RasPi + Arduino EC and Temp Sensors Report

Jonah Gonzales
ANT (Automated Nutrient Technique)
November 8, 2021

Opening:

This module will allow us to process the data through the RasPi, which will in turn let us control the peristaltic pumps that will inject the system with the nutrients as needed to balance the system. It will be adjusted once we settle on a sampling rate for the measurements, and control of the peristaltic pumps will be added to the Python code. Currently, the Python code takes the average of the 5 most recent measurements. The average is taken to account for any fluctuations in the readings. We imagine we will take a sample over one minute, every 20 minutes, for example. Once the system is built and we get peristaltic pump control, we will adjust and set those times.

Fritzing Schematic:

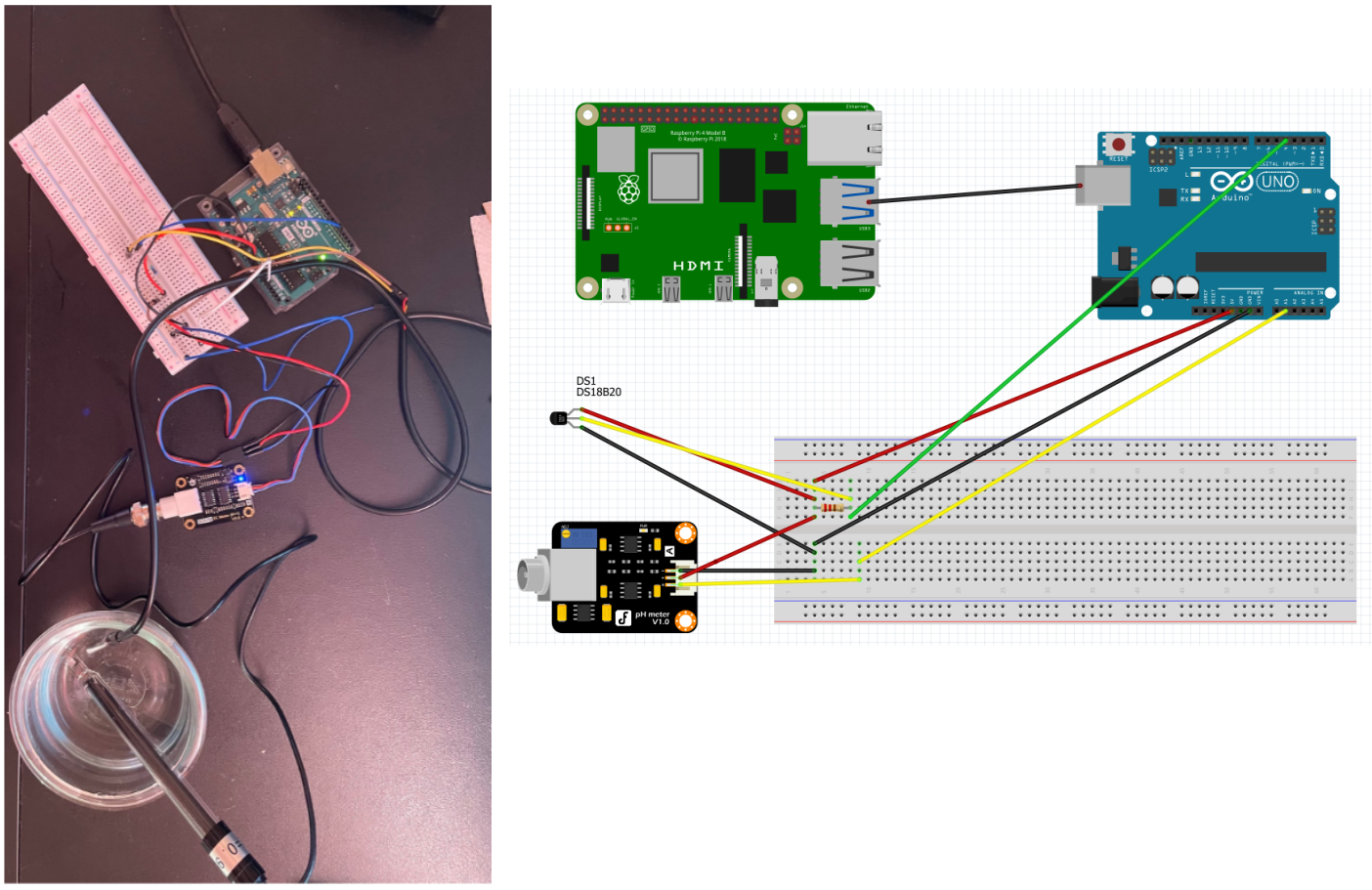

Figure 1. Left: Photo of built schematic. Right: Fritzing Diagram of RasPi + Arudino Temp/EC Sensor setup.

Code:

RasPi Python Code (Will be adjusted once we settle on sampling rate):

```
import serial
import re
import sys
import os
import string
import time

#open serial port
ser=serial.Serial('/dev/ttyACM0', 115200)

temp = [0,0,0,0,0]
EC = [0,0,0,0,0]
counter = 0

while True:
   SensorData = str(ser.readline().decode("utf-8")).split(' ')
   #print(SensorData)

   temp[counter] = float(SensorData[0])
   EC[counter] = float(SensorData[1])

   TempCount = 0
   ECCount = 0

   for i in range(0,5):
       TempCount = TempCount + temp[i]
       ECCount = ECCount + EC[i]

   TempAvg = TempCount/5
   ECAvg = ECCount/5
   print("Temp[]: ", temp, "EC[]: ", EC, "\n")
   print("Temp: ", TempAvg, "EC: ", ECAvg, "\n")

   counter = counter + 1
   if counter > 4:
       counter = 0
```

Arduino Code:

```
#include <DS18B20.h>
#include "DFRobot_EC.h"
#include <EEPROM.h>
#include <DallasTemperature.h>

#define EC_PIN A1
#define Temp_Sensor 4

float voltage,ecValue,temperature = 25;
DFRobot_EC ec;
OneWire oneWire(Temp_Sensor);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(115200);
  ec.begin();
  sensors.begin();
}

void loop()
{
    static unsigned long timepoint = millis();
    if(millis()-timepoint>1000U)  //time interval: 1s
    {
      timepoint = millis();
      voltage = analogRead(EC_PIN)/1024.0*5000;   // read the voltage
      //temperature = 25;                         //Temporary value until temp sensor added
      temperature = readTemperature();          // read your temperature sensor to execute
temperature compensation
      ecValue =  ec.readEC(voltage,temperature);  // convert voltage to EC with temperature
compensation
      //Serial.print("temperature:");
      Serial.print(temperature,1);
      Serial.print(" ");
      //Serial.print("^C  EC:");
      Serial.print(ecValue,2);
      Serial.print("\n");
      //Serial.println("ms/cm");
    }
    ec.calibration(voltage,temperature);          // calibration process by Serail CMD
}

float readTemperature() {
```

```
  //Serial.print(" Requesting temperatures...");
  sensors.requestTemperatures(); // Send the command to get temperature readings
  //Serial.println("DONE");
/********************************************************************/
  //Serial.print("Temperature is: ");
  temperature = (sensors.getTempCByIndex(0)); // Why "byIndex"?
   // You can have more than one DS18B20 on the same bus.
   // 0 refers to the first IC on the wire
  delay(1000);
  return temperature;
}
```

Results Screenshots: (Calibrated as in previous EC Sensor Report)
Temperatures verified by food thermometer
12.88 ms/cm Solution

```
pi@Noah:~ $ /usr/bin/python3 /home/pi/EC_Test2.py
Temp[]:  [24.4, 0, 0, 0, 0] EC[]:  [12.98, 0, 0, 0, 0]

Temp:  4.88 EC:  2.596

Temp[]:  [24.4, 24.4, 0, 0, 0] EC[]:  [12.98, 12.96, 0, 0, 0]

Temp:  9.76 EC:  5.188000000000001

Temp[]:  [24.4, 24.4, 24.4, 0, 0] EC[]:  [12.98, 12.96, 12.96, 0, 0]

Temp:  14.639999999999997 EC:  7.780000000000001

Temp[]:  [24.4, 24.4, 24.4, 24.4, 0] EC[]:  [12.98, 12.96, 12.96, 12.93, 0]

Temp:  19.52 EC:  10.366000000000001

Temp[]:  [24.4, 24.4, 24.4, 24.4, 24.4] EC[]:  [12.98, 12.96, 12.96, 12.93, 12.96]

Temp:  24.4 EC:  12.958000000000002

Temp[]:  [24.4, 24.4, 24.4, 24.4, 24.4] EC[]:  [12.95, 12.96, 12.96, 12.93, 12.96]

Temp:  24.4 EC:  12.952000000000002

Temp[]:  [24.4, 24.4, 24.4, 24.4, 24.4] EC[]:  [12.95, 13.01, 12.96, 12.93, 12.96]

Temp:  24.4 EC:  12.962

Temp[]:  [24.4, 24.4, 24.4, 24.4, 24.4] EC[]:  [12.95, 13.01, 13.03, 12.93, 12.96]

Temp:  24.4 EC:  12.975999999999999

Temp[]:  [24.4, 24.4, 24.4, 24.4, 24.4] EC[]:  [12.95, 13.01, 13.03, 13.03, 12.96]

Temp:  24.4 EC:  12.996

Temp[]:  [24.4, 24.4, 24.4, 24.4, 24.4] EC[]:  [12.95, 13.01, 13.03, 13.03, 13.04]
```

Figure 2: Terminal results of Python Code when testing the 12.88 ms/cm solution

1413 us/cm Solution:

```
Temp[]:  [24.4, 0, 0, 0, 0] EC[]:  [1.29, 0, 0, 0, 0]
Temp:  4.88 EC:  0.258
Temp[]:  [24.4, 24.5, 0, 0, 0] EC[]:  [1.29, 1.26, 0, 0, 0]
Temp:  9.78 EC:  0.51
Temp[]:  [24.4, 24.5, 24.6, 0, 0] EC[]:  [1.29, 1.26, 1.29, 0, 0]
Temp:  14.7 EC:  0.768
Temp[]:  [24.4, 24.5, 24.6, 24.6, 0] EC[]:  [1.29, 1.26, 1.29, 1.32, 0]
Temp:  19.619999999999997 EC:  1.032
Temp[]:  [24.4, 24.5, 24.6, 24.6, 24.7] EC[]:  [1.29, 1.26, 1.29, 1.32, 1.29]
Temp:  24.56 EC:  1.29
Temp[]:  [24.7, 24.5, 24.6, 24.6, 24.7] EC[]:  [1.29, 1.26, 1.29, 1.32, 1.29]
Temp:  24.62 EC:  1.29
Temp[]:  [24.7, 24.7, 24.6, 24.6, 24.7] EC[]:  [1.29, 1.32, 1.29, 1.32, 1.29]
Temp:  24.66 EC:  1.302
Temp[]:  [24.7, 24.7, 24.7, 24.6, 24.7] EC[]:  [1.29, 1.32, 1.29, 1.32, 1.29]
Temp:  24.68 EC:  1.302
Temp[]:  [24.7, 24.7, 24.7, 24.7, 24.7] EC[]:  [1.29, 1.32, 1.29, 1.26, 1.29]
Temp:  24.7 EC:  1.29
Temp[]:  [24.7, 24.7, 24.7, 24.7, 24.8] EC[]:  [1.29, 1.32, 1.29, 1.26, 1.25]
Temp:  24.72 EC:  1.282
```

Figure 3: Terminal results of 1413 us/cm solution test

References:

https://yantraas.com/send-sensor-data-from-arduino-to-raspberry-pi/#Setting_Up_Serial_Communication_On_Arduino