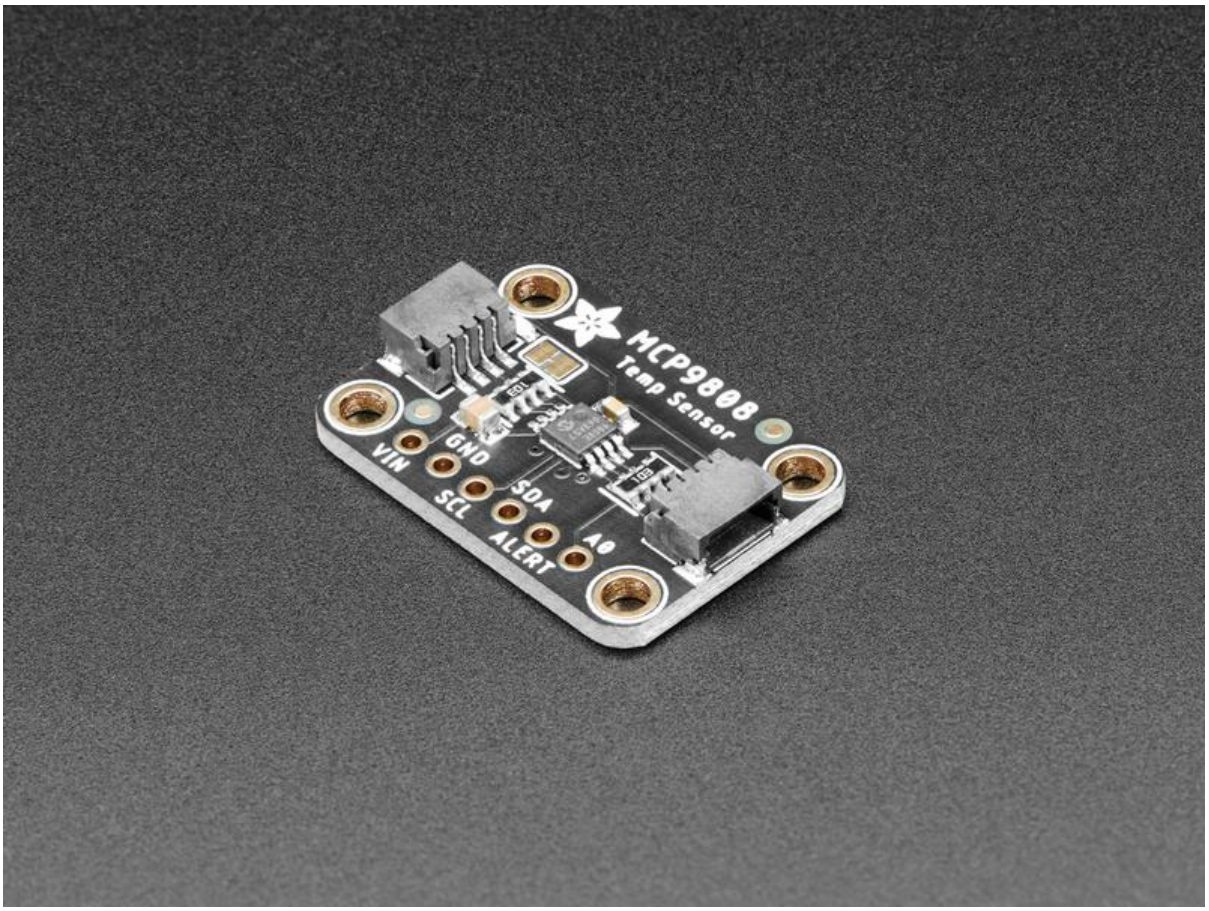
Adafruit MCP9808 Precision I2C Temperature Sensor Guide

Created by lady ada

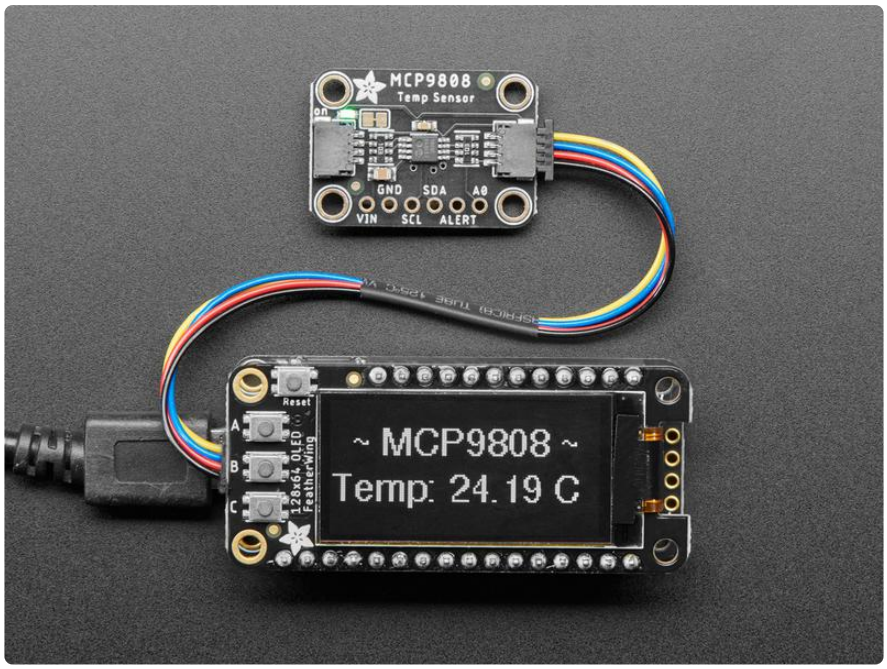


https://learn.adafruit.com/adafruit-mcp9808-precision-i2c-temperature-sensor-guide

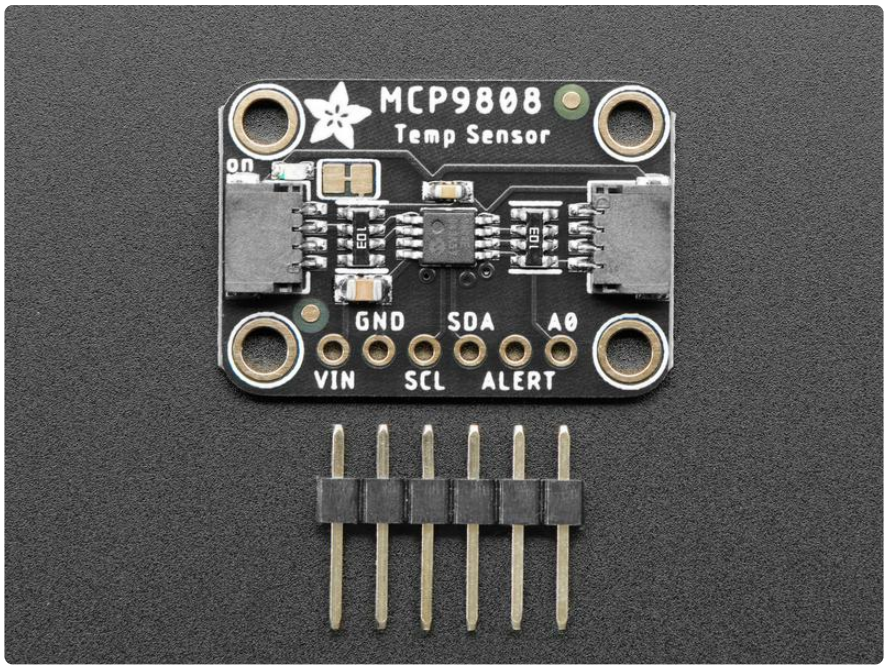
Last updated on 2021-11-15 06:11:40 PM EST

Table of Contents

Overview

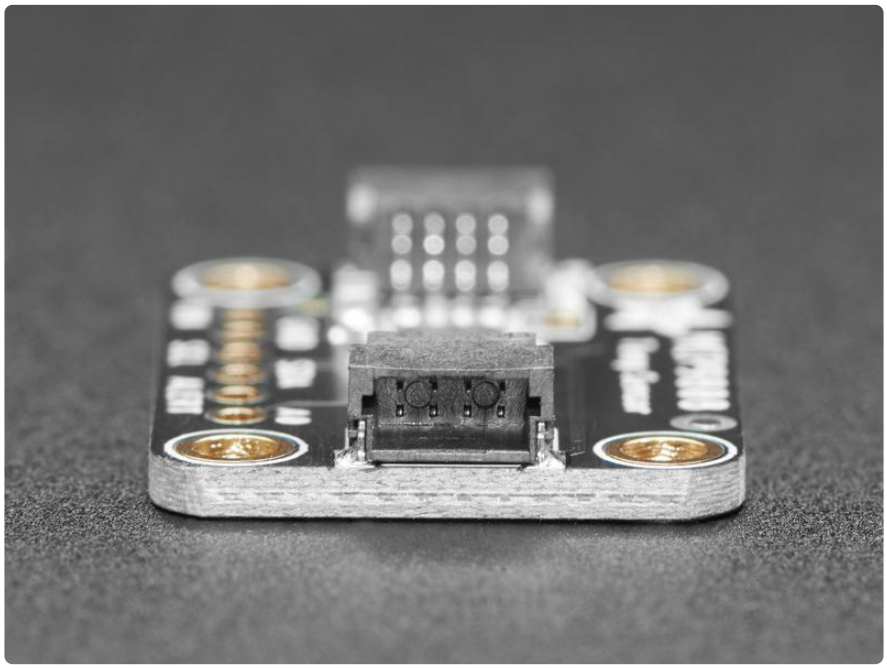


This I2C digital temperature sensor is one of the more accurate/precise we've ever seen, with a typical accuracy of ±0.25°C over the sensor's -40°C to +125°C range and precision of +0.0625°C. They work great with any microcontroller using standard i2c. There are 3 address pins so you can connect up to 8 to a single I2C bus without address collisions. Best of all, a wide voltage range makes is usable with 2.7V to 5.5V logic!

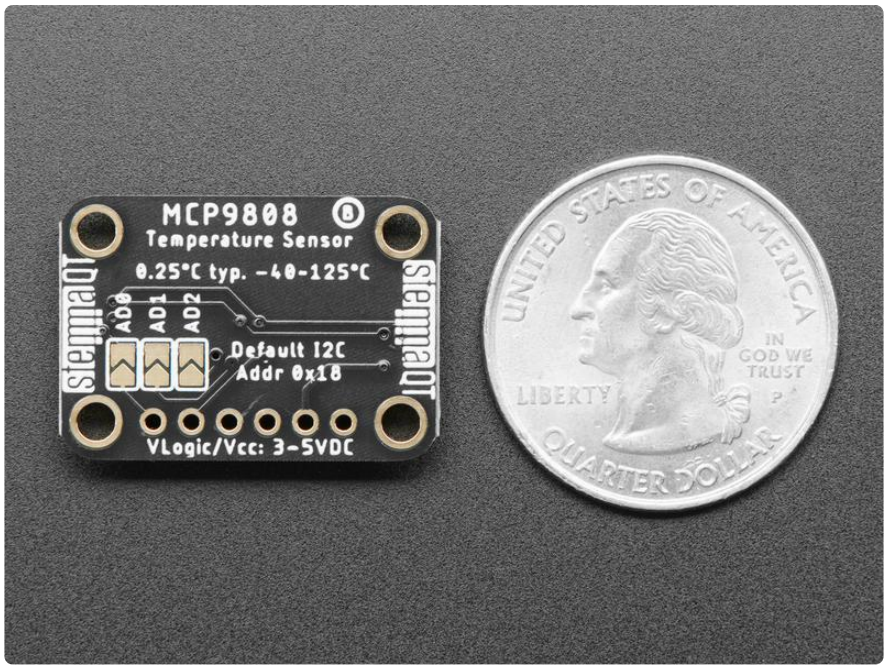


Unlike the DS18B20, this sensor does not come in through-hole package so we placed this small sensor on a breakout board PCB for easy use. The PCB includes

mounting holes, and pull down resistors for the 3 address pins. We even wrote a lovely little library for Arduino that will work with any Arduino compatible. You'll be up and running in 15 minutes or less.

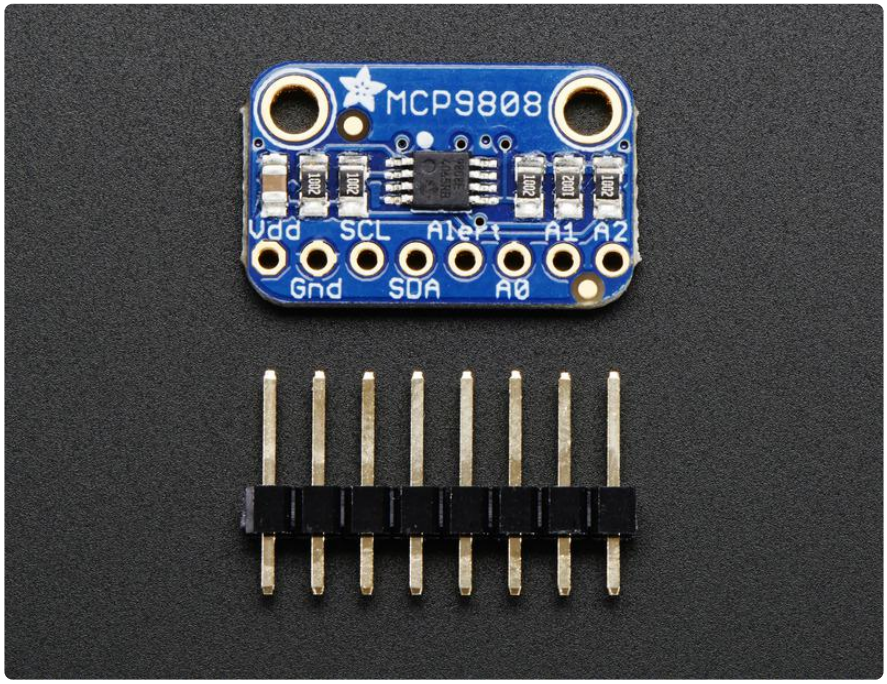


To get you going fast, we spun up a custom made PCB with the MCP9808 and some supporting circuitry such as pull-up resistors and capacitors, in the STEMMA QT form factor (https://adafru.it/LBQ), making them easy to interface with. The STEMMA QT connectors (https://adafru.it/JqB) on either side are compatible with the SparkFun Qwiic (https://adafru.it/Fpw) I2C connectors. This allows you to make solderless connections between your development board and the MCP9808 or to chain them with a wide range of other sensors and accessories using a compatible cable (https://adafru.it/JnB). QT Cable is not included, but we have a variety in the shop (https://adafru.it/JnB).

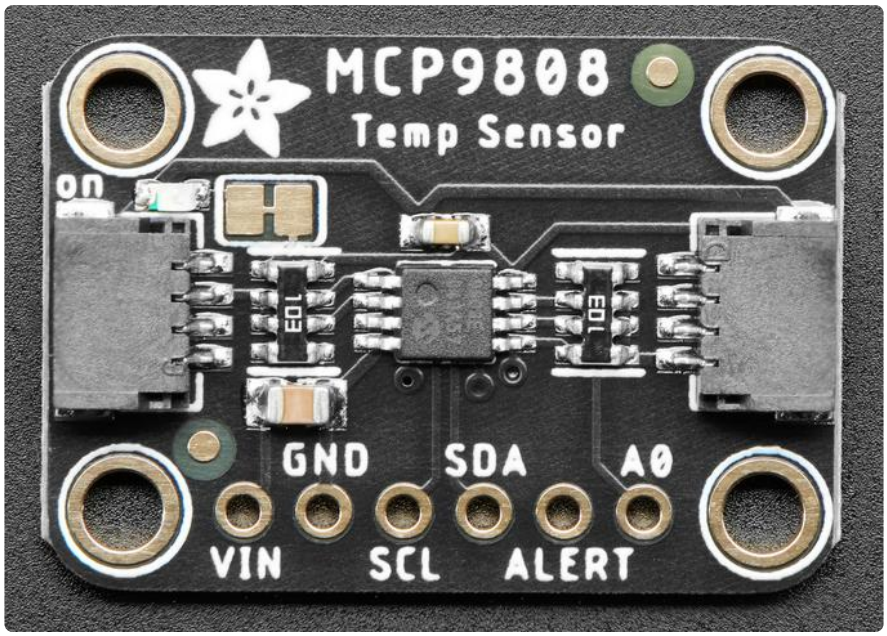


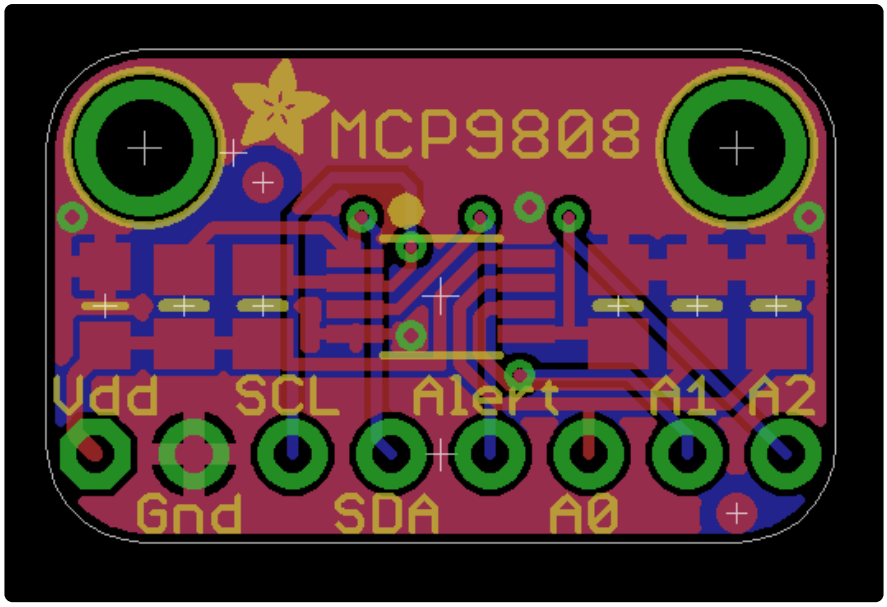
- Simple I2C control
- Up to 8 on a single I2C bus with adjustable address pins
- 0.25°C typical precision over -40°C to 125°C range (0.5°C guaranteed max from -20°C to 100°C)
- 0.0625°C resolution
- 2.7V to 5.5V power and logic voltage range
- Operating Current: 200 µA (typical)

There are two versions of this board - the STEMMA QT version shown above, and the original header-only version shown below. Code works the same on both!



Pinouts





The MCP9808 is a very straight-forward sensor, lets go thru all the pins so you can understand what you need to connect to get started

Power Pins

- VIN (VDD on header-only version) - This is the positive power and logic level pin. It can be 2.7-5.5VDC, so fine for use with 3 or 5V logic. Power VIN (VDD) with whatever logic level you plan to use on the i2c lines.
- GND - this is the ground power and logic reference pin.

I2C Data Pins

- SCL - this is the I2C clock pin. There's a 10K pull-up already on the board, so connect this directly to the i2c master clock pin on your microcontroller
- SDA - this is the I2C data pin. There's a 10K pull-up already on the board, so connect this directly to the i2c master data pin on your microcontroller
- STEMMA QT (https://adafru.it/Ft4) - These connectors allow you to connect to development boards with STEMMA QT connectors, or to other things, with various associated accessories (https://adafru.it/Ft6).

Optional Pins

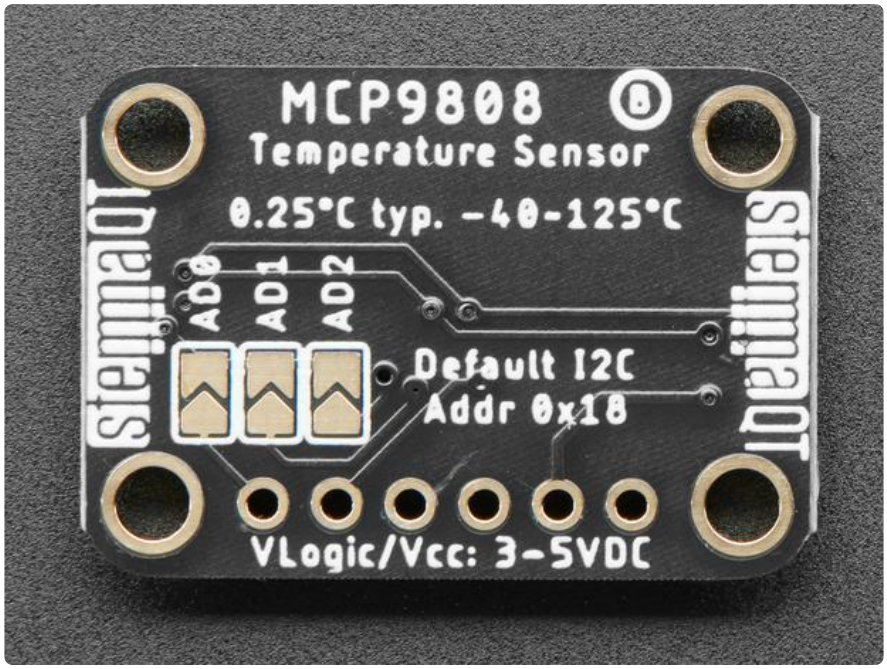
These are pins you don't need to connect to unless you want to!

- Alert - This is the interrupt/alert pin from the MCP9808. The chip has some capability to 'alert' you if the chip temperature goes above or below a set amount. This output can trigger to let you know. It is open collector so you need to use a pull-up resistor if you want to read signal from this pin.
- A0 (as well as A1 and A2 on the original version) - These are the address select pins. Since you can only have one device with a given address on an i2c bus, there must be a way to adjust the address if you want to put more than one MCP9808 on a shared i2c bus. The A0/A1/A2 pins set the bottom three bits of the i2c address. There are pull-down resistors on the board so connect them to VDD to set the bits to '1'. They are read on power up, so de-power and re-power to reset the address

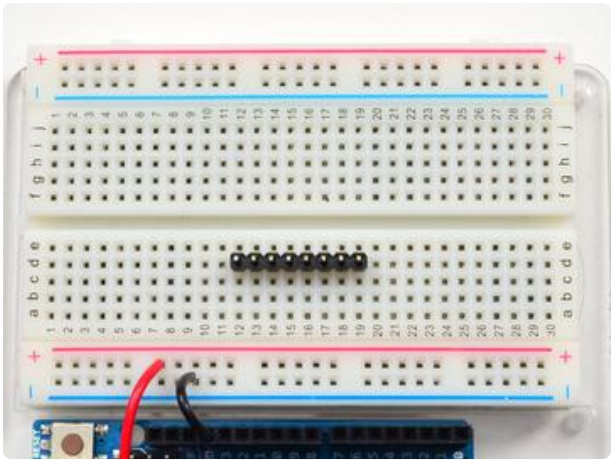
The default address is 0x18 and the address can be calculated by 'adding' the A0/A1/A2 to the base of 0x18
A0 sets the lowest bit with a value of 1, A1 sets the middle bit with a value of 2 and A2 sets the high bit with a value of 4. The final address is 0x18 + A2 + A1 + A0.
So for example if A2 is tied to VDD and A0 is tied to VDD, the address is 0x18 + 4 + 1 =

0x1D.
If only A0 is tied to VDD, the address is 0x18 + 1 = 0x19
If only A1 is tied to VDD, the address is 0x18 + 2 = 0x1A
If only A2 is tied to VDD, the address is 0x18 + 4 = 0x1C

This address information applies to both versions of the MCP9808 breakout. The STEMMA QT version has jumpers on the back that you can use to tie the address pins to VDD.

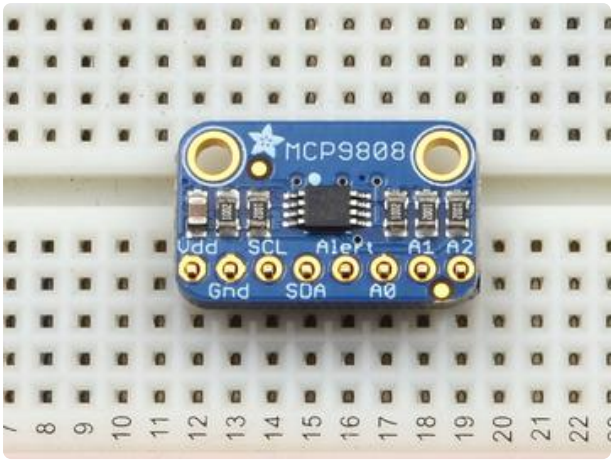


Arduino Code



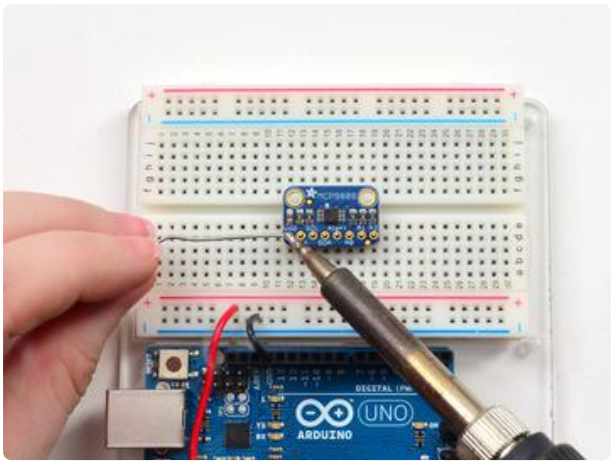
Prepare the header strip:

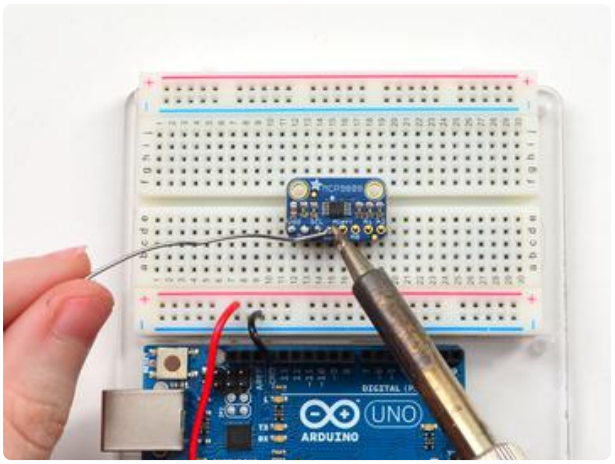
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - long pins down

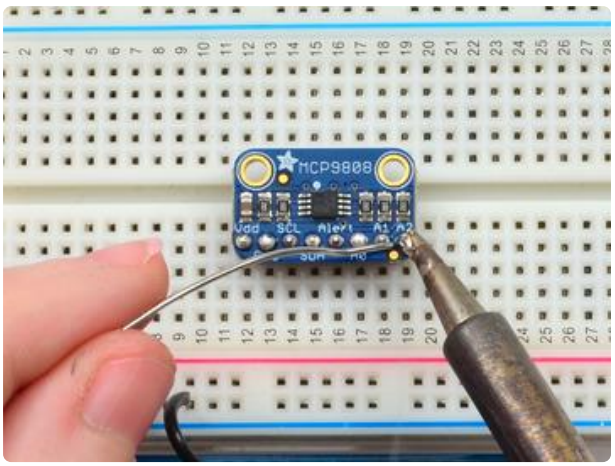


Add the breakout board:

Place the breakout board over the pins so that the short pins poke through the breakout pads



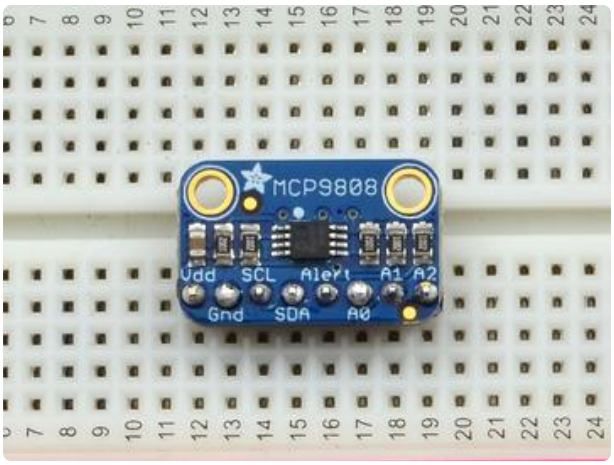




And Solder!

Be sure to solder all pins for reliable electrical contact.

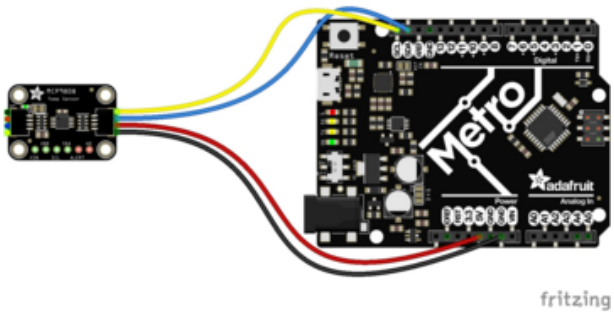
(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk)).

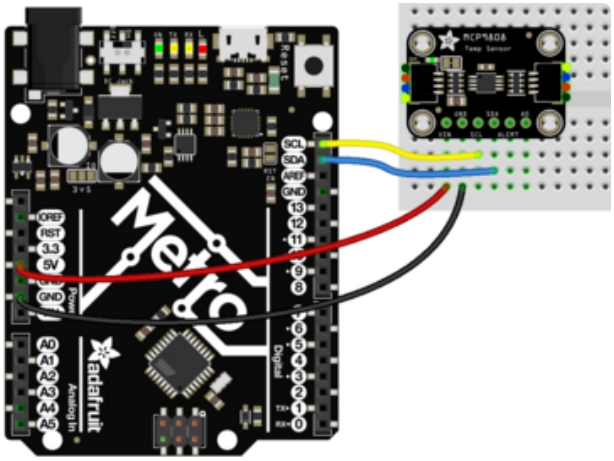


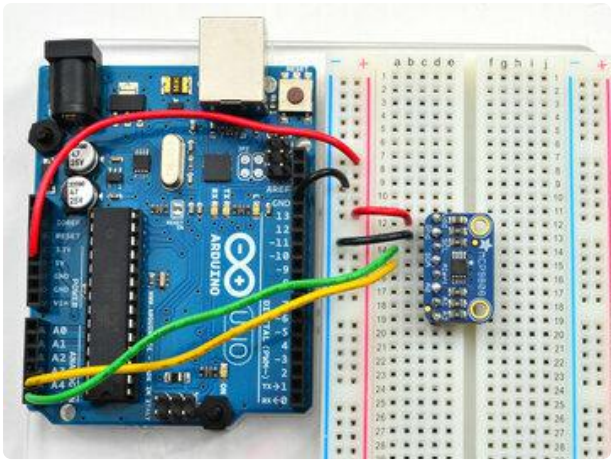
You're done! Check your solder joints visually and continue onto the next steps

Arduino Wiring

You can easily wire this sensor to any microcontroller, we'll be using an Arduino







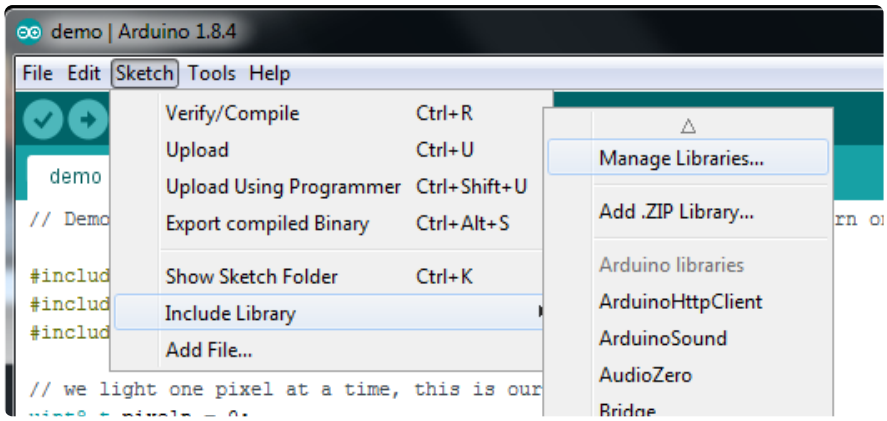
- Connect VIN (Vdd) (red wire on the STEMMA QT version) to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect GND (black wire on the STEMMA QT version) to common power/data ground
- Connect the SCL pin to the I2C clock SCL pin on your Arduino (yellow wire on STEMMA QT version). On an UNO & '328 based Arduino, this is also known as A5, on a Mega it is also known as digital 21 and on a Leonardo/Micro, digital 3
- Connect the SDA pin to the I2C data SDA pin on your Arduino (blue wire on STEMMA QT version). On an UNO & '328 based Arduino, this is also known as A4, on a Mega it is also known as digital 20 and on a Leonardo/Micro, digital 2

The MCP9808 has a default I2C address of 0x18 but you can set the address to any of 8 values between 0x18 and 0x1F so you can have up to 8 of these sensors all sharing the same SCL/SDA pins.

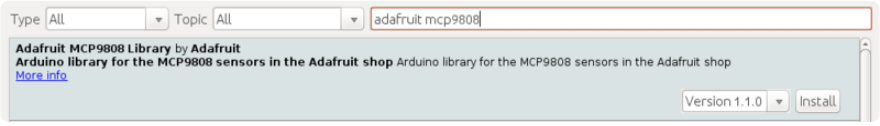
Download Adafruit_MCP9808

To begin reading sensor data, you will need to download the Adafruit MCP9808 library from the Arduino library manager.

Open up the Arduino library manager:



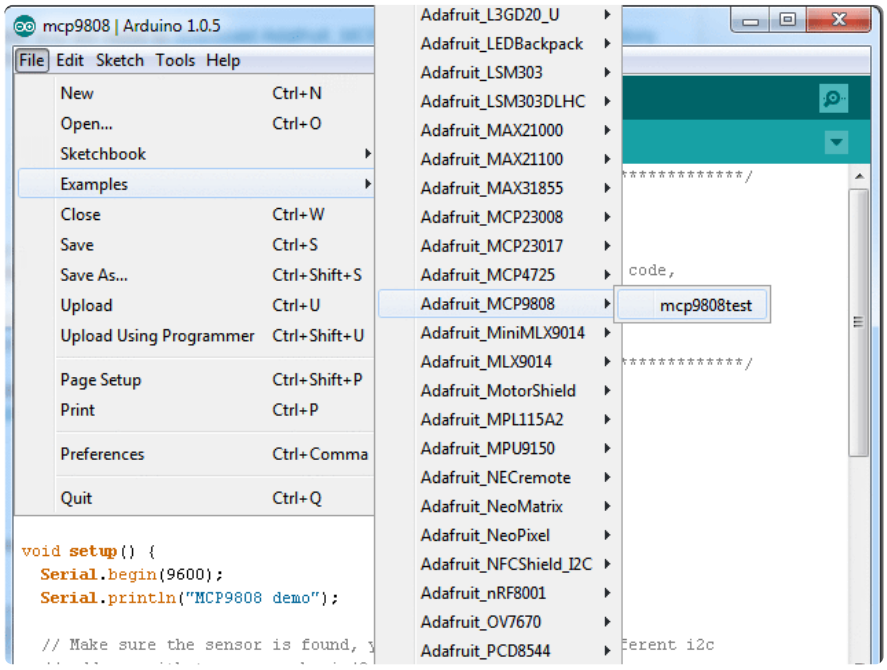
Search for the Adafruit MCP9808 library and install it



We also have a great tutorial on Arduino library installation at: http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (https://adafru.it/aYM)

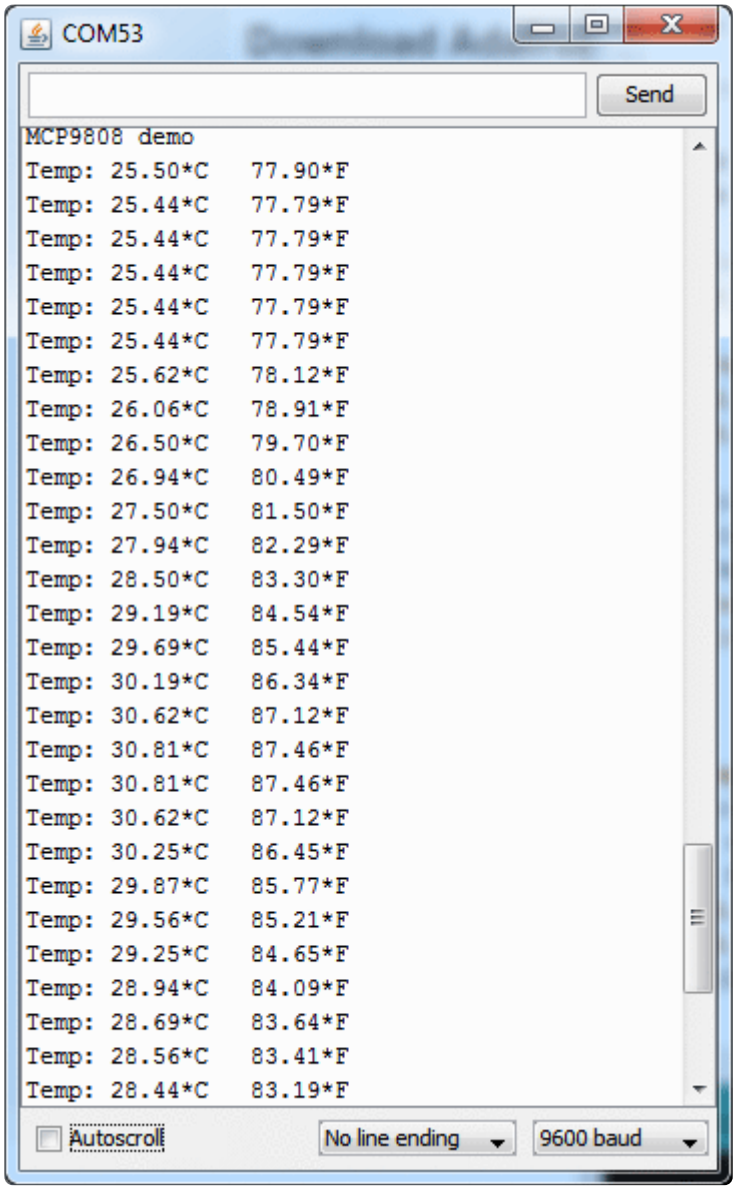
Load Demo

Open up File->Examples->Adafruit_MCP9808->mcp9808test and upload to your Arduino wired up to the sensor



Thats it! Now open up the serial terminal window at 9600 speed to see the temperature in real time. You can try touching your finger to the sensor to see the

temperature rise.



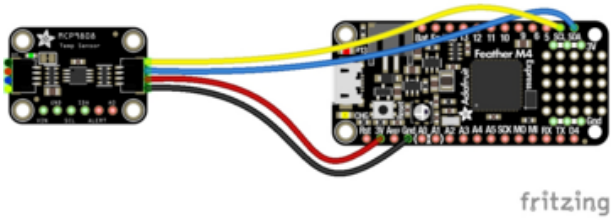
Python & CircuitPython

It's easy to use the MCP9808 sensor with Python or CircuitPython and the Adafruit CircuitPython MCP9808 (https://adafru.it/zcr) module. This module allows you to easily write Python code that reads the temperature from the sensor.

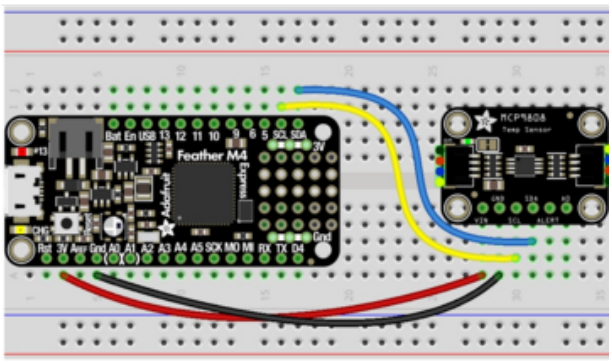
You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

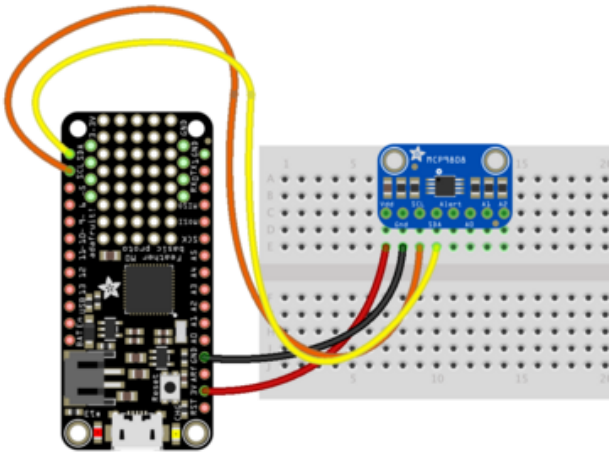
CircuitPython Microcontroller Wiring

First wire up a MCP9808 to your board exactly as shown on the previous pages for Arduino. Here's an example of wiring a Feather M0 to the sensor:



- Board 3V to sensor VIN (Vdd) (red wire on STEMMA QT version)
- Board GND to sensor GND (black wire on STEMMA QT version)
- Board SCL to sensor SCL (yellow wire on STEMMA QT version)
- Board SDA to sensor SDA (blue wire on STEMMA QT version)

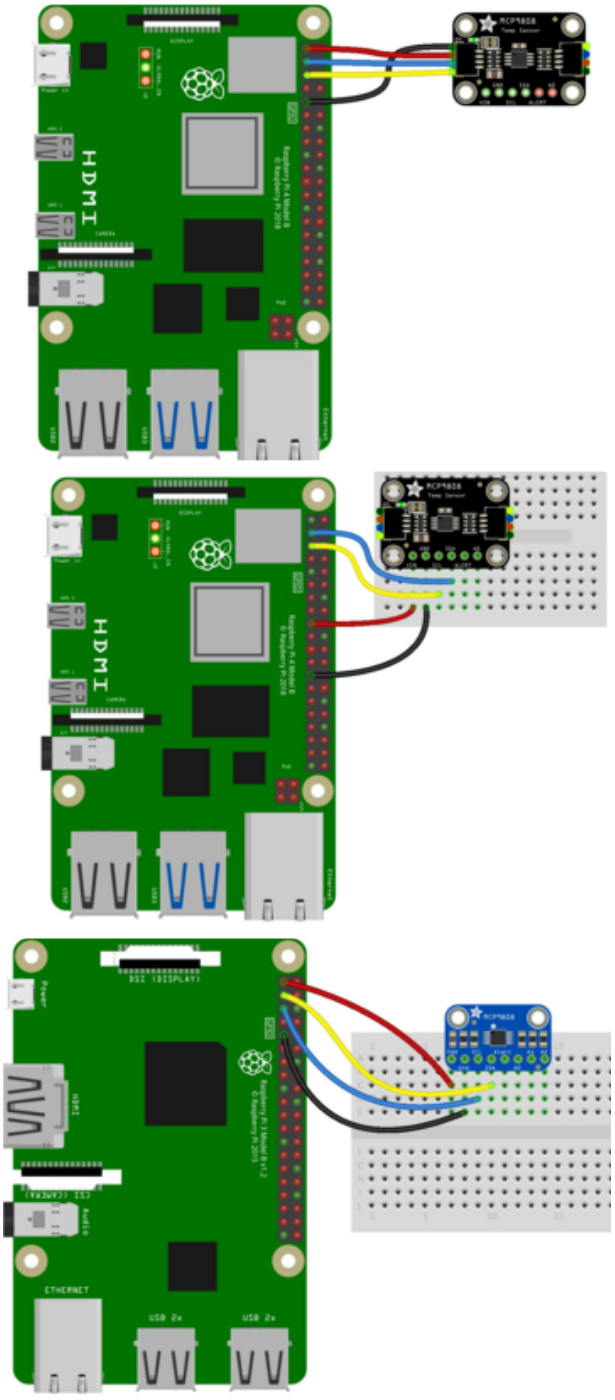




Python Computer Wiring

Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C:



- Pi 3V3 to sensor VIN (red wire on STEMMA QT version)
- Pi GND to sensor GND (black wire on STEMMA QT version)
- Pi SCL to sensor SCK (yellow wire on STEMMA QT version)
- Pi SDA to sensor SDA (blue wire on STEMMA QT version)

CircuitPython Installation of MCP9808 Library

Next you'll need to install the Adafruit CircuitPython MCP9808 (https://adafru.it/zcr) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/tBa) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/zdx). For example the Circuit Playground Express guide has a great page on how to install the library bundle (https://adafru.it/Bf2) for both express and non-express boards.

Remember for non-express boards like the Trinket M0, Gemma M0, and Feather/Metro M0 basic you'll need to manually install the necessary libraries from the bundle:

- adafruit_mcp9808.mpy
- adafruit_bus_device

Before continuing make sure your board's lib folder or root filesystem has the adafruit_mcp9808.mpy, and adafruit_bus_device files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/pMf)so you are at the CircuitPython >>> prompt.

Python Installation of MCP9808 Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

Once that's done, from your command line run the following command:

- `sudo pip3 install adafruit-circuitpython-mcp9808`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython & Python Usage

To demonstrate the usage of the sensor we'll initialize it and read the temperature. First initialize the I2C connection and library by running:

```
import board
import busio
import adafruit_mcp9808
i2c = busio.I2C(board.SCL, board.SDA)
mcp = adafruit_mcp9808.MCP9808(i2c)
```

Now you can read the temperature property to retrieve the temperature from the sensor in degrees Celsius:

```
print('Temperature: {} degrees C'.format(mcp.temperature))
```

That's all there is to reading temperature with the MCP9808 and CircuitPython code!

Full Example Code

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_mcp9808

i2c = board.I2C()  # uses board.SCL and board.SDA

# To initialise using the default address:
mcp = adafruit_mcp9808.MCP9808(i2c)

# To initialise using a specified address:
# Necessary when, for example, connecting A0 to VDD to make address=0x19
# mcp = adafruit_mcp9808.MCP9808(i2c_bus, address=0x19)

while True:
    tempC = mcp.temperature
    tempF = tempC * 9 / 5 + 32
    print("Temperature: {} C {} F ".format(tempC, tempF))
    time.sleep(2)
```

Python Docs

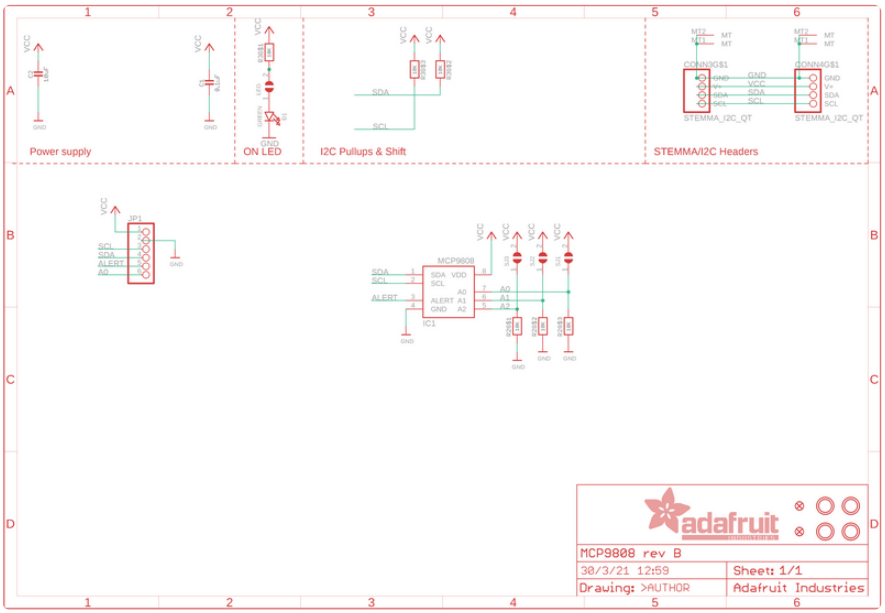
Python Docs (https://adafru.it/C3s)

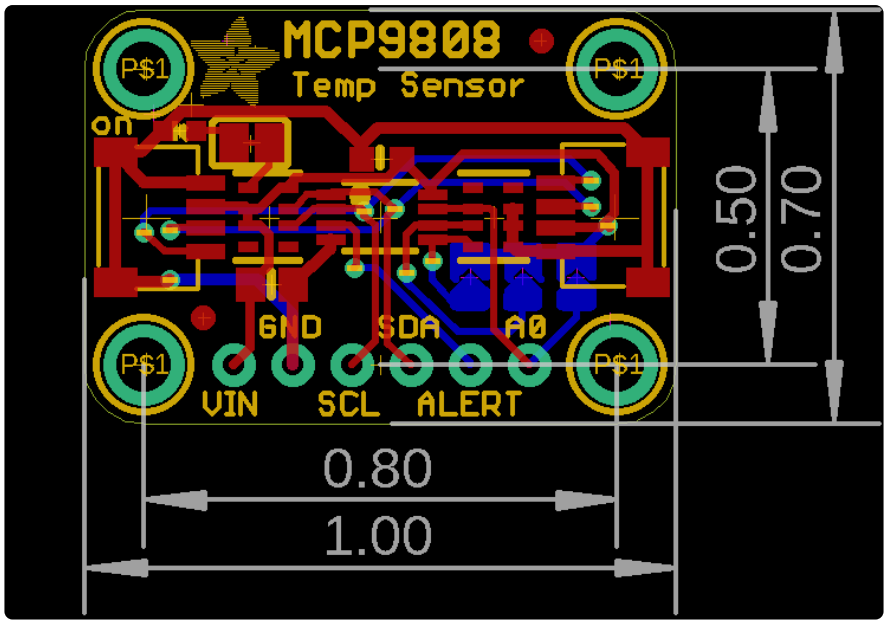
Downloads

Datasheets & Files

- MCP9808 datasheet (https://adafru.it/dfL)
- EagleCAD PCB files on GitHub (https://adafru.it/qib)
- Fritzing object in Adafruit Fritzing library (https://adafru.it/c7M)

Schematic and Fab Print for STEMMA QT Version





Schematic and Fab Print for Original Version