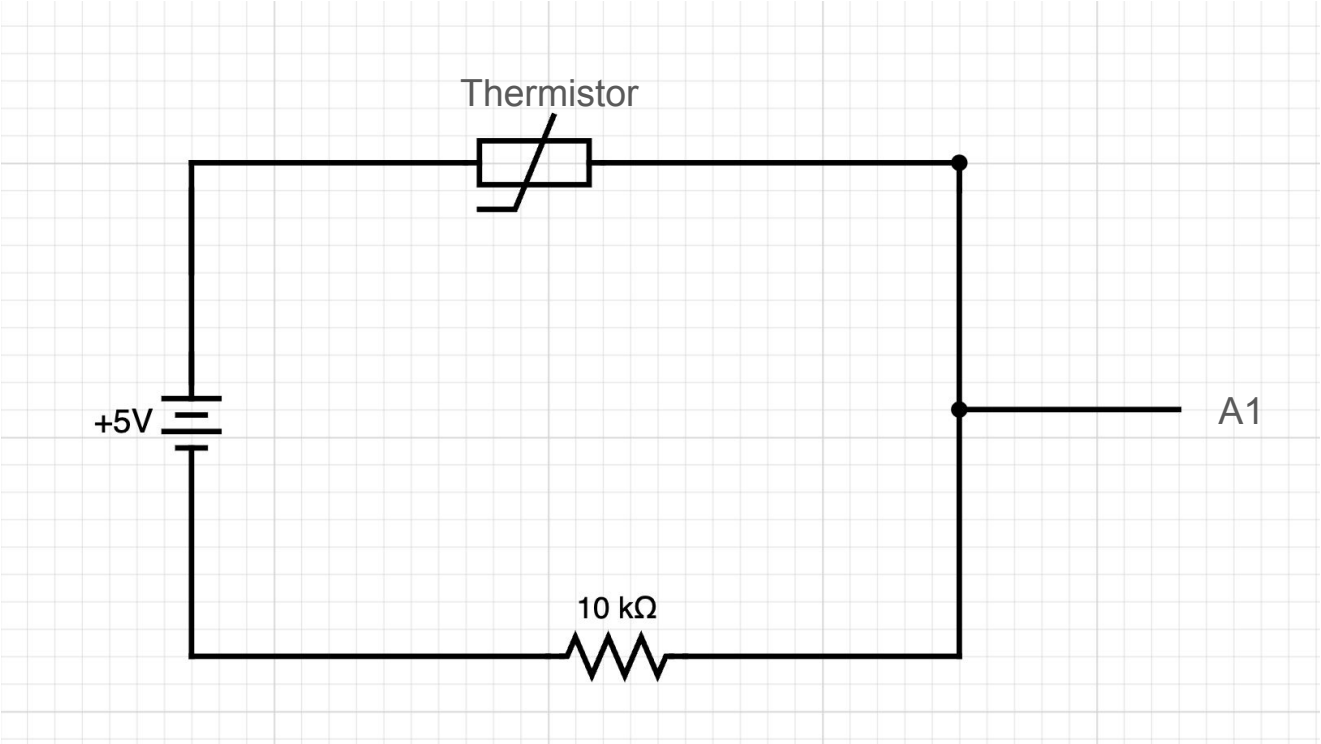
Serial Transmit of Temperature

Julian Torres

Project Requirements

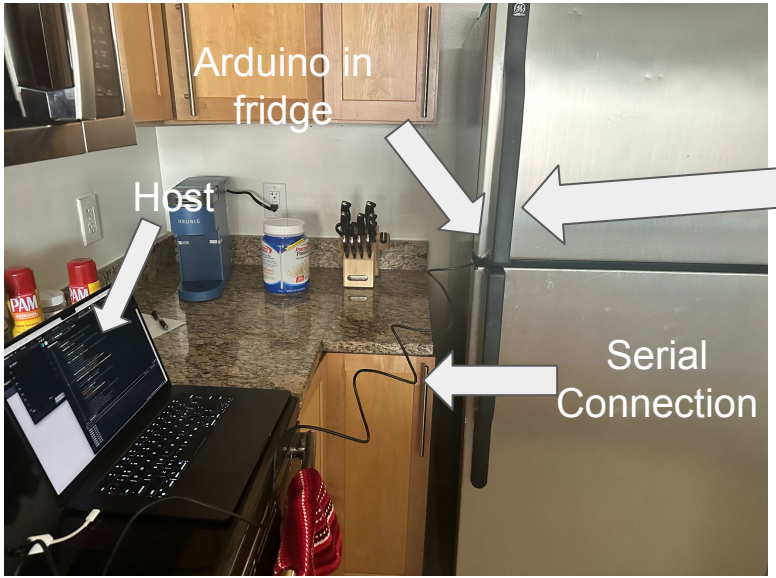
- Capture ambient temperature from Arduino at a rate of 0.1 Hz (every 10s)
- After allowing it to stabilize in room temperature, place in fridge for 5 minutes, then remove and place back into room temperature for 5 minutes
- Transmit (time, temp) to host
- Plot resulting data
- **Must use Round-Robin with Interrupts Architecture**

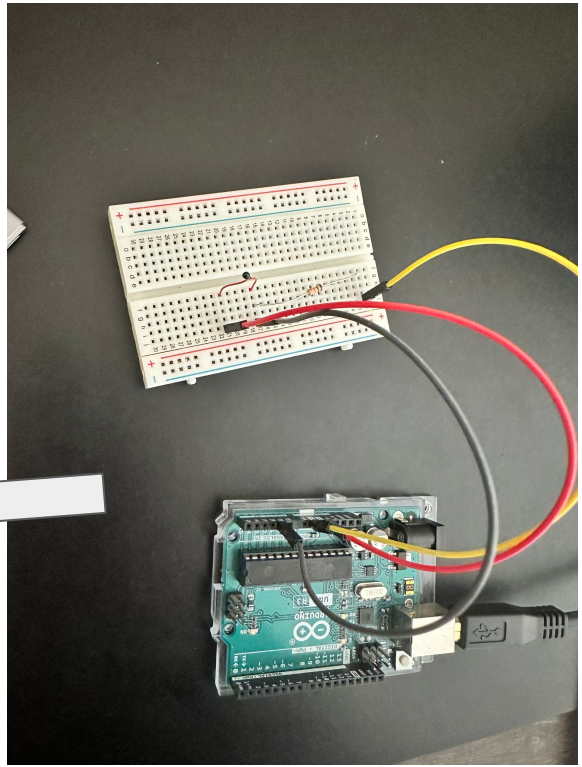
Circuit Design



- Simple voltage divider
- Varying Thermistor resistance modulates voltage read by A1
- This voltage serves as observed input to Arduino logic

Setup





Code (Arduino)

- Use arduino TimerOne library for periodic Interrupt functions (https://www.arduino.cc/reference/en/libraries/timerone/)
- Voltage Divider Equation
- Beta Equation
- Shared Data: tempReadFlag marked as volatile to avoid race condition
  - Tells compiler not to optimize and always fetch from memory
  - Could also disable/enable Interrupts in loop()

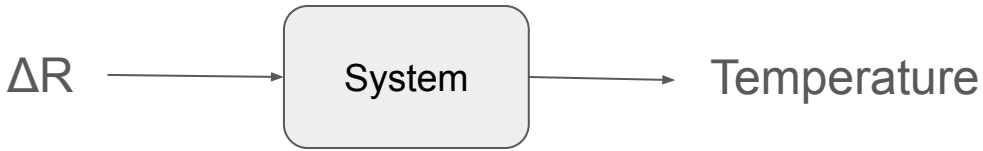
```
#include <TimerOne.h>

const double reference_temperature{ 77.0 };  // Reference temperature in Fahrenheit
const double R_reference_OHMS{ 10000 };      // Reference resistance in ohms (10k ohms)
const double beta{ 7035 };                   // Beta value 4035

double Vcc = 5.0;               // Assume 5V supply
double Vout;                    // Measured voltage across the thermistor
double thermistorResistance;    // Calculated thermistor resistance
double recordedTemperatureFar;  // Temperature in Fahrenheit
const double T0 = 298.15;       // Reference temperature in Kelvin (25°C = 298.15 K)

volatile bool tempReadFlag = false;  // Flag to signal temperature reading task

const bool DEBUG_MODE{ false };

double computeThermistorResistance(double V) {
  return R_reference_OHMS * (Vcc / V - 1);  // Voltage divider equation
}

double computeBetaEquation(double resistance) {

  double temperatureKelvin = 1 / ((1 / T0) + (1 / beta) * log(resistance / R_reference_OHMS));
  double temperatureCelsius = temperatureKelvin - 273.15;
  double temperatureFahrenheit = (temperatureCelsius * 9.0 / 5.0) + 32.0;

  return temperatureFahrenheit;
}


void timerISR() {
  tempReadFlag = true;  // Set flag to true when the timer triggers
}

void setup() {
  Serial.begin(9600);  // Initialize serial communication at 9600 baud rate
  pinMode(A1, INPUT);  // Set the analog pin A0 as input

  Timer1.initialize(10000000);        // 10,000,000 microseconds = 10 seconds
  Timer1.attachInterrupt(timerISR);  // Attach the ISR to Timer1
}

```

Code (Arduino)

```
void loop() {
  // Round-robin task 1: Temperature reading every 10 seconds
  if (tempReadFlag) {
    // Reset the flag to prevent re-execution
    tempReadFlag = false;

    // Read the analog input pin (e.g., A0) and convert to voltage
    int sensorValue = analogRead(A1);
    if (DEBUG_MODE) {
      Serial.print("sensorValue: ");
      Serial.println(sensorValue);
    }
    Vout = sensorValue * (Vcc / 1023.0);  // Convert analog value to voltage

    if (DEBUG_MODE) {
      Serial.print("Vout: ");
      Serial.println(Vout);
      Serial.println(Vout);
      Serial.println(sensorValue);
    }

    // Compute thermistor resistance and temperature
    thermistorResistance      = computeThermistorResistance(Vout);
    recordedTemperatureFar    = computeBetaEquation(thermistorResistance);

    if (DEBUG_MODE) {
      Serial.print("Computed resistance: ");
      Serial.println(thermistorResistance);
    }

    // Get the current time in milliseconds
    unsigned long timeMillis = millis();

    // Transmit time and temperature in CSV format over Serial port
    Serial.print(timeMillis);
    Serial.print(", ");
    Serial.println(recordedTemperatureFar);

  }
  // Other Round Robin tasks would be added here
}
```

Code Breakdown

- ISR function pointer passed to attachInterrupt()
  - ISR only sets a flag to true
- All task code execution is contingent on this flag being true
  - Sets back to false as its first line
  - Executes all logic to print time and temperature
- High level recap:

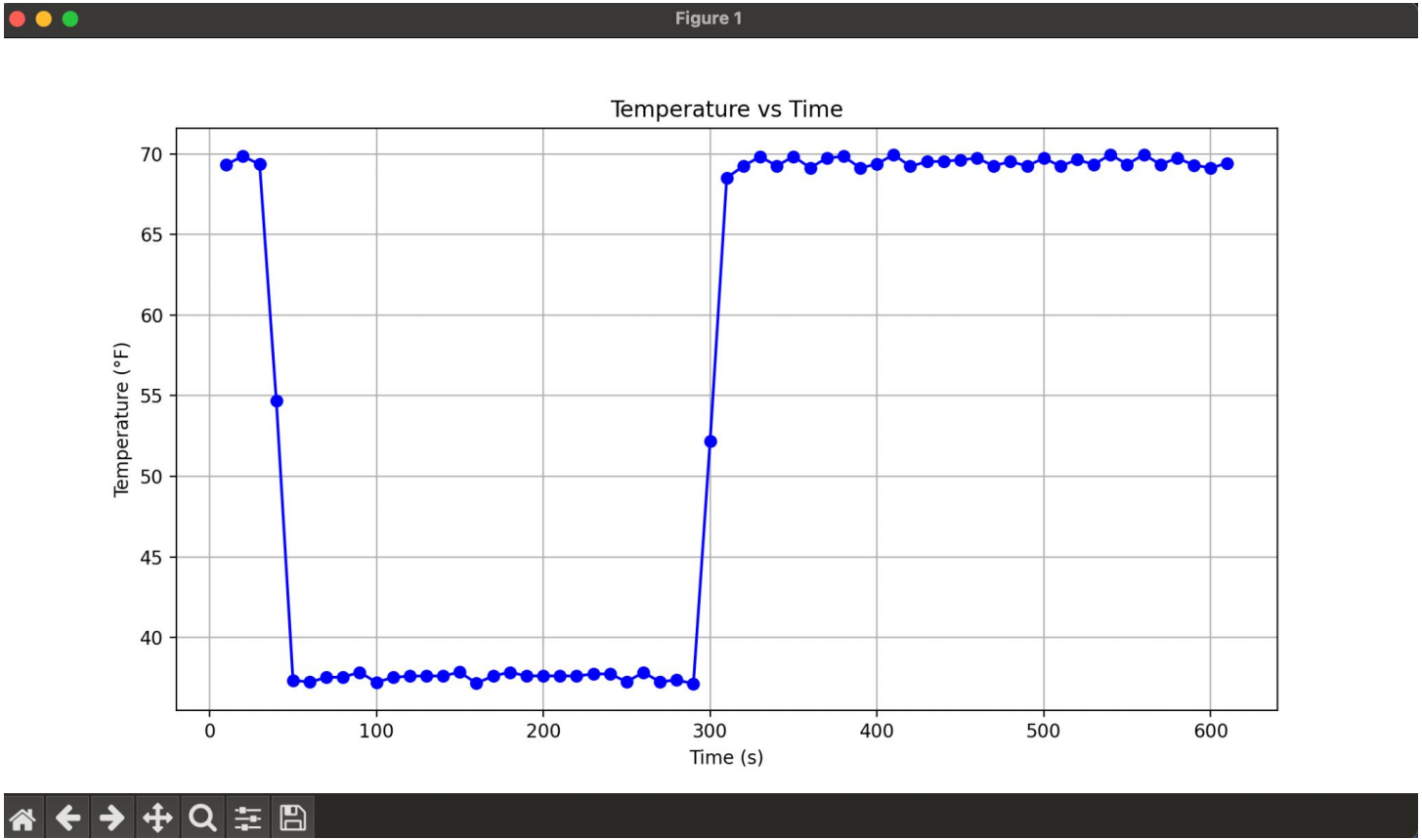


Code (Python Plotter)

- (time[s], temp[℉]) simply printed to Serial Monitor and copied to a csv
- Trivial Python script to plot
- Another option: have arduino sending data to port, and Python script listening to that port to plot in real time

```python
import matplotlib.pyplot as plt
import csv

time = []
temps = []

with open('tempVtime.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)
    for row in reader:
        time.append(float(row[0]))
        temps.append(float(row[1]))


plt.plot(time, temps, marker='o', linestyle='-', color='b')

plt.xlabel('Time (s)')
plt.ylabel('Temperature (°F)')
plt.title('Temperature vs Time')

plt.grid(True)
plt.show()
```

Results



Video Presentation: https://youtu.be/pqwzBx7BgKU