Supporting information for

Aquatic quality watch informed by communities (AQWIC) facilitating the adoption of low-cost sensor systems for underserved communities: A review and tutorial

Hacker, Josiah[1]; Iqbal, Fatima[1]; Osuna, Matías[2]; Perea, Osely[3]; Sanchéz, Keiner[3]; Page, Christopher[4]; Torres, Jamie[5]; Uddin, Md. Nizam[1]; Walden, Hannah[6]; Westerbeke, Mikko[6]; Woody, Astana[1]; Rubio Murillo, Nazly Enith[3]; Cubas-Suazo, Francisco[1]; Díaz-Arriaga, Farith[2]; Rowles, Lewis S.[1]*

[1]*Department of Civil Engineering and Construction, Georgia Southern University, Statesboro, Georgia 30458, United States*
[2]*Department of Civil and Environmental Engineering Universidad Escuela de Ingenieria de Antioquia, Envigado, Antioquia, Colombia*
[3]*Department of Environmental Engineering, Universidad Tecnológica Del Chocó, Quibdo, Istmina, Choco, Colombia*
[4]*Department of Computer Science, Georgia Southern University, Statesboro, Georgia 30458, United States*
[5]*Department of Electrical Engineering, Georgia Southern University, Statesboro, Georgia 30458, United States*
[6]*Department of Mechanical Engineering, Georgia Southern University, Statesboro, Georgia 30458, United States*
**Corresponding author: Lewis S. Rowles, email: lrowles@georgiasouthern.edu, phone: (912) 478-0772*

Prepared for submission to
Environmental Science: Advances

Table of Contents (11 total pages):

Table S1. Review of articles with individual spreadsheets for all literature reviewed, specific articles related to this work, and details on how the literature review was conducted.
(Table S1 is included as an Excel file:
Water_Sensor_Lit_Review.xlsx)

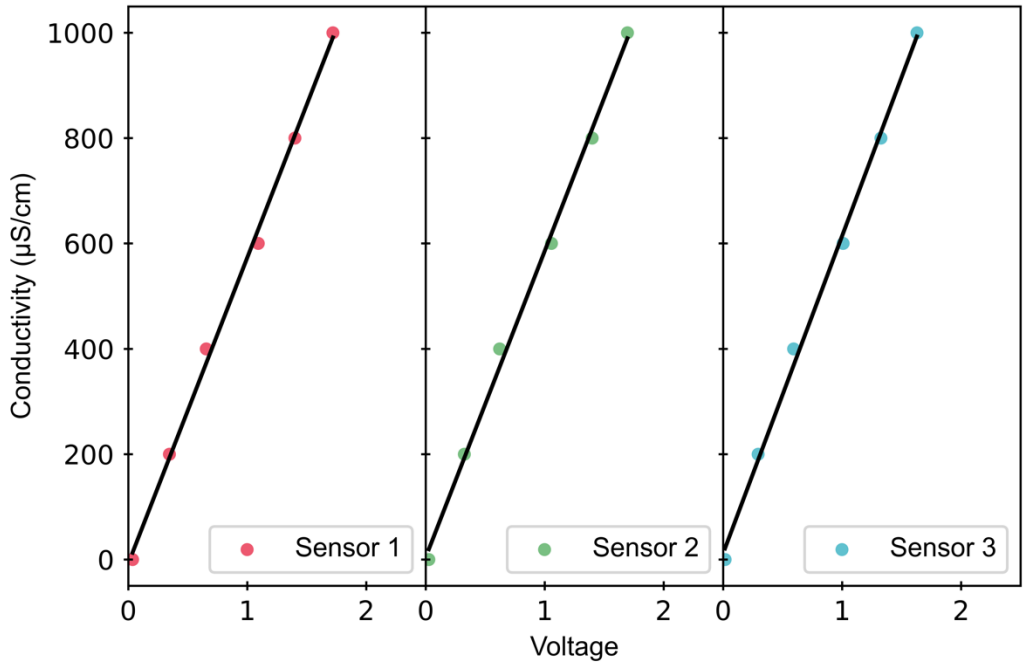

Figure S1. Calibration curves for conductivity. The R^2 of Sensor 1 was 0.997, Sensor 2 was 0.997, and Sensor 3 was 0.997.

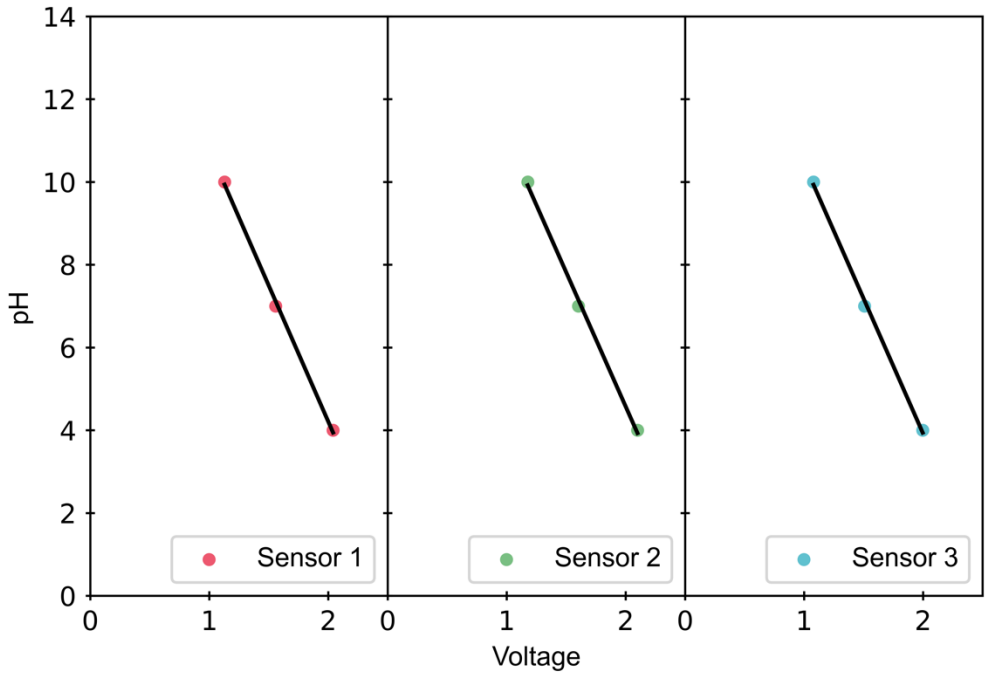

Figure S2. Calibration curves for pH. The R^2 of Sensor 1 was 0.999, Sensor 2 was 0.998, and Sensor 3 was 0.999.

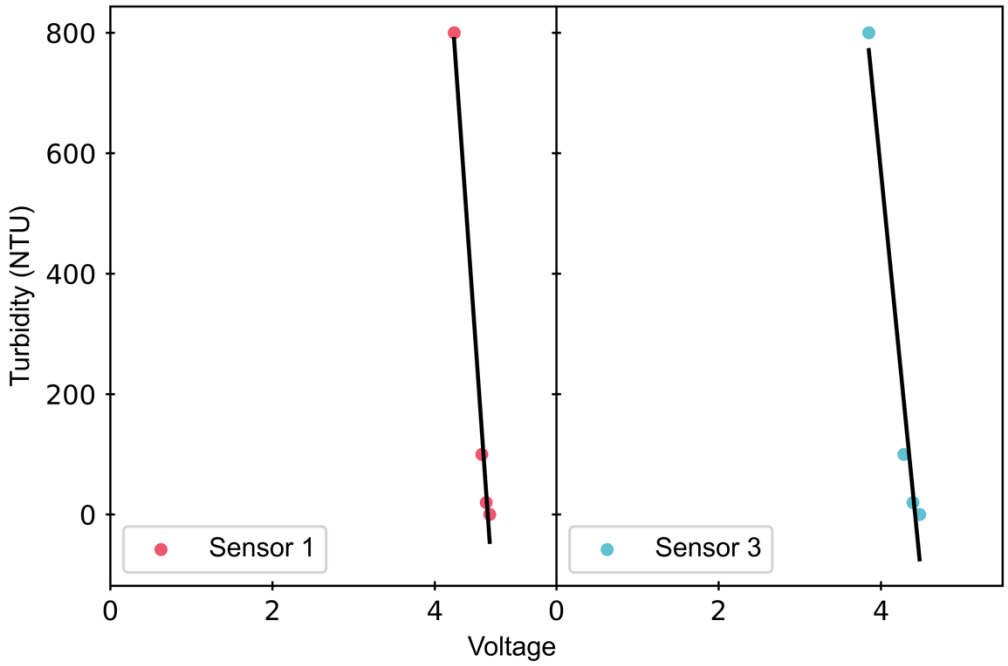

Figure S3. Calibration curves for turbidity. The R^2 of Sensor 1 was 0.991 and Sensor 2 was 0.968.

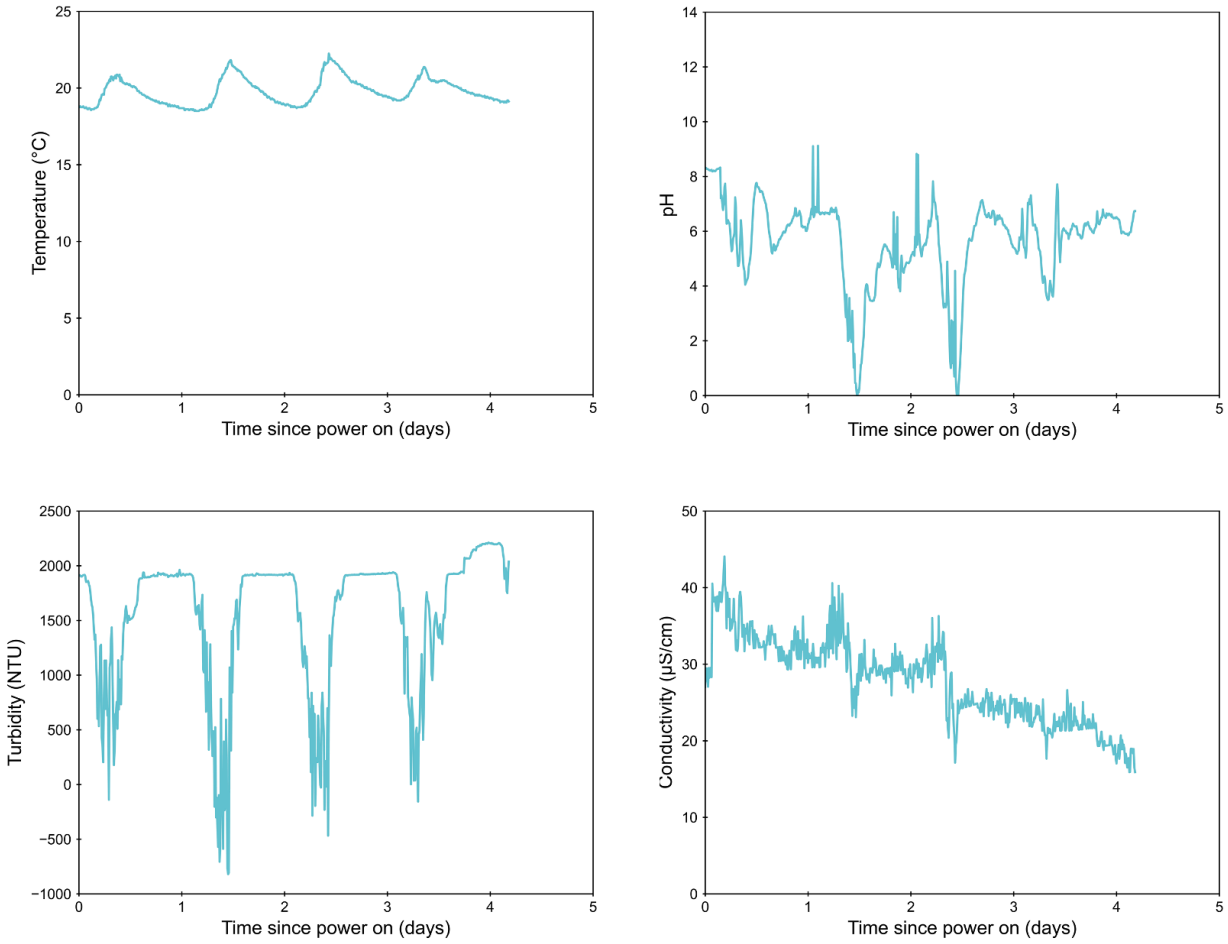

Figure S4. Field tests of the sensor systems in Colombia.

Figure S5. Photos of the sensor systems in deployed Colombia.

Section S1. Arduino code

```
//Indicate libraries
#include <OneWire.h>
#include <DallasTemperature.h>
#include <EEPROM.h>
#include <SD.h>
File myFile;

//Define pins
#define pHSensorPin A0 //pH
#define TDSSensorPin A1 //Total Dissolved Solids
#define turbSensorPin A2 //Turbidity
#define pinTemp 7
#define maxBuffer 10 //Quantity of average values


//For calibration, in form y=mx+b
//pH Calibration Values
float add_calibration_pH = 17.6; //b
float multiply_cal_pH = -6.49; //m
//TDS Calibration Values
float add_calibration_TDS = 3.91; //b
float multiply_cal_TDS = 581; //m
//turb Calibration Values
float add_calibration_turb = 0; //b
float multiply_cal_turb = 0; //m


int buffer_pH[maxBuffer], temporal;//Initialize variables
int buffer_TDS[maxBuffer];
int buffer_turb[maxBuffer];
float averageTDSVoltage = 0.0,averagepHVoltage = 0.0,averageturbVoltage = 0.0;
float Celsius=0.0,pHValue=0.0,tdsValue = 0.0,turbValue=0.0;
int bufferIndex = 0,copyIndex = 0;
bool flag = false;
static unsigned long printTimepoint = 0;

OneWire oneWire(pinTemp); //Initialize libraries
DallasTemperature sensors(&oneWire);

int sumVal(int buffer_method[10]){//Method to take the sum of the median data points

  int sum=0;
  int aux;
```

```
  int copyBuffer[10];
  for(copyIndex=0;copyIndex<maxBuffer;copyIndex++){//Copy to other buffer
    copyBuffer[copyIndex]= buffer_method[copyIndex];
  }
  for (int i = 0; i < 9; i++) {
    for (int j = i + 1; j < 10; j++) {
      if (copyBuffer[i] > copyBuffer[j]) {//Order from smallest to greatest
        aux = copyBuffer[i];
        copyBuffer[i] = copyBuffer[j];
        copyBuffer[j]=aux;
      }
    }
  }
  for(int i=2;i<8;i++){//To sum the intermediate data
    sum+=copyBuffer[i];
  }
  return sum;
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200); //Begin communication with computer
  sensors.begin(); //Begin the temperature sensor
  pinMode(TDSSensorPin,INPUT); //TDS, ph, and Turb sensor pins begin
  pinMode(pHSensorPin,INPUT);
  pinMode(turbSensorPin,INPUT);
  printTimepoint = millis(); //Begin measuring time
  if (SD.begin())
  {
    Serial.println("SD card is ready to use.");
  } else
  {
    Serial.println("SD card initialization failed");
    return;
  }
}

void loop() {
  // put your main code here, to run repeatedly:


  if(millis()-printTimepoint > 2000){//every x time, do:
    printTimepoint = millis();//Reset timer
    flag = true;//true: need new data
```

```
  while(flag){//while it needs new data, do:
    static unsigned long analogSampleTimepoint = millis();
    if(millis()-analogSampleTimepoint > 40){//each 40ms
      analogSampleTimepoint = millis();
      buffer_pH[bufferIndex]=analogRead(pHSensorPin);    //Take data from pH, TDS, and Turb
      buffer_TDS[bufferIndex]=analogRead(TDSSensorPin);
      buffer_turb[bufferIndex]=analogRead(turbSensorPin);
      bufferIndex++;
      if(bufferIndex==maxBuffer){//if the vectors get filled
        bufferIndex=0;//restart vector index
        flag = false;//if it already does not need new data, leave the while cycle
      }
    }
  }


  sensors.requestTemperatures(); //measure temperature in degrees celcius
  Celsius=sensors.getTempCByIndex(0);

  float compensationCoefficient=1.0+0.02*(Celsius-25.0);//calculate conductivity compensation
by temperature body
  averageTDSVoltage=(float)sumVal(buffer_TDS)*5.0/(6.0*1024);//calculate average tds
voltage
  float compensationVoltage=averageTDSVoltage/compensationCoefficient;//compensation
voltage tds
  tdsValue=multiply_cal_TDS*compensationVoltage+add_calibration_TDS;//Conductivity in
uS/cm to 25 degrees

  averagepHVoltage=(float)sumVal(buffer_pH)*5.0/(6.0*1024); //Calculate average pH voltage
  pHValue=multiply_cal_pH * averagepHVoltage + add_calibration_pH;//pH final

  averageturbVoltage=(float)sumVal(buffer_turb)*5.0/(6.0*1024);
  turbValue=multiply_cal_turb * averageturbVoltage + add_calibration_turb;//turbidity final

  //Print pods or save to memory

  Serial.println("Calibration:");
  Serial.print(averagepHVoltage,4);
  Serial.print(" V pH, ");
  Serial.print(compensationVoltage,4);
  Serial.print(" V conductivity, ");
  Serial.print(averageturbVoltage, 4);
  Serial.println(" V turbidity");
  Serial.println("Data:");
```

```
    Serial.print(pHValue,4);
    Serial.println(" pH");
    Serial.print(tdsValue,4);
    Serial.println(" µS/cm(conductivity)");
    Serial.print(turbValue, 4);
    Serial.println(" NTU(turbidity)");
    Serial.print(Celsius);
    Serial.println(" °C \n");
    {
     myFile = SD.open("test.txt", FILE_WRITE);
    if (myFile){
    myFile.print(millis());
    myFile.print(",");
    myFile.print(Celsius);
    myFile.print(",");
    myFile.print(pHValue,4);
    myFile.print(",");
    myFile.print(turbValue, 4);
    myFile.print(",");
    myFile.println(tdsValue,4);
    //myFile.print(",");
    myFile.close(); // close the file
    }
  //if the file didn't open, print an error:
    else {
    Serial.println("error opening test.txt");
    }
   delay (500); // use 500 for short term troubleshooting/lab testing,300000 for field testing
  }
  }
}
```