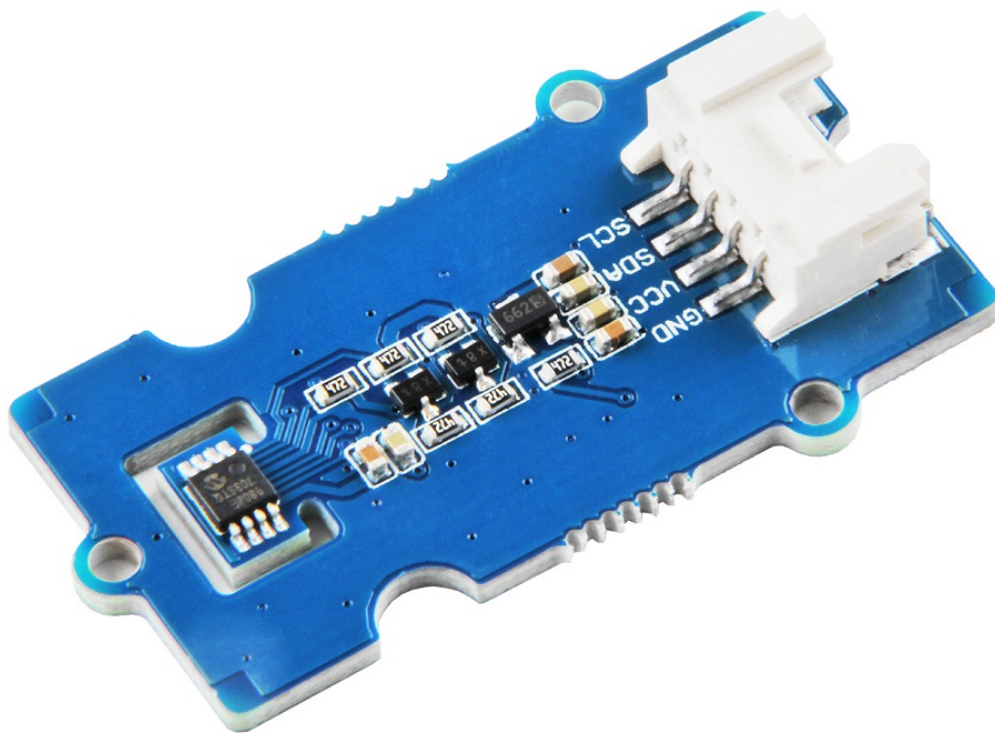


# Grove - I2C High Accuracy Temperature Sensor(MCP9808)



The Grove - I2C High Accuracy Temperature Sensor(MCP9808) is a high accuracy digital module based on MCP9808. Unlike other sensors, you can choose the measurement resolution of this sensor. In addition to high-precision temperature measurements, we also offer programmable temperature alert. We use a separate

pin to output the alarm signal, you will find it so convenient to use this signal as an interruption to control other board.

All in all, we believe this sensor will be a new star for temperature control.



[<https://www.seeedstudio.com/Grove-I2C-High-Accuracy-Temperature-Sensor%28MCP9808%29-p-3108.html>]

## Features

- High Accuracy
  - $\pm 0.25$  (typical) from  $-40^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$
  - $\pm 0.5^{\circ}\text{C}$  (maximum) from  $-20^{\circ}\text{C}$  to  $100^{\circ}\text{C}$
  - $\pm 1^{\circ}\text{C}$  (maximum) from  $-40^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$
- User-Selectable Measurement Resolution
  - $+0.5^{\circ}\text{C}$ ,  $+0.25^{\circ}\text{C}$ ,  $+0.125^{\circ}\text{C}$ ,  $+0.0625^{\circ}\text{C}$
- User-Programmable Temperature Alert Output
- I<sup>2</sup>C interface

## Specification

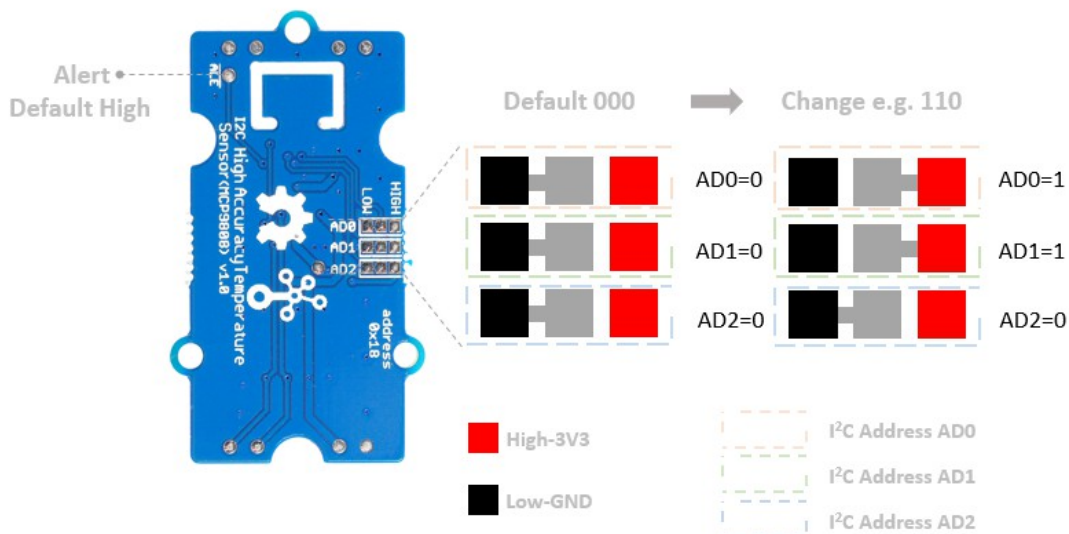
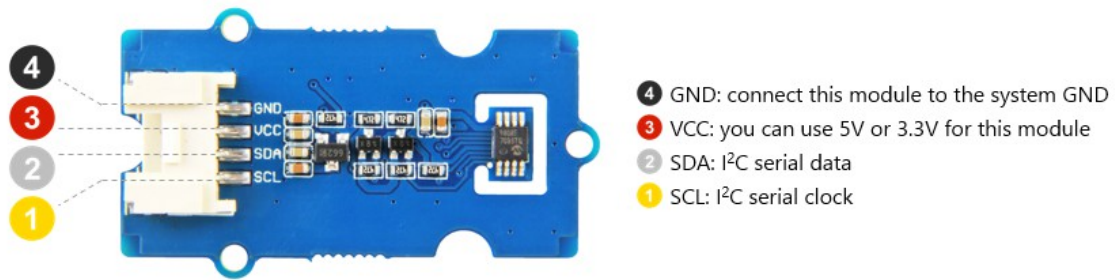
Item	Value
Working voltage	3.3V/5V
Operating range	-40°C to +125°C
Digital interface	I <sup>2</sup> C standard 400 kHz
I <sup>2</sup> C address	0x18(default)/ 0x18~0x1F(optional)

## Applications

- Industrial Applications
- Industrial Freezers and Refrigerators
- Food Processing
- Personal Computers and Servers
- PC Peripherals
- Consumer Electronics
- Handheld/Portable Devices

## Hardware Overview

## Pin Map



## I2C Address

We offer 3 sets of pads on the back of the PCB. The default AD0~AD2 are all connected to the Low level pads, you can cut those pads and solder them to the other side(High level). The I<sup>2</sup>C address is a 7bits address `0011A0A1A2`. `0011` is the address code, which is the factory setting, we can not change it. `A0A1A2` is the slave address, we can change it. The default setting is `A0=0/A1=0/A2=0`, so the default I<sup>2</sup>C address is `0011000`. Normally the address should be 8bits, so we need to add one bit 0 to the MSB(Most Significant Bit), then we get `0001,1000`. This is a binary address, we often use the hexadecimal address in the code, so let's convert the binary address to a hexadecimal address, here we get `0x18`. By the same token, if we solder all the pads to the high level,

we will get **0001,1111**, which is **0x1F**. So the I<sup>2</sup>C address range from 0x18 to 0x1F, among them, you can choose whatever you want, just make sure you will change the I<sup>2</sup>C address in the file **Seeed\_MCP9808.h** in the **Grove\_Temperature\_sensor\_MCP9808-master** library.

```
#define DEFAULT_IIC_ADDR 0X18
```



### Address map

A <sub>2</sub> =0	A <sub>0</sub> =0	A <sub>0</sub> =1
A <sub>1</sub> =0	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> -000,0x18	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> -001,0x19
A <sub>1</sub> =1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> -010,0x1A	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> -011,0x1B

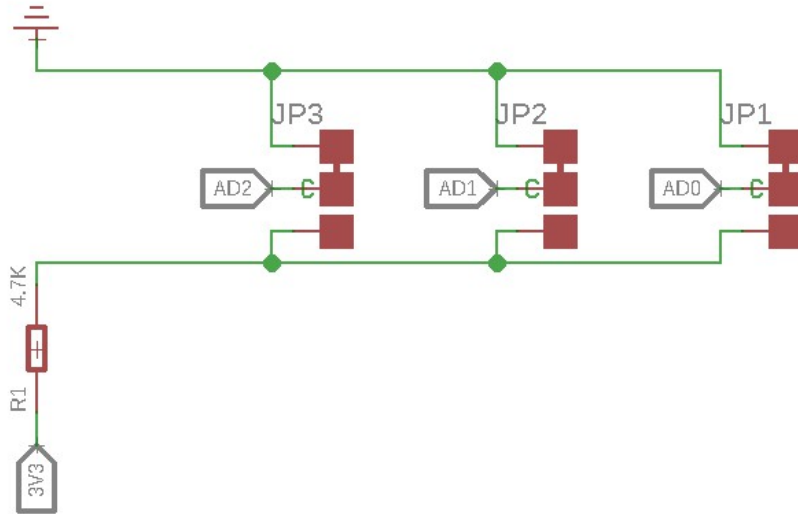
A <sub>2</sub> =1	A <sub>0</sub> =0	A <sub>0</sub> =1
A <sub>1</sub> =0=0	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> -100,0x1C	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> -101,0x1D
A <sub>1</sub> =0=1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> -110,0x1E	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> -111,0x1F

### ALE Pad

You can see the ALE Pad on the back of the PCB. The alert signal output from this pad can be used as an external interrupt signal for other controllers. The default output is high, in this board it should be 3.3V. When the condition is met, the output voltage becomes low(0V). You can set the condition when you finish this wiki 😊

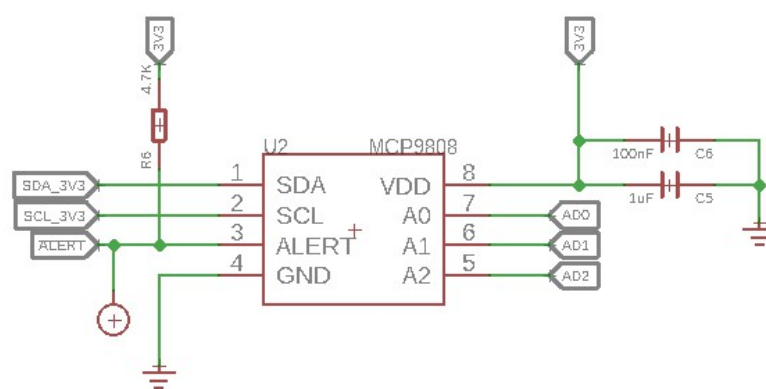
## Schematic

### I<sup>2</sup>C Address



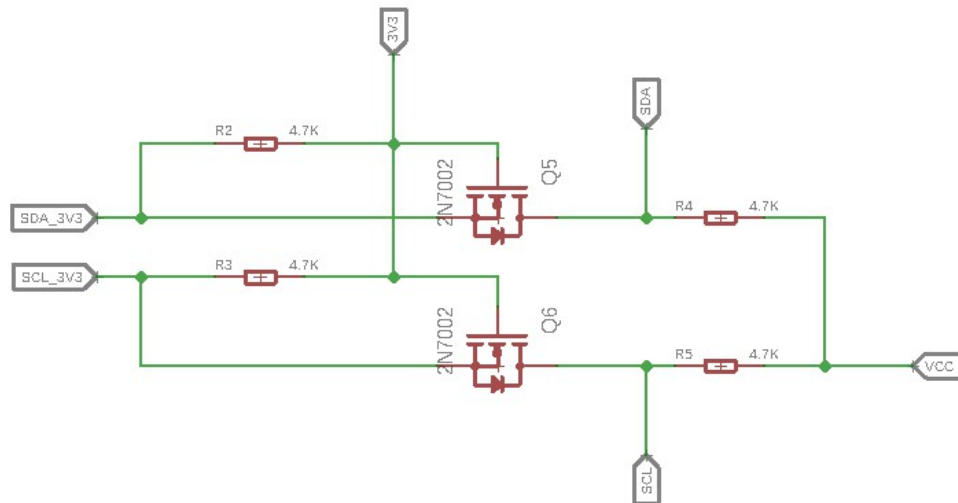
As we mentioned above, we use those three sets of pads to select the I<sup>2</sup>C address, if you want to change the default address, you can cut of the wire and re-solder it.

### MCP9808



As you can see, the  $\overline{ALE}$  pad is connected to the 3.3V through a pull-up resistor.

## Bi-directional level shifter circuit



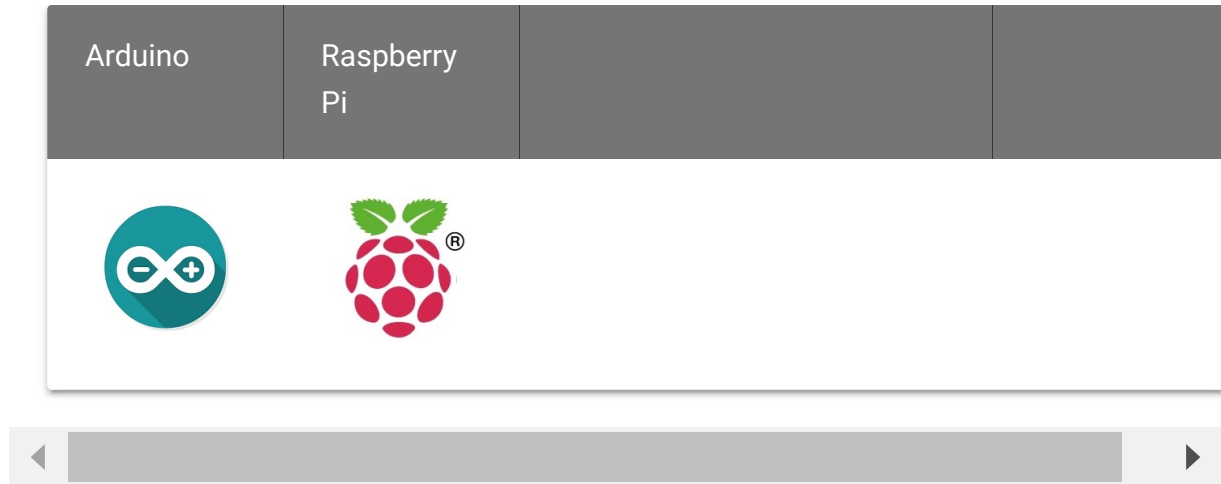
This is a typical Bi-directional level shifter circuit to connect two different voltage section of an I<sup>2</sup>C bus. The I<sup>2</sup>C bus of this sensor use 3.3V, if the I<sup>2</sup>C bus of the Arduino use 5V, this circuit will be needed. In the schematic above, **Q6** and **Q5** are N-Channel MOSFET **2N7002A** [[https://files.seeedstudio.com/wiki/Grove-I2C\\_High\\_Accuracy\\_Temperature\\_Sensor-MCP9808/res/2N7002A\\_datasheet.pdf](https://files.seeedstudio.com/wiki/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808/res/2N7002A_datasheet.pdf)], which act as a bidirectional switch. In order to better understand this part, you can refer to the **AN10441** [[https://files.seeedstudio.com/wiki/Grove-I2C\\_High\\_Accuracy\\_Temperature\\_Sensor-MCP9808/res/AN10441.pdf](https://files.seeedstudio.com/wiki/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808/res/AN10441.pdf)]



### Tip

In this section we only show you part of the schematic, for the full document please refer to the **Resources** [[https://wiki.seeedstudio.com/Grove-I2C\\_High\\_Accuracy\\_Temperature\\_Sensor-MCP9808/#resources](https://wiki.seeedstudio.com/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808/#resources)]

## Platforms Supported



### Caution

The platforms mentioned above as supported is/are an indication of the module's software or theoretical compatibility. We only provide software library or code examples for Arduino platform in most cases. It is not possible to provide software library / demo code for all possible MCU platforms. Hence, users have to write their own software library.

## Getting Started

### Play With Arduino

#### Hardware

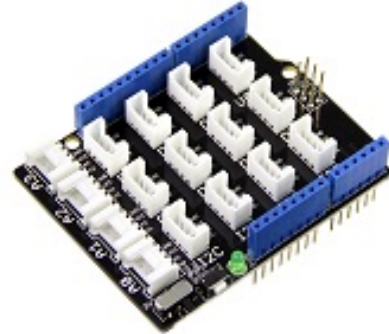
#### Materials required



Seeeduino V4.2



Base Shield



Get One Now

[<https://www.seeedstudio.com/Seeeduino-V4.2-p-2517.html>]

Get One Now

[<https://www.seeedstudio.com/Base-Shield-V2-p-1378.html>]



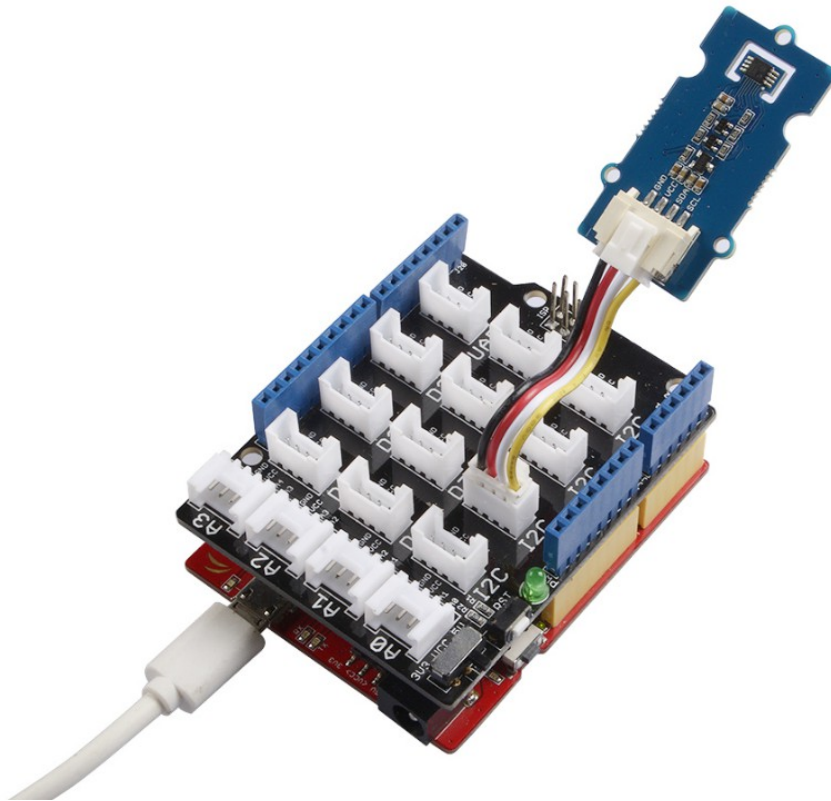
#### Note

**1** Please plug the USB cable gently, otherwise you may damage the port. Please use the USB cable with 4 wires inside, the 2 wires cable can't transfer data. If you are not sure about the wire you have, you can click [here](https://www.seeedstudio.com/Micro-USB-Cable-48cm-p-1475.html) [<https://www.seeedstudio.com/Micro-USB-Cable-48cm-p-1475.html>] to buy

**2** Each Grove module comes with a Grove cable when you buy. In case you lose the Grove cable, you can click [here](https://www.seeedstudio.com/Grove-Universal-4-Pin-Buckled-20cm-Cable-%285-PCs-pack%29-p-936.html) [<https://www.seeedstudio.com/Grove-Universal-4-Pin-Buckled-20cm-Cable-%285-PCs-pack%29-p-936.html>] to buy.

- **Step 1.** Connect the Grove - I2C High Accuracy Temperature Sensor to port **I<sup>2</sup>C** of Grove-Base Shield.
- **Step 2.** Plug Grove - Base Shield into Seeeduino.

- **Step 3.** Connect Seeeduino to PC via a USB cable.

**Note**

If we don't have Grove Base Shield, We also can directly connect this module to Seeeduino as below.

Seeeduino	Grove-MCP9808
5V	Red
GND	Black
SDA	White
SCL	Yellow

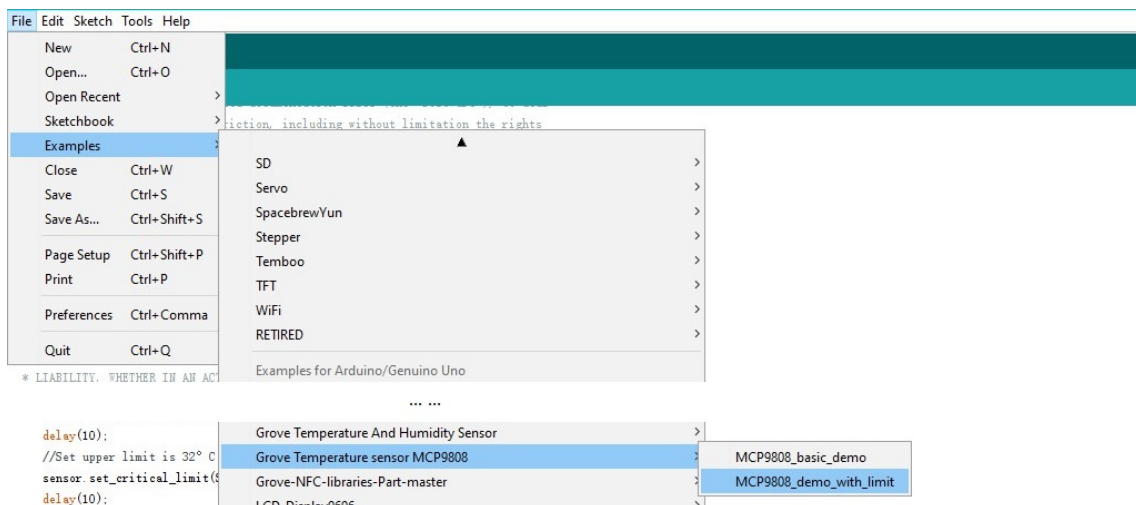
## Software



### Note

If this is the first time you work with Arduino, we strongly recommend you to see [Getting Started with Arduino](https://wiki.seeedstudio.com/Getting_Started_with_Arduino/) [https://wiki.seeedstudio.com/Getting\_Started\_with\_Arduino/] before the start.

- **Step 1.** Download the [Grove MCP9808](https://github.com/Seeed-Studio/Grove_Temperature_sensor_MCP9808) [https://github.com/Seeed-Studio/Grove\_Temperature\_sensor\_MCP9808] Library from Github.
- **Step 2.** Refer to [How to install library](https://wiki.seeedstudio.com/How_to_install_Arduino_Library) [https://wiki.seeedstudio.com/How\_to\_install\_Arduino\_Library] to install library for Arduino.
- **Step 3.** Restart the Arduino IDE. Open example via the path: **File** → **Examples** → **Grove Temperature Sensor MCP9808** → **MCP9808\_demo\_with\_limit**.



### Tips

As shown in the picture above, we provide two demos for you, **MCP9808\_basic\_demo** and **MCP9808\_demo\_with\_limit**. The **MCP9808\_basic\_demo** only provide the temperature, the alert fuction is disable. And for the **MCP9808\_demo\_with\_limit** demo, the alert function is enable. If you just want the temperature, the basic demo will be fine. If you want to use the alert function, you should choose the limit demo.

- **Step 4.** Upload the demo. If you do not know how to upload the code, please check [How to upload code](https://wiki.seeedstudio.com/Upload_Code/) [https://wiki.seeedstudio.com/Upload\_Code/].
- **Step 5.** Open the **Serial Monitor** of Arduino IDE by click **Tool->Serial Monitor**. Or tap the `Ctrl + Shift + M` key at the same time. if every thing goes well, you will get the result.

The result should be like

```
1  sensor init!!
2  temperature value is: 29.31
3  temperature value is: 29.31
4  temperature value is: 29.31
5  temperature value is: 29.25
6  temperature value is: 29.25
7  temperature value is: 29.25
8  temperature value is: 29.25
9  temperature value is: 29.25
10 temperature value is: 29.19
11 temperature value is: 29.25
```

Now, let's see how to use the ALE Pad.

The code in the demo **MCP9808\_demo\_with\_limit**:

```
1  #include "Seeed_MCP9808.h"
2
```

```
3
4 MCP9808 sensor;
5
6 void setup()
7 {
8     Serial.begin(115200);
9     if(sensor.init())
10    {
11        Serial.println("sensor init failed!!");
12    }
13    //Set upper limit is 30°C
14    sensor.set_upper_limit(SET_UPPER_LIMIT_ADDR,0x01e0);
15    delay(10);
16    //Set upper limit is 32°C
17    sensor.set_critical_limit(SET_CRITICAL_LIMIT_ADDR,0x01f0);
18    delay(10);
19    //Enable the alert bit.The alert bit outputs low when
20    sensor.set_config(SET_CONFIG_ADDR,0x0008);
21
22    Serial.println("sensor init!!");
23 }
24
25
26 void loop()
27 {
28     float temp=0;
29     //Get temperature ,a float-form value.
30     sensor.get_temp(&temp);
31     Serial.print("temperature value is: ");
32     Serial.println(temp);
33     delay(1000);
34 }
```

In addition to measuring temperature, this code also implements a function. When the temperature is lower than 30°C, the **ALE Pad** output default high-3.3v. When the temperature is higher than 30°C, the **ALE Pad** will output low-0v.

So you may ask, what if i want to change the threshold temperature. OK, please come to the line 14:

```
sensor.set_upper_limit(SET_UPPER_LIMIT_ADDR,0x01e0);
```



We use this function to control the temperature, the first parameter is the UPPER\_LIMIT register address and the second parameter **0x01e0** is the Hexadecimal temperature we set, as we mentioned above, it's 30°C. The **0x01e0** is a four bit Hexadecimal number, the last bit in the right represent the fractional part. We set it as 0, then the valid number is **0x1e**. **e** means 14 in decimal, and the higher bit **1** means 16 in decimal. So **0x1e** equals 16+14=30.

We provide 3 functions in the file **Seeed\_MCP9808.cpp**.

```
sensor.set_upper_limit(SET_UPPER_LIMIT_ADDR,u16);  
sensor.set_lower_limit(SET_LOWER_LIMIT_ADDR,u16);  
sensor.set_critical_limit(SET_CRITICAL_LIMIT_ADDR,u16);
```

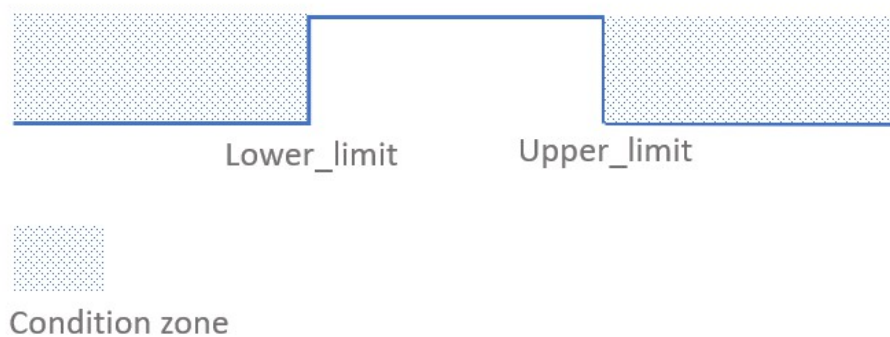
As we mentioned before, the default output of the **ALE Pad** is high, and the output level goes low when the temperature meets certain conditions. You can use those 3 functions to set your own conditions.

**sensor.set\_lower\_limit(SET\_LOWER\_LIMIT\_ADDR,u16)** is used to set the lower temperature limit, **u16** is the 4 bit Hexadecimal temperature we set. When the temperature is lower than the value we set, the output of the **ALE Pad** will goes down.

**sensor.set\_upper\_limit(SET\_UPPER\_LIMIT\_ADDR,u16)** is used to set the upper temperature limit, also **u16** is the 4 bit Hexadecimal temperature we set. When the temperature is higher than the value we set, the output of the **ALE Pad** will goes down.

**sensor.set\_critical\_limit(SET\_CRITICAL\_LIMIT\_ADDR,u16)** is used for the interrupt mode, in this wiki we only show you how to work as a comparator. If you want to know more, please check the [datasheet](https://files.seeedstudio.com/wiki/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808/res/MCP9808_datasheet.pdf) [https://files.seeedstudio.com/wiki/Grove-I2C\_High\_Accuracy\_Temperature\_Sensor-MCP9808/res/MCP9808\_datasheet.pdf] .

Now we can set a condition zone by **lower\_limit** and **upper\_limit**, when the temperature comes to the condition zone, the output will go low.



For example, if you want the **ALE Pad** output high between 28°C and 30°C, and output low when the temperature is higher than 30°C or lower than 28°C. The code should be like:

```
1 sensor.set_lower_limit(SET_LOWER_LIMIT_ADDR,0x01c0);
2 delay(10);
3 sensor.set_upper_limit(SET_UPPER_LIMIT_ADDR,0x01e0);
4 delay(10);
```




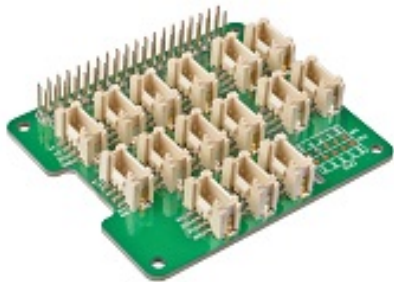
#### Attention

Please make sure the **upper\_limit** is higher than the **lower\_limit**, otherwise it will not output properly. And please make sure the **critical\_limit** is higher than the **upper\_limit**. A certain delay() is required to ensure that the registers are written correctly.

## Play With Raspberry Pi

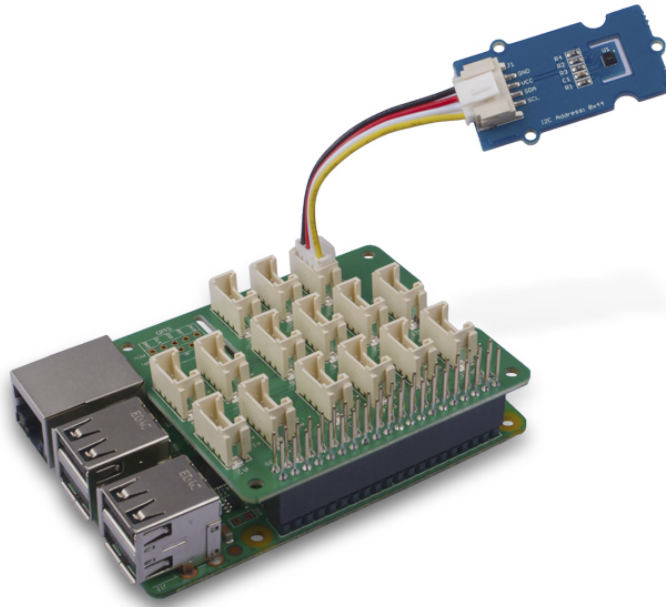
### Hardware

- **Step 1.** Things used in this project:

Raspberry pi	Grove Base Hat for RasPi
	
<p>Get ONE Now</p> <p>[<a href="https://www.seeedstudio.com/Raspberry-Pi-3-Model-B-p-2625.html">https://www.seeedstudio.com/Raspberry-Pi-3-Model-B-p-2625.html</a>]</p>	<p>Get ONE Now</p> <p>[<a href="https://www.seeedstudio.com/Grove-Base-Hat-for-Raspberry-Pi-p-3186.html">https://www.seeedstudio.com/Grove-Base-Hat-for-Raspberry-Pi-p-3186.html</a>]</p>

- **Step 2.** Plug the Grove Base Hat into Raspberry.
- **Step 3.** Connect the Grove - I2C High Accuracy Temperature Sensor to I2C port of the Base Hat.
- **Step 4.** Connect the Raspberry Pi to PC through USB cable.





## Software



### Attention

If you are using **Raspberry Pi with Raspberrypi OS >= Bullseye**, you have to use this command line **only with Python3**.

- **Step 1.** Follow [Setting Software](https://wiki.seeedstudio.com/Grove_Base_Hat_for_Raspberry_Pi/#installation) [https://wiki.seeedstudio.com/Grove\_Base\_Hat\_for\_Raspberry\_Pi/#installation] to configure the development environment.
- **Step 2.** Download the source file by cloning the grove.py library.

```
1 cd ~  
2 git clone https://github.com/Seeed-Studio/grove.py
```



- **Step 3.** Excute below commands to run the code.

```
1 cd grove.py/grove
2 python3 grove_high_accuracy_temperature.py
```

Following is the grove\_high\_accuracy\_temperature.py code.

```
1 import sys
2 import time
3 from grove.factory import Factory
4 from grove.temperature import Temper
5
6 def main():
7     print("Insert Grove - I2C-High-Accuracy-Temperature"
8         print(" to Grove-Base-Hat any I2C slot")
9
10    sensor = Factory.getTemper("MCP9808-I2C")
11    sensor.resolution(Temper.RES_1_16_CELSIUS)
12
13    print('Detecting temperature...')
14    while True:
15        print('{} Celsius'.format(sensor.temperature))
16        time.sleep(1)
17
18
19 if __name__ == '__main__':
20     main()
```



### Success

If everything goes well, you will be able to see the following result

```
1 pi@raspberrypi:~/grove.py/grove $ python3 grove_high_acc
2 Insert Grove - I2C-High-Accuracy-Temperature
3 to Grove-Base-Hat any I2C slot
4 Detecting temperature...
5 24.5 Celsius
6 24.5 Celsius
7 24.375 Celsius
```

```
8 ^CTraceback (most recent call last):  
9   File "grove_high_accuracy_temperature.py", line 54, in  
10     main()  
11   File "grove_high_accuracy_temperature.py", line 50, in  
12     time.sleep(1)  
13 KeyboardInterrupt
```

You can quit this program by simply press `Ctrl + C`.

## Schematic Online Viewer



## Resources

- **[Zip]** [Grove - I2C High Accuracy Temperature Sensor\(MCP9808\) Eagle files](https://files.seeedstudio.com/wiki/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808/res/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808.zip) [https://files.seeedstudio.com/wiki/Grove-I2C\_High\_Accuracy\_Temperature\_Sensor-MCP9808/res/Grove-I2C\_High\_Accuracy\_Temperature\_Sensor-MCP9808.zip]
- **[Zip]** [Seeed MCP9808 Library](https://files.seeedstudio.com/wiki/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808/res/Grove_Temperature_sensor_MCP9808-master.zip) [https://files.seeedstudio.com/wiki/Grove-I2C\_High\_Accuracy\_Temperature\_Sensor-MCP9808/res/Grove\_Temperature\_sensor\_MCP9808-master.zip]
- **[PDF]** [Datasheet of MCP9808](https://files.seeedstudio.com/wiki/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808/res/MCP9808_datasheet.pdf) [https://files.seeedstudio.com/wiki/Grove-I2C\_High\_Accuracy\_Temperature\_Sensor-MCP9808/res/MCP9808\_datasheet.pdf]
- **[PDF]** [Datasheet of 2N7002A](https://files.seeedstudio.com/wiki/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808/res/2N7002A_datasheet.pdf) [https://files.seeedstudio.com/wiki/Grove-I2C\_High\_Accuracy\_Temperature\_Sensor-MCP9808/res/2N7002A\_datasheet.pdf]
- **[PDF]** [AN10441](https://files.seeedstudio.com/wiki/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808/res/AN10441.pdf) [https://files.seeedstudio.com/wiki/Grove-I2C\_High\_Accuracy\_Temperature\_Sensor-MCP9808/res/AN10441.pdf]

## Project

This is the introduction Video of this product, simple demos, you can have a try.

## Grove - I2C High Accuracy Temperature Sensor (...)



## Tech Support

Please do not hesitate to submit the issue into our [forum](https://forum.seeedstudio.com/)  
[<https://forum.seeedstudio.com/>].



[[https://www.seeedstudio.com/act-4.html?utm\\_source=wiki&utm\\_medium=wikibanner&utm\\_campaign=newproducts](https://www.seeedstudio.com/act-4.html?utm_source=wiki&utm_medium=wikibanner&utm_campaign=newproducts)]