AD/DA Converter（PCF8591）

Overview

In this chapter, we will learn how to read analog quantities through AD/DA Module,convert it into digital quantity and convert the digital quantity into analog output. That is, ADC and DAC.

Experimental Materials:

Raspberry Pi *1

5mm red LED light *1

T-type expansion board *1

220 ohm resistor *1

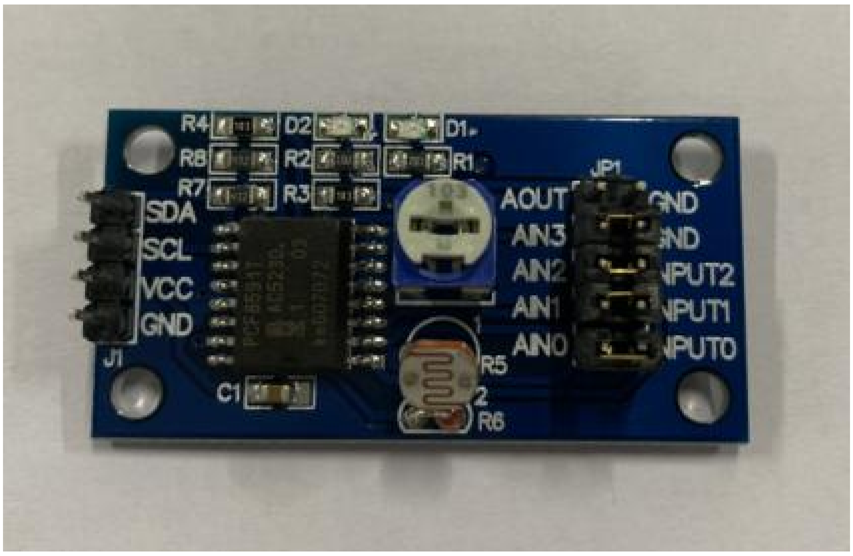
Breadboard*1

PCF8591 *1

Potentiometer *1

Some DuPont lines

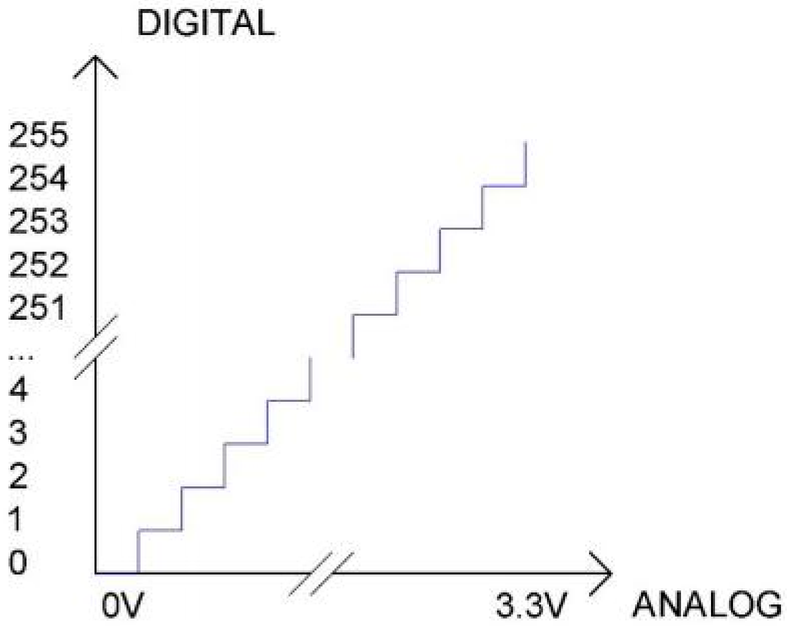
Product description：



Circuit knowledge：

ADC

ADC, Analog-to-Digital Converter, is a device used to convert analog to digital. The range of the ADC on PCF8591 is 8 bits, that means the resolution is 2^8=256, and it represents the range (here is 3.3V) will be divided equally to 256 parts. The analog of each range corresponds to one ADC values. So the more bits ADC has, the denser the partition of analog will be, also the higher precision of the conversion will be.



Subsection 1: the analog in rang of 0V-3.3/256 V corresponds to digital 0;

Subsection 2: the analog in rang of 3.3 /256 V-2*3.3 /256V corresponds to digital 1;

…

The following analog will be divided accordingly.

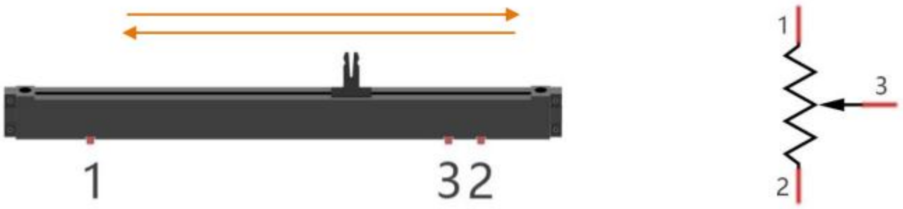
DAC

DAC, that is, Digital-to-Analog Converter, is the reverse process of ADC. The digital I/O port can output high level and low level, but cannot output an intermediate voltage value, which can be solved by DAC. PCF8591 has a DAC output pin with 8-bit accuracy, which can divide VDD (here is 3.3V) into 2^8=256 parts. For example, when the digital quantity is 1, the output voltage value is 3.3/256 *1 V, and when the digital quantity is 128, the output voltage value is 3.3/256 *128=1.65V, the higher accuracy of DAC is, the higher the accuracy of output voltage value is.

Component knowledge

Potentiometer

Potentiometer is a resistive element with three Terminal part and the resistance can be adjusted according to a certain variation. Potentiometer is often made up by resistance and removable brush. When the brush moves along the resistor body, there will be resistance or voltage that has a certain relationship with displacement on the output side (3). Figure shown below is the linear sliding potentiometer and its symbol

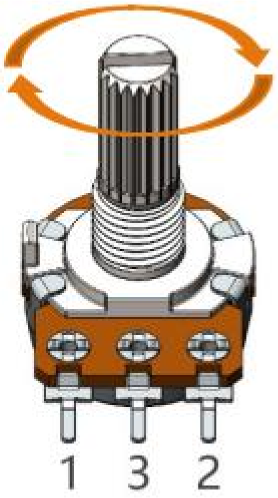


What between potentiometer pin 1 and pin 2 is the resistor body, and pins 3 is connected to brush. When brush moves from pins 1 to pin 2, the resistance between pin 1, and pin 3 will increase up to body resistance linearly, and the resistance between pin 2 and pin 3 will decrease down to 0 linearly.In the circuit. The both sides of resistance body are often connected to the positive and negative electrode of the power. When you slide the brush pin 3, you can get a certain voltage in the range of the power supply

Rotary potentiometer

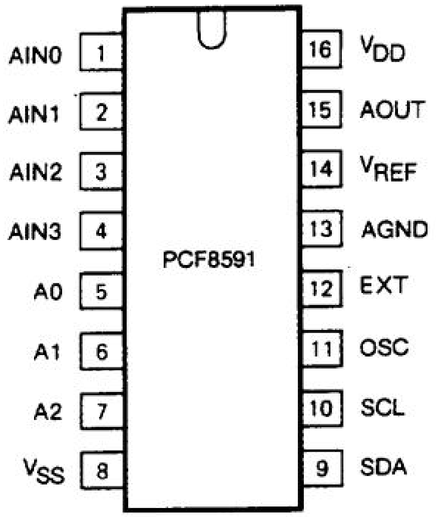
Rotary potentiometer and linear potentiometer have similar function; the only difference is: the resistance is

adjusted through rotating the potentiometer.



PCF8591

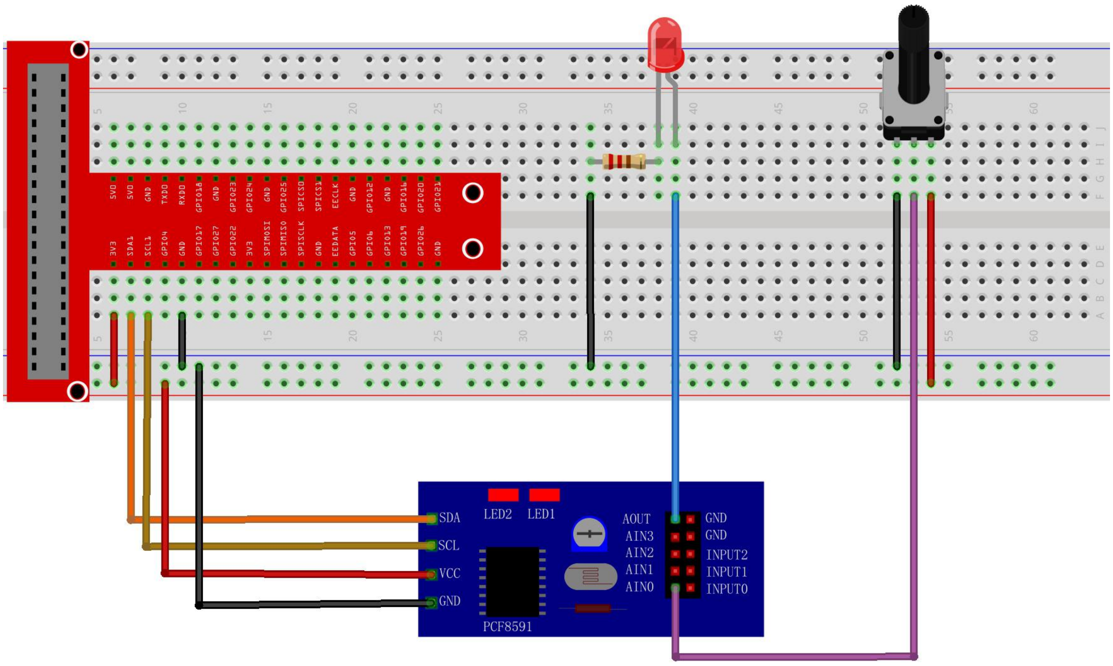
The PCF8591 is a single-chip, single-supply low power 8-bit CMOS data acquisition device with four analog inputs, one analog output and a serial I2C-bus interface. The following table is the pin definition diagram of PCF8591

SYMBOL	PIN	DESCRIPTION	TOP VIEW
AIN0	1	Analog inputs (A/D converter)	
AIN1	2		
AIN2	3		
AIN3	4		
A0	5	Hardware address	
A1	6		
A2	7		
Vss	8	Negative supply voltage	
SDA	9	I2C-bus data input/output	
SCL	10	I2C-bus clock input	
OSC	11	Oscillator input/output	
EXT	12	external/internal switch for oscillator input	
AGND	13	Analog ground	
Vref	14	Voltage reference input	
AOUT	15	Analog output(D/A converter)	
Vdd	16	Positive supply voltage	

I2C communication

I2C(Inter-Integrated Circuit) is a two-wire serial communication mode, which can be used to connection of micro controller and its peripheral equipment. Devices using I2C communication must be connected to the serial data (SDA) line, and serial clock (SCL) line (called I2C bus). Each device has a unique address and can be used as a transmitter or receiver to communicate with devices connected to the bus

Wiring diagram：

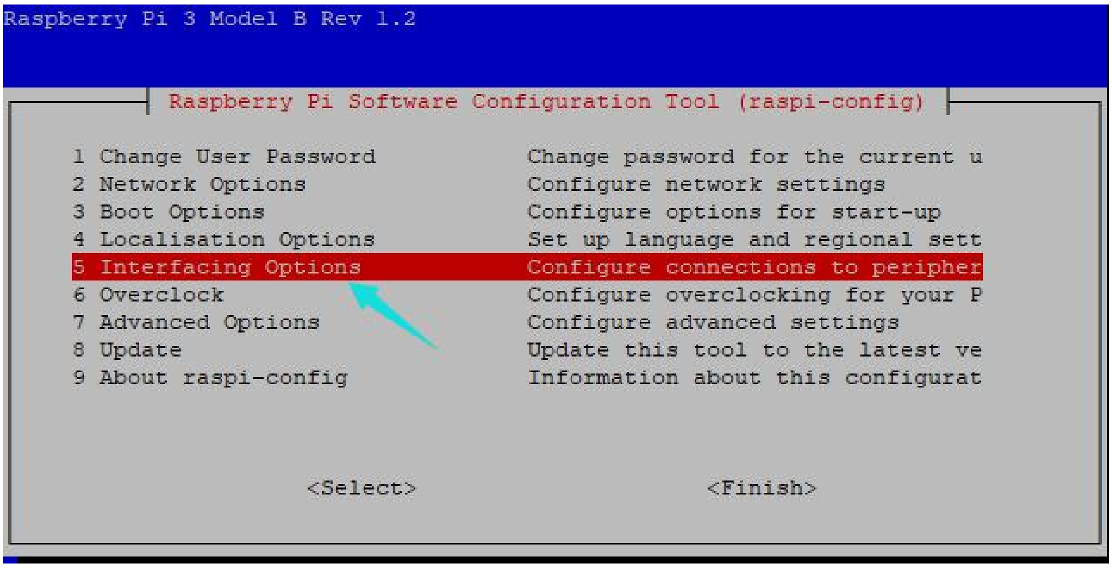


Enable I2C

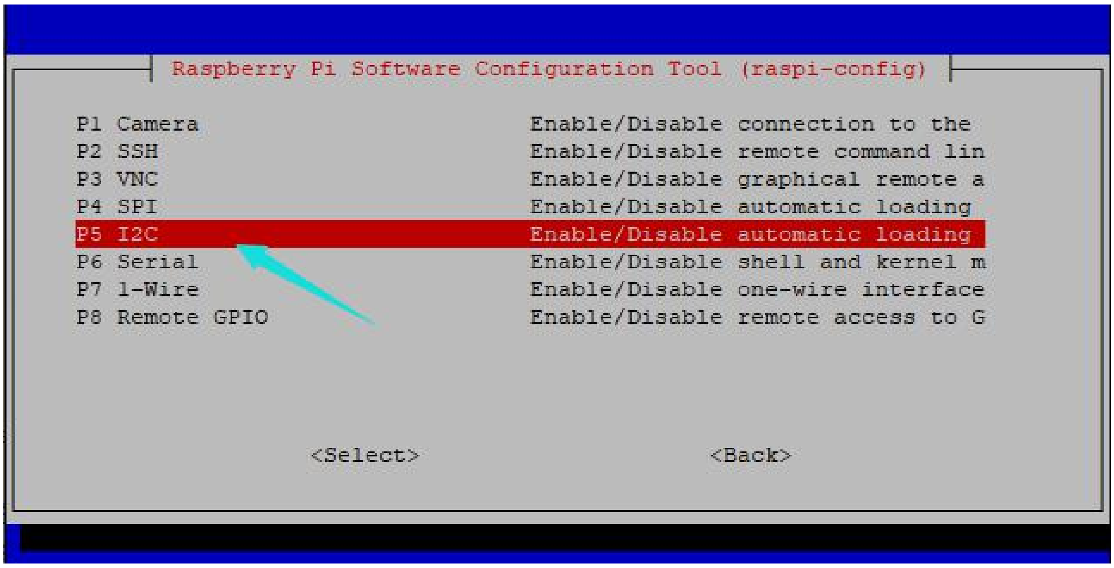
In terminal, type the following command:

```
sudo raspi-config
```

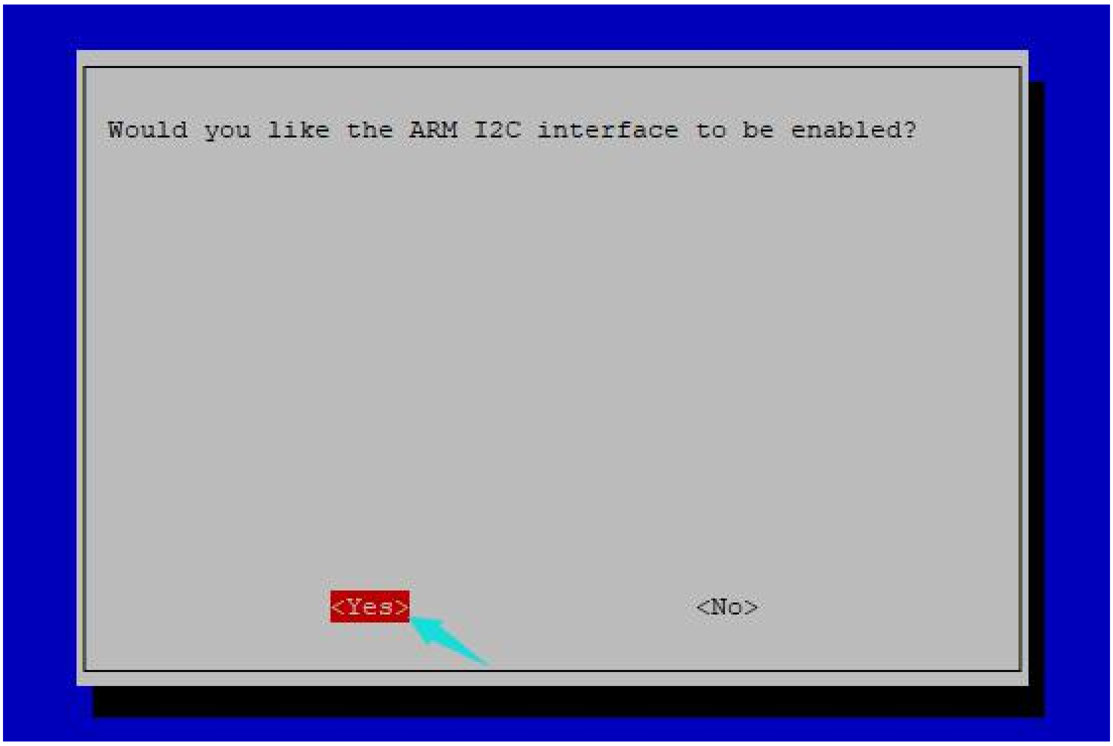
Follow these steps to open I2C



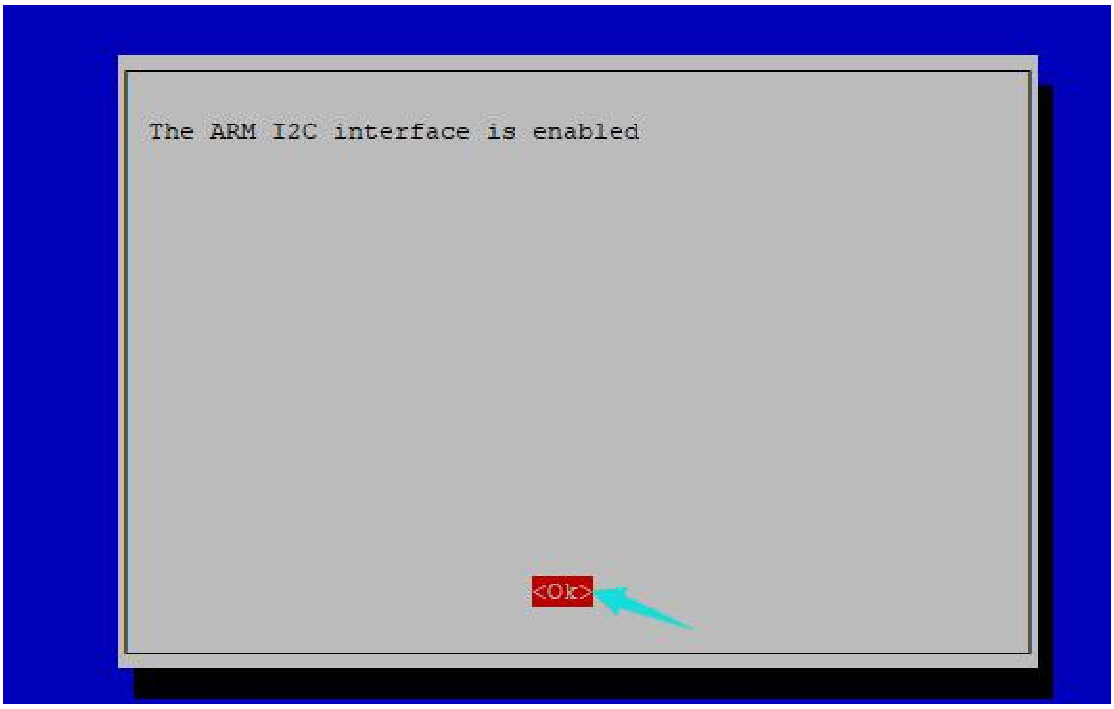
```
Raspberry Pi 3 Model B Rev 1.2

         ┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

    1 Change User Password           Change password for the current u
    2 Network Options                Configure network settings
    3 Boot Options                   Configure options for start-up
    4 Localisation Options           Set up language and regional sett
    5 Interfacing Options            Configure connections to peripher
    6 Overclock                      Configure overclocking for your P
    7 Advanced Options               Configure advanced settings
    8 Update                         Update this tool to the latest ve
    9 About raspi-config             Information about this configurat


                    <Select>                     <Finish>
```



```
         ┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

    P1 Camera                        Enable/Disable connection to the
    P2 SSH                           Enable/Disable remote command lin
    P3 VNC                           Enable/Disable graphical remote a
    P4 SPI                           Enable/Disable automatic loading
    P5 I2C                           Enable/Disable automatic loading
    P6 Serial                        Enable/Disable shell and kernel m
    P7 1-Wire                        Enable/Disable one-wire interface
    P8 Remote GPIO                   Enable/Disable remote access to G


                    <Select>                     <Back>
```



Would you like the ARM I2C interface to be enabled?



The ARM I2C interface is enabled

we need to modify the module’s config file. Type the following command in terminal:

```
sudo nano /etc/modules
```

Add following two lines in modules file if they do not exist:

```
i2c-bcm2708
i2c-dev
```

After the modification, press the key combination “ctrl+X”, then press Y, and finally press the Enter key.

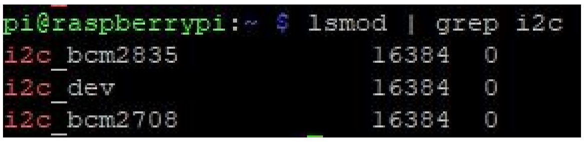
Install smbus and i2c-tools

```
sudo apt-get update
sudo apt-get install -y python-smbus i2c-tools
sudo reboot
```

After rebooting the system，Type a command to check whether the I2C module is started:

```
lsmod | grep i2c
```

If the I2C module has been started, the following content will be shown:

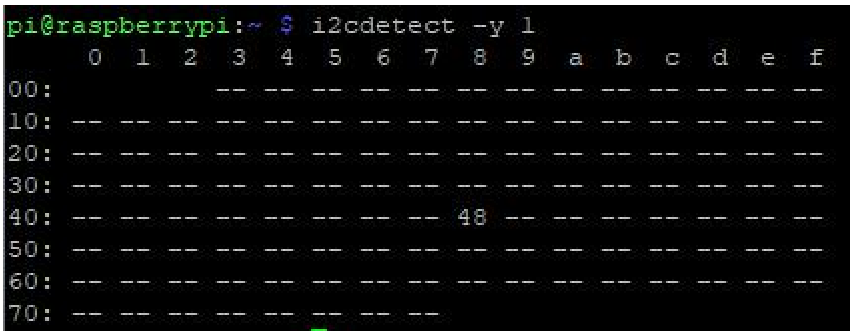


```
pi@raspberrypi:~ $ lsmod | grep i2c
i2c_bcm2835            16384  0
i2c_dev                16384  0
i2c_bcm2708            16384  0
```

You should see i2c_bcm2708 in a list

I2C device address detection:

```
i2cdetect -y 1
```



```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Here 48 (HEX) is the I2C address of PCF8591.

C code：

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define address 0x48       //pcf8591 default address
#define pinbase 64         //any number above 64
#define A0 pinbase + 0
#define A1 pinbase + 1
#define A2 pinbase + 2
#define A3 pinbase + 3

int main(void){
    int value;
    float voltage;
    wiringPiSetup();
    pcf8591Setup(pinbase,address);

    while(1){
        value = analogRead(A0);  //read A0 pin
        analogWrite(pinbase+0,value);
        voltage = (float)value / 255.0 * 3.3;  // calculate voltage
        printf("ADC value : %d  ,\tVoltage : %.2fV\n",value,voltage);
        delay(100);
    }
}
```

Python code：

```
#!/usr/bin/env python3
import smbus
import time
```

```
address = 0x48 #default address of PCF8591
bus=smbus.SMBus(1)
cmd=0x40       #command

def analogRead(chn):#read ADC value,chn:0,1,2,3
    value = bus.read_byte_data(address,cmd+chn)
    return value

def analogWrite(value):#write DAC value
    bus.write_byte_data(address,cmd,value)

def loop():
    while True:
        value = analogRead(0)  #read the ADC value of channel 0
        analogWrite(value)     #write the DAC value
        voltage = value / 255.0 * 3.3  #calculate the voltage value
        print (’ADC Value : %d, Voltage : %.2f’%(value,voltage))
        time.sleep(0.01)

def destroy():
    bus.close()

if __name__ == ’__main__’:
    print (’Program is starting ... ’)
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Experimental results:

In the directory where the code file is located, execute the following command

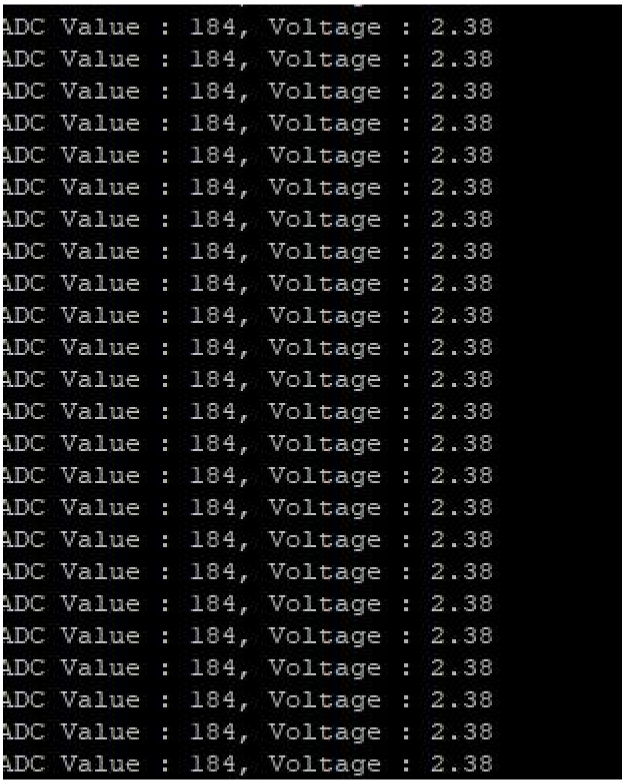
C：

```
gcc -Wall -o adc adc.c -lwiringPi
sudo ./adc
```

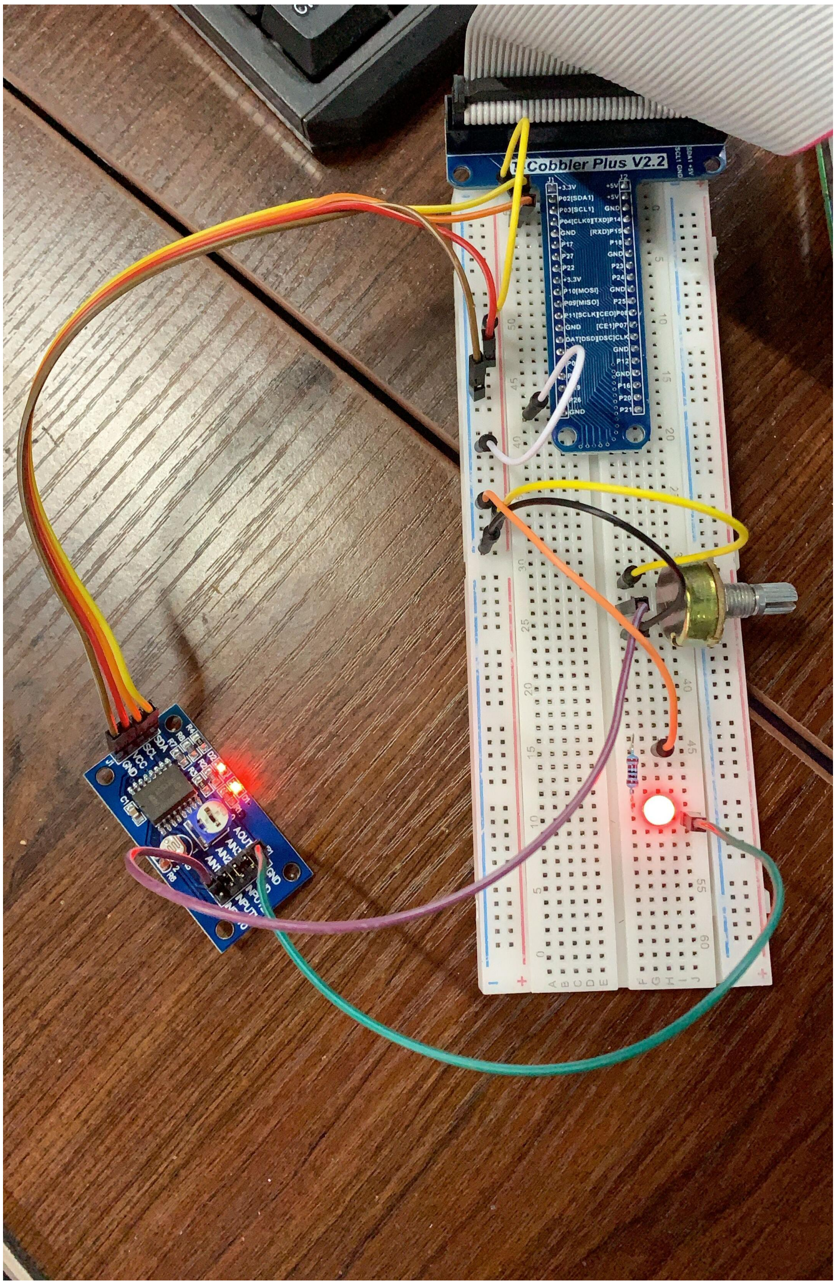
Python:

```
python adc.py
```

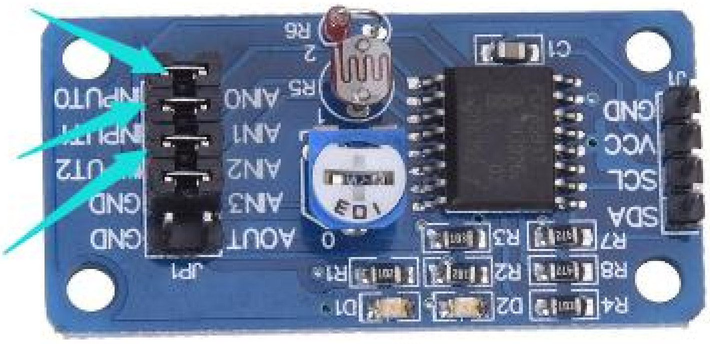
After the program is executed, turn the potentiometer, and the voltage analog value of the potentiometer and the actual voltage value after the conversion will be printed in the terminal. The larger the voltage value, the brighter the LED will be.



```
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
```


Cobbler Plus V2.2

Module introduction:



Black connection cap instructions:

The module has 3 black connection caps, which function as follows:

AIN2 is connected to the black connector cap, select the thermistor access circuit

AIN1 is connected to the black connection cap, select the photoresistor access circuit.

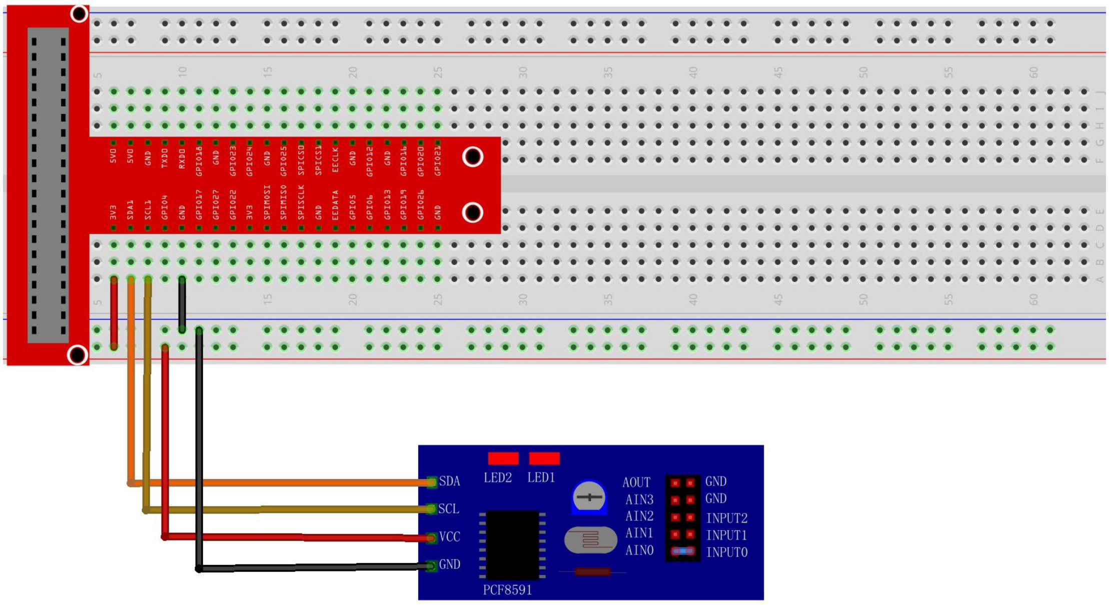
AIN0 is connected to the black connector cap, select 0-5V (VCC=5V) / 0-3.3V(VCC=3.3V) adjustable voltage access circuit

When you need to access other devices, remove the black jumper cap.

The module has a DA output indicator(D2). When the output voltage of the module DA output reaches a certain value, the DA output indicator on the board will be illuminated. The higher the voltage, the more obvious the brightness of the indicator.

Vref is determined by VCC

The following example is the black connection cap access circuit of AIN0(The same is true for using AIN1 and AIN2 external devices. Just change the channel 0 of "analogRead(0)" in the code to 1 or 2.):



Experimental results:

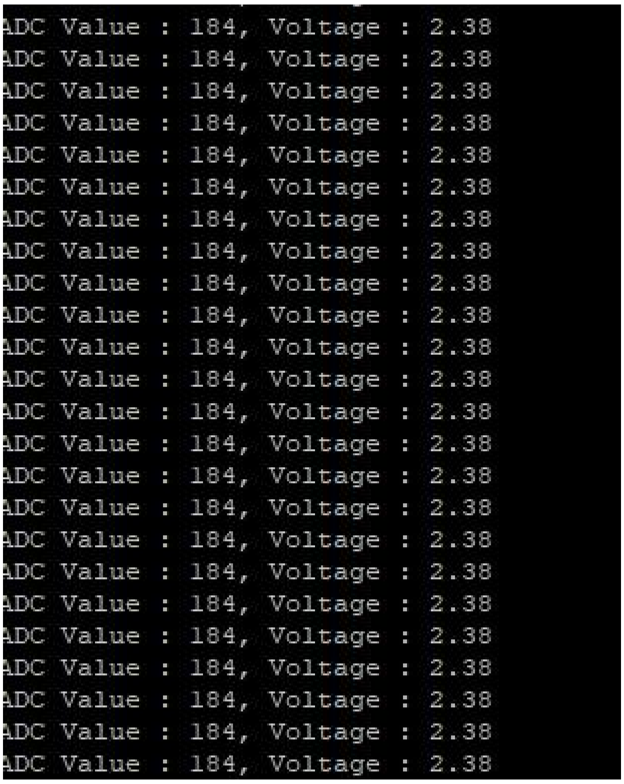
In the directory where the code file is located, execute the following command

C：

```
gcc -Wall -o adc adc.c -lwiringPi
sudo ./adc
```

Python:

```
python adc.py
```



```
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
ADC Value : 184, Voltage : 2.38
```