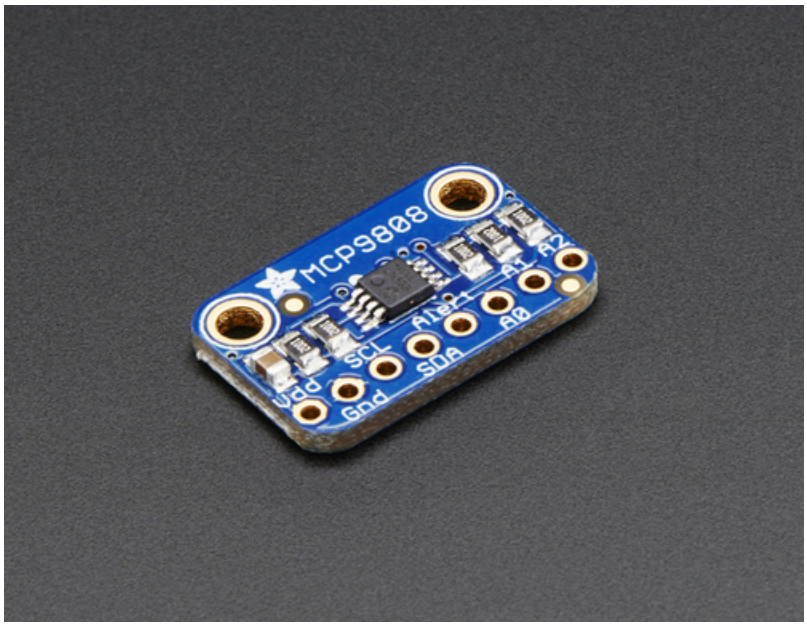
adafruit learning system

Adafruit MCP9808 Precision I2C Temperature Sensor Guide

Created by lady ada



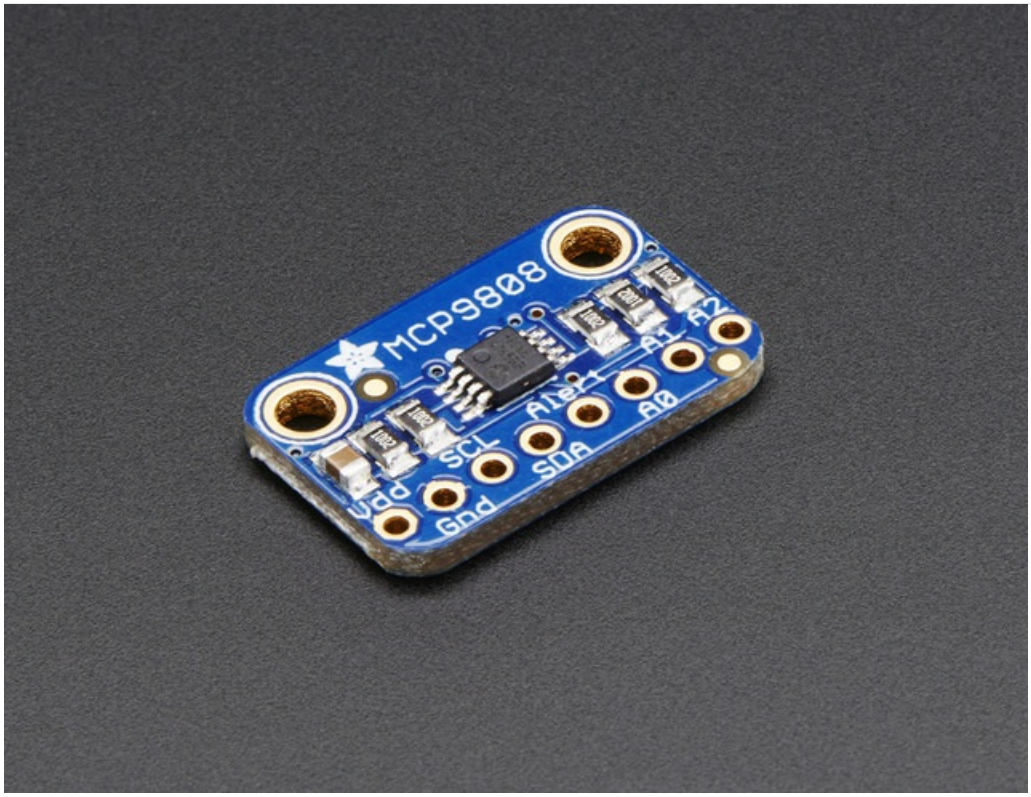
Last updated on 2017-11-12 06:09:49 PM UTC

Guide Contents

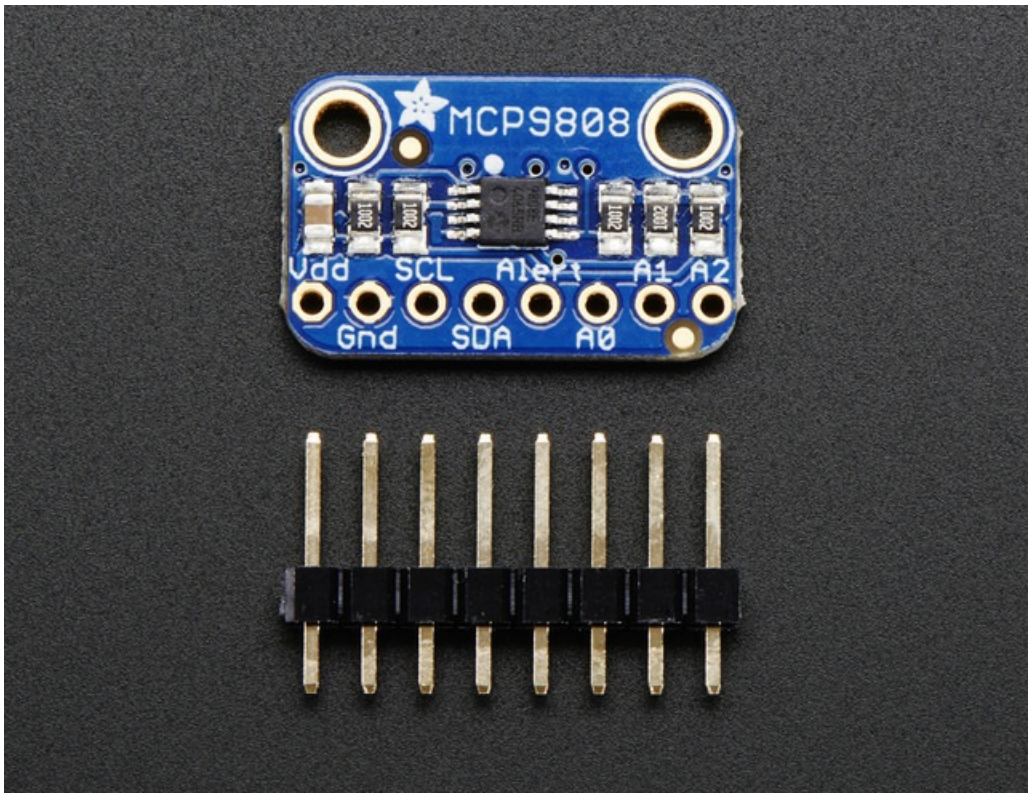
Guide Contents 2
Overview 3
Pinouts 6
Power Pins 6
I2C Data Pins 6
Optional Pins 6
Arduino Code 8
- Prepare the header strip: 8
- Add the breakout board: 8
- And Solder! 9

Arduino Wiring 10
Download Adafruit_MCP9808 11
Load Demo 11
CircuitPython Code 13
Usage 14
Downloads 15
Datasheets & Files 15
Schematic and Diagrams 15

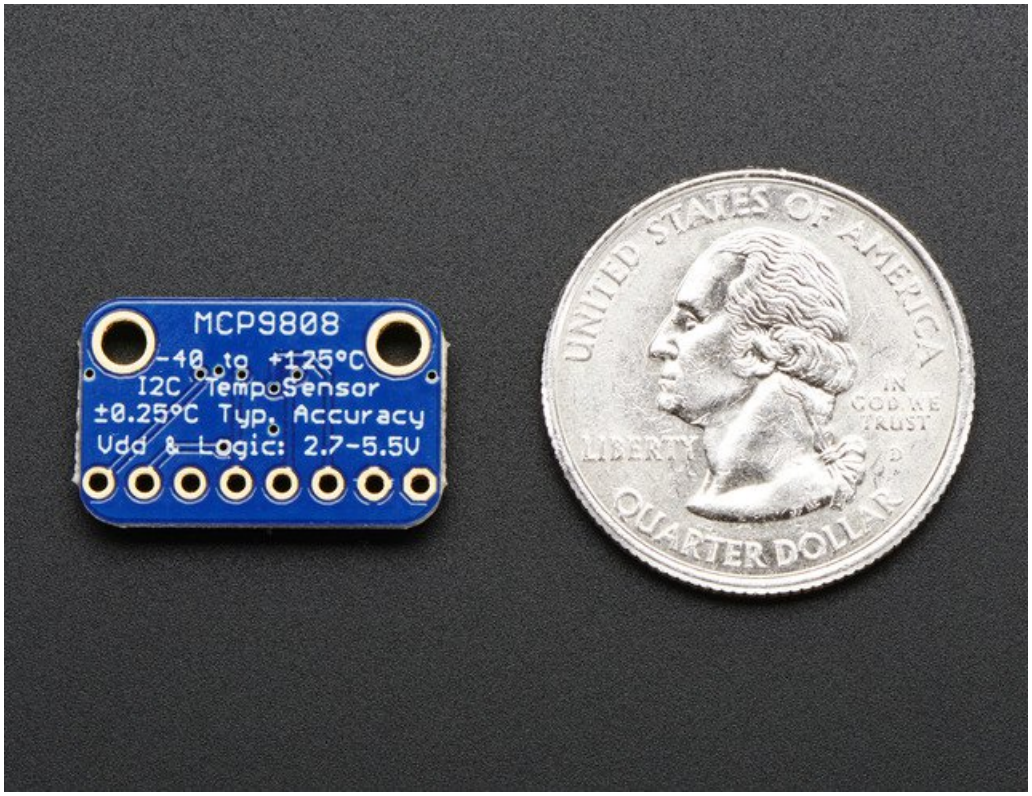
Overview



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!

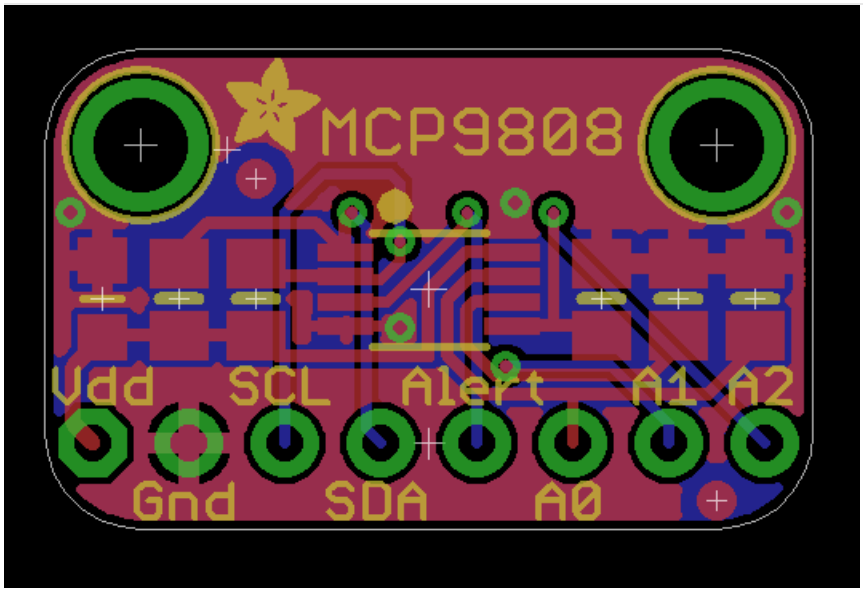


Unlike the DS18B20, this sensor does not come in through-hole package so we placed this small sensor on a breakout board PCB for easy use. The PCB includes mounting holes, and pull down resistors for the 3 address pins. We even wrote a lovely little library for Arduino that will work with any Arduino compatible. You'll be up and running in 15 minutes or less.



- 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)

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

- **VDD** - 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 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

Optional Pins

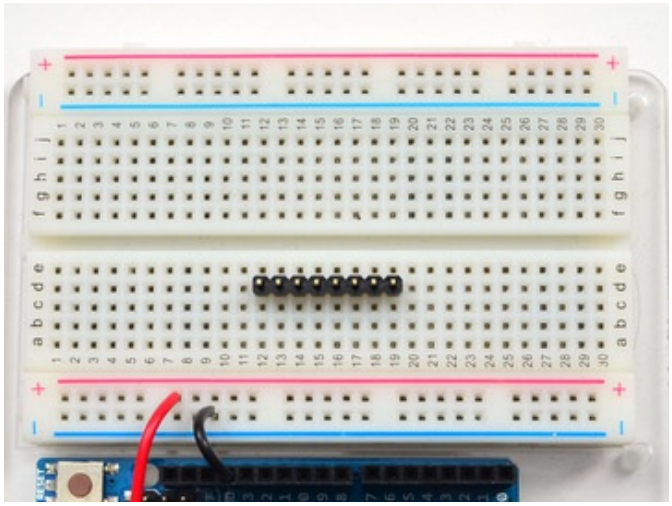
These are pins you don't need to connect to unless you want to!

- **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 A1 A2** - 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 pins 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**

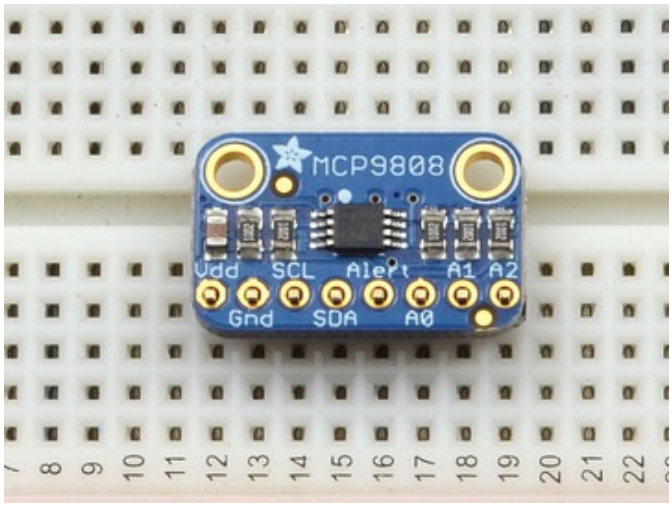
If only A1 is tied to VDD, the address is **0x18** + **2** = **0x1A**
If only A2 is tied to VDD, the address is **0x18** + **4** = **0x1C**

Arduino Code



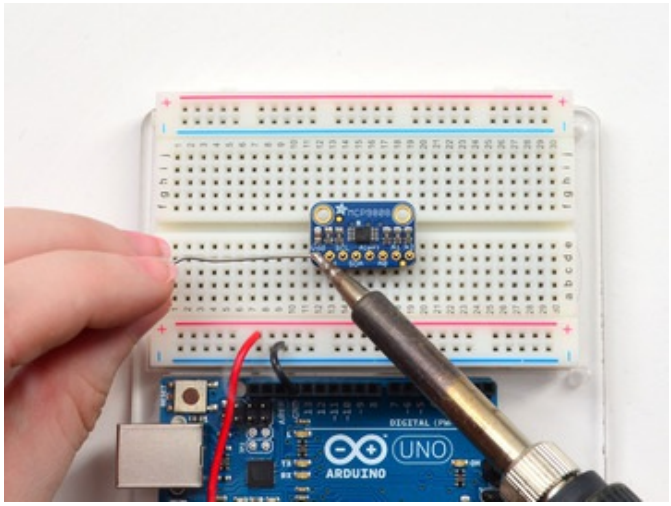
Prepare the header strip:

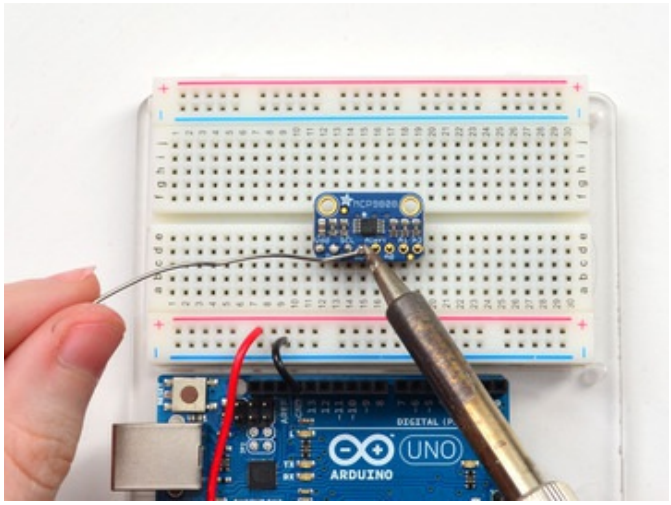
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

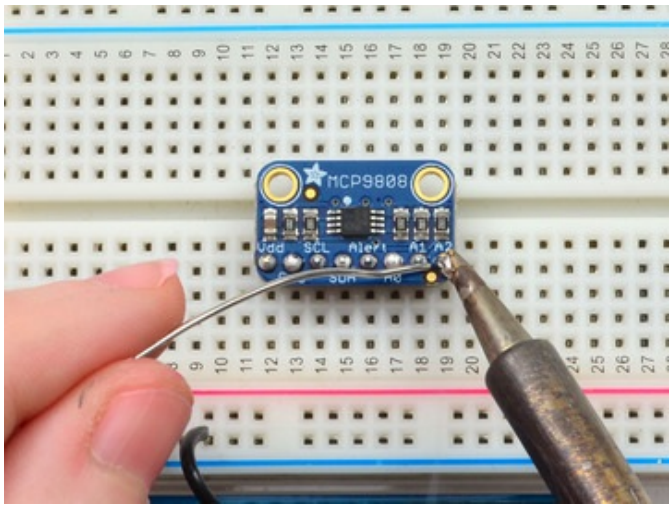


Add the breakout board:

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



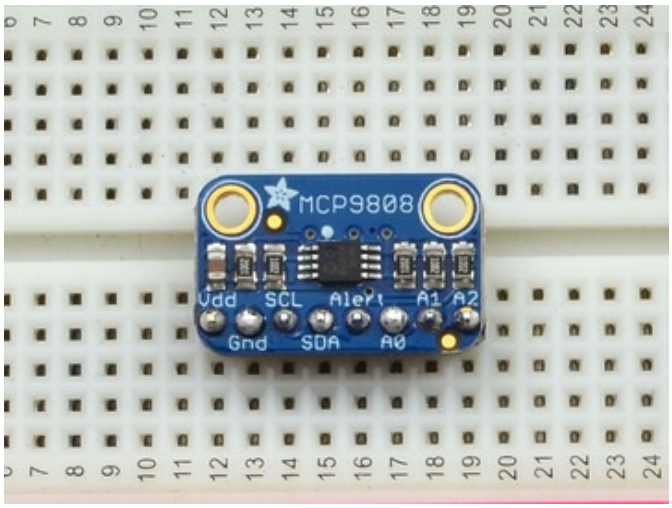




And Solder!

Be sure to solder all pins for reliable electrical contact.

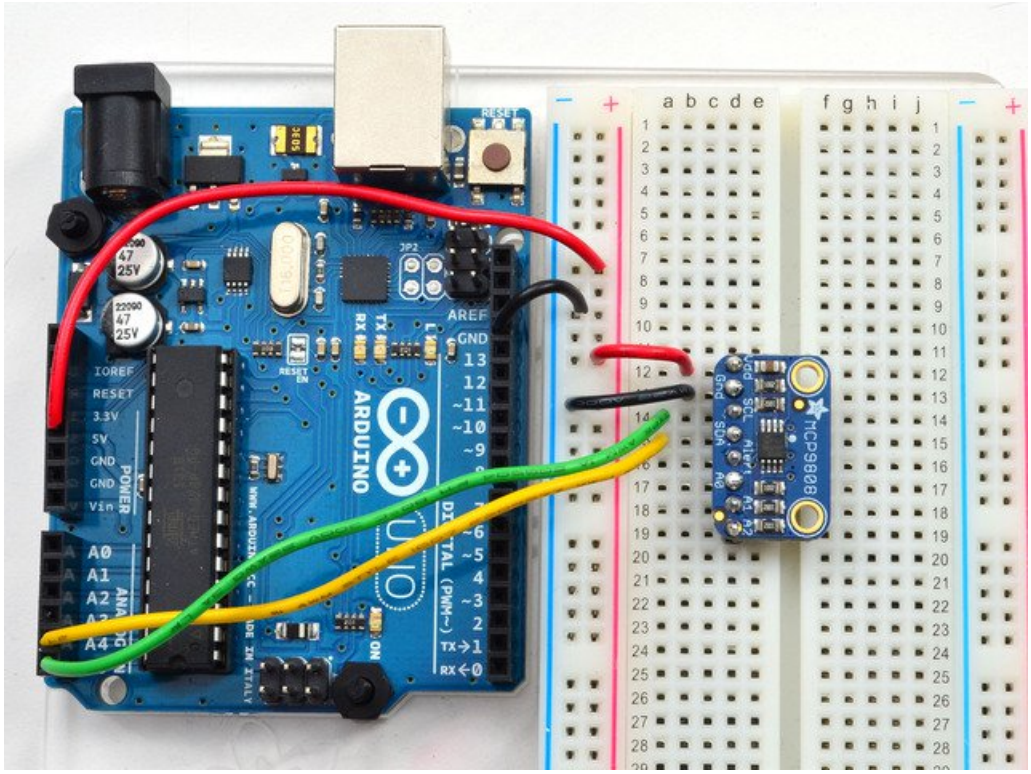
(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk).



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 **Vdd** 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** to common power/data ground
- Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino. 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. 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 Adafruit_MCP9808 from our github repository. You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

Download Adafruit MCP9808 Library

https://adafru.it/dfQ

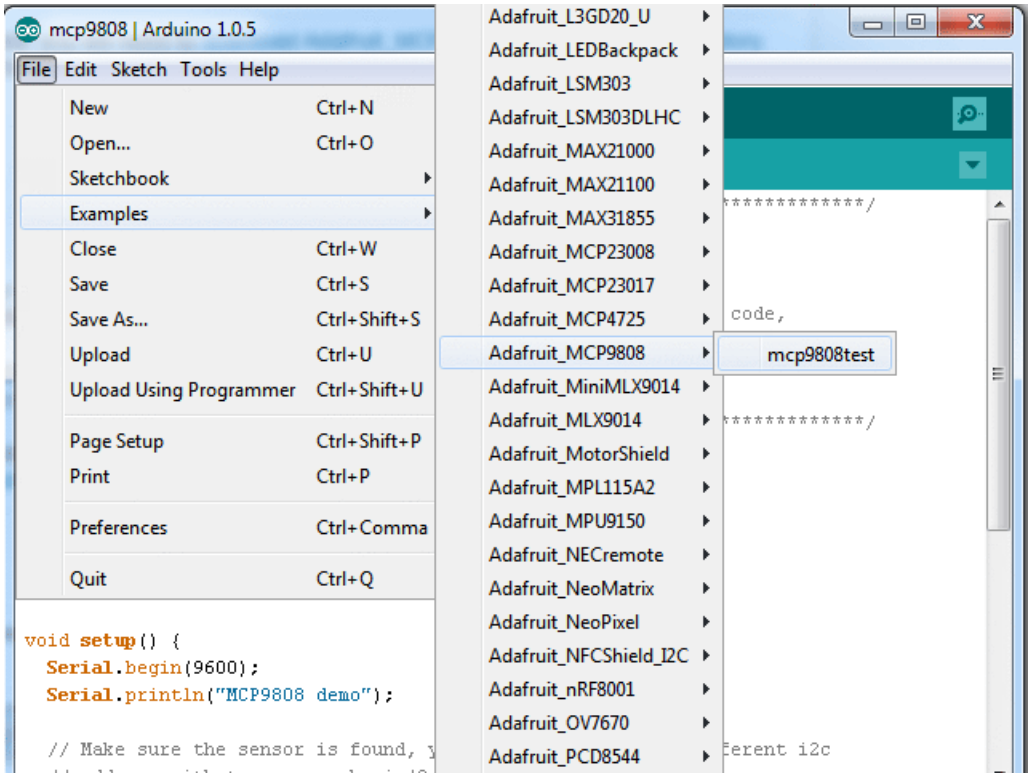
Rename the uncompressed folder **Adafruit_MCP9808** and check that the **Adafruit_MCP9808** folder contains **Adafruit_MCP9808.cpp** and **Adafruit_MCP9808.h**

Place the **Adafruit_MCP9808** library folder your **arduinosketchfolder/libraries/** folder.
You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

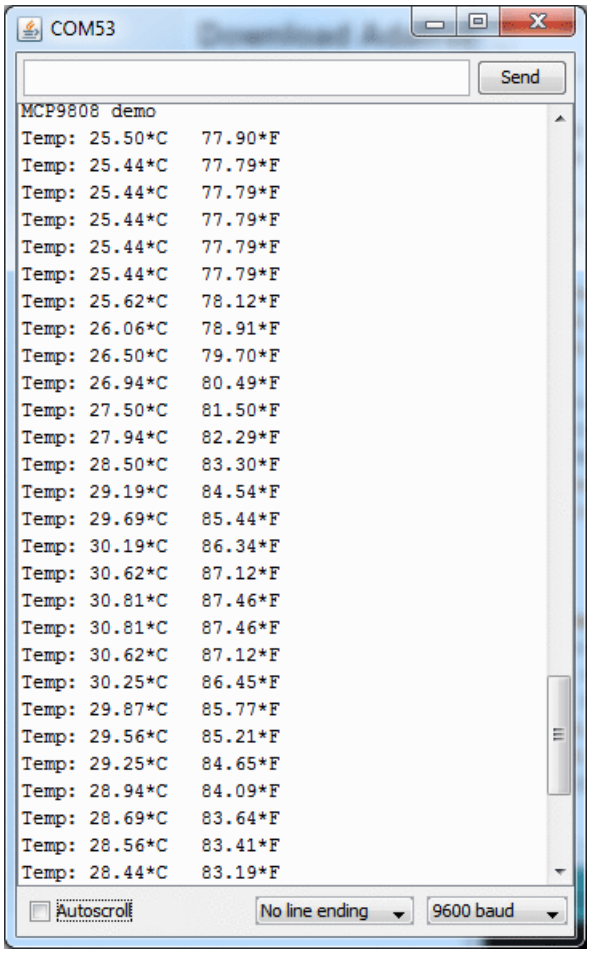
We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use

Load Demo

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



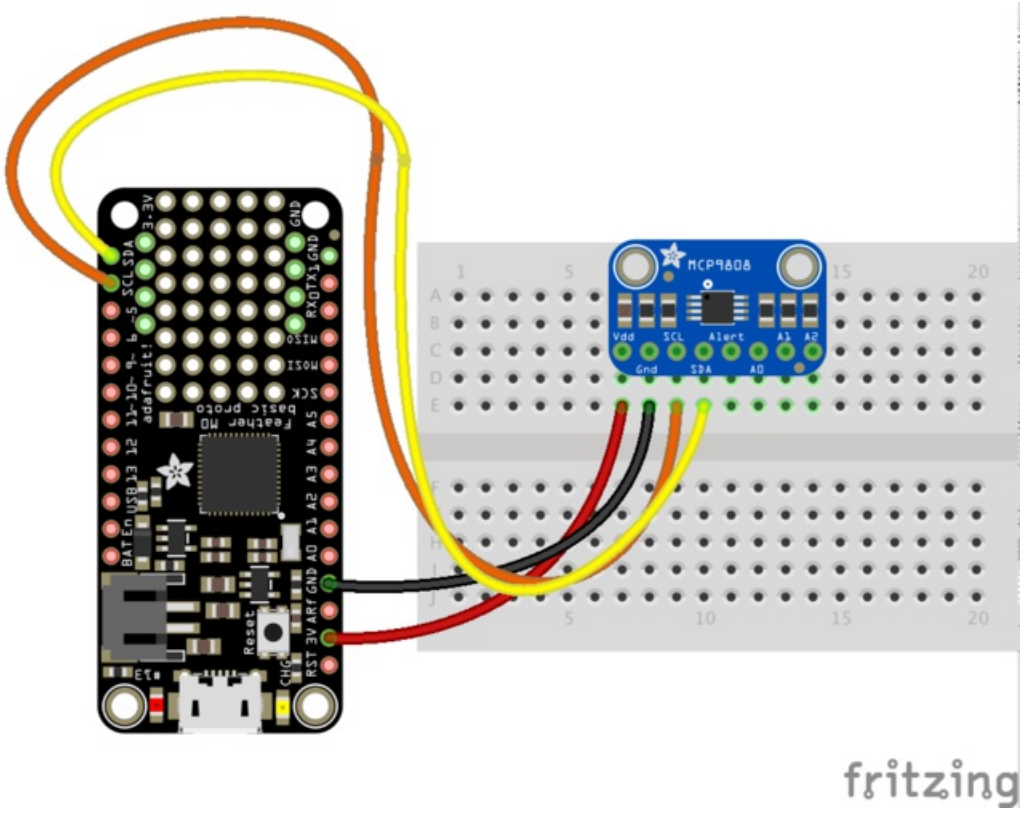
Thats it! Now open up the serial terminal window at 9600 speed to see the temperature in real time. You can try touching your finger to the sensor to see the temperature rise.


COM53
Send
MCP9808 demo
Temp: 25.50*C 77.90*F
Temp: 25.44*C 77.79*F
Temp: 25.44*C 77.79*F
Temp: 25.44*C 77.79*F
Temp: 25.44*C 77.79*F
Temp: 25.44*C 77.79*F
Temp: 25.62*C 78.12*F
Temp: 26.06*C 78.91*F
Temp: 26.50*C 79.70*F
Temp: 26.94*C 80.49*F
Temp: 27.50*C 81.50*F
Temp: 27.94*C 82.29*F
Temp: 28.50*C 83.30*F
Temp: 29.19*C 84.54*F
Temp: 29.69*C 85.44*F
Temp: 30.19*C 86.34*F
Temp: 30.62*C 87.12*F
Temp: 30.81*C 87.46*F
Temp: 30.81*C 87.46*F
Temp: 30.62*C 87.12*F
Temp: 30.25*C 86.45*F
Temp: 29.87*C 85.77*F
Temp: 29.56*C 85.21*F
Temp: 29.25*C 84.65*F
Temp: 28.94*C 84.09*F
Temp: 28.69*C 83.64*F
Temp: 28.56*C 83.41*F
Temp: 28.44*C 83.19*F
Autoscroll
No line ending
9600 baud

CircuitPython Code

It's easy to use the MC9808 sensor with CircuitPython and the Adafruit CircuitPython MCP9808 module. This module allows you to easily write Python code that reads the temperature from the sensor.

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 Vdd**
- **Board GND** to **sensor GND**
- **Board SCL** to **sensor SCL**
- **Board SDA** to **sensor SDA**

Next you'll need to install the Adafruit CircuitPython MCP9808 library on your CircuitPython board. **Remember this module is for Adafruit CircuitPython firmware and not MicroPython.org firmware!**

First make sure you are running the latest version of Adafruit CircuitPython 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. For example the Circuit Playground Express guide has a great page on how to install the library bundle 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**

You can also download the **adafruit_mcp9808.mpy** from its releases page on Github.

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 so you are at the CircuitPython >>> prompt.

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)
```

Remember if you're using a board that doesn't support hardware I2C (like the ESP8266) you need to use the **bitbangio** module instead:

```
import board
import bitbangio
import adafruit_mcp9808
i2c = bitbangio.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))
```

```
>>> print('Temperature: {} degrees C'.format(mcp.temperature))
Temperature: 21.25 degrees C
>>>
```

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

Downloads

Datasheets & Files

- MCP9808 datasheet
- EagleCAD PCB files on GitHub
- Fritzing object in Adafruit Fritzing library

Schematic and Diagrams