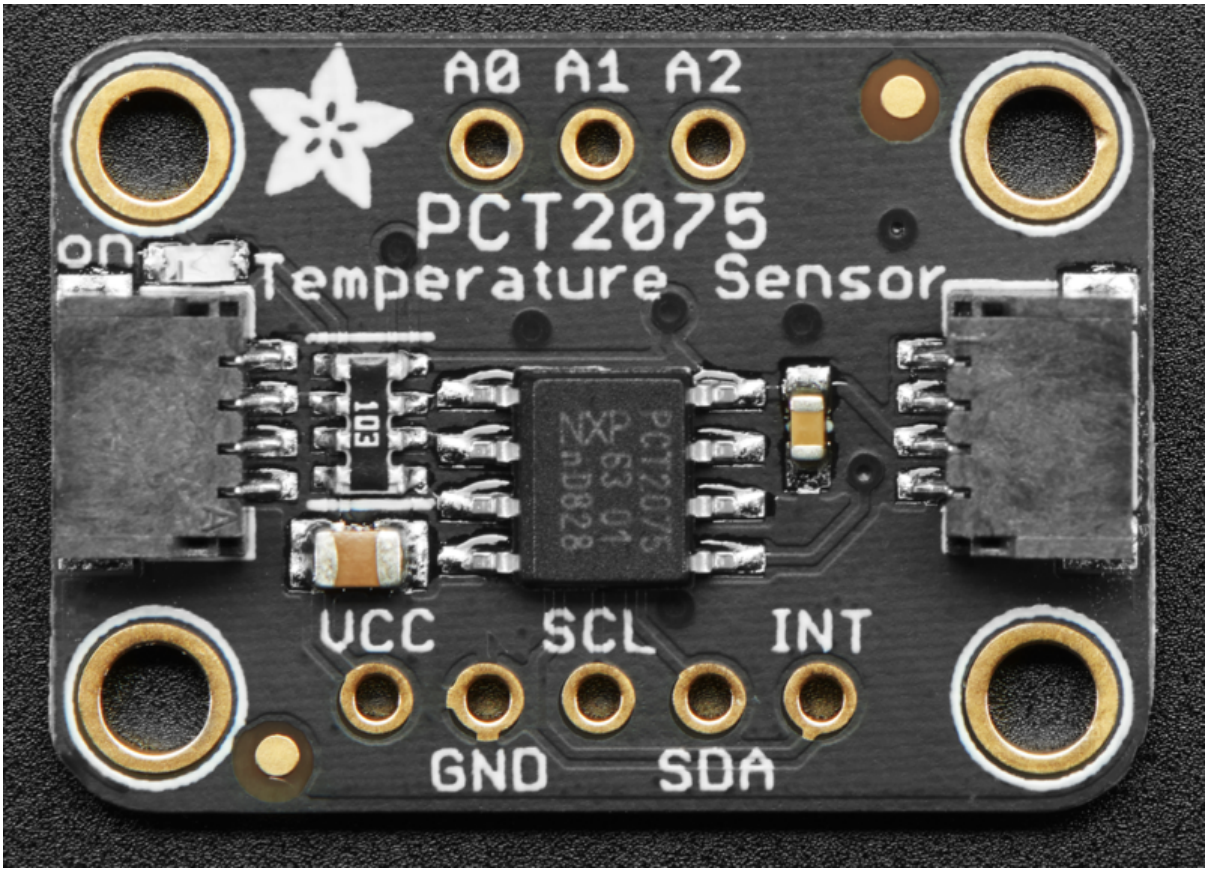
Adafruit PCT2075 Temperature Sensor

Created by Abigail Torres



https://learn.adafruit.com/adafruit-pct2075-temperature-sensor

Last updated on 2024-03-08 03:35:55 PM EST

Table of Contents

Overview 3

Pinouts 5

- Power Pins
- I2C Logic Pins
- Other Pins
- I2C Address Selection

Arduino 7

- Wiring
- Library Installation
- Basic Temperature Example
- Temperature Alert Example
- Temperature Alert Settings
- Handy Interstitial Diagram
- Alert Modes
- Activating the Temperature Alert
- Temperature Example Code
- Temperature Alert Example Code

Arduino Docs 13

Python & CircuitPython 13

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- CircuitPython Installation of PCT2075 Library
- Python Installation of PCT2075 Library
- CircuitPython & Python Usage
- Temperature Alert Example
- CircuitPython Wiring
- Python Computer Wiring
- Temperature Alert Settings
- Handy Interstitial Diagram
- Alert Modes
- Testing the Code
- Is it hot in here?
- Temperature Example Code
- Temperature Alert Example

Python Docs 22

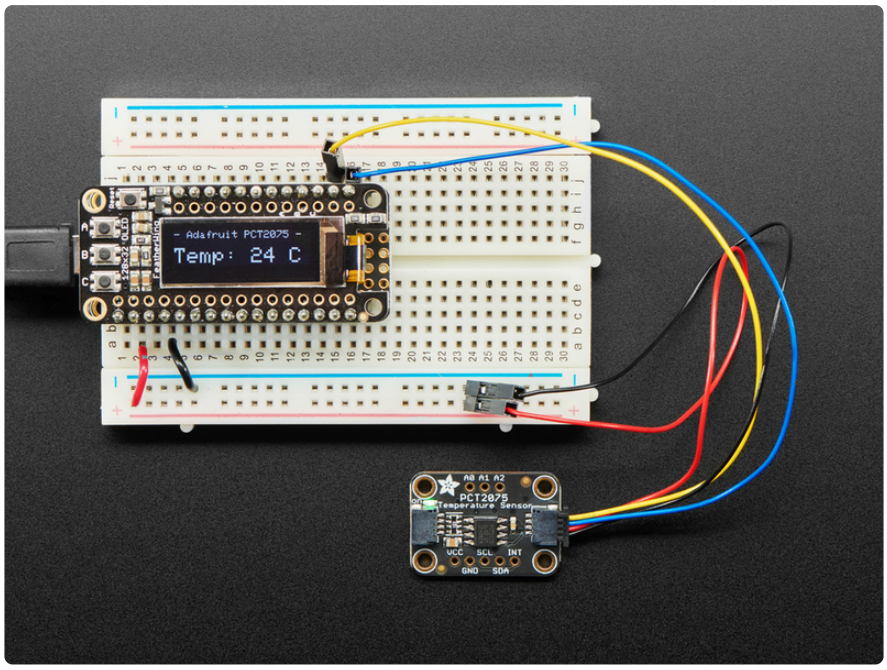
WipperSnapper 22

- What is WipperSnapper
- Wiring
- Usage

Downloads 29

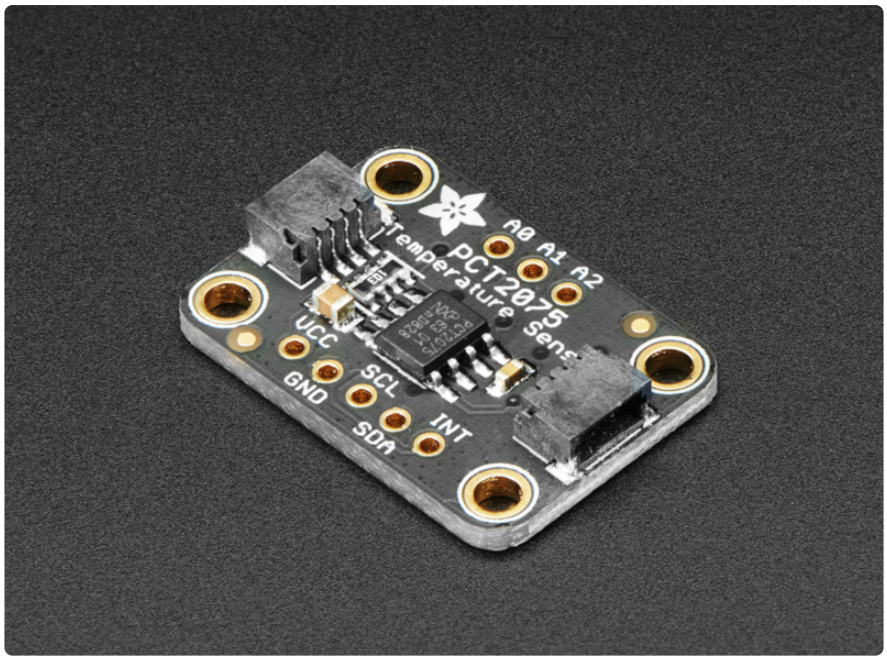
- Files
- Schematic
- Fab Print

Overview



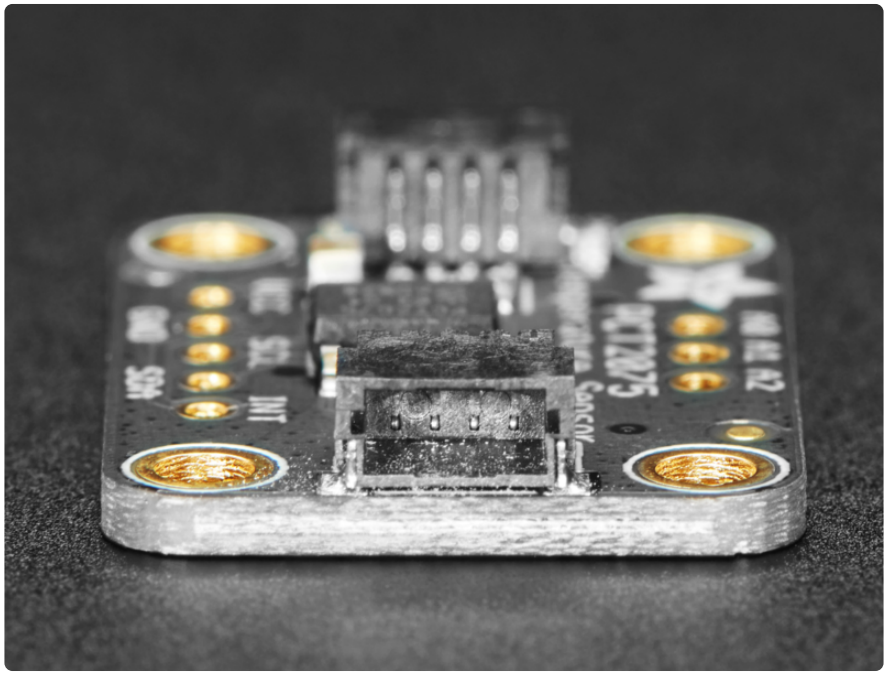
The PCT2075 by NXP is a pin compatible drop in replacement for a very common I2C temperature sensor, the LM75. Compared to the LM75, however, the 11-bit ADC in the PCT2075 provides more precise measurements when compared to the LM75's 9-bit ADC. Additionally because the PCT2075 allows the address pins to work in three states (high, low, floating), you can have 27 PCT2075s on the same bus as opposed to only 8 for the LM75. Now you can finally measure the individual temperatures of the tentacles of **three octopuses** instead of just one!

Otherwise the two sensors are the same. The PCT2075 will report temperature and allow you to a set a high temperature threshold that the sensor will compare to the current temperature and raise an alert when the current temperature exceeds the threshold. There are also a few (metaphorical) knobs to twist to change the alerting and measurement behavior.



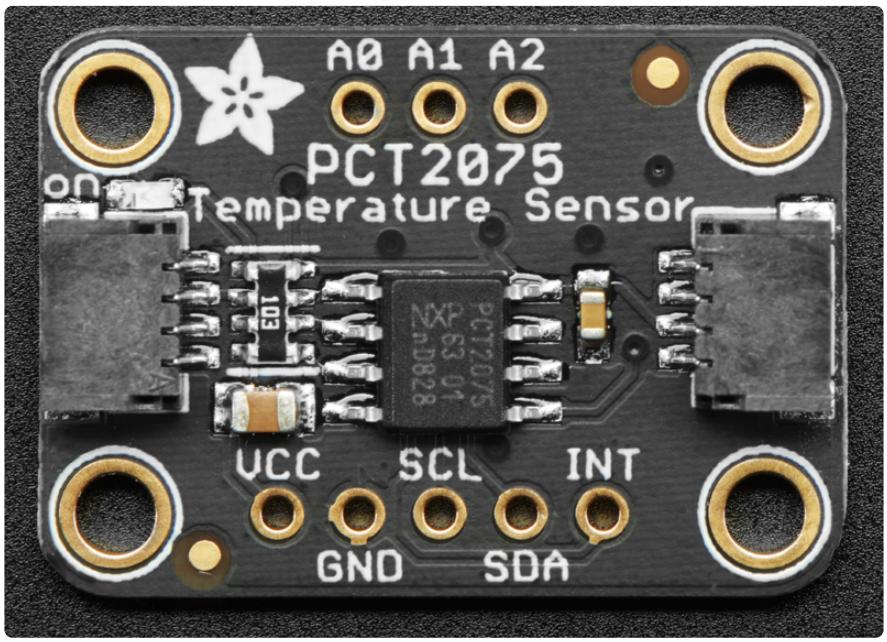
As with all Adafruit breakouts, we've done the work to make this handy temperature helper super easy to use. We've put it on a breakout board with the required support circuitry and connectors to make it easy to work with, and is now a trend we've added SparkFun Qwiic (https://adafru.it/Fpw) compatible **STEMMA QT** (https://adafru.it/Ft4) JST SH connectors that allow you to get going **without needing to solder.** Just use a STEMMA QT adapter cable (http://adafru.it/4209), plug it into your favorite micro or Blinka supported SBC and you're ready to rock!

"OK" you say "connections are one thing but how do I tell it what to do?" (Not much of a) Surprise! We've written libraries for Arduino and CircuitPython with examples that make it super easy to integrate the PCT2075 with your project.



The PCT2075 is a handy and inexpensive temperature sensor that does one thing well. Try adding one to your project and have it keep an eye on the temperature and let you know when things get hot.

Pinouts



Power Pins

The sensor on the breakout requires between a 2.7V and 5.5V, and can be easily used with most microcontrollers from an Arduino to a Feather or something else.

- **Vcc** - 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 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
- STEMMA QT **(https://adafru.it/Ft4) -** These connectors allow you to connectors to dev boards with **STEMMA QT** connectors or to other things with various associated accessories (https://adafru.it/Ft6)

Other Pins

- **INT** -This is the interrupt pin. You can setup the PCT2075 to activate this pin when a temperature threshold is reached or exceeded.
- **A0, A1, A2** - I2C Address pins. See the "I2C Address Selection" section below

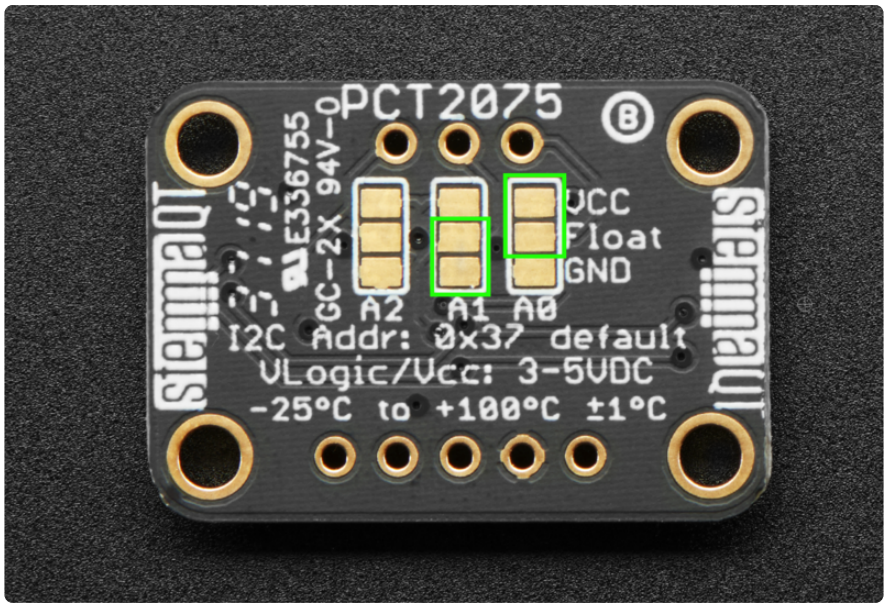
I2C Address Selection

One of the improvements that the PCT2075 has over the LM75 that it can replace is the ability to have three times the number of I2C addresses. Interestingly, they achieve this by allowing the **three** address pins **A0, A1,** and **A2** be in one of **three states, low (0), high(1), or floating(F).** This means that the are 3^3 **(27)** possible addresses. Attach the address pin to ground for low, the logic level of your device for hight, or leave the pin unconnected for floating.

We leave all three address pins floating on the breakout for maximum configurability. This means that the **default address** for the breakout **is 0x37.**

Pins			Address		Pins			Address		Pins			Address	
A2	A1	A0	Binary	Hex	A2	A1	A0	Binary	Hex	A2	A1	A0	Binary	Hex
0	0	0	0b1001000	0x48	F	0	F	0b1110001	0x71	1	F	0	0b0101010	0x2a
0	0	1	0b1001001	0x49	F	0	1	0b1110010	0x72	1	F	1	0b0101011	0x2b
0	1	0	0b1001010	0x4a	F	1	0	0b1110011	0x73	0	0	F	0b0101100	0x2c
0	1	1	0b1001011	0x4b	F	1	F	0b1110100	0x74	0	1	F	0b0101101	0x2d
1	0	0	0b1001100	0x4c	F	1	1	0b1110101	0x75	1	0	F	0b0101110	0x2e
1	0	1	0b1001101	0x4d	F	F	0	0b1110110	0x76	1	1	F	0b0101111	0x2f
1	1	0	0b1001110	0x4e	F	F	1	0b1110111	0x77	0	F	F	0b0110101	0x35
1	1	1	0b1001111	0x4f	0	F	0	0b0101000	0x28	1	F	F	0b0110110	0x36
F	0	0	0b1110000	0x70	0	F	1	0b0101001	0x29	F	F	F	0b0110111	0x37

For a more permanent address selection, you can use the solder jumpers on the back of the breakout board to connect the address pins to GND or VCC.

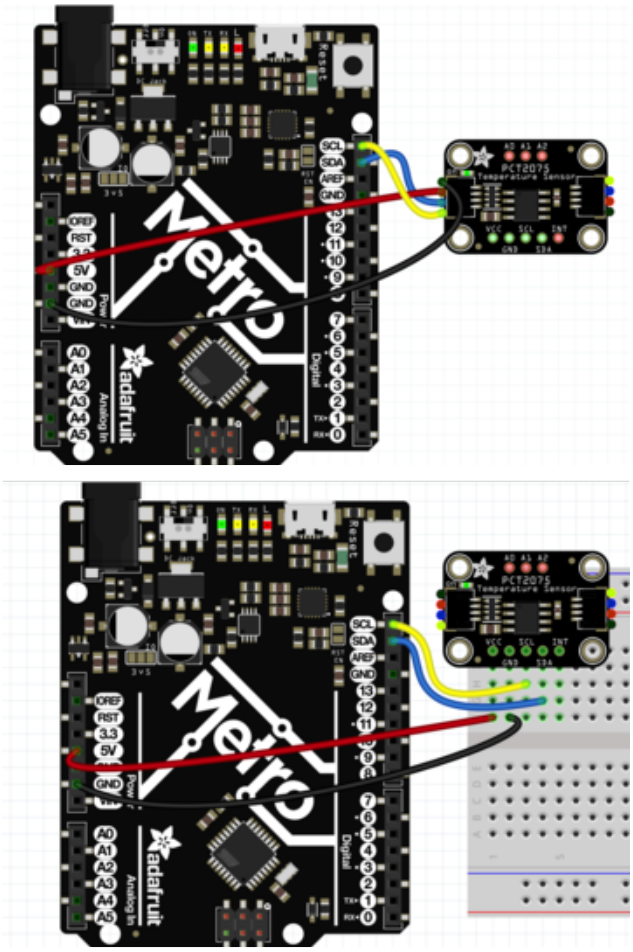


All you need to do is bridge the "Float" pad to either the VCC or GND pad for one or more of the address pins. In the picture above if you bridged the solder pads that are circled, you would pull A0 high to VCC and A1 low to GND. Referring to the table above this would set the address of the breakout to **0x72**

Arduino

Wiring

Wiring the PCT2075 to communicate with your microcontroller is straight forward thanks to the I2C interface. For these examples we can use the Metro or Arduino to take temperature measurements. The instructions below show a Metro (http://adafru.it/2488), but the same applies to an Arduino

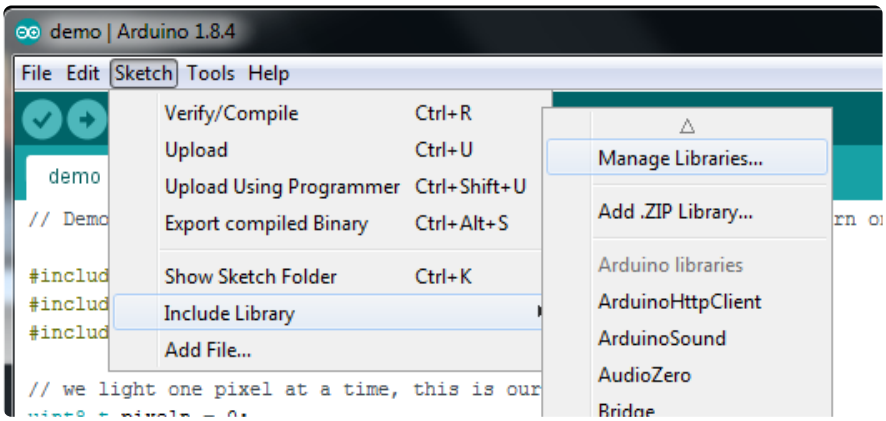


Connect **board VCC (red wire)** to **Arduino 5V** if you are running a **5V** board Arduino (Uno, etc.). If your board is **3V,** connect to that instead.
Connect **board GND (black wire)** to **Arduino GND**
Connect **board SCL (yellow wire)** to **Arduino SCL**
Connect **board SDA (blue wire)** to **Arduino SDA**

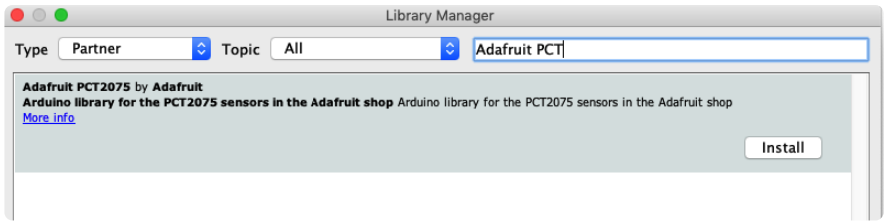
Library Installation

Once wired up, to start using the PCT2075 you'll need to install the Adafruit_PCT2075 library (https://adafru.it/FTR). The library is available through the Arduino library manager so we recommend taking that approach.

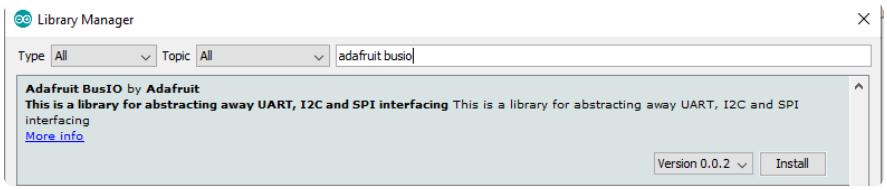
From the Arduino IDE, open up the Library Manager:



Click the **Manage Libraries ...** menu item, search for **Adafruit PCT2075**, and select the **Adafruit PCT2075** library and click **Install**:



Then follow the same process for the **Adafruit BusIO** library.



Basic Temperature Example

Open up **File -> Examples -> Adafruit PCT2075 -> pct2075_test** and upload to your Arduino wired up to the sensor.

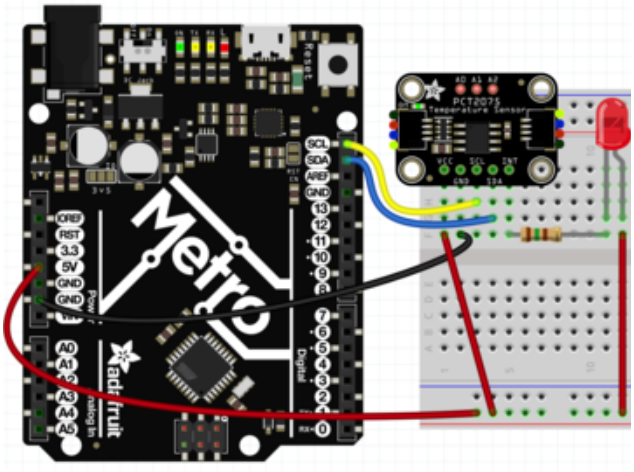
One you've uploaded the sketch to your board open up the Serial Monitor (**Tools->Serial Monitor**) at **115200 baud**. You should see the temperature readings being printed like so:

The temperature will be different depending on the weather where you are. You can see it change by either putting a particularly warm finger on the sensor, or by blowing hot air across it with a hot air station (not too hot!) or a hair dryer.

Temperature Alert Example

Next we'll wire up an LED with a current limiting resistor (https://adafru.it/FTS) to the **INT** pin on the PCT2075 and then ask the PCT2075 to tell us when a certain temperature has been reached.

To start we'll take the wiring for the previous example and add the LED and resistor:



Connect **sensor VCC (red wire)** to **Arduino 5V** if you are running a **5V** board Arduino (Uno, etc.). If your board is **3V,** connect to that instead.
Connect **sensor GND (black wire)** to **Arduino GND**
Connect **sensor SCL (yellow wire)** to **Arduino SCL**
Connect **sensor SDA (blue wire)** to **Arduino SDA**
Connect **sensor INT** to **Resistor leg 1**
Connect **LED Cathode** (https://adafru.it/FTT) to **Resistor leg 2**
LED Anode (https://adafru.it/FTT) to **Arduino 5V**

Open up **File -> Examples -> Adafruit PCT2075 -> temp_alert_demo** 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**) at **115200 baud**. You should see the current value of the various **temperature alert settings** printed out, followed by temperature measurements:

Temperature Alert Settings

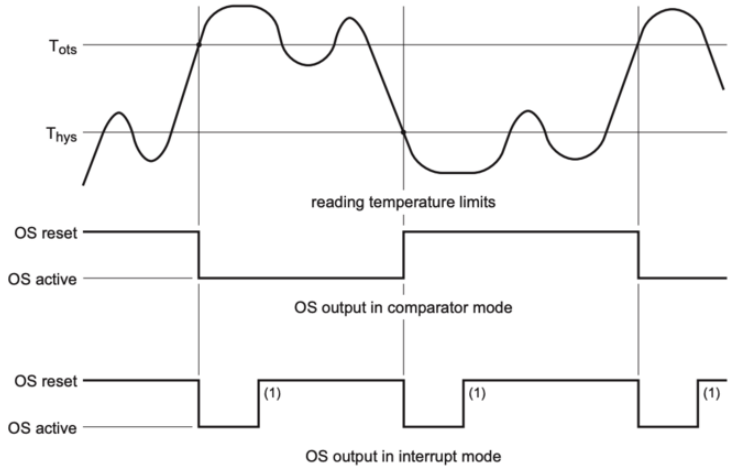
High temperature threshold (https://adafru.it/FN9) is the temperature that the sensor compares to the measured temperature each time it makes a measurement. If the current temperature is greater than or equal to the high temperature threshold, the sensor registers a temperature fault. What happens with that alert depends on the rest of the settings.

Temperature hysteresis (https://adafru.it/FN9) is the temperature that defines the bottom of the "active" range of temperature where the sensor sill considers the temperature fault as being active. This temperature must be **less than the high temperature threshold.**

The **fault count** (https://adafru.it/FN9) is how many consecutive temperature faults are required to raise an alert on the INT pin. In the example code the fault count is set to **4** meaning that four measurements by the sensor in a row must be equal to or greater than the high temperature threshold for the sensor to raise an alert.

Handy Interstitial Diagram

This diagram from the datasheet (https://adafru.it/FTU) shows how the different parameters work together. High temperature threshold (here as $\mathbf{T_{ots}}$) and temperature hysteresis ($\mathbf{T_{hys}}$) together define a temperature range that specify when the **INT** pin (**OS**) responds to, depending on the alert mode.



Alert Modes

The Alert mode **(https://adafru.it/FN9)** dictates how the **INT pin** will be used to communicate with the user about the state of the temperature monitoring.
In **Comparator mode** like in the example, the sensor acts similar to a thermostat controlling an air conditioning unit. When the sensor raises a high temperature alert according to the high temperature threshold and fault count, the **INT** pin is connected to ground, completing the LED circuit and lighting the LED. Once the temperature falls below the **temperature hysteresis**, the sensor puts the **INT** pin in a high impedance state, effectively disconnecting the LED from ground.

With the sensor in **Comparator mode**, the sensor can control a cooling mechanism such as a fan by having the **INT pin** activate a transistor or relay that controls the current flowing to the fan's motor (https://adafru.it/dtG). This way the decision of when to activate the fan is controlled directly by the PCT2075, allowing the microcontroller to focus on other things.

In **Interrupt mode** the sensor activates the **INT pin** twice: once when the alert is first raised, and a second time when the temperature falls below the **temperature hysteresis.** In interrupt mode the **INT pin** is deactivated by reading from the sensor.

Activating the Temperature Alert

Once wired as above, find a way to **safely** warm up the PCT2075 (**definitely NOT with fire)**, and once it is past the threshold temperature of **32.5 degrees C (90.5 degrees F)** for four consecutive measurements**,** the LED will light up until the temperature again falls below the temperature hysteresis.

If you aren't able to get the temperature up to 32.5 degrees, you can modify the temperature threshold and hysteresis so that they're in a range that is manageable.

Temperature Example Code

```
#include <Adafruit_PCT2075.h>


Adafruit_PCT2075 PCT2075;

void setup() {
  PCT2075 = Adafruit_PCT2075();

  Serial.begin(115200);
  // Wait until serial port is opened
  while (!Serial) { delay(1); }
  Serial.println("Adafruit PCT2075 Test");

  if (!PCT2075.begin()) {
    Serial.println("Couldn't find PCT2075 chip");
    while (1);
  }

  Serial.println("Found PCT2075 chip");

}

void loop() {
  Serial.print("Temperature: ");
Serial.print(PCT2075.getTemperature());Serial.println(" C");
  delay(1000);
}
```

Temperature Alert Example Code

```
#include <Adafruit_PCT2075.h>


Adafruit_PCT2075 pct;

void setup() {
  pct = Adafruit_PCT2075();
```

```
  Serial.begin(115200);
  // Wait until serial port is opened
  while (!Serial) { delay(1); }
  Serial.println("Adafruit PCT2075 Test");

  if (!pct.begin()) {
    Serial.println("Couldn't find PCT2075 chip");
    while (1);
  }

  Serial.println("Found PCT2075 chip");
  pct.setIdleTime(1.0);
  pct.setActiveHigh(true);

  pct.setHighTemperatureThreshold(32.5);
  Serial.print("High temperature threshold:
");Serial.print(pct.getHighTemperatureThreshold()); Serial.println("");

  pct.setTemperatureHysteresis(30.5);
  Serial.print("Temperature hysteresis:
");Serial.print(pct.getTemperatureHysteresis()); Serial.println("");

  pct.setMode(PCT2075_MODE_COMPARITOR);
  Serial.print("Alert mode set to: ");
  switch (pct.getMode()) {
    case PCT2075_MODE_INTERRUPT: Serial.println("Interrupt"); break;
    case PCT2075_MODE_COMPARITOR: Serial.println("Comparitor"); break;
  }

  pct.setFaultCount(PCT2075_FAULT_COUNT_4); // since the loop timing and idle delay
are in sync, it will take 4 loops to fault
  Serial.print("Fault count set to: ");
  switch (pct.getFaultCount()) {
    case PCT2075_FAULT_COUNT_1: Serial.println("1"); break;
    case PCT2075_FAULT_COUNT_2: Serial.println("2"); break;
    case PCT2075_FAULT_COUNT_4: Serial.println("4"); break;
    case PCT2075_FAULT_COUNT_6: Serial.println("6"); break;
  }

}

void loop() {
  // checking every
  Serial.print("Temperature: ");
Serial.print(pct.getTemperature());Serial.println(" C");
  delay(1000); // wait one second to match the idle time
}
```

Arduino Docs

Arduino Docs (https://adafru.it/FN9)

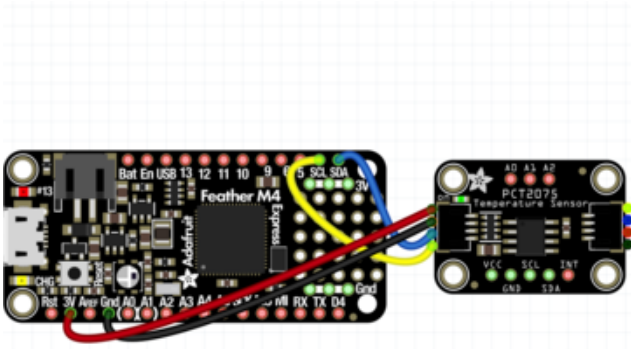
Python & CircuitPython

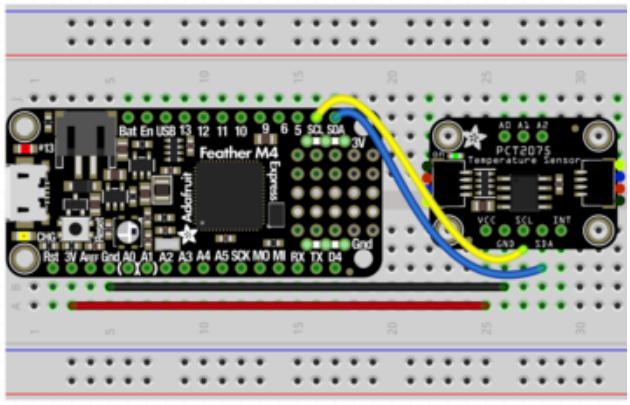
It's easy to use the PCT2075 temperature sensor with CircuitPython and the Adafruit CircuitPython PCT2075 (https://adafru.it/FTV) module. This module allows you to easily write Python code that reads the temperature and set an alert to go off when a given temperature is reached or exceeded.

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

CircuitPython Microcontroller Wiring

First you'll need to wire up the sensor for a basic I2C connection. The STEMMA QT connectors give you flexibility with how you choose to make the connections, but the required connections are fairly simple:





Board 3V to **sensor VCC (red wire)**
Board GND to **sensor GND (black wire)**
Board SCL to **sensor SCL (yellow wire)**
Board SDA to **sensor SDA (blue wire)**

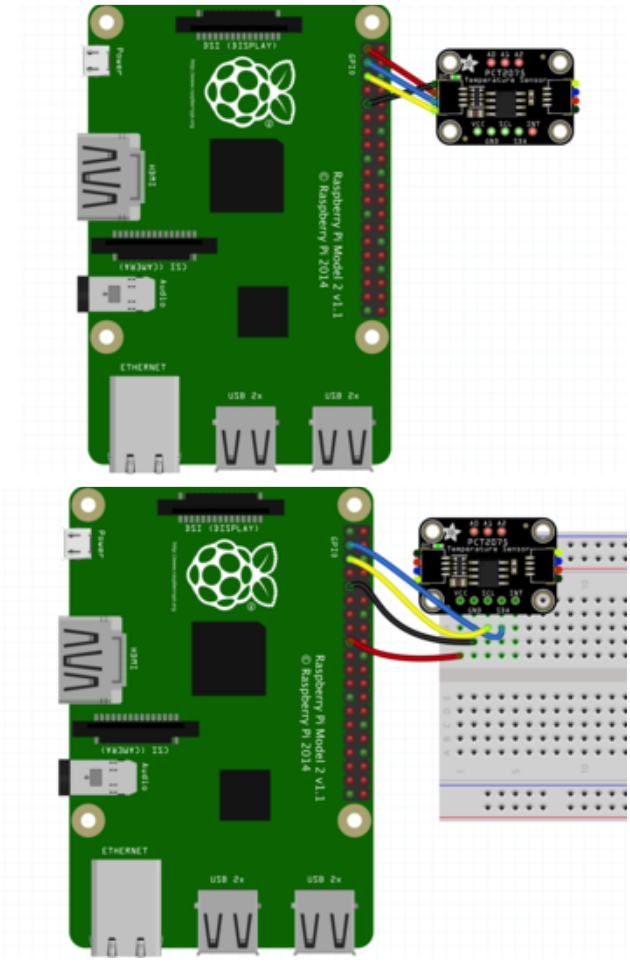
Note: This breakout includes pullup resistors on the I2C lines, no external pullups are required.

Python Computer Wiring

Since there are dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi (http://adafru.it/3055). 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 3V to **sensor VCC (red wire)**
Pi GND to **sensor GND (black wire)**
Pi SCL to **sensor SCL (yellow wire)**
Pi SDA to **sensor SDA (blue wire)**

CircuitPython Installation of PCT2075 Library

You'll need to install the Adafruit CircuitPython PCT2075 (https://adafru.it/FTW) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) 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/uap). Our CircuitPython starter guide has a great page on how to install the library bundle (https://adafru.it/ABU).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- **adafruit_pct2075.mpy**
- **adafruit_bus_device**
- **adafruit_register**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_pct2075.mpy, adafruit_bus_device**, and **adafruit_register** files and folders copied over.

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

Python Installation of PCT2075 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-pct2075`

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 measurements from the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import time
import board
import adafruit_pct2075
i2c = board.I2C()
```

```
pct = adafruit_pct2075.PCT2075(i2c)
```

```
>>> import time
>>> import board
>>> import adafruit_pct2075
>>> i2c = board.I2C()
>>> pct = adafruit_pct2075.PCT2075(i2c)
```

Now you're ready to read values from the sensor using the **temperature** property:

```
>>> print("Temperature: %.2f C"%pct.temperature)
Temperature: 26.88 C
>>>
```

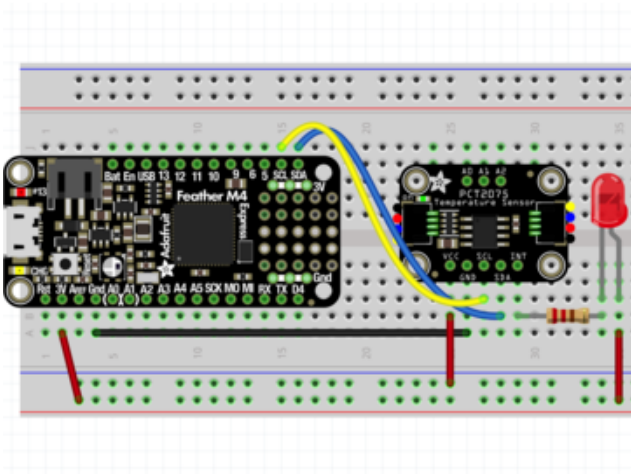
Temperature Alert Example

This next example will show you how to use the temperature alerting functions to let you know when the measured temperature gets higher than a configurable point.

First, we will have to wire up an LED and current limiting resistor that the sensor will use to alert us about the temperature. This requires just a few more connections than the basic temperature functionality. If you were using the STEMMA QT connector without a breadboard, you'll need to add one so that you have a place to plug in the LED and resistor.

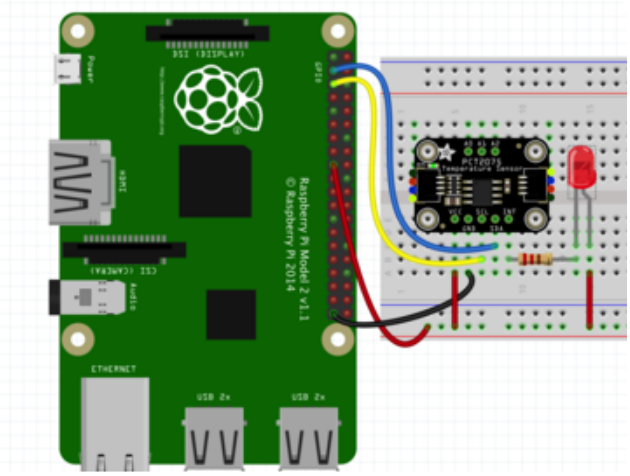
Wire up the sensor according to the diagram for your type of platform below:

CircuitPython Wiring



Connect **sensor VCC (red wire)** to **Feather 3V**.
Connect **sensor GND (black wire)** to **Feather GND**
Connect **sensor SCL (yellow wire)** to **Feather SCL**
Connect **sensor SDA (blue wire)** to **Feather SDA**
Connect **sensor INT** to **Resistor leg 1**
Connect **LED Cathode** (https://adafru.it/FTT) to **Resistor leg 2**
LED Anode (https://adafru.it/FTT) to **Feather 5V**

Python Computer Wiring



Connect **sensor VCC (red wire)** to **Feather 3V**.
Connect **sensor GND (black wire)** to **Feather GND**
Connect **sensor SCL (yellow wire)** to **Feather SCL**
Connect **sensor SDA (blue wire)** to **Feather SDA**
Connect **sensor INT** to **Resistor leg 1**
Connect **LED Cathode** (https://adafru.it/FTT) to **Resistor leg 2**
LED Anode (https://adafru.it/FTT) to **Feather 5V**

Temperature Alert Settings

High temperature threshold (https://adafru.it/FTX) is the temperature that the sensor compares to the measured temperature each time it makes a measurement. If the current temperature is greater than or equal to the high temperature threshold, the sensor registers a temperature fault. What happens with that alert depends on the rest of the settings.

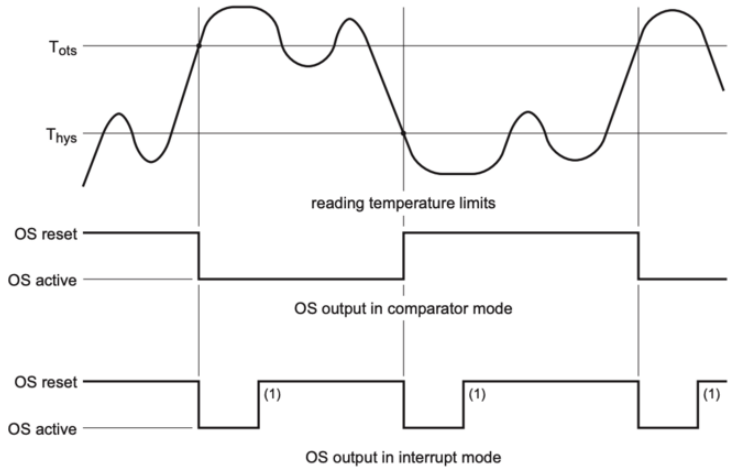
Temperature hysteresis (https://adafru.it/FTX) is the temperature that defines the bottom of the "active" range of temperature where the sensor sill considers the temperature fault as being active. This temperature must be **less than the high temperature threshold.**

The **fault count** (https://adafru.it/FTX)is how many consecutive temperature faults are required to raise an alert on the INT pin. In the example code the fault count is set to **4** meaning that four measurements by the sensor in a row must be equal to or greater than the high temperature threshold for the sensor to raise an alert.

Handy Interstitial Diagram

This diagram from the datasheet (https://adafru.it/FTU) shows how the different parameters work together. High temperature threshold (here as T_{ots}) and

temperature hysteresis (T_{hys}) together define a temperature range that specify when the **INT** pin (**OS**) responds to, depending on the alert mode.



Alert Modes

The Alert mode **(https://adafru.it/FTX)** dictates how the **INT pin** will be used to communicate with the user about the state of the temperature monitoring.
In **Comparator mode** like in the example, the sensor acts similar to a thermostat controlling an air conditioning unit. When the sensor raises a high temperature alert according to the high temperature threshold and fault count, the **INT** pin is connected to ground, completing the LED circuit and lighting the LED. Once the temperature falls below the **temperature hysteresis**, the sensor puts the **INT** pin in a high impedance state, effectively disconnecting the LED from ground.

With the sensor in **Comparator mode**, the sensor can control a cooling mechanism such as a fan by having the **INT pin** activate a transistor or relay that controls the current flowing to the fan's motor (https://adafru.it/dtG). This way the decision of when to activate the fan is controlled directly by the PCT2075, allowing the microcontroller to focus on other things.

In **Interrupt mode** the sensor activates the **INT pin** twice: once when the alert is first raised, and a second time when the temperature falls below the **temperature hysteresis.** In interrupt mode the **INT pin** is deactivated by reading from the sensor.

Testing the Code

As in the previous example we start by making the required imports, setting up an I2C bus and creating the `pct2075.PCT2075` class instance.

```
import time
import board
import adafruit_pct2075

i2c = board.I2C()
pct = adafruit_pct2075.PCT2075(i2c)
```

```
>>> import time
>>> import board
>>> import adafruit_pct2075
>>> i2c = board.I2C()
>>> pct = adafruit_pct2075.PCT2075(i2c)
```

Next we can set and verify the temperature threshold and hysteresis

```
pct.high_temperature_threshold = 35.5
print("High temperature threshold: %.2f"%pct.high_temperature_threshold)

pct.temperature_hysteresis = 30.0
print("Temperature hysteresis: %.2f"%pct.temperature_hysteresis)
```

```
>>> pct.high_temperature_threshold = 35.5
>>> print("High temperature threshold: %.2f"%pct.high_temperature_threshold)
High temperature threshold: 35.50
>>> pct.temperature_hysteresis = 30.0
>>> print("Temperature hysteresis: %.2f"%pct.temperature_hysteresis)
Temperature hysteresis: 30.00
```

Finally we set the number of consecutive temperature faults required to raise an alert.

```
pct.faults_to_alert = adafruit_pct2075.FaultCount.FAULT_4
if pct.faults_to_alert is adafruit_pct2075.FaultCount.FAULT_4:
    print("Faults to alert: 4")
```

```
>>> pct.faults_to_alert = adafruit_pct2075.FaultCount.FAULT_4
>>> if pct.faults_to_alert is adafruit_pct2075.FaultCount.FAULT_4:
...     print("Faults to alert: 4")
...
Faults to alert: 4
```

Finally we start taking temperature measurements in a loop. While it runs find a **safe method** of gently warming the sensor such as a hair dryer or chilled with warm breath and a longer than average attention span **(definitely not fire).** Once it has been warmed to or above **35.5 C (90.5 F)** for 4 measurement cycles the LED will light up

```
while True:
    print("Temperature: %.2f C"%pct.temperature)
    time.sleep(0.5)
```

```
Temperature: 32.00 C
Temperature: 32.75 C
Temperature: 33.13 C
Temperature: 33.62 C
Temperature: 34.00 C
Temperature: 34.38 C
Temperature: 34.75 C
Temperature: 35.25 C
```

Is it hot in here?

That's all it takes! The PCT2075 is a versatile temperature sensor that you'll find all sorts of uses for.

Don't wonder how hot it is, ask the PCT2075!

Temperature Example Code

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

import time
import board
import adafruit_pct2075

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
pct = adafruit_pct2075.PCT2075(i2c)

pct.high_temperature_threshold = 35.5
pct.temperature_hysteresis = 30.0
pct.high_temp_active_high = False
print("High temp alert active high? %s" % pct.high_temp_active_high)

# Attach an LED with the Cathode to the INT pin and Anode to 3.3V with a current
limiting resistor

while True:
    print("Temperature: %.2f C" % pct.temperature)
    time.sleep(0.5)
```

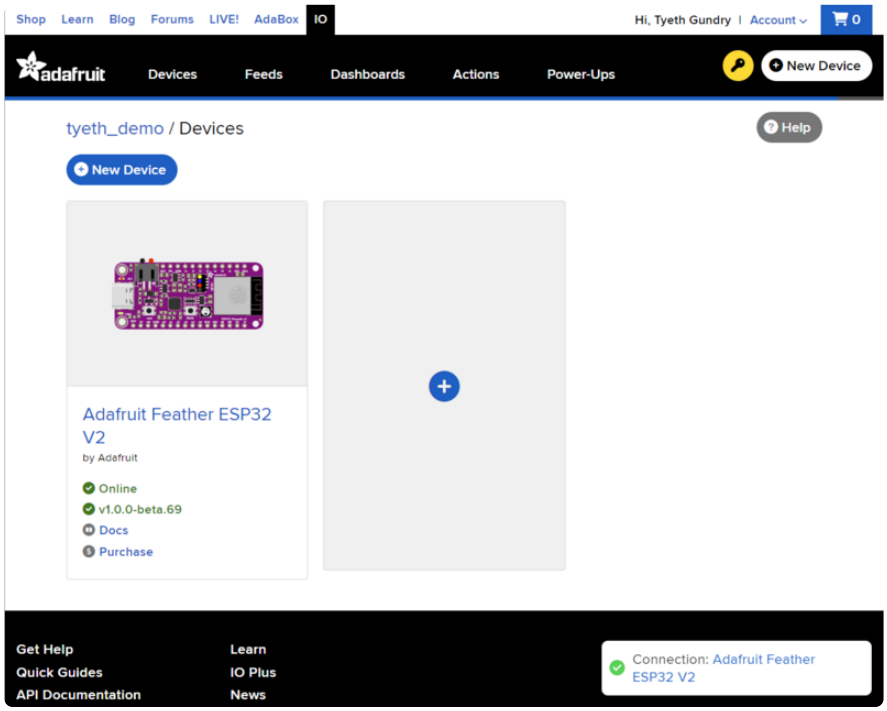
Temperature Alert Example

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

import time
import board
import adafruit_pct2075

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
pct = adafruit_pct2075.PCT2075(i2c)

while True:
    print("Temperature: %.2f C" % pct.temperature)
    time.sleep(0.5)
```

Python Docs

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

WipperSnapper



What is WipperSnapper

WipperSnapper is a firmware designed to turn any WiFi-capable board into an Internet-of-Things device without programming a single line of code. WipperSnapper connects to Adafruit IO (https://adafru.it/fsU), a web platform designed (by Adafruit! (https://adafru.it/Bo5)) to display, respond, and interact with your project's data.

Simply load the WipperSnapper firmware onto your board, add credentials, and plug it into power. Your board will automatically register itself with your Adafruit IO account.

From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

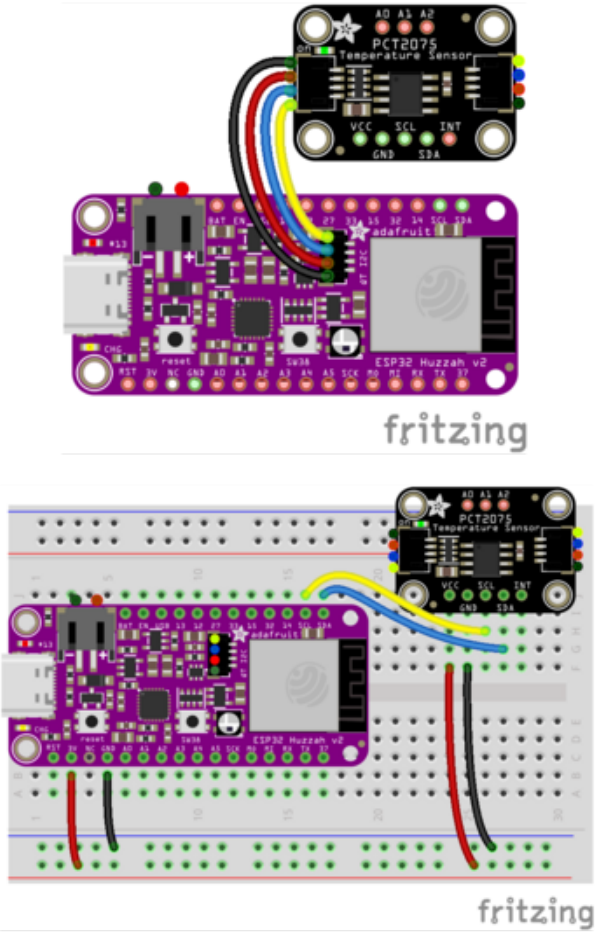
If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

Quickstart: Adafruit IO WipperSnapper

https://adafru.it/Vfd

Wiring

First, wire up a PCT2075 to your board exactly as follows. Here is an example of the PCT2075 wired to an Adafruit ESP32 Feather V2 (http://adafru.it/5400) using I2C with a STEMMA QT cable (no soldering required) (http://adafru.it/4210)

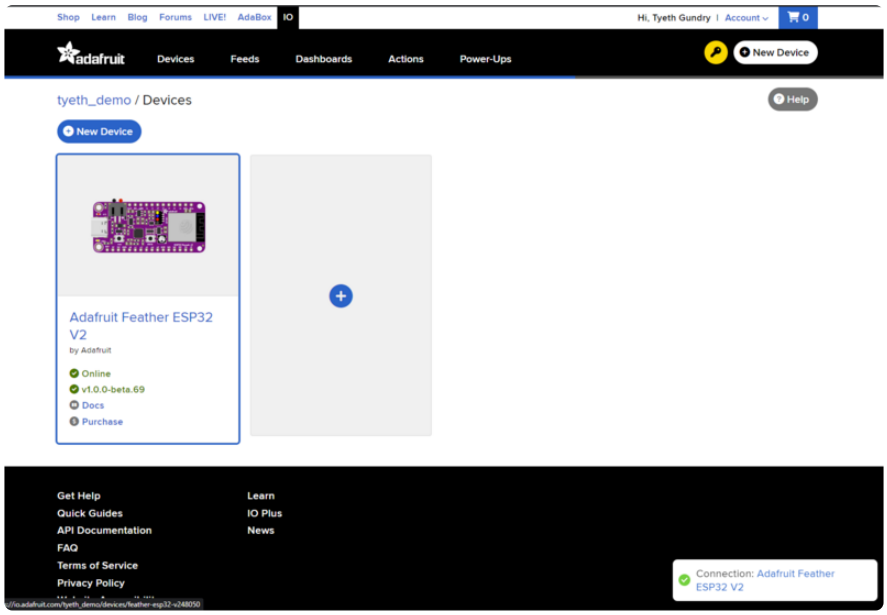


Board 3V to **sensor VIN (red wire on STEMMA QT)**
Board GND to **sensor GND (black wire on STEMMA QT)**
Board SCL to **sensor SCL (yellow wire on STEMMA QT)**
Board SDA to **sensor SDA (blue wire on STEMMA QT)**

Usage

Connect your board to Adafruit IO Wippersnapper and navigate to the WipperSnapper board list **(https://adafru.it/TAu)**.

On this page, **select the WipperSnapper board you're using** to be brought to the board's interface page.



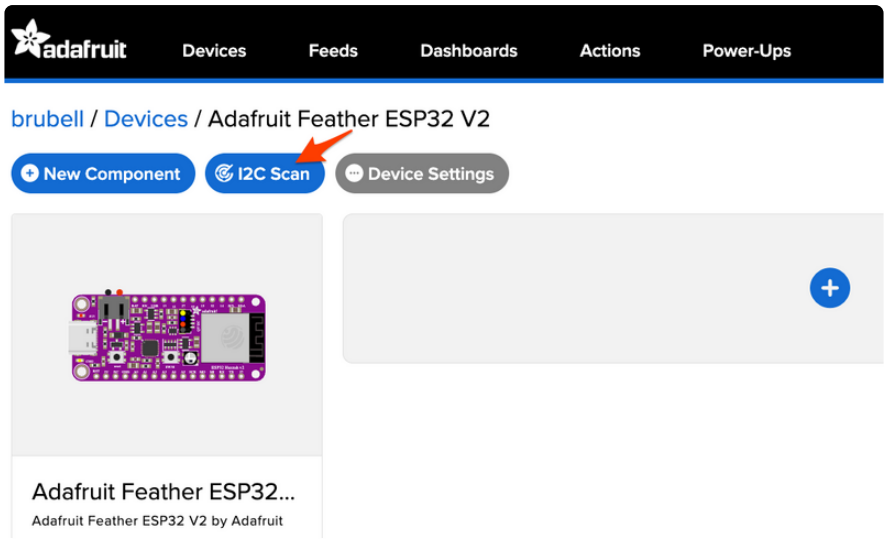
If you do not see your board listed here - you need to connect your board to Adafruit IO (https://adafru.it/Vfd) first.

On the device page, quickly **check that you're running the latest version of the WipperSnapper firmware**.

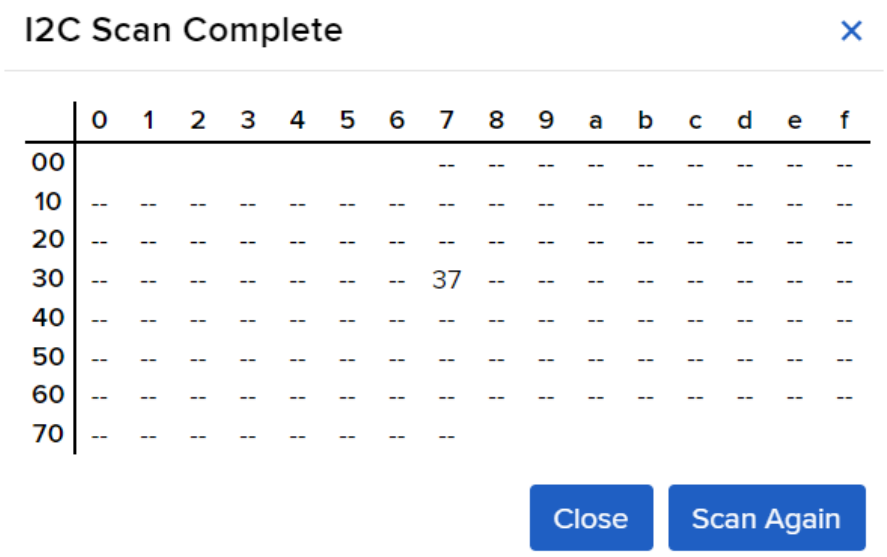
The device tile on the left indicates the version number of the firmware running on the connected board.

If the firmware version is green with a checkmark - continue with this guide.
If the firmware version is red with an exclamation mark "!" - update to the latest WipperSnapper firmware (https://adafru.it/Vfd) on your board before continuing.

Next, make sure the sensor is plugged into your board and click the **I2C Scan** button.



You should see the PCT2075's default I2C address of `0x37` pop-up in the I2C scan list.



I don't see the sensor's I2C address listed!

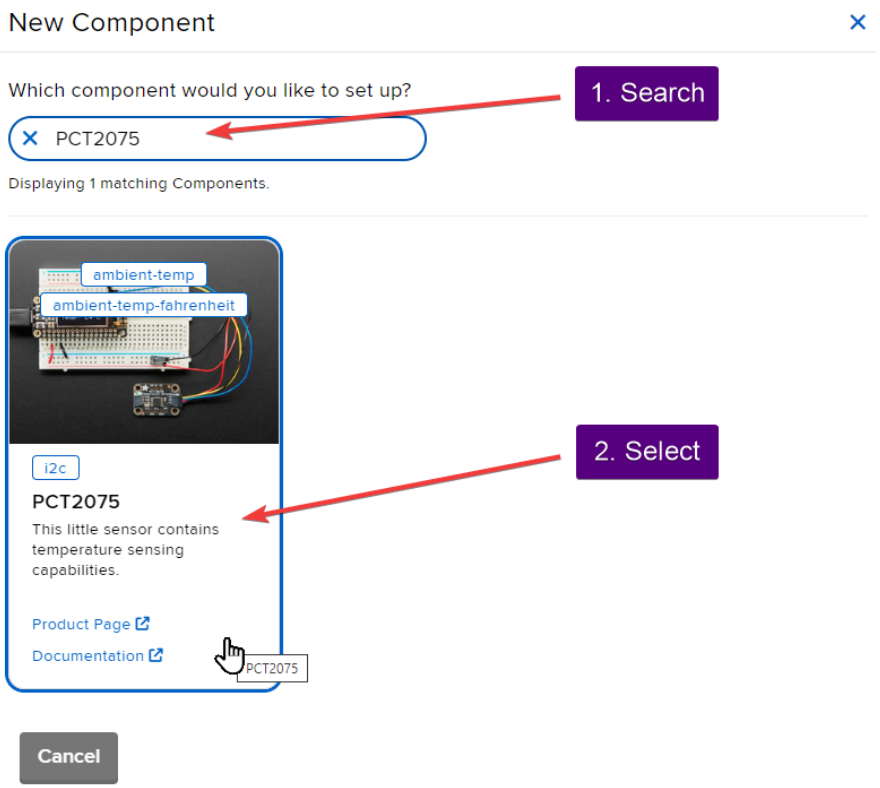
First, double-check the connection and/or wiring between the sensor and the board.

Then, reset the board and let it re-connect to Adafruit IO WipperSnapper.

With the sensor detected in an I2C scan, you're ready to add the sensor to your board.

Click the New Component button or the + button to bring up the component picker.

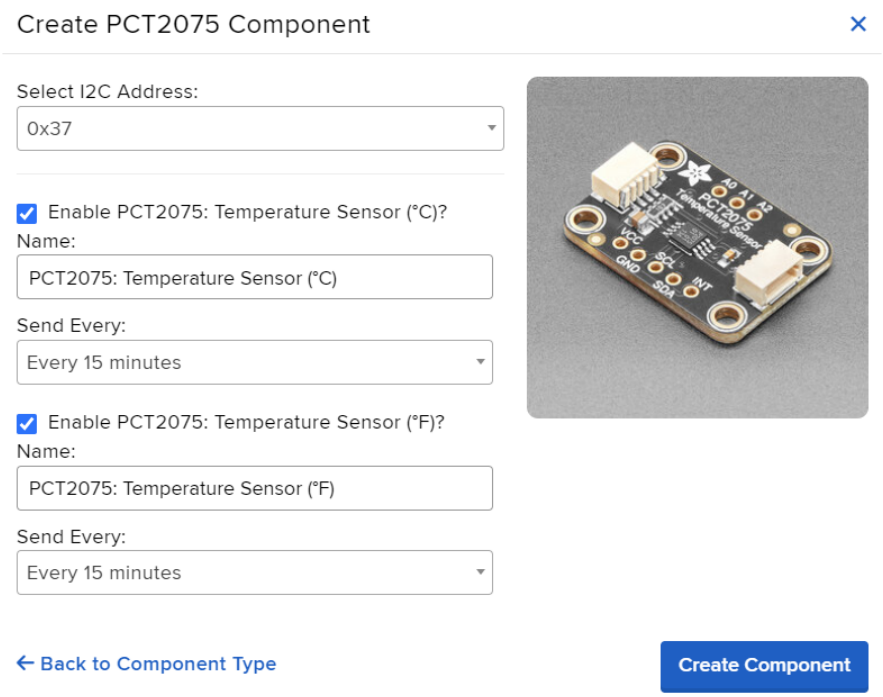
Adafruit IO supports a large amount of components. To quickly find your sensor, type `PCT2075` into the search bar, then select the **PCT2075** component.



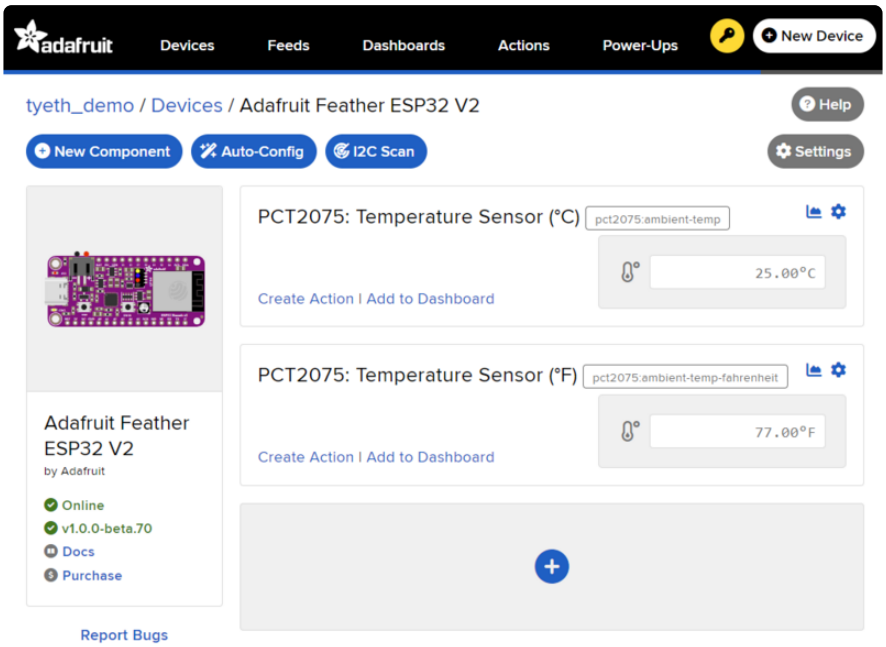
On the component configuration page, the PCT2075's sensor address should be listed along with the sensor's settings.

The **Send Every** option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the PCT2075's temperature sensor and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

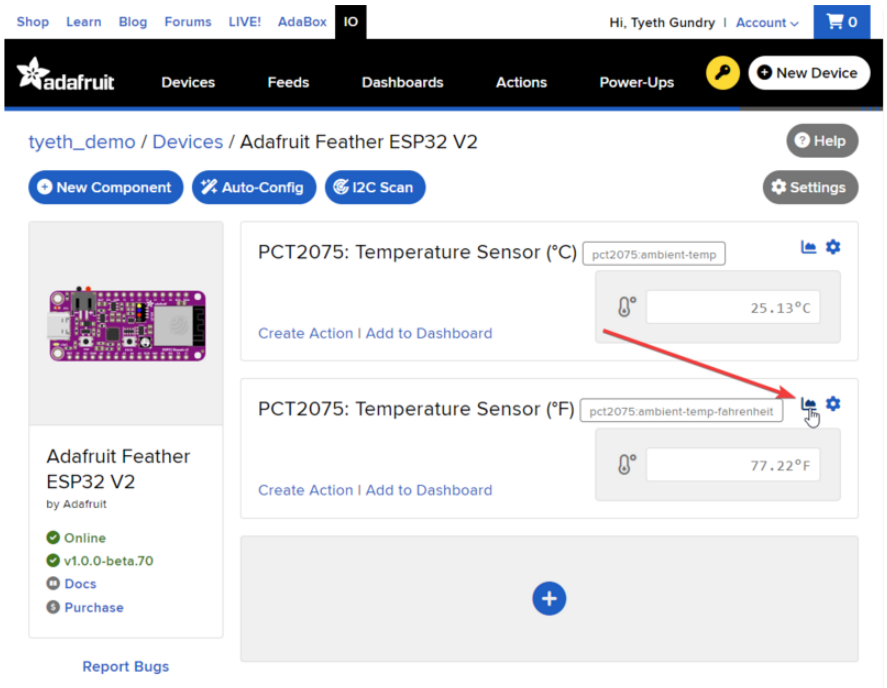
For this example, change the **Send Every** interval from the default of every 15 minutes to every 30 seconds.



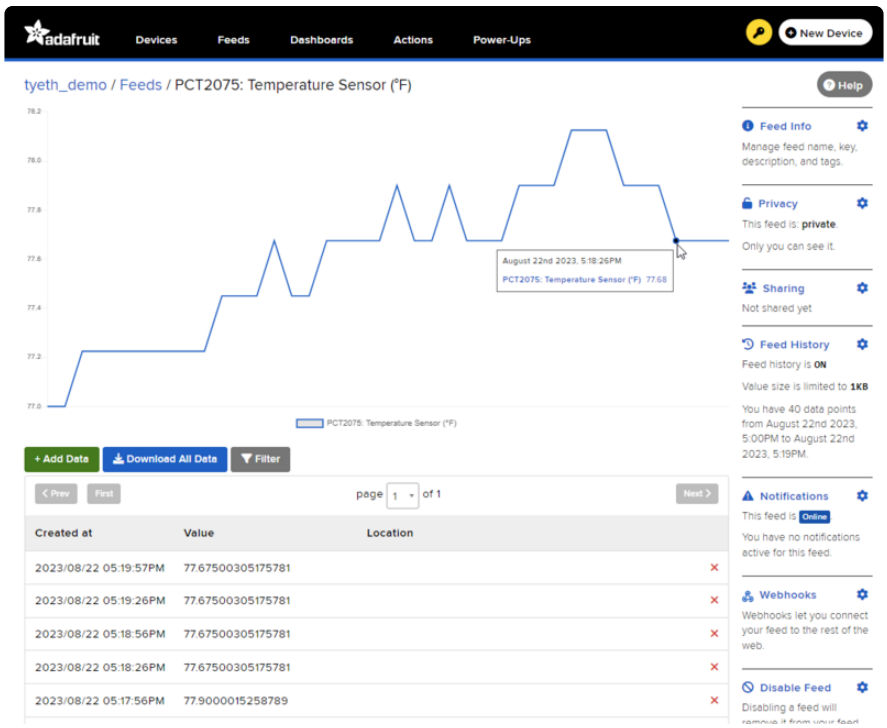
Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.



To view the data that has been logged from the sensor, click on the graph next to the sensor name.



Here you can see the feed history and edit things about the feed such as the name, privacy, webhooks associated with the feed and more. If you want to learn more about how feeds work, check out this page (https://adafru.it/10aZ).

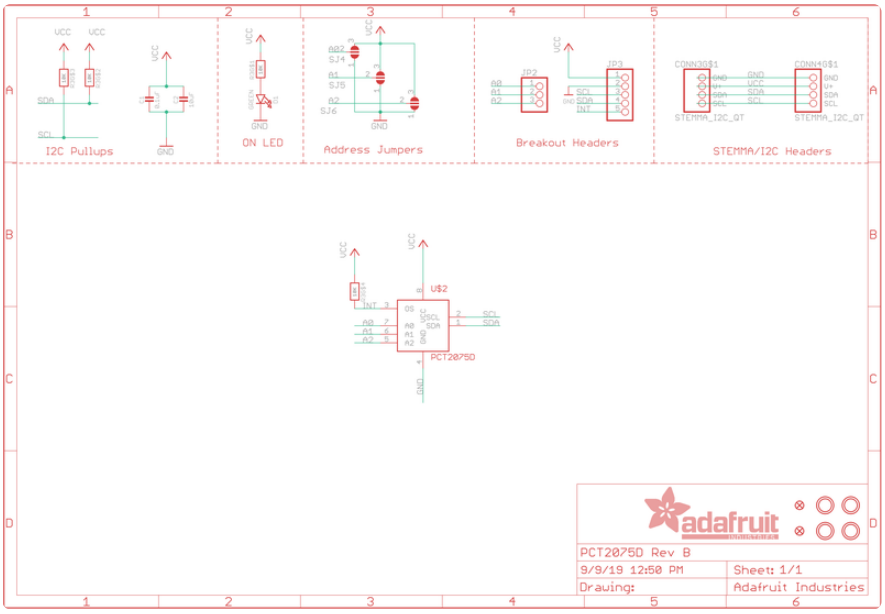
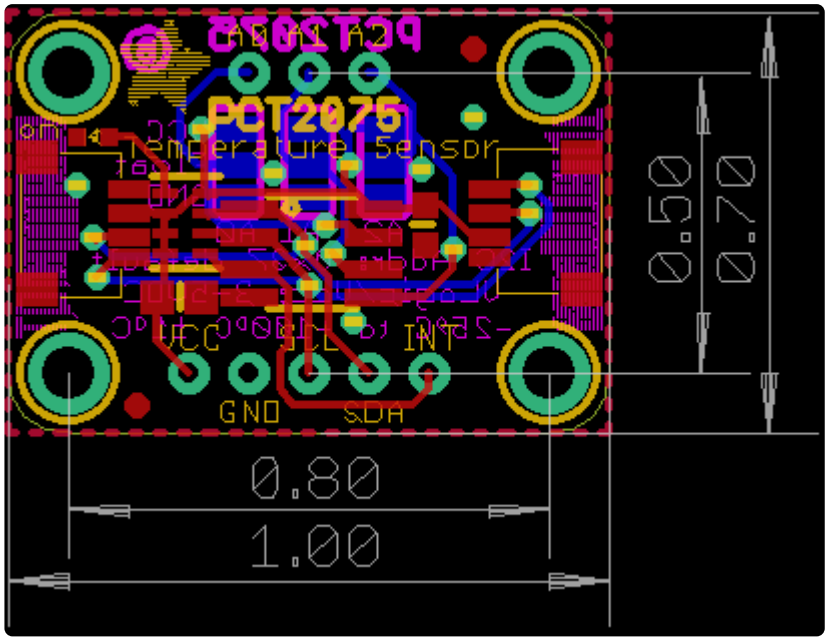


Downloads

Files

- PCT2075 Datasheet (https://adafru.it/FTU)
- EagleCAD files on GitHub (https://adafru.it/FTY)
- Fritzing object from Adafruit Fritzing Library (https://adafru.it/FTZ)

Schematic



Fab Print