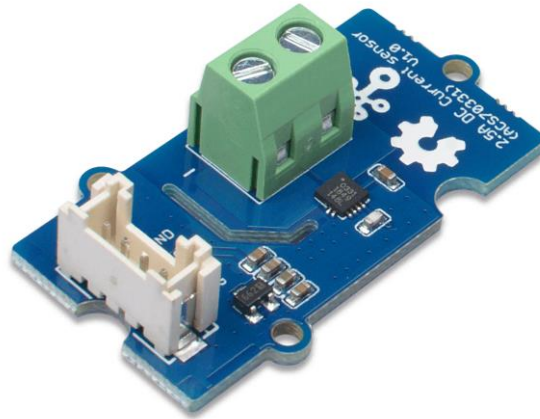


Grove-2.5A-DC-Current-Sensor-ACS70331



The Grove - 2.5A DC Current Sensor(ACS70331) is a high precision DC current sensor based on ACS70331. The ACS70331 is a chip series, this module uses ACS70331EESATR-2P5U3, which is Allegro's high sensitivity, current sensor IC for <2.5 A current sensing applications. It incorporates giant magneto-resistive (GMR) technology that is 25 times more sensitive than traditional Hall-effect sensors to sense the magnetic field generated by the current flowing through the low resistance, integrated primary conductor.

The Grove - 2.5A DC Current Sensor(ACS70331) can measure the DC current up to 2.5A and has a base sensitivity of 800mV/A. This sensor do not support AC current, if you want to measure the AC load please check the:

Feature

- 1 MHz bandwidth with response time <550 ns
- Low noise: 8 mA(rms) at 1 MHz
- 1.1 mΩ primary conductor resistance results in low power loss
- High DC PSRR enables use with low accuracy power supplies or batteries (3 to 4.5 V operation)
- Analog output

Specification

Parameter	Value
Supply voltage	3.3V / 5V
Operating ambient temperature	-40 – 85°C
Storage temperature	- 65°C – 125°C
Working Voltage	<100V
Current sensing range	0 – 2.5A
Sensitivity	800mV/A(Typ.)
Output interface	Analog
Input interface	Screw terminal

Working Principle

There are two types of current sensing: direct and indirect. Classification is mainly based on the technology used to measure current.

Direct sensing:

- Ohm's Law

Indirect sensing:

- Faraday's Law of Induction
- Magnetic field sensors
- Faraday Effect

The Grove - 2.5A DC Current Sensor(ACS70331) uses magnetic field sensors technology. And there are three kinds of Magnetic field sensors technology:

- Hall effect
- Flux gate sensors
- Magneto-resistive current sensor

The Grove - 2.5A DC Current Sensor(ACS70331) is based on the Magneto-resistive current sensor principle, which is also known as GMR. A magneto-resistor (MR) is a two

terminal device which changes its resistance parabolically with applied magnetic field. This variation of the resistance of MR due to the magnetic field is known as the Magnetoresistive Effect.

The internal construction of the ACS70331 QFN package is shown in Figure 2. The die sits above the primary current path such that magnetic field is produced in plane with the GMR elements on the die. GMR elements 1 and 2 sense field in the +X direction for positive IP current flow, and GMR elements 3 and 4 sense field in the -X direction for positive IP current flow. This enables differential measurement of the current and rejection of external stray fields.

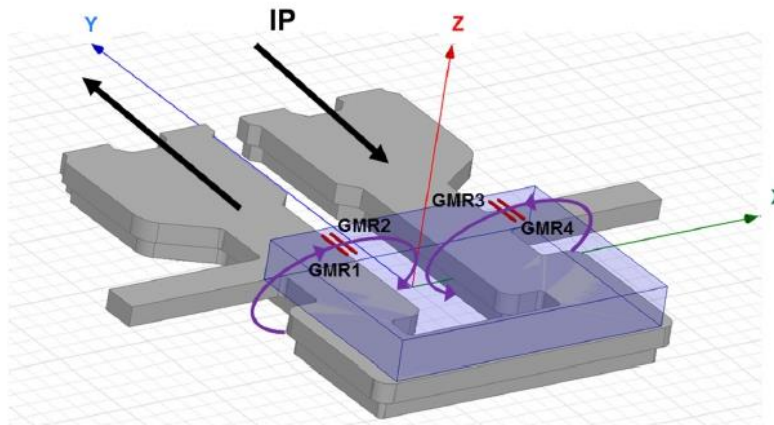


Figure 1. ACS70331 Internal Construction

The four GMR elements are arranged in a Wheatstone bridge configuration as shown in Figure 2 such that the output of the bridge is proportional to the differential field sensed by the four elements, rejecting common fields.

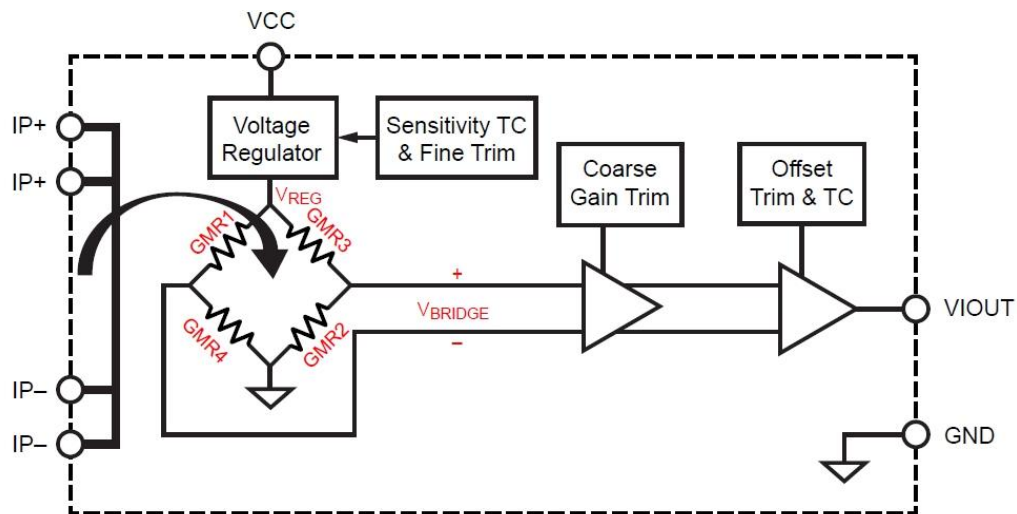


Figure 2. Wheatstone Bridge Configuration

Hardware Overview

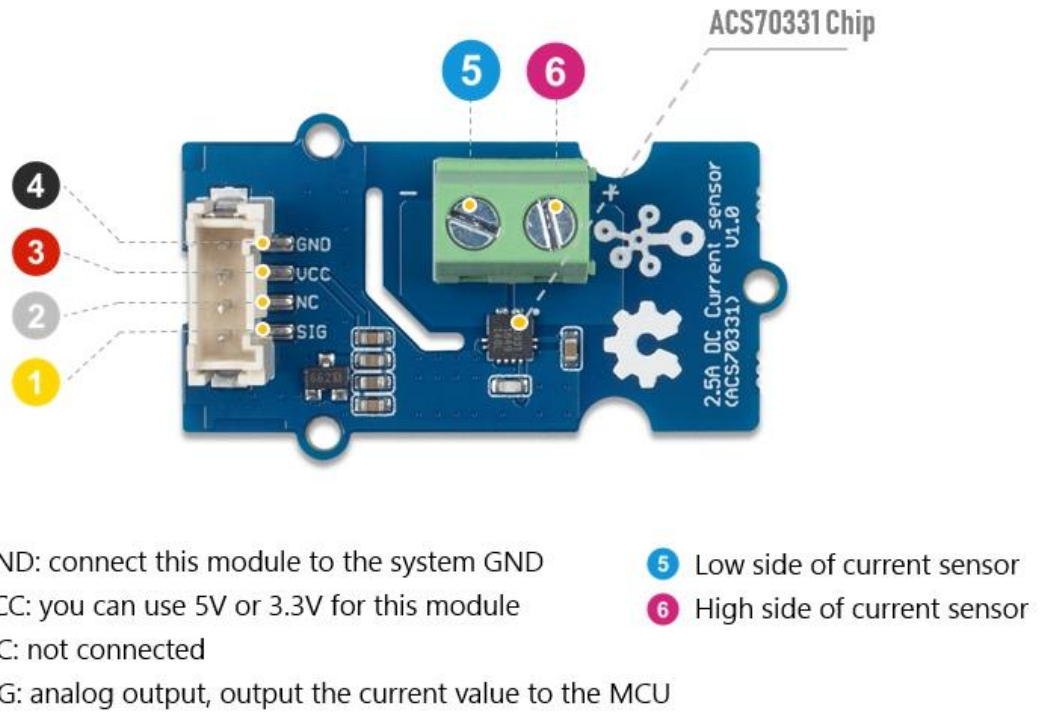


Figure 3. Pinout

Platforms Supported

Arduino	Raspberry Pi	BeagleBone	Wio	LinkIt ONE
				

Getting Started

Danger

The human body is forbidden to touch the module during the test, otherwise there is danger of electric shock.

Play With Arduino

Materials required

Seeeduino V4.2	Base Shield	2.5A DC Current Sensor(ACS70331)
		

In addition, you can consider our new [Seeeduino Lotus M0+](#), which is equivalent to the combination of Seeeduino V4.2 and Baseshield.

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](#) 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](#) to buy.

Hardware Connection

- **Step 1.** Connect the Grove - 2.5A DC Current Sensor(ACS70331) to the **A0** port of the Base Shield.
- **Step 2.** Connect the positive and negative poles of the circuit to be tested to the corresponding positive and negative poles of the screw terminal.

Tip

If you reverse the positive and negative poles, the reading will be reversed. This sensor need calibration before use, so please do not power on the circuit first.

- **Step 3.** Plug Grove - Base Shield into Seeeduino.
- **Step 4.** Connect Seeeduino to PC via a USB cable.

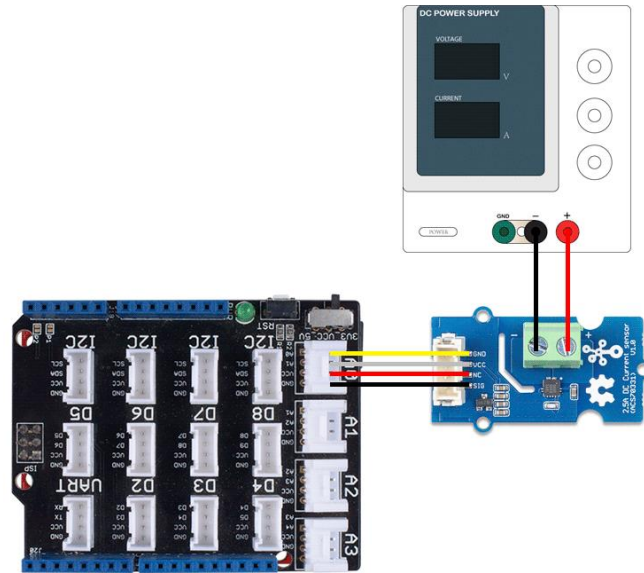


Figure 4. We use the DC Power Supply in this demo, please set the current to 0A or do not power on it at first

Software

Attention

If this is the first time you work with Arduino, we strongly recommend you to see [Getting Started with Arduino](#) before the start.

- **Step 1.** Download the [Grove Current Sensor Library](#) from Github.
- **Step 2.** In the /example/ folder, you can find the demo code. Here we take the **Grove_2_5A_Current_Sensor.ino** for instance. Just click the Grove_2_5A_Current_Sensor.ino to open the demo. Or you can copy the following code:

```

1 #ifndef ARDUINO_SAMD_VARIANT_COMPLIANCE
2   #define RefVal 3.3
3   #define SERIAL SerialUSB
4 #else
5   #define RefVal 5.0
6   #define SERIAL Serial
7 #endif
8 //An OLED Display is required here
9 //use pin A0
10 #define Pin A0
11
12 // Take the average of 10 times
13
14 const int averageValue = 10;
15
16 int sensorValue = 0;
17
18 float sensitivity = 1000.0 / 800.0; //1000mA per 800mV
19
20

```

```

21float Vref = 265; //Firstly,change this!!!
22
23void setup()
24{
25  SERIAL.begin(9600);
26}
27
28void loop()
29{
30  // Read the value 10 times:
31  for (int i = 0; i < averageValue; i++)
32  {
33    sensorValue += analogRead(Pin);
34
35    // wait 2 milliseconds before the next loop
36    delay(2);
37
38  }
39
40  sensorValue = sensorValue / averageValue;
41
42
43  // The on-board ADC is 10-bits
44  // Different power supply will lead to different reference sources
45  // example: 2^10 = 1024 -> 5V / 1024 ~= 4.88mV
46  //          unitValue= 5.0 / 1024.0*1000 ;
47  float unitValue= RefVal / 1024.0*1000 ;
48  float voltage = unitValue * sensorValue;
49
50  //When no load,Vref=initialValue
51  SERIAL.print("initialValue: ");
52  SERIAL.print(voltage);
53  SERIAL.println("mV");
54
55  // Calculate the corresponding current
56  float current = (voltage - Vref) * sensitivity;
57
58  // Print display voltage (mV)
59  // This voltage is the pin voltage corresponding to the current
60  /*
61  voltage = unitValue * sensorValue-Vref;
62  SERIAL.print(voltage);
63  SERIAL.println("mV");
64  */
65
66  // Print display current (mA)
67  SERIAL.print(current);
68  SERIAL.println("mA");
69
70  SERIAL.print("\n");
71
72  // Reset the sensorValue for the next reading
73  sensorValue = 0;
74  // Read it once per second
75  delay(1000);
76}

```

- **Step 3.** Upload the demo. If you do not know how to upload the code, please check [How to upload code](#).
- **Step 4.** Open the **Serial Monitor** of Arduino IDE by click **Tool-> Serial Monitor**. Or tap the **Ctrl + Shift + M** key at the same time. Set the baud rate to **9600**.
- **Step 5. Calibration**
When there is no current flowing, the sensor will still have a small output value. We call this value **zero offset**.

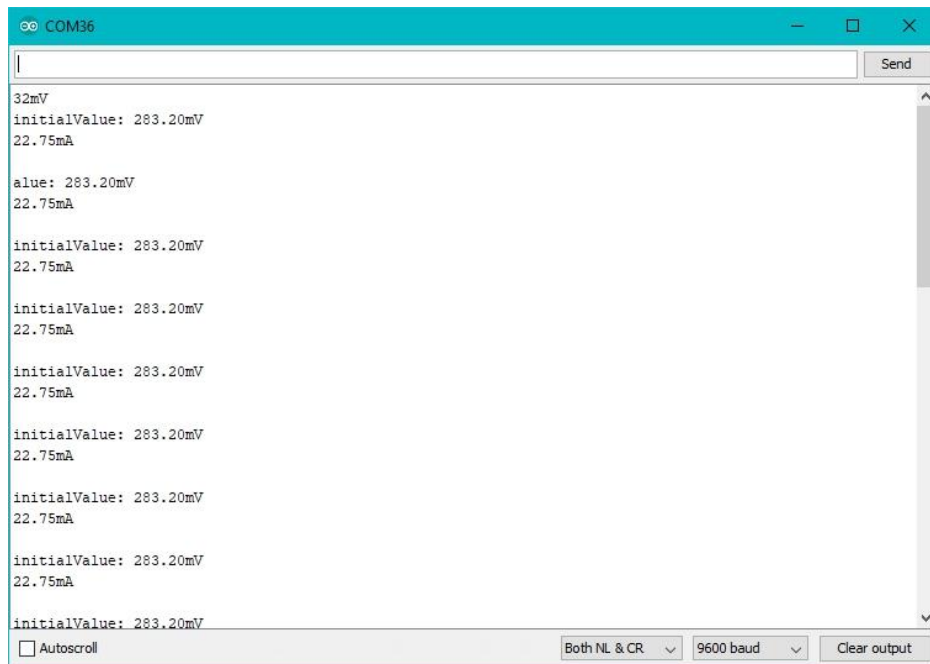


Figure 5. The zero offset of this board is 283.20mV, Converted into current is 22.75mA

Due to the presence of zero offset, the sensor will also have a reading when there is no current. So we set a parameter **Vref** to fix it, you can find it in the code block above.

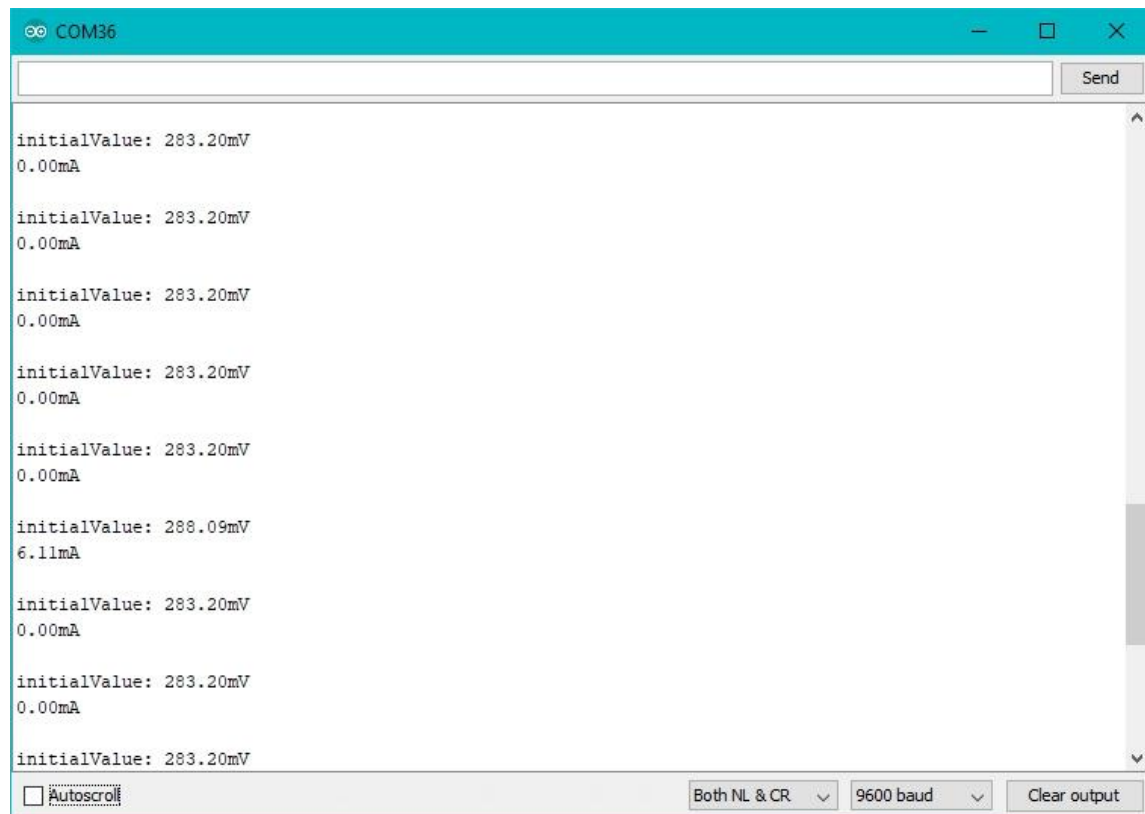
Line 21:

```
1float Vref = 265;
2//Vref is zero drift value, you need to change this value to the value you
  actually measured before using it.
```

In the demo code, we set the Vref to 265, however, the zero offset value varies from board to board. As you know, the board we use in this demo is 288.09. So let's modify the Line 21:

```
1float Vref = 283.20;
```


Then save the code and upload the code again, follow the Step 2. and Step 3. Now let's see :



The screenshot shows a serial terminal window with the following output:

```
initialValue: 283.20mV  
0.00mA  
  
initialValue: 283.20mV  
0.00mA  
  
initialValue: 283.20mV  
0.00mA  
  
initialValue: 283.20mV  
0.00mA  
  
initialValue: 283.20mV  
0.00mA  
  
initialValue: 288.09mV  
6.11mA  
  
initialValue: 283.20mV  
0.00mA  
  
initialValue: 283.20mV  
0.00mA  
  
initialValue: 283.20mV
```

The terminal interface includes a 'Send' button at the top right, an 'Autoscroll' checkbox at the bottom left, and dropdown menus for 'Both NL & CR' and '9600 baud' at the bottom right, along with a 'Clear output' button.

Figure 6. Now the current zero offset turns to 0mA

When the current output becomes to 0mA or a small value, you have completed the calibration.

- **Step 5.** Now it's all yours, you can power up the current. Please feel free to use it, remember this is a 2.5A DC Current Sensor, current cannot exceed 2.5A!

If you want to know the calculation formula of the result, please refer to the [FAQ Q1](#)

Play with Raspberry

Materials required

Raspberry pi	Grove Base Hat for RasPi	2.5A DC Current Sensor
		

Hardware Connection

- **Step 1.** Plug the Grove Base Hat into Raspberry Pi.
- **Step 2.** Connect the Grove - 2.5A DC Current Sensor(ACS70331) to port **A0** of the Base Hat.
- **Step 3.** Connect the positive and negative poles of the circuit to be tested to the corresponding positive and negative poles of the screw terminal.

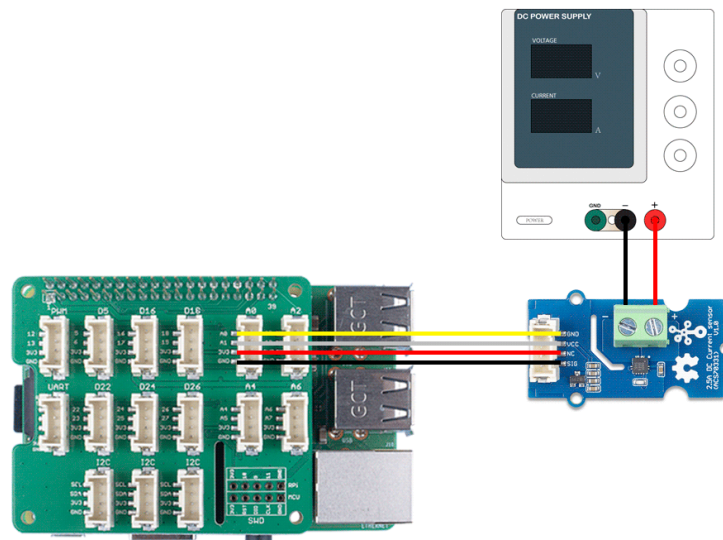


Figure 7. We use the DC Power Supply in this demo, please set the current to 0A or do not power on it at first

Tip

If you reverse the positive and negative poles, the reading will be reversed. This sensor need calibration before use, so please do not power on the circuit first.

- **Step 4.** Power the Raspberry Pi via the Micro-USB cable.

Attention

You can power the Raspberry Pi by computer USB port or DC adapter, however, if you are using the Raspberry pi 3B+, we strongly recommend you to power it by DC adapter, if you use the USB port of the PC, you may damage the Raspberry Pi 3B+.

Software

- **Step 1.** Follow [Setting Software](#) to configure the development environment.
- **Step 2.** Download the source file by cloning the [grove.py](#) library.

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

- **Step 3.** Execute following commands to run the code.

```
1cd grove.py/grove # to enter the demo file folder
2python grove_current_sensor.py 0 2.5A # to run the demo program
```

Then the terminal will output as following:

```
1pi@raspberrypi:~/grove.py/grove $ python grove_current_sensor.py 0 2.5A
2pin_voltage (mV) :
3270
4current (mA) :
513.0
6()
7pin_voltage (mV) :
8270
9current (mA) :
1013.0
11()
12pin_voltage (mV) :
13270
14current (mA) :
1513.0
16()
17pin_voltage (mV) :
18269
19current (mA) :
2011.0
21()
22pin_voltage (mV) :
23270
24current (mA) :
2513.0
26()
27^CTraceback (most recent call last):
28 File "grove_current_sensor.py", line 200, in <module>
29     main()
30 File "grove_current_sensor.py", line 185, in main
31     time.sleep(1)
```

Tap `Ctrl+C` to quit.

Note

Please note the second command, There are two parameters after the file name:

- **0** means the sensor is connected to port A0. If you connect the sensor to port A2, then you need to change this parameter to 2. This parameter has a range of 0-7, but if you use the Grove base hat, you can only use 0/2/4/6 because of the physical limitations of the interface.
- **2.5A** means the current sensor type is 2.5A DC

Sensor	Current type	Parameter Value
Grove - 2.5A DC Current Sensor(ACS70331)	DC	2.5A
Grove - ±5A DC/AC Current Sensor (ACS70331)	DC	5A_DC
	AC	5A_AC
Grove - 10A DC Current Sensor (ACS725)	DC	10A

This series has three current sensors, the parameter list is as above

- **Step 4 Calibration.**

When there is no current flowing, the sensor will still have a small output value. We call this value zero offset. As you can see, in the step 3, the zero offset of this board is 270mV, converted into current is 13mA.

Due to the presence of zero offset, the sensor will also have a reading when there is no current. So we set a parameter **Vref** to fix it, you can find it in the **python grove_current_sensor.py**. For the Grove - 2.5A DC Current Sensor(ACS70331), we set the **Vref** to 260 by default, however the zero offset varies from board to board. That's why we need to do the calibration first.

Check the python code below :

```

1#!/usr/bin/env python
2# -*- coding: utf-8 -*-
3#
4# The MIT License (MIT)
5# Copyright (C) 2018 Seeed Technology Co.,Ltd.
6#
7# This is the library for Grove Base Hat
8# which used to connect grove sensors for Raspberry Pi.
9'''
10This is the code for

```

```

11 - `Grove - 2.5A DC current sensor <https://www.seeedstudio.com/Grove-
122-5A-DC-Current-Sensor-ACS70331-p-2929.html>`_
13 - `Grove - 5A AC/DC current sensor <https://www.seeedstudio.com/Grove-
145A-DC-AC-Current-Sensor-ACS70331-p-2928.html>`_
15 - `Grove - 10A current sensor <https://www.seeedstudio.com/Grove-
1610A-DC-Current-Sensor-ACS725-p-2927.html>`_
17Examples:
18 .. code-block:: python
19     import time
20     from grove_current_sensor import Current
21     pin = 0
22     sensor_type = "2.5A"
23     #if use 10A current sensor input: pin = 0 , sensor_type = "10A"
24     if (sensor_type == "2.5A"):
25         sensitivity = 1000.0 / 800.0
26         Vref = 260
27     if (sensor_type == "5A_DC"):
28         sensitivity = 1000.0 / 200.0
29         Vref = 1498
30     if (sensor_type == "5A_AC"):
31         sensitivity = 1000.0 / 200.0
32         Vref = 1498
33     if (sensor_type == "10A"):
34         sensitivity = 1000.0 / 264.0
35         Vref = 322
36     averageValue = 500
37     ADC = Current()
38     while True:
39         if(sensor_type == "5A_AC"):
40             pin_voltage =
41ADC.get_nchan_vol_milli_data(pin,averageValue)
42             current =
43ADC.get_nchan_AC_current_data(pin,sensitivity,Vref,averageValue)
44             else:
45                 temp =
46ADC.get_nchan_current_data(pin,sensitivity,Vref,averageValue)
47                 current = temp[0]
48                 pin_voltage = temp[1]
49
50             current = round(current)
51             print("pin_voltage(mV):")
52             print(pin_voltage)
53             print("current(mA):")
54             print(current)
55             print()
56             time.sleep(1)
57
58'''
59
60import sys
61import time
62from grove.i2c import Bus
63
64ADC_DEFAULT_IIC_ADDR = 0X04
65
66ADC_CHAN_NUM = 8
67

```

```

68REG_RAW_DATA_START = 0X10
69REG_VOL_START = 0X20
70REG_RTO_START = 0X30
71
72REG_SET_ADDR = 0XC0
73
74__all__ = ['Current', 'Bus']
75
76class Current():
77    '''
78    Grove Current Sensor class
79    '''
80
81    def __init__(self, bus_num=1, addr=ADC_DEFAULT_IIC_ADDR):
82        '''
83        Init iic.
84        Args:
85            bus_num(int): the bus number;
86            addr(int): iic address;
87        '''
88        self.bus = Bus(bus_num)
89        self.addr = addr
90
91    def get_nchan_vol_milli_data(self, n, averageValue):
92        '''
93        Get n chanel data with unit mV.
94        :param int n: the adc pin.
95        :param int averageValue: Average acquisition frequency.
96        Returns:
97            int: voltage value
98        '''
99        val = 0
100        for i in range(averageValue):
101            data =
102self.bus.read_i2c_block_data(self.addr, REG_VOL_START+n, 2)
103            val += data[1]<<8|data[0]
104            val = val / averageValue
105        return val
106
107    def get_nchan_current_data(self, n, sensitivity, Vref, averageValue):
108        '''
109        2.5A/5A DC/10A cunrrent sensor get n chanel data with unit mA.
110        :param int n: the adc pin.
111        :param float sensitivity: The coefficient by which voltage is
112converted into current.
113        :param int Vref: Initial voltage at no load.
114        :param int averageValue: Average acquisition frequency.
115        Returns:
116            int: current value
117        '''
118        val = 0
119        for i in range(averageValue):
120            data =
121self.bus.read_i2c_block_data(self.addr, REG_VOL_START+n, 2)
122            val += data[1]<<8|data[0]
123            val = val / averageValue
124            currentVal = (val - Vref) * sensitivity

```

```

125     return currentVal, val
126
127     def get_nchan_AC_current_data(self, n, sensitivity, Vref, averageValue):
128         '''
129         5A current sensor AC output and get n channel data with unit mA.
130         :param int n: the adc pin.
131         :param float sensitivity: The coefficient by which voltage is
132 converted into current.
133         :param int Vref: Initial voltage at no load.
134         :param int averageValue: Average acquisition frequency.
135         Returns:
136             int: current value
137         '''
138         sensorValue = 0
139         for i in range(averageValue):
140             data=self.bus.read_i2c_block_data(self.addr, REG_VOL_START+n, 2)
141             val=data[1]<<8|data[0]
142             if(val > sensorValue):
143                 sensorValue=val
144             time.sleep(0.00004)
145         currentVal = ((sensorValue - Vref) * sensitivity)*0.707
146         return currentVal
147
148 ADC = Current()
149 def main():
150     if(len(sys.argv) == 3):
151
152         pin = int(sys.argv[1])
153         sensor_type = sys.argv[2]
154         if (pin < 8 and (sensor_type == "2.5A" or sensor_type == "5A_DC"
155 or sensor_type == "5A_AC" or sensor_type == "10A") ):
156             if (sensor_type == "2.5A"):
157                 sensitivity = 1000.0 / 800.0
158                 Vref = 260
159             if (sensor_type == "5A_DC"):
160                 sensitivity = 1000.0 / 200.0
161                 Vref = 1498
162             if (sensor_type == "5A_AC"):
163                 sensitivity = 1000.0 / 200.0
164                 Vref = 1498
165             if (sensor_type == "10A"):
166                 sensitivity = 1000.0 / 264.0
167                 Vref = 322
168             averageValue = 500
169
170             while True:
171
172                 if(sensor_type == "5A_AC"):
173                     pin_voltage =
174 ADC.get_nchan_vol_milli_data(pin, averageValue)
175                     current =
176 ADC.get_nchan_AC_current_data(pin, sensitivity, Vref, averageValue)
177                 else:
178                     temp =
179 ADC.get_nchan_current_data(pin, sensitivity, Vref, averageValue)
180                     current = temp[0]
181                     pin_voltage = temp[1]

```

```

182
183         current = round(current)
184         print("pin_voltage (mV):")
185         print(pin_voltage)
186         print("current (mA):")
187         print(current)
188         print()
189         time.sleep(1)
190
    else:
        print("parameter input error!")
        print("Please enter parameters for example: python
grove_current_sensor 0 2.5A")
        print("parameter1: 0-7")
        print("parameter2: 2.5A/5A_DC/5A_AC/10A")

    else:
        print("Please enter parameters for example: python
grove_current_sensor 0 2.5A")
        print("parameter1: 0-7")
        print("parameter2: 2.5A/5A_DC/5A_AC/10A")

if __name__ == '__main__':
    main()

```

You can modify the **Vref** at line 147 of the code block above:

```

1         if (pin < 8 and (sensor_type == "2.5A" or sensor_type == "5A_DC" or
2sensor_type == "5A_AC" or sensor_type == "10A") ):
3             if (sensor_type == "2.5A"):
4                 sensitivity = 1000.0 / 800.0
5                 Vref = 260
6             if (sensor_type == "5A_DC"):
7                 sensitivity = 1000.0 / 200.0
8                 Vref = 1498
9             if (sensor_type == "5A_AC"):
10                sensitivity = 1000.0 / 200.0
11                Vref = 1498
12            if (sensor_type == "10A"):
13                sensitivity = 1000.0 / 264.0
14                Vref = 322
                averageValue = 500

```

As you can see, for the 2.5A Current Sensor the default **Vref** is 260, and in the **Step 3**, we can find it when there is no current the zero offset value is 270mV. So let's change it into 270.

```

1         if (sensor_type == "2.5A"):
2             sensitivity = 1000.0 / 800.0
3             Vref = 270

```


Now, let's run this demo again.

```
1pi@raspberrypi:~/grove.py/grove $ python grove_current_sensor.py 0 2.5A
2pin_voltage (mV) :
3269
4current (mA) :
5-1.0
6()
7pin_voltage (mV) :
8270
9current (mA) :
100.0
11()
12pin_voltage (mV) :
13270
14current (mA) :
150.0
16()
17pin_voltage (mV) :
18270
19current (mA) :
200.0
21()
22pin_voltage (mV) :
23270
24current (mA) :
250.0
26()
27^CTraceback (most recent call last):
28 File "grove_current_sensor.py", line 200, in <module>
29   main()
30 File "grove_current_sensor.py", line 185, in main
31   time.sleep(1)
32KeyboardInterrupt
```

Well, better than before, now you can measure the current more accurately 😊

FAQ

Q1# What's the current calculation formula?

A1: If you think the **principle part** is very complicated, let's put it in a easy way. The current in the circuit to be tested excites the magnetic field, which causes the resistance value of the GMR elements change. And the resistance change in the bridge causes a change in the voltage at the output of the chip. We call the voltage output as V_{IOUT} .

$$V_{IOUT} = Sens \times I_P + V_{IOUT}(Q)$$

Sens: Sens is the coefficient that converts the current into an output voltage. For this module it is 800mA/V.

I_p: I_p is the current value in the circuit to be tested, Unit mA.

V_{IOUT(Q)}: V_{IOUT(Q)} is the voltage output when the I_p is 0mA(which means there is no current in the circuit to be tested), Unit mV.

Here comes the current value:

$$I_p = \frac{V_{IOUT} - V_{IOUT(Q)}}{Sens}$$

Now, Let's review the figure 5, we will explain why the current value of the output is not 0 when the actual current value in the circuit to be tested is 0. As you can see in the figure 5, the **initialValue** is 283.20mV, which is the **V_{IOUT}**; the current is 22.75mA, which is the **I_p**. As for the **V_{IOUT(Q)}**, it is the **Vref** we set in the code. In figure 5, it is 265. And the **Sens** is 800mA/V, which is 800mA/1000mV. Now, just do some math:

$$\frac{283.20mV - 265mV}{800mA/1000mV} = 22.75mA$$

So, in the figure 6, when we set the **Vref** to 283.20, the **I_p** turns to 0mA.

Resources

- [\[ZIP\] Grove - 2.5A DC Current Sensor\(ACS70331\) Schematic file](#)
- [\[PDF\] ACS70331 Datasheet](#)

Tech Support

Please submit any technical issue into our [forum](#) or drop mail to techsupport@seeed.cc