Digital Thermometer

Balu – Ai25BTECH11017
Manohar – Ai25BTECH11028

AIM :-

The objective of this project is to design and Implement a digital thermometer that measures temperature using a PT-100 Resistance Temperature Detector (RTD), Processes the signal through an Arduino microcontroller, and Displays the Temperature on a 16x2 LCD. The Temperature is Determined using linear regression (least squares method).

Components :-

- PT-100 RTD
- Arduino Uno
- Jumper wires
- Bread board
- Potentiometer
- LCD (16x2)
- 100Ω Resistor

Procedure :-

1) Build the circuit.

2) Write arduino code to measure voltage.

3) ~~Take~~ Take data set of temparature and voltage using thermometer.

4) Callendur Van dusen equations.

   1) $V = n_0 + n_1 T + n_2 T^2$

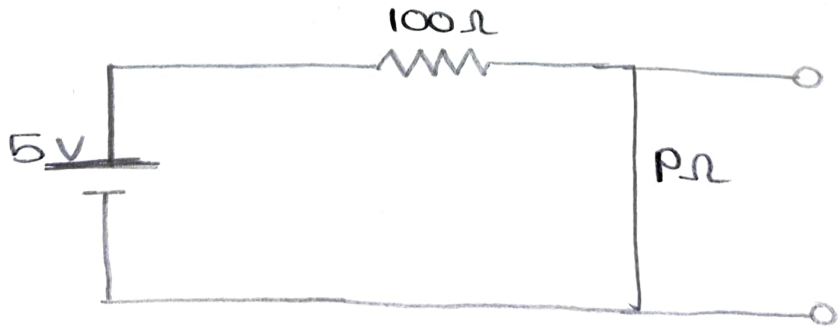
   2) $T = a_0 + a_1 V + a_2 V^2$

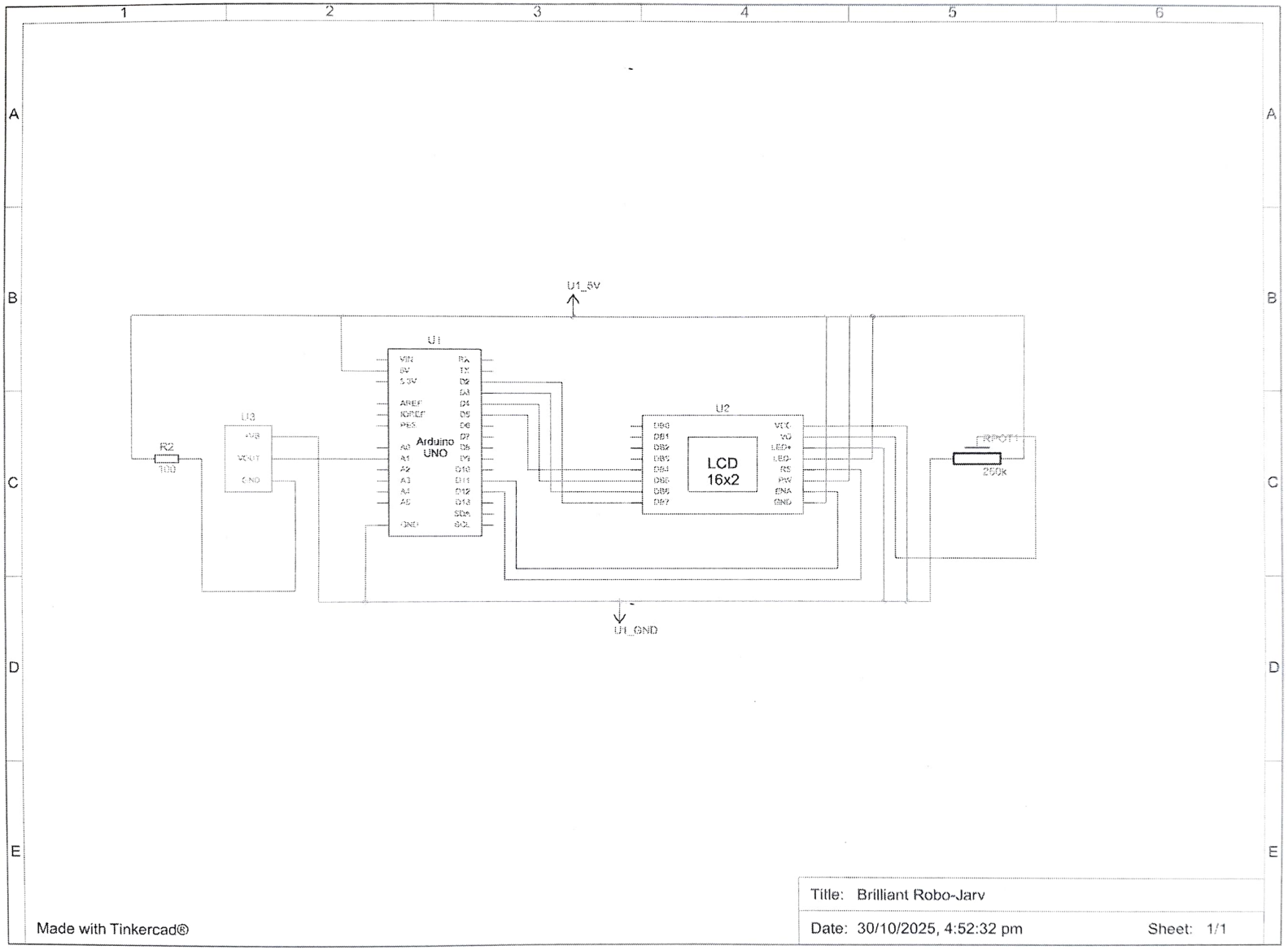
5) Use least squares method on data set and find the coefficents.

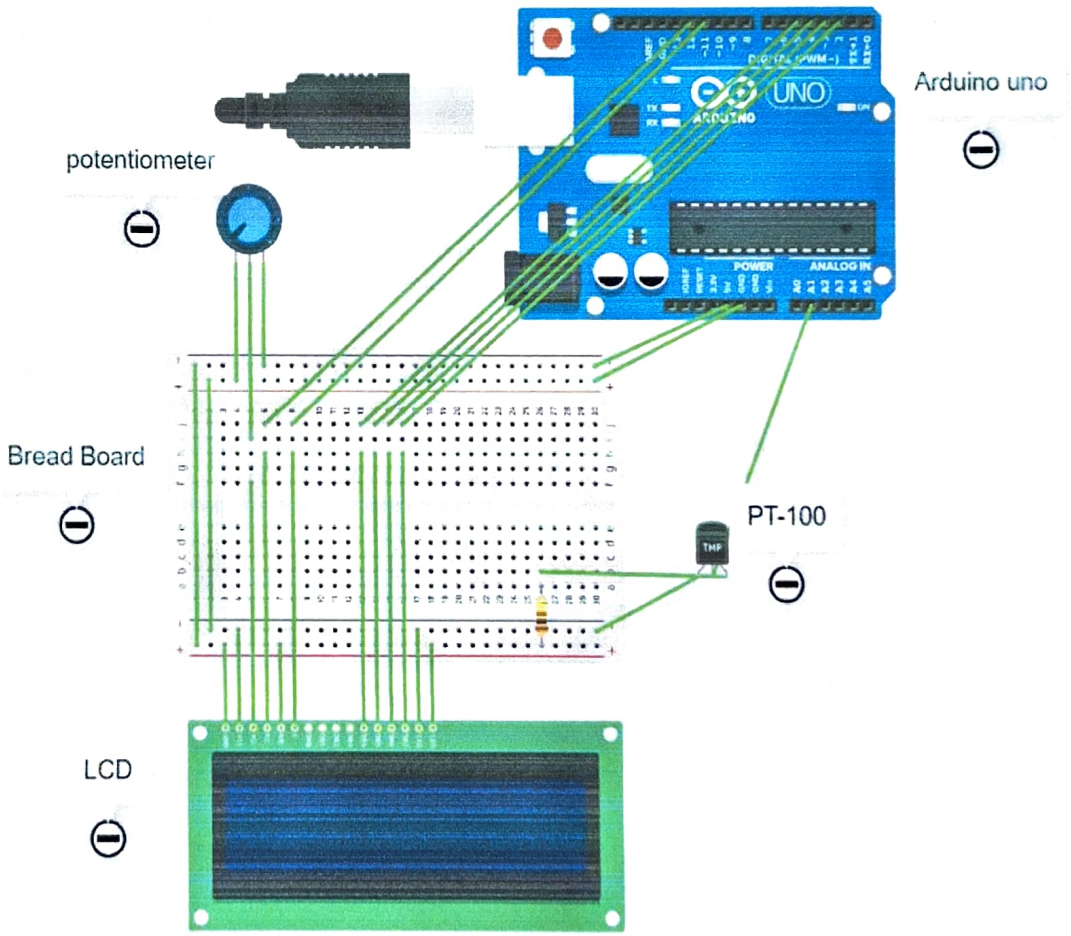
6) Validate this model using 10 data points (T, V).

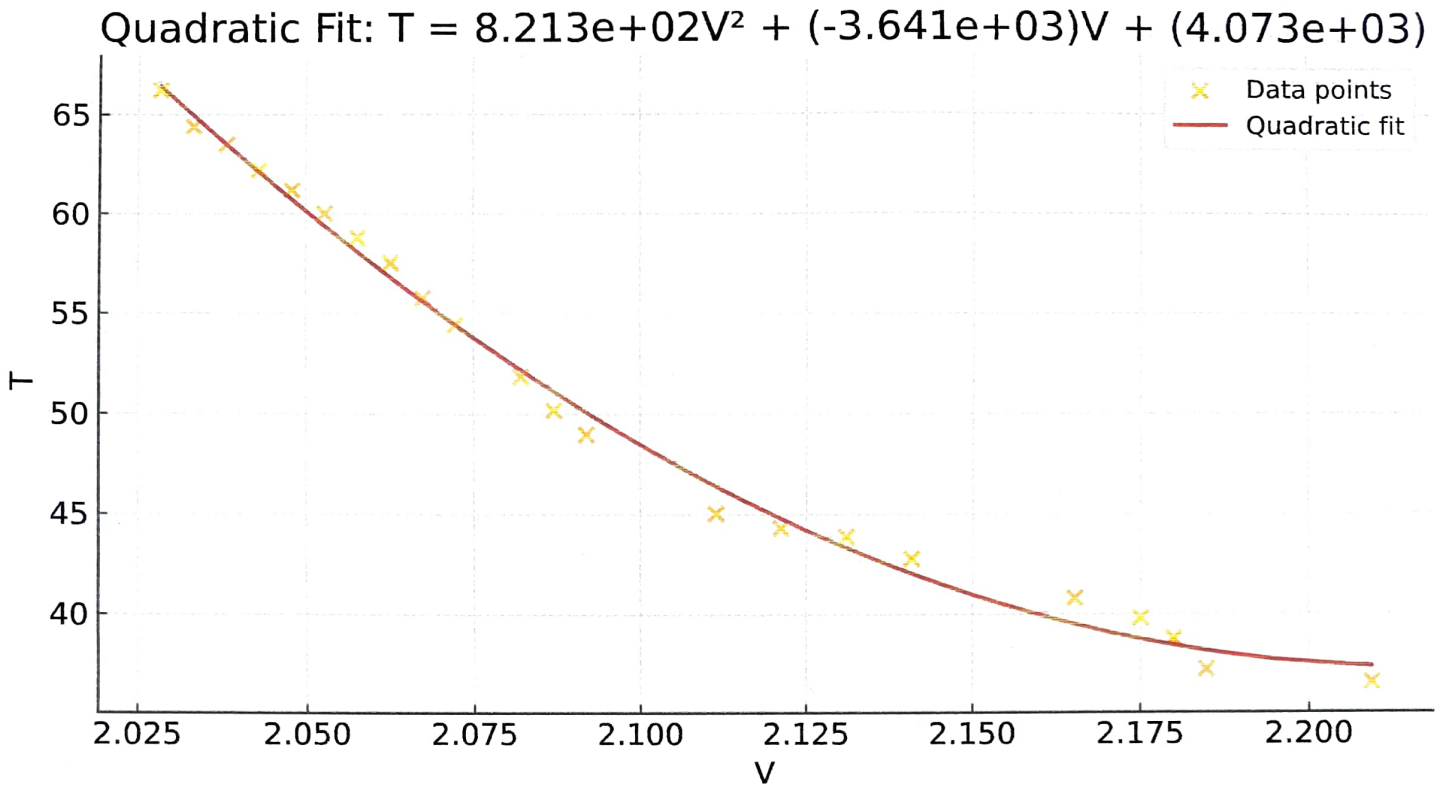
7) Calculate mean Absolute error (M.A.E) and analyze parameters.

Circuit :-




1
2
3
4
5
6
A
B
C
D
E
U1_5V
U1
Arduino
UNO
U3
R2
100
U2
LCD
16x2
RPOT1
250k
U1_GND
Made with Tinkercad®
Title: Brilliant Robo-Jarv
Date: 30/10/2025, 4:52:32 pm
Sheet: 1/1


potentiometer
Arduino uno
Bread Board
PT-100
LCD



V	T
2.0283	66.25
2.0332	64.4
2.0381	63.5
2.043	62.2
2.0479	61.2
2.0528	60.0
2.0577	58.8
2.0626	57.54
2.0674	55.77
2.0723	54.45
2.0821	51.85
2.087	50.2
2.0919	49.0
2.1114	45.06
2.1212	44.34
2.131	43.9
2.1408	42.8
2.1652	40.8
2.175	39.8
2.1799	38.76
2.1848	37.23
2.2092	36.58

Method of Solving to find coefficents :-

let the measured data points be (T_i, V_i) $i = 1, 2, \dots n$

Callendor van Dusen equation

$$V = n_0 + n_1 T + n_0 T^2$$

In matrix form

$$C = X n$$

where $C = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$ $X = \begin{pmatrix} 1 & T_1 & T_1^2 \\ \vdots & \vdots & \vdots \\ 1 & T_n & T_n^2 \end{pmatrix}$ $n = \begin{pmatrix} n_0 \\ n_1 \\ n_2 \end{pmatrix}$

In linear regression we tend to minimize the Sum of Squared residuals

$$E(n) = \|C - Xn\|^2 = (C - Xn)^T (C - Xn)$$

$$= (C^T - n^T X^T)(C - Xn)$$

$$= \|C\|^2 - C^T Xn - n^T X^T C - \|n\|^2 \|n\|^2 \|X\|^2$$

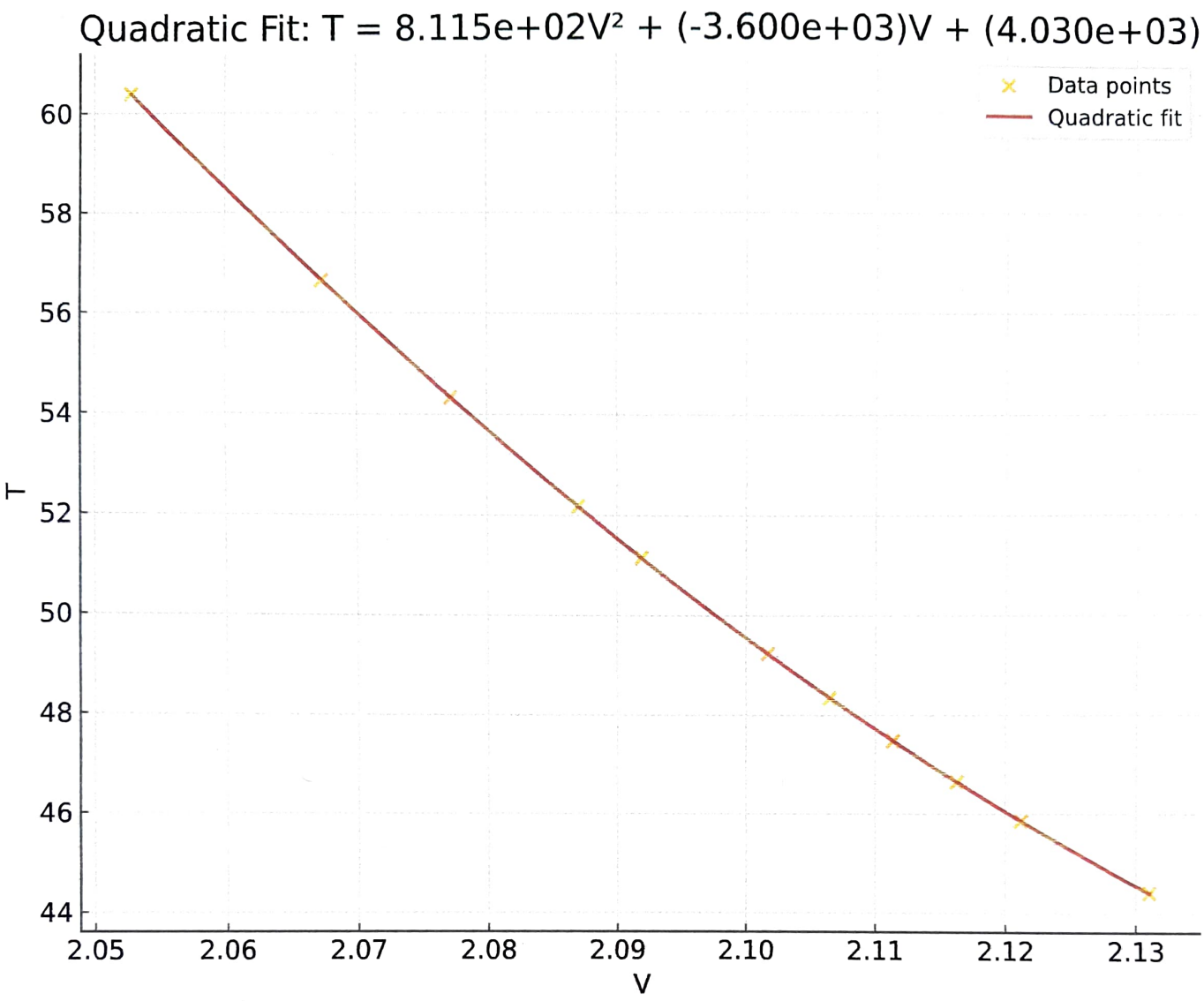
we want $\frac{d(E(n))}{dn} = 0$

$$n = (X^T X)^{-1} X^T V$$

we got $\begin{pmatrix} n_0 \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} 2.773782 \\ -2.134048e-02 \\ 1.540037e-04 \end{pmatrix}$

For $T = a_0 + a_1 V + a_2 V^2$

we got $\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 4026.380134 \\ -3597.164799 \\ 810.934776 \end{pmatrix}$



V	T
2.0528	60.39
2.0674	56.64
2.0772	54.33
2.087	52.18
2.0919	51.16
2.1017	49.24
2.1065	48.34
2.1114	47.48
2.1163	46.65
2.1212	45.86
2.131	44.41

Calculating error :-

By mean Abosute error (M.A.E) for 10 data points

$$MAE = \frac{\sum_{i=1}^{10} |T_i - T_i'|}{10}$$

T_i – Temparature by model

T_i' – Temparature by Theromeler

$$= 2.511$$

Performance of model :-

The model is moderate – good with deviation of 2.5°C.

Source of errors :-

1) availability of poor quality and defect materials.
2) providing small data set to model.
3) Human errors while measuring.
4) Approximation errors in the least squares method.
5) Sensor material uncertainties.

Arduino Code

```
#include <LiquidCrystal.h>

// Initialize LCD with pin numbers: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12,11,5,4,3,2);
float a0 = 4026.380134;
float a1 = -3597.164799;
float a2 = 810.934776;
// Pin definitions
Const int PT100_PIN = A1;
// Constants for PT100 and voltage divider
Const float VCC = 5.0;
// Constants
//calibration Parameters (adjust after calibration)
float offset = 1.0;
float sensitivity = 1.0;
Void setup() {
 Serial.begin(9600);
// Initialize LCD, 16 columns and 2 rows
 lcd.begin(16,2);
// Display startup message
 lcd.Print("PT100 Temp");
 lcd.clear();
```

```
analogReference (DEFAULT);
}

Void loop() {
 // Read analog value from voltage divider
 int adcValue = analogRead (PT100_PIN);
 // Convert ADC value to voltage
 float voltage = (adcValue * Vcc) / 1023.0;
 float temperature = a0 + a1 * voltage + a2 * voltage * voltage;
 //Calibrating the temperature by IC
  temperature = (temperature * Sensitivity) + offset;
 // Show temperature on LCD
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temp: ");
  lcd.print (temperature, 2);
  lcd.print ("C");
  lcd.setCursor (0, 1);
  lcd.print ("V: ");
  lcd.print (voltage, 4);
  lcd.print ("V");
  delay (1000);
}
```

linear regression code

```
import numpy as np

T = np.array([66.25, 64.40, 63.50, 62.20, 61.20, 60.00, 58.80
 57.54, 55.77, 54.45, 51.85, 50.20, 49.00, 45.06, 44.34,
 43.9, 42.8, 40.8, 39.80, 38.76, 37.23, 36.58])

V = np.array([2.0283, 2.0332, 2.0381, 2.0430, 2.0479,
2.0528, 2.0577, 2.0626, 2.0674, 2.0723, 2.0821, 2.0870, 2.0919,
2.114, 2.1212, 2.1310, 2.1408, 2.1652, 2.1750, 2.1799, 2.1848,
 2.2092])
# V = n0 + n1* T + n2* T**2
X = np.vstack((np.ones_like(T), T, T**2)).T
coeffs = np.linalg.lstsq(X, V, rcond=None)[0]
n0, n1, n2 = coeffs
print("voltage model: V(T) = {:.6f} + {:.6f}*T + {:.6f}*T**2.
 format(n0, n1, n2))
# T = a0 + a1*V + a2*V**2
X_inv = np.vstack((np.ones_like(V), V, V**2)).T
coeffs_inv = np.linalg.lstsq(X_inv, T, rcond=None)[0]
a0, a1, a2 = coeffs_inv
print("Temperature Model: T(V) = {:.6f} + {:.6f}*V + {:.6f}*V**2.
 format(a0, a1, a2))
```