# Table of Contents

## Overview

## Pinouts
- Power Pins
- I2C Logic Pins
- Other Pins

## Python & CircuitPython
- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of MCP9600 Library
- CircuitPython Installation of MCP9600 Library
- CircuitPython and Python Usage
- Alerts and More

## Python Docs

## Arduino
- Wiring
- Library Installation
- Load Example
- Example Code

## Arduino Docs

## Downloads
- Schematic and Fab Print

©Adafruit Industries
Overview

Thermocouples are very sensitive, requiring a good amplifier with a cold-compensation reference. The Adafruit MCP9601 I2C Thermocouple Breakout (a.k.a MCP96L01) 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 types 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). Match it up with our 1m K-type thermocouple (http://adafruit.it/270).

- 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 this is just resolution of the ADC, not accuracy
- We use the MCP96L01 in this breakout which has ±2.0°C/±4.0°C (typ./max.) thermocouple accuracy (which is not including the innate inaccuracy of thermocouples, K thermocouples have about ±2°C to ±6°C accuracy)
- Internal temperature reading ('cold junction')
- 3.3 to 5v power supply and logic level compliant
- I2C data connection
Additionally, since it speaks I2C, you can easily connect it up with two wires (plus power and ground!). We've even included SparkFun qwiic (compatible STEMMA QT) connectors for the I2C bus so you don't even need to solder! Use a plug-and-play STEMMA QT cable to get temperature data ASAP. QT Cable is not included, but we have a variety in the shop.

Comes assembled and tested. Some soldering is required to attach the terminal block or header to the breakout PCB. The header is only required if you want to use it in a breadboard.
Pinouts

Power Pins

- VIN - This is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 3V microcontroller like a Feather M4, use 3V, or for a 5V microcontroller like Arduino, use 5V.
- GND - This is common ground for power and logic.

I2C Logic Pins

The default I2C address for the MCP9601 is 0x67.

- SCL - I2C clock pin, connect to your microcontroller I2C clock line. There’s a 10K pullup on this pin.
- SDA - I2C data pin, connect to your microcontroller I2C data line. There’s a 10K pullup on this pin.
- STEMMA QT () - These connectors allow you to connectors to development boards with STEMMA QT connectors or to other things with various associated accessories ().
Other Pins

- A1, A2, A3, A4 - These are the alert pins. They are user-programmable push-pull outputs which can be used to detect rising or falling temperatures. The device outputs signal when the ambient temperature exceeds the user-programmed temperature alert limit.
- ADDR - The MCP9601 supports 8 addresses, ranging from 0x60 to 0x67. The default is 0x67, with the ADDR pin not connected to anything. To get 0x60, connect the ADDR pin directly to GND. To get the addresses between 0x60 and 0x67, you must add a resistor of a given value between ADDR and GND - check out Table 6-2 in the datasheet for a list of suggested resistor values for each address.
- Addr jumpers - There are two jumpers on the back, labeled 43k and 22k, each of which change the I2C address. If you solder the 43k jumper closed, the address will be 0x66. If you solder the 22k jumper closed, the address will be 0x65.

Python & CircuitPython

It's easy to use the MCP9601 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.

CircuitPython Microcontroller Wiring

First wire up a MCP9601 to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C:
Board 3V to sensor VIN (red wire)
Board GND to sensor GND (black wire)
Board SCL to sensor SCL (yellow wire)
Board SDA to sensor SDA (blue wire)

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:
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 Installation of MCP9600 Library

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

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, open the folder that matches your CircuitPython version, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

```python
# 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)
```

The MCP9601 does not like zero-length writes directed to it, and will often not respond (it will NAK instead of ACK). This means it will often not respond to probes by `i2c.scan()` and similar scans of the I2C bus.

If you're running into other issues with your MCP9601 connecting to your non-SAMD microcontroller, consider changing the frequency to 20000 as a possible fix. SAMD-based microcontrollers MUST use 100kHz frequency or higher.

Ensure the following file and folders are in the lib folder on your CIRCUITPY drive:

- `adafruit_mcp9600.mpy`
- `adafruit_bus_device/`
- `adafruit_register/`

Before continuing make sure your board's lib folder or root filesystem has the adafruit_mcp9600.mpy, adafruit_bus_device and adafruit_register/ file and folders copied over.
Next connect to the board's serial REPL so you are at the CircuitPython >>> prompt.

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 MCP9601 to work:

```python
import board
import busio
import adafruit_mcp9600

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

If you find that your MCP9601 isn't connecting to your non-SAMD microcontroller (SAMD-based microcontrollers MUST use 100kHz frequency or higher!), consider changing the I2C initialisation to the following as a possible fix:

```python
i2c = busio.I2C(board.SCL, board.SDA, frequency=20000)
```

Now that you've done imports and setup, 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 change in temperature.
Enter the following line of code into the REPL.

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

An extended example is available under the CircuitPython Installation section. That's all there is to reading temperature using the MCP9601 and CircuitPython!

### Alerts and More

The MCP9601 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](#)

### Arduino

### Wiring

Connecting the MCP9600 to your Feather or Arduino is easy:
If you are running a 5V Arduino (Uno, etc.), connect Arduino 5V to MCP9601 VIN (red wire). If you are running a Feather (3.3V), connect Feather 3V to MCP9601 VIN. Connect Feather or Arduino GND to MCP9601 GND (black wire) Connect Feather or Arduino SCL to MCP9601 SCL (yellow wire) Connect Feather or Arduino SDA to MCP9601 SDA (blue wire) Connect thermocouple + to MCP9601 screw terminal + Connect thermocouple - to MCP9601 screw terminal -

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

The MCP9601 may return a temperature for the hot junction even if there is no thermocouple connected. There will not be an error!

Library Installation

The MCP9601 uses the MCP9600 library.

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 -> mcp9601_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 MCP9601 for the hot junction, cold junction and ADC values:

```cpp
#include "Adafruit_MCP9601.h"
```
```c
#define I2C_ADDRESS (0x67)

Adafruit_MCP9601 mcp;

void setup()
{
    Serial.begin(115200);
    while (!Serial) {
        delay(10);
    }
    Serial.println("Adafruit MCP9601 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 MCP9601!");

    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()
{
    uint8_t status = mcp.getStatus();
    Serial.print("MCP Status: 0x");
    Serial.print(status, HEX);
    Serial.print(":");
    if (status & MCP9601_STATUS_OPENCIRCUIT) {
        Serial.println("Thermocouple open!");
        return; // don't continue, since there's no thermocouple
    }
    if (status & MCP9601_STATUS_SHORTCIRCUIT) {
```
Serial.println("Thermocouple shorted to ground!");
return; // don't continue, since the sensor is not working
}
if (status & MCP960X_STATUS_ALERT1) { Serial.print("Alert 1, "); }
if (status & MCP960X_STATUS_ALERT2) { Serial.print("Alert 2, "); }
if (status & MCP960X_STATUS_ALERT3) { Serial.print("Alert 3, "); }
if (status & MCP960X_STATUS_ALERT4) { Serial.print("Alert 4, "); }
Serial.println();

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:

![Serial Monitor Output]
Schematic and Fab Print