A REPORT

ON

THERMISTOR BASED TEMPERATURE CONTROL SYSTEM

Submitted to :

Dr. Sujan Yenuganti

Assistant Professor BITS Pilani

Department of Electrical & Electronics Engineering

Instructor-in-charge

EEE F246 Electrical and Electronic Circuits Laboratory

Prepared By,

Kunal Gupta	2020B4AA0474P
Devansh Sharma	2021AAPS0602P
Rishab Maloo	2021B2AA2767P
Stiti SamBhab Das	2021B4AA0770P

April 14, 2024

Acknowledgements

We, the students of Group 4, are immensely grateful to our esteemed institution, Birla Institute of Technology and Science, Pilani, for providing us with the platform and resources to undertake this project. We extend our heartfelt appreciation to Dr. Sujan Yenuganti, our instructor In-Charge for the course Electrical and Electronic Circuits Laboratory (EEE F246), whose guidance and support have been invaluable throughout our journey.

Our sincere thanks also go to the entire faculty of the Electrical and Electronic Engineering Department for their encouragement and assistance. Their expertise and encouragement have played a significant role in shaping our understanding and approach towards this project.

Furthermore, we would like to express our gratitude to our peers and colleagues who have offered their insights and feedback, enriching our project experience.

Last but not least, we acknowledge the contributions of all individuals, organizations, and resources that have directly or indirectly facilitated the completion of this report.

This project would not have been possible without the collaborative efforts and support from all the aforementioned individuals and entities. We are deeply thankful for their contributions towards our academic and professional growth.

INDEX

Introduction

Temperature sensing and control are fundamental aspects of various applications, ranging from environmental monitoring to industrial automation. In this project, we explore the utilization of a thermistor, an electronic component whose resistance varies with temperature, for temperature measurement and control. We integrate Arduino, a popular microcontroller platform, with Python programming for data analysis and control algorithm development.

Initially, we leverage the capabilities of Arduino to interface with the thermistor and acquire temperature data through analog-to-digital conversion. Subsequently, Python is employed to perform linear regression analysis on the collected data, enabling us to calibrate the thermistor and obtain accurate temperature readings.

Building upon the temperature sensing capability, we extend the project to incorporate temperature-controlled fan functionality. By interfacing a DC motor with Arduino and utilizing a bipolar junction transistor (BJT) to modulate the collector current, we develop a closed-loop control system. The system adjusts the fan speed based on real-time temperature measurements from the thermistor, ensuring efficient thermal management in diverse environments.

This project combines principles of electronics, programming, and control systems to create a versatile solution for temperature monitoring and regulation. The integration of hardware and software components demonstrates the synergy between different technologies in addressing practical engineering challenges. Through experimentation and analysis, we aim to showcase the effectiveness and adaptability of our approach in various temperature-sensitive applications.

Components used:

Thermistor

A thermistor is a type of electrical resistor whose resistance is dependent on temperature. The word "thermistor" is a combination of "thermal" and "resistor." Unlike typical resistors, which have a constant resistance, the resistance of a thermistor changes significantly with temperature variations.

There are two main types of thermistors:

1. **Negative Temperature Coefficient (NTC) thermistors:** In these thermistors, resistance decreases as temperature increases. They are commonly used in temperature sensing and control applications.
2. **Positive Temperature Coefficient (PTC) thermistors:** In contrast, PTC thermistors exhibit an increase in resistance as temperature rises. They find applications in self-regulating heating elements and overcurrent protection circuits.

Arduino

Arduino is an open-source electronics platform that simplifies the process of creating interactive projects. It consists of a microcontroller board and a development environment, offering an accessible entry point for both beginners and experts in the world of electronics and programming. Arduino boards are equipped with various input and output pins, allowing users to connect sensors, actuators, and other components to build diverse projects. The Arduino programming language, which is based on C/C++, is easy to learn and enables users to write code to control the behavior of their projects. With its vast ecosystem of libraries, tutorials, and community support, Arduino has become a staple tool for makers, hobbyists, educators, and professionals alike, facilitating innovation in fields ranging from home automation and robotics to Internet of Things (IoT) applications and beyond.

Two Arduino boards have been used in the project. The first Arduino board is where all the implementation has been done, while the second Arduino board is used for 5 volt supply to the motor.

BC547B

NPN general purpose transistors, especially suited for use in driver stages of audio amplifiers, low noise input stages of tape recorders, HI-FI amplifiers, and signal processing circuits of television receivers.

DC Motor

A DC motor, short for direct current motor, is a device that converts electrical energy into mechanical energy. It operates on the principle of electromagnetism, where the interaction between magnetic fields and electric currents generates rotational motion. It consists of a stationary stator with permanent magnets, or electromagnets, and a rotating rotor with conductors. When current flows through the rotor conductors, a magnetic field is produced, interacting with the stator's field to create torque, thus causing rotation. Commutation, facilitated by a commutator and brushes, ensures continuous rotation by reversing the current direction in the rotor windings. By controlling voltage or magnetic field strength, the speed and torque of the motor can be adjusted, making DC motors versatile for various applications. DC motors are commonly used in a wide range of applications due to their simplicity, reliability, and controllability.

Theory

Steinhardt equation

The Steinhardt equation, also known as the Steinhart-Hart equation, is an empirical formula used to describe the temperature-dependent resistance of a thermistor. It is commonly expressed as:

$$\frac{1}{T} = A + B \ln R + C(\ln R)^3$$

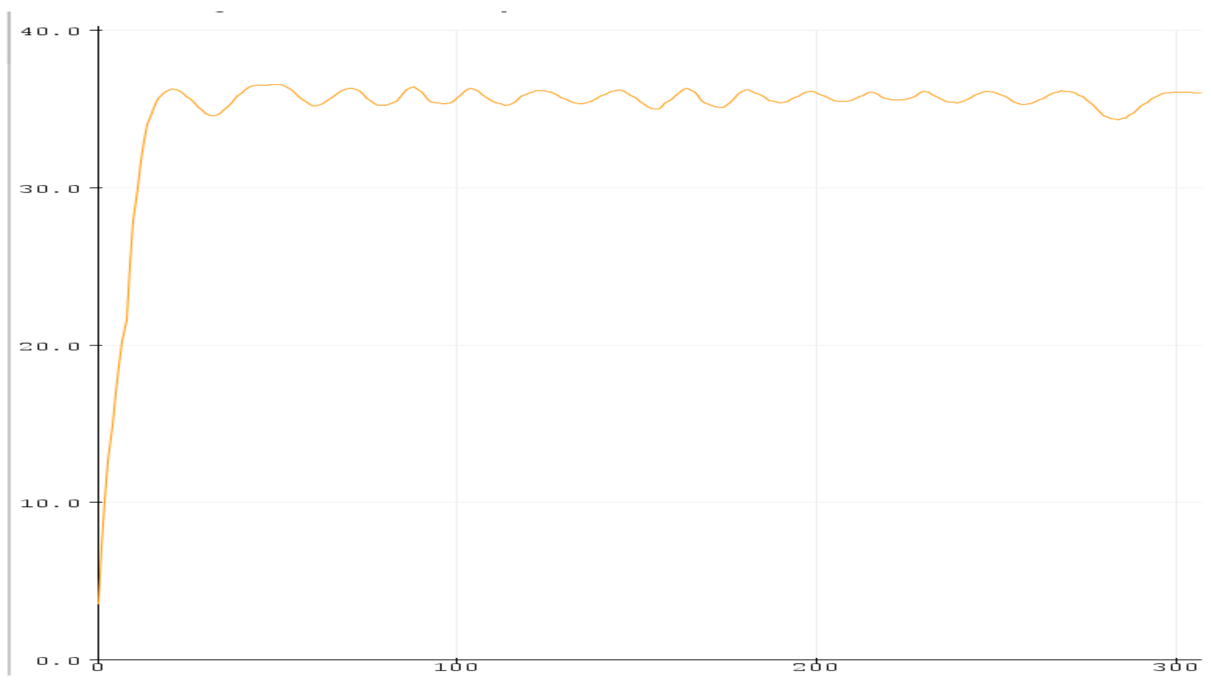
where:

- *T* is the absolute temperature in Kelvin,
- *R* is the resistance of the thermistor at temperature *T*,
- *A*, *B*, and *C* are coefficients determined through calibration.

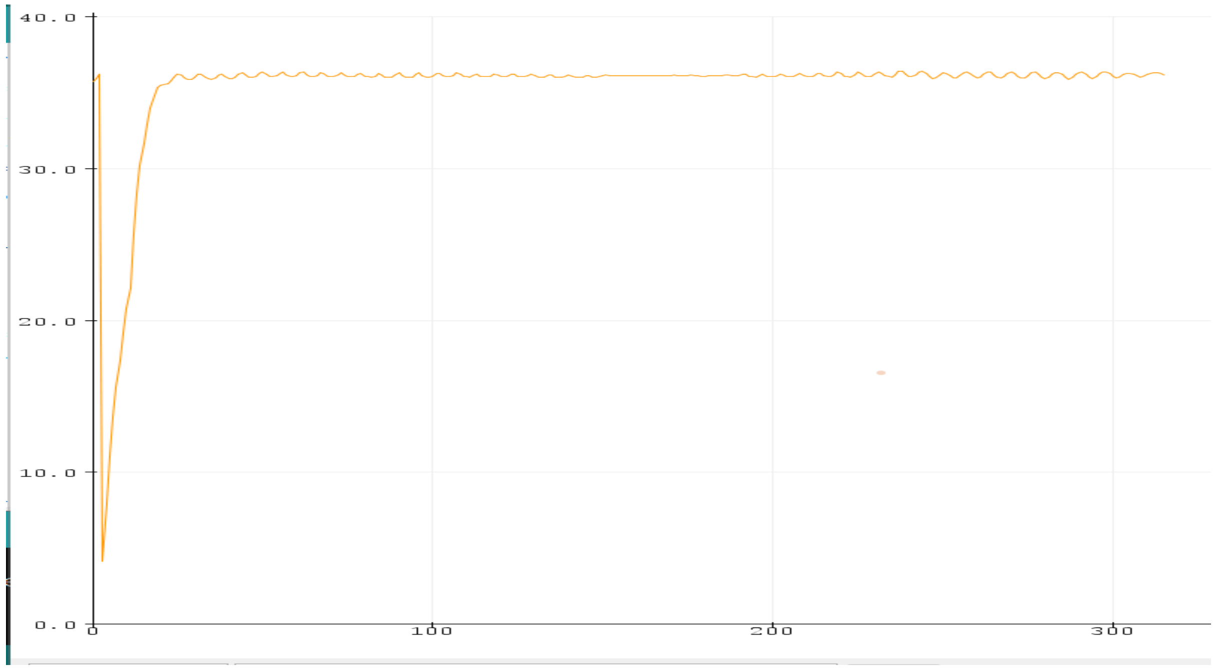
The Steinhart-Hart equation is used because the resistance of thermistors varies nonlinearly with temperature, making it challenging to directly relate resistance to temperature using a simple linear equation. The Steinhart-Hart equation provides a more accurate representation of this nonlinear relationship, allowing for precise temperature measurement or control using thermistors.

Temperature Readings from Thermistor

This Arduino code reads temperature from a thermistor connected to pin A3. It utilizes the Steinhart-Hart equation to convert analog readings (ADCvalue) to temperature in Celsius. The moving average filter is applied to stabilize temperature measurements by averaging past readings. The filter maintains a buffer of readings, updating the average with each new reading. It prints the average temperature to the serial monitor every second. The setup function initializes the serial communication and readings array. In the loop function, it reads the ADC value, calculates temperature, updates the moving average, and prints the result. If the ADC value is invalid (zero or negative), it prints an error message. The delay ensures a consistent sampling interval.



Temperature reading without moving average filter



Temperature reading with moving average filter

Linear Regression

Linear regression was used in the procedure you described to model the relationship between temperature and voltage. Here's why it was used:

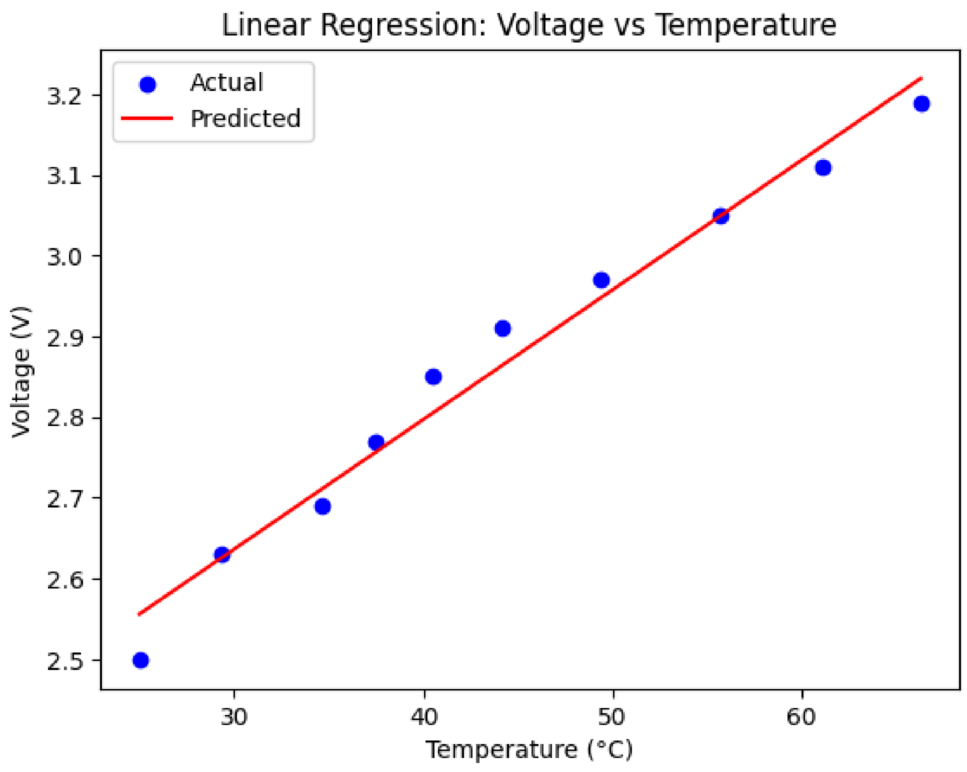
1. Modeling Relationship: The data provided consists of temperature readings and corresponding voltage values. Linear regression allows us to model the relationship between these two variables. Specifically, it helps us understand how changes in temperature affect the voltage output.
2. Predictive Analysis: Once we have established a linear relationship between temperature and voltage through regression, we can use this model to predict voltage values for given temperature values. This predictive capability can be valuable for various applications, such as temperature monitoring and control systems.
3. Understanding Linearity: Linear regression assumes that the relationship between the independent variable (temperature) and the dependent variable (voltage) is linear. By performing linear regression, we assess the validity of this assumption. If the relationship is linear, linear regression provides an accurate and interpretable model.
4. Quantifying the Relationship: By fitting a linear regression model, we obtain estimates for the slope and intercept parameters. These parameters quantify the strength and direction of the relationship between temperature and voltage. This information is valuable for understanding the system and making informed decisions based on the data.

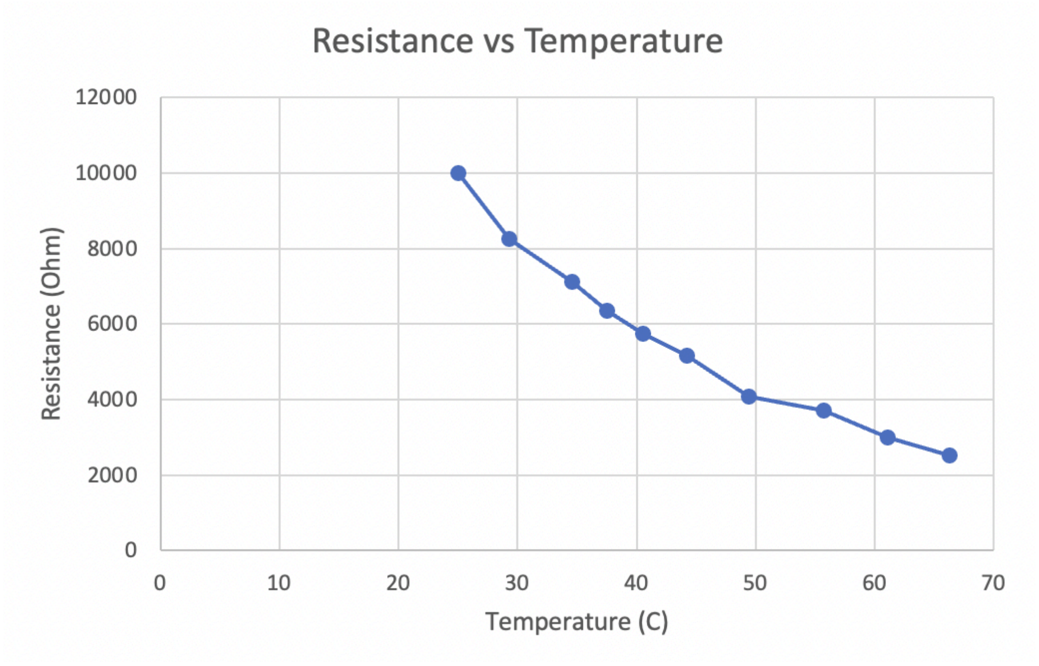
This Python code performs linear regression on the provided data to model the relationship between temperature and voltage. It first creates a DataFrame df from the provided data, containing columns for temperature (Temp), resistance (R), and output voltage (Vout). The linear regression is then performed using scikit-learn's LinearRegression model.

The independent variable (X) is the temperature (Temp), while the dependent variable (y) is the output voltage (Vout). The model is trained on the temperature and voltage data, and then used to predict voltages for the given temperatures.

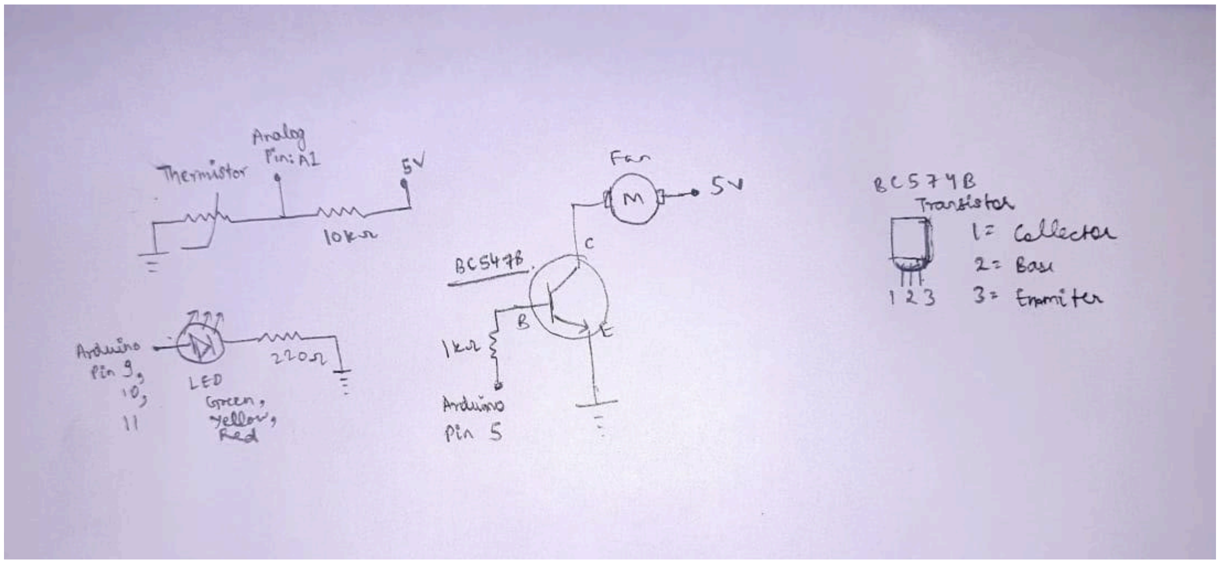
The model coefficients (intercept and coefficient) are printed to the console. Finally, the actual and predicted voltages are plotted against temperature using matplotlib. The scatter plot shows the actual data points, while the red line represents the linear regression model's predictions.

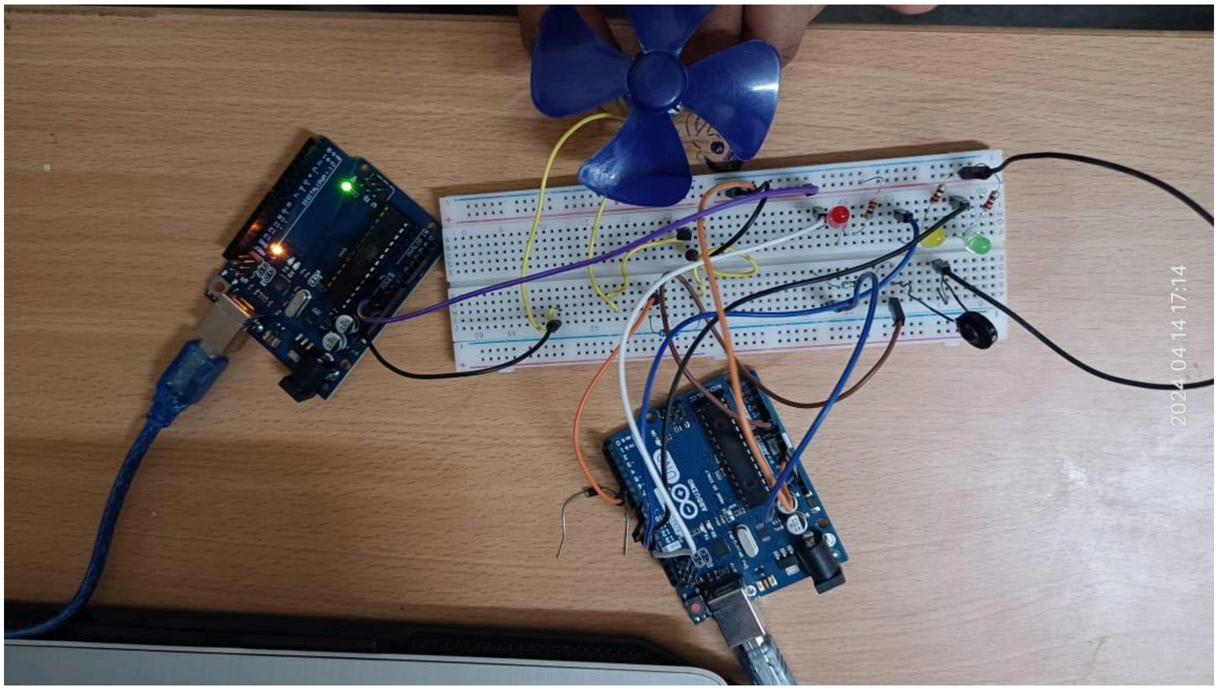
```
Intercept: 2.155022694100251
Coefficient: 0.016049984353015086
```




Resistance vs Temperature
Resistance (Ohm)
12000
10000
8000
6000
4000
2000
0
0
10
20
30
40
50
60
70
Temperature (C)

Circuit Diagram





Future Ideas

- We can integrate connectivity features like WiFi or Bluetooth for remote data monitoring and control via a smartphone app that could provide user-friendly access to real-time and historical data, along with the ability to adjust settings from anywhere.
- Data logging with an SD card or cloud storage can be used for the analysis of temperature trends and system performance over time, which could be helpful for optimizing energy usage in climate control applications.
- By adding additional environmental sensors, such as humidity, air quality, or light sensors, we can transform the project into a comprehensive home environmental monitoring system, providing a holistic view of indoor conditions.
- We can add voice control capabilities through integration with smart home assistants like Amazon Alexa or Google Assistant, making it more accessible for all users.
- Applying machine learning algorithms to predict future temperature changes and automate system responses could significantly improve efficiency, potentially learning user preferences and adjusting the environment preemptively for optimal comfort and energy consumption.

Conclusion

This report concludes that thermistors are highly effective for temperature sensing due to their significant resistance change with temperature variations. The Arduino platform proved to be a useful tool for interfacing with the thermistor and facilitating data acquisition and processing. The application of the Steinhardt equation allowed for precise temperature reading and served as a reliable method for modeling and predicting the relationship between temperature and voltage.

The dual utilization of Arduino boards—one for the main implementation and the other for powering the DC motor—demonstrated the expandability of the Arduino ecosystem in experimental setups.

The findings from this study highlight the potential of integrating simple electronic components like thermistors and Arduino boards in more complex systems, such as automated temperature control or adjustable-speed motor drives.

In summary, the project not only supported theoretical predictions about thermistor behavior but also provided a practical demonstration of electronic component interfacing and data analysis techniques, which are important in the field of electronics and instrumentation engineering.

Applications

- **HVAC Systems:** Heating, Ventilation, and Air Conditioning systems utilise thermistor-based temperature control to maintain comfortable indoor temperatures in homes, offices, and commercial buildings.
- **Refrigeration and Freezers:** Refrigerators and freezers employ thermistors to regulate the internal temperature, ensuring proper storage conditions for perishable goods.
- **Medical Equipment:** Thermistor-based temperature control is crucial in medical devices such as incubators, blood storage units, and temperature-controlled chambers, where precise temperature regulation is essential for preserving biological samples and medications.
- **Automotive:** Automobiles use thermistors for engine temperature monitoring and climate control systems, helping to maintain engine efficiency and passenger comfort.
- **Industrial Processes:** Industries rely on thermistors for temperature control in various manufacturing processes, including chemical reactions, food processing, and semiconductor fabrication.
- **Food Service Industry:** Commercial kitchens, food warmers, and food display cases utilise thermistor-based temperature control to maintain safe serving temperatures and comply with health regulations.
- **Laboratory Equipment:** Scientific instruments such as ovens, incubators, and chromatography systems rely on thermistors for accurate temperature control during experiments and sample processing.
- **Aquariums and Vivariums:** Tanks for aquatic life and reptiles employ thermistor-based temperature control to maintain the appropriate water or air temperature for the inhabitants' health and well-being.

References

1. Linearization of NTC Thermistor Characteristic Using Op-Amp Based Inverting Amplifier

   https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6527951

2. Linearization Circuit of the Thermistor Connection

   https://ieeexplore.ieee.org/abstract/document/4620168

3. Design of Linearizing Circuit for Thermistor

   https://www.irjet.net/archives/V7/i5/IRJET-V7I5239.pdf

4. Methods to Reduce Thermistor Linearization Error, Memory, and Power Requirements Over Wide Operating Temperature Ranges

   https://www.ti.com/lit/an/snoaa12/snoaa12.pdf?ts=1713156629833&ref_url=https%253A%252F%252Fwww.google.com%252F

Code

Arduino Code for sensing temperature

```
const int ThermistorPin = A3;

const float SeriesResistor = 10000;

const float ThermistorNominal = 10000;

const float TemperatureNominal = 25;

const float BValue = 3950;

const float KelvinConversion = 273.15;

const int numReadings = 10;

int readingsIndex = 0;

float readingsSum = 0;

float readings[numReadings];

bool firstCycle = true;


void setup() {

Serial.begin(9600);

for (int thisReading = 0; thisReading < numReadings; thisReading++)

    readings[thisReading] = 0;

}

// Function to calculate temperature from ADC value

float calculateTemperature(int ADCvalue) {

        float Resistance = SeriesResistor / ((1023.0 / ADCvalue) - 1);

        float Steinhart = Resistance / ThermistorNominal;

        Steinhart = log(Steinhart);

        Steinhart /= BValue;
        Steinhart += 1.0 / (TemperatureNominal + KelvinConversion);

        Steinhart = 1.0 / Steinhart;

        return Steinhart - KelvinConversion;

}


void loop() {

        int ADCvalue = analogRead(ThermistorPin);

        if (ADCvalue > 0) {

                float temperature = calculateTemperature(ADCvalue);

                readingsSum -= readings[readingsIndex];

                readings[readingsIndex] = temperature;

                readingsSum += readings[readingsIndex];

                readingsIndex++;

                if (readingsIndex >= numReadings) {

                        readingsIndex = 0;

                        firstCycle = false;

                }

                float averageTemperature = readingsSum / (firstCycle ? (readingsIndex + 1) : nu
                Readings);

                Serial.print("Average Thermistor Temperature: ");

                Serial.print(averageTemperature);

                Serial.println(" C");

        }

        else {

                Serial.println("Invalid thermistor reading.");

        }

        delay(1000);
```

```
}
```

Arduino code for varying fan speed with temperature

```
#include <math.h>

// Analog pin where thermistor is connected

const int ThermistorPin = A3;

// Pin for LED 1-green (20°C to 30°C)

const int LEDPin1 = 9;

// Pin for LED 2-yellow (30°C to 40°C)

const int LEDPin2 = 10;

 // Pin for LED 3-red (40°C to 50°C)

const int LEDPin3 = 11;

// PWM pin for controlling the fan

const int FanPin = 5;

const float SeriesResistor = 10000;

const float ThermistorNominal = 10000;

const float TemperatureNominal = 25;

const float BValue = 3950;

const float KelvinConversion = 273.15;

// Smoothing Parameters

const int numReadings = 10;

int readingsIndex = 0;

float readingsSum = 0;

float readings[numReadings];

bool firstCycle = true;
```

```
void setup() {
	Serial.begin(9600);
	pinMode(LEDPin1, OUTPUT);
	pinMode(LEDPin2, OUTPUT);
	pinMode(LEDPin3, OUTPUT);
	pinMode(FanPin, OUTPUT);

	for (int thisReading = 0; thisReading < numReadings; thisReading++) {
		readings[thisReading] = 0;
	}
}

float calculateTemperature(int ADCvalue) {
	if (ADCvalue == 0) ADCvalue = 1;
	float Resistance = SeriesResistor / ((1023.0 / ADCvalue) - 1);
	float Steinhart = Resistance / ThermistorNominal;
	Steinhart = log(Steinhart);
	Steinhart /= BValue;
	Steinhart += 1.0 / (TemperatureNominal + KelvinConversion);
	Steinhart = 1.0 / Steinhart;
	return Steinhart - KelvinConversion;
}

void loop() {
	int ADCvalue = analogRead(ThermistorPin);
	if (ADCvalue > 0) {
```

```
float temperature = calculateTemperature(ADCvalue);

readingsSum -= readings[readingsIndex];

readings[readingsIndex] = temperature;

readingsSum += readings[readingsIndex];

readingsIndex++;

if (readingsIndex >= numReadings) {

        readingsIndex = 0;

        firstCycle = false;

}

// Calculate the average temperature

float averageTemperature = readingsSum / (firstCycle ? (readingsIndex + 1) :
numReadings);

// Control LEDs based on temperature range

digitalWrite(LEDPin1, LOW);

digitalWrite(LEDPin2, LOW);

digitalWrite(LEDPin3, LOW);

if (averageTemperature >= 20 && averageTemperature < 30) {

        digitalWrite(LEDPin1, HIGH);

}

else if (averageTemperature >= 30 && averageTemperature < 40) {

        digitalWrite(LEDPin2, HIGH);

}

else if (averageTemperature >= 40 && averageTemperature <= 50) {

        digitalWrite(LEDPin3, HIGH);

}

// Map temperature to PWM values for fan speed

int pwmValue = map(averageTemperature, 20, 50, 0, 255);
```

```
                analogWrite(FanPin, pwmValue);
                Serial.print("Average Thermistor Temperature: ");
                Serial.print(averageTemperature);
                Serial.println(" C");
        }
        else {
                Serial.println("Invalid thermistor reading.");
        }
        delay(1000);
}
```

Code for linear regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
# Create a DataFrame from the provided data
data = {
  'Temp': [25, 29.3, 34.6, 37.5, 40.5, 44.2, 49.4, 55.7, 61.1, 66.3],
  'R': [10000, 8260, 7125, 6350, 5740, 5160, 4080, 3704, 2995, 2520],
  'Vout': [2.5, 2.63, 2.69, 2.77, 2.85, 2.91, 2.97, 3.05, 3.11, 3.19]
}
df = pd.DataFrame(data)
X = df['Temp'].values.reshape(-1, 1)
y = df['Vout'].values
```

```
model = LinearRegression()

model.fit(X, y)

predicted_voltages = model.predict(X)

print("Intercept:", model.intercept_)

print("Coefficient:", model.coef_[0])


# Plotting the results

plt.scatter(df['Temp'], df['Vout'], color='blue', label='Actual')

plt.plot(df['Temp'], predicted_voltages, color='red', label='Predicted')

plt.title('Linear Regression: Voltage vs Temperature')

plt.xlabel('Temperature (°C)')

plt.ylabel('Voltage (V)')

plt.legend()

plt.show()
```