DIGITAL THERMOMETER USING PT-100

Harsha B J and Bhargav K

Contents

I. Objective

The objective of this project is to design and implement a digital thermometer that measures the temperature using a PT-100 Resistance Temperature Detector (RTD), processes the signal through an Arduino microcontroller, and displays the temperature on a 16×2 LCD. In this experiment, the relationship between the voltage across the PT-100 and the temperature is determined using linear regression (least squares method).

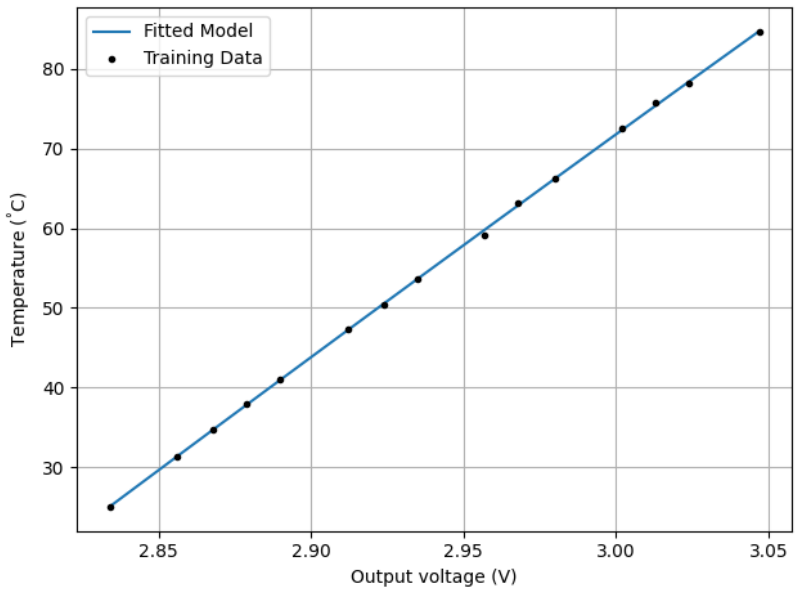
II. Training Data

The training data gathered by the PT-100 to train the Arduino is shown in Table I.

Temperature (°C)	**Voltage (V)**
25.00	2.834
31.31	2.856
34.46	2.868
37.61	2.879
40.76	2.890
47.07	2.912
50.22	2.924
53.37	2.935
59.68	2.957
62.83	2.968
65.98	2.980
72.29	3.002
75.44	3.013
78.59	3.024
84.90	3.047

TABLE I
Training data.

The C++ source `codes/data.cpp` was used along with *platformio* to drive the Arduino. The approximation is shown in Fig. II.



III. Theory

The PT-100 sensor changes resistance with temperature. Its nominal resistance is 100 Ω at 0 °C, and the resistance increases approximately linearly with temperature:

$$R_T = R_o(1 + \alpha T) \tag{1}$$

where $\alpha = 0.00385\ °C^{-1}$. When placed in a Wheatstone bridge circuit, the resistance variation produces a corresponding voltage change, which can be measured and used to infer the temperature.

IV. Linear Regression Model

To obtain an empirical model relating the measured voltage V to temperature T, we collect calibration data by measuring both quantities over a range of known temperatures. Let the measured data points be $(T_i\ ,\ V_i)$, i = 1, 2,. . , n. We assume a quadratic model for the voltage-temperature relationship

$$V(T) = n_0 + n_1 T + n_2 T^2 \tag{2}$$

$$\tag{3}$$

$$\implies \mathbf{C} = \mathbf{X^T n} \tag{4}$$

where

$$\mathbf{X^T} = \begin{pmatrix} 1 & T_1 & T_1^2 \\ 1 & T_2 & T_2^2 \\ . & . & . \\ . & . & . \\ . & . & . \\ 1 & T_n & T_n^2 \end{pmatrix}, \mathbf{C} = \begin{pmatrix} V_1 \\ V_2 \\ . \\ . \\ . \\ V_n \end{pmatrix} \tag{5}$$

V. Solution

We approximate $\mathbf{n^T} = \begin{pmatrix} n_0 & n_1 & n_2 \end{pmatrix}$ using the least squares method. Using the pseudo-inverse method, the solution to (4) is

$$\mathbf{n} = \left(\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{X}\mathbf{C} \tag{6}$$

The Python code `codes/main.py` solves for **n**. The calculated value of **n** is

$$\mathbf{n} = \begin{pmatrix} 2.7451 \\ 3.5566 \times 10^{-3} \\ -5.0234 \times 10^{-8} \end{pmatrix} \tag{7}$$

To obtain temperature as a function of measured voltage (for Arduino implementation), we rearrange or numerically invert the above relation:

$$T(V) = a_0 + a_1 V + a_2 V^2 \tag{8}$$

The coefficients a_i can again be found by applying the least squares method to the $(V_i,\ T_i)$ data, which yields

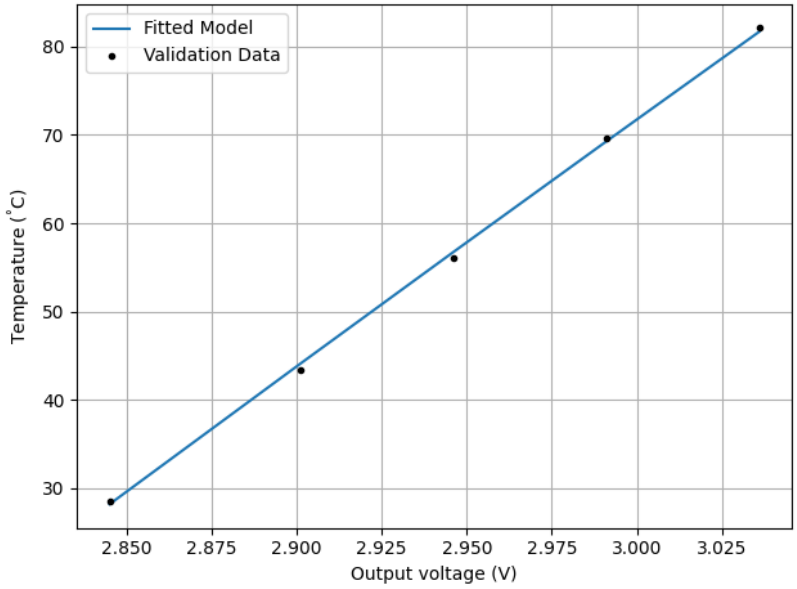
$$\mathbf{n} = \begin{pmatrix} -996.9724 \\ 435.5577 \\ -26.4382 \end{pmatrix} \tag{9}$$

VI. Validation

The validation data set is shown in Table II. The results of the validation are shown in Fig. VI.

Temperature (°C)	**Voltage (V)**
28.15	2.845
43.92	2.901
56.53	2.946
69.14	2.991
81.75	3.036

TABLE II
Validation data.



VII. Conceptual logic implemented in the code

The provided program implements a quadratic regression model to establish the relationship between temperature and output voltage. It begins by loading the training data consisting of temperature (in °C) and voltage (in V) readings. A design matrix is then constructed using three terms: a constant, the temperature, and the square of the temperature. By applying the least squares method, the program determines the model coefficients that best fit the data. This fitted model is then used to generate predicted voltage values, which are plotted against the actual training data to visualize the accuracy of the fit.

Subsequently, the program loads a separate validation dataset to evaluate the performance of the derived model on unseen data. The fitted curve and the actual validation data points are plotted together, allowing a clear comparison of the predicted and measured values. The resulting plots are saved for documentation, and the computed model parameters are printed for further analysis. In essence, the code effectively demonstrates the use of polynomial regression for sensor calibration and performance verification.

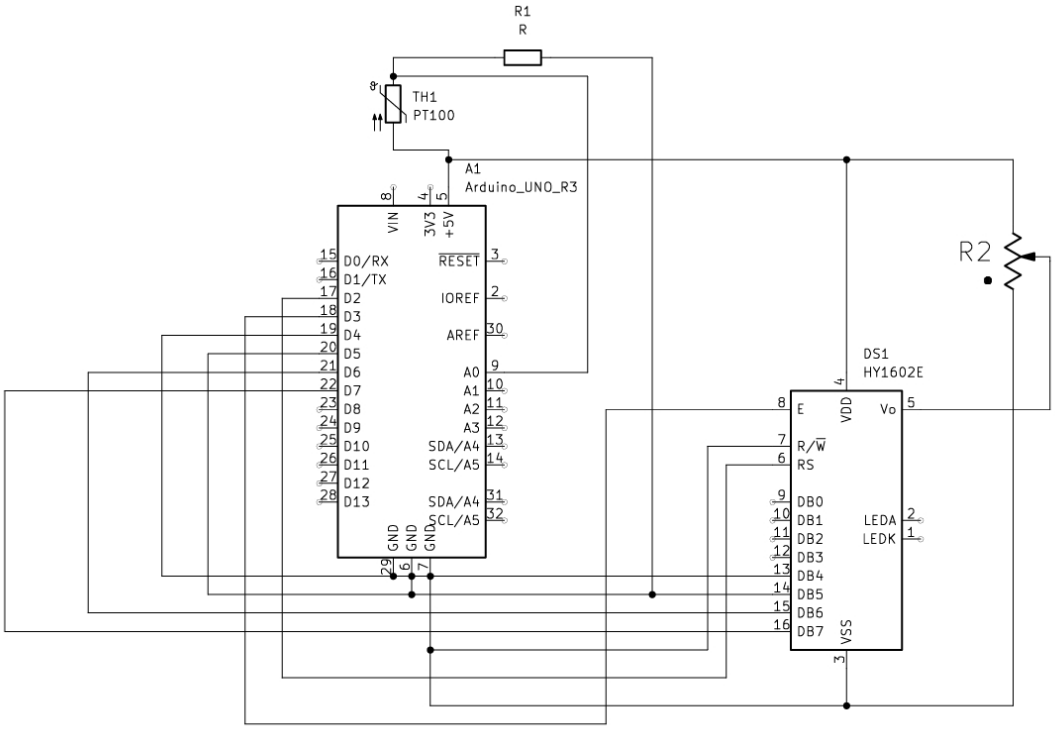
VIII. Observation

The observations made while conducting the experiment are as follows,

1) The readings of arduino had an offset of a few millivolts for every cycle of reading. Solution: We wrote an algorithm to take an average of a few sample readings and computed the temperature using it which gave a good precision.

2) The values of the temperature had an offset of ± 3° celcius due to the hardware malfunctioning.
3) the accuracy of the training model and validation data is close enough with a less validation.

IX. Circuit Diagram



X. Conclusion

This project used Python for machine learning to calibrate the temperature sensor with linear regression. Arduino collected sensor data and showed real-time results. Python made the analysis easier and more accurate, while Arduino handled the hardware side. This simple combination of Python and Arduino creates a smart, affordable device by merging data science and embedded systems effectively.