Jim Foley and Josh Collins

ENGR114/Winter 2019:

3/19/19

Revision Number: 1

## **Ph Sensor→Arduino→Python→IOT→Python**

#### **Problem Statement:**

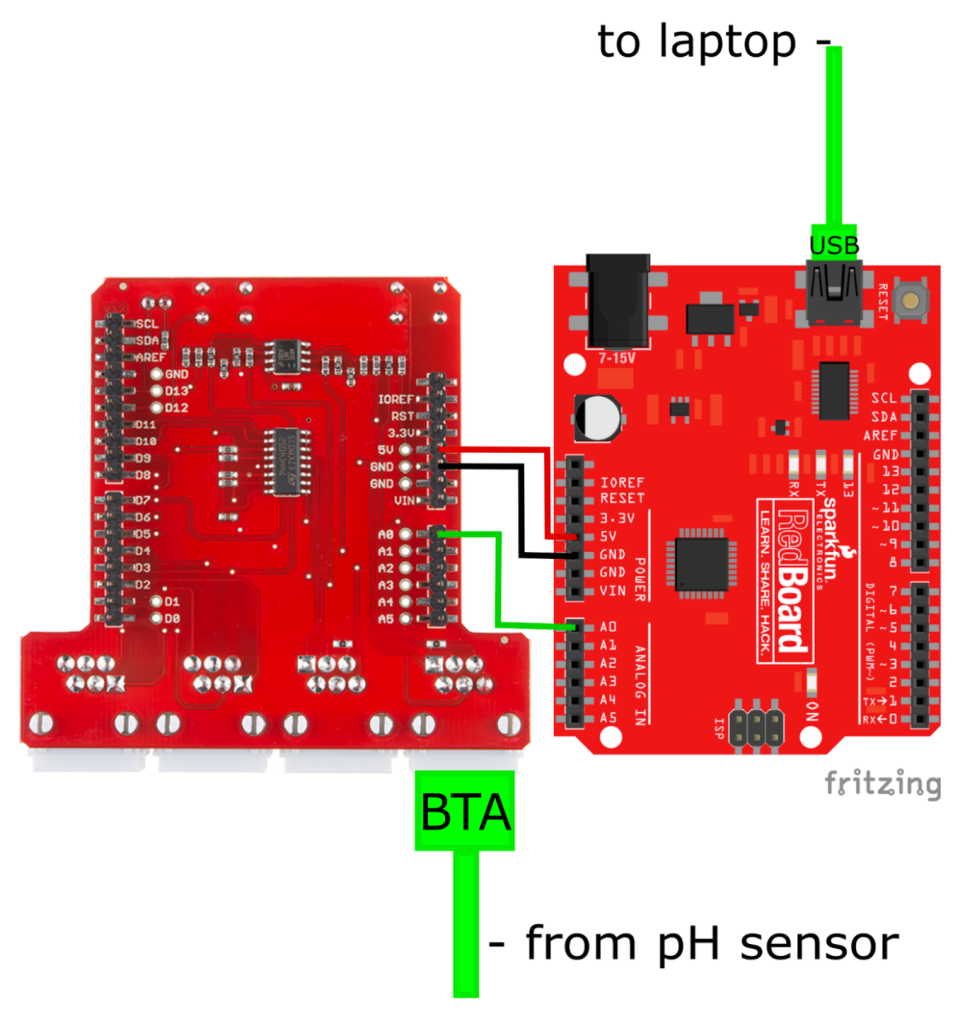
Our group was tasked with using a ph-sensor to measure pH level in the Engineering lab’s fish tank. These values will be processed by an Arduino microcontroller and will be uploaded to Thingspeak.com via a connected laptop. From the thingspeak.com server, the data will be retrieved by a user running a Python script to build a graph of the retrieved data.

#### **Hardware Setup:**

**Bill of Materials:** *(excludes peripherals; only items used/needed are listed in table)*

|  |  |
| --- | --- |
| Component Name | Vendor and url link |
| Arduino | [SparkFun RedBoard - Programmed with Arduino](https://www.sparkfun.com/products/13975) |
| pH Sensor | [Vernier pH Sensor PH-BTA](https://www.sparkfun.com/products/12872) |
| Sensor Shield | [SparkFun Vernier Interface Shield](https://www.sparkfun.com/products/12858) |
| Mini-B USB cable | [SparkFun USB Mini-B Cable - 6 Foot](https://www.sparkfun.com/products/11301) |
| ph 4 solution | Standard Buffered pH 4 Solution - from PCC Engineering Lab |
| ph 7 solution | Standard Buffered pH 7 Solution - from PCC Engineering Lab |
| ph 10 solution | Standard Buffered pH 10 Solution - from PCC Engineering Lab |

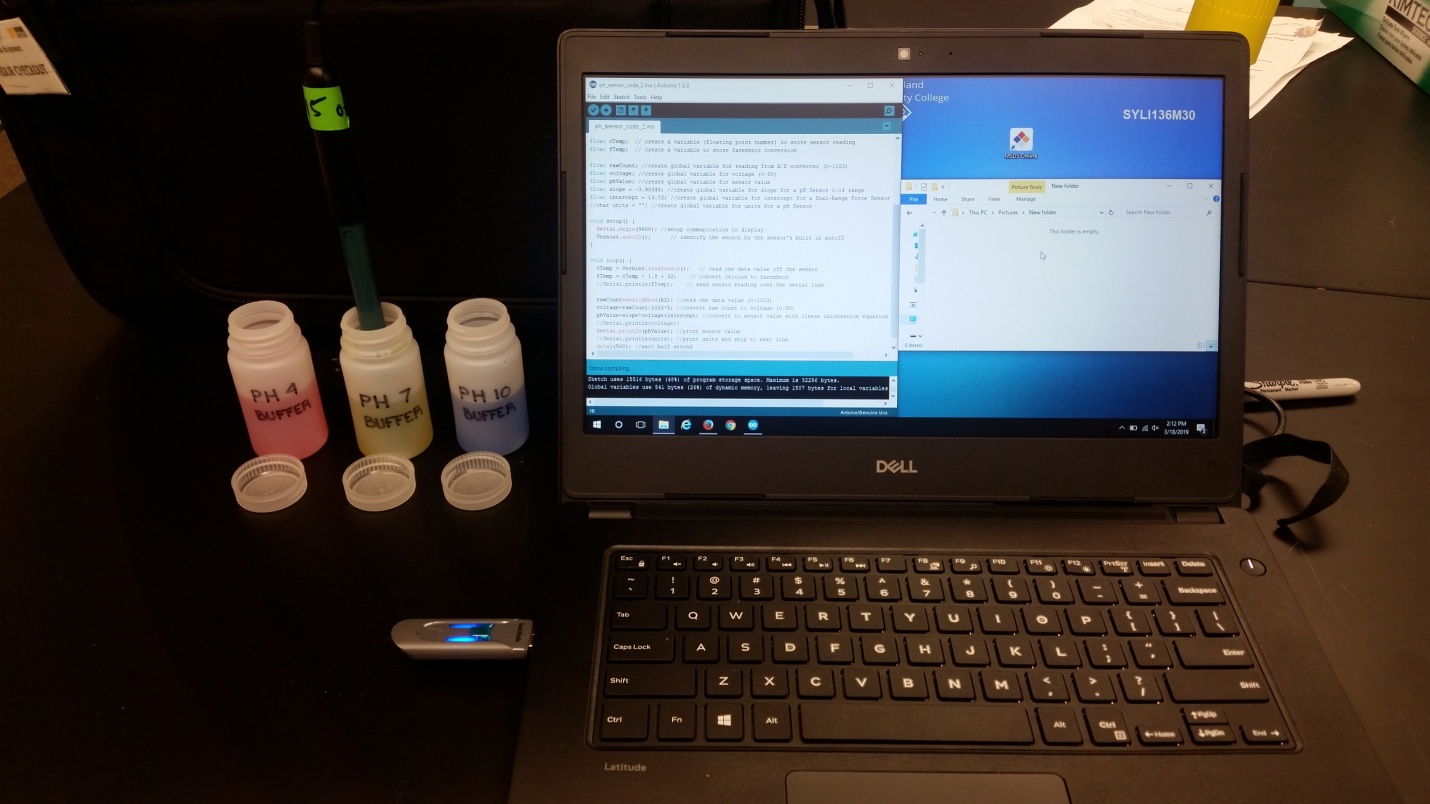
**Hardware Schematic:**

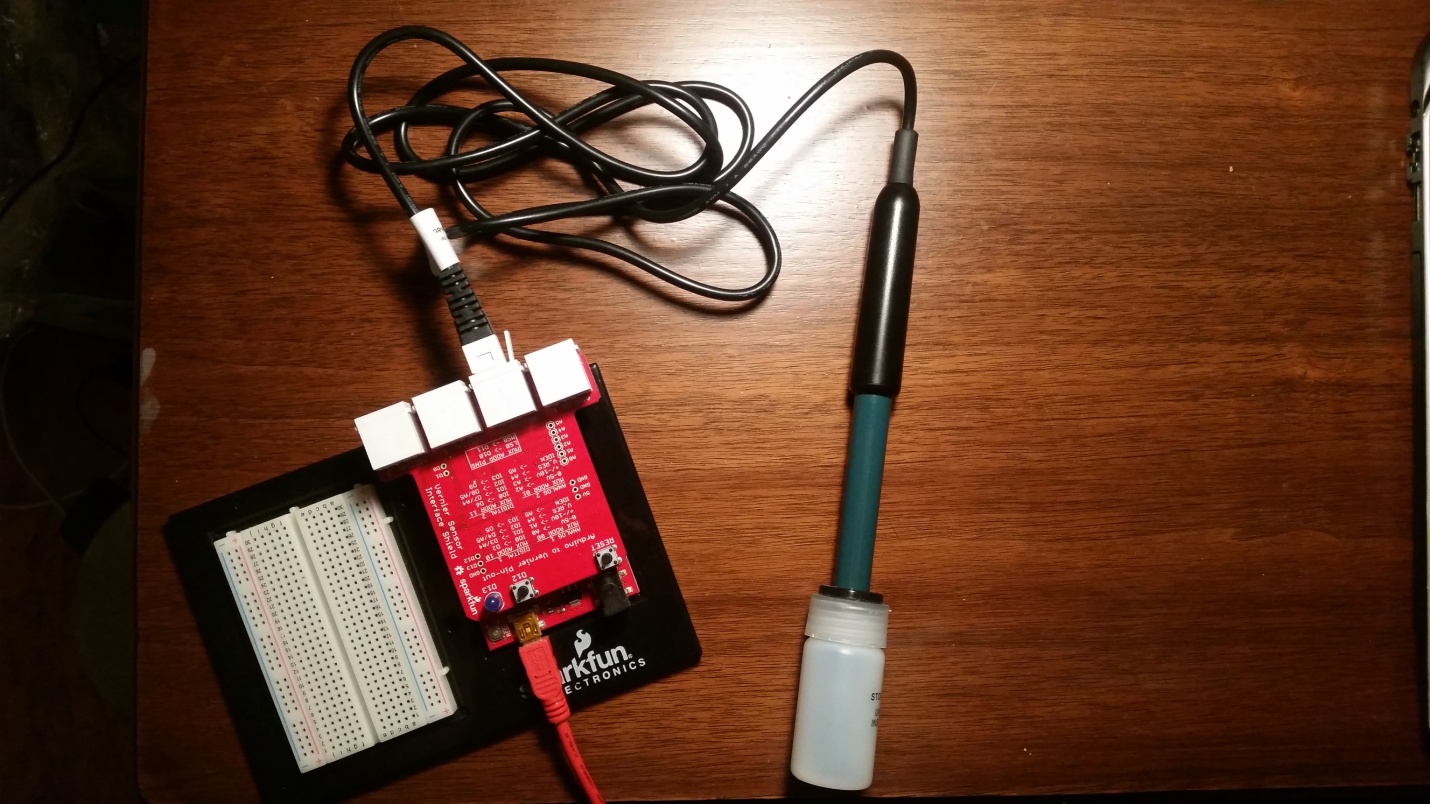


**Hookup Guide:**

The pH sensor (Vernier part# PH-BTA) is connected to the Vernier Interface Shield (Sparkfun part# DEV-12858) using the the pH sensor's British Telecom Analog (BTA) connector. This is then plugged into either the Analog 1 or Analog 2 connector of the shield. The Vernier Interface Shield was placed on top of the Arduino (Sparkfun part# DEV-13975). The boards are aligned properly so the rows of pins on the shield and female Arduino connectors are oriented correctly. The Arduino was powered using the same red Mini-B USB cable (Sparkfun part# CAB-11301) that connects it to a laptop running Jupyter Notebook.

**Images:**





**Python Code:**

**Read data from Arduino and upload to Thingspeak:**

import requests

import serial

import time

# Read and record the data

data =[] # initialize and empty list to store the data

while True:

ser = serial.Serial('COM4', 9600) # set up the serial connection with arduino

time.sleep(1)

b = ser.readline() # read a byte string line from the Arduino's serial output

b\_string = b.decode() # decode byte string into regular Python string

string = b\_string.rstrip() # remove \n and \r from the string

flt = float(string) # convert the string to a float

#https://api.thingspeak.com/update?api\_key=U7HEWMOITHOD9LBM&field1=0

base\_url = "https://api.thingspeak.com/update?api\_key="

api\_key = "U7HEWMOITHOD9LBM"

mid\_url\_1 ='&field1=' # field number, if only one field use 1

url = base\_url + api\_key + mid\_url\_1 + string # data\_point

#print(url) # prints url to screen for feedback on when/what data is uploaded

r = requests.get(url) # upload data point to thingspeak channel

time.sleep(15) # delays loop for thingspeak upload rate

ser.close() # closes the serial connection with arduino

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Reading and Plotting the Thingspeak Data:**

## Python code for reading, parsing, and plotting data from Thingspeak

# import

import time

import numpy as np

import requests

import json

%matplotlib inline

# ask user for number of data points

results\_num = input('How many data points?:')

## Read data stored on ThingSpeak.com

#https://api.thingspeak.com/channels/714132/fields/1.json?api\_key=HDWL8KCODG8WJKMV&results=2

base\_url = 'https://api.thingspeak.com/channels/'

channel\_num = '714132'

mid\_url = '/fields/'

field\_num = '1'

next\_url = '.json?'

# api\_url = 'HDWL8KCODG8WJKMV'

end\_url = 'results='

results\_num #number of datapoints, saved as a string

url = base\_url + channel\_num + mid\_url + field\_num + next\_url + end\_url + results\_num

url

#use the requests library to pull down the data from thingspeak into a variable

r = requests.get(url)

json\_data = r.json()

#print(json\_data)

# Test Code- Use this section to test if field1 is retrieved

#print(json\_data['feeds'][0]['field1'])

#feeds = json\_data["feeds"]

# iterate over the list:

#for feed in feeds:

# get the value for the "field1" key

#print (feed["field1"])

# Store data into lists named data, time

feeds = json\_data["feeds"]

data = []

time = []

for feed in feeds:

# get the value for the "field1" key

data.append(feed["field1"])

time.append(feed["created\_at"])

#print (feed["field1"])

#print(feed["created\_at"])

# Read Data and build graph

# plot the data

fig, ax = plt.subplots()

ax.plot(time, data,'r',linewidth=0.8)

ax.set\_xlabel('Time')

ax.set\_xticklabels(time, rotation=-45, ha='left')

ax.set\_ylabel('PH from Sensor')

#data was graphed max to min, this arranged the y-axis min to max

ax.invert\_yaxis()

ax.set\_title('Ph Sensor from fish tank')

plt.tight\_layout()

#save plot image

plt.savefig('phsensor.png', dpi=72)

plt.show()

**Arduino Code:**

/\* VernierTutorialLinearCalibration (v2017)

\*

\* This sketch reads the raw count from a Vernier Analog (BTA)

\* sensor once every half second, and uses its algebraic slope

\* and intercept to convert it to standard units.

\*

\* Plug the sensor into the Analog 2 port on the Vernier Arduino

\* Interface Shield or into an Analog Protoboard Adapter wired

\* to Arduino pin A2.

\*/

float rawCount; //create global variable for reading from A/D converter (0-1023)

float voltage; //create global variable for voltage (0-5V)

float phValue; //create global variable for sensor value

float slope = -3.90345; //create global variable for slope for a pH Sensor 0-14 range

float intercept = 13.72; //create global variable for intercept for a Dual-Range Force Sensor 0-14 range

void setup() {

Serial.begin(9600); //setup communication to display

}

void loop() {

rawCount=analogRead(A0); //read one data value (0-1023)

voltage=rawCount/1023\*5; //convert raw count to voltage (0-5V)

phValue=slope\*voltage+intercept; //convert to sensor value with linear calibration equation

//Serial.println(voltage); //prints sensor voltage to serial monitor for calibration

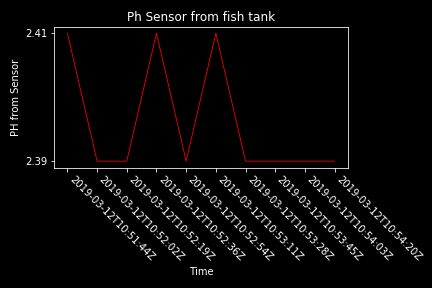
Serial.println(phValue); //print sensor value

delay(500); //wait half second

}

#### **Results:**

Image of Python Plot from Ph-Sensor

****

#### **Future Work:**

* To take this project to the next step, it would have been interesting to try and set up the ph-sensor to send upload data at specified time intervals and have a graph that could graph the incoming results in real time.
* Additionally, if time was not an issue it would be relatively straightforward to connect a temperature probe to the Arduino for added capability.

#### License



Copyright 2019 MIT

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.