## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
<td>3</td>
</tr>
<tr>
<td>• Technical Specs</td>
<td></td>
</tr>
<tr>
<td><strong>Pinouts</strong></td>
<td>6</td>
</tr>
<tr>
<td>• Power Pins</td>
<td></td>
</tr>
<tr>
<td>• I2C Logic Pins</td>
<td></td>
</tr>
<tr>
<td>• Address Pins and Jumpers</td>
<td></td>
</tr>
<tr>
<td>• Other Pins</td>
<td></td>
</tr>
<tr>
<td>• Power LED and Jumper</td>
<td></td>
</tr>
<tr>
<td><strong>Assembly</strong></td>
<td>8</td>
</tr>
<tr>
<td>• Assembly</td>
<td></td>
</tr>
<tr>
<td>• Add the ADT7410:</td>
<td></td>
</tr>
<tr>
<td>• And Solder!</td>
<td></td>
</tr>
<tr>
<td><strong>Arduino</strong></td>
<td>11</td>
</tr>
<tr>
<td>• Arduino Wiring</td>
<td></td>
</tr>
<tr>
<td>• Install Adafruit_ADT7410 Library</td>
<td></td>
</tr>
<tr>
<td>• Load Example</td>
<td></td>
</tr>
<tr>
<td>• Library Reference</td>
<td></td>
</tr>
<tr>
<td><strong>Arduino API</strong></td>
<td>14</td>
</tr>
<tr>
<td>• CircuitPython Microcontroller Wiring</td>
<td></td>
</tr>
<tr>
<td>• Python Computer Wiring</td>
<td></td>
</tr>
<tr>
<td>• Python Installation of ADT7410 Library</td>
<td></td>
</tr>
<tr>
<td>• CircuitPython Installation of ADT7410 Library</td>
<td></td>
</tr>
<tr>
<td>• Example Code</td>
<td></td>
</tr>
<tr>
<td>• Python Usage</td>
<td></td>
</tr>
<tr>
<td><strong>Python Docs</strong></td>
<td>18</td>
</tr>
<tr>
<td><strong>Downloads</strong></td>
<td>19</td>
</tr>
<tr>
<td>• Files</td>
<td></td>
</tr>
<tr>
<td>• STEMMA QT Schematic and Fab Print</td>
<td></td>
</tr>
<tr>
<td>• Original Schematic and Fab Print</td>
<td></td>
</tr>
</tbody>
</table>
Overview

Analog Devices, known for their reliable and well-documented sensor chips - has a high precision and high resolution temperature sensor on the market, and we've got a breakout to make it easy to use! The Analog Devices ADT7410 gets straight to the point - it's an I2C temperature sensor, with 16-bit 0.0078°C temperature resolution and 0.5°C temperature tolerance. Wire it up to your microcontroller or single-board computer to get reliable temperature readings with ease.
The ADT7410 has 2 address pins, so you can have up to 4 sensors on one I2C bus. There's also interrupt and critical-temperature alert pins. The sensor is good from 2.7V to 5.5V power and logic, for easy integration.

We've got both Arduino (C/C++) (1) and CircuitPython (Python 3) (2) libraries available so you can use it with any microcontroller like Arduino (1), ESP8266, Metro, etc. or with Raspberry Pi or other Linux computers (1) thanks to Blinka (our CircuitPython library support helper) (1).

To get you going fast, we spun up a custom made PCB with the ADT7410 and some supporting circuitry such as pullup resistors and capacitors, in the STEMMA QT form factor (1), making them easy to interface with. The STEMMA QT connectors (1) on either side are compatible with the SparkFun Qwiic (1) I2C connectors. This allows you to make solderless connections between your development board and the ADT7410 or to chain them with a wide range of other sensors and accessories using a compatible cable (1). A QT Cable is not included, but we have a variety in the shop (1).
Each order comes with a fully tested and assembled breakout and some header for soldering to a PCB or breadboard. You'll be up and running in under 5 minutes!

Thanks to Digi-Key () and Analog Devices () for sponsoring the development of this breakout board - we've made the PCB "Digi-Key red ()" in their honor!

Technical Specs

- Wide input-voltage range: 2.7 V to 5.5 V.
- Up to 16-bit temperature resolution (0.0078°C per lsb), default is 13 bits (0.0625°C per lsb).
- Highly accurate temperature tolerances:
  - ±0.5°C from −40°C to +105°C (2.7 V to 3.6 V)
  - ±0.4°C from −40°C to +105°C (3.0 V)
- Configurable I2C address allows up to four sensors on the I2C bus.
- Operates over I2C, so only two shared lines required.

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 default I2C address is 0x48.

Power Pins

- **VIN**: This is the voltage input to power for the sensor. You can connect either 5V or 3.3V to this, depending on the logic level of the MCU you are using. (Do not exceed 5V on this pin or you will permanently damage the sensor!)
- **GND**: Common ground for data and power.

I2C Logic Pins

- **SCL - I2C clock pin**, connect to your microcontroller’s I2C clock line. This pin has a 10K pullup to VIN
• SDA - I2C data pin, connect to your microcontroller's I2C data line. This pin has a 10K pullup to VIN.
• **STEMMA QT** - These connectors allow you to connect to development boards with STEMMA QT connectors, or to other things, with various associated accessories.

**Address Pins and Jumpers**

On the back of the board are two address jumpers, labeled A0 and A1, above the I2C Addr label on the board silk. These jumpers allow you to chain up to 4 of these boards on the same pair of I2C clock and data pins. To do so, you solder the jumper "closed" by connecting the two pads.

On the front of the board are two address pins, labeled A0 and A1. Just like the jumpers, these pins allow you to change the I2C address to connect multiple boards by connecting them to VIN.

The default I2C address is 0x48. The other address options can be calculated by “adding” the A0 and A1 to the base of 0x48.

A0 sets the lowest bit with a value of 1 and A1 sets the next bit with a value of 2. The final address is 0x48 + A0 + A1 which would be 0x4B.

If A0 is soldered closed, the address is 0x48 + 1 = 0x49.

If A1 is soldered closed, the address is 0x48 + 2 = 0x4A.

The table below shows all possible addresses, and whether the pin(s) should be high (closed) or low (open).

<table>
<thead>
<tr>
<th>ADDR</th>
<th>A0</th>
<th>A1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x48</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>0x49</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>0x4A</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>0x4B</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>
Other Pins

- **INT** - This is the 'open-drain' interrupt output pin, and can be optionally connected to your MCU to trigger a HW interrupt whenever an appropriate event happens with the sensor. See the datasheet and driver for further details. It will go low or logic '0' when it is asserted.
- **CT** - This 'open-drain' pin can be configured to trigger to go low or to logic '0' when a Critical Temperature (CT) threshold is passed.

Power LED and Jumper

- **Power LED** - In the upper left corner, above the STEMMA connector, on the front of the board, is the power LED, labeled on. It is the green LED.
- **LED jumper** - To the right of the power LED is a jumper for the power LED. If you wish to disable the power LED, simply cut the trace on this jumper.

Assembly

Start by preparing the header strip, cut to length if necessary. It will be easier to solder if you insert it into a breadboard - long pins down.
Add the ADT7410:

Place the board over the pins so that the short pins poke through the top of the breakout pads.
And Solder!

Be sure to solder all pins for reliable electrical contact.

(For tips on soldering, be sure to check out the Guide to Excellent Soldering (]).

OK, you're done!

You can now plug in your ADT7410 breakout and enjoy using this high precision and high resolution temperature sensor.
You can easily wire this breakout to any Arduino-compatible microcontroller.

Make the following connections between the Metro and the ADT7410:

- ADT7410 Vin to 3V or 5V (depending on the logic level of your board).
- ADT7410 GND to Metro GND.
- ADT7410 SCL to Metro SCL
- ADT7410 SDA to Metro SDA

Install Adafruit_ADT7410 Library

To read data from your ADT7410, you will need to [install the Adafruit_ADT7410 library](https://www.adafruit.com). It is available from the Arduino library manager so we recommend using that.

From the Arduino IDE, open the Library Manager (Sketch -> Include Library -> Manage Libraries)
Type in Adafruit ADT7410 and click Install

You'll also need to install Adafruit Unified Sensor Library:

Load Example

Open up File -> Examples -> Adafruit ADT7410 Library -> adt7410test and upload to your Arduino wired up to the sensor

```cpp
носторонних источниках

/*
This is a demo for the Adafruit ADT7410 breakout
----- http://www.adafruit.com/products/4089
Adafruit invests time and resources providing this open source code, please support Adafruit and open-source hardware by purchasing products from Adafruit!
*/

#include <Wire.h>
#include "Adafruit_ADT7410.h"

// Create the ADT7410 temperature sensor object
Adafruit_ADT7410 tempsensor = Adafruit_ADT7410();

void setup() {
  Serial.begin(115200);
  Serial.println("ADT7410 demo");
}
// Make sure the sensor is found, you can also pass in a different i2c address with tempsensor.begin(0x49) for example
if (!tempsensor.begin()) {
    Serial.println("Couldn't find ADT7410!");
    while (1);
}

// sensor takes 250 ms to get first readings
delay(250);
}

void loop() {
    // Read and print out the temperature, then convert to °F
    float c = tempsensor.readTempC();
    float f = c * 9.0 / 5.0 + 32;
    Serial.print("Temp: "); Serial.print(c); Serial.print("°C	");
    Serial.print(f); Serial.println("°F");
    delay(1000);
}

Upload the sketch to your board and open up the Serial Monitor (Tools->Serial Monitor). You should see the temperature in Celsius and Fahrenheit.

| Temp: 23.25°C | 73.85°F |
| Temp: 23.19°C | 73.74°F |
| Temp: 23.25°C | 73.85°F |
| Temp: 23.25°C | 73.85°F |

Library Reference

Create the ADT7410 Sensor Object:

```
Adafruit_ADT7410 tempsensor = Adafruit_ADT7410();
```

Initialize the sensor with:

```
tempsensor.begin()
```

This function returns True if the ADT7410 was initialized correctly, and False if it was not.

Once initialized, you can query the temperature in °C with

```
tempsensor.readTempC()
```

Which will return floating point (decimal + fractional) temperature. You can convert to Fahrenheit by multiplying by 1.8 and adding 32 as you have learned in grade school!
Arduino API

Python and CircuitPython

It's easy to use the ADT7410 sensor with Python or CircuitPython, and the Adafruit CircuitPython ADT7410 module. This module allows you to easily write Python code that reads 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 ADT7410 to your board exactly as shown on the previous page for Arduino.

Make the following connections between the CircuitPython board and the ADT7410:
Board 3V to ADT7410 VIN
Board GND to ADT7410 GND
Board SCL to ADT7410 SCL
Board SDA to ADT7410 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.

Make the following connections between the Pi and the ADT7410:
Python Installation of ADT7410 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:

```
sudo pip3 install adafruit-circuitpython-adt7410
```

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 ADT7410 Library

To use with CircuitPython, you need to first install the Adafruit CircuitPython ADT7410 () library, and its dependencies, into the lib folder on your CIRCUITPY drive. Then you need to update code.py with the example script.
Thankfully, we can do this in one go. In the example below, 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, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY/lib folder should contain the following folders and file:

- adafruit_adt7410.mpy
- /adafruit_bus_device
- /adafruit_register

Example Code

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

import time
import board
import adafruit_adt7410

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a microcontroller
adt = adafruit_adt7410.ADT7410(i2c, address=0x48)
adt.high_resolution = True

while True:
    print(adt.temperature)
    time.sleep(0.5)
```

Python Usage

To demonstrate the usage of the ADT7410, we'll use the Python REPL.

First, we'll import the following modules:

```python
import time
import board
```
import busio
import adafruit_adt7410

Next, we'll initialize the I2C bus and create the ADT object. We'll set its `high_resolution` property to `True`, to use 16-bit resolution (instead of the default 13-bit).

```python
i2c_bus = busio.I2C(board.SCL, board.SDA)
adt = adafruit_adt7410.ADT7410(i2c_bus, address=0x48)
adt.high_resolution = True
```

You can read the temperature using the `.temperature` property (the output will be in degrees Celsius). Try putting your finger on the sensor or holding it against something cold to see the values change.

```python
adt.temperature
```

You can convert to Fahrenheit by multiplying by 1.8 and adding 32:

```python
tempC = adt.temperature
tempF = tempC * 1.8 + 32.0
print(tempF)
```

That's all there is to reading the temperature with the ADT7410. Now you can use the ADT7410 temperature sensor to read the temperature in your project!
Downloads

Files

- ADT7410 Datasheet ()
- Original Fritzing Object available in the Adafruit Fritzing Library ()
- STEMMA QT Fritzing Object available in the Adafruit Fritzing Library ()
- EagleCAD PCB files on GitHub ()
- 3D models on GitHub ()

STEMMA QT Schematic and Fab Print

Dimensions are in inches.