**Group Member Names:** Engineering Students at Portland Community College

**Course and Quarter:** Engineering 114 Fall 2019

**Date: November** 18, 2019

**Revision:** Version 1.0

# **Problem Statement**

The Oregon Drinking Water Services, Department of Environmental Quality, and the Water Resources Department go through rigorous steps to cleanse, purify, and treat the water before it is distributed to our households. In between each of these steps, the water treatment facilities take a lot of measurements to analyze the purity of the water relative to the quality goals of each step. We want to use recreate the quality measurement process with our project. Our scope is to test temperature, pH, and turbidity of different water samples using measurement devices and be able to record the data using Arduino Ide as well as to create a plot using a Python script. This way we can recreate the quality measurement process on a smaller scale.

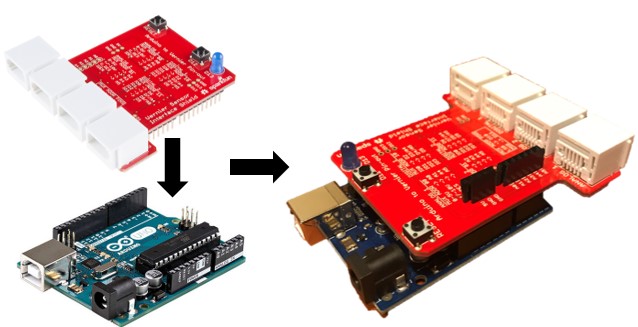
# **Hardware Setup**

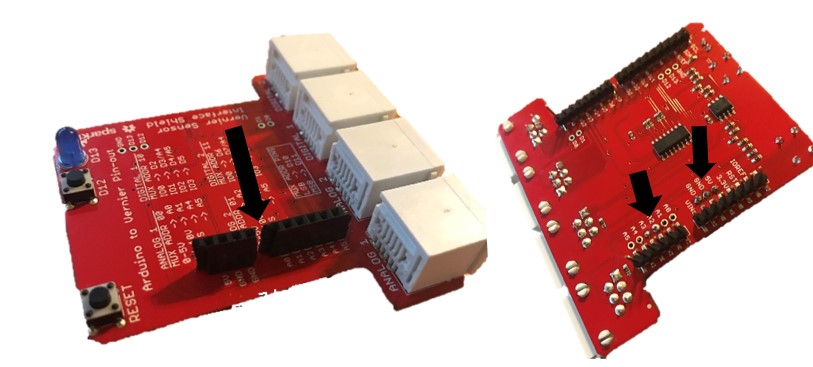
**Bill of Materials**

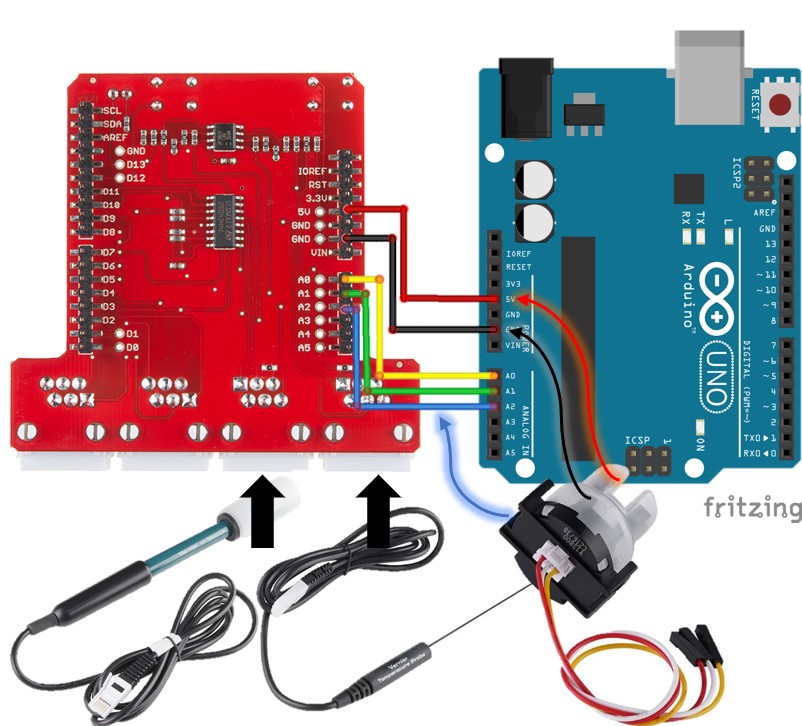
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Part Name** | **Image** | **Purpose** | **Link** | **Price (USD)** |
| Arduino Uno/  R3 Pinout  (DEV-11021) |  | Connecting to the interface shield as well as sending output to computer using MicroUSB | [Arduino Uno R3](https://www.sparkfun.com/products/11021) | **$22.95** |
| Vernier Interface Shield  (DEV-12858) |  | Connecting to the measurement devices as well as sending output to Uno via analog pins | [Interface Shield](https://www.sparkfun.com/products/12858) | **$24.95** |
| pH Sensor  (SEN-12872) |  | Used to measure pH of different water samples and output data to .ino file | [pH Sensor](https://www.sparkfun.com/products/12872) | **$78.95** |
| Turbidity Sensor  (B075H91PY1) |  | Used to measure turbidity of water samples and output data to .ino file | [Turbidity Sensor](https://www.amazon.com/Turbidity-Sensor-Suspended-Particles-Detection/dp/B075H91PY1/ref=asc_df_B075H91PY1/?tag=hyprod-20&linkCode=df0&hvadid=317609719188&hvpos=1o5&hvnetw=g&hvrand=14986883350030508695&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9032826&hvtargid=pla-636687453107&psc=1) | **$9.16** |
| Temperature Probe  (SEN-12871) |  | Used to measure temperature of different water samples and output data to .ino file | [Temperature Sensor](https://www.sparkfun.com/products/12871) | **$28.95** |
| Photon Header- 12 Pin Female  (PRT-14321) |  | Used to connect the analog pin inputs on the vernier interface shield in order to plug in more devices | [Photon Header](https://www.sparkfun.com/products/14321) | **$0.50** |

***\*All images taken from*** [***Sparkfun.com***](http://www.sparkfun.com) ***and*** [***Amazon.com***](http://www.amazon.com)

## **Hardware Schematic / Setup**

**Connect the Arduino Uno R3 to the Vernier Interface Shield.** 

**Solder on some headers to allow connection for a third analog sensor**

**Connect the sensors to Vernier Interface Shield and the third sensor to the headers.**

# **Hookup Guide**

For our hardware hookup, we first start with the Arduino Uno R3 (DEV-11021) and plug it into the Vernier Interface Shield (DEV-12858) that gives it the capability to communicate with the vernier sensors and output the data to the ino files using the Arduino Uno R3. We can then plug in the individual sensors. We first start with the pH sensor (SEN-12872) which plugs into the A0 input of the interface shield. The interface shield allows us to use the 5V input, GND as well as the A0 pin for the sensor reading output. We can then plug in the Turbidity sensor (B075H91PY1) into A1 of the interface shield. The setup is the same as the pH sensor but instead of A0 for the output, we use A1. The Vernier Interface Shield only has 2 plug-in ports for analog devices, but has 5 pins on the board itself which means we can use up to 5 devices at once.

In order to use the pins on the board, we must solder some headers (PRT-14321) onto the Vernier interface shield in order to plug in the third sensor which is the Temperature Sensor (SEN-12871). Once all three sensors are plugged in, we can use the MicroUSB cable to connect the UnoR3/Interface block to the computer. We can then use the Arduino Ino files as well as a Python Script in a jupyter notebook to plot out the readings from the three sensors.

# **Implemented Code**

* For this project we implement the use of Arduino Ino files as well as a Python Script constructed in a Jupyter Notebook for carry out the plots for the sensor readings from each of the devices used.

1. **Arduino Ino for Temperature and pH**

#include <math.h> // Used for calculations

int TempPIN = A0;// Analog Pin 0 for temperature

int pHPIN = A2; // Analog Pin 2 for pH sensor

int TimeInterval = 500; // Time between readings (in ms)

int InputNumber = 0;

void setup()

{

Serial.begin(9600);

}

void loop()

{

int count; //reading from the A/D converter (10-bit)

float temp; //the print below does the division first to avoid overflows

count = analogRead(TempPIN); // read count from the A/D converter

temp = thermistor(count); // and convert it to CelsiusSerial.print(Time/1000); //display in seconds, not milliseconds

Serial.print(temp,1); // display temperature to one digit

Serial.print(" , ");

float voltage, pH; //global variable for reading from A/D converter

voltage = getVoltage(pHPIN); //Measure the voltage at the analog pin

pH = ((1.75 - voltage) / .25) + 7 ; // Convert the voltage to pH levels

Serial.println(pH);

delay(TimeInterval); // Delay a bit...

InputNumber++;

}

float thermistor(int raw) //This function calculates temperature from ADC count

{

long resistance;

float resistor = 15000; //fixed resistor

float temp; // Dual-Purpose variable to save space.

resistance=( resistor\*raw /(1024-raw));

temp = log(resistance); // Saving the Log(resistance) to not calculate it 4 times later

temp = 1 / (0.00102119 + (0.000222468 \* temp) + (0.000000133342 \* temp \* temp \* temp));

temp = temp - 273.15; // Convert Kelvin to Celsius

temp = (temp \* 9.0)/ 5.0 + 32.0; //Convert Celsius to Fahrenheit

return temp; // Return the Temperature

}

float getVoltage(int pin)

{

return (analogRead(pin) \* 0.004882814); //calculate and return pH value from raw voltage data

}

1. **Arduino Ino for Turbidity**

#include <EEPROM.h> \\ libarary that acts as a memory whose values are kept when the board is turned off (like a tiny hard drive). This library enables you to read and write those bytes.

int input = 0; // Analog sensor value (0-1023)

int sensor\_samp = 0; // Averaging variable, sums samples

int n = 10; // Number of samples to average

float voltage = 0.00; // Voltage value derived from analog output (~0-4.5V)

// float Vclear = 4.240; // Calibration variable; voltage at ~0 NTU

// float rel\_turb = 00.00; // Relative turbidity value, based on comparison of measured voltage with Vclear

void setup()

{

Serial.begin(9600); // Baud rate: 9600 bps

// EEPROM.get(0,Vclear); // Retrieve previous Vclear calibration value

}

void loop()

{

for (int i=0; i < n; i++)

{

sensor\_samp += analogRead(A0); // read the input on analog pin 0 (turbidity sensor analog output)

delay(10);

}

input = sensor\_samp / n; // Averages n samples

voltage = input \* (5.000 / 1023.000); // Convert the analog reading (0 - 1023) to a voltage (~0 - 4.5)

// rel\_turb = 100.00 - (voltage/Vclear)\*100.00; // Relative percentage, 0% = ideal clear water

Serial.println(voltage); // Print out the value

sensor\_samp = 0;

delay(500); // Repeat every 500ms

}

1. **Python Script to plot the sensor reading**

import serial # Import serial module

import time

ser = serial.Serial('COM3', 9600) # Set up the serial line

time.sleep(2) # Time delay to read and record the data

Temp = [] # empty lists to store the data

pH = []

for i in range(50):

b = ser.readline() # read a byte string

string\_n = b.decode() # decode byte string into Unicode

string = string\_n.rstrip() # remove \n and \r

temp = string[0:5] # Choose the columns to separate incoming data

ph = string[6:]

Temp.append(temp) # Update the empty lists we created in the beginning

pH.append(ph)

time.sleep(0.1) # Time delay to collect more data

ser.close() # So it stops collecting data after reaching i range value

import matplotlib.pyplot as plt # Import matplotlib for plotting

# If using a Jupyter notebook include %matplotlib inline or %matplotlib notebook

%matplotlib inline

fig, ax = plt.subplots() # Create axes and plot Temperature readings with customized plot

plt.plot(Temp)

plt.xlabel('Reading Value (#)')

plt.ylabel('Temperature (\u00b0 F)')

plt.title('Temperature Reading for Water Sample')

plt.show()

fig, ax = plt.subplots() # Create axes and plot pH readings with customized plot

ax.plot(pH)

ax.set\_xlabel('Reading Value (#)')

ax.set\_ylabel('pH')

ax.set\_title('pH Reading for Water Sample')

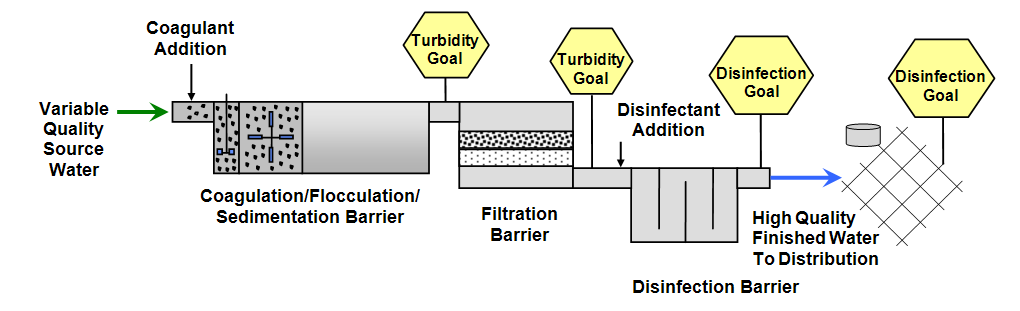
plt.show()

# **Results**

The demonstration for this project consists of several different water samples that will be used to test each of the sensor and its output changing over time. For example, the pH demonstration would consist of placing the probe in tap water to get a steady pH reading, then pouring an additive like vinegar or lime juice that will offset the pH which we will be able to see on the plot. The plot should look similar to the one below:

# **Future Work**

There is a lot of room for improvement when it comes to a project like this; especially when we use to Arduino parts from Sparkfun which support a lot of ways to incorporate further design in order to carry out more tasks and collect more data. We want to be able to use more devices to mimic the process flow of the water treatment facilities. We could possibly start out with dirty water with contaminants and use the sensors at multiple steps to make sure we reach the goal for water quality and then end up with water that is either cleaner than the initial sample or even drinkable. The diagram below shows the process flow followed by the water treatment facilities and how they have set placement for the checking the water quality. Using the code, we could also store the sensor readings onto an SD card or to the cloud for future analysis. We could also implement the Python script to allow live plotting of real time data which is more feasible for the water treatment facilities as they may require continuous monitoring of the water quality parameters.

[*www.oregon.gov/oha/PH/HealthyEnvironments/DrinkingWater/Operations/Treatment/Pages/index.aspx*](https://www.oregon.gov/oha/PH/HealthyEnvironments/DrinkingWater/Operations/Treatment/Pages/index.aspx)

Being able to use Arduino microprocessors and Python code to construct and understand how large scale facilities operate would be an excellent engineering outcome as well as a commemorable achievement.

# **License**

**Creative Commons Attribution-ShareAlike CC BY-SA**

This license lets others remix, adapt, and build upon your work even for commercial purposes, as long as they credit you and license their new creations under the identical terms. This license is often compared to “copyleft” free and open source software licenses. All new works based on yours will carry the same license, so any derivatives will also allow commercial use. This is the license used by Wikipedia, and is recommended for materials that would benefit from incorporating content from Wikipedia and similarly licensed projects.

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

# **Sources**

* <https://www.oregon.gov/oha/PH/HealthyEnvironments/DrinkingWater/Operations/Treatment/Pages/index.aspx#treat>
* <https://www.sparkfun.com/products/13975>
* <https://www.sparkfun.com/products/12858>
* <https://www.sparkfun.com/products/12872>
* <https://www.sparkfun.com/products/12871>
* <https://www.sparkfun.com/products/14321>
* <https://www.amazon.com/Turbidity-Sensor-Suspended-Particles-Detection/dp/B075H91PY1/ref=asc_df_B075H91PY1/?tag=hyprod-20&linkCode=df0&hvadid=317609719188&hvpos=1o5&hvnetw=g&hvrand=14986883350030508695&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9032826&hvtargid=pla-636687453107&psc=1>
* <https://github.com/sparkfun/Fritzing_Parts/tree/master/products>
* <https://creativecommons.org/licenses/>