High-Precision Temperature Measurement using PT100

Puni Aditya, Vivek K Kumar and Dr. G.V.V. Sharma
Department of Electrical Engineering,
Indian Institute of Technology Hyderabad,
Kandi, India 502284
gadepall@ee.iith.ac.in

***Abstract*—This paper details the design, implementation, and validation of a temperature measurement system using a PT100 resistance temperature detector (RTD), a high-precision ADS1115 ADC, and an Arduino microcontroller. The system employs a differential voltage reading from a simple divider circuit for signal acquisition. A quadratic model, $V = aT^2 + bT + c$, was derived using the Least Squares method on a training dataset. This model was then validated against a separate, unseen dataset to verify its accuracy. The final calibrated temperature is displayed on a JHD 162A parallel LCD, creating a validated and accurate measurement device.**

I. Introduction

Accurate temperature measurement is critical in various scientific and industrial applications. While many sensors exist, platinum resistance thermometers like the PT100 offer high accuracy and stability over a wide temperature range. However, their small resistance change necessitates precise measurement techniques. This project moves beyond the Arduino's internal ADC by interfacing with an external 16-bit ADS1115 ADC to achieve higher resolution. This paper presents the hardware setup, the software implementation, and the mathematical modeling used to calibrate the system for optimal accuracy.

II. Hardware Setup

The components used to construct the temperature measurement system are listed in Table I. The complete circuit schematic is shown in Fig. 1. The key connections for the ADC and LCD are detailed in Table II and Table III.

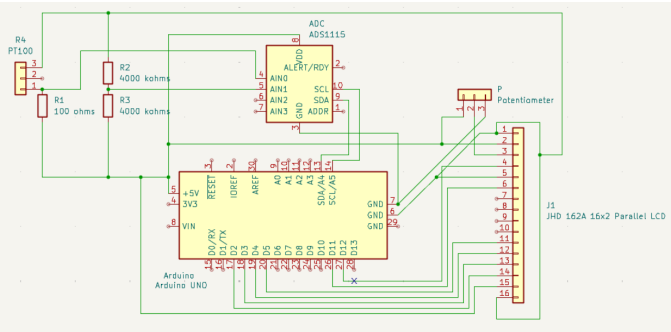

Fig. 1. Circuit Schematic

TABLE I
List of Components

Component	Qty.	Description
Arduino Uno	1	Microcontroller for processing and control.
ADS1115 ADC	1	16-bit external ADC for high-precision voltage measurement.
PT100 RTD Sensor	1	Platinum resistance thermometer for sensing temperature.
100Ω Resistor	1	Precision reference resistor for the voltage divider.
JHD 162A Parallel LCD	1	For displaying the final temperature reading.
10kΩ Potentiometer	1	To adjust the contrast of the LCD.
Breadboard	1	For creating solderless circuit connections.
Jumper Wires	Set	For connecting all the components.

TABLE II
ADS1115 and Arduino Connections

ADS1115 Pin	Arduino Pin	Purpose
VDD	5V	Power Supply
GND	GND	Common Ground
SCL	A5 (SCL)	I2C Clock Line
SDA	A4 (SDA)	I2C Data Line
A0	-	Input from Voltage Divider Node (between PT100 and R_REF)
A1	GND	Reference for differential reading

TABLE III
JHD 162A Parallel LCD and Arduino Connections

LCD Pin	Arduino Pin
VSS	GND
VDD	5V
V0	Potentiometer Middle Pin
RS	Digital 12
RW	GND
E	Digital 11
D4	Digital 5
D5	Digital 4
D6	Digital 3
D7	Digital 2
A (Anode)	5V
K (Cathode)	GND

III. Role of the ADS1115 ADC

The ADS1115 is a high-precision, 16-bit Analog-to-Digital Converter that serves as the core measurement component of

the system. Its primary role is to overcome the significant limitations of the Arduino's built-in 10-bit ADC for this specific application.

- **High Resolution**: A 10-bit ADC has 2^{10} (1,024) steps of resolution. The 16-bit ADS1115 has 2^{16} (65,536) steps. This 64-fold increase in resolution allows it to detect the tiny, microvolt-level changes produced by the PT100 sensor, which would be completely lost in the quantization noise of the Arduino's internal ADC.
- **Differential Measurement**: The circuit is specifically configured to read the voltage difference between the ADC's A0 and A1 channels. This differential reading, performed by the `ads.readADC_Differential_0_1()` function, is crucial for reducing common-mode noise. Electrical noise from the power supply or other environmental sources tends to affect both input lines equally. A differential amplifier rejects this common noise and only measures the true signal, leading to a much cleaner and more stable reading.
- **Programmable Gain**: The ADS1115 contains a Programmable Gain Amplifier (PGA). The code sets this internal amplifier's gain to 8x using `ads.setGain(GAIN_EIGHT)`. This amplifies the small voltage from the circuit before it is digitized, ensuring the signal uses a larger portion of the ADC's full 16-bit range and further improving the effective resolution. For a gain of 8x, the measurable voltage range is ± 0.512V.
- **Voltage Conversion**: The raw digital value from the ADC (`adc_raw`) is converted back into a real-world voltage using a specific multiplier based on the gain. For a gain of 8x, the multiplier is 0.015625 millivolts per bit. The code applies this conversion to get the final voltage (`v_out`) used in the temperature calculation.

IV. Software Implementation

The system's logic is implemented in C++ using the Arduino IDE. The code is responsible for initializing the ADS1115 ADC and the LCD, reading the differential voltage, applying the quadratic model to calculate temperature, and displaying the result. A moving average filter is used to smooth the output. The complete source code can be found at: https://github.com/PuniAditya/EE1030-2025-Project-Submissions/blob/main/ee25btech11046_ee25btech11062/Hardware-Assignment/codes/arduino/code.cpp.

V. Mathematical Modeling and Validation

A. Model Training

To calibrate the sensor, a quadratic model of the form $V = aT^2 + bT + c$ was fitted to an empirical training dataset, which is provided in Table IV. The coefficients were determined using a Python script employing the 'numpy.linalg.lstsq' function.

The derived coefficients are:

- $a = 1.7826 \times 10^{-5}$
- $b = 1.4987 \times 10^{-3}$
- $c = 0.1570$

To convert the measured voltage (V) back to temperature (T), the quadratic formula is used by first rearranging the equation to $aT^2 + bT + (c - V) = 0$. Solving for T gives:

$$T = \frac{-b + \sqrt{b^2 - 4a(c-V)}}{2a}$$

The plot of the training data against the fitted parabolic curve is shown in Fig. 2.

TABLE IV
Training Data: Temperature vs. Voltage

Temp (°C)	Voltage (V)	Temp (°C)	Voltage (V)
26.7	0.20171	72.0	0.35171
36.7	0.23471	79.1	0.37671
43.9	0.26471	85.3	0.39771
47.8	0.28671	87.5	0.43271
56.6	0.29541	90.0	0.44671
59.1	0.30671	91.4	0.45271
60.0	0.30821	94.6	0.45871

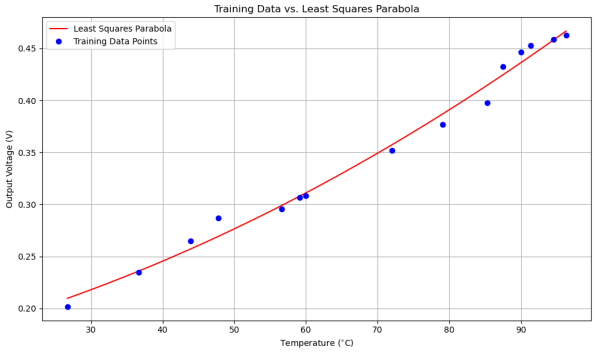

Fig. 2. Training data points plotted against the fitted Least Squares parabola.

B. Model Validation

The accuracy of the trained model was verified using a separate validation dataset (Table V) that was not used during training. The model's predictions for the validation data points were compared against their true temperature values. The results, shown in Table VI, indicate a close agreement between predicted and actual temperatures, confirming the model's validity. The graphical representation of this validation is shown in Fig. 3.

TABLE V
Validation Data: Temperature vs. Voltage

Temp (°C)	Voltage (V)
36.8	0.23571
93.1	0.45391
57.3	0.29921
73.4	0.35621

TABLE VI
VALIDATION RESULTS

Actual Temp (°C)	Voltage (V)	Predicted Temp (°C)
36.8	0.23571	36.64
93.1	0.45391	92.83
57.3	0.29921	57.17
73.4	0.35621	73.20

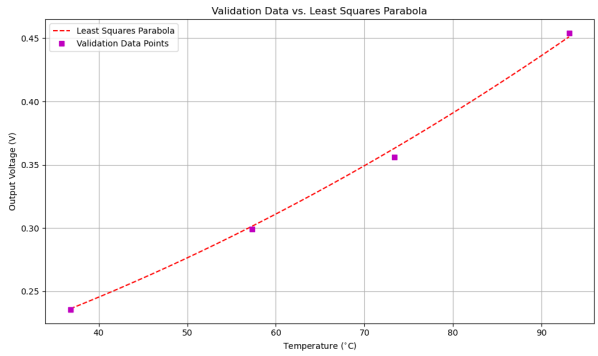

Fig. 3. Validation data points plotted against the model's predictive curve.

VI. CONCLUSION

This project successfully demonstrates the construction and calibration of a high-precision temperature sensor. The use of a quadratic model derived from a Least Squares fit proved to be an effective method for converting raw voltage data into accurate temperature readings. The model's performance was successfully verified against an independent validation dataset, confirming its reliability. While the voltage divider circuit is functional, its accuracy is fundamentally limited by noise and sensor self-heating. Future work should focus on implementing a Wheatstone bridge circuit with a dedicated instrumentation amplifier to further improve noise rejection and overall system accuracy.