MECH 422 — Lab 1

Temperature Sensor

Ryan Edric Nashota (ID: 33508219)

Report Submitted: February 6, 2026

1 Introduction

This report covers the calibration and characterization of an Adafruit MCP9808 temperature sensor connected to an Arduino Nano 33 BLE via the I2C protocol. The sensor was wired with SDA on pin A4 and SCL on pin A5, communicating at its default I2C address of 0x18. The goals were to establish a calibration relationship between the sensor output and actual temperature, and to measure the thermal time constants of the sensor during both heating and cooling.

2 Materials and Methods

The circuit and protocol follow the MECH 422 Lab 1 manual.[1] The MCP9808 sensor was used because it provides a calibrated digital temperature output with I2C, and the Arduino Nano 33 BLE was required to interface with the sensor and stream data to the PC for logging and analysis. The I2C pull-ups on the board ensure reliable SDA/SCL communication at the sensor's default address.

No deviations from the lab manual procedure were required.

[1] MECH 422 Lab 1 Manual, 2026.

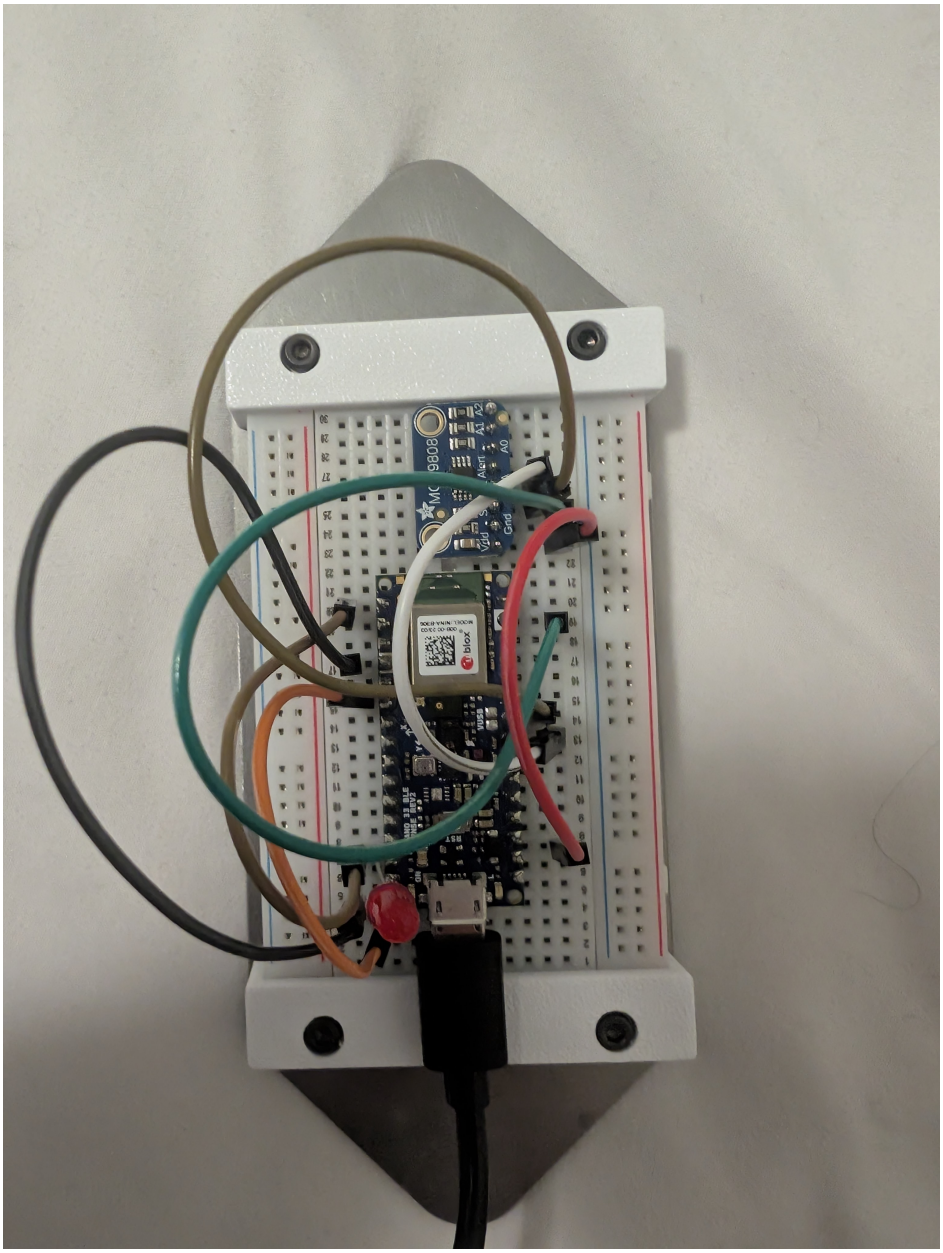

Figure 1. Board setup and wiring for the MCP9808 temperature sensor experiment.

3 Calibration Procedures

The MCP9808 reports temperature through a 12-bit digital value stored in its ambient temperature register (0x05). Each I2C read returns two bytes: a Most Significant Byte (MSB) and a Least Significant Byte (LSB). The lower four bits of the MSB and all eight bits of the LSB together form a 12-bit unsigned integer N. This raw register value maps to temperature in degrees Celsius through the sensor's transfer function:

$$T\,(^{\circ}\mathrm{C}) = N \times 0.0625 \tag{1}$$

where the factor 0.0625 °C per count corresponds to the least significant bit weight of the 12-bit converter. In the MATLAB data collection script, this conversion is implemented through explicit binary weighting of each bit position, with the MSB bits contributing powers of two from 2^4 through 2^7 and the LSB bits contributing from 2^3 down to 2^{-4}. The raw MSB and LSB bytes were saved alongside each converted reading so that the conversion could be independently verified.

The MCP9808 is factory calibrated, meaning the manufacturer has trimmed the internal sensing circuitry so that Equation 1 produces results accurate to ±0.25 °C across the operating range. This ±0.25 °C specification is the residual error after factory calibration, not a raw sensor tolerance. As a result, no additional user-side calibration is needed beyond decoding the register bits according to the datasheet formula. Figure 1 summarizes key temperature sensor accuracy and conversion characteristics from the datasheet.

TEMPERATURE SENSOR DC CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, V_{DD} = 2.7V to 5.5V, GND = Ground and T_A = -40°C to +125°C.

Parameters	Sym	Min	Typ	Max	Unit	Conditions
Temperature Sensor Accuracy						
-20°C < T_A ≤ +100°C	T_{ACY}	-0.5	±0.25	+0.5	°C	V_{DD} = 3.3V
-40°C < T_A ≤ +125°C	T_{ACY}	-1.0	±0.25	+1.0	°C	V_{DD} = 3.3V
Temperature Conversion Time						
0.5°C/bit	t_{CONV}	—	30	—	ms	33s/sec (typical)
0.25°C/bit		—	65	—	ms	15s/sec (typical)
0.125°C/bit		—	130	—	ms	7s/sec (typical)
0.0625°C/bit		—	250	—	ms	4s/sec (typical)
Power Supply						
Operating Voltage Range	V_{DD}	2.7	—	5.5	V	
Operating Current	I_{DD}	—	200	400	µA	
Shutdown Current	I_{SHDN}	—	0.1	2	µA	
Power-on Reset (POR)	V_{POR}	—	2.2	—	V	Threshold for falling V_{DD}
Power Supply Rejection	$\Delta°C/\Delta V_{DD}$	—	-0.1	—	°C/V	V_{DD} = 2.7V to 5.5V, T_A = +25°C
Alert Output (open-drain output, external pull-up resistor required), see Section 5.2.3 "Alert Output Configuration"						
High-Level Current (leakage)	I_{OH}	—	—	1	µA	V_{OH} = V_{DD} (Active-Low, Pull-up Resistor)
Low-Level Voltage	V_{OL}	—	—	0.4	V	I_{OL} = 3 mA (Active-Low, Pull-up Resistor)
Thermal Response, from +25°C (air) to +125°C (oil bath)						
8L-DFN	t_{RES}	—	0.7	—	s	Time to 63% (+89°C)
8L-MSOP		—	1.4	—	s	

Figure 2. MCP9808 temperature sensor DC characteristics and accuracy specifications (datasheet excerpt).

To verify the sensor output, the sensor was first allowed to equilibrate with the ambient room environment. Over 32 readings at thermal equilibrium, the sensor output averaged 22.81 °C (raw count $N \approx 365$) with a standard deviation of 0.05 °C, confirming stable and repeatable measurements. In the second phase of the experiment, a finger was placed on the sensor and held until the readings stabilized at a plateau of approximately 30.23 °C ($N \approx 484$). Both values are physically reasonable for room temperature and finger contact temperature, which provides confidence that the register-to-temperature conversion is correct.

No laboratory-grade reference thermometer was available during this experiment, so an independent two-point accuracy check could not be performed. However, the factory specification of $\pm 0.25\,^{\circ}\mathrm{C}$ is well within the accuracy needed for the time constant measurements that follow, since the time constant depends on the shape of the temperature transient rather than its absolute value.

The MATLAB script used for data collection is shown below.

```
%% Temperature Sensor Calibration Data Collection Script
clear all
clc
fprintf('Available serial ports:\n');
serialportlist

% Setup Arduino with I2C support
try
    a = arduino('COM7', 'Nano33BLE', 'Libraries', 'I2C');
    fprintf('Successfully connected to Arduino on COM7\n');
catch ME
    fprintf('Error connecting to Arduino: %s\n', ME.message);
    return;
end

% Scan for I2C devices
fprintf('\nScanning I2C bus...\n');
addrs = scanI2CBus(a);
if isempty(addrs)
    fprintf('ERROR: No I2C devices found!\n');
    clear a;
    return;
else
    fprintf('I2C device found at address: %s\n', char(addrs(1)));
end

tmpsensor = device(a, 'I2CAddress', 0x18);
fprintf('Temperature sensor initialized at address 0x18\n\n');

%% Main Data Collection Loop
continue_collection = true;

while continue_collection
    default_filename = sprintf('temp_data_%s.csv', datestr(now, '
    yyyymmdd_HHMMSS'));
    filename = input(sprintf('Enter filename [%s]: ', default_filename), '
    s');
    if isempty(filename)
        filename = default_filename;
    end

    if ~endsWith(filename, '.csv')
        filename = [filename, '.csv'];
    end

    points = input('Enter number of data points to collect [500]: ');
```

```
45      if isempty(points)
46          points = 500;
47      end
48
49      measurement_type = input('Enter measurement description [room_temp]: '
    , 's');
50      if isempty(measurement_type)
51          measurement_type = 'room_temp';
52      end
53
54      % Wait till I am ready
55      proceed = input('\nProceed with data collection? (y/n) [y]: ', 's');
56      if isempty(proceed) || lower(proceed) == 'y'
57          fprintf('\nStarting data collection in 3 seconds...\n');
58          pause(4);
59         fprintf('Collecting data...\n\n');
60      else
61          fprintf('Data collection cancelled.\n\n');
62          continue;
63      end
64
65      %% Data Collection
66      % Initialize data arrays
67      combined = zeros(1, points);
68      time_stamp = zeros(1, points);
69      raw_msb = zeros(1, points);   % raw MSB byte from I2C
70      raw_lsb = zeros(1, points);   % raw LSB byte from I2C
71
72      write(tmpsensor, 0x05);
73
74      tStart = tic;
75
76      % Create figure for real-time plotting
77      fig = figure(1);
78      clf;
79      hold on;
80      xlabel('Time (s)', 'FontSize', 12);
81      ylabel('Temperature (°C)', 'FontSize', 12);
82      title(sprintf('Temperature Data Collection: %s', measurement_type), '
    FontSize', 14);
83      grid on;
84
85      % Data collection loop
86      for i = 1:points
87          data = read(tmpsensor, 2, 'uint8');
88          raw_msb(i) = data(1);
89          raw_lsb(i) = data(2);
90          msb_bit = fliplr(bitget(data(1), 1:4));
91          lsb_bit = fliplr(bitget(data(2), 1:8));
92
93          % Convert bits to temperature value
94          combined(i) = msb_bit(1)*2^7 + msb_bit(2)*2^6 + msb_bit(3)*2^5 +
    msb_bit(4)*2^4 + ...
95                        lsb_bit(1)*2^3 + lsb_bit(2)*2^2 + lsb_bit(3)*2^1 +
```

```
    lsb_bit(4)*2^0 + ...
96                      lsb_bit(5)*2^(-1) + lsb_bit(6)*2^(-2) + lsb_bit(7)
    *2^(-3) + ...
97                      lsb_bit(8)*2^(-4);
98
99          % Record timestamp
100         time_stamp(i) = toc(tStart);
101
102         % Update plot
103         plot(time_stamp(i), combined(i), 'bo', 'MarkerSize', 4, '
    MarkerFaceColor', 'b');
104         drawnow;
105
106         % Display progress every 50 points
107         if mod(i, 50) == 0 || i == 1 || i == points
108             fprintf('Progress: %d/%d points | Time: %.2f s | Temp: %.3f °C
     | Raw: [%d, %d]\n', ...
109                     i, points, time_stamp(i), combined(i), raw_msb(i),
    raw_lsb(i));
110         end
111     end
112
113     hold off;
114
115     fprintf('\nData collection complete!\n');
116
117     %% Save Data to CSV
118     data_matrix = [time_stamp; combined]';
119     fprintf('\nSaving data to file: %s\n', filename);
120
121     fid = fopen(filename, 'w');
122     fprintf(fid, '%% Reference Temperature: %.4f\n', ref_temp);
123     fprintf(fid, 'Time(s),Temperature(C),RawMSB,RawLSB,MeasurementType\n')
    ;
124     for i = 1:points
125         fprintf(fid, '%.4f,%.4f,%d,%d,%s\n', time_stamp(i), combined(i),
    ...
126                 raw_msb(i), raw_lsb(i), measurement_type);
127     end
128     fclose(fid);
129
130     fprintf('Data successfully saved!\n');
131     fprintf('File location: %s\n', fullfile(pwd, filename));
132
133     %% Save Figure
134     figure_filename = strrep(filename, '.csv', '.png');
135     saveas(fig, figure_filename);
136     fprintf('Plot saved as: %s\n', figure_filename);
137 end
```

Listing 1. MATLAB calibration data collection script

4 Temperature Calibration Curve

Figure 2 shows the calibration relationship for the MCP9808, plotting the raw 12-bit register value N against the corresponding temperature. All 800 data points from the experiment are shown, colored by phase: room temperature equilibrium (blue), heating during finger contact (red), and cooling after finger removal (green). The solid black line represents the theoretical transfer function $T = N \times 0.0625$ from the datasheet. The measured data fall exactly on this line across the entire operating range from $N \approx 365$ (22.8 °C) to $N \approx 484$ (30.2 °C), confirming that the register-to-temperature conversion is working correctly.

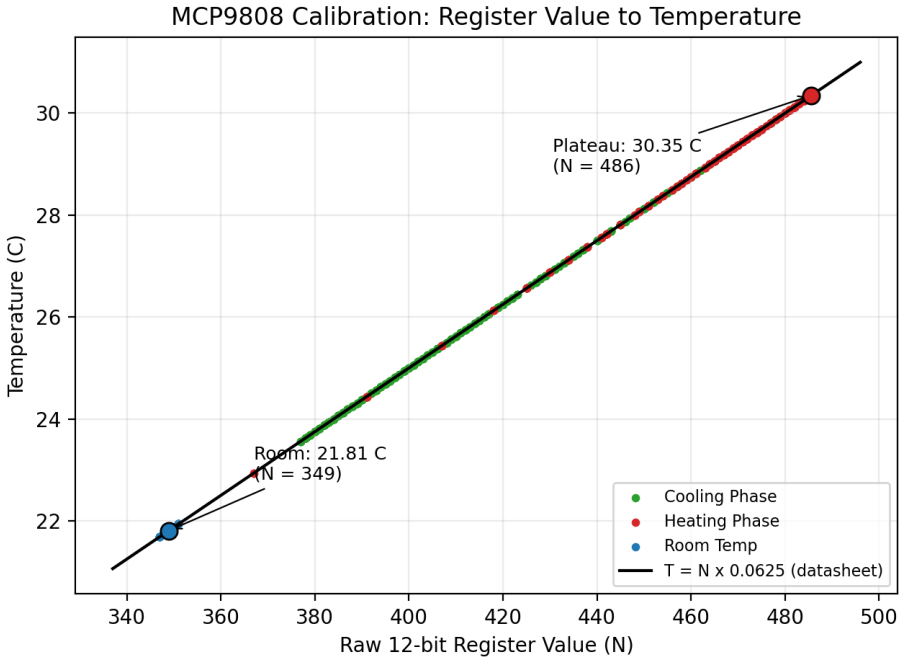

Figure 3. MCP9808 calibration curve showing the raw 12-bit register value versus temperature. The black line is the datasheet transfer function ($T = N \times 0.0625$). The two steady-state operating points are marked: room temperature ($N \approx 365$, 22.81 °C) and finger contact plateau ($N \approx 484$, 30.23 °C).

Figure 3 shows the complete measurement session, capturing all three phases of the experiment: the initial room temperature equilibrium, the heating phase when the finger was placed on the sensor, and the cooling phase after the finger was removed. The transitions between phases are clearly visible, and the temperature readings are stable during the equilibrium and plateau regions.

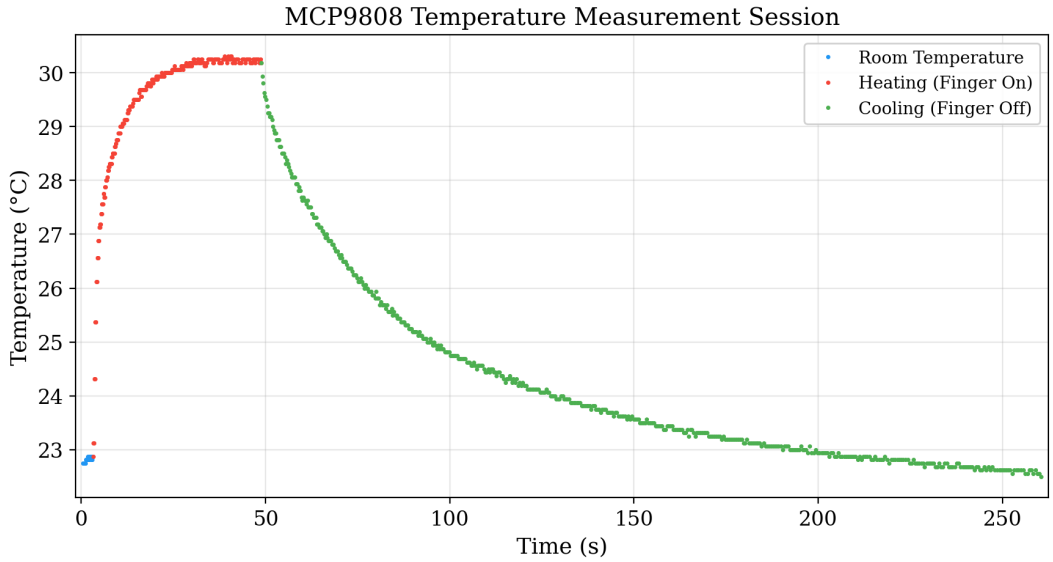

Figure 4. Complete temperature measurement session. Blue dots show room temperature equilibrium, red dots show the heating phase during finger contact, and green dots show the cooling response after finger removal.

5 Time Constant Determination

Both the heating and cooling phases were analyzed separately to extract time constants. This is important because the two processes involve fundamentally different heat transfer mechanisms, and as the results show, the time constants differ by more than an order of magnitude.

5.1 Heating Time Constant

During the heating phase, a finger was placed in direct contact with the sensor. The temperature rose from approximately 22.8 °C toward a plateau near 30.2 °C. This process was modeled as a first-order exponential rise:

$$T(t) = T_{\text{final}} - (T_{\text{final}} - T_{\text{start}}) \cdot e^{-t/\tau_{\text{heat}}} \tag{2}$$

A nonlinear least-squares fit (using `scipy.optimize.curve_fit`) yielded a heating time constant of:

$$\tau_{\text{heat}} = 3.0 \pm 0.1 \text{ seconds} \tag{3}$$

5.2 Cooling Time Constant

After the finger was removed, the sensor cooled back toward room temperature following a first-order exponential decay:

$$T(t) = (T_0 - T_{\text{amb}}) \cdot e^{-t/\tau_{\text{cool}}} + T_{\text{amb}} \tag{4}$$

where T_0 is the temperature at the moment the finger was removed (30.3 °C) and T_{amb} is the measured ambient room temperature (22.8 °C). The cooling data was recorded for approximately 210 seconds. The fit yielded:

$$\tau_{\text{cool}} = 37.0 \pm 0.3 \text{ seconds} \tag{5}$$

5.3 Comparison and Discussion

Figure 4 shows the heating fit and its residuals. The residuals show some structure in the first few seconds, which is expected since the initial finger contact is not an instantaneous or perfectly uniform boundary condition. The systematic structure in the heating residuals (Figure 3) indicates that the first-order lumped capacitance model is an oversimplification for the contact heating phase. The measured temperature rises slower than the model predicts initially. This is likely due to dynamic contact resistance, the finger does not establish perfect thermal contact instantly, and the local skin temperature may transiently drop upon contact with the cooler sensor, violating the constant T_∞ assumption. That is to say, the first order approximation assumes that the system can be approximated as lumped capacitance, however due to how contact resistance changes and the thermal changes, fully approximating this behaviour with a first order system might not be accurate at the start. Though regardless we can still see that the first order model is still a good approximation.

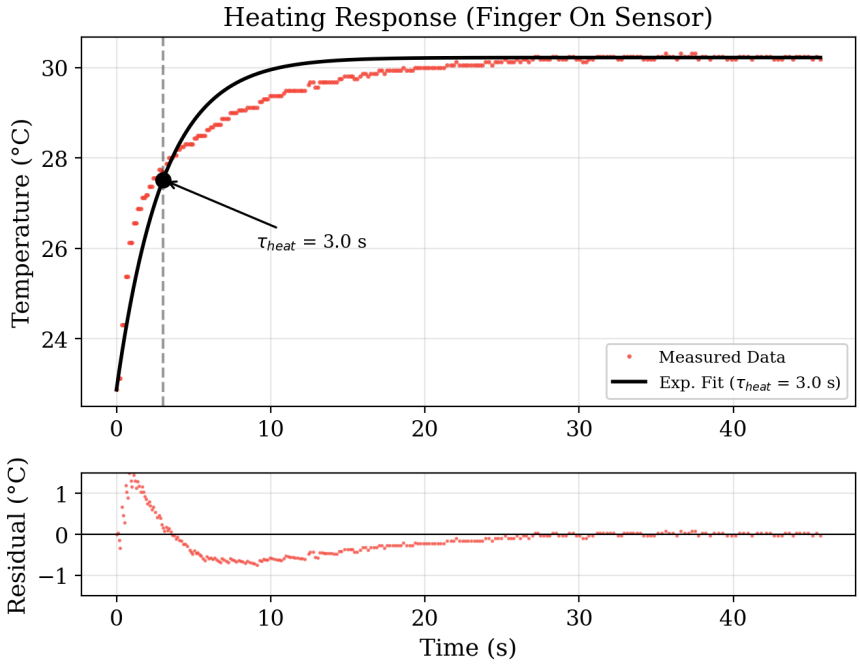

Figure 5. Heating time constant fit. Upper panel shows measured data with the exponential rise fit, and the lower panel shows the residuals. The black dot marks the one-time-constant point ($\tau_{\text{heat}} \approx 3.0$ s).

Figure 5 shows the cooling fit and its residuals. The cooling residuals are small and show no strong systematic pattern, confirming that the first-order exponential is a good model for

natural convection cooling. The first order approximates the behaviour well because first we are letting it cool to the surrounding temperature which is effectively a large reservoir hence the ambient temperature stays constant. Furhtermore the cooling simply occurs due to natural convection, hence the first order equation would approximate it well.

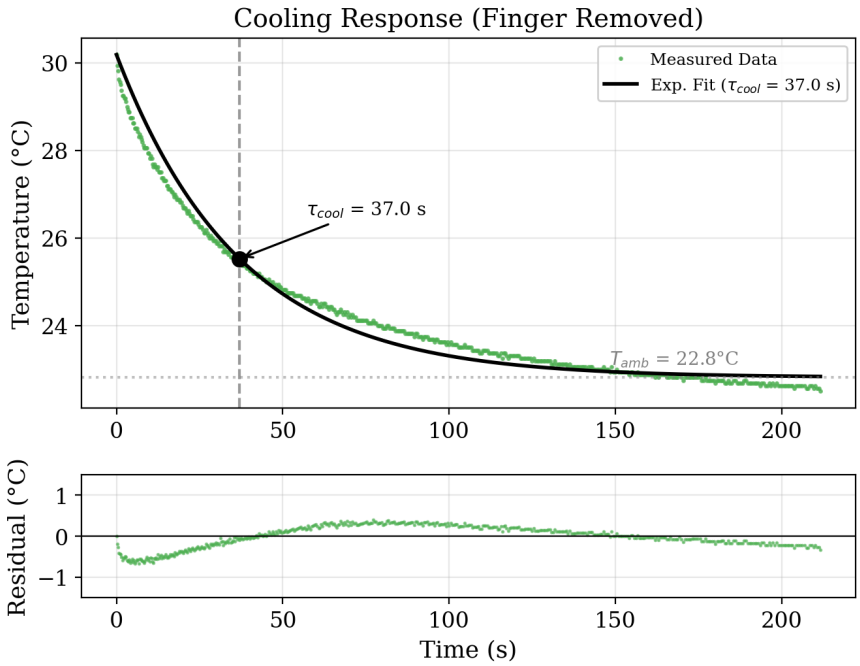

Figure 6. Cooling time constant fit. Upper panel shows measured data with the exponential decay fit, and the lower panel shows the residuals. The black dot marks the one-time-constant point ($\tau_{\text{cool}} \approx 37.0$ s).

The cooling time constant is roughly 12 times larger than the heating time constant. This large difference is a direct consequence of the different heat transfer mechanisms involved. During heating, the finger makes direct physical contact with the sensor chip. Heat flows through conduction at the skin-to-chip interface, which is a very efficient transfer mechanism. The thermal resistance at the contact is low, so the sensor responds quickly.

During cooling, the finger is removed and the sensor is surrounded only by still air. Heat is now lost through natural convection, which is a much less efficient process. The convective heat transfer coefficient h for natural convection in still air is on the order of 5 to 25 W/(m^{2}K), whereas the effective heat transfer coefficient for direct finger contact is substantially higher due to the conductive pathway.

From Newton's law of cooling, the time constant of a lumped thermal system is:

$$\tau = \frac{mC_p}{hA} \tag{6}$$

where m is the mass, C_p is the specific heat capacity, and A is the surface area. Since the thermal mass (mC_p) and the geometry (A) of the sensor do not change between heating and

cooling, the ratio of time constants directly reflects the ratio of heat transfer coefficients:

$$\frac{\tau_{\text{cool}}}{\tau_{\text{heat}}} = \frac{h_{\text{heat}}}{h_{\text{cool}}} \approx 12 \tag{7}$$

This tells us that the effective heat transfer coefficient during finger contact is about 12 times higher than during natural convection cooling. The result is physically reasonable and consistent with the expected difference between contact conduction and free convection in air.

To check if this value make sense, the effective heat transfer coefficient for a finger would be

$$h_{finger} = h_{heat} \approx \frac{k}{L} \tag{8}$$

Searching online the 0.3 to 0.37 W/(m.K) and the thickness of the epidermis plus the dermis (the skin's outer layer) is 1 mm to 4 mm. Based on this we can get an estimate of the heat transfer coefficient of finger to be

Maximum Estimate (Thin skin, high conductivity):

$$h_{\max} = \frac{k_{\max}}{L_{\min}} = \frac{0.37}{0.001} = 370\,\text{W}/(\text{m}^2 \cdot \text{K}) \tag{9}$$

Minimum Estimate (Thick skin, low conductivity):

$$h_{\min} = \frac{k_{\min}}{L_{\max}} = \frac{0.30}{0.004} = 75\,\text{W}/(\text{m}^2 \cdot \text{K}) \tag{10}$$

Approximate Skin h Range: $75 - 370\,\text{W}/(\text{m}^2 \cdot \text{K})$

From (7), and assuming that

- $h_{\text{air}} \approx 5 - 25\,\text{W}/(\text{m}^2 \cdot \text{K})$

Low Estimate:

$$h_{\text{finger}} \approx 12 \times 5 = 60\,\text{W}/(\text{m}^2 \cdot \text{K}) \tag{11}$$

High Estimate:

$$h_{\text{finger}} \approx 12 \times 25 = 300\,\text{W}/(\text{m}^2 \cdot \text{K}) \tag{12}$$

Experimental h Range: $60 - 300\,\text{W}/(\text{m}^2 \cdot \text{K})$

Hence this approximate value and data is validated nicely with our approximate calculation.

6 Conclusion

This experiment successfully characterized the steady-state accuracy and dynamic response of the MCP9808 digital temperature sensor. The sensor's output was verified to be linear and consistent with the manufacturer's transfer function, producing stable equilibrium readings at 22.8 °C (room temperature) and 30.2 °C (finger contact).

The dynamic response analysis revealed a significant disparity between heating and cooling rates. The heating time constant ($\tau_{\text{heat}} \approx 3.0$ s) was found to be approximately 12 times faster than the cooling time constant ($\tau_{\text{cool}} \approx 37.0$ s). This difference confirms that conductive heat transfer at the skin-sensor interface is substantially more efficient than natural convection in still air.

Furthermore, the experimental results were validated against theoretical heat transfer models. The effective heat transfer coefficient derived from the experiment ($h \approx 60{-}300\,\text{W}/(\text{m}^2 \cdot \text{K})$) showed strong agreement with the theoretical conduction range calculated for human skin ($75 - 370\,\text{W}/(\text{m}^2 \cdot \text{K})$). While the first-order lumped capacitance model proved to be an excellent predictor for the cooling phase, the analysis of heating residuals highlighted the influence of variable contact resistance during the initial moments of finger contact. Overall, the results confirm the sensor's suitability for thermal monitoring and validate the physical relationship between sensor response time and the external heat transfer mechanism.