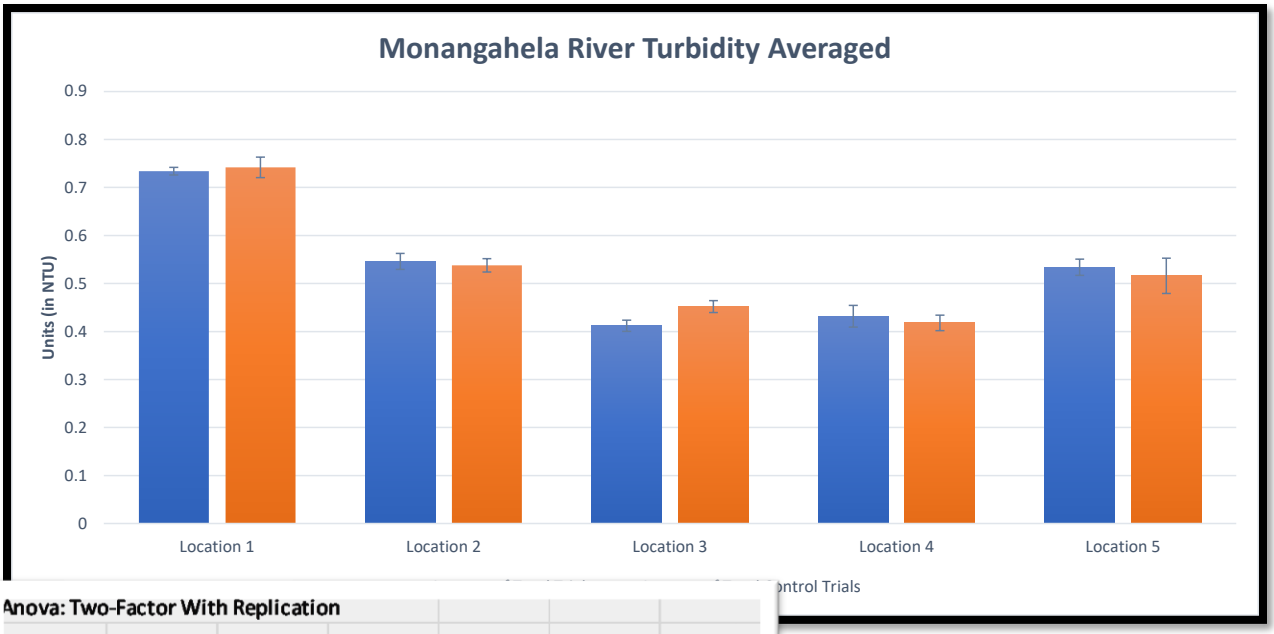
SUMMARY	Location 1	Location 2	Location 3	Location 4	Location 5	Total
Control						
Count	5	5	5	5	5	25
Sum	1048.8	940	996	987.2	994	4966
Average	209.76	188	199.2	197.44	198.8	198.64
Variance	188.208	175.6	38.64	161.808	76.24	156.4667
Treatment						
Count	5	5	5	5	5	25
Sum	807.4962	838.3063	915.2519	835.3038	934.4962	4330.854
Average	161.4992	167.6613	183.0504	167.0608	186.8992	173.2342
Variance	0.12433	3.792117	0.016587	0.18195	0.041037	102.7696
Total						
Count	10	10	10	10	10	
Sum	1856.296	1778.306	1911.252	1822.504	1928.496	
Average	185.6296	177.8306	191.1252	182.2504	192.8496	
Variance	730.676	194.6367	89.62795	328.356	73.24388	
ANOVA						
ce of Varia	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	8068.201	1	8068.201	125.1559	6.89E-14	4.08474
Columns	1541.005	4	385.2513	5.976113	0.000723	2.60597
Interaction	2102.056	4	525.514	8.151902	6.62E-05	2.60597
Within	2578.608	40	64.4652			
Total	14289.87	49				

BLUE bar - Using "Potamoi" my Arduino device
ORANGE bar - Using Industrial-Grade (Control)

- The x-axis represents each location (there are two bars for each location because I tested my device and the industrial grade device) and the y-axis represents the units of measurement for each

- Since the p-value is less than 0.05 and the f-value is greater than the f-crit, there was variation between my device's and the industrial grade device's data

Two- Factor ANOVA and Graph for Monongahela River Turbidity Levels



Anova: Two-Factor With Replication

SUMMARY	Location 1	Location 2	Location 3	Location 4	Location 5	Total
Control						
Count	5	5	5	5	5	25
Sum	3.71	2.69	2.26	2.09	2.58	13.33
Average	0.742	0.538	0.452	0.418	0.516	0.5332
Variance	0.00227	0.00097	0.00077	0.00127	0.00673	0.015289
Treatment						
Count	5	5	5	5	5	25
Sum	3.67	2.73	2.06	2.16	2.67	13.29
Average	0.734	0.546	0.412	0.432	0.534	0.5316
Variance	0.00033	0.00138	0.00067	0.00257	0.00143	0.014689
Total						
Count	10	10	10	10	10	
Sum	7.38	5.42	4.32	4.25	5.25	
Average	0.738	0.542	0.432	0.425	0.525	
Variance	0.001173	0.001062	0.001084	0.001761	0.003717	
ANOVA						
ce of Varia	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	3.2E-05	1	3.2E-05	0.017401	0.895715	4.084746
Columns	0.640332	4	0.160083	87.04894	3.45E-19	2.605975
Interaction	0.005588	4	0.001397	0.759652	0.557724	2.605975
Within	0.07356	40	0.001839			
Total	0.719512	49				

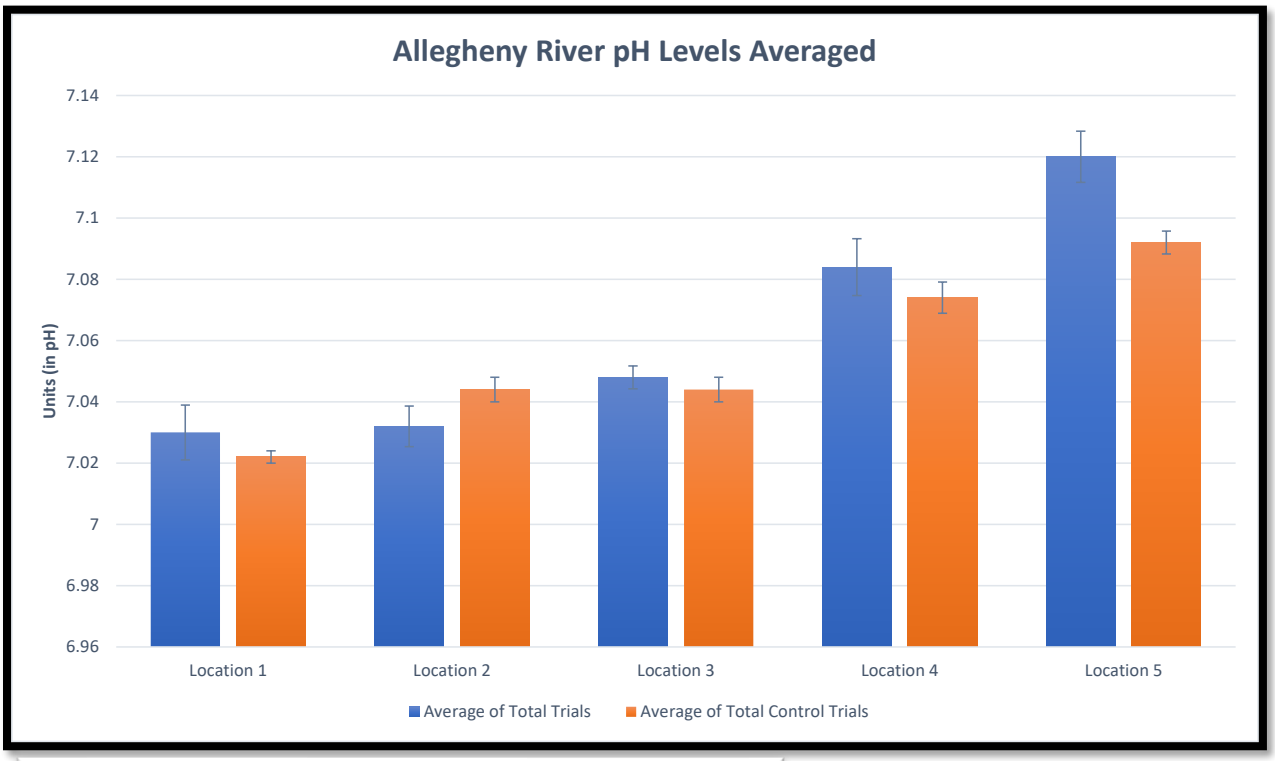
BLUE bar - Using "Potamoi" my Arduino device
ORANGE bar - Using Industrial-Grade (Control)

- The x-axis represents each location (there are two bars for each location because I tested my device and the industrial grade device) and the y-axis represents the units of measurement for each

- Since the p-value is greater than 0.05 and the f-crit is greater than the f-value, there was not significant variation between my device's and the industrial grade device's data

6.4 Allegheny River Water Quality Metrics

Two- Factor ANOVA and Graph for Allegheny River pH Levels



Anova: Two-Factor With Replication

SUMMARY	Location 1	Location 2	Location 3	Location 4	Location 5	Total
Control						
Count	5	5	5	5	5	25
Sum	35.11	35.22	35.22	35.37	35.46	176.38
Average	7.022	7.044	7.044	7.074	7.092	7.0552
Variance	2E-05	8E-05	8E-05	0.00013	7E-05	0.000701
Treatment						
Count	5	5	5	5	5	25
Sum	35.15	35.16	35.24	35.42	35.6	176.57
Average	7.03	7.032	7.048	7.084	7.12	7.0628
Variance	0.0004	0.00022	7E-05	0.00043	0.00035	0.001488
Total						
Count	10	10	10	10	10	
Sum	70.26	70.38	70.46	70.79	71.06	
Average	7.026	7.038	7.046	7.079	7.106	
Variance	0.000204	0.000173	7.11E-05	0.000277	0.000404	

ANOVA						
ce of Varia	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	0.000722	1	0.000722	3.902703	0.055136	4.084746
Columns	0.04308	4	0.01077	58.21622	3.79E-16	2.605975
Interaction	0.002048	4	0.000512	2.767568	0.040271	2.605975
Within	0.0074	40	0.000185			
Total	0.05325	49				

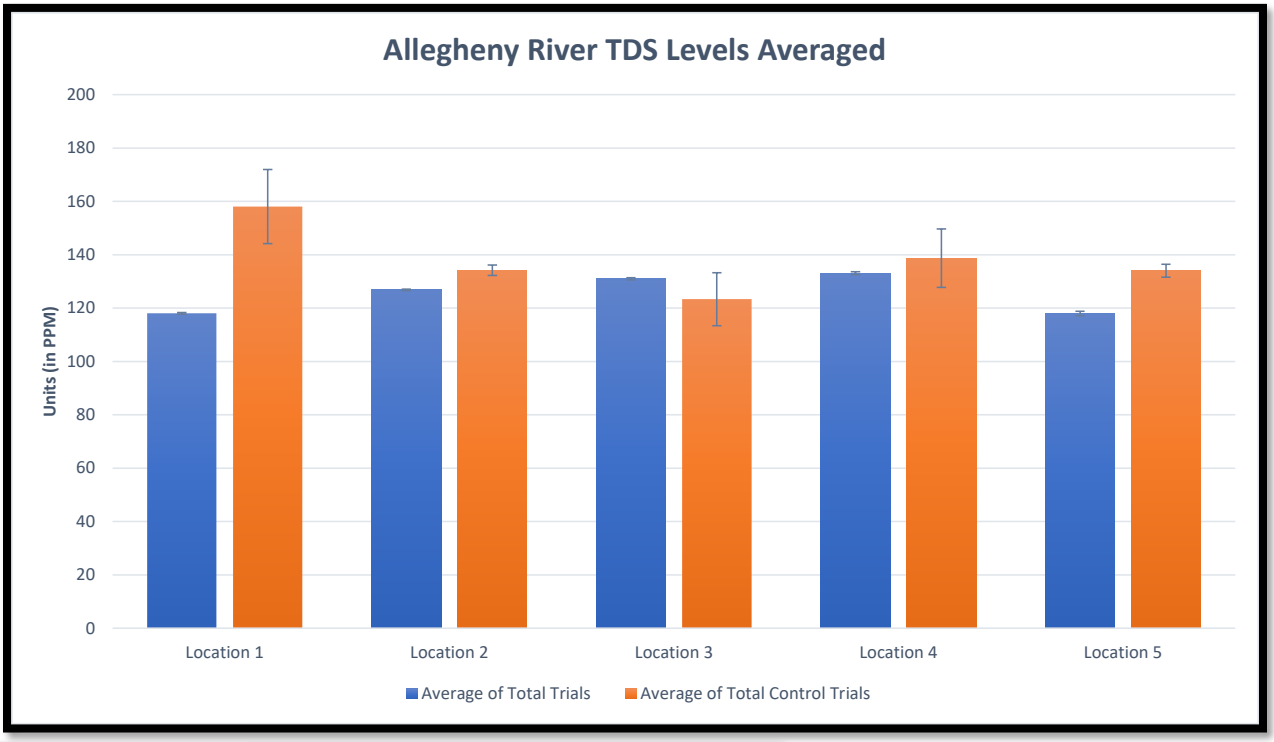
BLUE bar - Using "Potamoi" my Arduino device

ORANGE bar - Using Industrial-Grade (Control)

- The x-axis represents each location (there are two bars for each location because I tested my device and the industrial grade device) and the y-axis represents the units of measurement for each

- Since the p-value is less than 0.05 and the f-value is greater than the f-crit, there was variation between my device's and the industrial grade device's data

Two- Factor ANOVA and Graph for Allegheny River TDS Levels



Anova: Two-Factor With Replication

SUMMARY	Location 1	Location 2	Location 3	Location 4	Location 5	Total
Control						
Count	5	5	5	5	5	25
Sum	790.4	670.8	616.8	693.6	670	3441.6
Average	158.08	134.16	123.36	138.72	134	137.664
Variance	965.072	19.328	492.848	599.952	29.2	486.1157
Treatment						
Count	5	5	5	5	5	25
Sum	590.1257	634.075	655.0135	665.2	589.7351	3134.149
Average	118.0251	126.815	131.0027	133.04	117.947	125.366
Variance	0.568632	0.367062	0.862477	1.55675	3.83599	43.2176
Total						
Count	10	10	10	10	10	
Sum	1380.526	1304.875	1271.814	1358.8	1259.735	
Average	138.0526	130.4875	127.1814	135.88	125.9735	
Variance	874.8381	23.7392	235.6521	276.299	86.26542	
ANOVA						
ce of Varia	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	1890.518	1	1890.518	8.94458	0.004746	4.084746
Columns	1123.374	4	280.8434	1.32875	0.275969	2.605975
Interaction	3126.263	4	781.5657	3.69781	0.011833	2.605975
Within	8454.364	40	211.3591			
Total	14594.52	49				

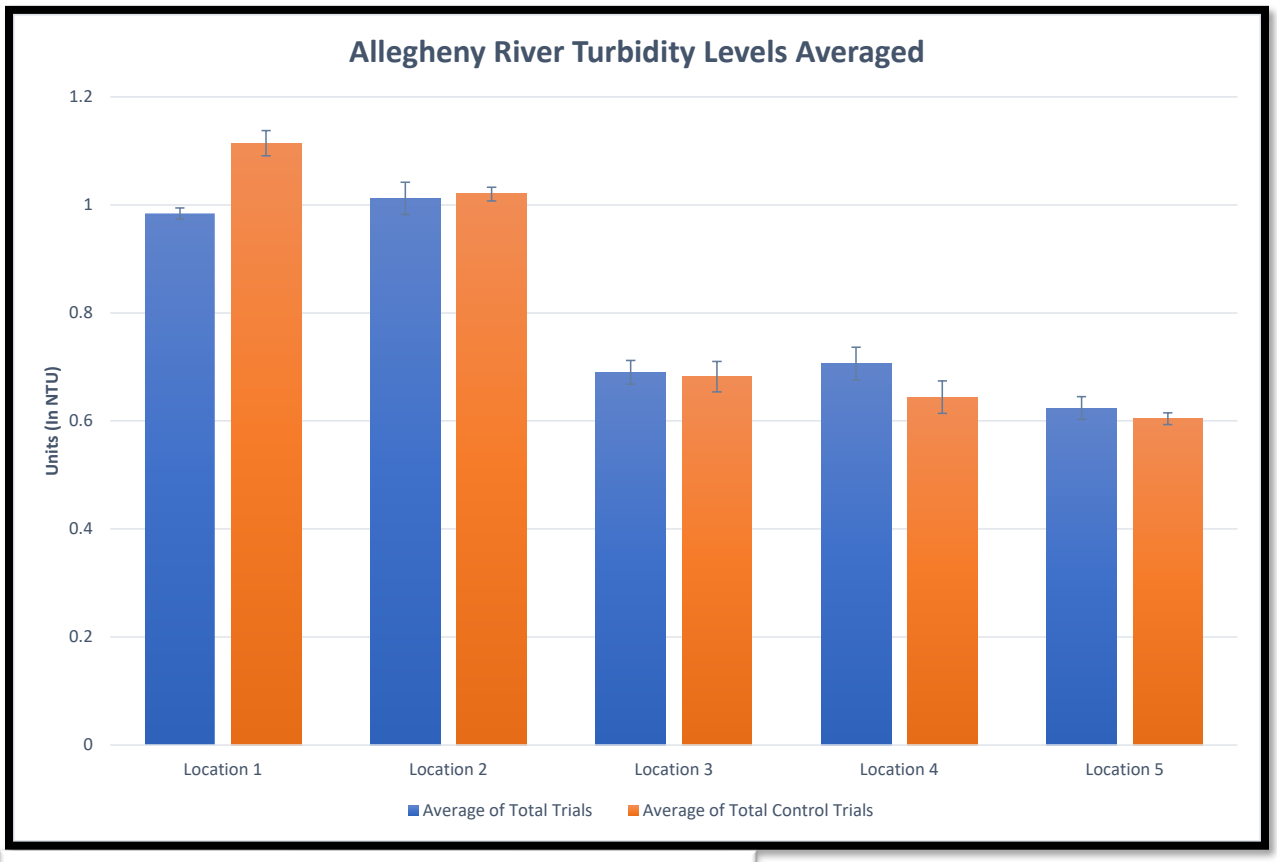
BLUE bar - Using "Potamoi" my Arduino device

ORANGE bar - Using Industrial-Grade (Control)

- The x-axis represents each location (there are two bars for each location because I tested my device and the industrial grade device) and the y-axis represents the units of measurement for each parameter

- Since the p-value is less than 0.05 and the f-value is greater than the f-crit, there was variation between my device's and the industrial grade device's data

Two- Factor ANOVA and Graph for Allegheny River Turbidity Levels



Anova: Two-Factor With Replication

SUMMARY	Location 1	Location 2	Location 3	Location 4	Location 5	Total
Control						
Count	5	5	5	5	5	25
Sum	3.71	2.69	2.26	2.09	2.58	13.33
Average	0.742	0.538	0.452	0.418	0.516	0.5332
Variance	0.00227	0.00097	0.00077	0.00127	0.00673	0.015289
Treatment						
Count	5	5	5	5	5	25
Sum	3.67	2.73	2.06	2.16	2.67	13.29
Average	0.734	0.546	0.412	0.432	0.534	0.5316
Variance	0.00033	0.00138	0.00067	0.00257	0.00143	0.014689
Total						
Count	10	10	10	10	10	
Sum	7.38	5.42	4.32	4.25	5.25	
Average	0.738	0.542	0.432	0.425	0.525	
Variance	0.001173	0.001062	0.001084	0.001761	0.003717	
ANOVA						
ce of Varia	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	3.2E-05	1	3.2E-05	0.017401	0.895715	4.084746
Columns	0.640332	4	0.160083	87.04894	3.45E-19	2.605975
Interaction	0.005588	4	0.001397	0.759652	0.557724	2.605975
Within	0.07356	40	0.001839			
Total	0.719512	49				

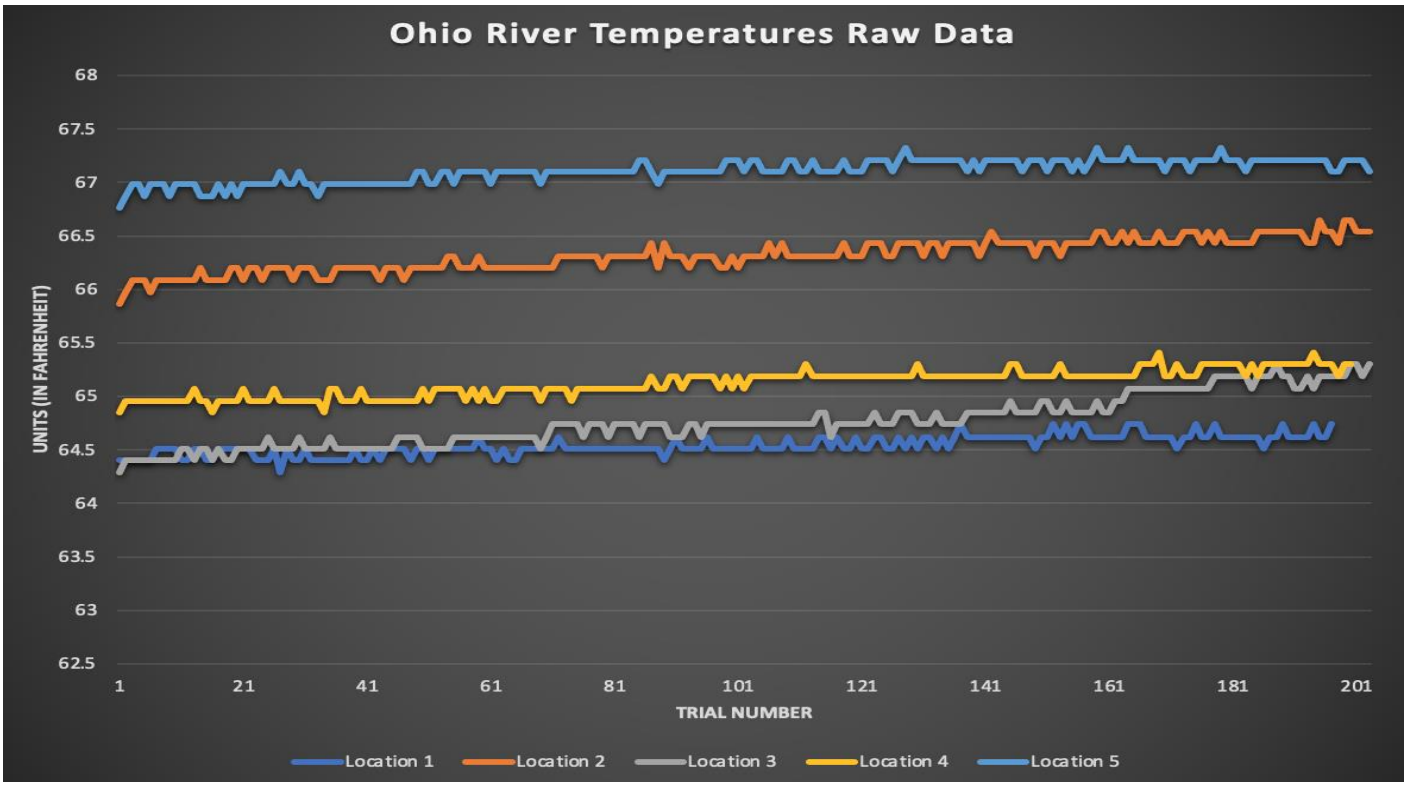
BLUE bar - Using "Potamoi" my Arduino device
ORANGE bar - Using Industrial-Grade (Control)

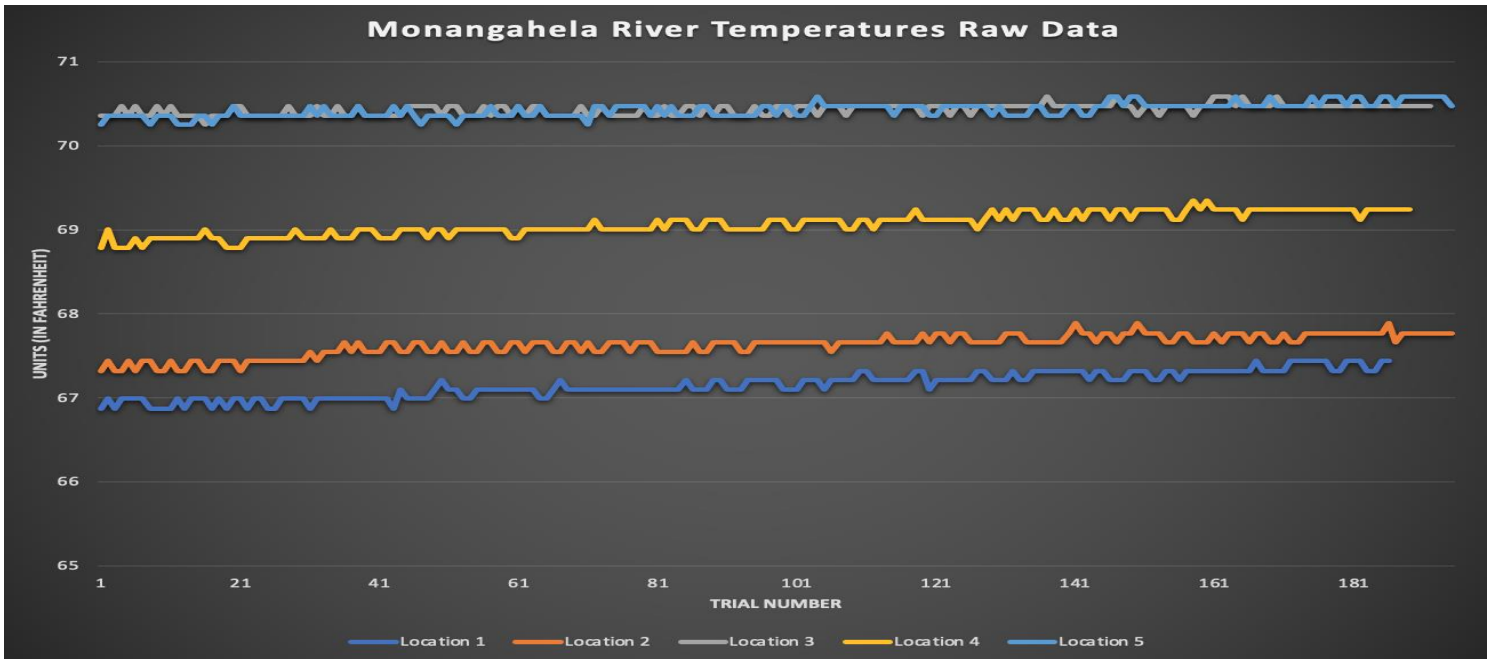
- The x-axis represents each location (there are two bars for each location because I tested my device and the industrial grade device) and the y-axis represents the units of measurement for each parameter

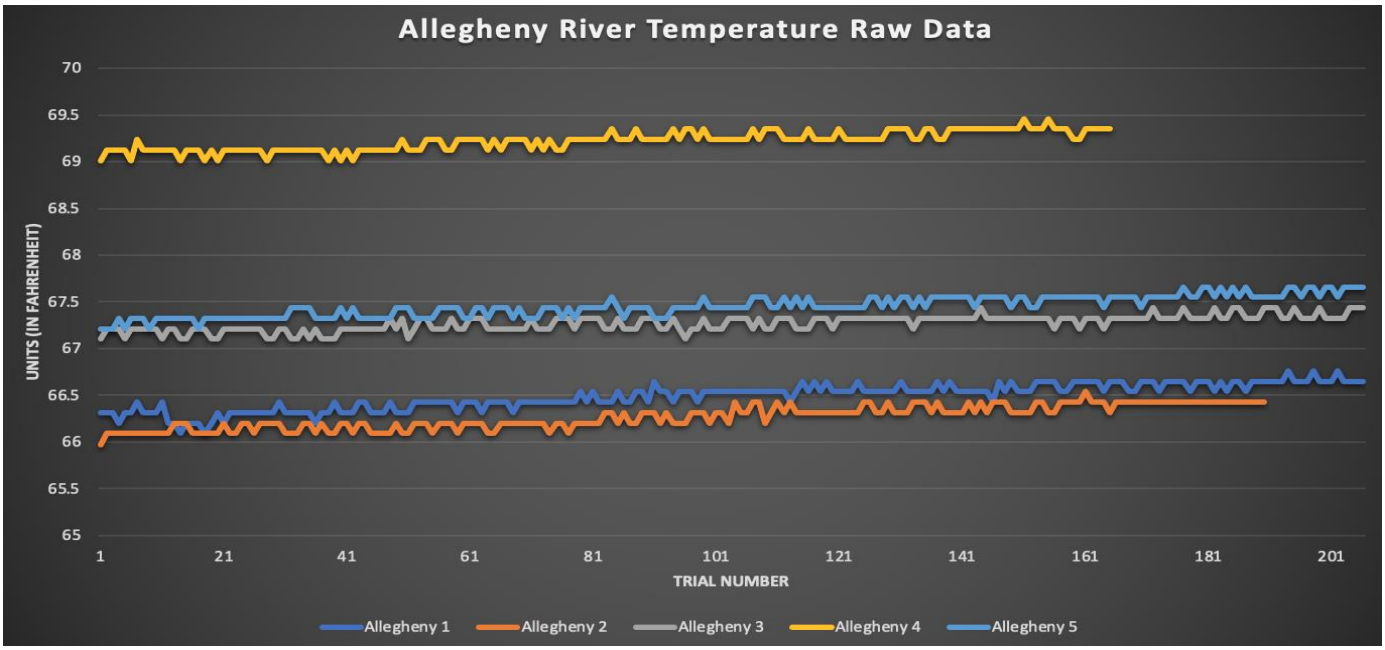
- Since the p-value is greater than 0.05 and the f-crit is greater than the f-value, there was not significant variation between my device's and the industrial grade device's data

6.5 Temperature Readings

- Since the temperature of the water samples rose to room temperature when I brought them in my house, I couldn't use temperature as a parameter in this project
- To show just my device's data, I graphed all 200 trials for each location in this line graph
- The x-axis corresponds with the trial number and the y-axis is the units for the parameter
- Each line represents one of the five locations of each river




Monangahela River Temperatures Raw Data
71
70
69
68
67
66
65
UNITS (IN FAHRENHEIT)
1
21
41
61
81
101
121
141
161
181
TRIAL NUMBER
Location 1
Location 2
Location 3
Location 4
Location 5


Allegheny River Temperature Raw Data
70
69.5
69
68.5
68
67.5
67
66.5
66
65.5
65
UNITS (IN FAHRENHEIT)
1
21
41
61
81
101
121
141
161
181
201
TRIAL NUMBER
Allegheny 1
Allegheny 2
Allegheny 3
Allegheny 4
Allegheny 5

7. Conclusion

- My Water Quality Measurement device (Potamoi) was proved to be accurate based on the comparisons against the industrial-grade device, the ANOVA tests, and the error bars.

- Since there was only significant variation in the TDS results, it is likely that my sensor wasn't calibrated properly or the industrial grade sensor wasn't calibrated properly

- The Bluetooth Module was able to connect to Blynk, which is an app especially for controlling an Arduino, but it wasn't able control all four sensors simultaneously

- When I brought the temperature samples into my house, the temperature rose to room temperature, so I couldn't use temperature as a parameter, but I still recorded the data "Potamoi" received from the temperature sensor

- I successfully incorporated a battery pack into "Potamoi" so that it didn't have to be attached to a computer and could be taken anywhere

Engineering Goals were met

I successfully accomplished all three of my Engineering Goals for this project:

- **Engineering Goal #1: Historical Data display**
    - Data was exported to the Airtable as well as saved in SD card (a way to transfer historical data)
    - Data was also transmitted to my app, "MyH_2O" using Airtable APIs offered by Airtable
    - "MyH_2O" displayed the data seamlessly and was easy to read and interpret the historical data.
- **Engineering Goal # 2: Simple user interface**
    - Developed two solutions, LCD display for real time reading and created an easy to develop mobile app (The data can be retrieved either using mobile app or browser)
    - LCD displayed the data in real time for the user to view similar to those in the market
    - "MyH_2O" Mobile App was very easy to use and displayed the data selected by user instantly
    - The app also was able to record the location of where the data was taken from via a tool called Mapline
- **Engineering Goal #3 : Minimize Cost**
    - The cost of my device was less than $100 whereas the cost of the industrial-grade device I used as the control and many other water quality measuring devices were $200 - $4000

8. Limitations

- Sensors needed to be calibrated once in a while
- I got errors during code upload to Arduino most every change for no apparent reason, but I was able to fix the issues reading, process of elimination like any other problem and applying bug fixes others shared in software user forums on the web. Therefore the project took long time than I expected to complete.
- The browser based software I used to build "MyH_2O" couldn't display the data in a graphical form
- Temperature readings were not the same as when collected from the rivers because the water rose to room temperature as soon as it entered my house

9. Future Enhancements

- Make the device further portable by using a smaller microcontroller (Arduino Nano, Arduino Micro)
- Use an I2C Module for the LCD to decrease the amount of wires
- Include more parameters for water quality testing (dissolved oxygen, electrical conductivity)
- Implement graphical forms of the data being presented in "MyH_2O"
- Make an option in "MyH_2O" for the users to subscribe to a river/water treatment plant location. When that location water quality data changed or new data is uploaded in my mobile App then user gets an email alert from my App, "MyH_2O"
- Find an automated method to collect water samples from various depths of the water for more variability (robot, drone etc.)

10. References

Water Quality Monitoring and Management, 1st Edition, Basis, Technology and Case Studies
Authors: Daoliang Li Shuangyin Liu

Water quality monitoring: a practical guide to the design and implementation of freshwater quality studies and monitoring programs
Author: Jamie Bartram

Ballance, Richard, and Jamie Bartram. Water Quality Monitoring: a Practical Guide to the Design and Implementation of Freshwater Quality Studies and Monitoring Programmes.
Author: Taylor & Francis, 2007.

Iphone Application Development All-in-One for Dummies. Wiley, 2009.
Authors: Goldstein, Neal, and Tony Bove.

Kitchener, Ben GB, et al. “A Review of the Principles of Turbidity Measurement.” Progress in Physical Geography: Earth and Environment, vol. 41, no. 5, 2017, pp. 620–642., doi:10.1177/0309133317726540.

Water Quality Monitoring and Management: Basis, Technology and Case Studies. 1st ed., Academic Press Is an Imprint of Elsevier, 2019.
Authors: Li, Daoliang, and Shuangyin Liu.

Enter the World of Arduino and Its Peripherals and Start Creating Interesting Projects. Packt Pub., 2015.
Authors: Perea, Francis. Arduino Essentials.

11. Appendix

11.1 Code

```
/*****************************************************************
 * Purpose: The program has some key groups: The Setup, Loop, and Error
 * First, the Setup calibrates the 4 sensors and LCD to perform the testing and creates
the SD card file
 *  The Loop programs the data logging shield to store data from the sensors
 * in the SD Card and prints the data on the Arduino Serial Monitor
*  The Error module captures any errors and handles them
 * Creator: Student
 ***************************************************************** */

#include <RTClib.h>
#include "SD.h"
#include <Wire.h>
#include <OneWire.h>
#include <GravityTDS.h>
#include <EEPROM.h>
#include <DallasTemperature.h>
#include<LiquidCrystal.h>
#define Offset 0.00            //deviation compensate for pH
#define PH A2
#define ONE_WIRE_BUS A0
#define TDS A1
#define Pushbutton 10
#define Turbidity A3
#define Backlight 11
#define LOG_INTERVAL  1000 // mills between entries
#define ECHO_TO_SERIAL   1 // echo data to serial port
#define WAIT_TO_START    0 // Wait for serial input in setup()
#define SYNC_INTERVAL 1000 // mills between calls to flush() - to write data to the
card
unsigned long int avgValue;    //Store the average value of pH sensor feedback
const int RS = 7, E = 6, D4 = 4, D5 = 5, D6 = 8, D7 = 9;
const int chipSelect = 10;      // for the data logging shield, we use digital pin 10 for the
SD cs line
int buttonVal = 1;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"};
GravityTDS gravityTds;
LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
```

```
DateTime now;
float Celcius=0;
float Fahrenheit=0;
float temperature = 25;
float tdsValue=0;
float volt;
float ntu;
uint32_t syncTime = 0; // time of last sync()
RTC_DS1307 RTC; // define the Real Time Clock object
File logfile;   // the logging file

void error(char *str)
{
  Serial.print("error: ");
  Serial.println(str);

  while(1);
}

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  RTC.begin();
  Serial.println();
  //RTC.adjust(DateTime(21,3,6,9,59,00));
  #if WAIT_TO_START
    Serial.println("Type any character to start");
    while (!Serial.available());
  #endif //WAIT_TO_START    // initialize the SD card
  Serial.print("Initializing SD card...");
  sensors.begin();
  gravityTds.setPin(TDS);
  gravityTds.setAref(5.0); //reference voltage on ADC, default 5.0V on Arduino UNO
  gravityTds.setAdcRange(1024); //1024 for 10bit ADC;4096 for 12bit ADC
  gravityTds.begin(); //initialization

  lcd.begin(20, 4);          //Sets up the LCD's number of columns and rows
  lcd.clear();               //Clears any existing data on LCD

  pinMode(Pushbutton, INPUT_PULLUP);
  pinMode(ONE_WIRE_BUS, INPUT);       //Temperature sensor data pin
  pinMode(TDS, INPUT);        //TDS sensor data pin
  pinMode(PH, INPUT);         //pH sensor data pin
  pinMode(10, OUTPUT);       // make sure that the default chip select pin is set to
output, even if you don't use it:
```

```
// see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    Serial.println("");
    Serial.println("Date & Time                Temp (In Fahrenheit)          pH          TDS (In
PPM)      Turbidity (In NTU)");
    return;
  }
  Serial.println("card initialized.");

  // create a new file
  char filename[] = "WQDATA00.txt";
  for (uint8_t i = 0; i < 100; i++) {
    filename[6] = i/10 + '0';
    filename[7] = i%10 + '0';
    if (! SD.exists(filename)) {
      // only open a new file if it doesn't exist
      logfile = SD.open(filename, FILE_WRITE);
      break;  // leave the loop!
    }
  }

  if (! logfile) {
    error("couldnt create file");
  }
  Serial.print("Logging to: ");
  Serial.println(filename);

Wire.begin();
  if (!RTC.begin()) {
    logfile.println("RTC failed");
#if ECHO_TO_SERIAL
    Serial.println("RTC failed");
#endif
  }
  logfile.println("Date & Time                Temp (In Fahrenheit)          pH          TDS (In
PPM)      Turbidity (In NTU)");
#if ECHO_TO_SERIAL
  Serial.println("Date & Time                Temp (In Fahrenheit)          pH          TDS (In
PPM)      Turbidity (In NTU)");
#endif ECHO_TO_SERIAL
}
void loop()
{
```

```
   now = RTC.now();

  // delay for the amount of time we want between readings
  delay((LOG_INTERVAL -1) - (millis() % LOG_INTERVAL));
  RTC.now();
  logfile.print(now.year(), DEC);
  logfile.print("/");
  logfile.print(now.month(), DEC);
  logfile.print("/");
  logfile.print(now.day(), DEC);
  logfile.print(" (");
  logfile.print(daysOfTheWeek[now.dayOfTheWeek()]);
  logfile.print(") ");
  logfile.print(now.hour(), DEC);
  logfile.print(":");
  logfile.print(now.minute(), DEC);
  logfile.print(":");
  logfile.print(now.second(), DEC);
  logfile.print("          ");
#if ECHO_TO_SERIAL
  RTC.now();
  Serial.print(now.year(), DEC);
  Serial.print("/");
  Serial.print(now.month(), DEC);
  Serial.print("/");
  Serial.print(now.day(), DEC);
  Serial.print(" (");
  Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
  Serial.print(") ");
  Serial.print(now.hour(), DEC);
  Serial.print(":");
  Serial.print(now.minute(), DEC);
  Serial.print(":");
  Serial.print(now.second(), DEC);
  Serial.print("          ");
#endif //ECHO_TO_SERIAL

  sensors.requestTemperatures();
  Celcius=sensors.getTempCByIndex(0);
  Fahrenheit=sensors.toFahrenheit(Celcius);
  lcd.setCursor(0,2);         //Sets cursor to point (0,4)
  lcd.print("Temperature: "); // Prints Temperature to LCD
  lcd.print(Fahrenheit);//Prints value of Temperature sensor to LCD
  delay(200);

  float phValue=(float)analogRead(PH)*5.0/1024; //convert the analog into millivolt
```

```
  phValue=3.5*phValue+Offset;                 //convert the millivolt into pH value
  lcd.setCursor(0,0);                       //Sets cursor to point (0,0)
  lcd.print("pH Level: ");                  //Prints pH Level: to LCD
  lcd.print(phValue);                       //Prints value of pH sensor to LCD
  delay(200);

  gravityTds.update(); //calculation done here from gravity library
  tdsValue = gravityTds.getTdsValue(); // then get the TDS value
  lcd.setCursor(0,1);         //Sets cursor to point (0,2)
  lcd.print("TDS Level: ");   //Prints TDS Level: to LCD
  lcd.print(tdsValue,0); //Prints value of TDS sensor to LCD
  lcd.print("ppm");
  delay(200);

  volt = ((float)analogRead(Turbidity)/1023)*7.5;
  ntu = volt - 4;
  lcd.setCursor(0,3);
  lcd.print("Turbidity: ");
  lcd.print(ntu);
  delay(200);

  logfile.print(Fahrenheit);
  logfile.print("                ");
  logfile.print(phValue);
  logfile.print("              ");
  logfile.print(tdsValue, 0);
  logfile.print("              ");
  logfile.print(ntu);
#if ECHO_TO_SERIAL
  Serial.print(Fahrenheit);
  Serial.print("                ");
  Serial.print(phValue);
  Serial.print("              ");
  Serial.print(tdsValue, 0);
  Serial.print("              ");
  Serial.println(ntu);
#endif //ECHO_TO_SERIAL

logfile.println();
#if ECHO_TO_SERIAL
#endif // ECHO_TO_SERIAL
if ((millis() - syncTime) < SYNC_INTERVAL) return;
  syncTime = millis();
logfile.flush();
}
```

11.2 My Daily Research Log (2021)

Date	Task	Resolution
1/3	Began learning app development and Arduino basics by reading and printing out online resources	
1/4 to 1/6	Ordered parts for the device (i.e. four sensors, potentiometer, soldering iron, data-logging shield, and LCD display) from Amazon.com	
1/9	I soldered the header pins onto the LCD display to be able to fit into breadboard	
1/10	I wired the 4 sensors, (Temperature, pH, TDS and Turbidity) in to the Arduino Uno	
1/11-1/13	I started writing the code in Arduino IDE to get these sensors working to display data; read books and looked up questions on different communities such as Arduino.com and Adafruit.com to learn how to program each sensor	
1/14	LCD Display was connected and added the code to get it to display real-time data	
1/14-1/20	Worked on refining the program for sensors while identifying the correct formula for each sensor to display the right units; This was the first time I read about PuTTY and got my sensor data to post on to PuTTY screen; PuTTY allows me to write the data I got from the Arduino into a .txt file	
1/21-1/23	Data-logging shield programming was very challenging. It took a lot of extra after-school time to find the right code that would accomplish what I wanted	Data-logging shield was not working. Debugging postponed to tomorrow

Date	Task	Resolution
1/24-1/25	I read further and looked into the Arduino.com community and found that others were having same problem as myself. To solve this problem, I added a different logfile command that was compatible with my computer to be able to write the data to the SD card. The date and time were still showing 1/1/2000, though.	
1/26-1/28	I implemented another app, Blynk, that would show what was being displayed on the LCD display via Bluetooth using a Bluetooth module.	Blynk wasn't getting a signal from my Arduino via the Bluetooth module. I decided to leave it aside and move on with the key tasks.
1/29	To make sure all of the sensors were working, I tested tap water	All results were normal so I was ready to get water samples
1/30	I went to Point State Park to collect the water samples. I collected water from five locations along each of three rivers: Allegheny, Ohio, and Monongahela	I collected 15 samples of water
1/31	I started uploading my sketch (program/code in Arduino IDE) with the first water sample, but I was getting this error: "avrdude: stk500_getsync(): not in sync: resp=0x00"	Looked through Arduino forums and found that many people do not know what to do when they encounter this problem
2/1	As I was working on a MacBook, one person on the forum advised me to download the "https://brew.sh/" package	Even after downloading this, I still got the error
2/7	After almost a week playing around with different installers to get rid of the "Sketch upload" error, I ran the program on a Windows computer instead of a Mac.	It worked without any errors.

Date	Task	Resolution
2/10	I worked on the data-logger shield and got the date and time to display accurately	I got it to work because I installed a different library as I had a different RTC clock chip than other data-logging shields.
2/12	Was able to get the Blynk app working with “Potamoi”	
2/14	When the Blynk code was merged with main code, the LCD stopped working	I thought this was because there were too many commands in the program, and the Arduino couldn’t handle all of them
2/16	Blynk app provided very limited information via the Bluetooth module	
2/17	Back to my app development - I got Bravo Studio to list all of my data using Airtable	I decided NOT to use the Blynk module as my own app provides more information
2/21	Complete all 200 trials for each of the 15 locations, and logged them into an SD card and to PuTTY	
2/22	After reading a Bravo forum, I came to know a website called Integromat that was able to create workflow and conditional logic statements via webhooks	
2/23	I Integrated a top bar into my data display page in my app so that the header would stay fixed when scrolling	
2/24	I created a text bar with the averages in the top bar for each parameter in my app	
2/28	I created the design for my 3d printed case for “Potamoi”	

Date	Task	Resolution
3/1	Sent the box design (hand drew and then drew in Tinkercad) to one of my teachers and answered any questions he had about it before he started printing it.	
3/3	After logging into Figma to update my app design again, I received the following error: "403 Forbidden from the operation" when importing Figma file to Bravo Studio	
3/3	Posted error message to Bravo Studio community	No response
3/4	Resolved Figma error on my own by deleting browser cache and re-logging in to the website	
3/5	Collected the box for Potamoi from my teacher	
3/6	Worked on conditional logic statements in Integromat to get data from three different tables from one “Select Data” screen in my app.	
3/8	As I was pressed for time and Integromat was still not working, I decided to leave it and work on my app	I decided not to use integromat for this time.
3/9	Added a location screen to my app using the Bravo Studio web view so users can view the location of where they took their data from	
3/10	Worked on research report for judges	App was finally completed
3/11 to 3/14	Worked on PowerPoint for fair day presentation	
3/14	Uploaded work to the PRSEF Zfairs site	Finally finished after 3 months of work!