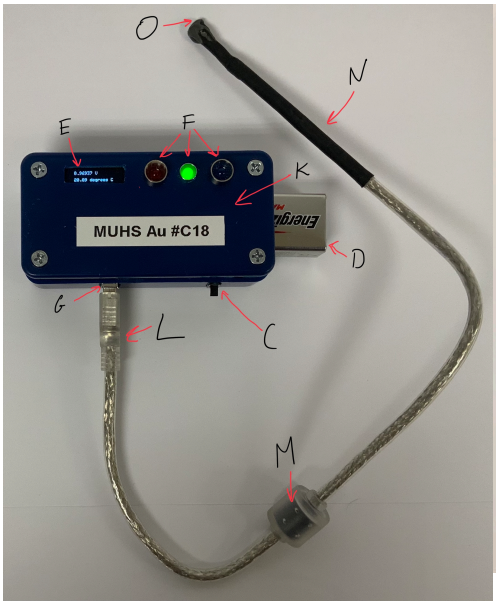
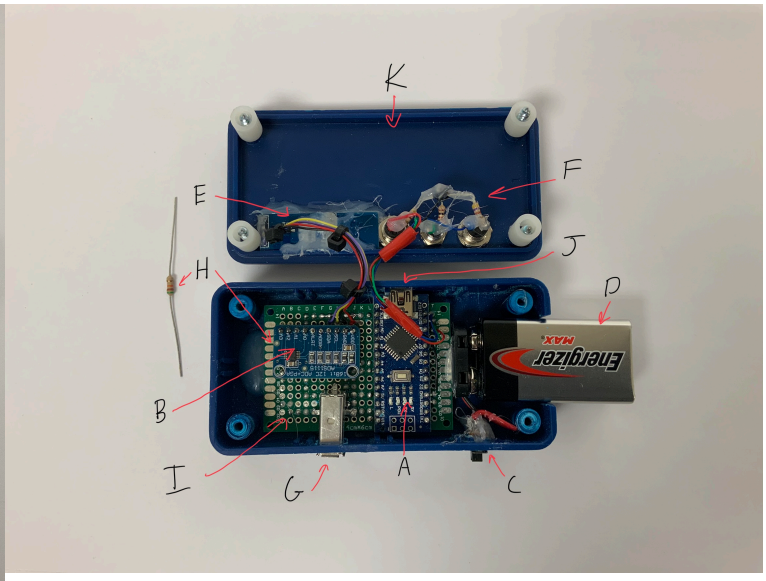
Detector Building Design Log

Marquette University High School Gold #C21

Hannah Rose
Ian Walsh
May 17, 2021

Annotated Photographs





**most pictures have the team labelling from the state tournament - the updated labelling is shown in the auxiliary pictures but the other pictures were not retaken. No other hardware modifications have been made.

Because the lid of the box covers the internal components, two top-down photos have been provided.

A. Arduino Nano Microprocessor Board:

- Regulates the 9 volt battery down to 5 volts to power the ADC, display, and LEDs and down to 3.3 volts to power the sensor
- Gathers the signal from the ADC, converts it to a voltage, and then computes the temperature using the model
- Controls the display to output the voltage and temperature
- Determines which LEDs should be lit depending on the temperature range and powers them

B. ADS1115 16 Bit ADC (analog to digital converter) with PGA (programmable gain amplifier):

- Measures the voltage the sensor produces and converts it to a digital format the Arduino can interpret
- The PGA serves primarily to buffer the sensor input so that the ADC does not affect the sensor voltage while it is taking a measurement (the gain is set to one).

C. Power Switch:

- Disconnects the battery from the circuit when the device is not in use

D. Standard 9 Volt Energizer MAX Battery:

- Supplies power to the circuitry

E. SSD1306 128X32 OLED Display:

- Displays the measured voltage and temperature

F. LEDs and current limiting resistors (470Ω for red and blue and 100Ω for green):

- The LEDs indicate which temperature range the measurement is in
- The current limiting resistors prevent the LEDs from overheating or damaging themselves by limiting the amount of current they can draw

G. FireWire port:

- provides an easy method to attach and detach the sensor while ensuring a clean electrical connection

H. 51 kΩ reference resistor:

- Acts as the fixed resistor in the voltage divider (the variable resistor is the thermistor)
- This causes the voltage to vary depending on the resistance of the thermistor $V_{\text{out}} = V_{\text{in}} \cdot \frac{R_{\text{Reference}}}{R_{\text{Reference}} + R_{\text{Thermistor}}}$
- Note: the resistor is obscured beneath the ADC so a copy of the component is shown to the side of the box

I. Soldered Prototyping board:

- Provides mechanical structure to the circuit that can be mounted in the box
- Wires and solder on the back side of the board form the electrical connections between components

J. USB mini port:

- Allows the computer to connect to the Arduino for programming and serial output when necessary

K. Project Box:

- Contains and protects the circuitry
- Provides structure for mounting the components
- Makes the device aesthetically pleasing

L. FireWire Cable:

- Connects to the FireWire port on the device
- Internal conductors connect the FireWire port to the thermistor at its end
- The braided shielding tied to the device's ground decreases external electromagnetic interference.

M. Ferrite Bead:

- Decreases high frequency electromagnetic noise in the cable if present

N. Heat Shrink Tubing:

- Helps waterproof the sensor by covering the section where the insulation was removed to attach the thermistor

O. Epoxy and NTC (negative temperature coefficient) Thermistor (hidden by epoxy):

- The epoxy helps waterproof the sensor by sealing the end of the cable around the thermistor

- The NTC Thermistor provides a resistance that varies with temperature so that it, in conjunction with the reference resistor, can produce a voltage that predictably varies with temperature

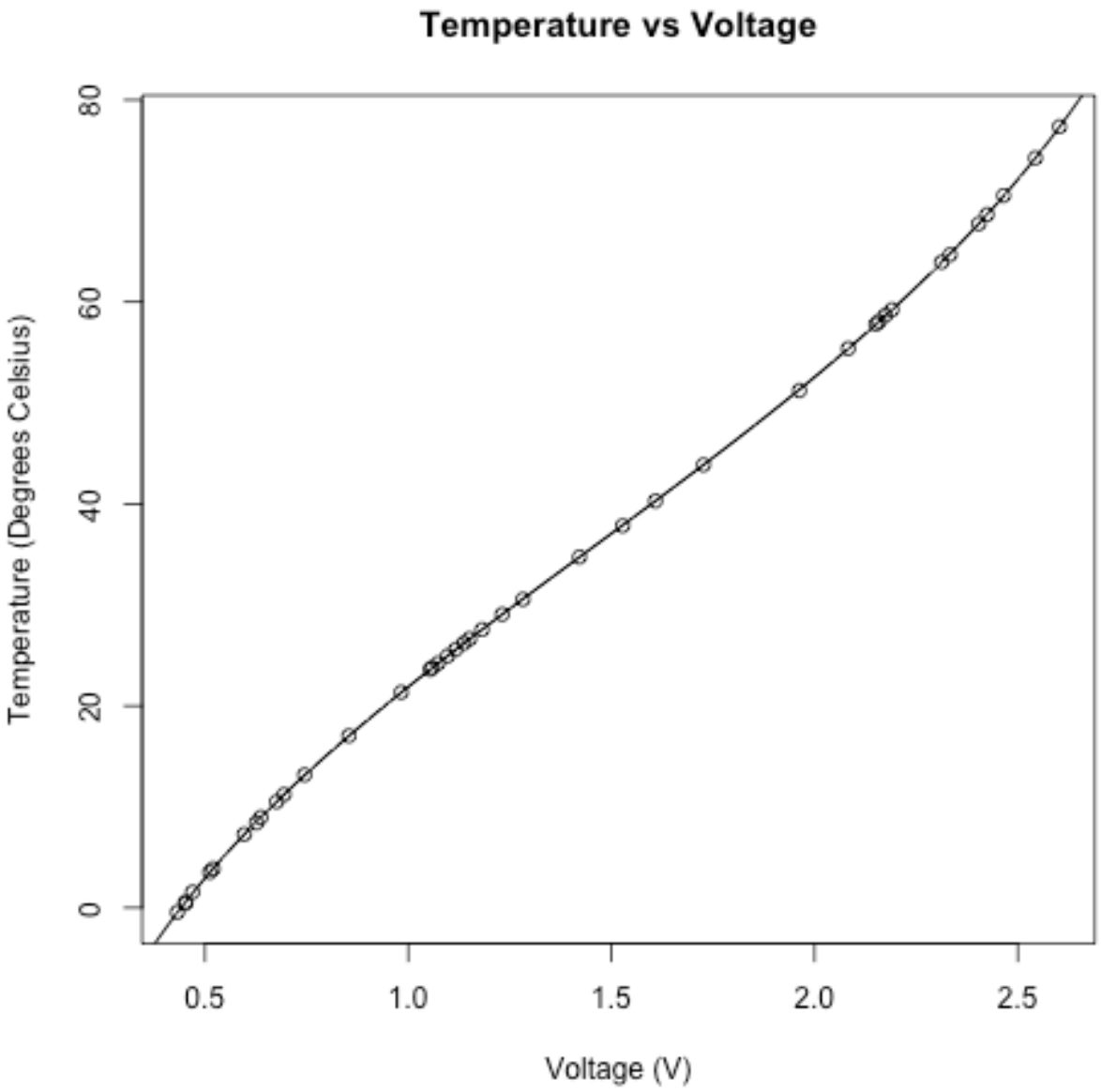
The device was constructed by soldering components to a soldered prototyping board. The required electrical connections were then made using solder and wires attached to the back of the board. Components that were attached to the box were secured into cutouts using locking nuts or hot glue. Before final assembly, wires were run from these components onto the board. The sensor cable began as a FireWire cable that was split in half. Between two of its conductors, the thermistor was then soldered. To waterproof the sensor, the heat shrink tubing was shrunk around the end of the cable and epoxy sealed all possible entrance points for water.

Data Table

Trial #	Sensor Voltage (Volts)	Temperature Value (Degrees C)
1	0.43190	-0.44
2	0.45176	0.43
3	0.45255	0.58
4	0.46938	1.56
5	0.51176	3.52
6	0.51985	3.87
7	0.59674	7.28
8	0.62651	8.41
9	0.63740	8.96
10	0.67610	10.50
11	0.69441	11.22
12	0.74511	13.17
13	0.85421	17.04
14	0.98271	21.34
15	1.05422	23.62
16	1.05920	23.78
17	1.07328	24.22
18	1.09680	24.94
19	1.11690	25.58
20	1.13701	26.21
21	1.15130	26.65
22	1.18191	27.55
23	1.23140	29.06
24	1.28232	30.55

Trial #	Sensor Voltage (Volts)	Temperature Value (Degrees C)
25	1.42125	34.74
26	1.52756	37.84
27	1.60850	40.29
28	1.72656	43.87
29	1.96300	51.23
30	2.08300	55.39
31	2.15120	57.80
32	2.15840	58.07
33	2.17430	58.68
34	2.19040	59.23
35	2.31260	63.94
36	2.33310	64.68
37	2.40400	67.74
38	2.42440	68.65
39	2.46500	70.54
40	2.54330	74.26
41	2.60260	77.35

Graph / Model



The line is the graph of the mathematical function, circles represent trials.

Equation:

$$T = \frac{1}{a + b \cdot \ln\left(51000 \cdot \left(\frac{3.3}{V} - 1\right)\right) + c \cdot \left(\ln\left(51000 \cdot \left(\frac{3.3}{V} - 1\right)\right)\right)^3}$$

$a = 0.0007495489110667 15$

$b = 0.0002104513301989 89$

$c = 1.153710513855 78 \cdot 10^{-7}$

RMSE: $0.042939 \deg C$

Code Printout with Voltage to Temperature Highlighted

```
//
// Thermometer.ino
//
// MUHS Au #C18
//

// Libraries to connect to the ADC and Display over I2C
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_ADS1015.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Define parameters for the Display
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32

#define OLED_RESET 12
Adafruit_SSD1306 screen(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
 OLED_RESET);

// Define parameters for the ADC
Adafruit_ADS1115 adc(0x4A);

// Calibration Constants
double a = 0.0007495489110667l5;
double b = 0.0002104513301989;
double c = 1.153710513855780-07;
double R0 = 51000;

// LED Ranges
// each range is an array specifying {start point, end point, red
 value, green value, blue value}
double ranges[][5] =
 {{0,10,0,0,255},{10,15,0,255,255},{15.0,30.0,0,255,0}, {30, 45,
 255, 255, 0}, {45, 1000, 255, 0, 0}};
int countRanges = sizeof(ranges)/sizeof(ranges[0]);

// Temp Variables
//  adc variables
long adcTotalRef = 0;
long adcTotalData = 0;
//  Temperature calculation variables
double A = 0;
double R = 0;
```

```
double L = 0;
double K = 0;
// LED color variables
int red = 0;
int green = 0;
int blue = 0;
//  Measurement Variables
double a0;
double T;
//  loop increment variable
int i;

// Single Sample voltage measurement
double readADC()
{
  int16_t Vref = adc.readADC_Differential_0_1(); // Measuring the
   3.3V rail
  int16_t Vdif = adc.readADC_Differential_2_3(); // Measuring the
   data line
  return 3.3*Vdif/Vref; // normalizing to 0-3.3V
}
// average of n samples voltage measurement
double readADCn(int num)
{
    adcTotalRef = 0;
    adcTotalData = 0;
    for (i = 0; i < num; i ++) {
        adcTotalRef += adc.readADC_Differential_0_1(); //
         Measuring the 3.3V rail
        adcTotalData += adc.readADC_Differential_2_3(); //
         Measuring the data line
    }
    return 3.3*adcTotalData/adcTotalRef; // normalizing to 0-3.3V
}

// Convert Voltage to Temperature
double H(double V)
{
  A = 3.3/V; // 1/V
  R = R0*(A-1); // calculate resistance of the thermistor
  L = log(R); // ln(R)
  K = 1/(a+b*L+c*L*L*L); // steinhart-hart equation
  return K-273.15; // convert kelvin to celsius
}
```

```
// Set the Red Green and Blue LEDs to the specified values
void RGB(int R, int G, int B)
{
  analogWrite(9,R);
  analogWrite(10,G);
  analogWrite(11,B);
}

// Initialize the device
void setup(void)
{
  Serial.begin(9600); // open a serial port for debugging to the
   computer

  for(int i=9; i<12; i++) pinMode(i, OUTPUT); // Configure LED
   pins for output

  if(!screen.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C
   for 128x32
    Serial.println(F("SSD1306 allocation failed"));
  }

  screen.display(); // Start the OLED display
  delay(1000); // Pause

  adc.begin(); // start the ADC
  adc.setGain(GAIN_ONE); // set the gain on the ADC

  // Configure the typestting on the OLED display
  screen.clearDisplay();
  screen.display();
  screen.setTextSize(1);
  screen.setTextColor(WHITE);
  screen.cp437(true);
}

// Sample, calculate, display loop
void loop()
{
  // read the input voltage, taking 1 sample, with the possibility
   of averaging more to reduce noise
  double anew = readADCn(1);
  // if the reading has changed significantly, update to the new
   reading
  if ((a0-anew)>0.1 || (a0-anew)<-0.1) {
```

```
      a0 = anew;
  } else {
    // otherwise add the new value to the exponential moving
     average to smooth small fluxuations
      double k = 0.1;
      a0 *= 1-k;
      a0 += anew*k;
  }

  // call the function to convert the voltage to a temperature
  T = H(a0);

  // Clear the display and output the voltage and temperature
  screen.clearDisplay();
  screen.setCursor(20,5);
  screen.print(a0, 5); screen.println(" V");
  screen.setCursor(20,20);
  screen.print(T); screen.println(" degrees C");
  screen.display();

  // default all LEDs to off
  red = 0;
  green = 0;
  blue = 0;
  // determine which LEDs should be lit
  for (i = 0; i < countRanges; i++) { // For each range
      if (T >= ranges[i][0] && T < ranges[i][1]) { // if
       temperature in range
        // set the appropriate colors
          red = (int) ranges[i][2];
          green = (int) ranges[i][3];
          blue = (int) ranges[i][4];
      }
  }
  // Output the colors to the LEDs
  RGB(red,green,blue);
}
```

Code Printout with LED code Highlighted

```
//
// Thermometer.ino
//
// MUHS Au #C18
//

// Libraries to connect to the ADC and Display over I2C
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_ADS1015.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Define parameters for the Display
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32

#define OLED_RESET 12
Adafruit_SSD1306 screen(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
 OLED_RESET);

// Define parameters for the ADC
Adafruit_ADS1115 adc(0x4A);

// Calibration Constants
double a = 0.000749548911066715;
double b = 0.000210451330198989;
double c = 1.15371051385578e-07;
double R0 = 51000;

// LED Ranges
// each range is an array specifying {start point, end point, red
 value, green value, blue value}
double ranges[][5] =
 {{0,10,0,0,255},{10,15,0,255,255},{15.0,30.0,0,255,0}, {30, 45,
 255, 255, 0}, {45, 1000, 255, 0, 0}};
int countRanges = sizeof(ranges)/sizeof(ranges[0]);

// Temp Variables
//  adc variables
long adcTotalRef = 0;
long adcTotalData = 0;
//  Temperature calculation variables
double A = 0;
double R = 0;
```

```
double L = 0;
double K = 0;
// LED color variables
int red = 0;
int green = 0;
int blue = 0;
//  Measurement Variables
double a0;
double T;
//  loop increment variable
int i;

// Single Sample voltage measurement
double readADC()
{
  int16_t Vref = adc.readADC_Differential_0_1(); // Measuring the
   3.3V rail
  int16_t Vdif = adc.readADC_Differential_2_3(); // Measuring the
   data line
  return 3.3*Vdif/Vref; // normalizing to 0-3.3V
}
// average of n samples voltage measurement
double readADCn(int num)
{
    adcTotalRef = 0;
    adcTotalData = 0;
    for (i = 0; i < num; i ++) {
        adcTotalRef += adc.readADC_Differential_0_1(); //
         Measuring the 3.3V rail
        adcTotalData += adc.readADC_Differential_2_3(); //
         Measuring the data line
    }
    return 3.3*adcTotalData/adcTotalRef; // normalizing to 0-3.3V
}

// Convert Voltage to Temperature
double H(double V)
{
  A = 3.3/V; // 1/V
  R = R0*(A-1); // calculate resistance of the thermistor
  L = log(R); // ln(R)
  K = 1/(a+b*L+c*L*L*L); // steinhart-hart equation
  return K-273.15; // convert kelvin to celsius
}
```

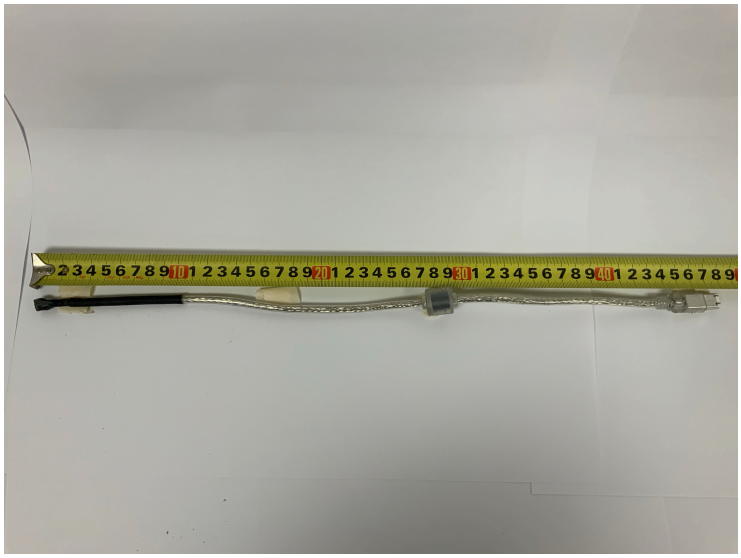
```
// Set the Red Green and Blue LEDs to the specified values
void RGB(int R, int G, int B)
{
  analogWrite(9,R);
  analogWrite(10,G);
  analogWrite(11,B);
}

// Initialize the device
void setup(void)
{
  Serial.begin(9600); // open a serial port for debugging to the
   computer

  for(int i=9; i<12; i++) pinMode(i, OUTPUT); // Configure LED
   pins for output

  if(!screen.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C
   for 128x32
    Serial.println(F("SSD1306 allocation failed"));
  }

  screen.display(); // Start the OLED display
  delay(1000); // Pause

  adc.begin(); // start the ADC
  adc.setGain(GAIN_ONE); // set the gain on the ADC

  // Configure the typestting on the OLED display
  screen.clearDisplay();
  screen.display();
  screen.setTextSize(1);
  screen.setTextColor(WHITE);
  screen.cp437(true);
}

// Sample, calculate, display loop
void loop()
{
  // read the input voltage, taking 1 sample, with the possibility
   of averaging more to reduce noise
  double anew = readADCn(1);
  // if the reading has changed significantly, update to the new
   reading
  if ((a0-anew)>0.1 || (a0-anew)<-0.1) {
```

```
      a0 = anew;
  } else {
    // otherwise add the new value to the exponential moving
     average to smooth small fluxuations
      double k = 0.1;
      a0 *= 1-k;
      a0 += anew*k;
  }

  // call the function to convert the voltage to a temperature
  T = H(a0);

  // Clear the display and output the voltage and temperature
  screen.clearDisplay();
  screen.setCursor(20,5);
  screen.print(a0, 5); screen.println(" V");
  screen.setCursor(20,20);
  screen.print(T); screen.println(" degrees C");
  screen.display();

  // default all LEDs to off
  red = 0;
  green = 0;
  blue = 0;
  // determine which LEDs should be lit
  for (i = 0; i < countRanges; i++) { // For each range
      if (T >= ranges[i][0] && T < ranges[i][1]) { // if
       temperature in range
        // set the appropriate colors
          red = (int) ranges[i][2];
          green = (int) ranges[i][3];
          blue = (int) ranges[i][4];
      }
  }
  // Output the colors to the LEDs
  RGB(red,green,blue);
}
```

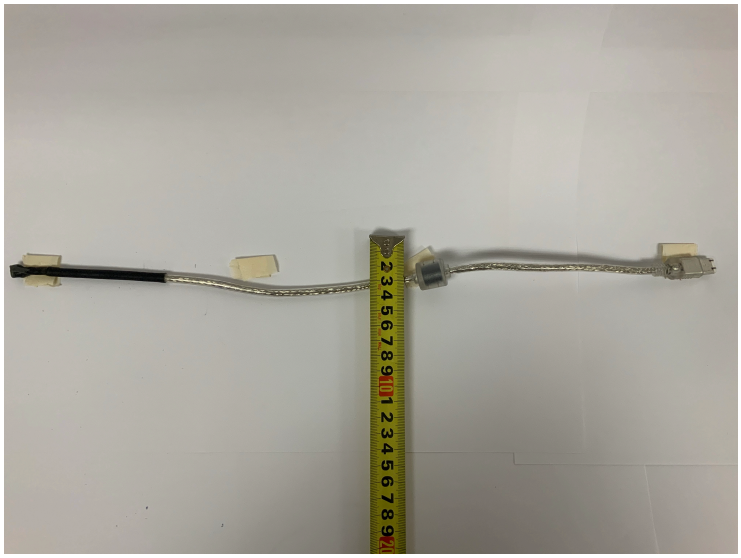
End of Device Log

Beginning of Auxiliary Pictures

Length of Sensor (Full length of cable can be submerged)



Width of Sensor



Labelling of Device

Circuit Diagram