Investigation into a Thermistor using an Arduino Uno

Natalia Tabja (Student Number: 1007818747)[1]
[1]*Lab Partner: Sanai Nezafat*
(Dated: February 27, 2025)

This experiment investigates the relationship between temperature and the resistance of a negative temperature coefficient (NTC) thermistor using an Arduino Uno microcontroller. A voltage divider circuit was used to measure the thermistor's resistance, and data was collected to analyze the temperature-dependent behavior. Theoretical and experimental values were compared for validation.

INTRODUCTION

A thermistor is a type of resistor whose resistance varies with temperature. The thermistor used in this experiment is a Vishay NTCLE100E3103HBO with a nominal resistance of 10kΩ at 25°C [1]. The Steinhart-Hart equation is used to relate resistance to temperature:

$$\frac{1}{T(R_t)} = A_1 + B_1 \ln\left(\frac{R_t}{R_{25}}\right) + C_1 \left(\ln\left(\frac{R_t}{R_{25}}\right)\right)^2 + D_1 \left(\ln\left(\frac{R_t}{R_{25}}\right)\right)^3 \quad (1)$$

where A_1, B_1, C_1, and D_1 are constants specific to the thermistor.

To monitor and analyze the thermistor's response to temperature variations, an Arduino Uno microcontroller was used. The Arduino's built-in Analog-to-Digital Converter (ADC) allows precise voltage readings, which were then converted into resistance values and subsequently mapped to temperature using the Steinhart-Hart equation.

RESULTS AND DISCUSSION

R1: Measuring Thermistor Resistance

The resistance of the thermistor was measured at room temperature (23°C), as determined using a temperature gun:

$$R_{th} = 10.363 \pm 0.03\text{k}\Omega \quad (2)$$

This value is reasonably close to the nominal resistance of 10kΩ at 25°C, but since the room temperature was measured to be 23°C, a slightly higher resistance is expected due to the negative temperature coefficient (NTC) characteristic of the thermistor.
The uncertainty was calculated based on the Keysight 34461A multimeter specifications [2] for measurements in the 10kΩ range:

$$\Delta R = (0.0020 \times 10.363\ \text{k}\Omega) + (0.0005 \times 10\ \text{k}\Omega) \quad (3)$$

$$\Delta R = 0.0259\ \text{k}\Omega \approx 0.03\ \text{k}\Omega \quad (4)$$

Thus, the measured resistance aligns with the expected behavior of the thermistor at room temperature.

R2: Thermistor Resistance Under Finger Press

For a measured voltage of 2.39V, the uncertainty from the 5V range is:

$$\Delta V = (0.0020 \times 2.39V) + 0.0006V \approx 0.01V \quad (5)$$

so the final voltage measurement is:

$$V = 2.39 \pm 0.01V \quad (6)$$

Using the voltage divider formula, the resistance of the thermistor is calculated as:

$$R_{th} = \frac{V \times R_{ref}}{V_{ref} - V} = \frac{2.39V \times 10.363\text{k}\Omega}{5V - 2.39V} \quad (7)$$

Using uncertainty propagation, the final measurement is:

$$R_{th} = 9.16 \pm 0.04\ \text{k}\Omega \quad (8)$$

This demonstrates the thermistor's sensitivity to temperature changes.

Estimating Hand Temperature

To estimate the temperature of my hand, we use the Steinhart-Hart equation:

$$\frac{1}{T} = A_1 + B_1 \ln R + C_1 (\ln R)^2 + D_1 (\ln R)^3 \quad (9)$$

where A_1, B_1, C_1, D_1 are constants specific to the thermistor. From the datasheet [1], the constants for this thermistor type (in units of K^{-1}) are:

$$A_1 = 3.354016 \times 10^{-3}, \quad (10)$$

$$B_1 = 2.884193 \times 10^{-4}, \quad (11)$$

$$C_1 = 4.118032 \times 10^{-6}, \quad (12)$$

$$D_1 = 1.786790 \times 10^{-7}. \quad (13)$$

Substituting $R = 9.16\text{k}\Omega$:

$$\frac{1}{T} = (3.354016 \times 10^{-3}) \tag{14}$$
$$+ (2.884193 \times 10^{-4}) \ln(9.16 \times 10^{3})$$
$$+ (4.118032 \times 10^{-6})(\ln(9.16 \times 10^{3}))^{2}$$
$$+ (1.786790 \times 10^{-7})(\ln(9.16 \times 10^{3}))^{3}. \tag{15}$$

Solving for T:

$$T \approx 305.2 \text{ K} \approx 32^{\circ}\text{C}. \tag{16}$$

Thus, the estimated temperature of my hand is $32^{\circ}C$, which is reasonable given typical human skin temperature ranges.

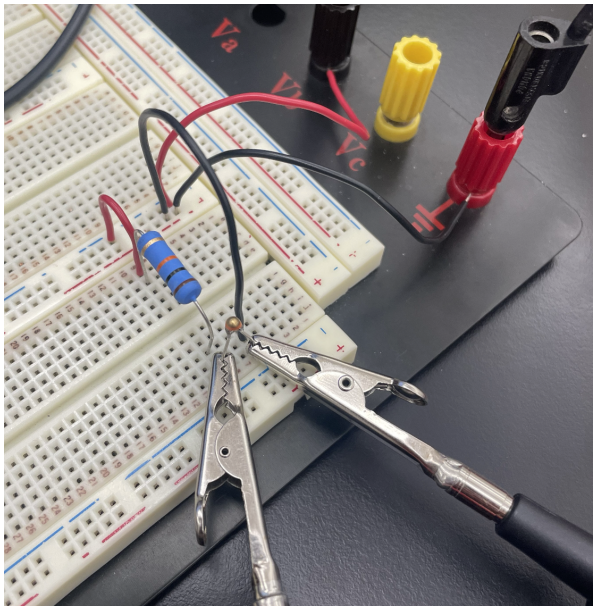


FIG. 1. Voltage divider circuit using a measured 10k resistor for R_{ref} and ± 5 VDC

R3 & R4: Arduino Blink Program

The LED turns on and stays lit for 2 seconds, then turns off for 100 milliseconds before repeating the cycle indefinitely. We note that the first time the LED transitions from on to off, the TX (Transmit) and RX (Receive) LEDs flash, indicating data transmission between the Arduino and the computer. The video recording of this pattern has been uploaded to Quercus.

State	Duration (ms)
LED ON	2000
LED OFF	100

The Arduino Blink program used in this experiment is as follows:

```
/*
  Blink
  Turns an LED on for one second, then off for one second, repeatedly.
*/

void setup() {
  pinMode(LED_BUILTIN, OUTPUT); // initialize digital pin
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn LED on
  delay(2000);                     // wait 2 sec
  digitalWrite(LED_BUILTIN, LOW);  // turn LED off
  delay(100);                      // wait 100 ms
}
```

Listing 1. Arduino Blink Program

This program continuously blinks the onboard LED following the timing pattern shown in the table above.

R5: Voltage Regulator Output

Measured voltage regulator output for different input voltages:

V_{in} (V)	V_{out} (V)
5.00 ± 0.01	4.033 ± 0.005
7.00 ± 0.012	5.065 ± 0.005
7.50 ± 0.013	5.065 ± 0.005
20.0 ± 0.025	5.067 ± 0.005
25.0 ± 0.030	5.068 ± 0.005

The regulator maintains an output near 5V, which is what we expected. The uncertainties were calculated based on the specified formulas in the Keysight E36300 Series power supply manual [3].

To analyze the performance of the voltage regulator, we calculate the percentage difference in the output voltage between the input voltages of 7V and 25V:

$$\text{Percentage Difference} = \frac{|V_{out}(25V) - V_{out}(7V)|}{V_{out}(7V)} \times 100\% \tag{17}$$

Substituting the values:

$$\text{Percentage Difference} = \frac{|5.068V - 5.065V|}{5.065V} \times 100\% \tag{18}$$

$$\approx \frac{0.003V}{5.065V} \times 100\% = 0.059\% \tag{19}$$

This small percentage difference confirms that the voltage regulator effectively maintains a stable 5V output despite significant variations in the input voltage. The observed deviations are minimal and within the expected range for a well-regulated power supply.

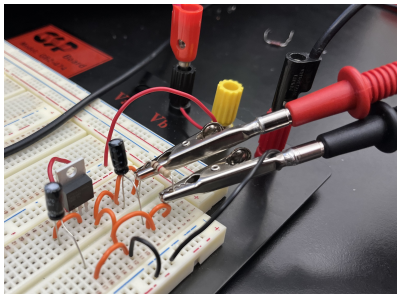


FIG. 2. Voltage regulator circuit used in R5.

R6: Voltage Step Size

The theoretical ADC resolution is calculated as:

$$\frac{V_{ref}}{1024} = \frac{5V}{1024} \approx 4.88\text{mV/bit} \tag{20}$$

The resolution based on the 4.8V reference is:

$$\frac{4.8V}{1023} \approx 4.69\text{mV/bit} \tag{21}$$

R7: Minimum and Maximum Measurable Voltages

The Arduino ADC is designed to measure voltages within the range of 0V to 5V. Applying voltages outside this range may damage the ADC, as stated in the Arduino Uno Rev3 SMD documentation [4].

- **Minimum Voltage:** $0V$
- **Maximum Voltage:** $5V$

R8: Analog Input Test

To verify the accuracy of the Arduino's Analog-to-Digital Converter (ADC), the recorded voltage using `AnalogReadSerial` can be determined using the equation:

$$V_{\text{in}} = \frac{\text{Digital Reading}}{1023} \times V_{\text{ref}} \tag{22}$$

With a recorded digital reading of 316 and a reference voltage of $V_{\text{ref}} = 4.8V$, we calculate:

$$V_{\text{in}} = \frac{316}{1023} \times 4.8V = 1.48V \tag{23}$$

To check if this value is reasonable, we recall that the input voltage was controlled via a potentiometer. Given that the potentiometer was adjusted to a middle position, an input voltage of approximately 1.48V aligns with expectations, as it falls within the expected range between 0V and 5V.

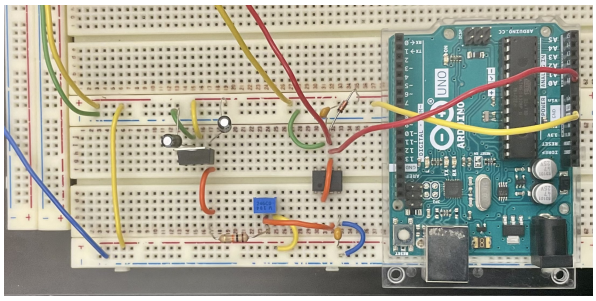


FIG. 3. Potentiometer circuit used to vary the input voltage for R8. The output voltage was read using the Arduino's ADC and compared to theoretical expectations.

R9: Arduino Temperature Measurement

Figure 4 from the Arduino Serial Plotter shows the thermistor's response to temperature variations over time. The graph exhibits two distinct peaks yet similar, indicating consistent temperature changes each of the two times the thermistor was squeezed. Between these peaks, the temperature drops, showing a return to ambient conditions when external heating was removed.

From the data:

- The baseline temperature starts around 16°C.
- Upon squeezing the thermistor, its temperature rises quickly, peaking at approximately 21–22°C, which aligns with the expected response when exposed to body heat.
- After reaching the peak, the temperature gradually declines back toward its initial state, demonstrating the thermistor's cooling as it equilibrates with the surrounding air.

This plot aligns well with the temperature readings recorded in R10, confirming the expected behavior of the thermistor: resistance decreases as temperature increases, leading to higher temperature readings when exposed to heat.

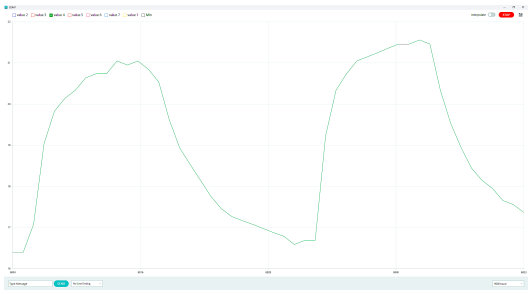


FIG. 4. Arduino Serial Plot showing the thermistor response to temperature changes.

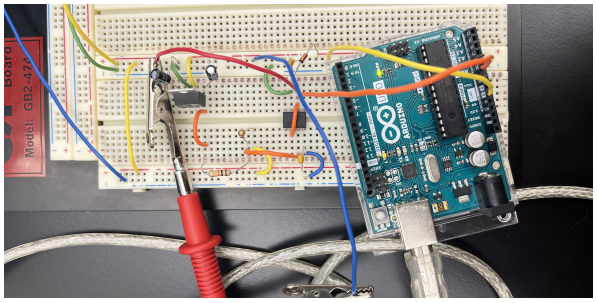


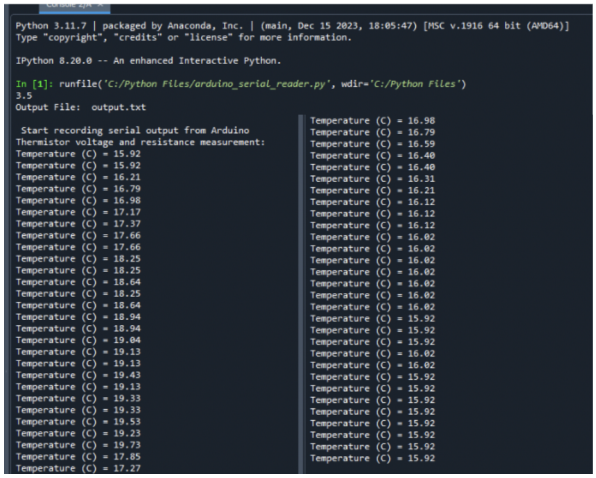
FIG. 5. Photograph of the circuit used for R9 and R10, showing the thermistor and voltage divider configuration.

R10: Python Communications with Arduino

Data was logged using the `arduino_serial_reader.py` script communicating with the Arduino. A sample of the recorded temperature values:

```
Temperature (C): 15.92
Temperature (C): 16.89
Temperature (C): 16.79
Temperature (C): 16.98
Temperature (C): 17.17
Temperature (C): 17.66
Temperature (C): 18.25
Temperature (C): 19.04
Temperature (C): 19.43
Temperature (C): 19.73
Temperature (C): 19.53
Temperature (C): 19.33
Temperature (C): 17.85
Temperature (C): 16.79
Temperature (C): 16.40
Temperature (C): 16.12
Temperature (C): 16.02
Temperature (C): 15.92
```

It is evident that when the thermistor was squeezed between our fingers, the recorded temperature increased. The baseline temperature was about 15.9°C and peaked at approximately 19.7°C, which aligns with expectations from R9. This confirms the expected response of the thermistor.



```
Python 3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.20.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Python Files/arduino_serial_reader.py', wdir='C:/Python Files')
3.5
Output File:  output.txt

 Start recording serial output from Arduino
Thermistor voltage and resistance measurement:
Temperature (C) = 15.92
Temperature (C) = 15.92
Temperature (C) = 16.21
Temperature (C) = 16.79
Temperature (C) = 16.98
Temperature (C) = 17.17
Temperature (C) = 17.37
Temperature (C) = 17.66
Temperature (C) = 17.66
Temperature (C) = 18.25
Temperature (C) = 18.25
Temperature (C) = 18.64
Temperature (C) = 18.25
Temperature (C) = 18.64
Temperature (C) = 18.94
Temperature (C) = 18.94
Temperature (C) = 19.04
Temperature (C) = 19.13
Temperature (C) = 19.13
Temperature (C) = 19.43
Temperature (C) = 19.13
Temperature (C) = 19.33
Temperature (C) = 19.33
Temperature (C) = 19.53
Temperature (C) = 19.23
Temperature (C) = 19.73
Temperature (C) = 17.85
Temperature (C) = 17.27
Temperature (C) = 16.98
Temperature (C) = 16.79
Temperature (C) = 16.59
Temperature (C) = 16.40
Temperature (C) = 16.40
Temperature (C) = 16.31
Temperature (C) = 16.21
Temperature (C) = 16.12
Temperature (C) = 16.12
Temperature (C) = 16.12
Temperature (C) = 16.02
Temperature (C) = 16.02
Temperature (C) = 16.02
Temperature (C) = 16.02
Temperature (C) = 16.02
Temperature (C) = 16.02
Temperature (C) = 16.02
Temperature (C) = 15.92
Temperature (C) = 15.92
Temperature (C) = 15.92
Temperature (C) = 16.02
Temperature (C) = 16.02
Temperature (C) = 15.92
Temperature (C) = 15.92
Temperature (C) = 15.92
Temperature (C) = 15.92
Temperature (C) = 15.92
Temperature (C) = 15.92
Temperature (C) = 15.92
Temperature (C) = 15.92
```

FIG. 6. Complete console output showing the change in temperature of the thermistor as it was squeezed between our fingers once and then released. The latter half of the data has been pasted on the right for formatting purposes.

CONCLUSION

The experiment successfully demonstrated the temperature dependence of an NTC thermistor. The measured ADC step size and voltage conversion agreed with theoretical expectations. The Arduino, combined with Python data logging, effectively recorded thermistor output, providing valuable insights into its behavior.

[1] Vishay, "NTCLE100E3103HBO Thermistor Datasheet," 2025. `https://www.farnell.com/datasheets/2792960.pdf`. [Accessed: Feb. 27, 2025].

[2] Keysight Technologies, "34461A Digital Multimeter Datasheet," `https://www.keysight.com/ca/en/assets/7018-06411/data-sheets/34461A.pdf`, 2025. [Accessed: Feb. 27, 2025].

[3] Keysight Technologies, "E36300 Series Triple Output Power Supplies Data Sheet," 2025. `https://www.keysight.com/ca/en/assets/7018-05629/data-sheets/5992-2124.pdf`. [Accessed: Feb. 28, 2025].

[4] Arduino, "Uno Rev3 SMD Specifications," 2025. `https://docs.arduino.cc/hardware/uno-rev3-smd/`. [Accessed: Feb. 27, 2025].