Team 13
"Water Tracker"

Table of Contents

Introduction

The Water Tracker is a device that assists researchers and scientists in gathering data about the quality of water sources by measuring and reporting their pH and turbidity. As the world is joining hands to tackle the impacts of climate change, this device will contribute towards a solution by making data collection of water quality more easily.

By using this device during research, scientists often use pH strips that can only be used once. Thus, the Water Tracker is able to reduce waste from pH strips and help improve the overall environment.

The Water Tracker is a unique tool because despite being able to collect data using current devices, scientists are unable to automatically create graphs to show the datas impact over time. After gathering data from the device, the Water Tracker uses external programs to organize data into useful formats so that scientists can easily create a visual interpretation of the compiled data. Therefore, through the Water Tracker, scientist are able to collect data overtime and the graph allows them to more easily analyze the data.

Materials

1. Turbidity sensor
2. Analog pH sensor kit
3. Connector wires and 1 breadboard
4. Arduino Uno microprocessor
5. Micro USB cable
6. Download Arduino
7. Processing program
8. CoolTerm
9. Google Sheets

Instructions

Let's us start by connecting the two devices (pH and turbidity sensors) to the Arduino Uno.

pH sensor

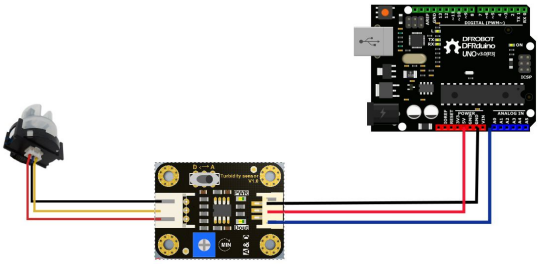
pH sensor measures pH through the voltage generated by the hydrogen concentration of the solution.

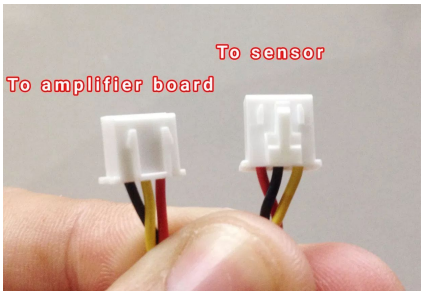
1) Connect 1 wire from the Arduino 5V pin to the positive and 1 from the GND to the negative rail of the breadboard.
2) Connect 1 wire from the "+" rail of the breadboard to the V+ pin of the kit's microprocessor and 1 wire from G pin to negative rail on board.
3) Connect 1 wire from the A0 pin of the Arduino to the Po pin of the probe, this connects the probe to the Arduino and can later be programmed to do what we want.
4) Other pins are irrelevant for this project and will not affect the result.
5) When the Arduino is plugged in, the green LED of the probe's microprocessor will light up.

Pinin		Pinout	
To	Temperature	Supply voltage	5V
Do	Limit pH Signal	Current	5-10mA
Po	Analog GND	Consumption	≤ 0.5 W
G	Supply GND	Temperature	10-50 ºC (this is the range temperature where the probe would be functional)
G	Ground GND	Green LED	Power
V+	Supply (5V)	Red LED	Temperature beyond range

Turbidity sensor

Turbidity is the cloudiness of a fluid caused by particles that rest in or on the fluid. A turbidity sensor detects suspended particles in water by shining a light and measures the light transmittance and scattering rate which changes depending on the total amount of the suspended solids. As the total suspended particles increase, the turbidity also increases.

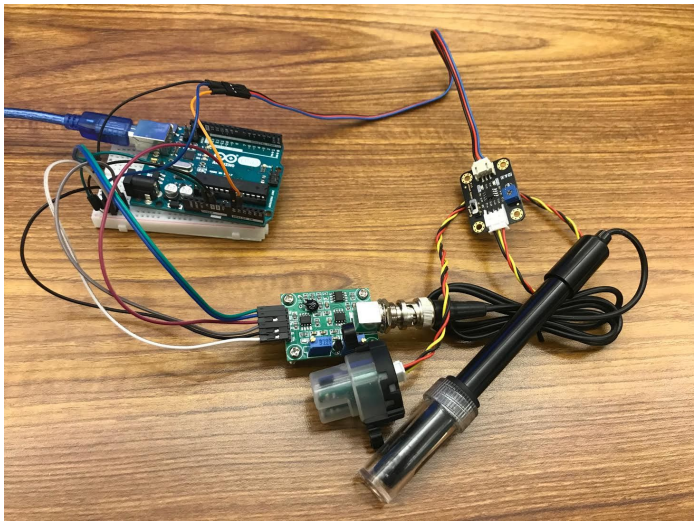




Instructions on connecting the turbidity are provided above, however, instead of connecting directly to the Arduino, the breadboard for both the pH and turbidity probe will share the same Arduino port.

-connect to A1 analog of Arduino

-A red green and yellow LED should light up when connected properly



Arduino Codes

This is the Arduino codes for the Turbidity and pH sensors.

1) The sensors don't measure the actual pH or turbidity thus we have to include an equation that would change voltage difference that is read by the sensors into interpretable units (pH from 1-14) and turbidity (into NTU).

2) The program will give the pH, maximum pH, minimum pH, average pH, and turbidity of the solution being measured.

3) The format of how these values will be printed out on the serial monitor is crucial for they will be save into a text file and imported with every value automatically fit into Google Sheets without them needing to be inputted or fixed manually. Keep in mind that we cannot copy and paste directly from the serial monitor of the Arduino.

4) Round values measured to the tenth place.

IEEE_pH_and_turbidity_sensor_111 | Arduino 1.8.7

File Edit Sketch Tools Help

IEEE_pH_and_turbidity_sensor_111 §

```
#include <EEPROM.h>

//The time between each EEPROM write function call in ms
#define SAMPLE_TIME 2000

const int analogInPin = A0;
int sensorValue = 0;
int sensorValue1 = 0;
const int turbPin=A1;
unsigned long int avgValue;
float b;
int buf[10],temp;
float min = 1000000000000;
float max = 0;
boolean varSet = false;
float volt;
float ntu;

void setup(void) {
 Serial.begin(9600);
 Serial.print(" pH, ");
 Serial.print("  Max, ");
 Serial.print("  Min, ");
 Serial.print("  Avg, ");
 Serial.println("  Turbidity");
}

void loop(void) {
   for(int i=0;i<10;i++)
 {

  buf[i]=analogRead(analogInPin);

  delay(100);
 }
 for(int i=0;i<9;i++)
 {
  for(int j=i+1;j<10;j++)
  {
   if(buf[i]>buf[j])
   {
    temp=buf[i];
    buf[i]=buf[j];
    buf[j]=temp;
   }
  }
 }

 avgValue=0;
 float pHAverage = 0;

 for(int i=2;i<8;i++)
 avgValue+=buf[i];
 float pHVol=(float)avgValue*5.0/1024/6;
 float pHValue = -5.70 * pHVol + 24.34;

 pHAverage += pHValue;

 if(min > pHValue){
    min = pHValue;
 }
 if(max < pHValue){
    max = pHValue;
 }

  Serial.print(pHValue);
  Serial.print(", ");

  //print max and min pH levels overall
  //Serial.print("Highest pH = ");
  Serial.print(max);
  Serial.print(", ");
  //Serial.print("Lowest pH = ");
  Serial.print(min);
  Serial.print(", ");

  //prints the average pH
  //Serial.print("Average pH: ");
  Serial.print(pHAverage);
  Serial.print(", ");

  //message to interpret pH level
  //if(pHAverage > 4 && pHAverage < 9) {
   //Serial.println("Water levels are normal!");}
  //else if (pHAverage <= 4) {
   //Serial.println("Water is too acidic!");
   //}
  //else if (pHAverage >= 9) {
   //Serial.println("Water is too basic!");
   //}
  delay(3000);

  float sensorValue1 = analogRead(turbPin);

  //Serial.print("Turbidity: ");
  Serial.println(sensorValue1 / 500);

  //Serial.println();

  delay (3000);


 }//end loop

 float round_to_dp( float in_value, int decimal_place )
{
  float multiplier = powf( 10.0f, decimal_place );
  in_value = roundf( in_value * multiplier ) / multiplier;
  return in_value;
}
```

CoolTerm

Download CoolTerm, keep in mind what OS you are using; this program will connect to the serial monitor of the Arduino and record all values measured from there.

Click *connection* from the toolbar, and *capture text file*, this would automatically save all the values from the Arduino serial monitor into a text file with the time and date recorded.

Click *connect* to connect to the serial monitor, *start*, and the values will start appear on the program. When you are satisfied with the data gathered, click *disconnect* and this would disconnect from the serial monitor.

Processing: gives visual representation of the pH values being measured by drawing a graph in real time using the input from the sensor.

Google Sheets

- Go to Google Sheets, click on File, choose Import.
- Click on Upload, choose Select a file from your computer
- Find and select the saved text file.
- Upload and choose "commas" to separate the imported texts.
- Choose *explore* in the lower right corner and this would automatically graphall datas from the sheets. Scientists can adjust the range and ask questions about their data.

Conclusion

We are proud of the device and tools we have worked with in this project. We learned a lot about programming, hardware, teamwork, time management, and budgeting.

These are the improvements that we would like to implemented if more resources were in our hands; we wish that future teams can build on this project.

1. A APRS module: An APRS module with a GPS antenna that when included with our device would be able to inform the location of the water sources being measured. When choosing a APRS module with a GPS antenna, be aware GSM frequencies of the module, the default baud rate, and the operating frequency for the GPS satellite.

2. A CO2 and temperature sensors: The inclusion of these two sensors would give the Water Tracker more function applications. Understand that sensors don't take in actual values but rather the voltage or potential difference of the solution that is being measured.

For our project, the greatest challenges were:

First, the pH values being read are not correct with every value always 3 units off. We fixed this problem by changing the equation used to convert the voltage measurement into actual pH measurement.

The second problem was converting the value the turbidity reads into the NTU, or nephelometric turbidity units. We fixed this by including an equation in the program to convert the measurement into NTU.

The third problem was transfering the data from the serial monitor into another program where they can be used by scientists since we cannot copy and paste from serial monitor. We were able to fix this using CoolTerm, a program that can automatically record the measurements into a text file with dates

and time provided for easier organization, and with the correct format to be able to paste on Google Sheets or Excel where the scientists can choose how they would want their datas to be presented.

Our proudest moment was when the text file we saved from CoolTerm was put in the right format into Google Sheets without the needing to fix anything.

References

We would like to thank Dillion Hicks for buying us the materials needed for this project, and let us overshoot the budget a little. We would also like to thank our mentor Bassel Hatoum for helping us with the planning, hardware, and software requirements for the project.

pH sensor

https://scidle.com/how-to-use-a-ph-sensor-with-arduino/

Turbidity sensor

https://www.teachmemicro.com/arduino-turbidity-sensor/