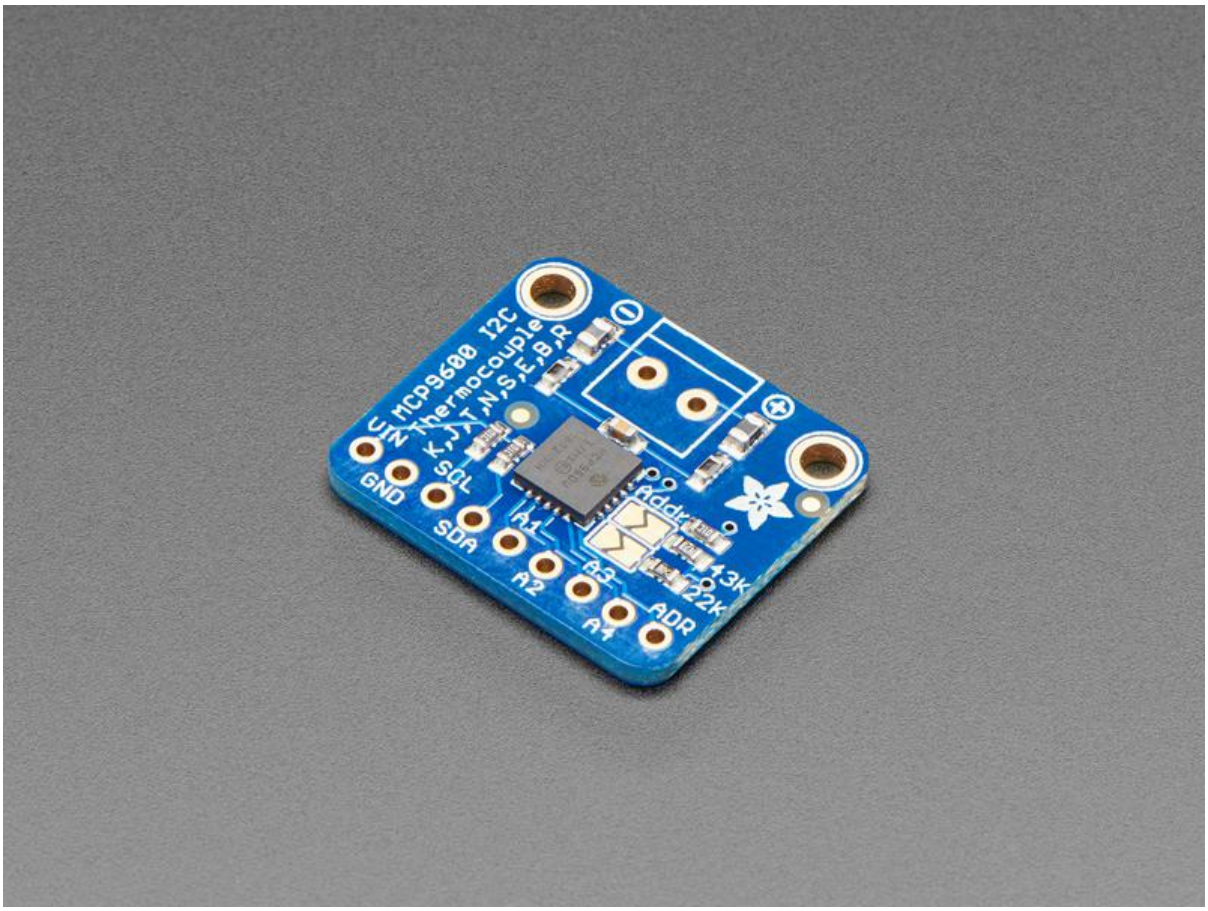
Adafruit MCP9600 I2C Thermocouple Amplifier

Created by Kattni Rembor

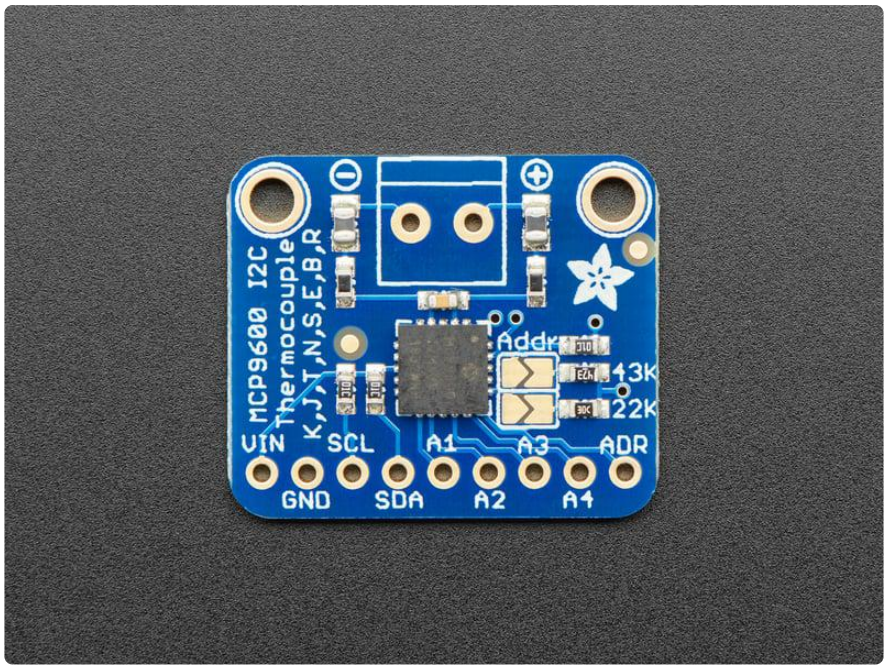


https://learn.adafruit.com/adafruit-mcp9600-i2c-thermocouple-amplifier

Last updated on 2022-12-01 03:37:54 PM EST

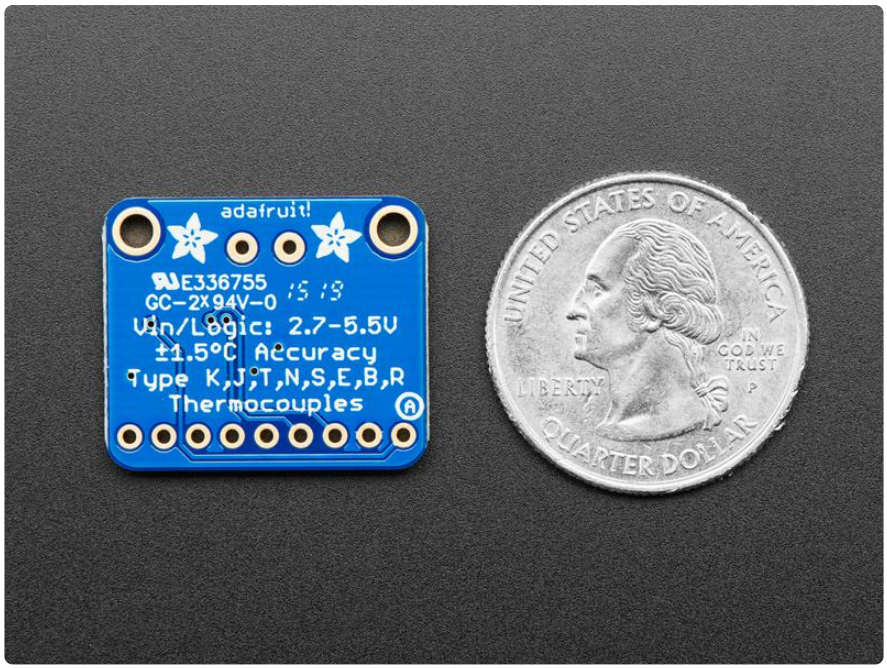
Table of Contents

Overview



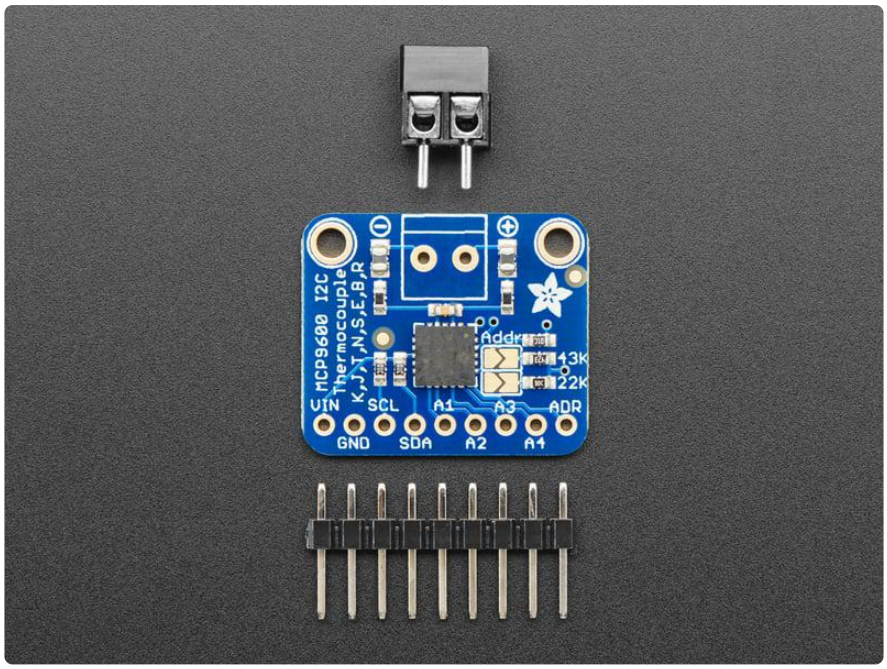
Thermocouples are very sensitive, requiring a good amplifier with a cold-compensation reference. The Adafruit MCP9600 does all that for you, and can be easily interfaced with any microcontroller or single-board-computer with I2C. Inside, the chip handles all the analog stuff for you, and can interface with just about any thermocouple type: K, J, T, N, S, E, B and R type are all supported! You can also set various alerts for over/under temperature, and read the thermocouple (hot) temperature and the chip (cold) temperature. All this over common I2C.

This breakout board has the chip itself, a 3.3V regulator and level shifting circuitry, all assembled and tested. Works great with 3.3V or 5V logic. Comes with a 2 pin terminal block (for connecting to the thermocouple) and pin header (to plug into any breadboard or perfboard).

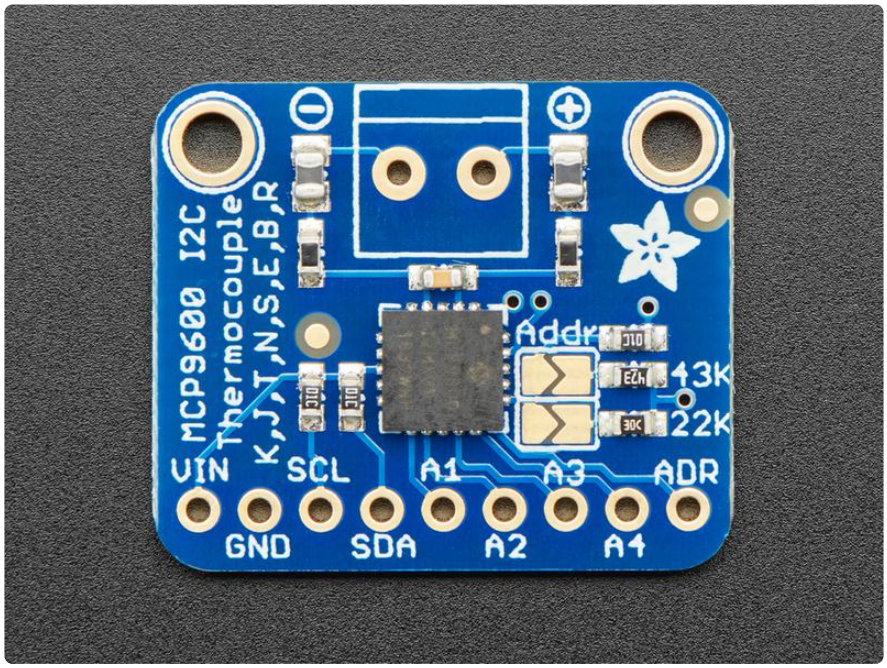


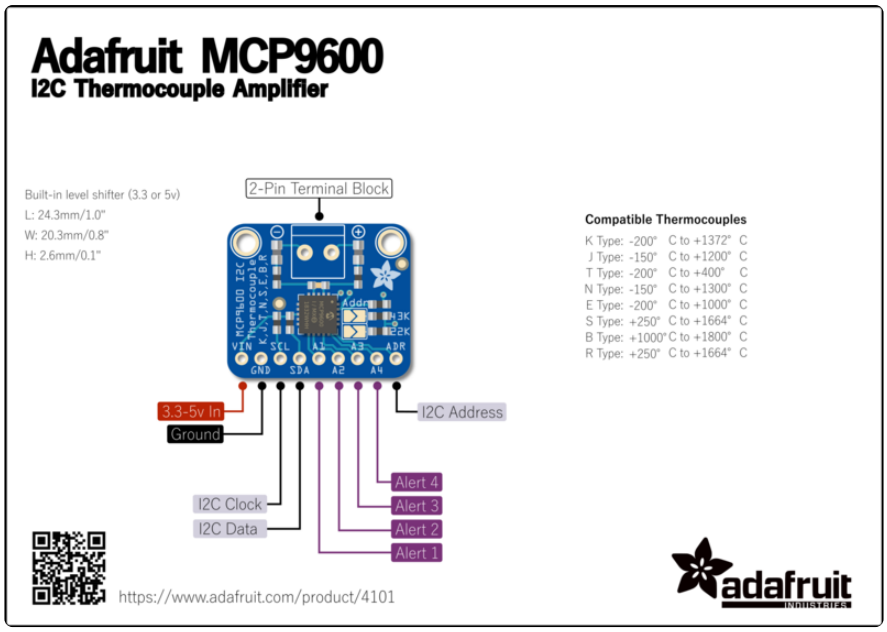
The Adafruit MCP9600 features:

- Works with any K, J, T, N, S, E, B and R type thermocouple
- Datasheet rated for:
  K Type: -200°C to +1372°C
  J Type: -150°C to +1200°C
  T Type: -200°C to +400°C
  N Type: -150°C to +1300°C
  E Type: -200°C to +1000°C
  S Type: +250°C to +1664°C
  B Type: +1000°C to +1800°C
  R Type: +250°C to +1664°C
- Resolution of ±0.0625 °C - note that K thermocouples have about ±2°C to ±6°C accuracy
- Internal temperature reading
- 3.3 to 5v power supply and logic level compliant
- I2C data connection


MCP9600 I2C
Thermocouple
K,J,T,N,S,E,B,R
Addr
13k
22k
VIN
SCL
A1
A3
ADR
GND
SDA
A2
A1

Pinouts


MCP9600 I2C
Thermocouple
K,J,T,N,S,E,B,R
Addr
13k
22k
VIN
SCL
A1
A3
ADR
GND
SDA
A2
A1



Power Pins

- Vin - this is the power pin. This chip can handle 2.7V to 5V. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
- GND - common ground for power and logic

I2C Logic Pins

- SCL - this is the I2C clock pin, connect to your microcontroller's I2C clock line.
- SDA - this is the I2C data pin, connect to your microcontroller's I2C data line.

Alert Pins

- A1 - A4 - Alert 1 - Alert 4 output pins

Address Pin

- ADR - Allows for setting I2C address

Arduino

Wiring

Connecting the MCP9600 to your Feather or Arduino is easy:

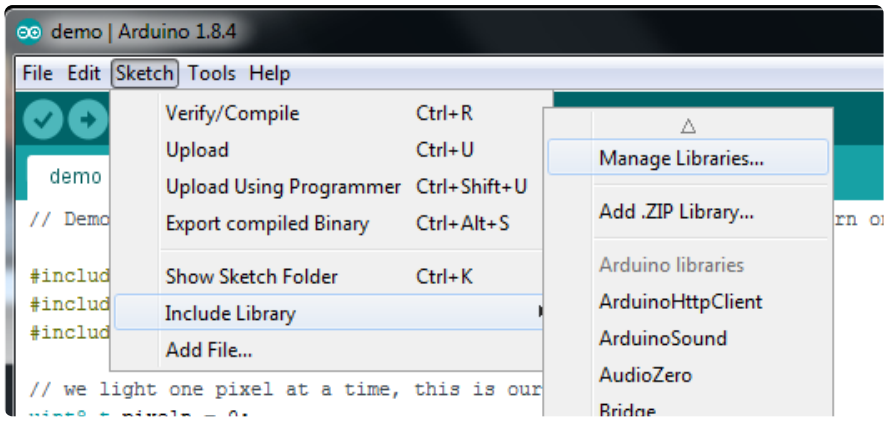
If you are running a Feather (3.3V), connect Feather 3V to board VIN
If you are running a 5V Arduino (Uno, etc.), connect Arduino 5V to board VIN
Connect Feather or Arduino GND to board GND
Connect Feather or Arduino SCL to board SCL
Connect Feather or Arduino SDA to board SDA
Connect thermocouple + to board screw terminal +
Connect thermocouple - to board screw terminal -

The final results should resemble the illustration above, showing an Adafruit Metro development board.

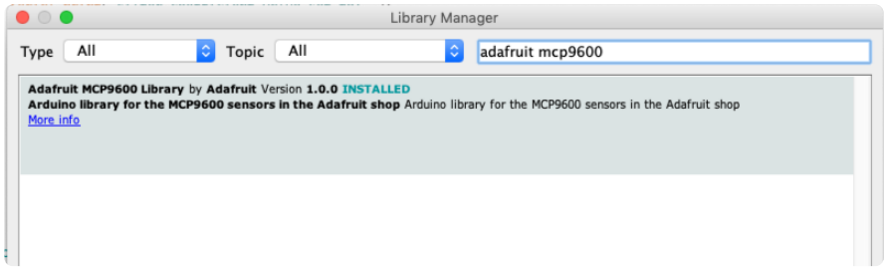
The MCP9600 will return a temperature for the hot junction even if there is no thermocouple connected. There will not be an error!

Installation

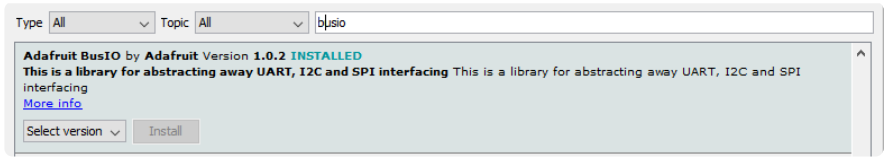
You can install the Adafruit MCP9600 Library for Arduino using the Library Manager in the Arduino IDE:



Click the Manage Libraries ... menu item, search for Adafruit MCP9600, and select the Adafruit MCP9600 library:



Also get the Adafruit BusIO library



Load Example

Open up File -> Examples -> Adafruit MCP9600 -> mcp9600_test and upload to your Arduino wired up to the sensor.

Upload the sketch to your board and open up the Serial Monitor (Tools->Serial Monitor). You should see the the values for hot junction, cold junction and ADC.

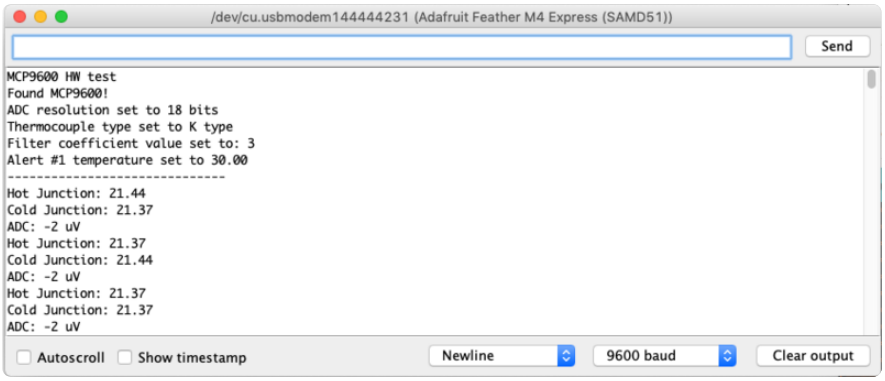
Example Code

The following example code is part of the standard library, but illustrates how you can retrieve sensor data from the MCP9600 for the hot junction, cold junction and ADC values:

```
#include <Wire.h>
#include <Adafruit_I2CDevice.h>
#include <Adafruit_I2CRegister.h>
#include "Adafruit_MCP9600.h"

#define I2C_ADDRESS (0x67)

Adafruit_MCP9600 mcp;

void setup()
{
    Serial.begin(115200);
    while (!Serial) {
      delay(10);
    }
    Serial.println("MCP9600 HW test");

    /* Initialise the driver with I2C_ADDRESS and the default I2C bus. */
    if (! mcp.begin(I2C_ADDRESS)) {
        Serial.println("Sensor not found. Check wiring!");
        while (1);
    }

  Serial.println("Found MCP9600!");

  mcp.setADCresolution(MCP9600_ADCRESOLUTION_18);
  Serial.print("ADC resolution set to ");
  switch (mcp.getADCresolution()) {
    case MCP9600_ADCRESOLUTION_18:   Serial.print("18"); break;
    case MCP9600_ADCRESOLUTION_16:   Serial.print("16"); break;
    case MCP9600_ADCRESOLUTION_14:   Serial.print("14"); break;
    case MCP9600_ADCRESOLUTION_12:   Serial.print("12"); break;
  }
  Serial.println(" bits");

  mcp.setThermocoupleType(MCP9600_TYPE_K);
  Serial.print("Thermocouple type set to ");
  switch (mcp.getThermocoupleType()) {
    case MCP9600_TYPE_K:  Serial.print("K"); break;
    case MCP9600_TYPE_J:  Serial.print("J"); break;
    case MCP9600_TYPE_T:  Serial.print("T"); break;
    case MCP9600_TYPE_N:  Serial.print("N"); break;
    case MCP9600_TYPE_S:  Serial.print("S"); break;
    case MCP9600_TYPE_E:  Serial.print("E"); break;
    case MCP9600_TYPE_B:  Serial.print("B"); break;
    case MCP9600_TYPE_R:  Serial.print("R"); break;
  }
  Serial.println(" type");

  mcp.setFilterCoefficient(3);
  Serial.print("Filter coefficient value set to: ");
  Serial.println(mcp.getFilterCoefficient());

  mcp.setAlertTemperature(1, 30);
  Serial.print("Alert #1 temperature set to ");
  Serial.println(mcp.getAlertTemperature(1));
  mcp.configureAlert(1, true, true);  // alert 1 enabled, rising temp

  mcp.enable(true);

  Serial.println(F("------------------------------"));
}

void loop()
{
  Serial.print("Hot Junction: "); Serial.println(mcp.readThermocouple());
  Serial.print("Cold Junction: "); Serial.println(mcp.readAmbient());
  Serial.print("ADC: "); Serial.print(mcp.readADC() * 2); Serial.println(" uV");
```

```
  delay(1000);
}
```

You should get something resembling the following output when you open the Serial Monitor at 115200 baud:



Note: The image above shows 9600 baud in the serial monitor. It should be 115200 to match the code!

Arduino Docs

Arduino Docs ()

Python & CircuitPython

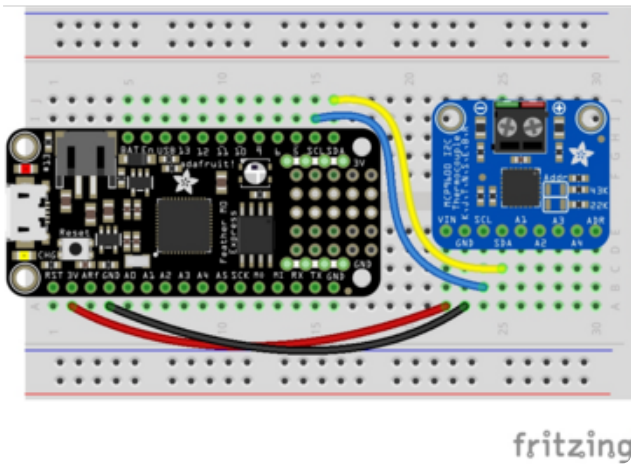
It's easy to use the MCP9600 thermocouple amplifier with CircuitPython or Python, and the Adafruit CircuitPython MCP9600 () module. This module allows you to easily write Python code that reads the temperature from the sensor.

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 ().

MCP960x devices do not work properly with the current CircuitPython I2C support on ESP32-S2 and -S3 boards. This problem is under investigation: https://github.com/adafruit/circuitpython/issues/6311

CircuitPython Microcontroller Wiring

First wire up a MCP9600 to your board exactly as shown on the previous pages for Arduino. Here's an example of wiring a Feather M0 to the sensor with I2C:

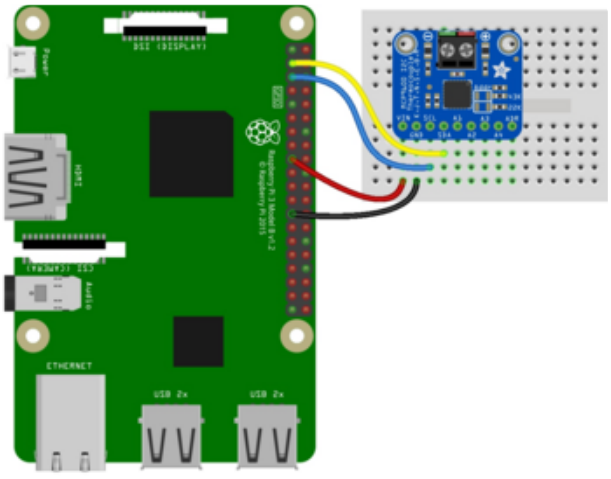


Board 3V to sensor VIN
Board GND to sensor GND
Board SCL to sensor SCL
Board SDA to sensor SDA

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 ().

Here's the Raspberry Pi wired with I2C:



Pi 3V3 to sensor VIN
Pi GND to sensor GND
Pi SCL to sensor SCL
Pi SDA to sensor SDA

Older versions of the Raspberry Pi firmware do not have I2C clock stretching support so they may not work well with the MCP9600. Please ensure your firmware is updated to the latest version before continuing and slow down the

I2C as explained here https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/i2c-clock-stretching

CircuitPython Installation of MCP9600 Library

Next you'll need to install the Adafruit CircuitPython MCP9600 () library on your CircuitPython board.

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 (). Our introduction guide has a great page on how to install the library bundle () for both express and non-express boards.

Copy the following files from the library bundle to your CIRCUITPY drive:

- adafruit_mcp9600.mpy
- adafruit_bus_device

Before continuing make sure your board's lib folder or root filesystem has the adafruit_mcp9600.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.

Python Installation of MCP9600 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 ()!

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

- `pip3 install adafruit-circuitpython-mcp9600`

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 and Python Usage

To demonstrate the usage of the sensor we'll initialize it and read the temperature from the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the sensor. Note that `frequency` must be set when I2C is initialised for the MCP9600 to work:

```
import board
import busio
import adafruit_mcp9600

i2c = busio.I2C(board.SCL, board.SDA, frequency=100000)
mcp = adafruit_mcp9600.MCP9600(i2c)
```

Now you're ready to read values from the sensor using any of these properties:

- temperature - The thermocouple or hot junction temperature in degrees Celsius.
- ambient_temperature - The ambient or cold-junction temperature in degrees Celsius.
- delta_temperature - The difference between the thermocouple (hot junction) and ambient (cold junction) temperatures in degrees Celsius.

```
print(mcp.temperature)
```

```
>>> print(mcp.temperature)
20.1875
```

Full Example Code

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

import time
import board
import busio
import adafruit_mcp9600

i2c = busio.I2C(board.SCL, board.SDA, frequency=100000)
mcp = adafruit_mcp9600.MCP9600(i2c)

while True:
```

```
    print((mcp.ambient_temperature, mcp.temperature, mcp.delta_temperature))
    time.sleep(1)
```

Alerts and More

The MCP9600 breakout allows you to configure four separate alerts on four pins. Connect the alert pins to digital output pins on your board or computer, and use the alert configuration in the MCP9600 library to configure them. Check out the documentation () for more information!

Python Docs

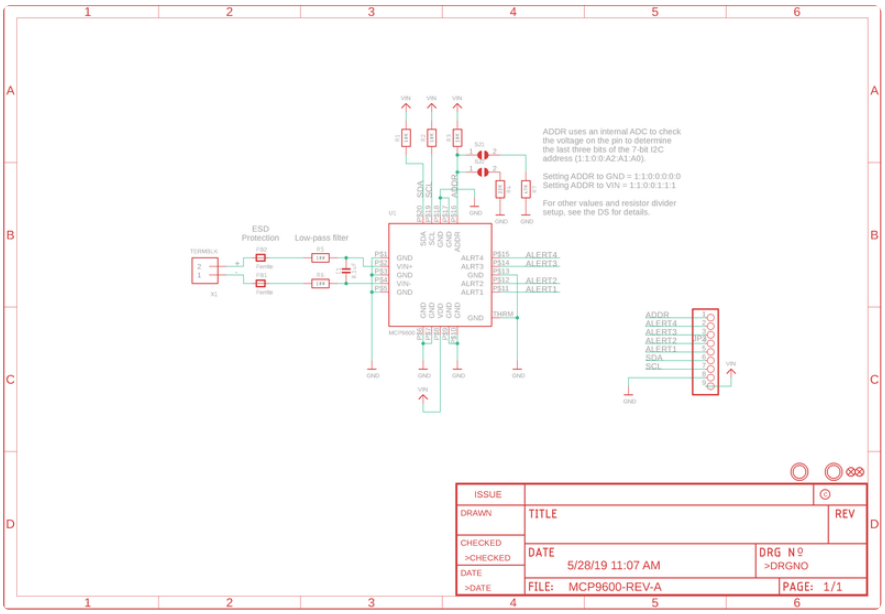
Python Docs ()

Downloads

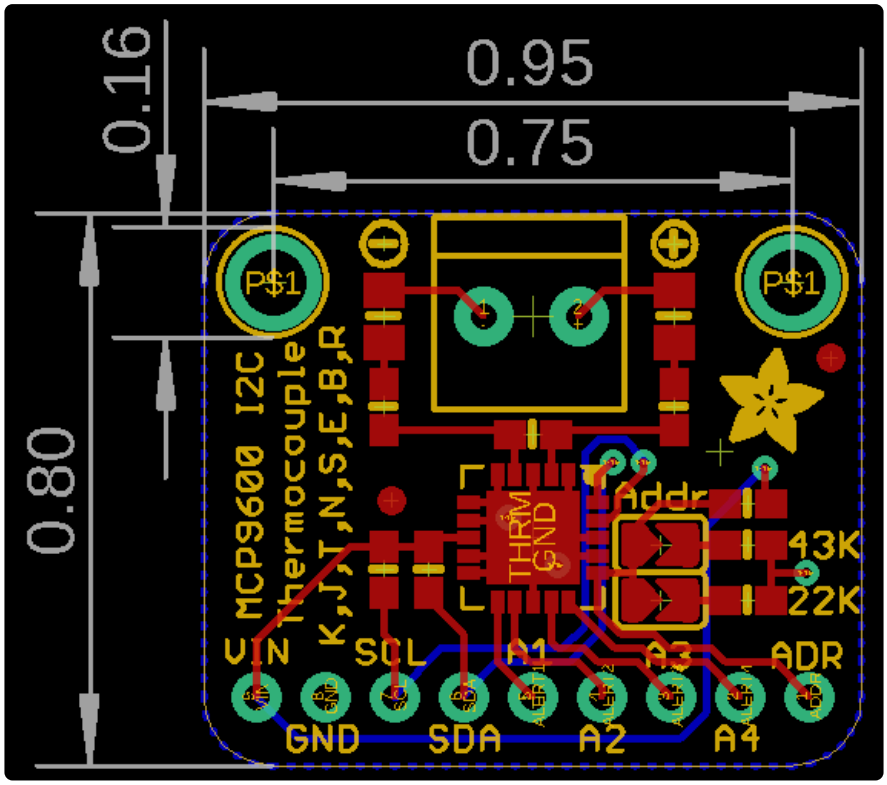
Files

- MCP9600 Datasheet ()
- EagleCAD files on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()

Schematic



Fab Print


0.95
0.75
0.16
0.80
P$1
P$1
MCP9600 I2C
Thermocouple
K,J,T,N,S,E,B,R
THRM
GND
Addr
43K
22K
VIN
SCL
A1
A3
ADR
GND
SDA
A2
A4