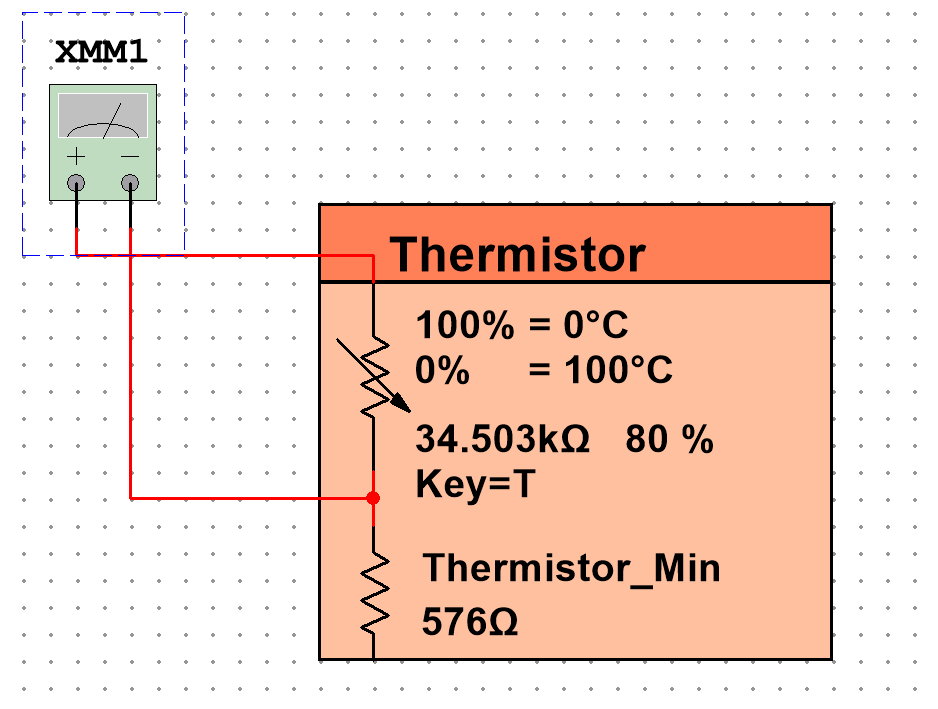
Engineering challenges:

Multisim

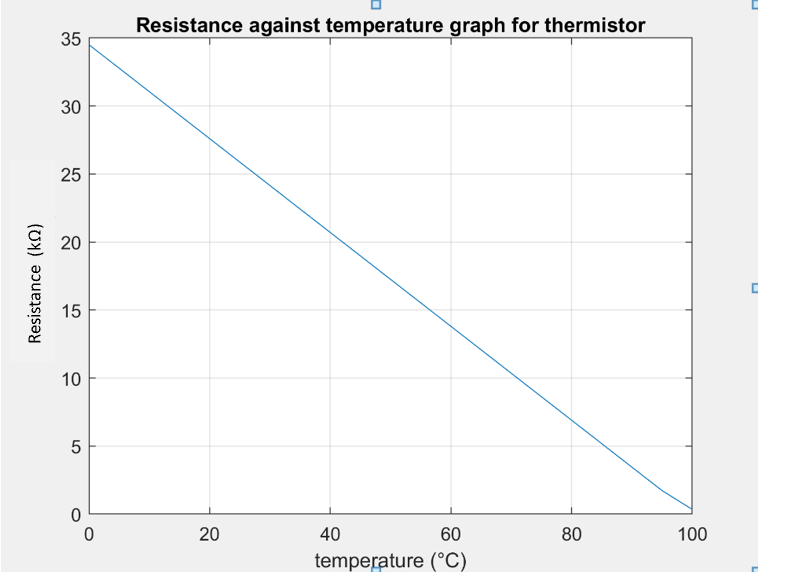
Callibration of thermistor:



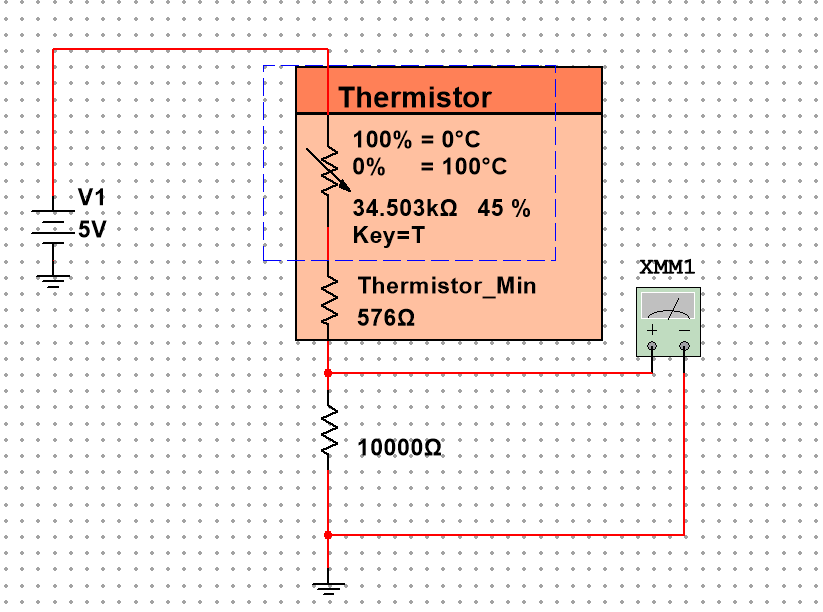
Connect DMM to thermistor as shown. Vary the temperature to find the different values of resistance for different temperatures.

|  |  |
| --- | --- |
| **Temp (°C)** | **Resistance (kΩ)** |
| 0 | 34.503 |
| 5 | 32.778 |
| 10 | 31.053 |
| 15 | 29.327 |
| 20 | 27.602 |
| 25 | 25.877 |
| 30 | 24.152 |
| 35 | 22.427 |
| 40 | 20.702 |
| 45 | 18.977 |
| 50 | 17.251 |
| 55 | 15.526 |
| 60 | 13.801 |
| 65 | 12.076 |
| 70 | 10.351 |
| 75 | 8.626 |
| 80 | 6.901 |
| 85 | 5.175 |
| 90 | 3.450 |
| 95 | 1.725 |
| 100 | 0.345 |

This produced this graph:



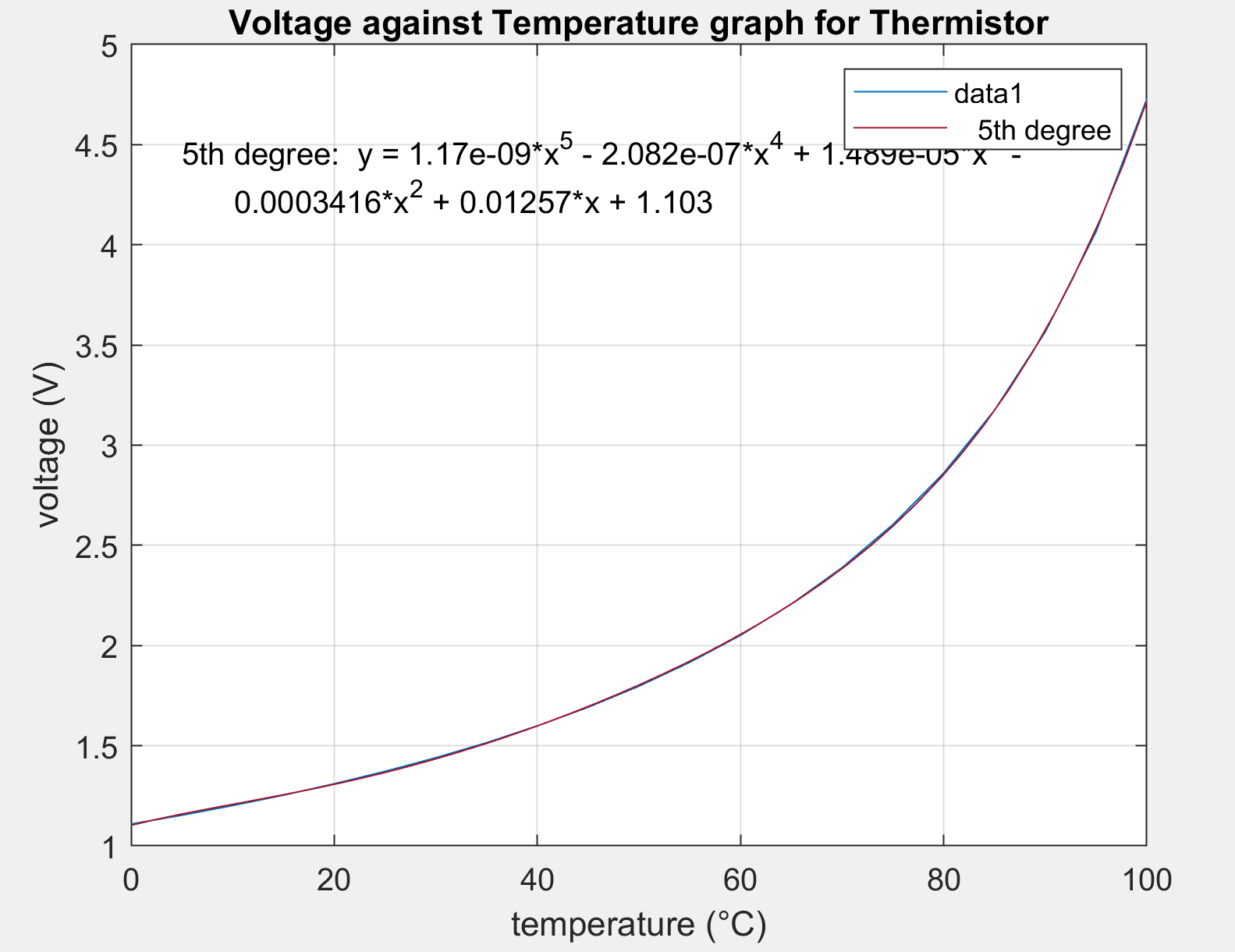
Graph shows that across temperature range we will be measuring, there is a linear relationship between temperature and resistance.



To find voltage against temperature add another fixed resistor (other fixed resistor is part of thermistor component). Measure voltage across resistor for different temperatures. Used 5v because that is output voltage from Arduino.

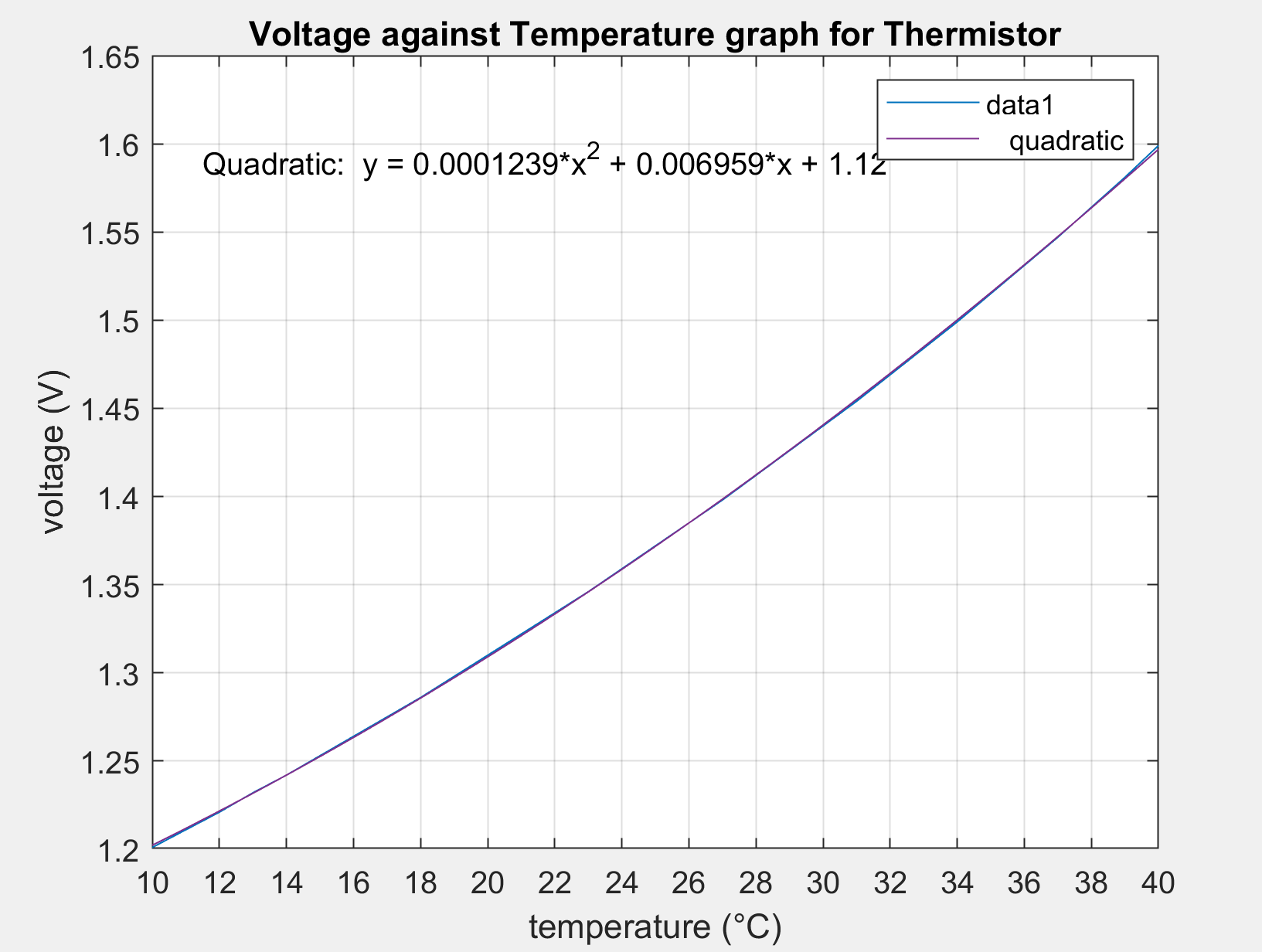
Results:

|  |  |
| --- | --- |
| **Temp (°C)** | **Voltage (V)** |
| 0 | 1.109 |
| 5 | 1.153 |
| 10 | 1.201 |
| 15 | 1.253 |
| 20 | 1.310 |
| 25 | 1.372 |
| 30 | 1.440 |
| 35 | 1.515 |
| 40 | 1.599 |
| 45 | 1.692 |
| 50 | 1.797 |
| 55 | 1.916 |
| 60 | 2.051 |
| 65 | 2.207 |
| 70 | 2.389 |
| 75 | 2.604 |
| 80 | 2.861 |
| 85 | 3.174 |
| 90 | 3.565 |
| 95 | 4.065 |
| 100 | 4.728 |



I used the fitting tool on matlab to find me a polynomial with the most accurate model to the line. This produced a line with the equation shown on the graph. However, this graph shows the voltage against temperature over the whole range of available temperatures but we are only going to be dealing with temperatures in a range of about 10-40. Therefore made another graph between these values so we can have a closer look at what happens in this range in more detail.

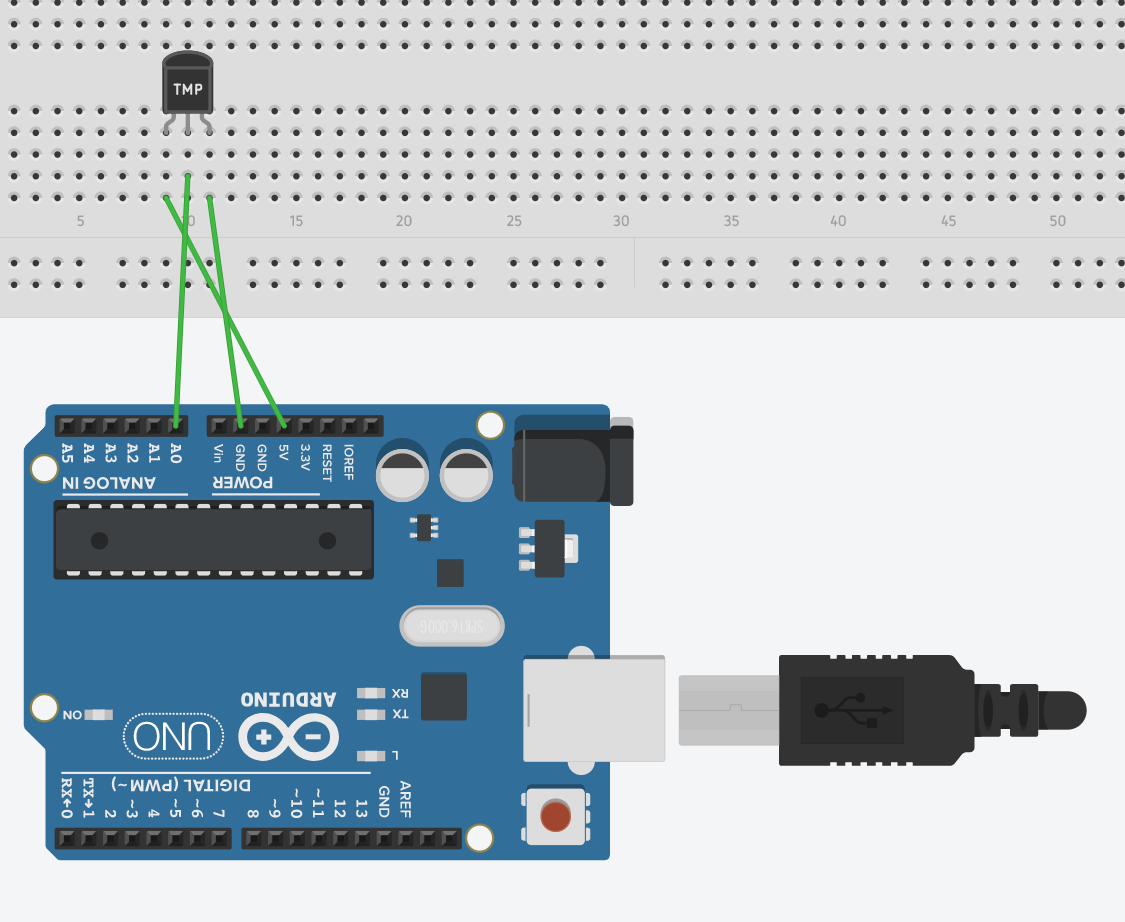
|  |  |
| --- | --- |
| **Temp (°C)** | **Voltage (V)** |
| 10 | 1.201 |
| 11 | 1.211 |
| 12 | 1.221 |
| 13 | 1.232 |
| 14 | 1.242 |
| 15 | 1.253 |
| 16 | 1.264 |
| 17 | 1.275 |
| 18 | 1.286 |
| 19 | 1.298 |
| 20 | 1.310 |
| 21 | 1.322 |
| 22 | 1.334 |
| 23 | 1.346 |
| 24 | 1.359 |
| 25 | 1.372 |
| 26 | 1.385 |
| 27 | 1.398 |
| 28 | 1.412 |
| 29 | 1.426 |
| 30 | 1.440 |
| 31 | 1.454 |
| 32 | 1.469 |
| 33 | 1.484 |
| 34 | 1.499 |
| 35 | 1.515 |
| 36 | 1.531 |
| 37 | 1.547 |
| 38 | 1.564 |
| 39 | 1.581 |
| 40 | 1.599 |



Fitted an approximate line to the graph.

Equation of fitted line:

Tinkercad:



Code for this system:

int tempPin=A0;

int tempVal;

float Voltage;

float temp;

void setup()

{

pinMode(tempPin, INPUT);

Serial.begin(9600);

}

void loop()

{

tempVal= analogRead(tempPin);

//convert the read value (between 0 & 1023) into a voltage (between 0 & 5)

Voltage=(5./1023. )\*tempVal;

//To find the temperature in degrees celsius, use this equation (provided from https://learn.adafruit.com/tmp36-temperature-sensor)

temp= ((Voltage/0.001)-500.0)/10.0;

Serial.print("Voltage= ");

Serial.print(Voltage);

Serial.print("v");

Serial.print(" Temperature= ");

Serial.print(temp);

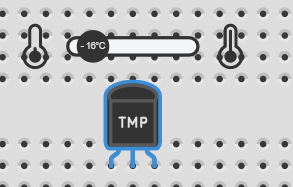
Serial.println("C");

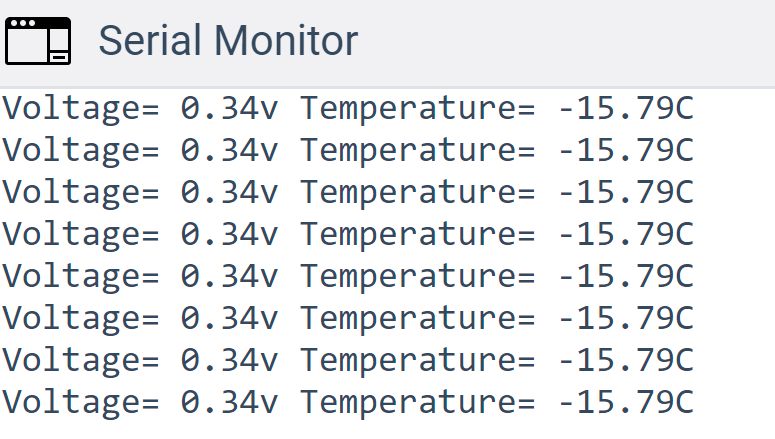
delay(500);

}

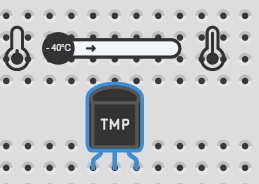
Test:

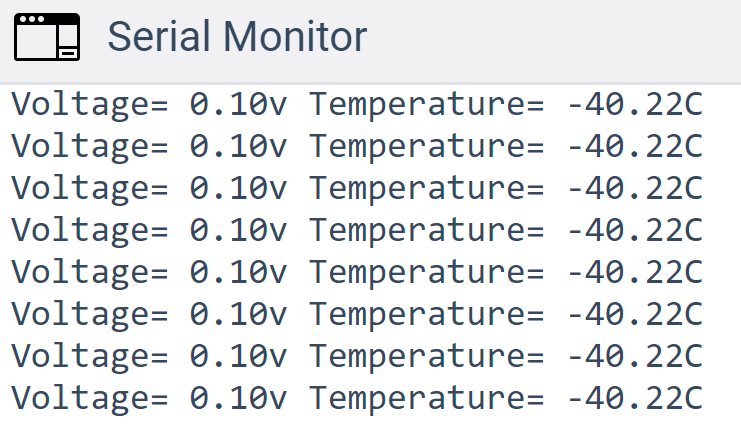
-16°C;



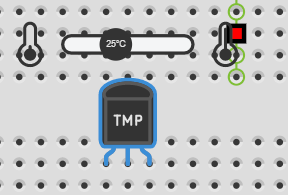


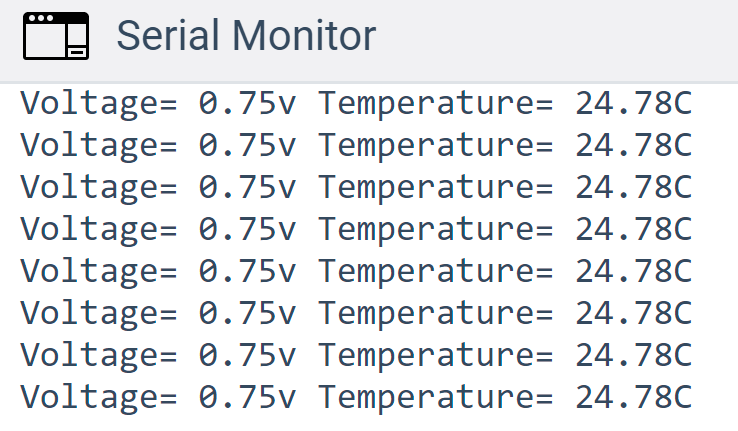
-47°C:



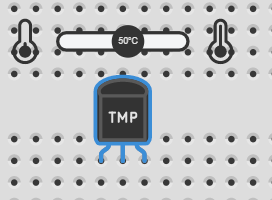


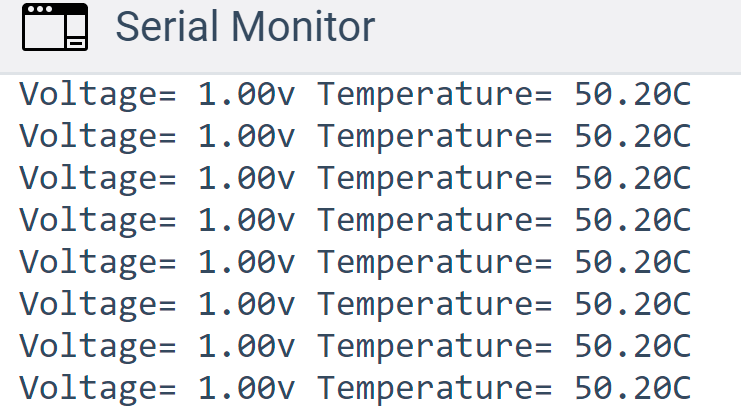
25°C:



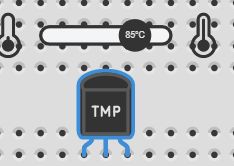
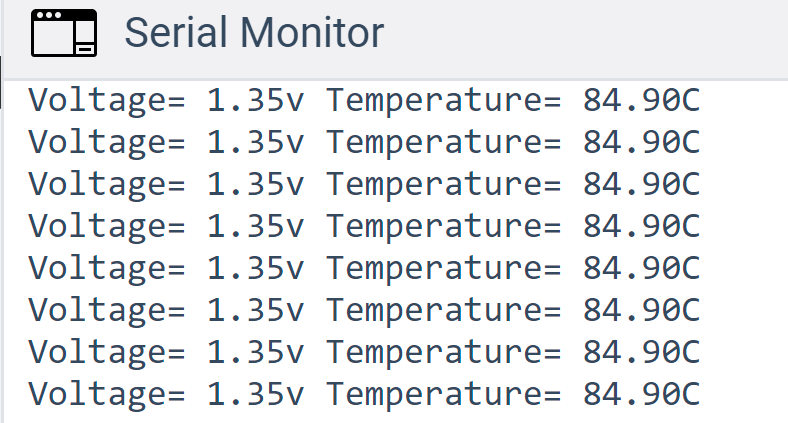


50°C:





85°C:

By taking random values for the temperature (in both the negative and positve regions) we can see that the code is successful in the conversion from a voltage reading to a temperature value. (note: we are not using the same thermistor as the one available in tinkercad, so the equation to convert voltage into temperature will be different to the one in the code).

SimAVR:

First attempt:

|  |  |  |
| --- | --- | --- |
| Temp (K) | Temp (°C) | ADC |

|  |  |  |
| --- | --- | --- |
| 294 | 21 | 543 |
| 296.7 | 23.7 | 536 |
| 299.8 | 26.8 | 488 |
| 301.3 | 28.3 | 473 |
| 302.9 | 29.9 | 450 |
| 304.1 | 31.1 | 436 |
| 305.6 | 32.6 | 463 |
| 306.7 | 33.7 | 424 |
| 307.8 | 34.8 | 422 |
| 308.8 | 35.8 | 397 |
| 308.8 | 35.8 | 385 |
| 310 | 37 | 398 |
| 310.4 | 37.4 | 351 |
| 311.6 | 38.6 | 424 |
| 311.5 | 38.5 | 379 |
| 312.1 | 39.1 | 353 |
| 294.1 | 21.1 | 539 |
| 296.9 | 23.9 | 540 |
| 299.1 | 26.1 | 492 |
| 300.8 | 27.8 | 482 |
| 303.2 | 30.2 | 447 |
| 304.2 | 31.2 | 459 |
| 305.7 | 32.7 | 412 |
| 306.8 | 33.8 | 432 |
| 307.3 | 34.3 | 423 |
| 308.4 | 35.4 | 400 |
| 309.2 | 36.2 | 388 |
| 310 | 37 | 393 |
| 310.4 | 37.4 | 394 |
| 291.8 | 18.8 | 593 |
| 291.7 | 18.7 | 590 |
| 291.9 | 18.9 | 557 |
| 292.7 | 19.7 | 602 |
| 292.6 | 19.6 | 615 |
| 293.4 | 20.4 | 605 |
| 293.4 | 20.4 | 537 |
| 293.2 | 20.2 | 586 |
| 293.7 | 20.7 | 558 |
| 294 | 21 | 558 |
| 294.3 | 21.3 | 589 |
| 294.6 | 21.6 | 580 |
| 294.4 | 21.4 | 530 |
| 294.8 | 21.8 | 513 |
| 295.3 | 22.3 | 527 |
| 295.4 | 22.4 | 561 |
| 295.9 | 22.9 | 546 |
| 295.1 | 22.1 | 552 |
| 295.9 | 22.9 | 552 |
| 295.5 | 22.5 | 530 |
| 295.6 | 22.6 | 520 |
| 295.7 | 22.7 | 516 |
| 296.1 | 23.1 | 547 |
| 296.4 | 23.4 | 516 |
| 297 | 24 | 531 |
| 295.6 | 22.6 | 543 |
| 296.9 | 23.9 | 506 |
| 296.9 | 23.9 | 504 |
| 297.2 | 24.2 | 544 |
| 296.9 | 23.9 | 515 |
| 296.6 | 23.6 | 507 |
| 297 | 24 | 537 |
| 296.7 | 23.7 | 523 |
| 297.8 | 24.8 | 536 |
| 296.8 | 23.8 | 518 |
| 297.6 | 24.6 | 540 |
| 297.2 | 24.2 | 556 |
| 297.2 | 24.2 | 539 |
| 297.6 | 24.6 | 522 |
| 297.4 | 24.4 | 504 |
| 297.3 | 24.3 | 498 |
| 297.5 | 24.5 | 544 |
| 297.1 | 24.1 | 520 |
| 297.4 | 24.4 | 498 |
| 297.3 | 24.3 | 540 |

//Find ADC value from thermistorPin.

tempVal=analogRead(thermistorPin);

//Turn heater on so you can read ADC across a large range of temperatures.

analogWrite(heaterPin,100);

Serial.println(tempVal);

delay(30000);

}

In order to find value of tempVal (ADC) across a wider range of temperatures, I turned on the heater. That produced this plot:

This graph shows us the value of the analogRead for different temperatures.

Second Attempt:

|  |  |  |
| --- | --- | --- |
| 292.9 | 19.9 | 599 |
| 294.8 | 21.8 | 535 |
| 295.8 | 22.8 | 517 |
| 297.7 | 24.7 | 561 |
| 298.2 | 25.2 | 487 |
| 299.9 | 26.9 | 510 |
| 301.6 | 28.6 | 431 |
| 302.6 | 29.6 | 497 |
| 304.3 | 31.3 | 441 |
| 304.5 | 31.5 | 431 |
| 305.7 | 32.7 | 406 |
| 306.4 | 33.4 | 427 |
| 307.6 | 34.6 | 417 |
| 307.9 | 34.9 | 420 |
| 309 | 36 | 379 |
| 309.5 | 36.5 | 393 |
| 310.6 | 37.6 | 416 |
| 311.3 | 38.3 | 401 |
| 311.7 | 38.7 | 392 |
| 312.1 | 39.1 | 359 |
| 293 | 20 | 592 |
| 294.6 | 21.6 | 577 |
| 296.1 | 23.1 | 548 |
| 297.1 | 24.1 | 514 |
| 298.6 | 25.6 | 514 |
| 299.7 | 26.7 | 488 |
| 301 | 28 | 472 |
| 302.3 | 29.3 | 457 |
| 303.4 | 30.4 | 446 |
| 304.8 | 31.8 | 435 |
| 305.2 | 32.2 | 425 |
| 306.4 | 33.4 | 414 |
| 307.7 | 34.7 | 416 |
| 307.7 | 34.7 | 386 |
| 309.4 | 36.4 | 403 |
| 309.7 | 36.7 | 388 |
| 310.2 | 37.2 | 386 |
| 311.1 | 38.1 | 372 |
| 312.2 | 39.2 | 388 |

We can use this graph to calculate the ADC value the thermistorPin is reading.

In the code, I want to find the temperature based off the ADC value that the thermistorPin is giving off.

The trend line is given by the equation y=-10.678x+782.86 where y is the ADC value and x is the temperature in Celsius. In order to calculate the error, I need to convert the ADC value that we read into a temperature, meaning we need to make x the subject of the equation. This gives us: x=(y-782.86)/10.678 . Now we can calculate the error (i.e. the difference between the setpoint and the read temperature).

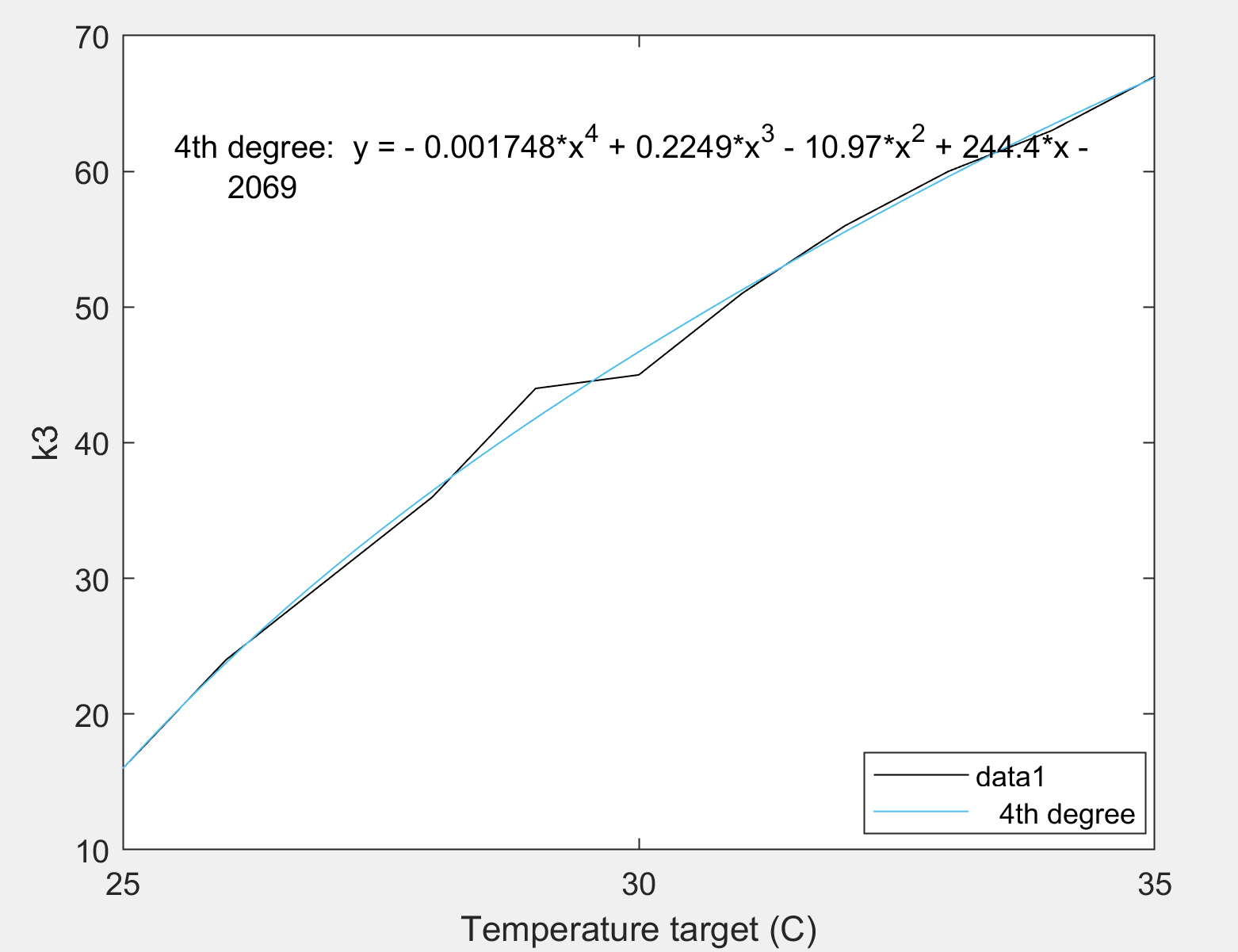
We then will need to calculate the error sum which is the sum of all the possible errors. (i.e when error=5, error sum= 5+4+3+2+1=15). We will use this for the Integral part of our control. We calculate this using a while loop.

Once we have these values, we need to come up with an equation in the form PWM=(k1\*error)+(k2\*errorsum)+k3.

After some trial and error, I found that the best values for k1, k2 & k3 respectively are 5, 0.5 &45. Note, if you are going to change the setpoint, k3 is going to change.

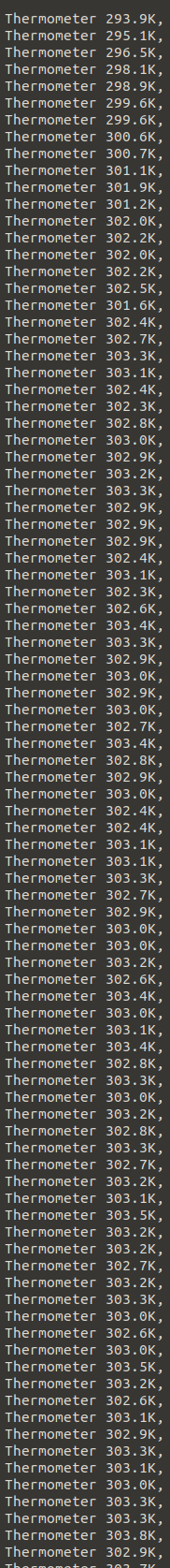
|  |  |
| --- | --- |
| Temperature (°C) | K3: |
| 25 | 16 |
| 26 | 24 |
| 27 | 30 |
| 28 | 36 |
| 29 | 44 |
| 30 | 45 |
| 31 | 51 |
| 32 | 56 |
| 33 | 60 |
| 34 | 63 |
| 35 | 67 |

This produced this graph:



Using the equation, we can find the value of k3 to input into the PWM equation. This means once the user has input their desired setpoint, the temperature sub-system will take care of the rest.

Finally, we need to make sure that the PWM doesn’t overshoot (when the temperature goes above the setpoint, the Temperature>target, therefore error will be negative. This means that we will still be applying a voltage (even if it’s negative) to the heater, so the heater will continue to dissipate energy. To ensure this doesn’t happen, I used an if statement so that If the pwm<0, it would equal 0 so no more energy dissipation.

SimAVR:

As can be seen from the readings, the temperature heats up and then fluctuates within +/- 0.5 k of the setpoint (Note, setpoint was set to 30°C which is equal to 303k.)

#include <stdio.h>

#include <string.h>

#define NO\_READINGS 30

// These define which pins are connected to what device on the virtual bioreactor

//

const byte lightgatePin = 2;

const byte stirrerPin = 5;

const byte heaterPin = 6;

const byte thermistorPin = A0;

const byte pHPin = A1;

int target ;

// The PCA9685 is connected to the default I2C connections. There is no need

// to set these explicitly.

void setup() {

Serial.begin(9600);

Serial.println("Hello World");

pinMode(lightgatePin, INPUT);

pinMode(stirrerPin, OUTPUT);

pinMode(heaterPin, OUTPUT);

pinMode(thermistorPin, INPUT);

pinMode(pHPin, INPUT);

// More setup...

String msg1 ="Good Day Sir";

String msg2="What temperature would you like to set your bio-reactor to?";

Serial.println(msg1);

Serial.println(msg2);

while(!Serial.available());

Serial.setTimeout(3000);

target=Serial.parseInt();

Serial.println(target);

}

void loop(){

float tempVal;

float Temperature = 0;

float error;

float errorsum=0;

float pwm;

//Find Temperature from Thermistor.

// get 10 readings and average it

for (int i = 0; i<NO\_READINGS; ++i)

{

tempVal=analogRead(thermistorPin);

Temperature += (tempVal-782.86)/-10.678;

delay(2);

}

Temperature /= NO\_READINGS;

Serial.print(" Temperature=");

Serial.print(Temperature);

Serial.print("C");

Serial.print(" Kelvin: ");

Serial.print(Temperature+273.15);

Serial.print("K ");

//calculate error and error sum

error=target-Temperature;

Serial.print("error=");

Serial.print(error);

while(error>=0){

errorsum+=error;

error-=1;

}

error=target-Temperature;

Serial.print(" errorsum=");

Serial.println(errorsum);

// calculate pwm

pwm=(5\*error)+(2\*errorsum)+((-0.001748\*(double) pow(target,4))+(0.2249\*(double) pow(target,3))-(10.97\*(double) pow(target,2))+(244.4\*(double) target)-2069);

//Don't let pwm go over 255 or under 0

if(pwm<0){

pwm=0;

}

if(pwm>255){

pwm=255;

}

analogWrite(heaterPin, pwm);

Serial.print("pwm=");

Serial.println(pwm);

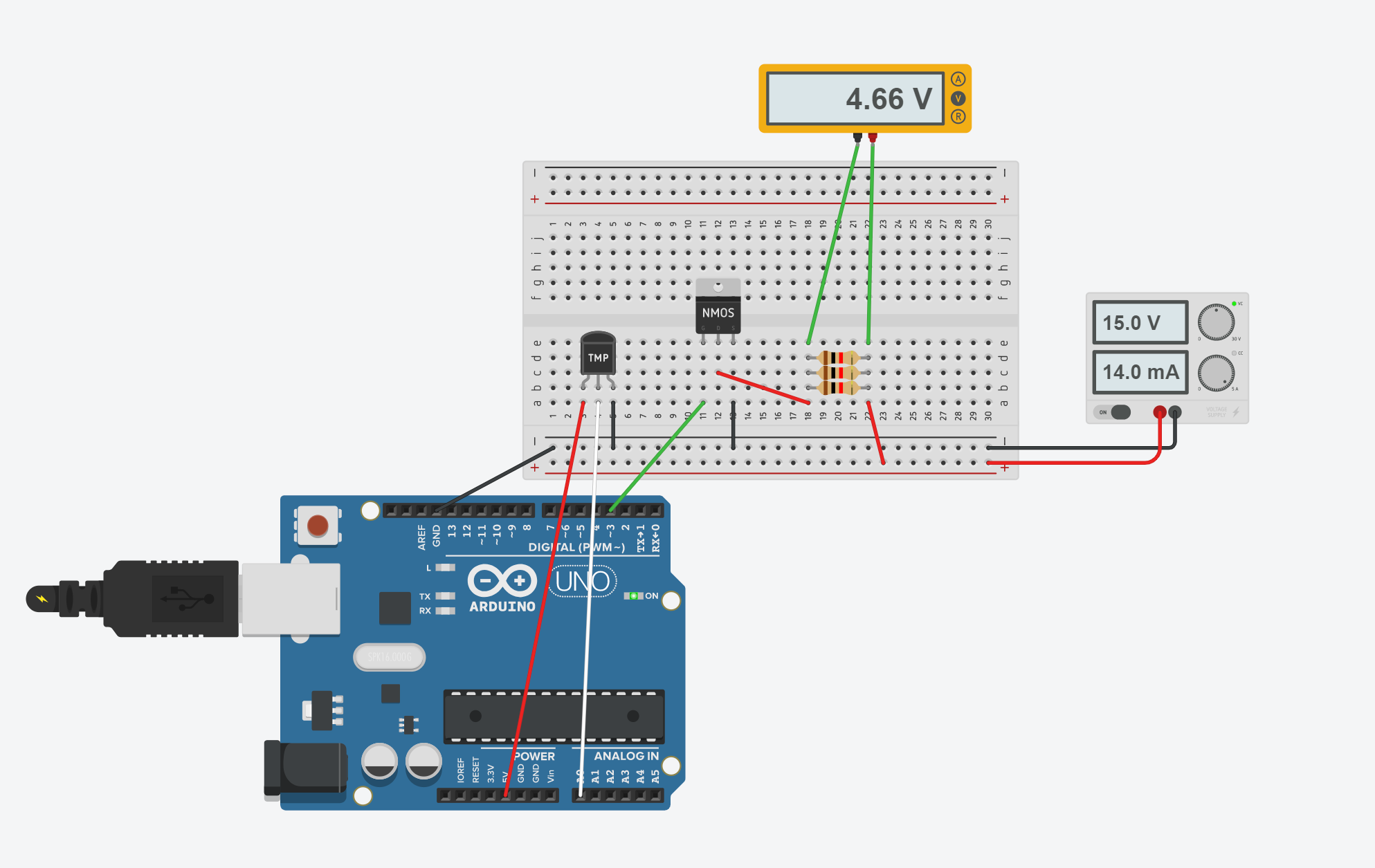
Serial.println("");

delay(10000);

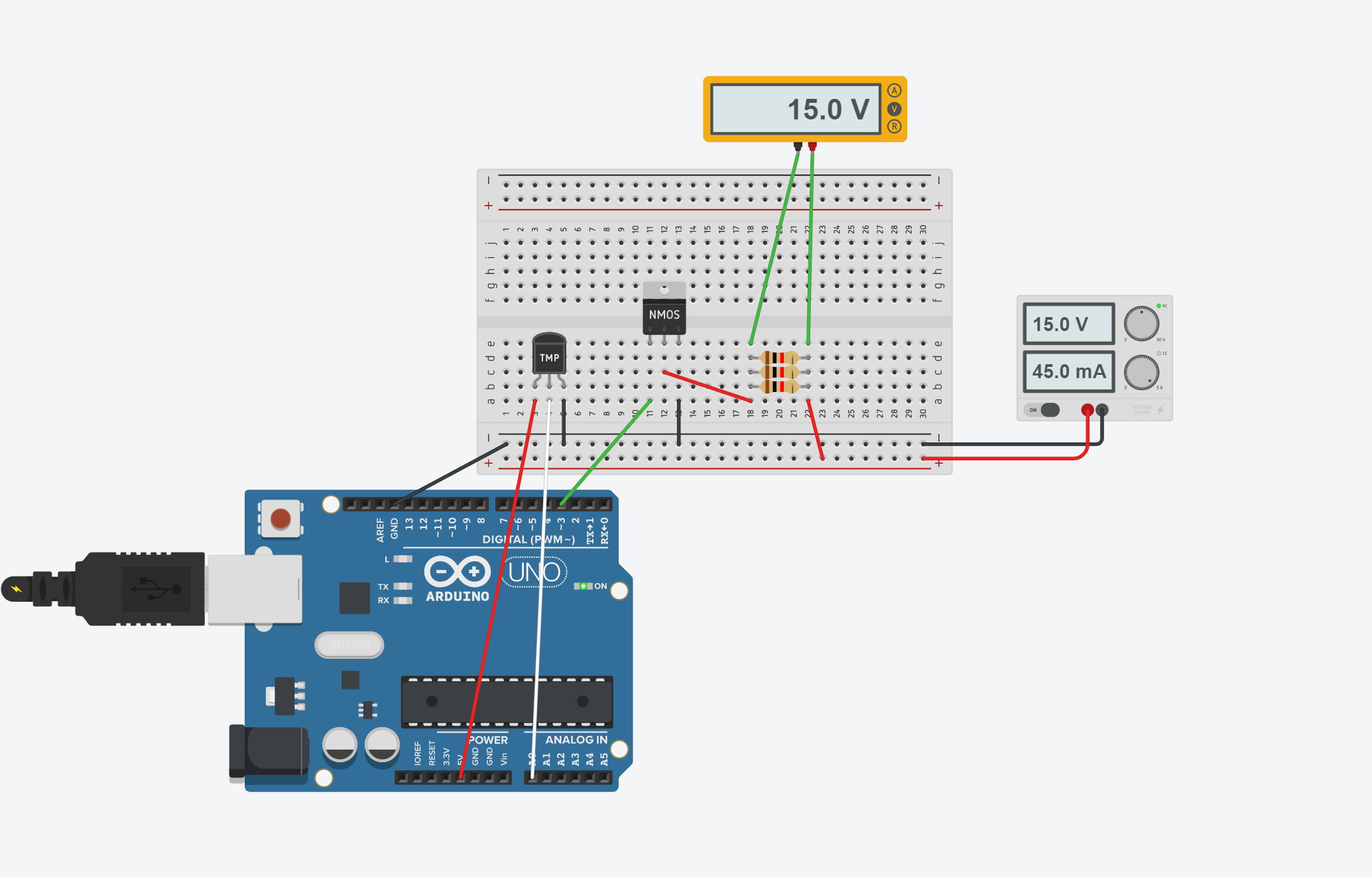
}

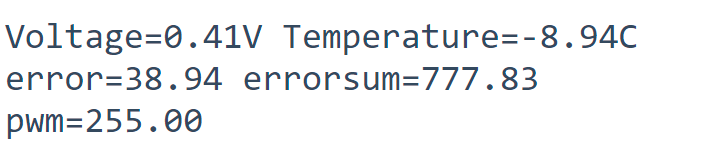
TinkerCAD:

TinkerCAD isn’t great for simulating the circuit, however it is useful for running code and seeing if it works. In my circuit, I replaced the heater with some resistors and used a voltmeter to see if there would be an output for different temperatures.

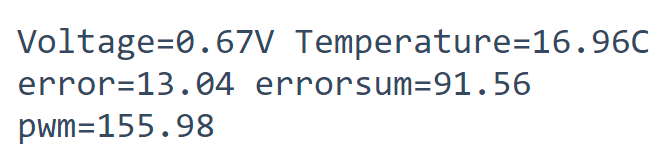
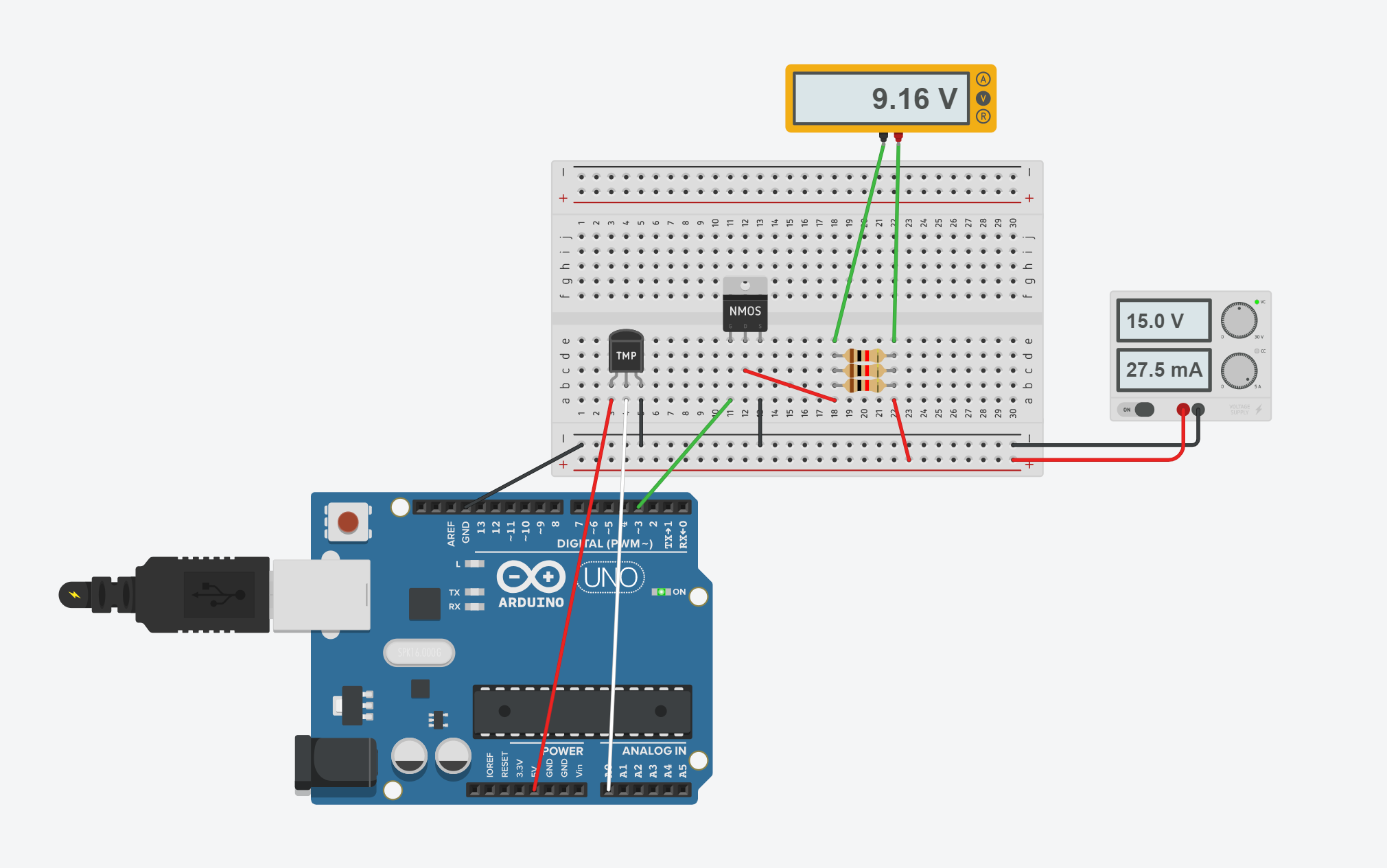


In this sketch, we can see the sensor circuit and the actuator circuit. It is important to note that since the components are different in SimAVR, the code used was slightly altered with that in mind in order to find the temperature measured by the thermistor pin. To see the calculations for temperature see the code for tinkerCAD in the calibration section. Besides for those equations, everything else was the same. Another important note is that the temperature range is from -40°C - 125°C and therefore, we can produce PWM’s higher than 255 and lower than 0, so the if statements prevented any unwanted outputs.

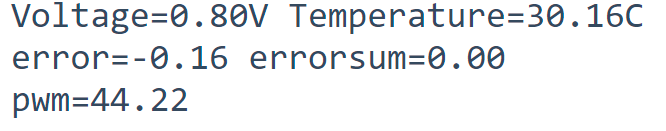
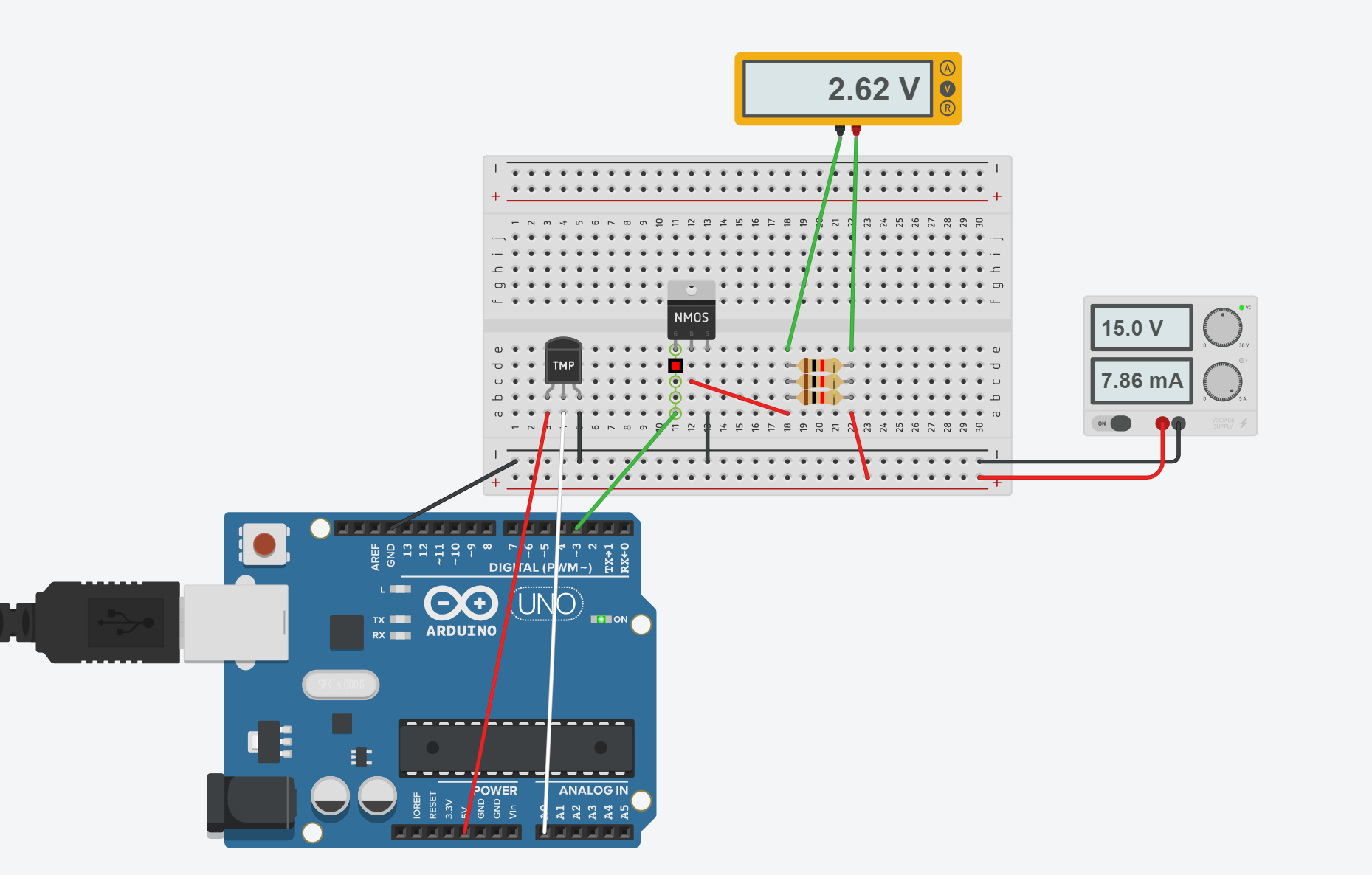




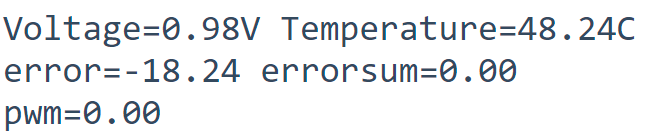
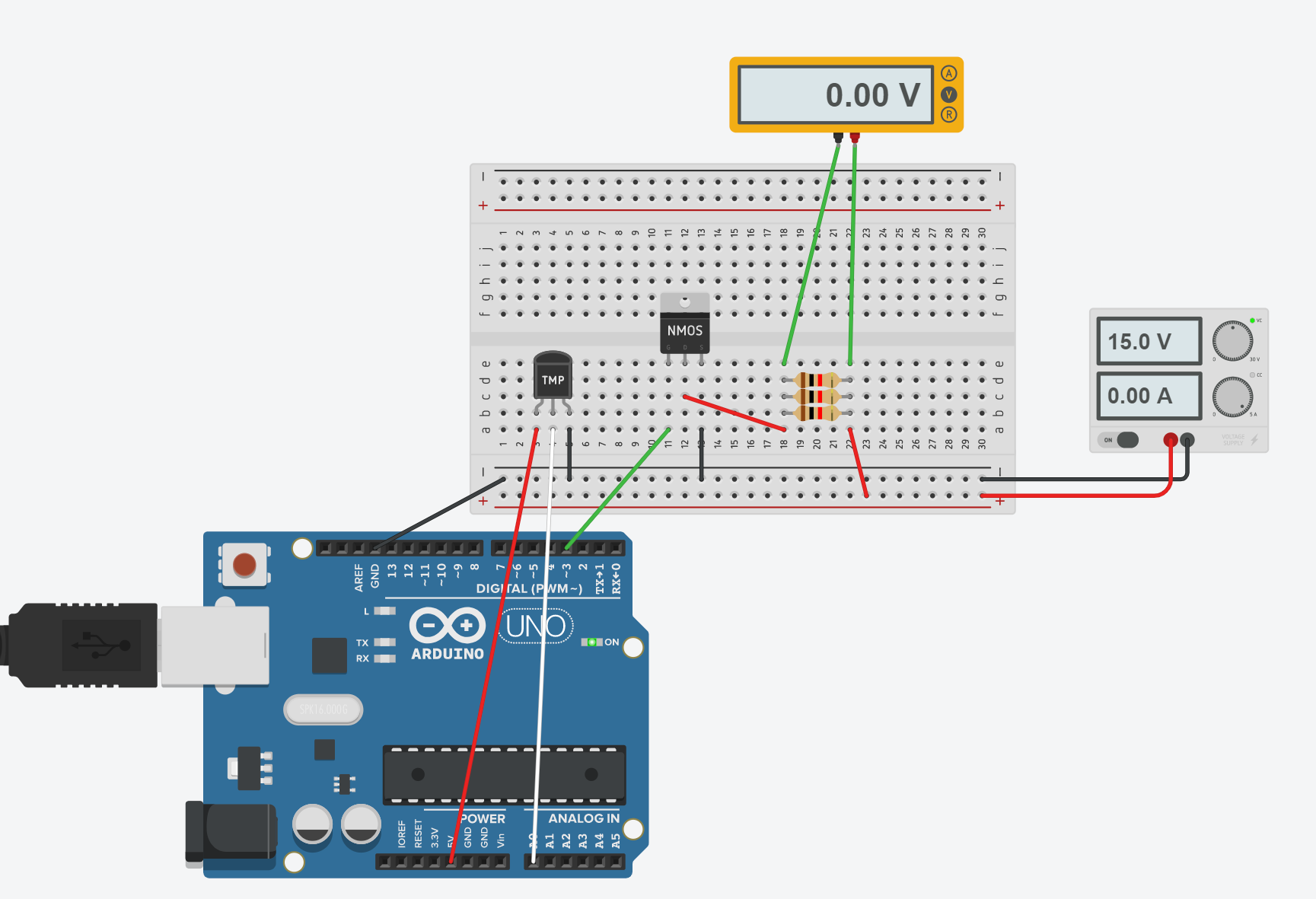
As shown in these figures, when the temperature is very low, the power output will be a maximum, as the heater will pull a maximum voltage through it.



As the temperature moves closer to the setpoint (30°C) the pwm output lowers, as does the voltage across the heater.

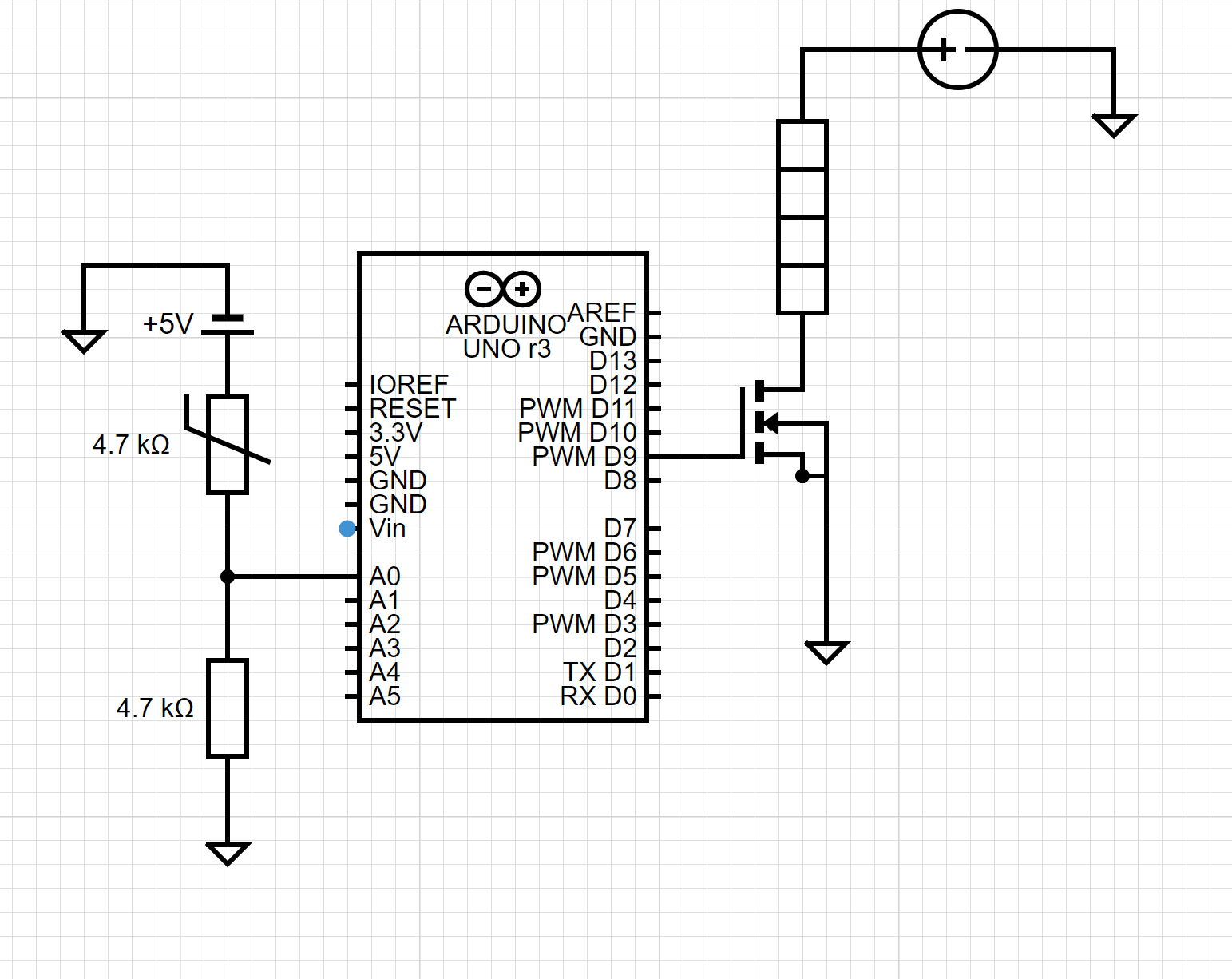


When the temperature is at the setpoint, there is still a small voltage across the heater, in order to keep the temperature of the system stable. (input energy = output energy)



When the temperature increases over a certain point above the set temperature, the heater turns off and allows the system to cool down.

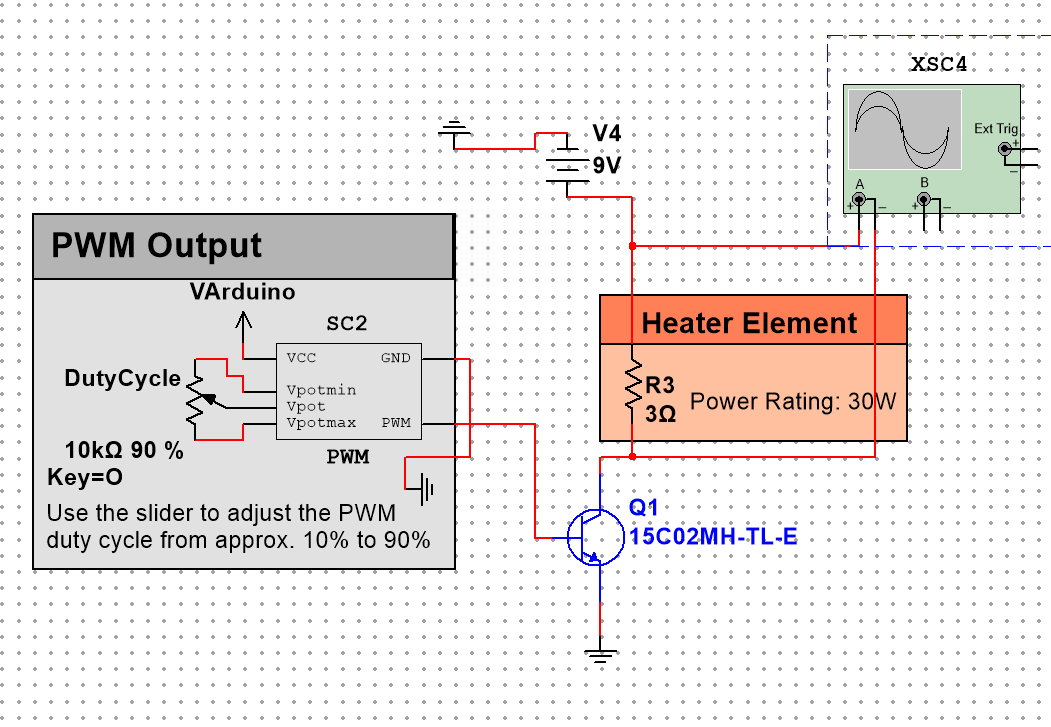
Circuit Diagram:



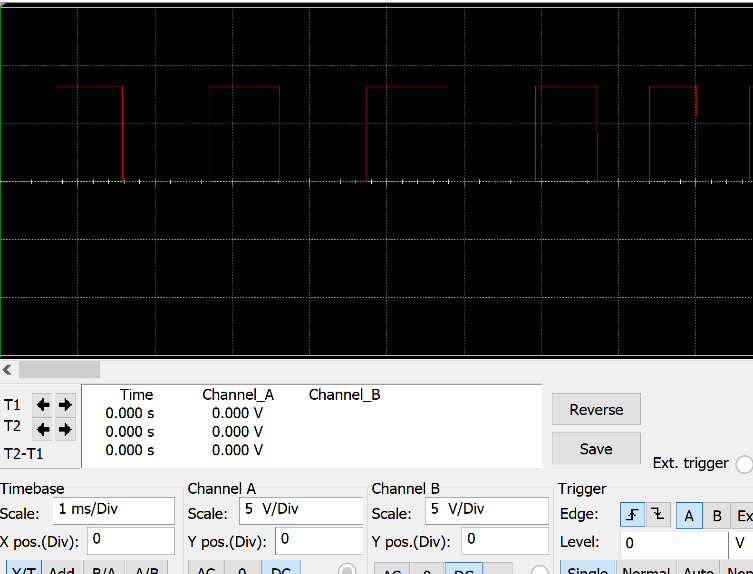
Multisim Actuator Circuit:

Multisim is unable to take feedback from the real world, and therefore isn’t an effective tool in showing that your circuit will respond to temperature. Instead, what I have done is I have built the input circuit (i.e the thermistor) and the actuator circuit (the heater) and I have proven that they are working as individuals. Since there is no way (as far as I am aware of creating a system in which the two interact with each other (i.e. the reading from the thermistor causes a change in pwm to the heater) on multisim, this task has been shown on tinkerCAD.

This is the actuator circuit:



**12V**



Oscilloscope reading at 50%. We can see that there is an output across the heater. We are unable to vary this output based on the temperature in the circuit. We will go to tinkerCAD to show the output variation based off readings made by thermistor.