I2C Temperature Sensor (MCP9808) How-To Guide for BeagleY-AI

By Team BeagleStorm
CMPT 433 D100
Spring 2025

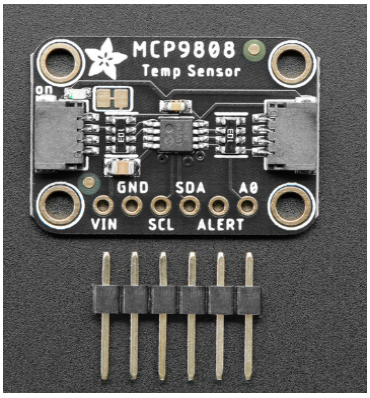
Table of Contents

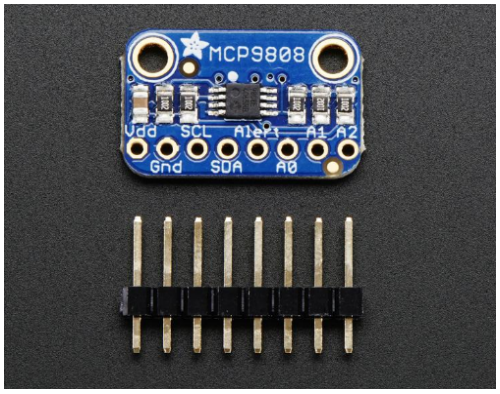
Introduction

This document's purpose is to guide users through connecting the MCP9808 I2C Temperature Sensor to the BeagleY-AI and reading temperature values from it. It contains an introduction to the temperature sensor and its pinout, instructions on how to wire the MCP9808 to the BeagleY-AI using the PYMNL header and breadboard, sample code and outputs for reading the temperature values, and a troubleshooting section.

Description

MCP9808 High Accuracy I2C Temperature Sensor Breakout Board Product Page: https://www.adafruit.com/product/1782

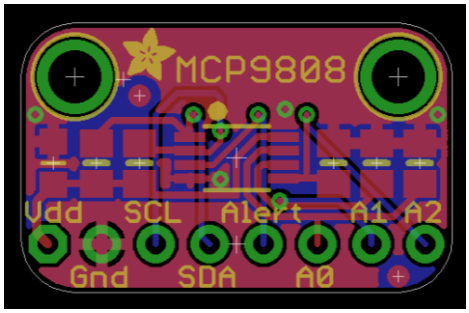




Note: There are two versions of the MCP9808. The STEMMA QT version (black) and the original header-only version (blue). The code should work the same on both versions, but we will be using the header-only version (blue) in this guide.

The MCP9808 is a digital temperature sensor that can be read through standard I2C (Inter-Integrated Circuit). The temperature sensor covers a range of -40°C to +125°C and has a typical accuracy of ±0.25°C, with a precision of +0.0625°C. The MCP9808 works well with any microcontroller that has access to standard I2C control. There are three address pins, allowing you to connect up to eight temperature sensors to a single I2C bus without any address collisions. Additionally, it has a wide voltage range, allowing it to be used with 2.7V to 5.5V logic.

Pinouts



- **Vdd**: This is the positive power pin, which can take between 2.7 and 5.5V
- **GND**: This is the ground power pin
- **SCL**: This is the I2C clock pin
- **SDA**: This is the I2C data pin
- **Alert**: This is the interrupt/alert pin, which has the ability to ‘alert’ you if the temperature goes above or below a set amount (We will not be using this)
- **A0, A1, A2**: These pins allow you to manipulate the address of the MCP9808 temperature sensor
    - **The default address of the MCP9808 temperature sensor is 0x18**

- If A0, A1, A2 are connected, the address can be calculated by adding the A0/A1/A2 to the base of 0x18
- For example, if A0 is connected to the VDD, the address becomes 0x18 + 1 = 0x19
- We will not be using these in this guide

Wiring Guide

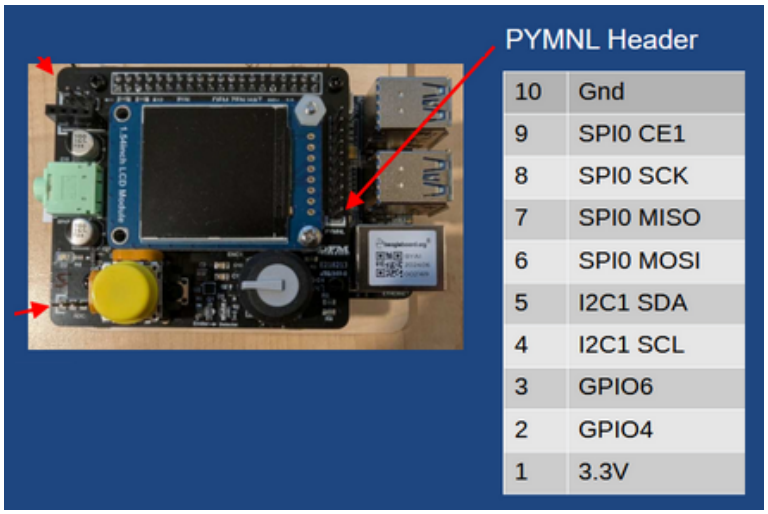
<u>* IMPORTANT: MAKE SURE TO POWER OFF YOUR BEAGLEY-AI WHEN PERFORMING WIRING.</u>

Required Supplies:

- BeagleY-AI with Zen Hat and Breadboard
- MCP9808 Temperature sensor (borrowed from professor)
- 4x male-female wires

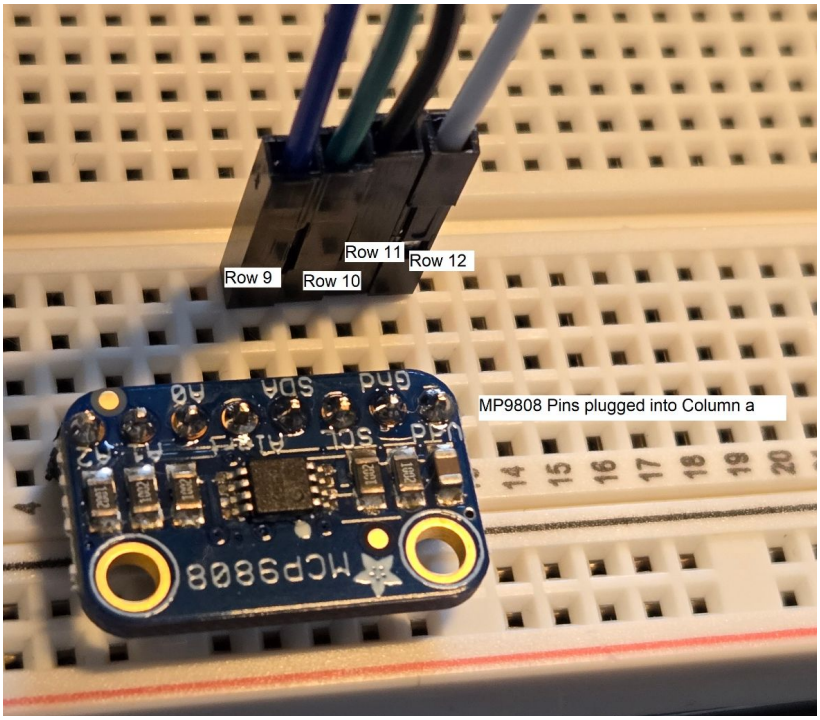
We will be using the PYMNL header on the Zen Hat

- Ensure you orient your board as shown below.
- The PYMNL header pins we are interested in is PYMNL 1 (3.3V), PYMNL 4 (I2C1 SCL), PYMNL 5 (I2C1 SDA), and PYMNL 10 (Gnd).



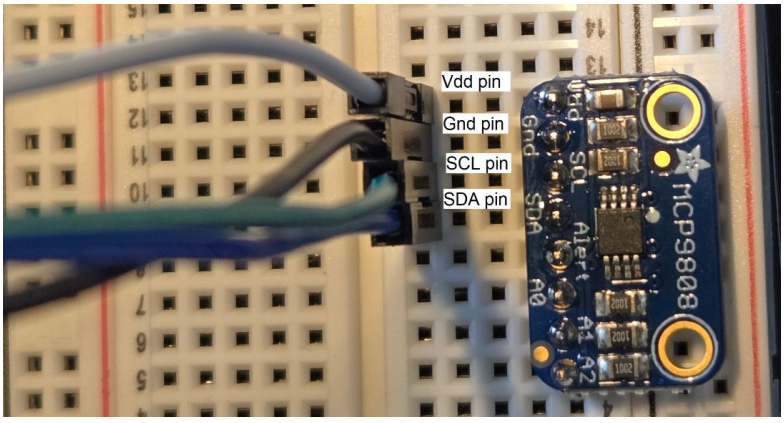
Step 1: Place MCP9808 into Breadboard

- Place the MCP9808 temperature sensor into the breadboard along any **column and rows** you'd like.
- In the image below, I placed them in **column a**, **rows 12, 11, 10, 9, 8, 7, 6, 5** (although only rows 12, 11, 10, and 9 will be used by the wires).
- Ensure that you place the MCP9808 pins along one **column** and not one **row**, otherwise it will not work.



Step 2: Connecting male end of wires

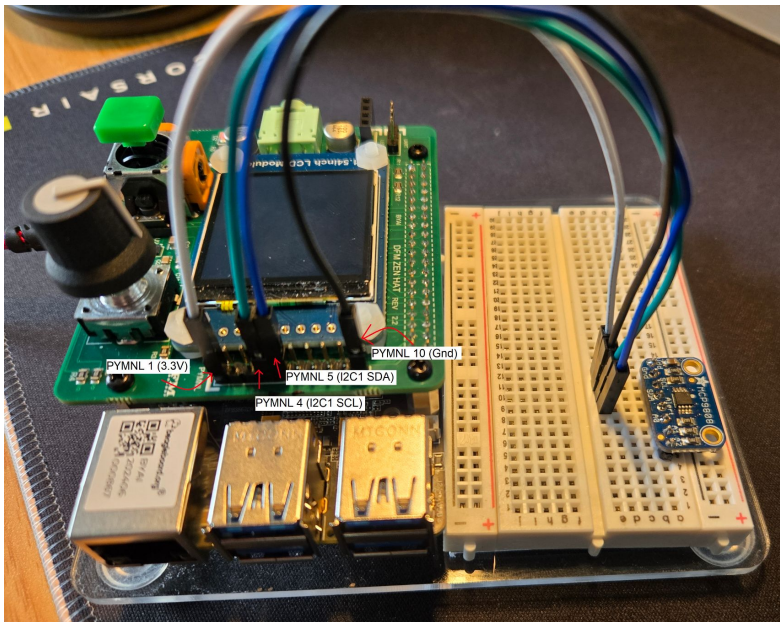
- Take your four male-female wires and place the male ends into rows 12, 11, 10, and 9, where the rows correspond to the MCP9808 temperature sensor as follows:
- Row 12 = Vdd pin
- Row 11 = Gnd pin
- Row 10 = SCL pin
- Row 9 = SDA pin
- Ensure that your pins line up with the rows exactly, as shown below.



Step 3: Connecting female end of wires

- Connect the female end of the wires into the PYMNL header on the Zen Hat

- The wire placed in the same row as the Vdd pin should be placed onto PYMNL 1 (3.3V)
- The wire placed in the same row as the Gnd pin should be placed onto PYMNL 10 (Gnd)
- The wire placed in th same row as the SCL pin should be placed onto PYMNL 4 (I2C1 SCL)
- The wire placed in th same row as the SDA pin should be placed onto PYMNL 5 (I2C1 SDA)
- See the image above for the PYMNL Header and how to orient your board, so that you do not place the wires into the wrong pins.



Step 4: Test the connection

Power on your BeagleY-AI, then in your virtual environment, SSH into the BeagleY-AI target and run the following command:

- (byai) $ i2cdetect -y -r 1
- If everything is connected properly, you should see something like this, where '**18**' appears on the **0x18** address:

```
root@raspberrypi:~# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- 18 -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
...etc...
```

-

Sample Code

Provided support files:

- **main.c**: main program driver that initializes HAL modules and prints temperatures
- **read_temp_sensor.h/.c**: manages the MCP9808 sensor, threading, and temperature conversion

- **i2c_control.h/.c**: provides I2C read/write helper functions

In the provided read_temp_sensor.c file, there is a handful of functions used to interact with the MCP9808 sensor through I2C.

```
void init_temp_sensor() {
    assert(!is_initialized);
    is_initialized = true;

    // Initialize I2C bus and MCP9808, including configurations
    i2c_file_desc = init_i2c_bus(I2C_BUS, I2C_ADDRESS);
    multiwrite_i2c_reg8(i2c_file_desc, CONFIG_REG,
        CONFIG_REG_VAL, CONFIG_REG_VAL);
    write_i2c_reg8(i2c_file_desc, RESOLUTION_REG,
        RESOLUTION_REG_VAL);
    sleep(1);

    // Start temperature reading thread
    if (pthread_create(&temp_thread, NULL, temp_thread_func, NULL) != 0) {
        printf("Failed to create temperature thread.\n");
        is_initialized = false;
    }
}
```

- **init_temp_sensor()** opens the I2C bus "/dev/i2c-1" and connects to the MCP9808 at address 0x18, assuming you are using the default address.
- The MCP9808 temperature sensor expects specific 8-bit data, so we will use functions in the i2c_control.c file that can read and write multiple bytes at once: **multiwrite_i2c_reg8()** and **multiread_i2c_reg8()**.
- We write default values to the configuration and resolution registers, then begin a temperature reading read.

```
void* temp_thread_func(void* arg) {
    (void) arg;

    while (running == 1) {
        // Read temperature data from MCP9808
        uint8_t* data = multiread_i2c_reg8(i2c_file_desc, TEMP_REGISTER);

        // Convert the raw data to temperature (Celsius / Fahrenheit)
        float temp_c, temp_f;
        convert_to_temp(data, &temp_c, &temp_f);

        // Update the current temperature values
        pthread_mutex_lock(&mutex);
        current_temp_c = temp_c;
        current_temp_f = temp_f;
        pthread_mutex_unlock(&mutex);

        sleep(SLEEP_TIME);
    }
```

- The **convert_to_temp()** function converts the raw data into Celsius and Fahrenheit, using the following formula:

    - temp = ((data[0] & 0x1F) * 256 + data[1]);
    - if (temp > 4095) temp -= 8192;
    - celsius = temp * 0.0625;
    - fahrenheit = celsius * 1.8 + 32;
- The main.c will continuously print the temperature values to the console once every second. It also handles the initialization and clean up of the MCP9808 temperature sensor module.

Sample Output:

```
Current Temperature: 25.50°C / 77.90°F
Current Temperature: 25.44°C / 77.79°F
Current Temperature: 25.44°C / 77.79°F
Current Temperature: 25.44°C / 77.79°F
Current Temperature: 25.44°C / 77.79°F
Current Temperature: 25.44°C / 77.79°F
Current Temperature: 25.62°C / 78.12°F
```

- Note: you can place your finger on the sensor to increase the temperatures.

Troubleshooting

- When running “i2cdetect -y -r 1”, ‘18’ does not appear at address 0x18
    - Ensure your wiring is connected as shown above, and that none of the wires are loose
    - Ensure the wiring lines up with the MCP9808 pinout on the breadboard correctly, as it can be misleading
    - Ensure female ends of wires are placed into correct PYMNL header pins
- When powering on the BeagleY-AI, the MCP9808 begins smoking
    - Immediately turn off your BeagleY-AI
    - Ensure you have your wires connected to the PYMNL header in the correct order, if you have them mixed up it can cause the sensor to smoke
- The provided code is not working
    - Ensure you downloaded the provided support files, including read_temp_sensor.h/.c, i2c_control.h/.c, main.c
    - You will need to cross-compile the executable for your desired architecture (ex. Arm64 for debian virtual environment)

References

- MCP9808 Product Page from: https://www.adafruit.com/product/1782
- PYMNL Header slide from “Electronics” lecture, CMPT433 D100 Spring 2025 Dr. Brian
- MCP9808 Reference Guide and images from: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-mcp9808-precision-i2c-temperature-sensor-guide.pdf