AD8495 Breakout Board Solution

by liverdye

Hey there! I purchased a bunch of AD8495 breakout boards to record (5) type-K thermocouples. These were the perfect match for my range of temp and application - or so I thought...My readings were fluctuating and inaccurate (51% error!).

After running some calibration tests, lots of swearing, crying, and acceptance of my doom (i kid!)- I found a solution to get **precise and accurate** readings from the AD8495 breakout board (ADBB). My colleagues and I worked on this together, so they deserve some credit too!

Now if you are having similar issues with your board, try this out. To keep things straightforward, I abbreviated my components. Below is a "glossary" of those abbreviations.

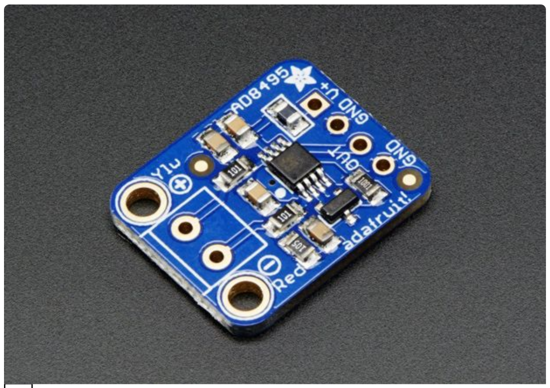
Glossary:

TC - Omega type-k thermocouple connected to ADBB.

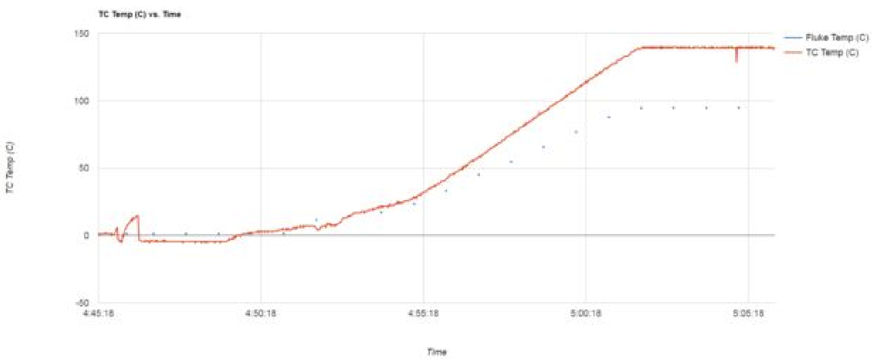
FTC - Fluke 179 type-k thermocouple.

CTC - controlled, reliable type-k thermocouple device

ADBB - ad8495 breakout board



1. Taken from Adafruit website, this is not my photo.

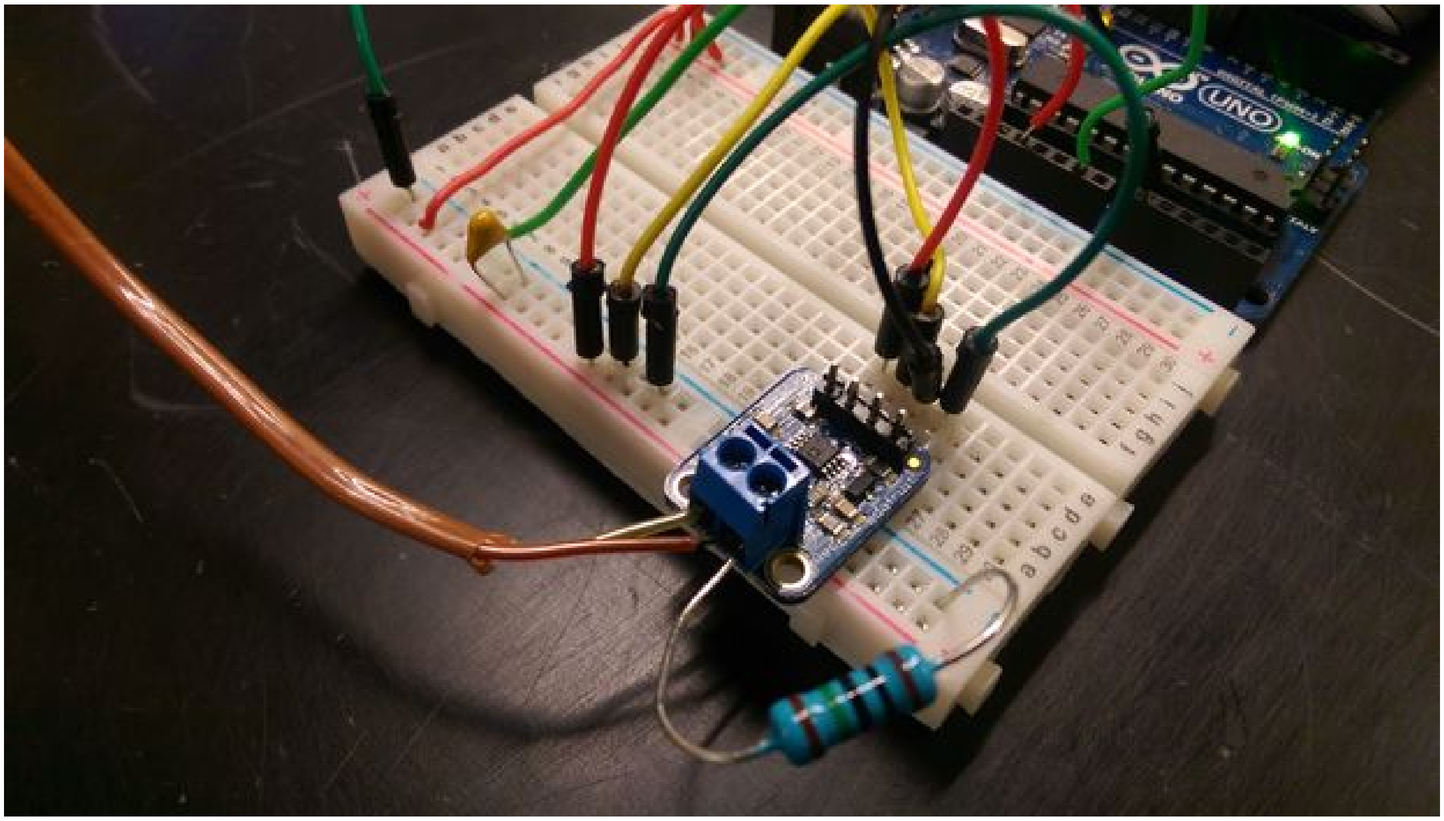


Step 1: Precision

My readings were fluctuating up to +/- 10 degrees. In order to make the ADBB precise, a simple low pass filter will make a world of difference.

Do this by placing a .1uF capacitor and a 1M Ohm resistor in your circuit (see photo). If you have a 1uF capacitor, that will work too.

That is it for precision. The addition of the RC low pass filter should clean up your signal A LOT.



Step 2: Accuracy

Objective:

In this next part, you will be conducting some tests using the equipment listed below. The purpose is to measure a "wide" temperature range using a reliable thermocouple (CTC) and the ADBB. Data from both thermocouples will be used to calculate a slope intercept equation. **I will include all the calculation steps!**

Equipment:

-At least (1) reliable type-k thermocouple measurement device. (I'm using Fluke 179 with type-K TC).
-Type k TC with ADBB
-Hot plate to boil water
-Glass container for boiling water
-Crushed ice
-DAQ equipment (Arduino & Python software in this example)

Procedure:

1. Initialize your DAQ system. For me, I use the Serial Comm. to send data from Arduino to Python, which compiles the data in a excel file w/ headers. Mark your start time.

2. Start with an ice bath (90% crushed ice, 10% water) on a cooled hot plate

3. Place both thermocouples in the container and turn on hot plate to around 150 deg C.

4. Make sure both thermocouples (TC & CTC) are in the center of the container - **not touching the sides or bottom!! (see pictures)**

5. Every 30-60 seconds (I prefer 30 sec for more samples), write down the temperature of the CTC. In this example, I'm reading it from my Fluke's LCD.

6. Repeat until water is boiling (about 17 minutes) and the CTC reaches steady state of 95-100 deg C (depending on your altitude). The ADBB should be at steady state as well, it just won't be accurate.

Calculations:

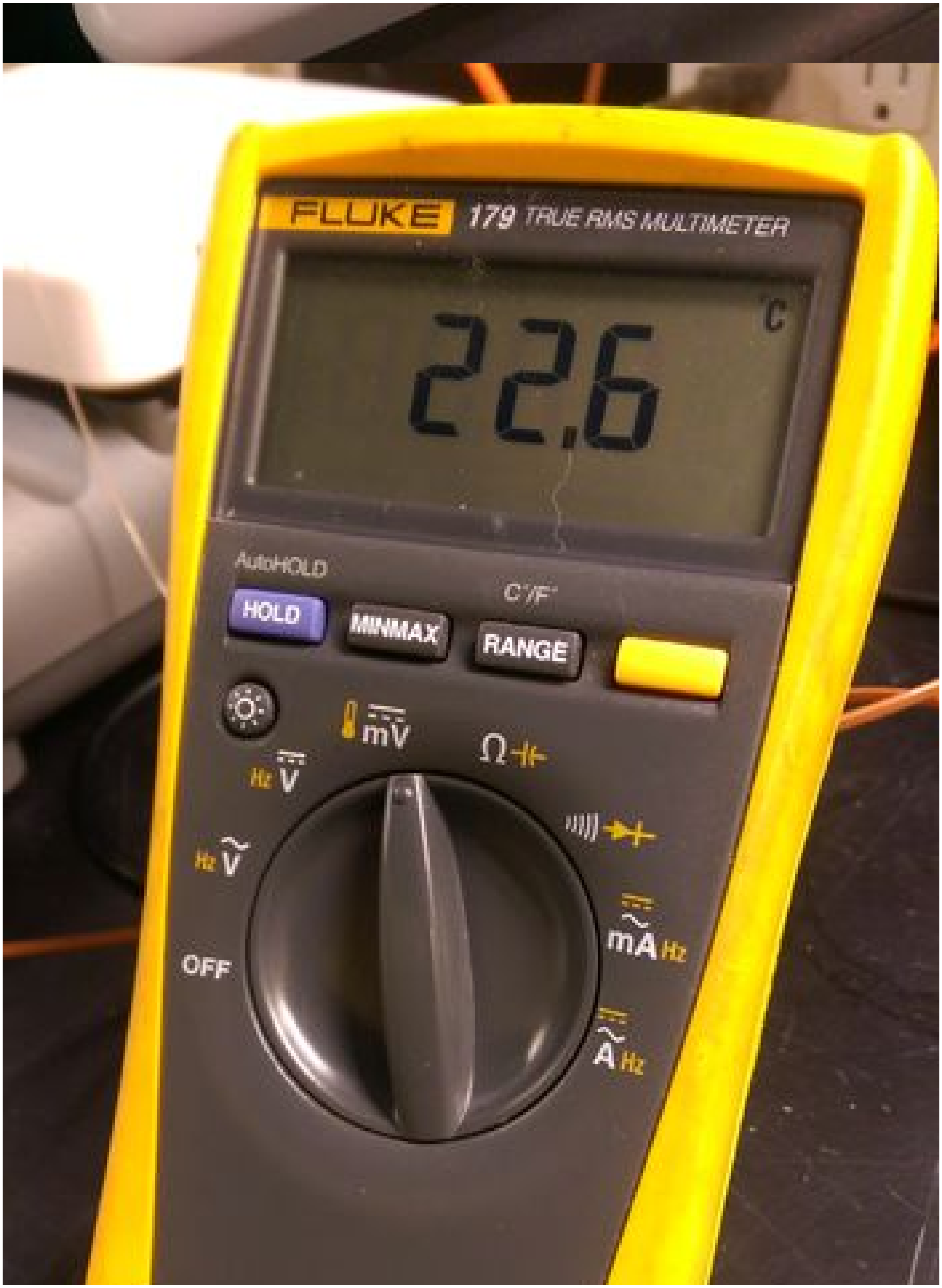
Your main variables will be temperature and voltage, in fact your temperature is a function of voltage, **T(V).** If you want to test for linearity, plot your ADBB temp vs voltage (see photos).

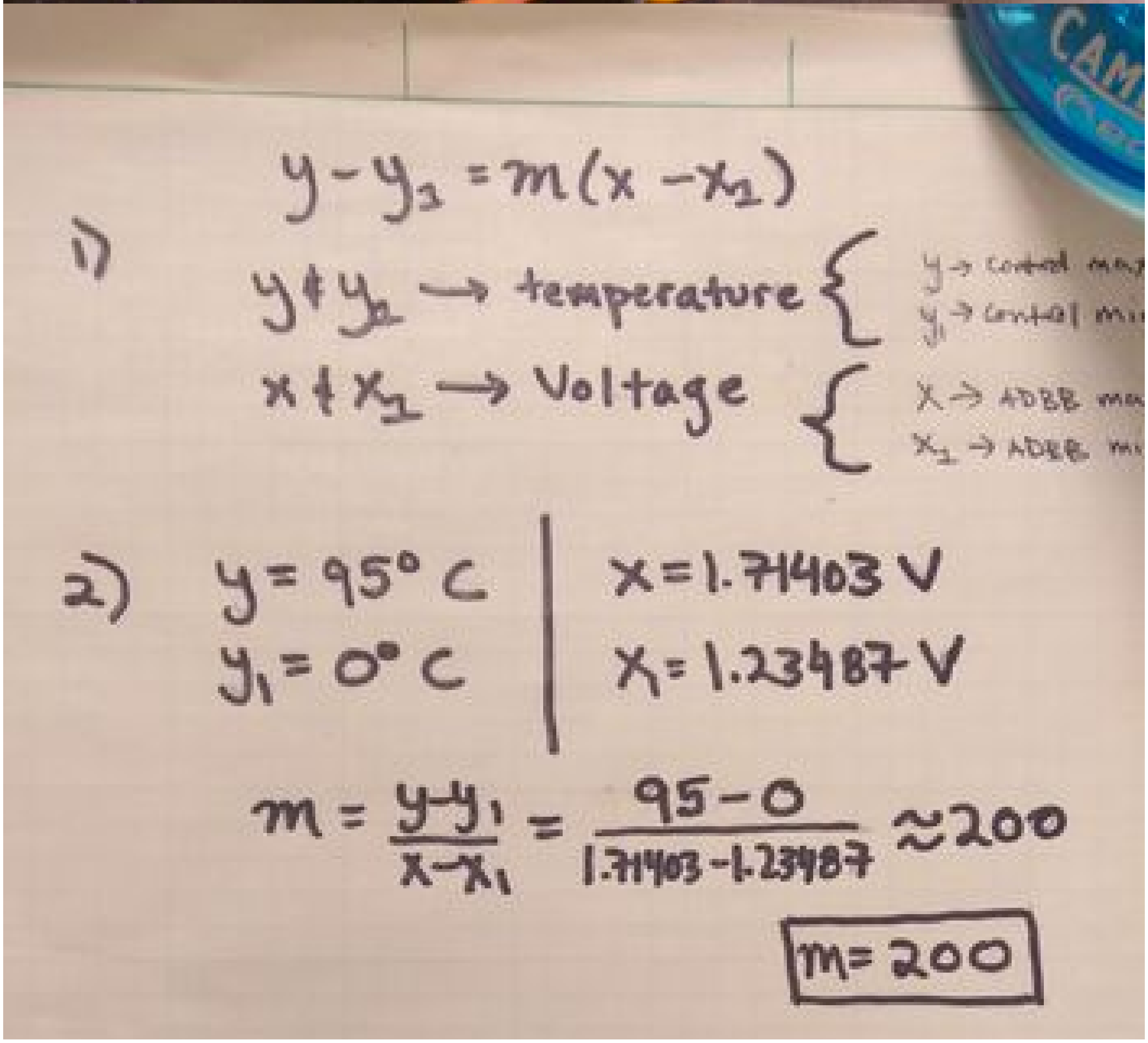
"Y = Control max" : this should be 95-100 deg C, read from your CTC.
"Y1= Control min" : this should be 0 deg C (the temp of an ice bath, but verify with your CTC).

"X = ADBB max " : this should be voltage read from Arduino once boiling point and steady state is reached (about 1.71 for me).

"X1= ADBB min" : voltage when TC is in ice bath, mine came out to be 1.23 V.

With those values, see the photo for the calculations and the solution to this problem!!!


FLUKE
179 TRUE RMS MULTIMETER
22.6
°C
AutoHOLD
HOLD
MINMAX
C°/F°
RANGE
mV
Hz V
Hz V
OFF
Ω
mA Hz
A Hz



$$y - y_1 = m(x - x_1)$$

1) y & $y_1 \longrightarrow$ temperature $\left\{ \begin{array}{l} y \rightarrow \text{control ma} \\ y_1 \rightarrow \text{control mi} \end{array} \right.$

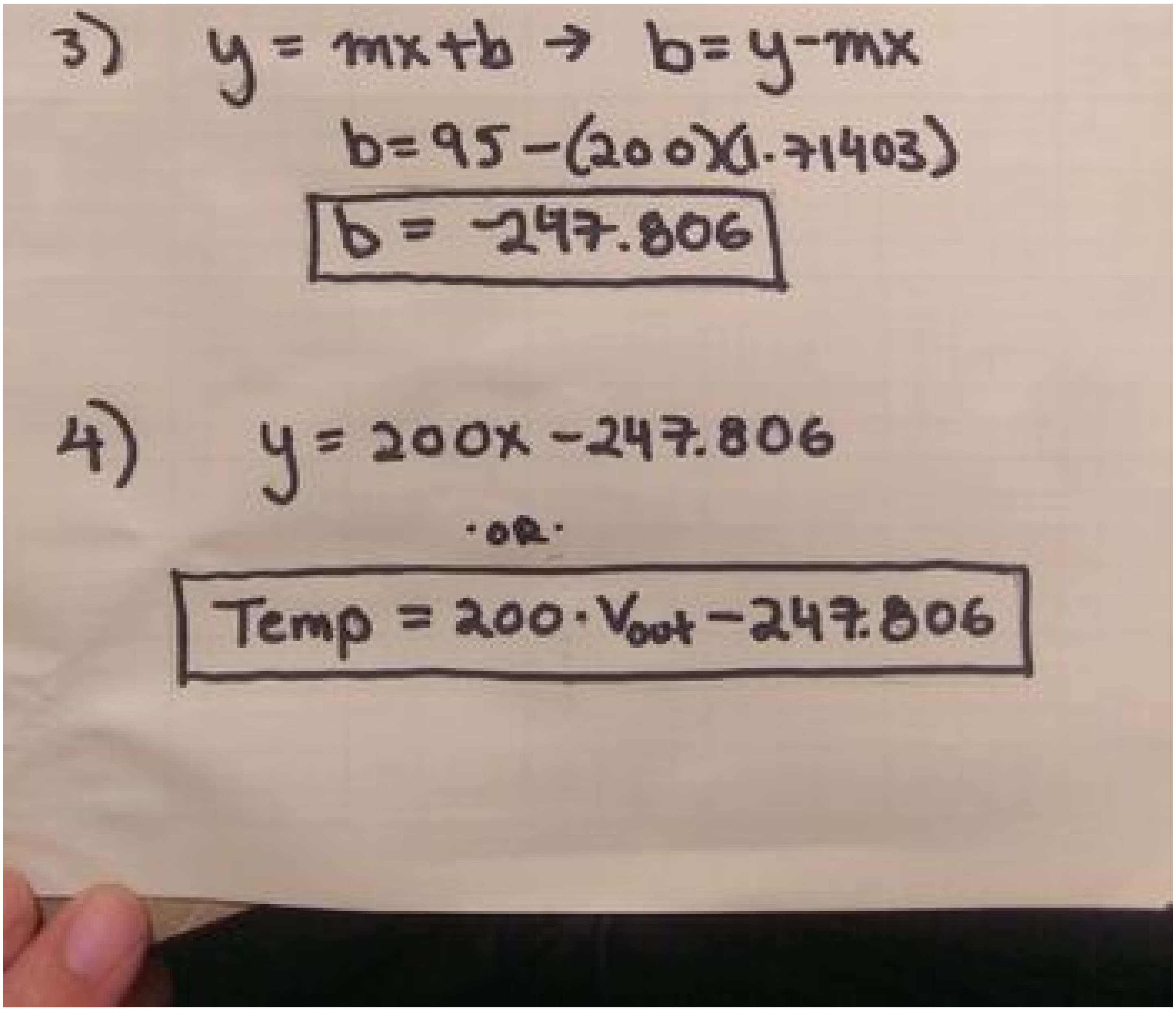
x & $x_1 \longrightarrow$ Voltage $\left\{ \begin{array}{l} x \rightarrow \text{AD8B ma} \\ x_1 \rightarrow \text{AD8B mi} \end{array} \right.$

2) $y = 95°C$ | $x = 1.71403\,V$

$y_1 = 0°C$ | $x_1 = 1.23487\,V$

$$m = \frac{y - y_1}{x - x_1} = \frac{95 - 0}{1.71403 - 1.23487} \approx 200$$

$$\boxed{m = 200}$$



3) $y = mx + b \rightarrow b = y - mx$

$b = 95 - (200)(1.71403)$

$$\boxed{b = -247.806}$$

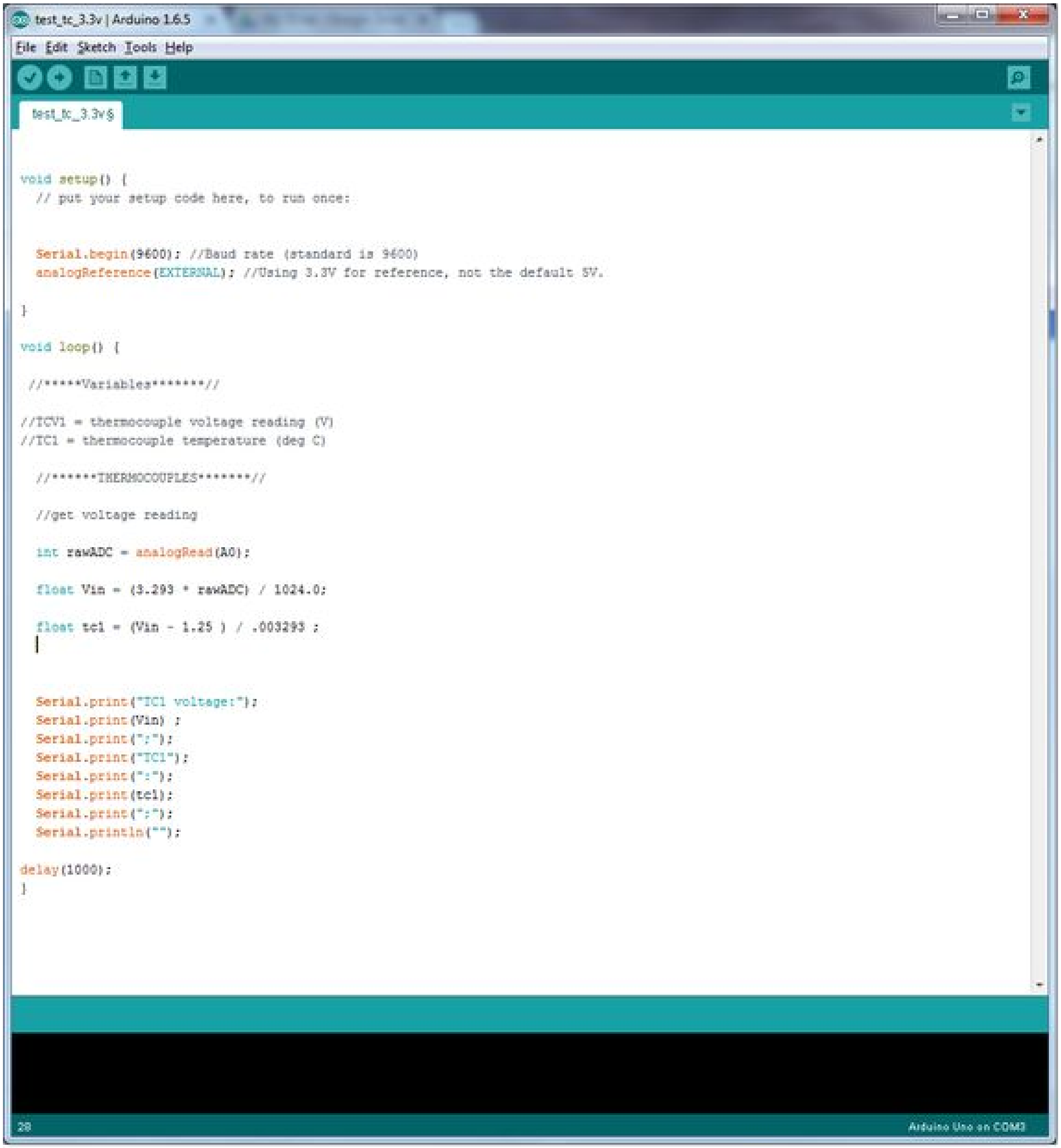
4) $y = 200x - 247.806$

·OR·

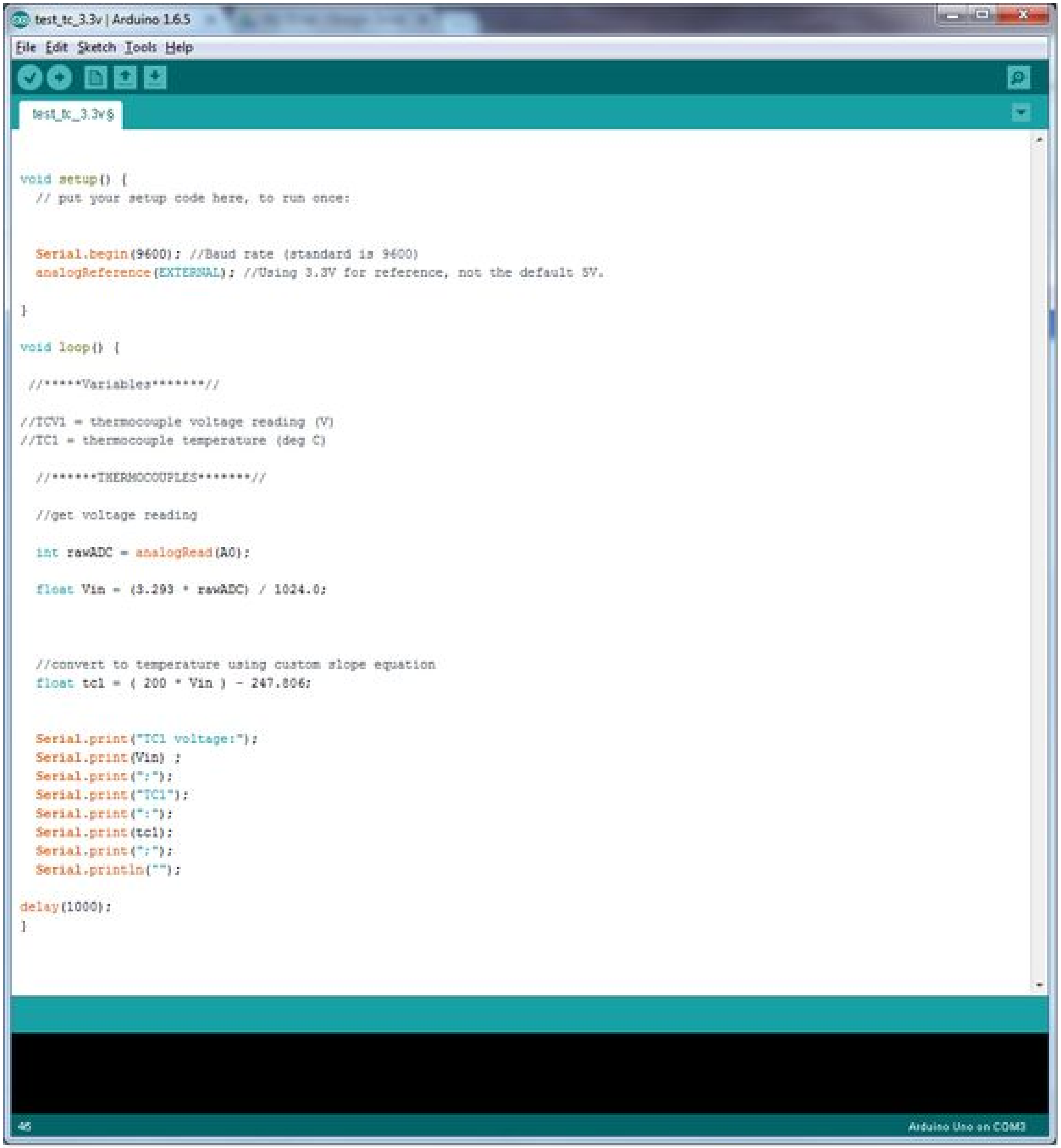
$$\boxed{Temp = 200 \cdot V_{out} - 247.806}$$

Step 3: Software

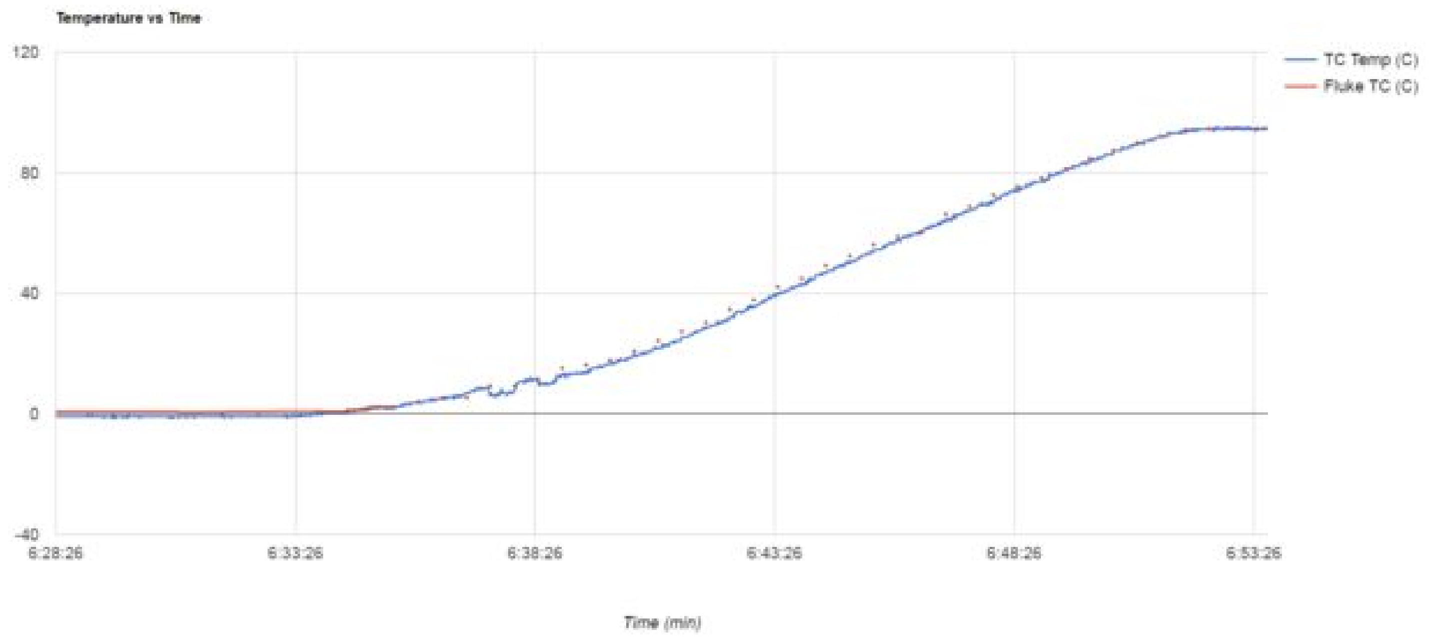
With the new equation, you will need to edit your Arduino code (or whatever you're using). I've included screen shots of before and after. Also, I've included my new tests results with the new equation.



```
void setup() {
  // put your setup code here, to run once:


  Serial.begin(9600); //Baud rate (standard is 9600)
  analogReference(EXTERNAL); //Using 3.3V for reference, not the default 5V.

}

void loop() {

 //*****Variables*******//

//TCV1 = thermocouple voltage reading (V)
//TC1 = thermocouple temperature (deg C)

  //******THERMOCOUPLES*******//

  //get voltage reading

  int rawADC = analogRead(A0);

  float Vin = (3.293 * rawADC) / 1024.0;

  float tc1 = (Vin - 1.25 ) / .003293 ;



  Serial.print("TC1 voltage:");
  Serial.print(Vin) ;
  Serial.print(":");
  Serial.print("TC1");
  Serial.print(":");
  Serial.print(tc1);
  Serial.print(":");
  Serial.println("");

delay(1000);
}
```



```
void setup() {
  // put your setup code here, to run once:


  Serial.begin(9600); //Baud rate (standard is 9600)
  analogReference(EXTERNAL); //Using 3.3V for reference, not the default 5V.

}

void loop() {

 //*****Variables*******//

//TCV1 = thermocouple voltage reading (V)
//TC1 = thermocouple temperature (deg C)

  //******THERMOCOUPLES*******//

  //get voltage reading

  int rawADC = analogRead(A0);

  float Vin = (3.293 * rawADC) / 1024.0;



  //convert to temperature using custom slope equation
  float tc1 = ( 200 * Vin ) - 247.806;


  Serial.print("TC1 voltage:");
  Serial.print(Vin) ;
  Serial.print(":");
  Serial.print("TC1");
  Serial.print(":");
  Serial.print(tc1);
  Serial.print(":");
  Serial.println("");

delay(1000);
}
```



Step 4: Conclusion

I tested the equation by plugging in 1.71 V and my output was 94.2 deg C. That is .86% error from 95 deg C! Much improvement from 51% error that I was receiving before.

- Comparing your ADBB data to reliable data from a controlled TC will help you find the new equation that will give you **accurate readings** (read: scrap the equation that is printed on the back of the AD8495 breakout board).
- Applying an RC low pass filter will give you **precise readings.**
- Combining both methods will solve your issues with the AD8495 breakout board that I encountered.

Follow the process I outlined and replace the equation on the back of AD8495 with the equation you develop from the process. If you don't have a hot plate, a pot of water on your stovetop will work too.

Thank you for your time, I hope this helps you with your projects. Please feel free to ask questions and I will do my best to answer. Happy tinkering!