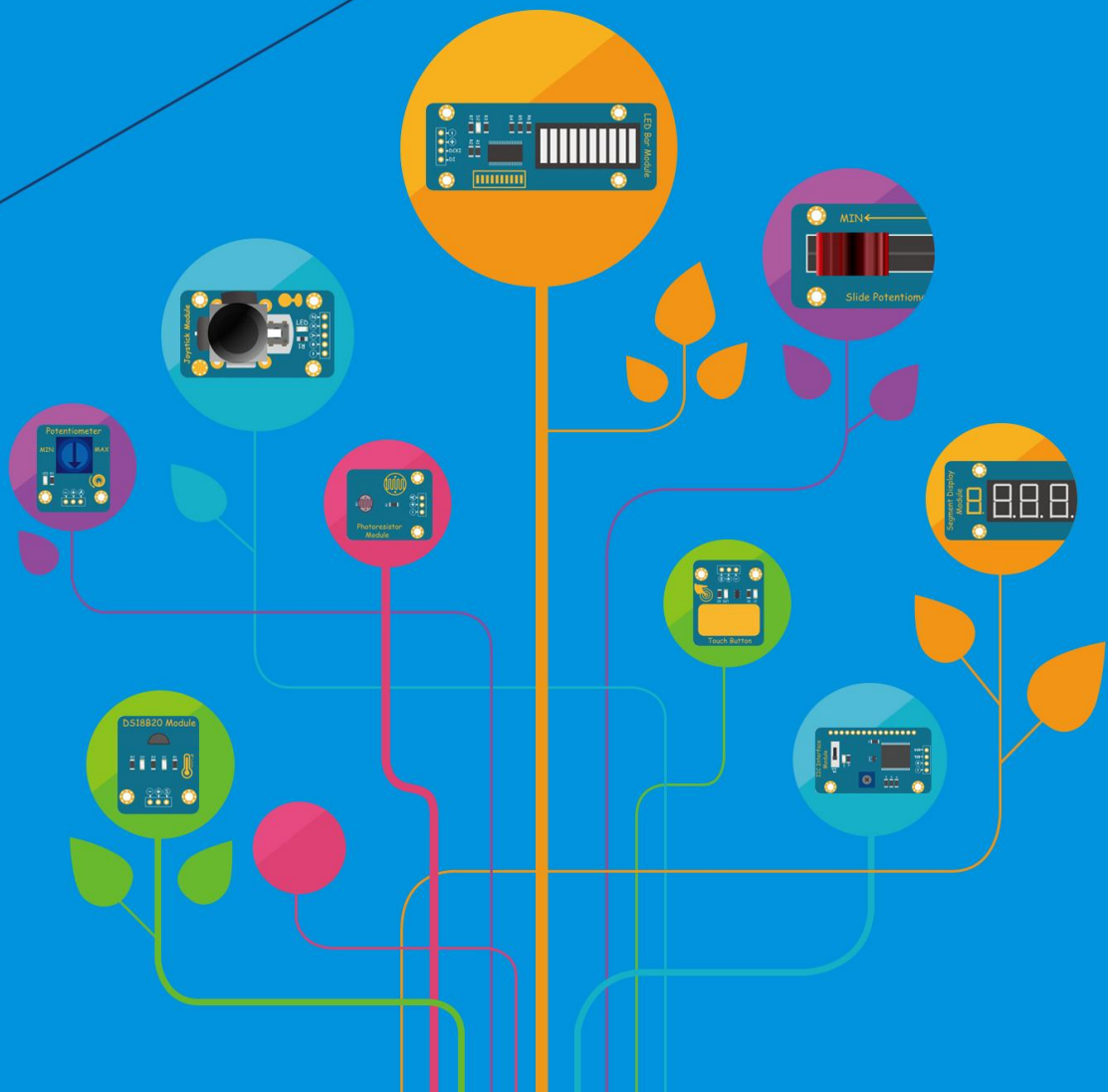



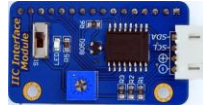








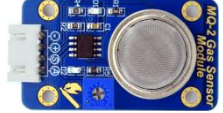
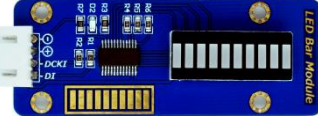














Sensor Kit for Raspberry Pi






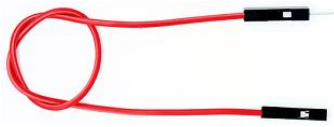
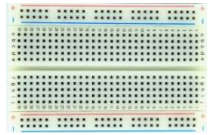
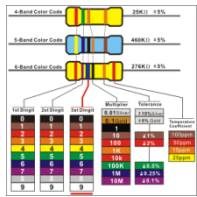
Sharing Perfects Innovation



Package List

No.	Name	Picture	Qty
1	LCD1602		1
2	I2C Interface Module		1
3	DS18B20 Digital temperature Sensor		1
4	Ultrasonic Distance Sensor Module		1
5	ADC0832 Module		1
6	PS2 Joystick Module		1
7	Segment Display Module		1
8	Potentiometer Module		1
9	Slide Potentiometer Module		1
10	Rotary Encoder Module		1
11	MQ-2 Gas Sensor Module		1
12	LED Bar Graph Module		1

13	Active Buzzer Module		1
14	Passive Buzzer Module		1
15	Touch Button Module		1
16	CM Module		1
17	Soil Moisture Sensor Module		1
18	Photoresistor Module		1
19	Analog Temperature Sensor(Thermistor Module)		1
20	RGB LED Module		1
21	Button Module		2
22	LED Module		2
23	GPIO Extension Board		1
24	40P GPIO Cable		1

25	3-Pin Wires		5
26	4-Pin Wires		4
27	5-Pin Wires		3
28	Hookup Wire Set		1
29	2-Pin Female to Female Wires		1
30	Male to Female Jumper Wires		20
31	Breadboard		1
32	Band Resistor Card		1

Preface

About Adeept

Adept is a technical service team of open source software and hardware. Dedicated to applying the Internet and the latest industrial technology in open source area, we strive to provide best hardware support and software service for general makers and electronic enthusiasts around the world. We aim to create infinite possibilities with sharing. No matter what field you are in, we can lead you into the electronic world and bring your ideas into reality.

If you have any problems for learning, please contact us at support@adeept.com, or please ask questions in our forum www.adeept.com. We will do our best to help you solve the problem.

Content

About the Raspberry Pi	7
Raspberry Pi Pin Numbering Introduction	8
Raspberry Pi GPIO Library Introduction	10
How to Use wiringPi and RPi.GPIO	12
Lesson 1 Blinking LED.....	16
Lesson 2 Controlling an LED by Button	19
Lesson 3 Controlling an RGB LED by PWM	22
Lesson 4 Active Buzzer.....	25
Lesson 5 Passive Buzzer	28
Lesson 6 Rotary Encoder	31
Lesson 7 Controlling an LED by Touch Button	35
Lesson 8 Measuring the Temperature via DS18B20	39
Lesson 9 Measuring the Distance	44
Lesson 10 LED Bar Graph	47
Lesson 11 How to Drive the Segment Display	50
Lesson 12 Potentiometer.....	53
Lesson 13 Photoresistor	58
Lesson 14 Thermistor	61
Lesson 15 Soil Moisture Detection	65
Lesson 16 MQ-2 Gas Sensor	68
Lesson 17 PS2 Joystick.....	72
Lesson 18 LCD1602 Display	75
Lesson 19 How to Make a Simple Thermometer(1)	81
Lesson 20 How to Make a Simple Thermometer(2)	83
Lesson 21 Make a Distance Measuring Device.....	86
Lesson 22 How to Make a Simple Voltmeter(1)	89
Lesson 23 How to Make a Simple Voltmeter(2)	92

About the Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work.

Learn more at:

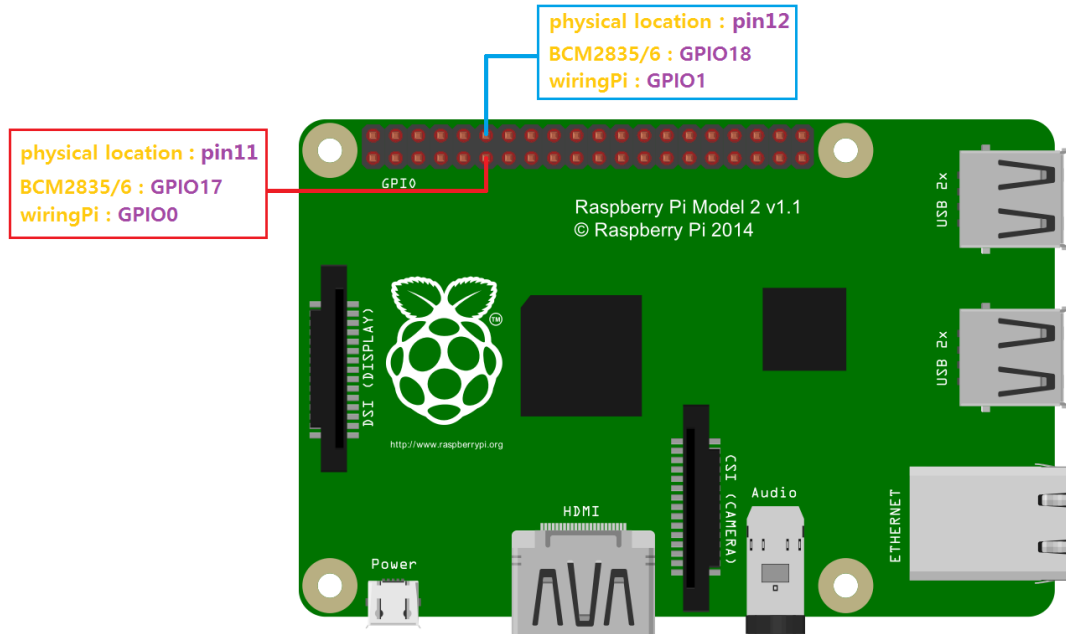
<https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>

Raspberry Pi Pin Numbering Introduction

WiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	WiringPi Pin
—	—	3.3v	1 2	5v	—	—
8	R1:0/R2:2	SDA1	3 4	5v	—	—
9	R1:1/R2:3	SCL1	5 6	0V	—	—
7	4	GPIO7	7 8	TXD	14	15
—	—	0V	9 10	RXD	15	16
0	17	GPIO0	11 12	GPIO1	18	1
2	R1:21/R2:27	GPIO2	13 14	0V	—	—
3	22	GPIO3	15 16	GPIO4	23	4
—	—	3.3v	17 18	GPIO5	24	5
12	10	MOSI	19 20	0V	—	—
13	9	MISO	21 22	GPIO6	25	6
14	11	SCLK	23 24	CE0	8	10
—	—	0V	25 26	CE1	7	11
30	0	SDA0	27 28	SCL0	1	31
21	5	GPIO21	29 30	0V	—	—
22	6	GPIO22	31 32	GPIO26	12	26
23	13	GPIO23	33 34	0V	—	—
24	19	GPIO24	35 36	GPIO27	16	27
25	26	GPIO25	37 38	GPIO28	20	28
		0V	39 40	GPIO29	21	29
WiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	WiringPi Pin

There are three methods for numbering Raspberry Pi's GPIO:

1. Numbering according to the physical location of the pins, from left to right, top to bottom – the left is odd and the right is even.
2. Numbering according the GPIO registers of BCM2835/2836/2837 SOC.
3. Numbering according the GPIO library wiringPi.



Raspberry Pi GPIO Library Introduction

Currently, there are two major GPIO libraries for Raspberry Pi: RPi.GPIO and wiringPi.

RPi.GPIO:

RPi.GPIO is a python module to control Raspberry Pi GPIO channels. For more information, please visit:

<https://pypi.python.org/pypi/RPi.GPIO/>

For examples and documentation:

<http://sourceforge.net/p/raspberry-gpio-python/wiki/Home/>

The RPi.GPIO module is pre-installed in the official Raspbian operating system, thus you can use it directly.

wiringPi:

The wiringPi is a GPIO access library written in C language for BCM2835/6/7 SOC used in the Raspberry Pi. It's released under the GNU LGPLv3 license and usable from C and C++ and many other languages with suitable wrappers. It's designed familiar to people who have practiced the wiring system in the Arduino software.

For more information about wiringPi, please visit: <http://wiringpi.com/>

Install wiringPi:

Step 1: Get the source code

```
$ sudo git clone git://git.drogon.net/wiringPi
```

Step 2: Compile and install

```
$ cd wiringPi
```

```
$ git pull origin
```

```
$ sudo ./build
```

Press Enter and the script *build* will automatically compile wiringPi source code and then install it to the Raspberry Pi.

Next, verify whether the wiringPi is installed successfully or not:

wiringPi includes a command-line utility `gpio` which can be used to program and set up the GPIO pins. You can use it to read and write the pins or even control them from shell scripts.

You can verify whether the wiringPi is installed successfully or not by the following commands:

`$ sudo gpio -v`

```
pi@raspberrypi ~ $ sudo gpio -v
gpio version: 2.26
Copyright (c) 2012-2015 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
Type: Model 2, Revision: 1.1, Memory: 1024MB, Maker: Sony
pi@raspberrypi ~ $
```

`$ sudo gpio readall`

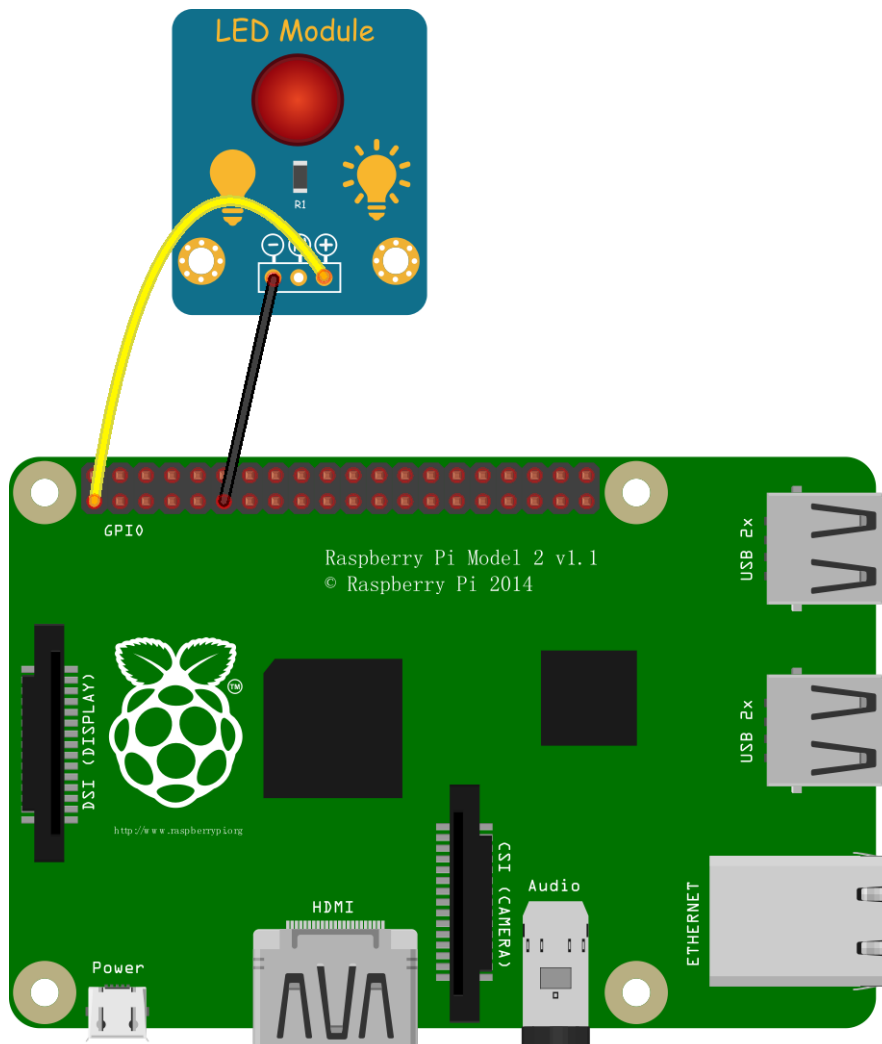
```
pi@raspberrypi ~ $ sudo gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi |   Name   | Mode | V | Physical | V | Mode |   Name   | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2   | 8   | SDA.1    | ALT0 | 1 | 3        | 4 |      | 5v       |     |     |
| 3   | 9   | SCL.1    | ALT0 | 1 | 5        | 6 |      | 5V       |     |     |
| 4   | 7   | GPIO. 7  | IN    | 1 | 7        | 8 | 1 ALT0 | TxD      | 15  | 14  |
|     |     | 0v       |      |   | 9        | 10| 1 ALT0 | RxD      | 16  | 15  |
| 17  | 0   | GPIO. 0  | IN    | 0 | 11       | 12| 0 IN   | GPIO. 1  | 1   | 18  |
| 27  | 2   | GPIO. 2  | IN    | 0 | 13       | 14| 0      | 0v       |     |     |
| 22  | 3   | GPIO. 3  | IN    | 0 | 15       | 16| 0 IN   | GPIO. 4  | 4   | 23  |
|     |     | 3.3v     |      |   | 17       | 18| 0 IN   | GPIO. 5  | 5   | 24  |
| 10  | 12  | MOSI     | ALT0 | 0 | 19       | 20| 0      | 0v       |     |     |
| 9   | 13  | MISO     | ALT0 | 0 | 21       | 22| 0 IN   | GPIO. 6  | 6   | 25  |
| 11  | 14  | SCLK     | ALT0 | 0 | 23       | 24| 1 ALT0 | CE0      | 10  | 8   |
|     |     | 0v       |      |   | 25       | 26| 1 ALT0 | CE1      | 11  | 7   |
| 0   | 30  | SDA.0    | IN    | 1 | 27       | 28| 1 IN   | SCL.0    | 31  | 1   |
| 5   | 21  | GPIO.21  | IN    | 1 | 29       | 30|      | 0v       |     |     |
| 6   | 22  | GPIO.22  | IN    | 1 | 31       | 32| 0 IN   | GPIO.26  | 26  | 12  |
| 13  | 23  | GPIO.23  | IN    | 0 | 33       | 34|      | 0v       |     |     |
| 19  | 24  | GPIO.24  | IN    | 0 | 35       | 36| 0 IN   | GPIO.27  | 27  | 16  |
| 26  | 25  | GPIO.25  | IN    | 0 | 37       | 38| 0 IN   | GPIO.28  | 28  | 20  |
|     |     | 0v       |      |   | 39       | 40| 0 IN   | GPIO.29  | 29  | 21  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi |   Name   | Mode | V | Physical | V | Mode |   Name   | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
pi@raspberrypi ~ $
```

If the information above is shown, it indicates that the wiringPi has been installed successfully.

How to Use wiringPi and RPi.GPIO

For how to use the wiringPi C library and RPi.GPIO Python module, here we take blinking an LED for example.

Step 1: Build the circuit according to the following schematic diagram



For Python users:

Step 2: Create a file named *led.py*

```
$ sudo touch led.py
```

```
pi@raspberrypi /home $ ls
pi
pi@raspberrypi /home $ sudo touch led.py
pi@raspberrypi /home $ ls
led.py  pi
pi@raspberrypi /home $
```

Step 3: Open the file *led.py* with vim or nano

```
$ sudo vim led.py
```

Write the following source code, then save and exit.

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import time

Led = 11      # pin11

def setup():
    GPIO.setmode(GPIO.BOARD)      # Numbering according to the physical location
    GPIO.setup(Led, GPIO.OUT)     # Set pin mode as output
    GPIO.output(Led, GPIO.HIGH)   # Output high level(+3.3V) to off the led

def loop():
    while True:
        print '...led on'
        GPIO.output(Led, GPIO.LOW) # led on
        time.sleep(0.5)
        print 'led off...'
        GPIO.output(Led, GPIO.HIGH) # led off
        time.sleep(0.5)

def destroy():
    GPIO.output(Led, GPIO.HIGH)     # led off
    GPIO.cleanup()                 # Release resource

if __name__ == '__main__':        # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:     # Press 'Ctrl+C' to end the program
        destroy()
```

Step 4: Run

```
$ sudo python led.py
```

```
pi@raspberrypi /home $ ls
led.py pi
pi@raspberrypi /home $ sudo python led.py
...led on
led off...
...led on
led off...
...led on
led off...
```

Now you should see the LED blinking. Press 'Ctrl+C' and the program execution will be terminated.

For C language users:

Step 2: Create a file named *led.c*

```
$ sudo touch led.c
```

```
pi@raspberrypi /home $ ls
led.py pi
pi@raspberrypi /home $ sudo touch led.c
pi@raspberrypi /home $
```

Step 3: Open the file *led.c* with vim or nano

```
$ sudo vim led.c
```

Write the following source code, then save and exit.

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

#define Led    0 //wiringPi GPIO0, pin11

int main(void)
{
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !\n");
        return -1;
    }

    pinMode(Led, OUTPUT);

    while(1){
        digitalWrite(Led, LOW); //led on
        printf("led on...\n");
        delay(500);
        digitalWrite(Led, HIGH); //led off
        printf("...led off\n");
        delay(500);
    }

    return 0;
}
```

Step 4: Compile the code

```
$ sudo gcc led.c -lwiringPi
```

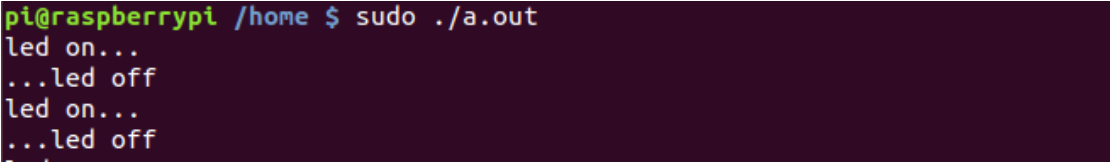
```
pi@raspberrypi /home $ ls
led.c led.py pi
pi@raspberrypi /home $ sudo gcc led.c -lwiringPi
pi@raspberrypi /home $
```

After the command is executed, you'll find a file named *a.out* appear in the current directory. It is an executable program.

```
pi@raspberrypi /home $ ls
a.out led.c led.py pi
pi@raspberrypi /home $
```

Step 5: Run

```
$ sudo ./a.out
```



```
pi@raspberrypi /home $ sudo ./a.out
led on...
...led off
led on...
...led off
```

Now you should see that the LED is blinking. Press 'Ctrl+C' and the program execution will be terminated.

Resources:

<http://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/>

<http://wiringpi.com/reference/>

NOTE:

Before you continue learning, please copy the source code provided with the kit to your Raspberry Pi's `/home/` directory, or download the source code directly from our github repository:

Python and C Language Source Code:

`git clone https://github.com/adeept/Adeept_Compact_Sensor_Kit_for_RPi`

Lesson 1 Blinking LED

Introduction

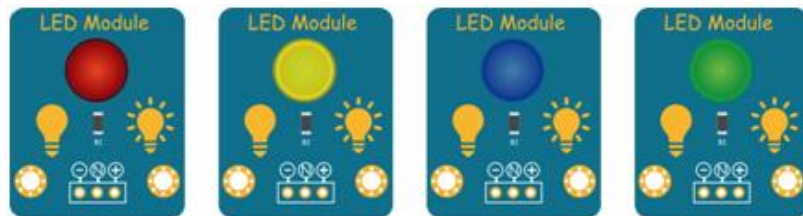
LED is usually used in office lighting, furniture, decoration, sign board, streetlight, etc.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * LED Module
- 1 * 3-Pin Wires

Experimental Principle

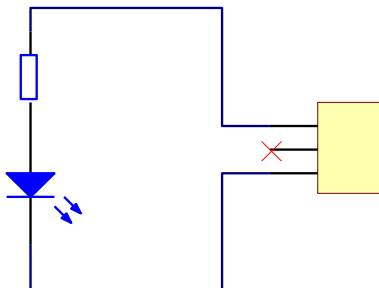
The Fritzing images:



The Physical picture:



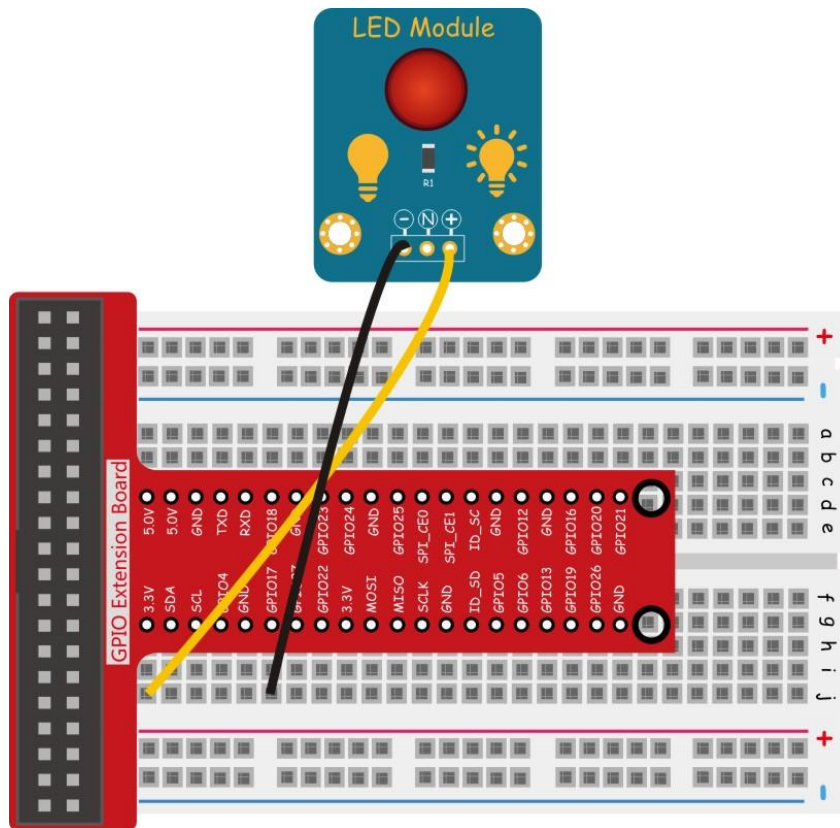
The schematic diagram:



In this experiment, we make the pin 11 of the Raspberry Pi output High/Low by programming, to control the LED to blink in a certain frequency.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/01_LED/blinkingLed.c)

Step 3: Compile

```
$ sudo gcc blinkingLed.c -o led -lwiringPi
```

Step 4: Run

```
$ sudo ./led
```

For Python users:

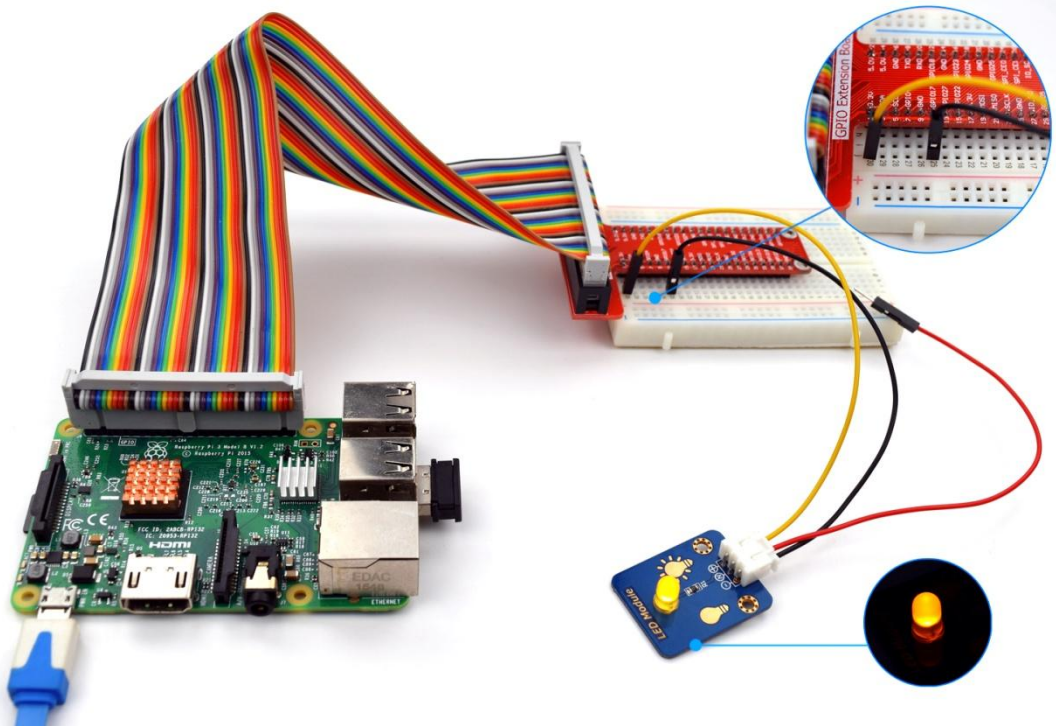
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/01_LED/blinkingLed_1.py)

Step 3: Run

```
$ sudo python blinkingLed_1.py
```

Now you can see the LED blinking.



Lesson 2 Controlling an LED by Button

Introduction

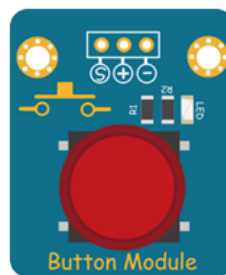
A button is an electronic switch and usually used for device control.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Button Module
- 1 * LED Module
- 2 * 3-Pin Wires

Experimental Principle

The Fritzing image:



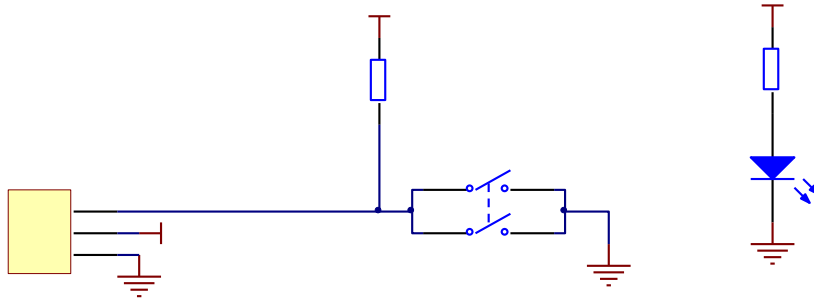
Pin definition:

S	Output
+	VCC
-	GND

The Physical picture:



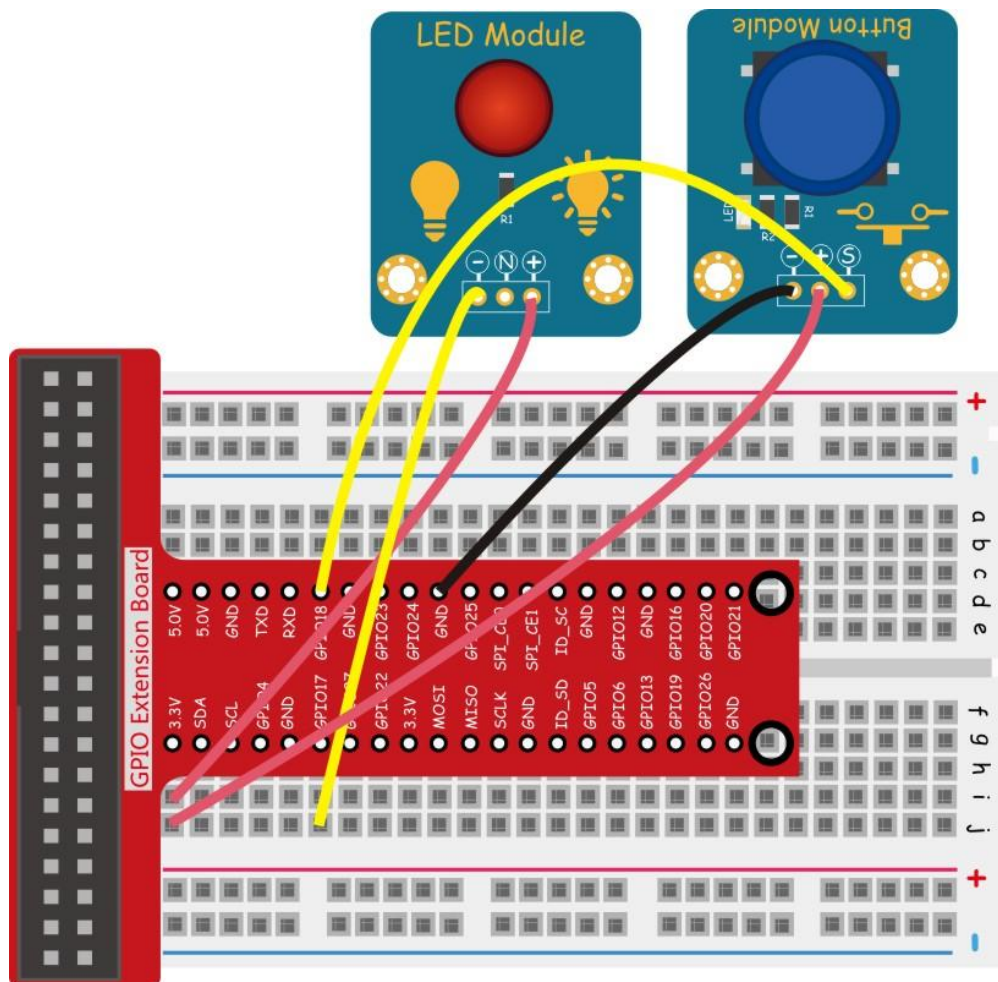
The schematic diagram:



In this experiment, we detect the High or Low level of pin 12 of the Raspberry Pi and then control the LED connected to pin 11 accordingly.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/02_btnAndLed/btnAndLed_1.c)

Step 3: Compile

```
$ sudo gcc btnAndLed_1.c -o btnAndLed -lwiringPi
```

Step 4: Run

```
$ sudo ./btnAndLed
```

For Python users:

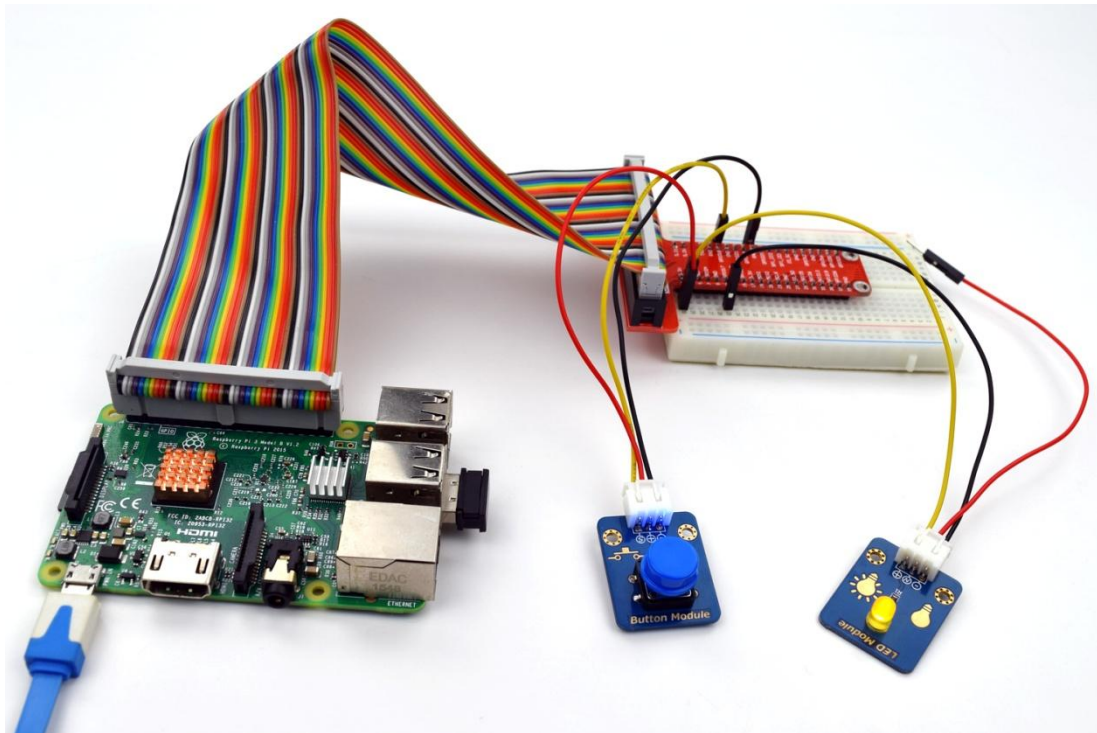
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/02_btnAndLed/btnAndLed_1.py)

Step 3: Run

```
$ sudo python btnAndLed_1.py
```

Press the button and you can see the LED toggle between on and off.



Lesson 3 Controlling an RGB LED by PWM

Introduction

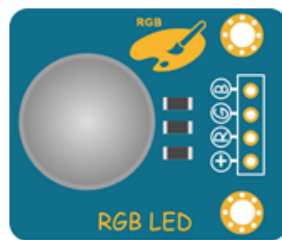
RGB LED is designed based on the principle of three primary colors. In an RGB LED, three LEDs in red, green, and blue respectively are packaged together, thus by controlling the brightness of three LEDs, making the RGB LED flash multiple colors.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * RGB LED Module
- 1 * 4-Pin Wires

Experimental Principle

The Fritzing image:



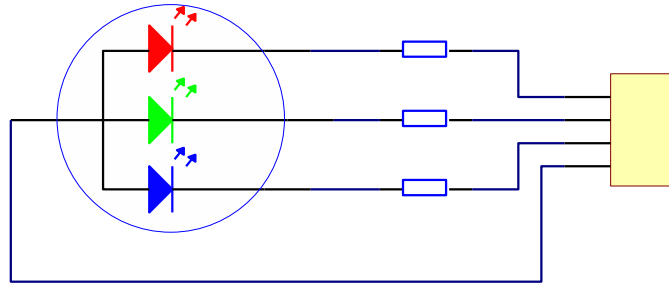
The Physical picture:



Pin definition:

B	Blue Channel
G	Green Channel
R	Red Channel
+	VCC

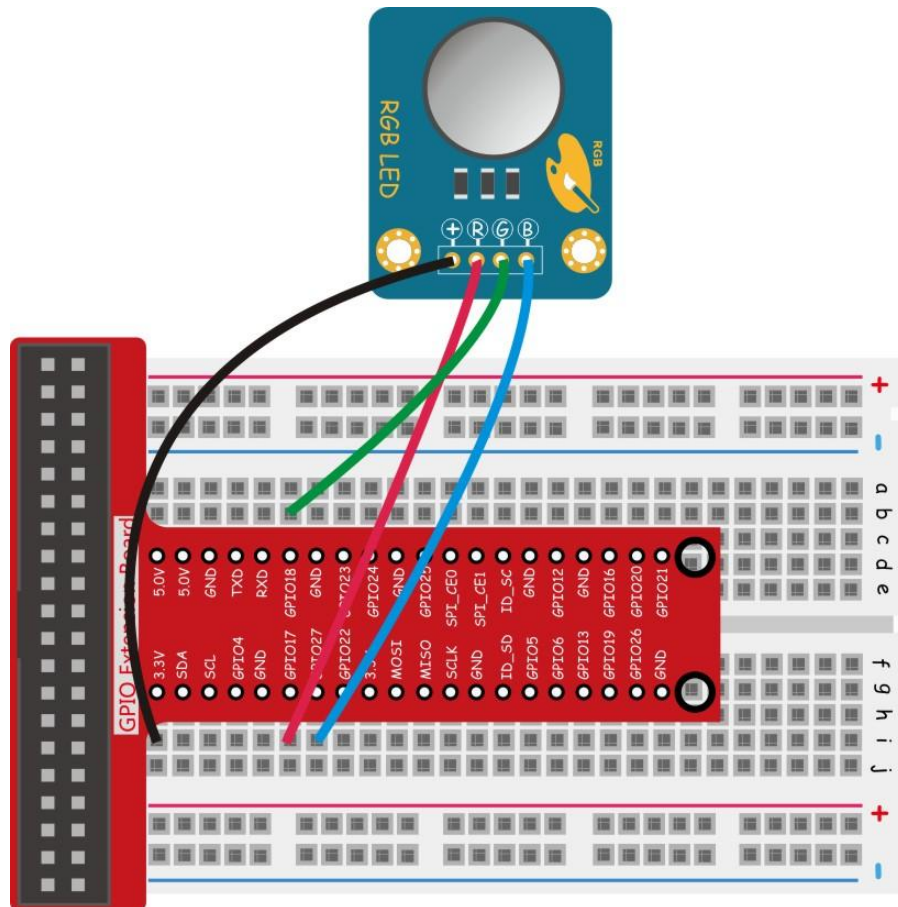
The schematic diagram:



In this experiment, we make the pin 11, 12, and 13 of the Raspberry Pi output PWM (pulse-width modulation) signals by programming, to make the RGB LED flash different colors.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/03_rgbLed/rgbLed.c)

Step 3: Compile

```
$ sudo gcc rgbLed.c -o rgbLed -lwiringPi -lpthread
```

Step 4: Run

```
$ sudo ./rgbLed
```

For Python users:

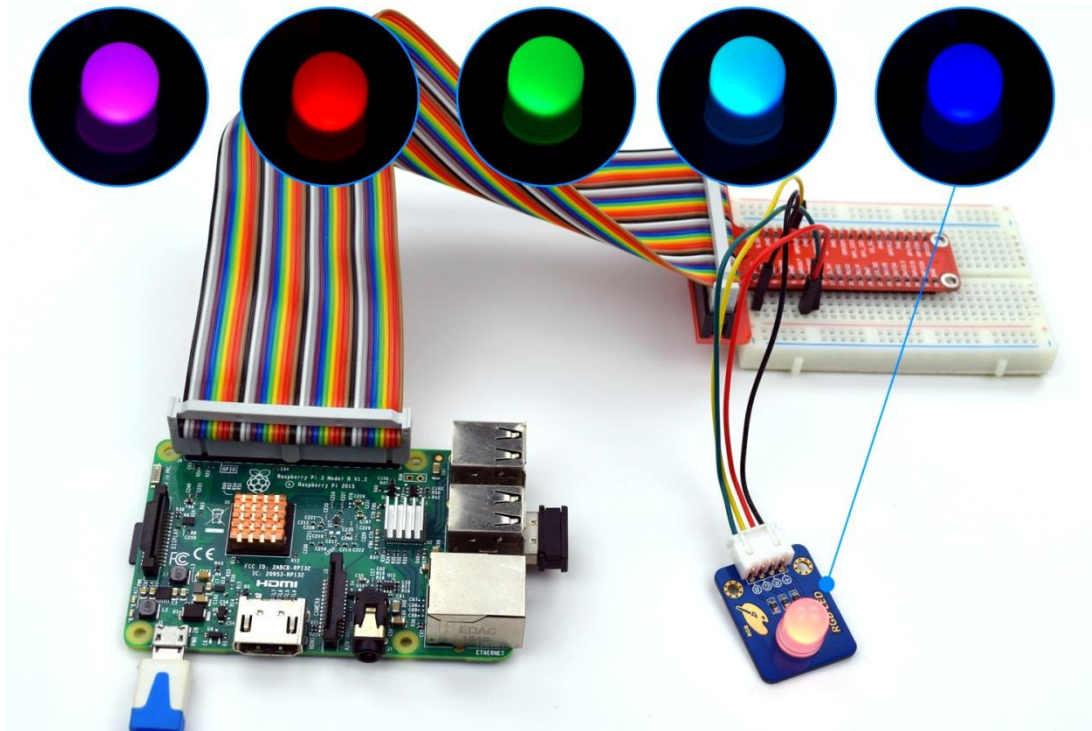
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Compact_Sensor_Kit_for_RPi/Python/03_rgbLed.py)

Step 3: Run

```
$ sudo python 03_rgbLed.py
```

Now you can see the RGB LED flash different colors alternately.



Lesson 4 Active Buzzer

Introduction

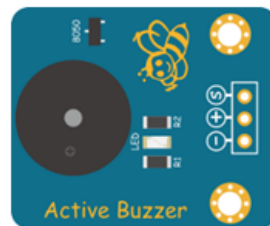
Buzzers are a type of integrated electronic alarm devices and powered by DC supply. They are widely applied for sound producing in devices such as computer, printer, duplicator, alarm, electronic toy, vehicle electronic equipment, phone, and timer and so on. Active buzzers can make sounds constantly when connected with a 5V DC supply. This Active Buzzer module is connected to a digital pin of the Raspberry Pi. When the pin outputs High level, the buzzer will beep; when the pin gives Low, it stays dumb.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Active Buzzer Module
- 1 * 3-Pin Wires

Experimental Principle

The Fritzing image:



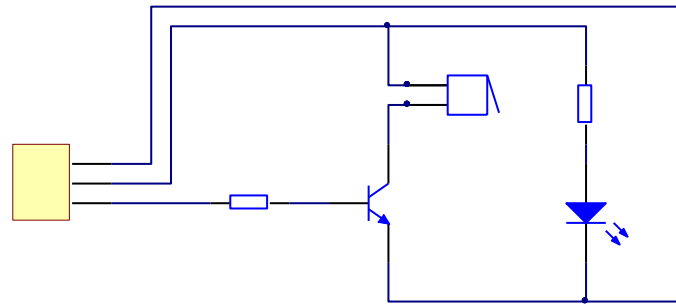
The Physical picture:



Pin definition:

S	Input
+	VCC
-	GND

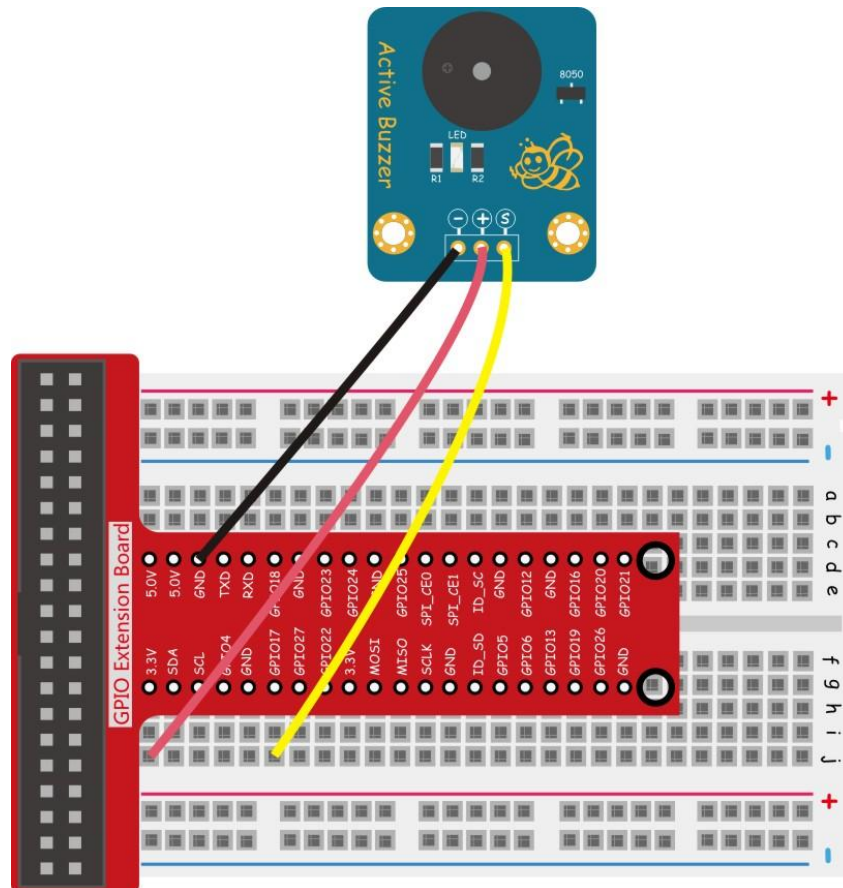
The schematic diagram:



In this experiment, by programming the Raspberry Pi, we make the pin 11 of the Raspberry Pi output High and Low alternately, so the active buzzer makes sounds accordingly.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/04_activeBuzzer/buzzer.c)

Step 3: Compile

```
$ sudo gcc buzzer.c -o buzzer -lwiringPi
```

Step 4: Run

```
$ sudo ./buzzer
```

For Python users:

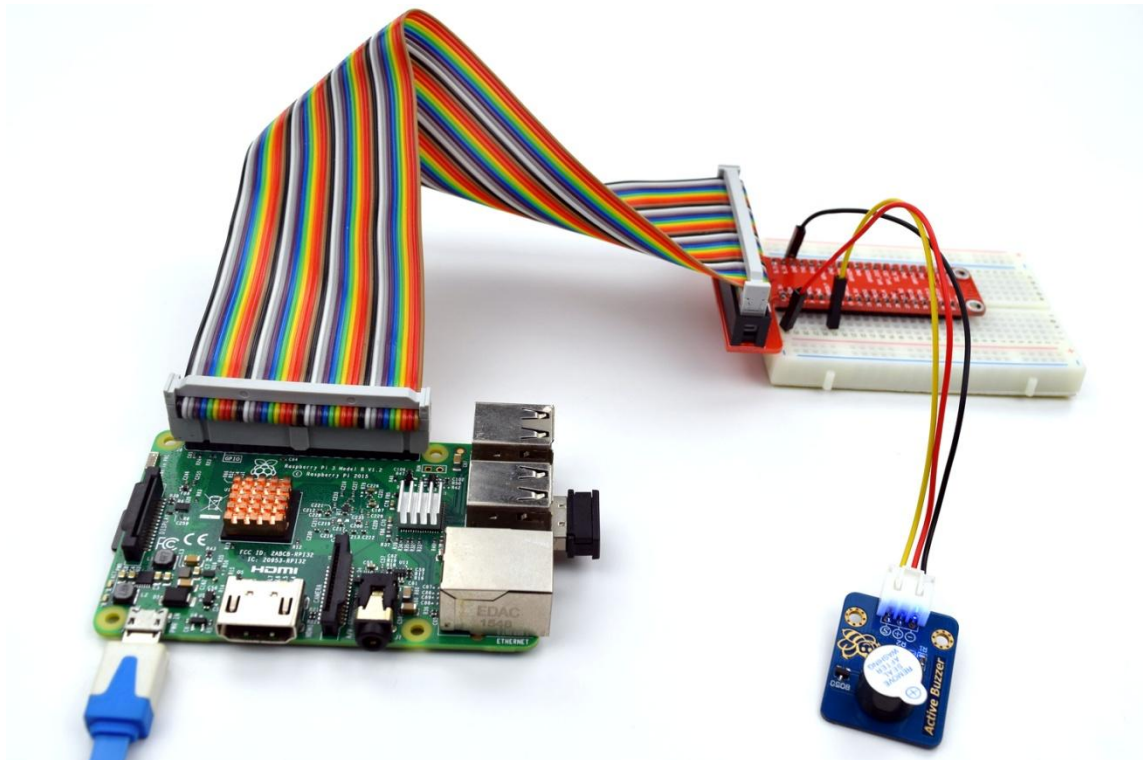
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/04_activeBuzzer.py)

Step 3: Run

```
$ sudo python 04_activeBuzzer.py
```

Now you can hear the active buzzer beeps like the sound of "Di Di".



Lesson 5 Passive Buzzer

Introduction

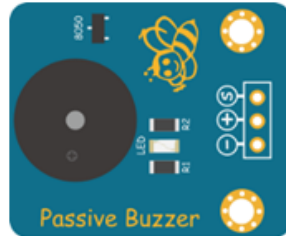
The difference between an active buzzer and a passive one radically lies in the requirement for input signals. The ideal signals for active buzzers are direct currents(DC), usually marked with VCC or VDD. Inside them there are a simple oscillation circuit that can convert constant direct currents into pulse signal of a certain frequency, causing magnetic fields alternation and then Mo sheet vibrating and making sounds. On the other hand, there is no driving circuit in a passive buzzer. So the ideal signal for passive buzzer is square wave. If DC is given, it will not respond since the magnetic field is unchanged, the vibration plate cannot vibrate and produce sounds.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Passive Buzzer Module
- 1 * 3-Pin Wires

Experimental Principle

The Fritzing image:



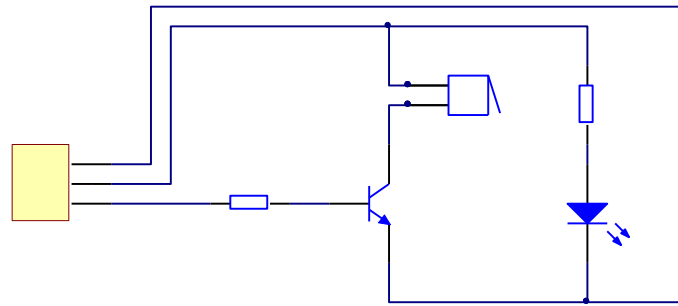
The Physical picture:



Pin definition:

S	Input
+	VCC
-	GND

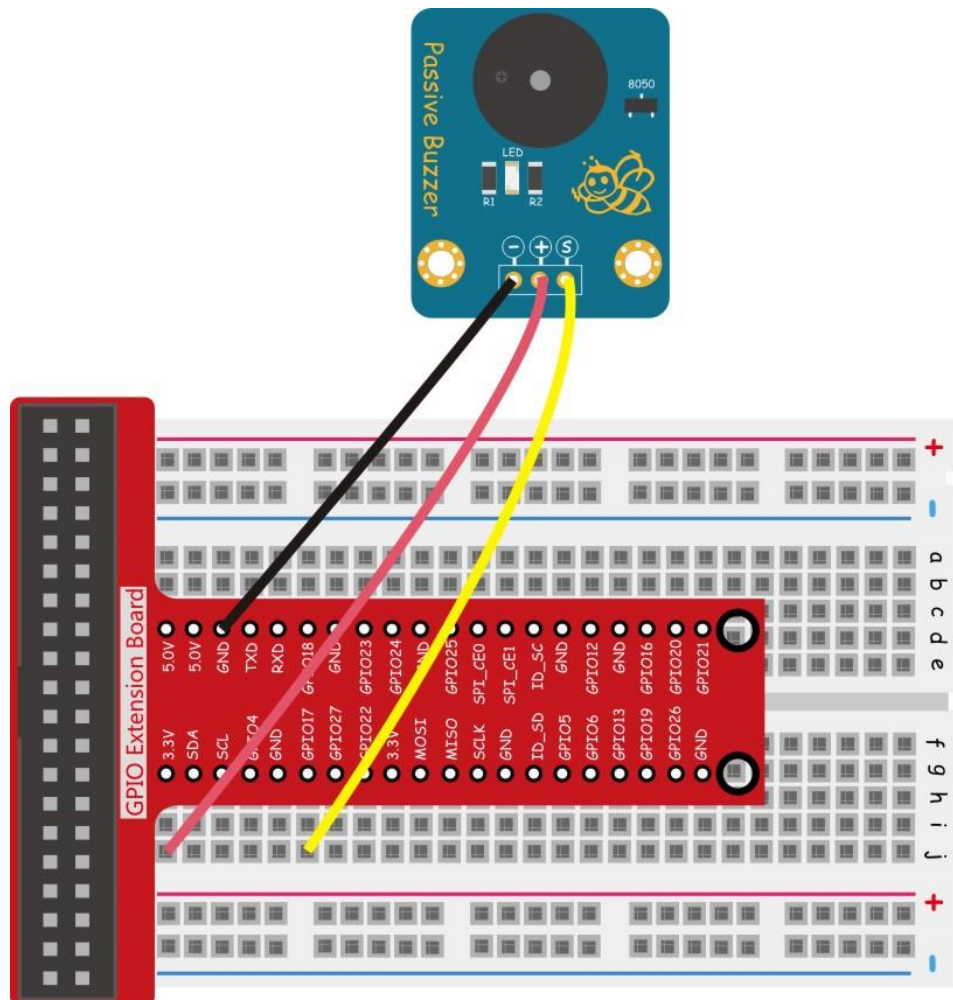
The schematic diagram:



In this experiment, by programming the Raspberry Pi, we make the pin 11 of the Raspberry Pi output square waves of different frequencies alternately, thus driving the passive buzzer to play music.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/05_passiveBuzzer/passiveBuzzer.c)

Step 3: Compile

```
$ sudo gcc passiveBuzzer.c -o passiveBuzzer -lwiringPi -lpthread
```

Step 4: Run

```
$ sudo ./passiveBuzzer
```

For Python users:

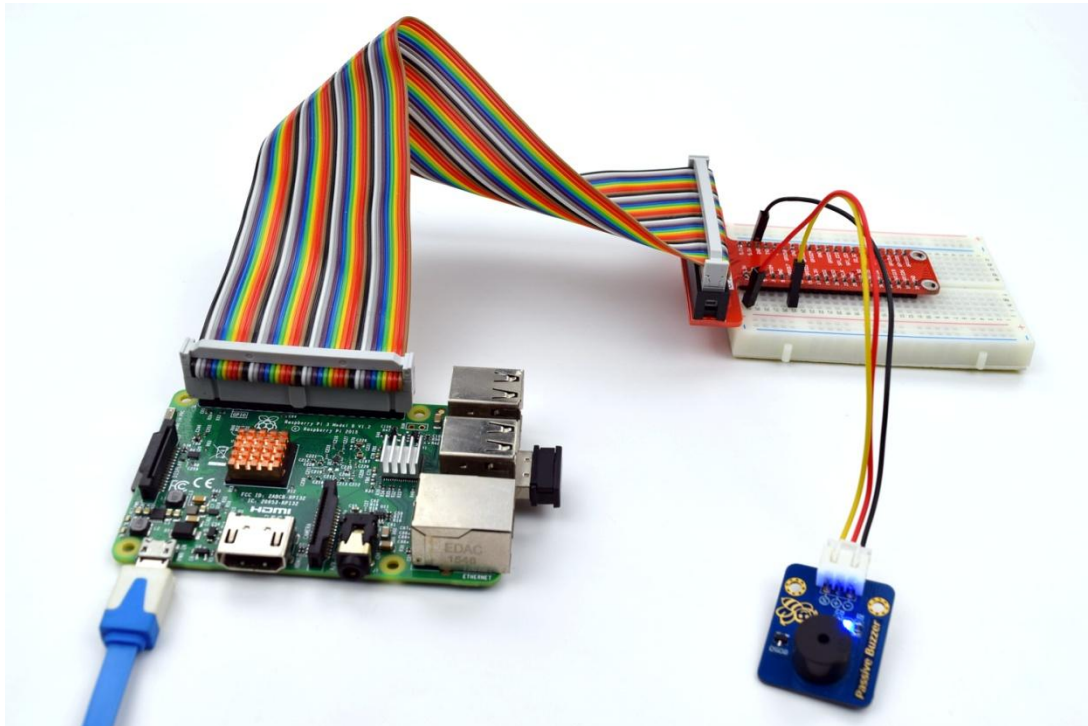
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/05_passiveBuzzer.py)

Step 3: Run

```
$ sudo python 05_passiveBuzzer.py
```

Now you can hear the passive buzzer play music.



Lesson 6 Rotary Encoder

Introduction

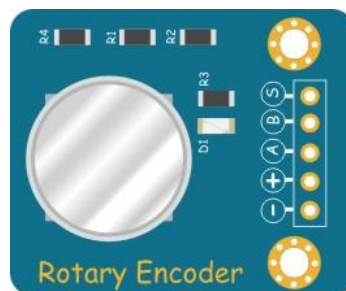
Rotary encoder switch, or small rotary encoder, is a switch electronic component that has a set of regular and strictly-sequenced pulses. The module supports functions such as increase, decrease, turn pages, etc., by collaboration with a microcontroller. For example, in daily life you can see page turning of the mouse, menu selection, volume adjustment of speakers, temperature adjustment of toaster, frequency adjustment of medical equipment, etc.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Rotary Encoder Module
- 1 * 5-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:

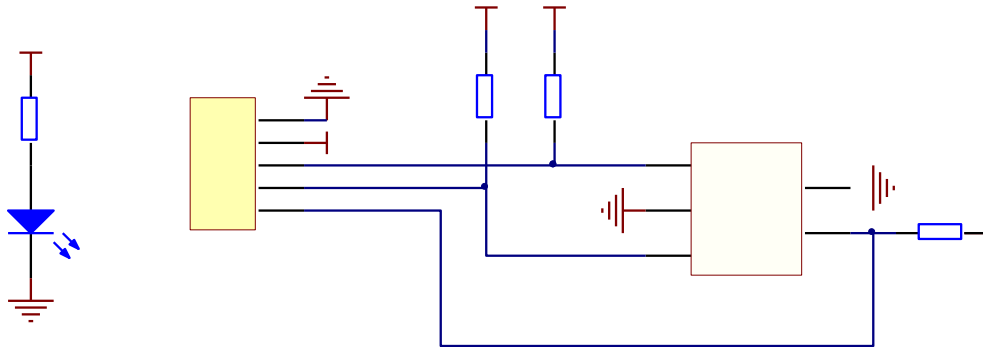


Pin definition:

S	Output
B	Output
A	Output

+	VCC
-	GND

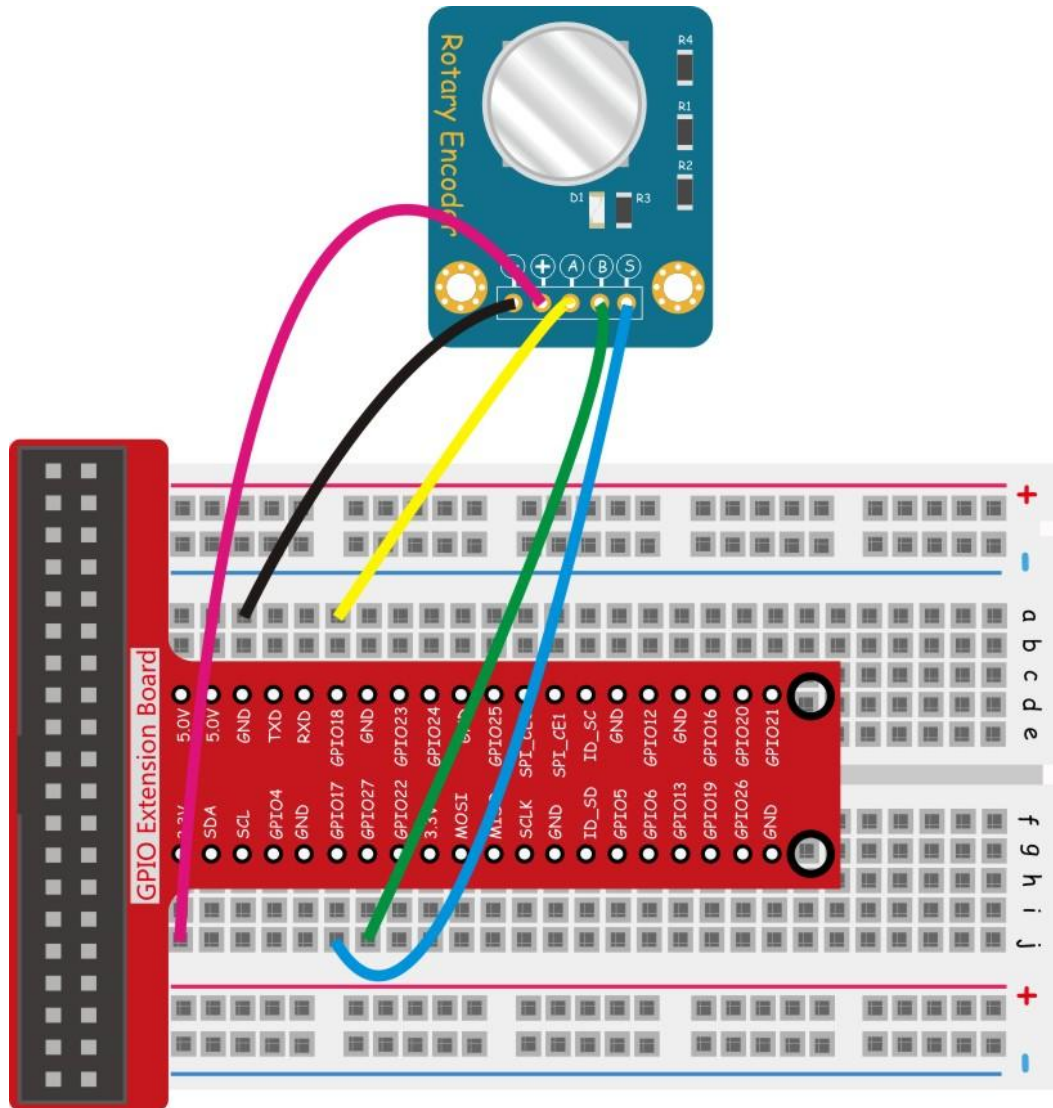
The schematic diagram:



In this experiment, by programming the Raspberry Pi, we change a value by reading the status of the Rotary Encoder. When we turn the knob of the Rotary Encoder clockwise, the value on the terminal will increase; when we turn the knob counterclockwise, the value will decrease. When we press down the switch, the value will be zeroed out.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/06_RotaryEncoder/rotaryEncoder.c)

Step 3: Compile

```
$ sudo gcc rotaryEncoder.c -o rotaryEncoder -lwiringPi
```

Step 4: Run

```
$ sudo ./rotaryEncoder
```

For Python users:

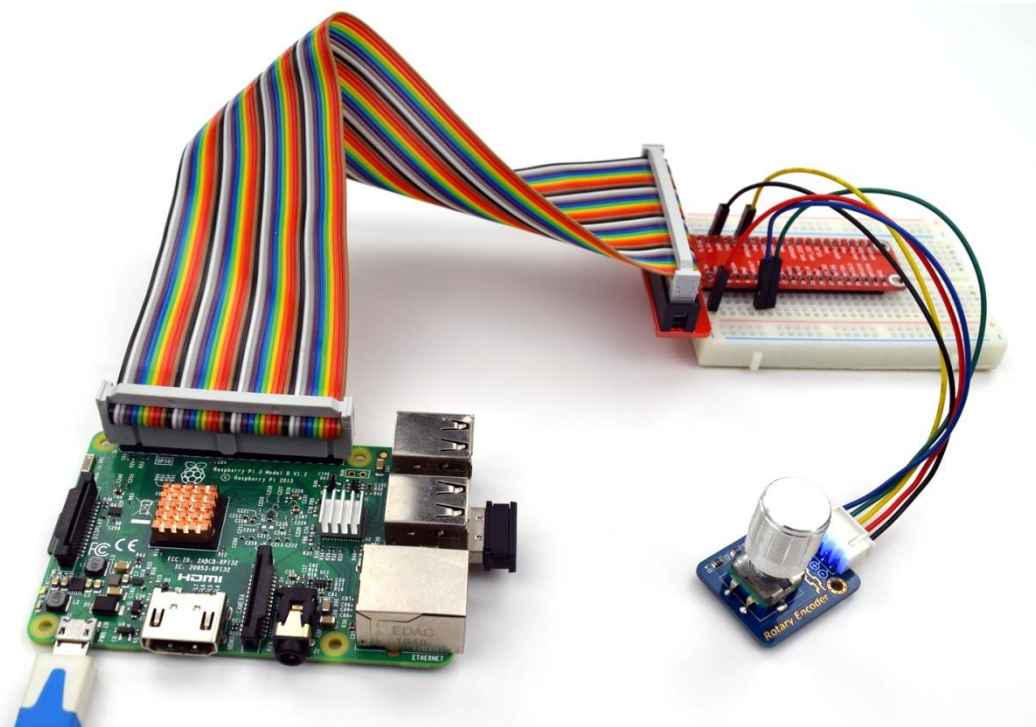
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/06_rotaryEncoder.py)

Step 3: Run

```
$ sudo python 06_rotaryEncoder.py
```

Now rotate the shaft of the rotary encoder, and the value printed on the screen will change. Rotate the rotary encoder clockwise, the value will increase; Rotate it counterclockwise, the value will decrease; Press the rotary encoder, the value will be reset to 0.



Lesson 7 Controlling an LED by Touch Button

Introduction

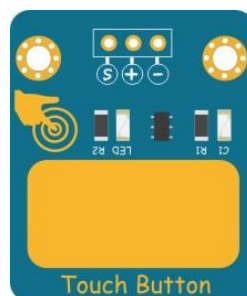
The Touch Button Module is a touch switch module developed based on the principle of capacitive sensing. Touch of human or metal onto the gilded touch surface can be sensed. Besides, it can also detect other such touch with certain materials like plastic and glass between. The sensitivity in these cases depends on the touched area and thickness of the material between the touch pad and human or metal. The module can be conveniently used to replace physical buttons.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Touch Button Module
- 1 * LED Module
- 2 * 3-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:

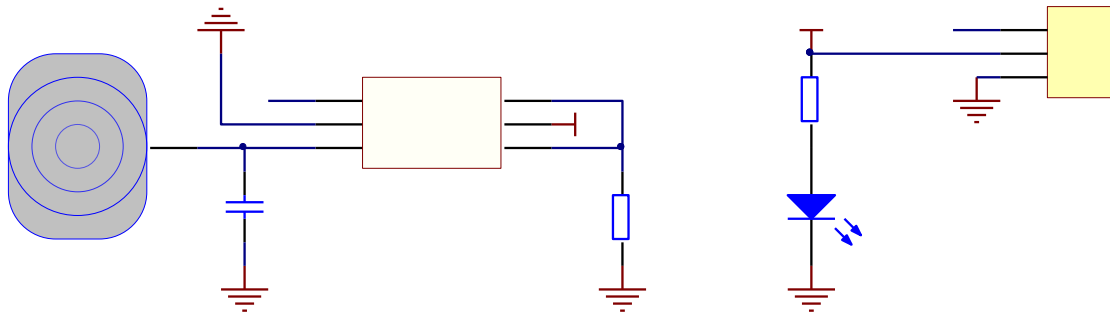


Pin definition:

S	Output
---	--------

+	VCC
-	GND

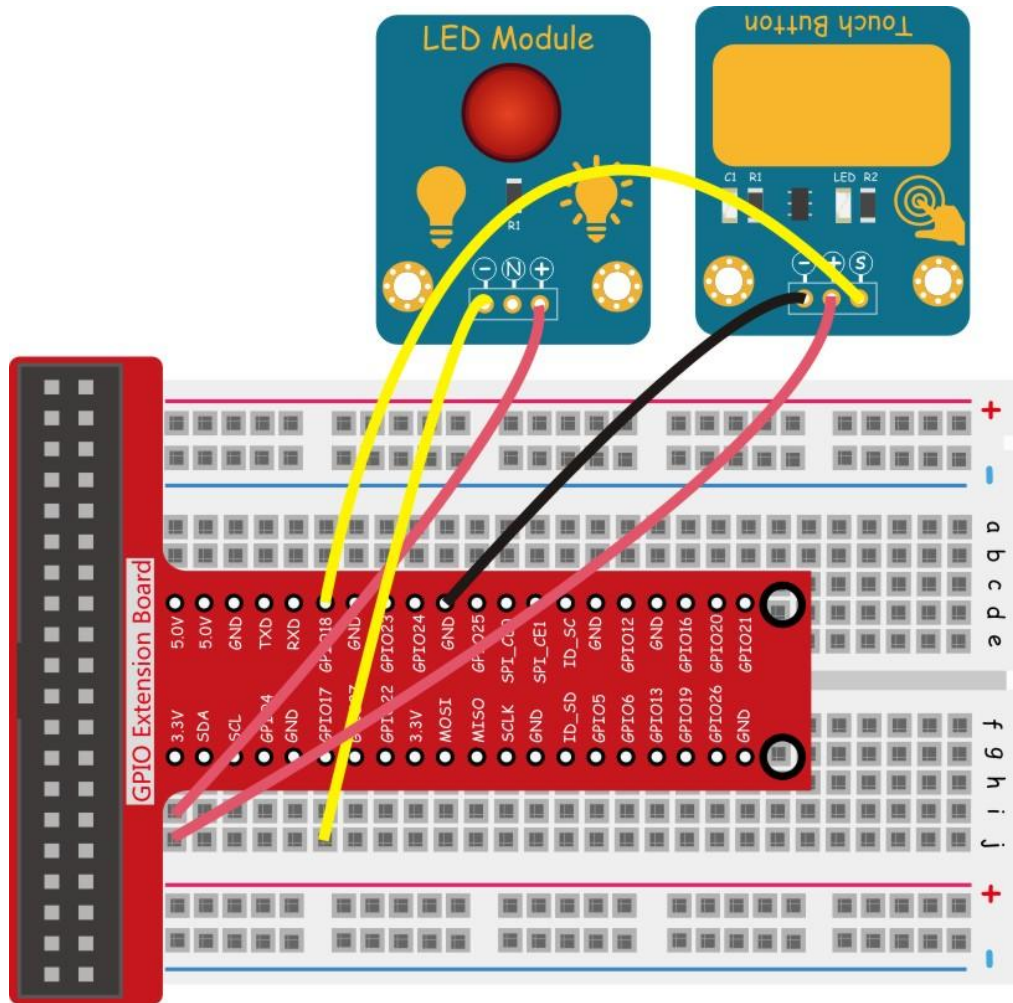
The schematic diagram:



In this experiment, by programming the Raspberry Pi, we detect the High or Low of the output terminal of the Touch Button Module by pin 12 of the Raspberry Pi, so as to tell whether fingers touched the touch button or not.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/07_touchBtn/touchBtn.c)

Step 3: Compile

```
$ sudo gcc touchBtn.c -o touchBtn -lwiringPi
```

Step 4: Run

```
$ sudo ./touchBtn
```

For Python users:

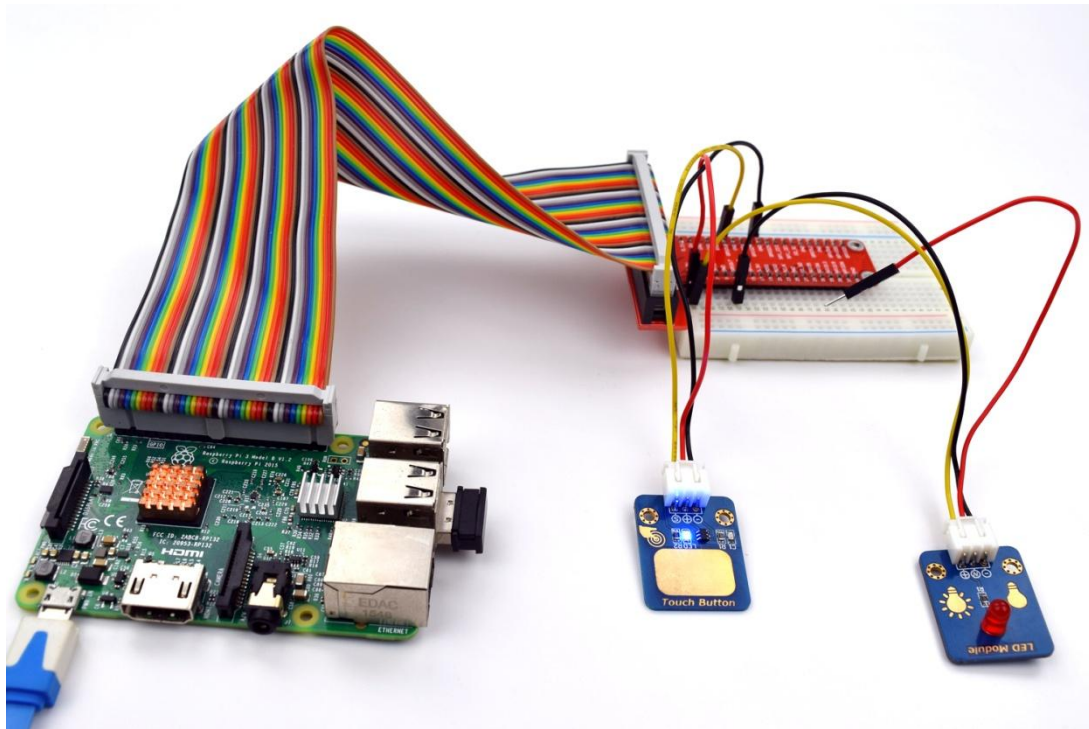
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/07_touchBtn.py)

Step 3: Run

```
$ sudo python 07_touchBtn.py
```

Touch the Touch Button Module, and you can see the LED toggle between on and off.



Lesson 8 Measuring the Temperature via DS18B20

Introduction

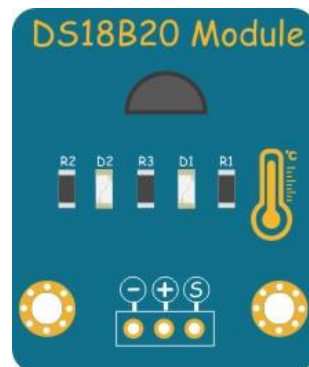
DS18B20 is a single-bus digital temperature sensor of high-precision. The measurement range is -55°C - $+125^{\circ}\text{C}$ and inherent temperature resolution is 0.5°C . The sensor support multi-point network and multi-point temperature measurement – the measured result is sent to the controller via serial port in the format of a 9-12-bit number. This digital sensor can be applied to various microcontrollers. It's simpler on Raspberry Pi.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * DS18B20 Module
- 1 * 3-Pin Wires

Experimental Principle

The Fritzing image:



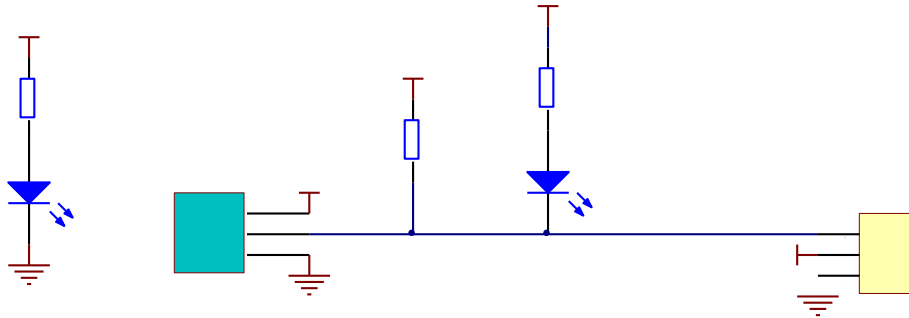
The Physical picture:



Pin definition:

S	Data
+	VCC
-	GND

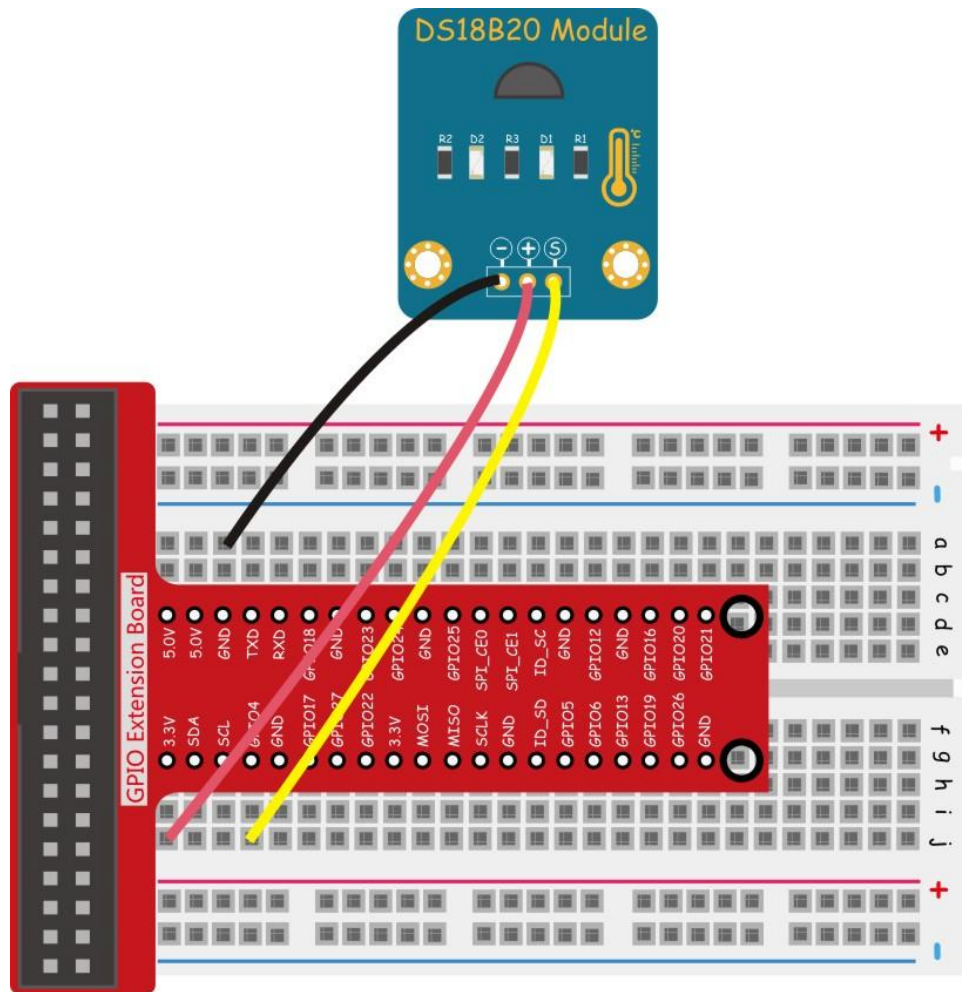
The schematic diagram:



In this experiment, by programming the Raspberry Pi, we read the temperature value collected by the DS18B20 module through pin 7 of the Raspberry Pi, and display it on the terminal.

Experimental Procedures

Step 1: Build the circuit



Step 2: Upgrade Raspberry Pi OS kernel

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

Step 3: Modify the configuration files

```
$ sudo vim /boot/config.txt
```

Then scroll to the bottom and type:

```
dtoverlay=w1-gpio
```

Then reboot Raspberry Pi

```
$ sudo reboot
```

Mount the device drivers and confirm whether the device is effective or not

```
$ sudo modprobe w1-gpio
```

```
$ sudo modprobe w1-therm
```

```
$ cd /sys/bus/w1/devices/
```

```
$ ls
```

The result is as follows:

```
root@rasberry_pi:/sys/bus/w1/devices# ls
```

```
28-00000355d573 w1_bus_master1
```

28-00000355d573 is an external temperature sensor device, but it may vary with every client. It is the serial number of your DS18B20.

Step 4: Check the current temperature

```
$ cd 28-00000355d573
```

```
$ ls
```

The result is as follows:

```
root@rasberry_pi:/sys/bus/w1/devices/28-00000355d573# ls
```

```
driver id name power subsystem uevent w1_slave
```

```
$ cat w1_slave
```

The result is as follows:

```
root@rasberry_pi:/sys/bus/w1_slave/28-00000495db35# cat w1_slave
```

```
a3 01 4b 46 7f ff 0d 10 ce : crc=ce YES
```

```
a3 01 4b 46 7f ff 0d 10 ce t=28154
```

The second line `t=28154` is current temperature value. If you want to convert it to degree Celsius, you can divide by 1000, that is, the current temperature is $28154/1000=28.154$ °C.

For C language users:

Step 5: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/08_ds18b20/ds18b20_2.c)

Step 6: Compile

```
$ sudo gcc ds18b20_2.c -o ds18b20 -lwiringPi
```

Step 7: Run

```
$ sudo ./ds18b20
```

For Python users:

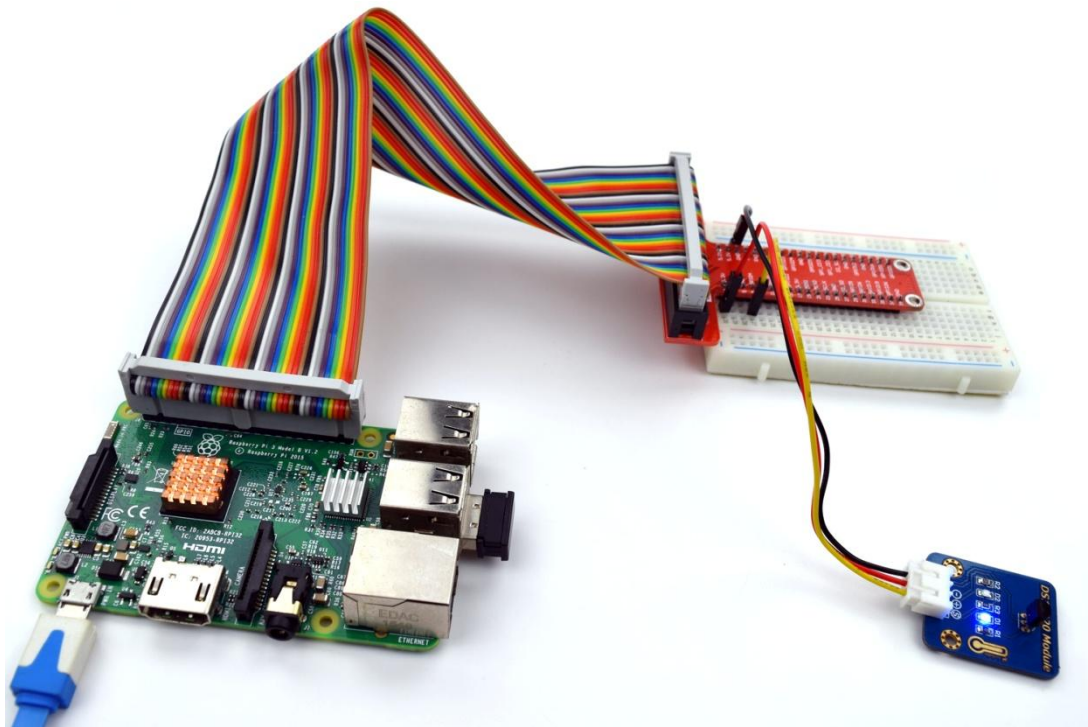
Step 5: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/08_ds18b20.py)

Step 6: Run

```
$ sudo python 08_ds18b20.py
```

Now, you can see the current temperature is printed on the terminal.



Lesson 9 Measuring the Distance

Introduction

Ultrasonic Distance Sensor module supports a contactless detection within a distance of 2cm-400cm. It contains an ultrasonic emitter, receiver and control circuits.

Notes:

1. The module is not suggested to connect wires when power is on. If you have to do so, please first connect the GND and then other pins; otherwise, the module may not work.
2. During the ranging, the area of the targeted object should be no less than 0.5cm and the surface facing the module should be as flat as possible; otherwise the result may be inaccurate.

Components

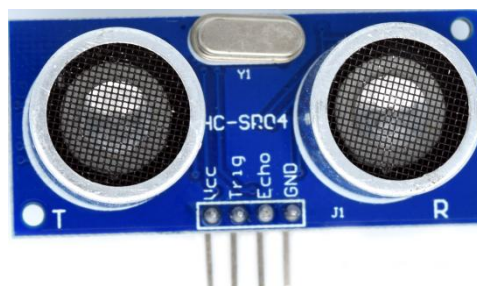
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Ultrasonic Sensor Module
- 4 * Jumper Wires

Experimental Principle

The Fritzing image:



The Physical picture:



Pin definition:

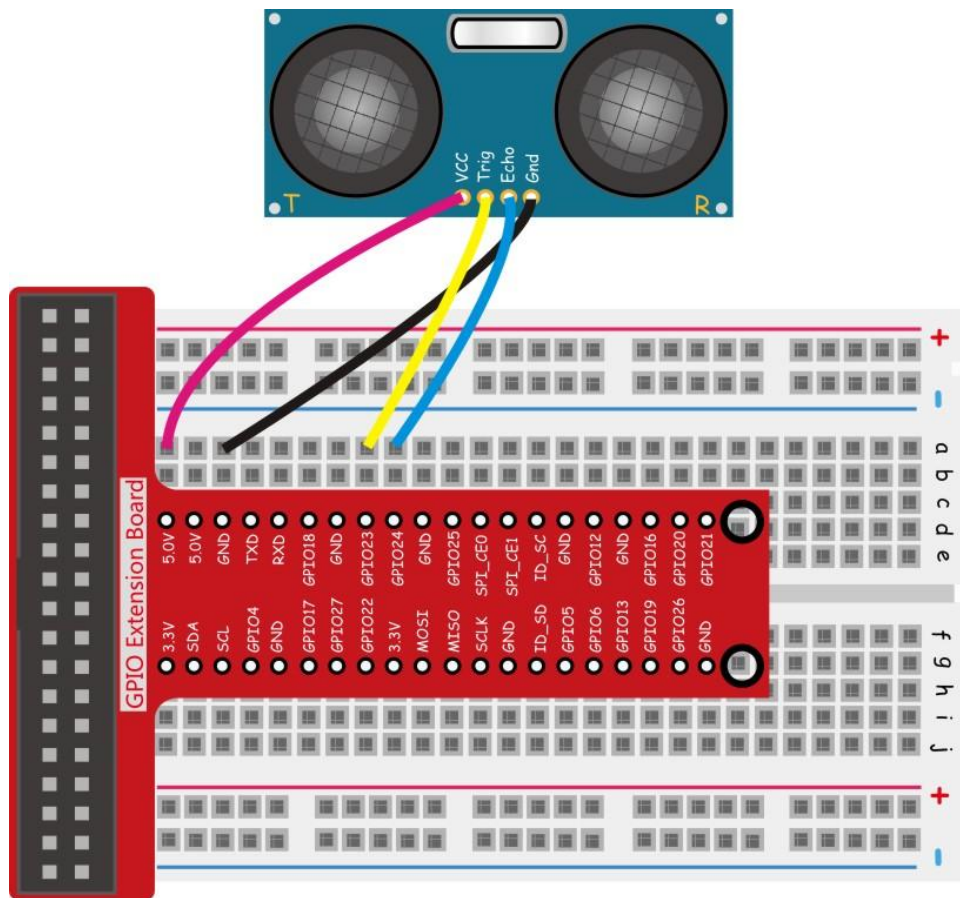
Trig	Input
Echo	Output

Vcc	VCC
GND	GND

This experiment uses the Ultrasonic Distance Sensor module to detect the distance between the obstacle and module and show the data sensed on the terminal.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/09_ultrasonicSensor/distance.c)

Step 3: Compile

```
$ sudo gcc distance.c -o distance -lwiringPi
```

Step 4: Run

```
$ sudo ./distance
```

For Python users:

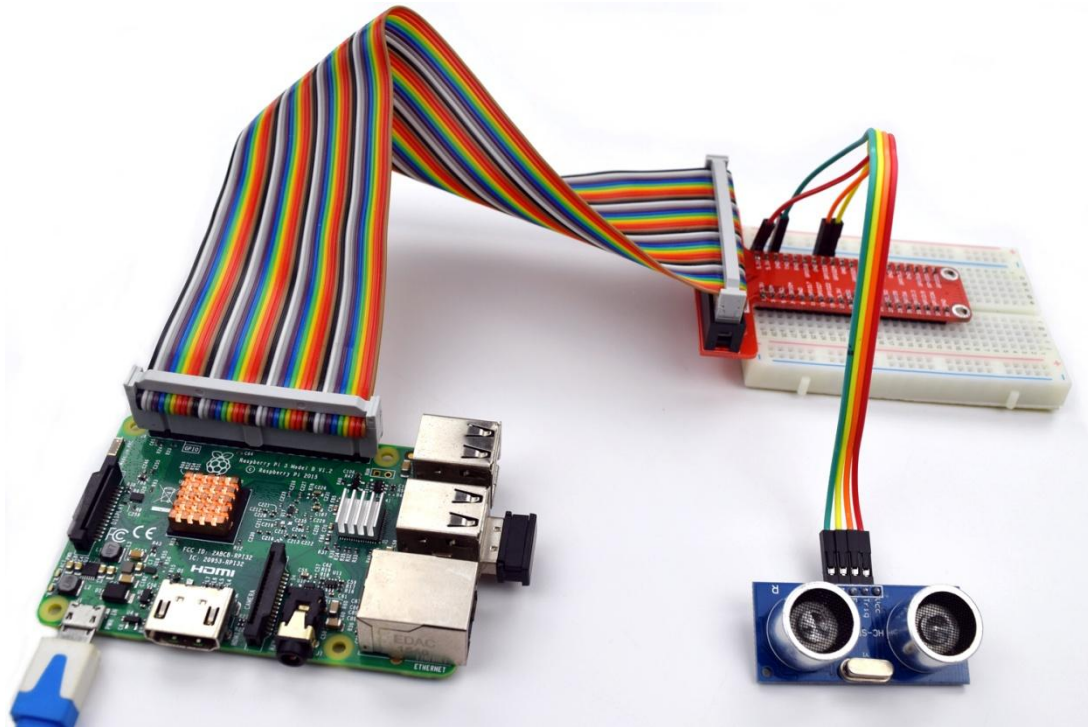
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/09_distance.py)

Step 3: Run

```
$ sudo python 09_distance.py
```

Now, you will see the distance to the obstacle at front of the Ultrasonic Distance Sensor module displayed on the terminal.



Lesson 10 LED Bar Graph

Introduction

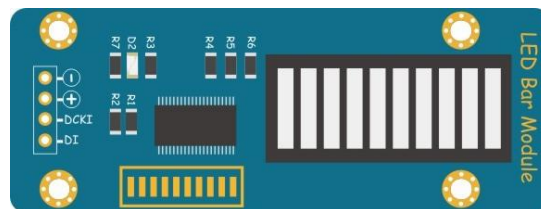
The LED bar is an analog indicating component usually used for volume indication.

Components

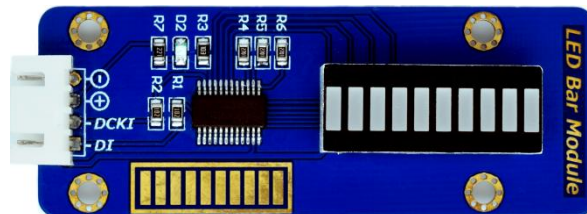
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * LED Bar Module
- 1 * 4-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



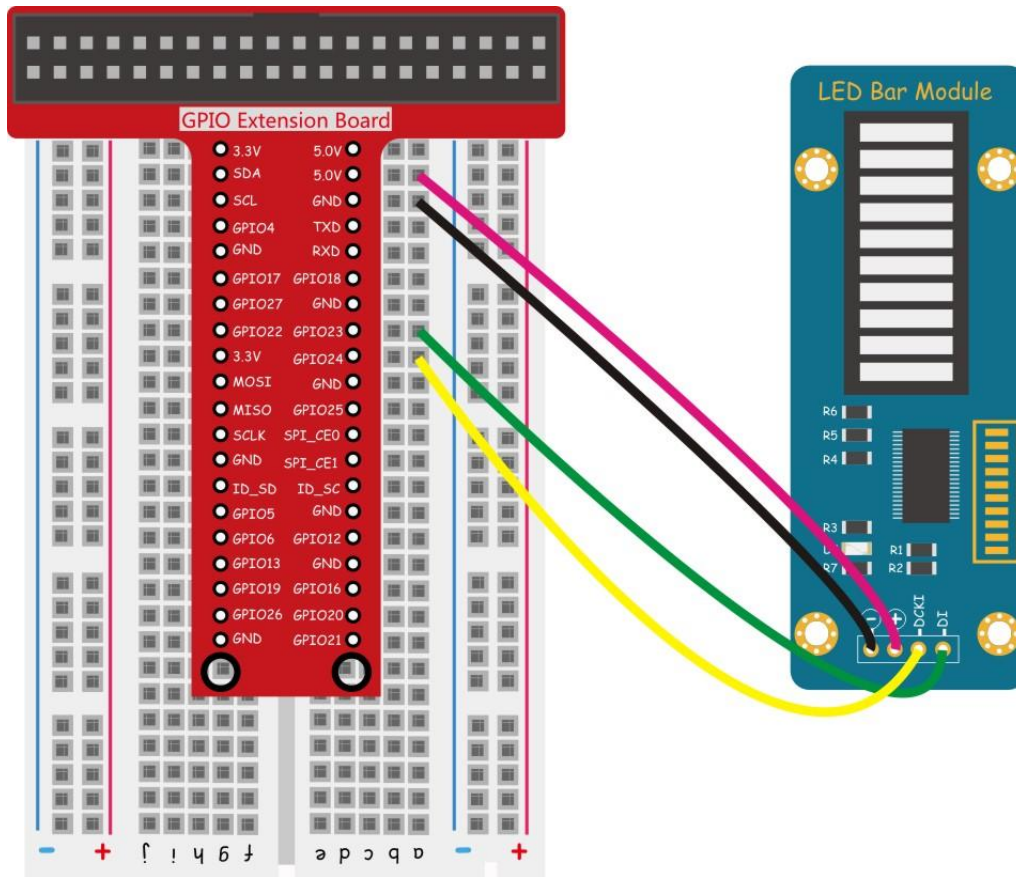
Pin definition:

DI	Data In
DCLK	Clock
+	VCC
-	GND

The experiment is to control the number of LEDs brightened on the LED bar graph by programming the Raspberry Pi.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/10_ledBar/ledBar.c)

Step 3: Compile

```
$ sudo gcc ledBar.c -o ledBar -lwiringPi -lm
```

Step 4: Run

```
$ sudo ./ledBar
```

For Python users:

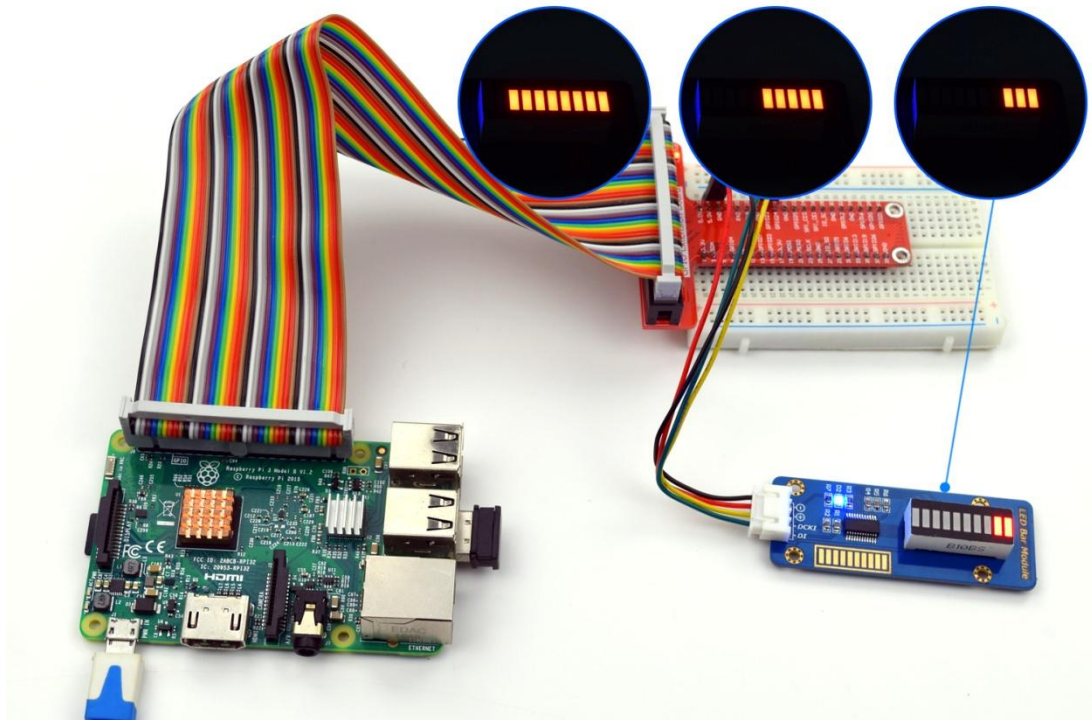
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/10_ledBar.py)

Step 3: Run

```
$ sudo python 10_ledBar.py
```

Now you can see the LEDs on the LED Bar Graph module light up and dim one by one repeatedly.



Lesson 11 How to Drive the Segment Display

Introduction

The module consists of a 4-digit 7-segment common-cathode (CC) diode and a chip TM1638. It communicates with the Raspberry Pi via three wires and can show numbers and simple characters.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Segment Display Module
- 1 * 5-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



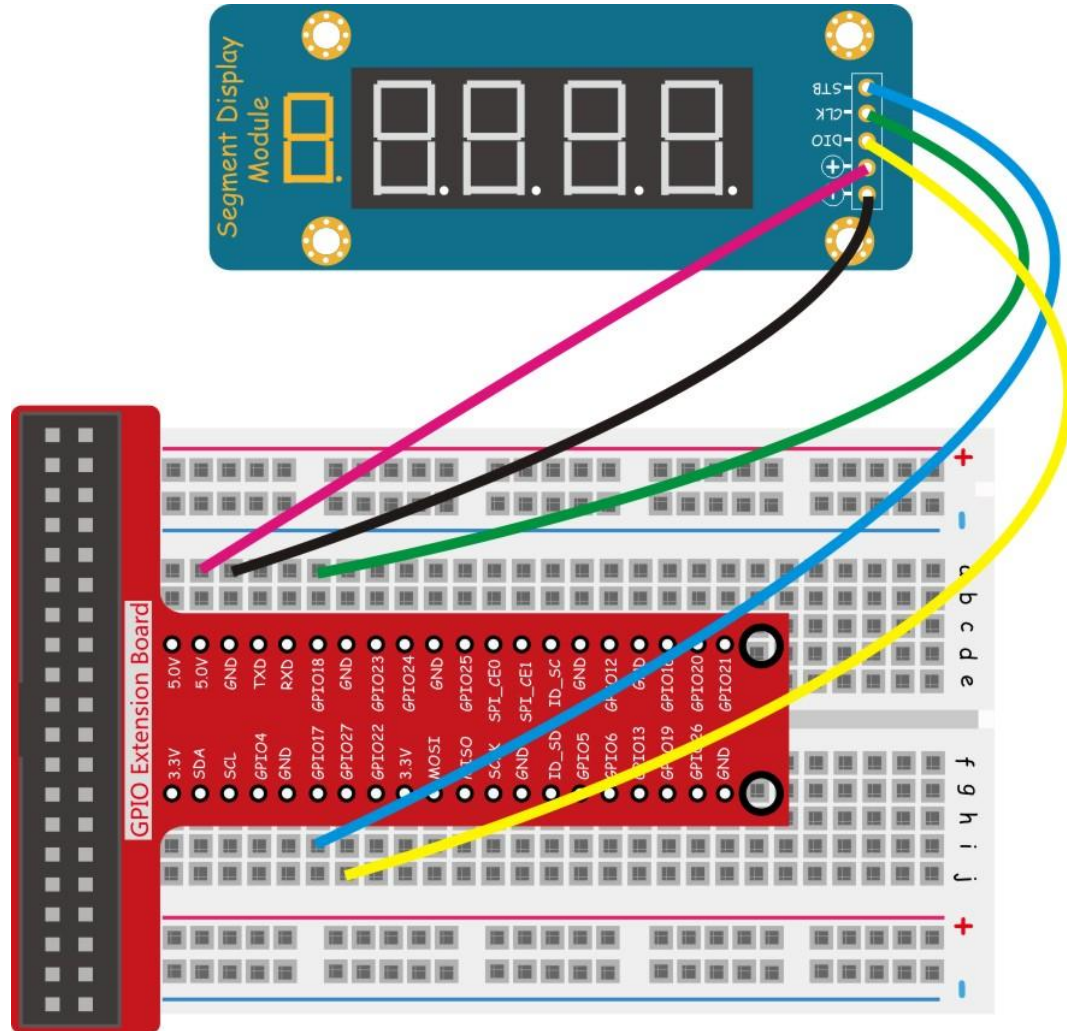
Pin definition:

STB	Chip Select
CLK	Clock
DIO	Data
+	VCC
-	GND

Through programming the Raspberry Pi, make the module display 0000~9999.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/11_segmentDisplay/segment.c)

Step 3: Compile

```
$ sudo gcc segment.c -o segment -lwiringPi
```

Step 4: Run

```
$ sudo ./segment
```

For Python users:

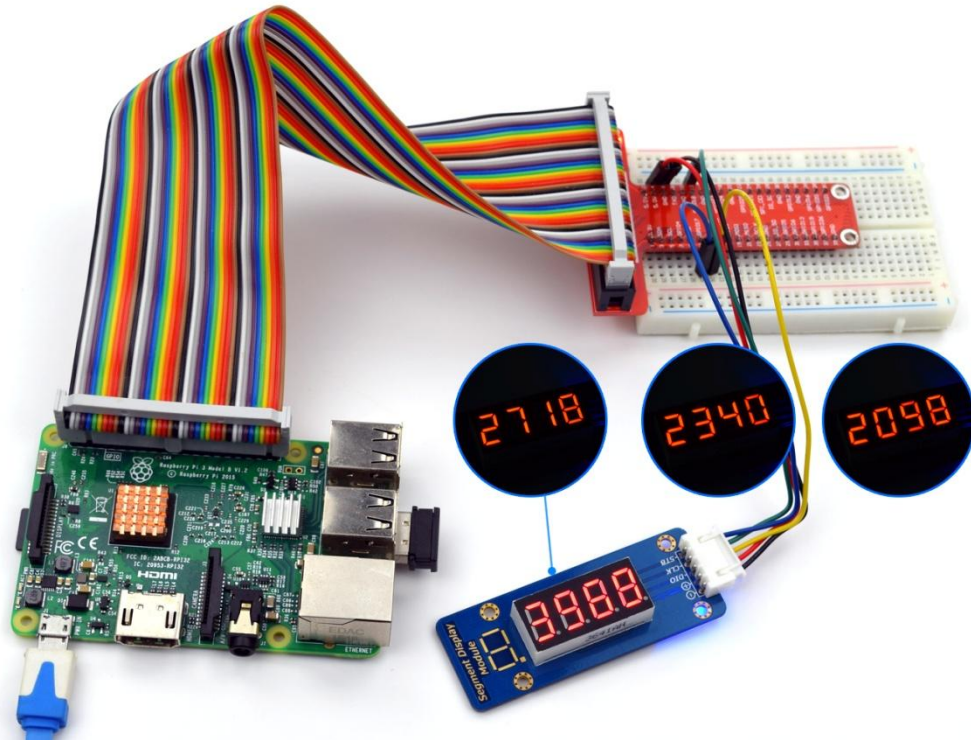
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/11_segment.py)

Step 3: Run

```
$ sudo python 11_segment.py
```

Now you can see the number 0~9999 shown repeatedly on the digital display.



Lesson 12 Potentiometer

Introduction

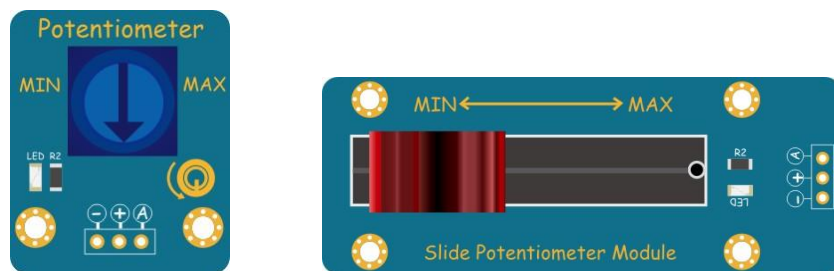
Potentiometer is a resistor whose resistance can be adjusted continuously. When its shaft is turned, the moving contact (or wiper) slides along the resistor.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * ADC0832 Module
- 1 * Potentiometer Module
- 1 * Slide Potentiometer Module
- 2 * 3-Pin Wires
- 1 * 5-Pin Wires

Experimental Principle

The Fritzing image:



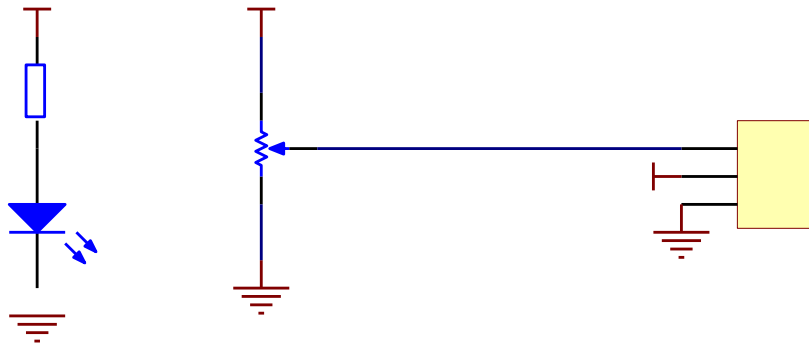
The Physical picture:



Pin definition:

A	Analog Output
+	VCC
-	GND

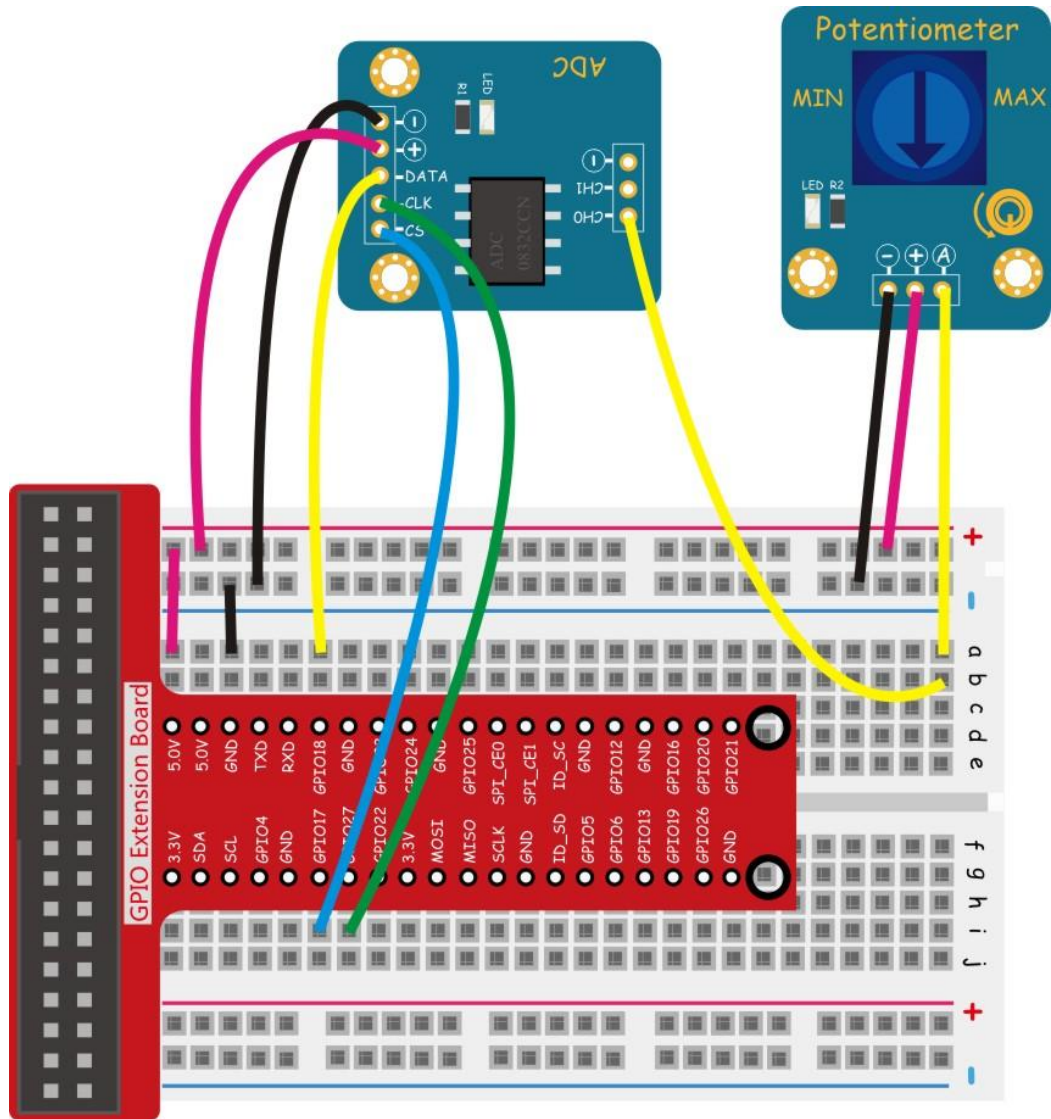
The schematic diagram:

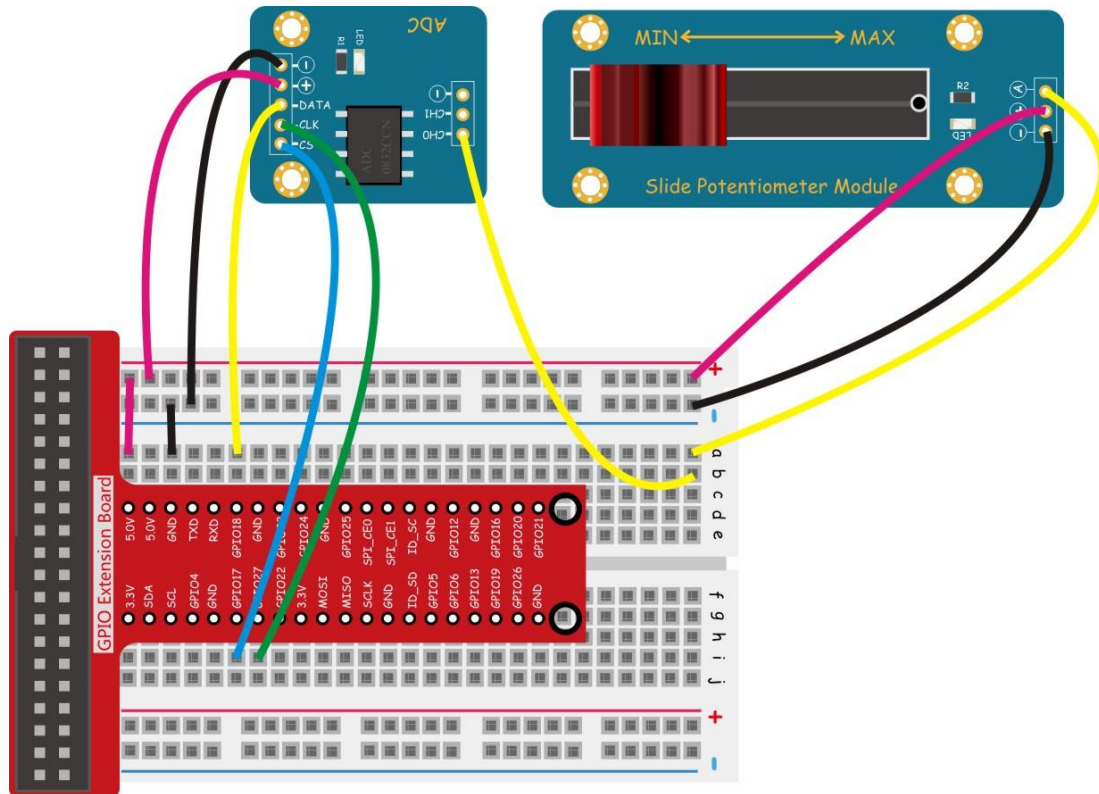


In this experiment, by programming the Raspberry Pi, we collect the analog values output by the Potentiometer module through pin CH0 of the ADC0832, convert it to digital values and display them on the terminal.

Experimental Procedures

Step 1: Build the circuit





For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/12_potentiometer/potentiometer.c)

Step 3: Compile

```
$ sudo gcc potentiometer.c -o potentiometer -lwiringPi
```

Step 4: Run

```
$ sudo ./potentiometer
```

For Python users:

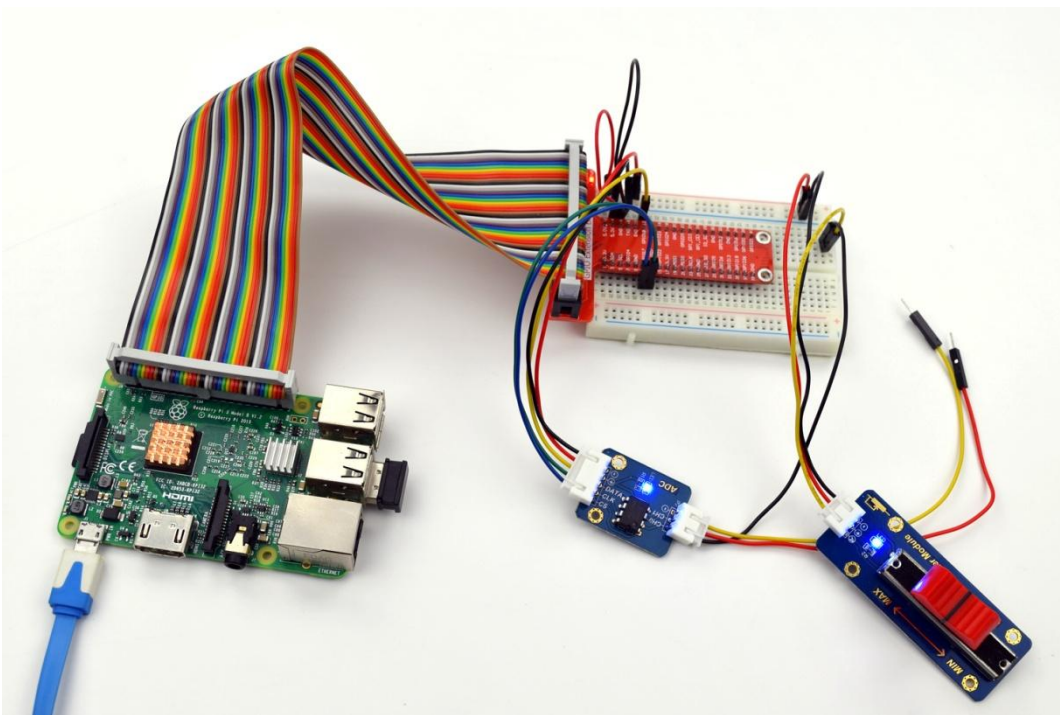
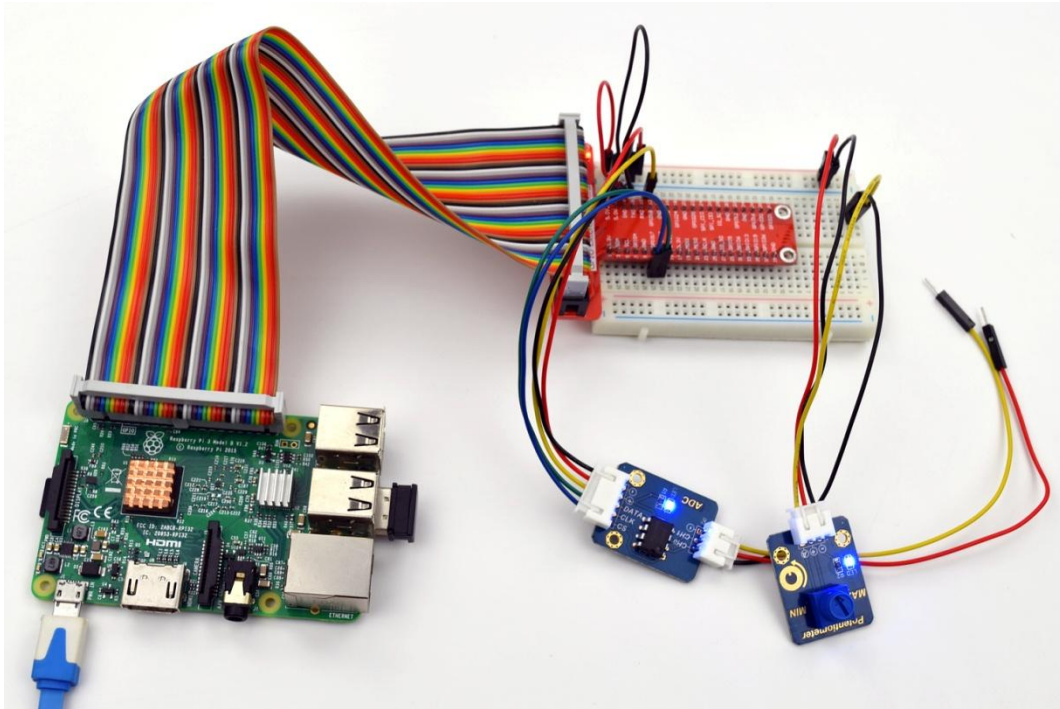
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/12_potentiometer.py)

Step 3: Run

```
$ sudo python 12_potentiometer.py
```

Turn the knob on the Potentiometer module, you will find that the value displayed on the terminal is changed.



Lesson 13 Photoresistor

Introduction

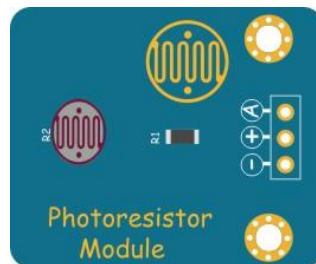
The photoresistor module is a resistor module designed based on the principle of photoconductive effect of semiconductors, of which the resistance varies with the intensity of incident light. The resistance of the photoresistor we use decreases with stronger incident light and increases with weaker light. In experiments and daily light, photoresistors are usually used for detecting light intensity.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * ADC0832 Module
- 1 * Photoresistor Module
- 2 * 3-Pin Wires
- 1 * 5-Pin Wires

Experimental Principle

The Fritzing image:



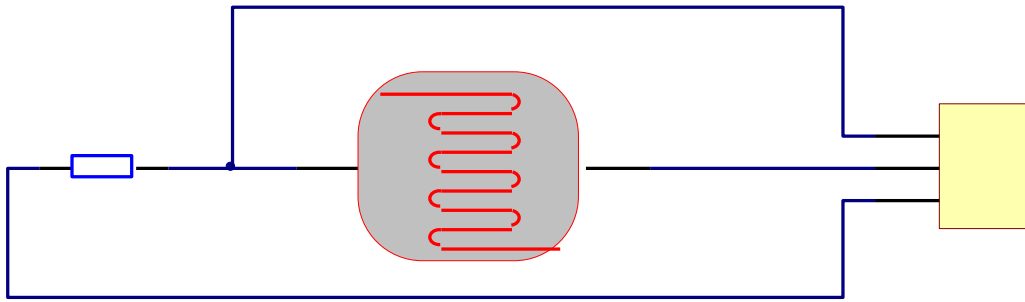
The Physical picture:



Pin definition:

A	Analog Output
+	VCC
-	GND

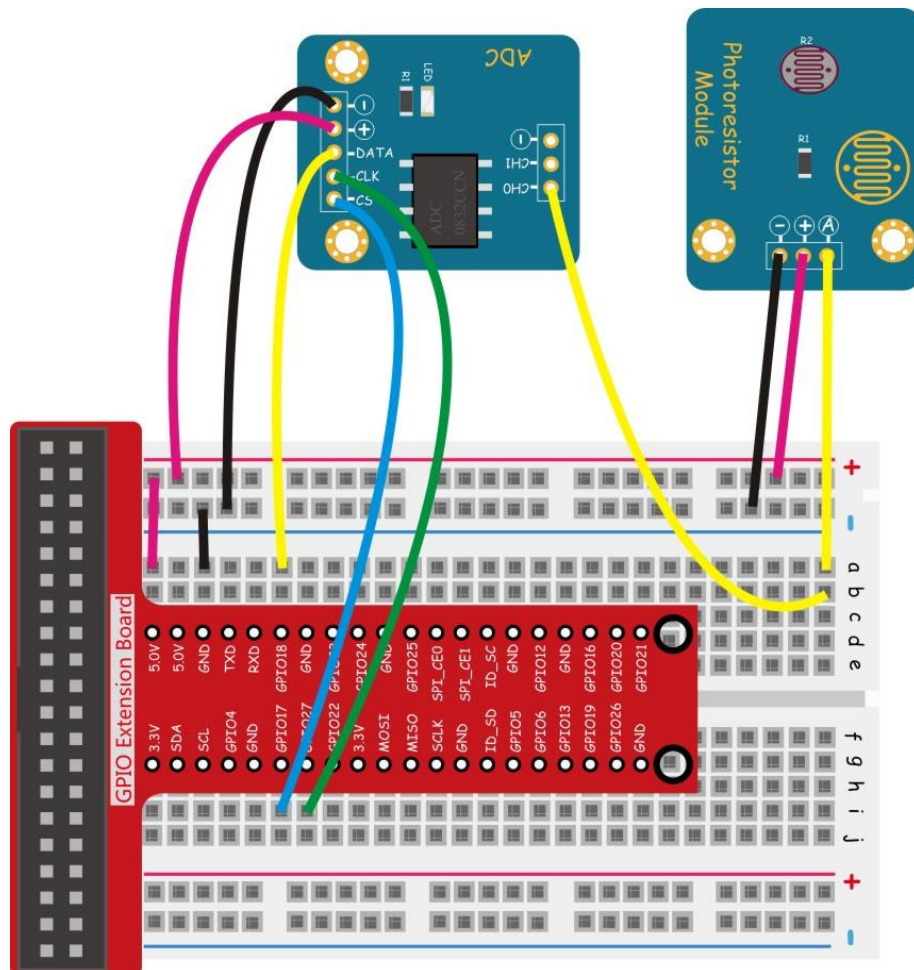
The schematic diagram:



This experiment is to collect the data of light intensity by the Photoresistor module and then display it on the terminal.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/13_photoresistor/photoresistor.c)

Step 3: Compile

```
$ sudo gcc photoresistor.c -o photoresistor -lwiringPi
```

Step 4: Run

```
$ sudo ./photoresistor
```

For Python users:

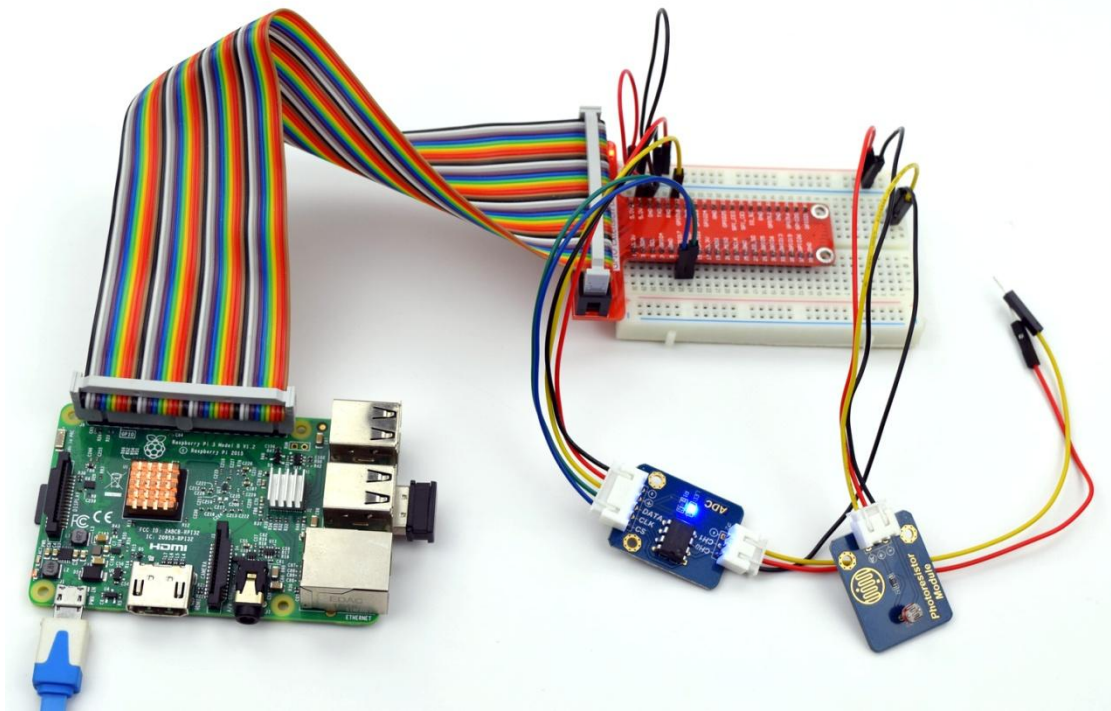
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeptt_Compact_Sensor_Kit_for_RPi/Python/13_photoresistor.py)

Step 3: Run

```
$ sudo python 13_photoresistor.py
```

Now, when you try to cover to the photoresistor, you will find that the value displayed on the screen decreasing. On the contrary, when you shine the photoresistor with strong light, the value displayed will increase.



Lesson 14 Thermistor

Introduction

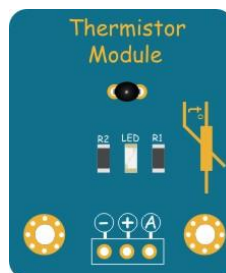
Thermistors can be divided into two types by temperature coefficient: positive temperature coefficient (PTC) and negative temperature coefficient (NTC). The typical feature of a thermistor is sensitive to temperature – its resistance varies with ambient temperature changes. For PTC thermistor, when the temperature gets high, its resistance increases; for NTC thermistor, the case is the opposite. The thermistor we use is an NTC one.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * ADC0832 Module
- 1 * Thermistor Module
- 2 * 3-Pin Wires
- 1 * 5-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:

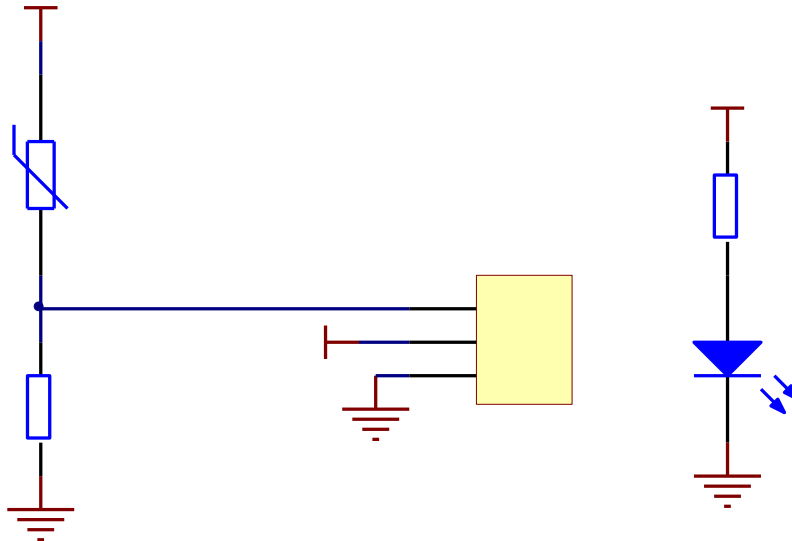


Pin definition:

A	Analog Output
---	---------------

+	VCC
-	GND

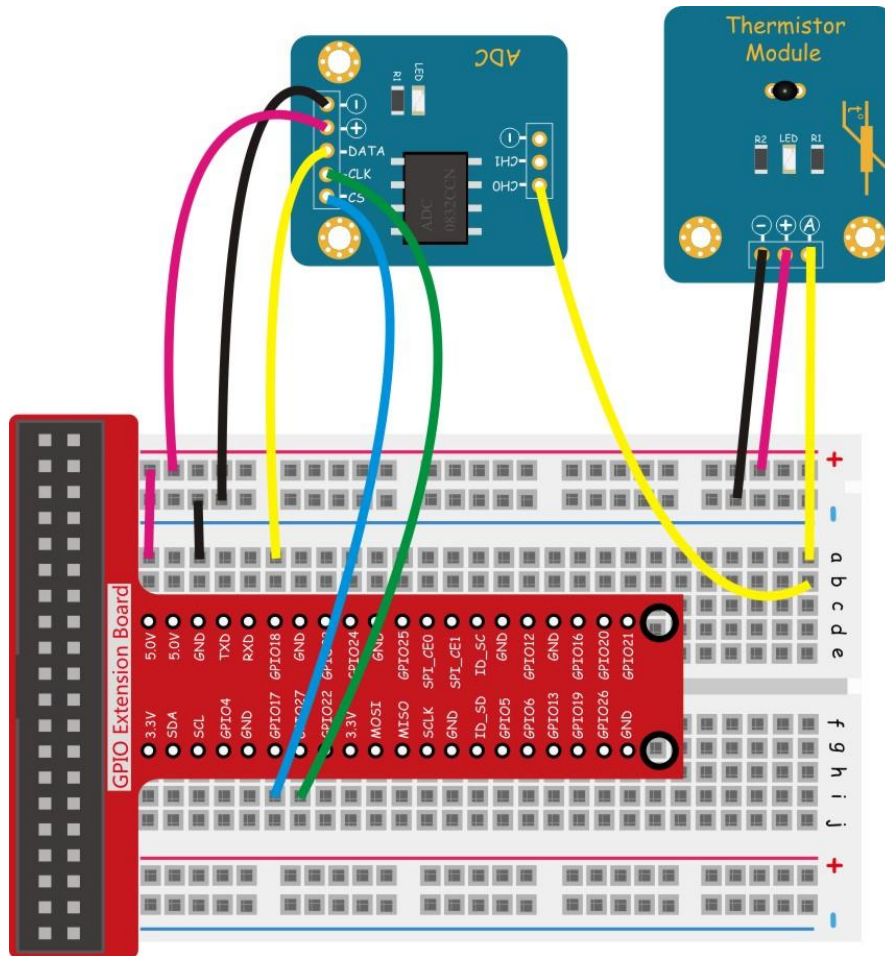
The schematic diagram:



In this experiment, by programming the Raspberry Pi, we collect the analog values output by the Thermistor module through CH0 of the ADC0832, change it to digital values and display them on terminal.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/14_thermistor/thermistor.c)

Step 3: Compile

```
$ sudo gcc thermistor.c -o thermistor -lwiringPi
```

Step 4: Run

```
$ sudo ./thermistor
```

For Python users:

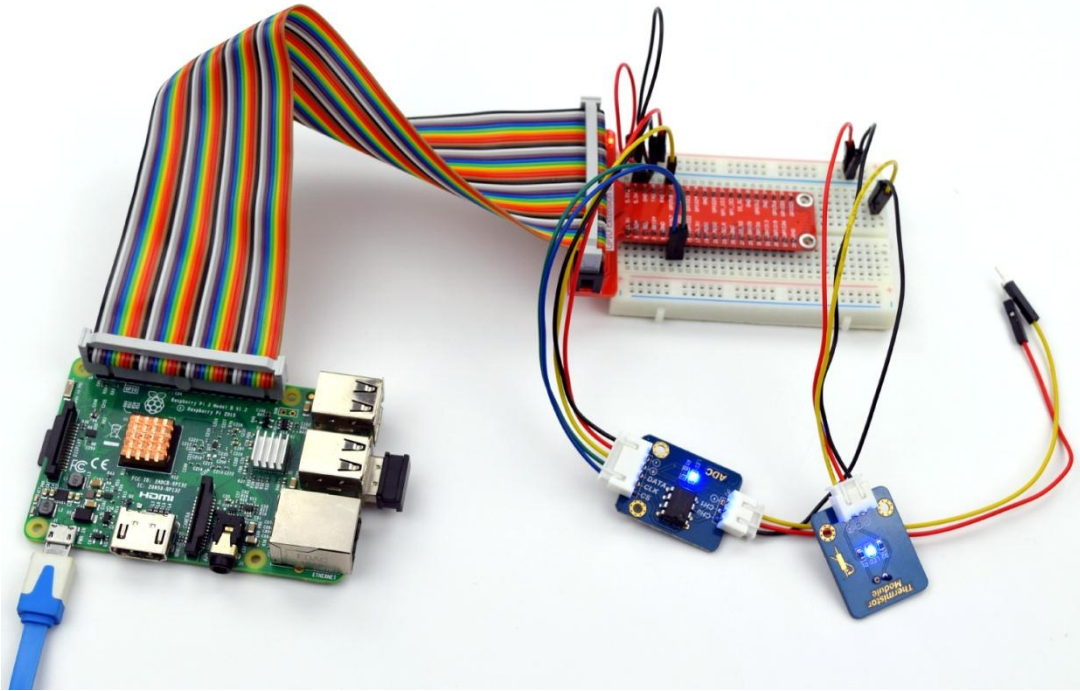
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/14_thermistor.py)

Step 3: Run

```
$ sudo python 14_thermistor.py
```

Now, touch the thermistor, and you can see the value displayed on the screen, which changes accordingly.



Lesson 15 Soil Moisture Detection

Introduction

The Soil Moisture Sensor module is a simple sensor that measures the soil moisture. When the soil moisture is insufficient, the output value of the sensor will decrease; on the other hand, the value will increase when there's enough water. The surface of the sensor is gilded to prolong its life.

The CM Module consists of a comparator LM393 and extremely simple external circuits. When using the module, you can set a threshold via the blue potentiometer beforehand. When the input analog value reaches the threshold, the digital pin S will output a Low level.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Soil Moisture Sensor Module
- 1 * LM393 CM Module
- 1 * ADC0832 Module
- 1 * 2-Pin Wires
- 1 * 3-Pin Wires
- 1 * 4-Pin Wires
- 1 * 5-Pin Wires

Experimental Principle

The Fritzing images:



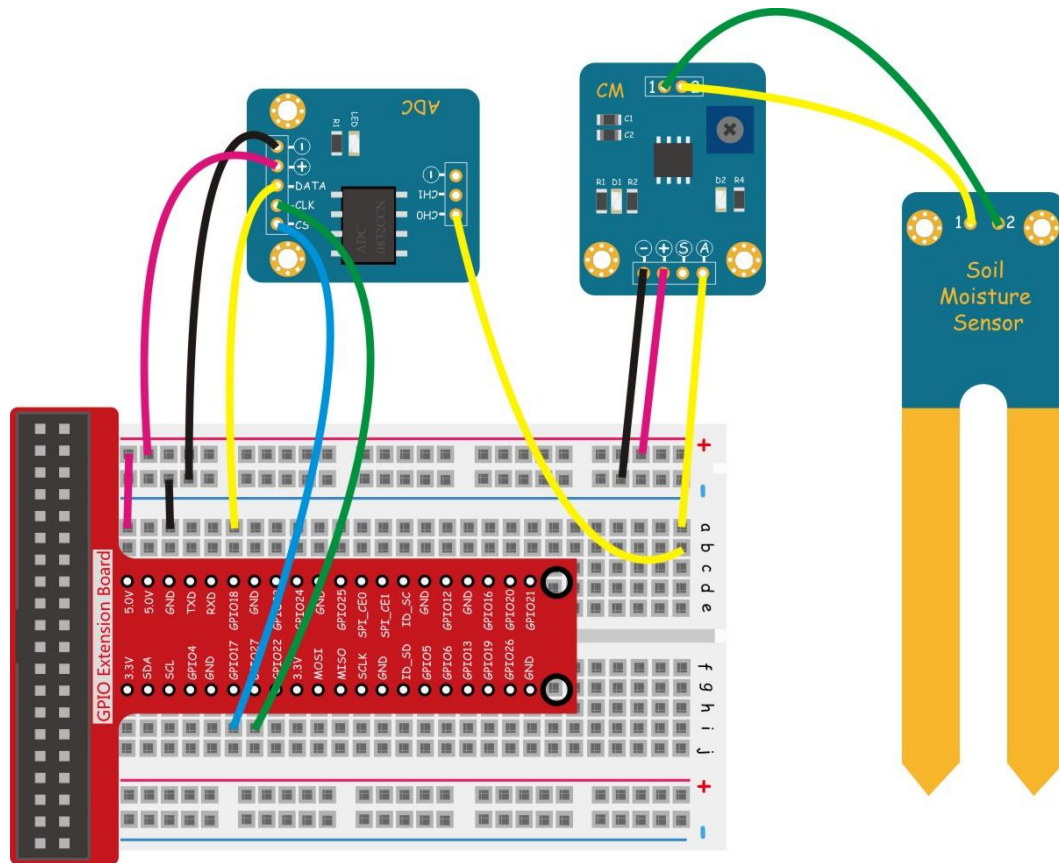
The Physical picture:



The experiment uses the Soil Moisture Sensor module to collect data of soil moisture and display it on terminal.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/15_soilMoisture/soilMoisture.c)

Step 3: Compile

```
$ sudo gcc soilMoisture.c -o soilMoisture -lwiringPi
```

Step 4: Run

```
$ sudo ./soilMoisture
```

For Python users:

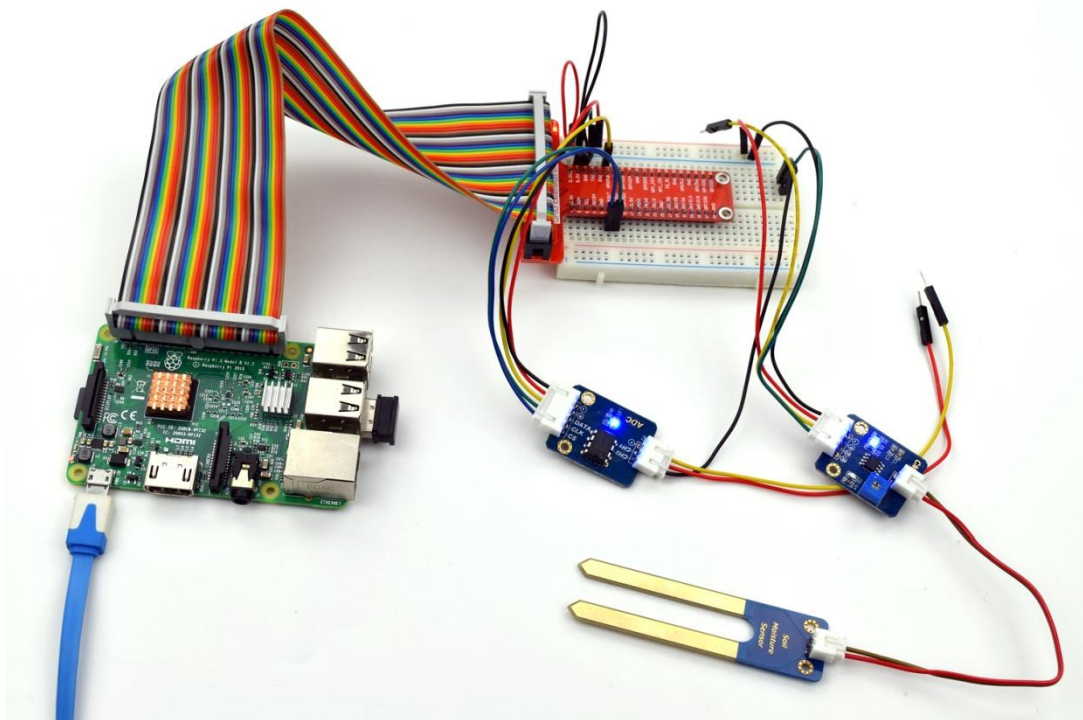
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/15_soilMoisture.py)

Step 3: Run

```
$ sudo python 15_soilMoisture.py
```

Plug the sensor into the soil, you will see the value of soil moisture collected by the module displayed on the terminal.



Lesson 16 MQ-2 Gas Sensor

Introduction

MQ-2 is a sensor that can detect flammable gases such as methane, hydrogen, and propane and so on. It adopts the low conductivity stannic oxide for the basic material. When there are flammable gases in the ambient environment, the conductivity of the sensor will increase as the gases become denser. This type can detect a wide range of gases thus making it a low cost multifunctional sensor. The sensor can be used for methane leak alarm and automatic smoke exhaust fan. With the features, it boasts a perfect sensor for indoor air regulation that meets the environmental standards.

Notes:

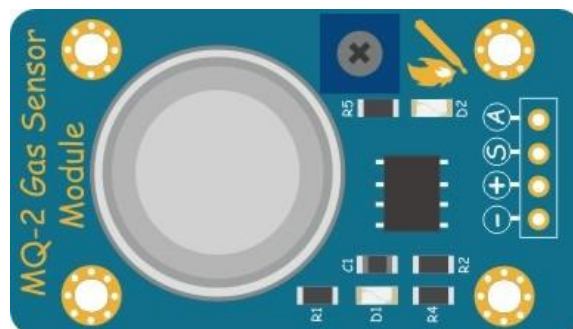
1. This sensor is equipped with an adjustment potentiometer for the alarm threshold. Spin the knob of the pot clockwise, and the threshold will be increased; spin it counterclockwise, the threshold will be reduced.
2. The sensor may not output a steady and accurate data immediately; it needs to be preheated for about 1 minute to collect data steadily.

Components

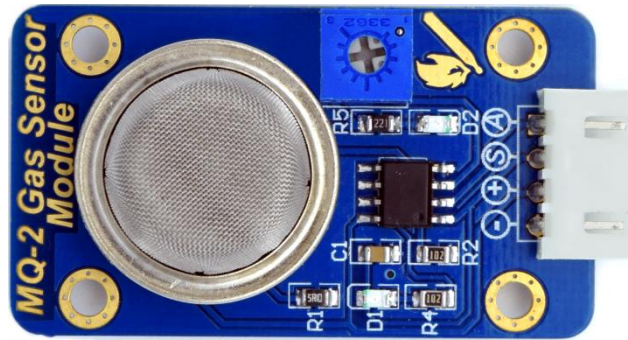
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * ADC0832 Module
- 1 * MQ-2 Gas Sensor Module
- 1 * 3-Pin Wires
- 1 * 4-Pin Wires
- 1 * 5-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



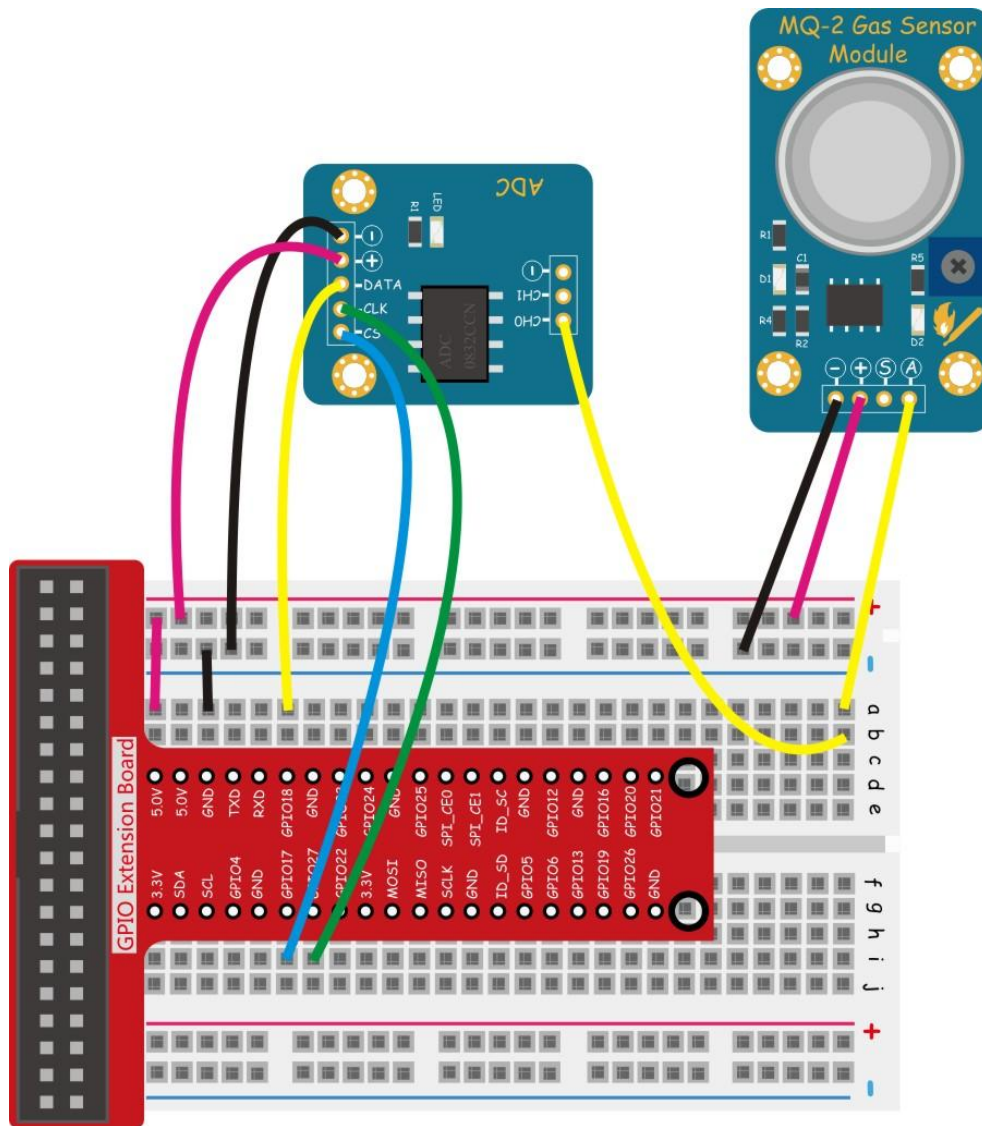
Pin definition:

S	Digital Output
A	Analog Output
+	VCC
-	GND

In this experiment, by programming the Raspberry Pi, we read the analog values collected by the MQ-2 Gas Sensor and display them on the terminal.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/16_MQ-2/mq-2.c)

Step 3: Compile

```
$ sudo gcc mq-2.c -o mq-2 -lwiringPi
```

Step 4: Run

```
$ sudo ./mq-2
```

For Python users:

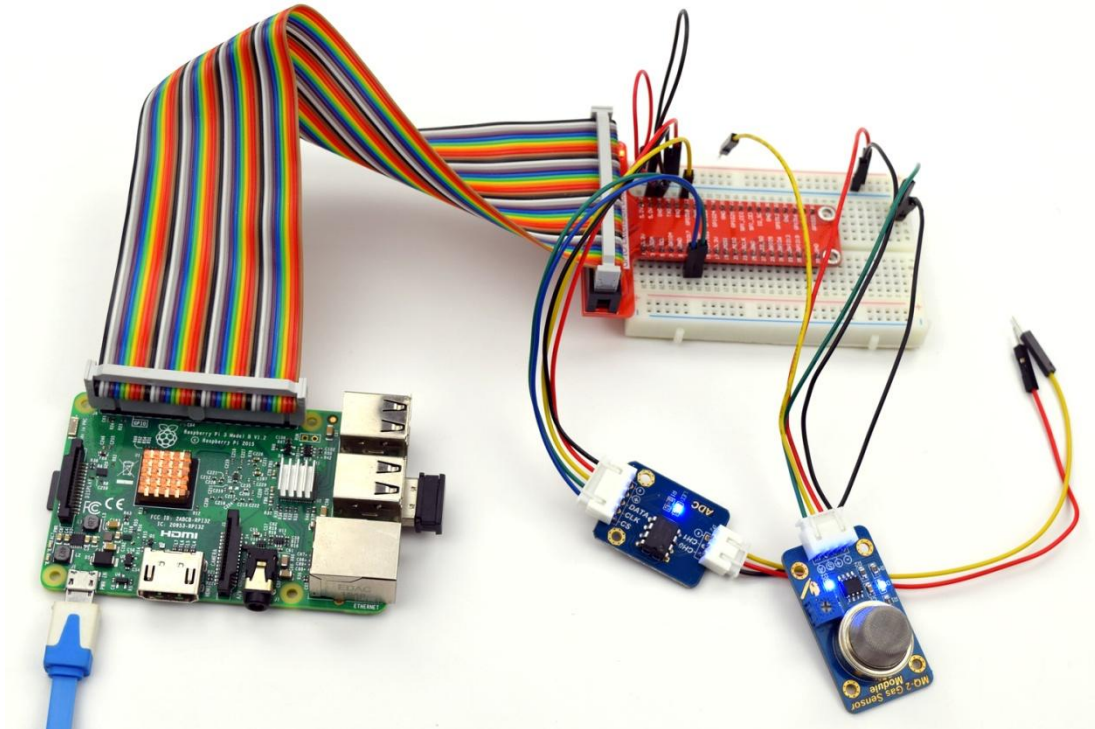
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/16_mq-2.py)

Step 3: Run

```
$ sudo python 16_mq-2.py
```

Release some methane near the module, and you will see the corresponding message on the terminal indicating flammable gases. Also the value output by the analog pin of the module will be printed.



Lesson 17 PS2 Joystick

Introduction

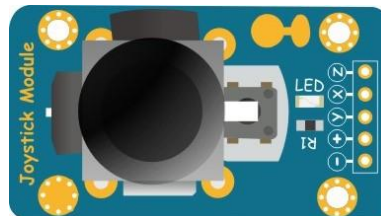
The PS2 Joystick Module is an input device. It consists of a station and the control knob onside. It functions by sending angle or direction signals to the device controlled. The button on the module can also be recognized by the microcontroller. The module supports two-channel analog output, namely, x- and y-axis offset, and one-channel digital output which indicates whether the user has pressed the button at z-axis or not. The Joystick Module can be used to easily control the object to move in a three-dimensional space. For example, it can be applied to control crane, truck, electronic games, robots, etc.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * PS2 Joystick Module
- 1 * ADC0832 Module
- 1 * 3-Pin Wires
- 2 * 5-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



Pin definition:

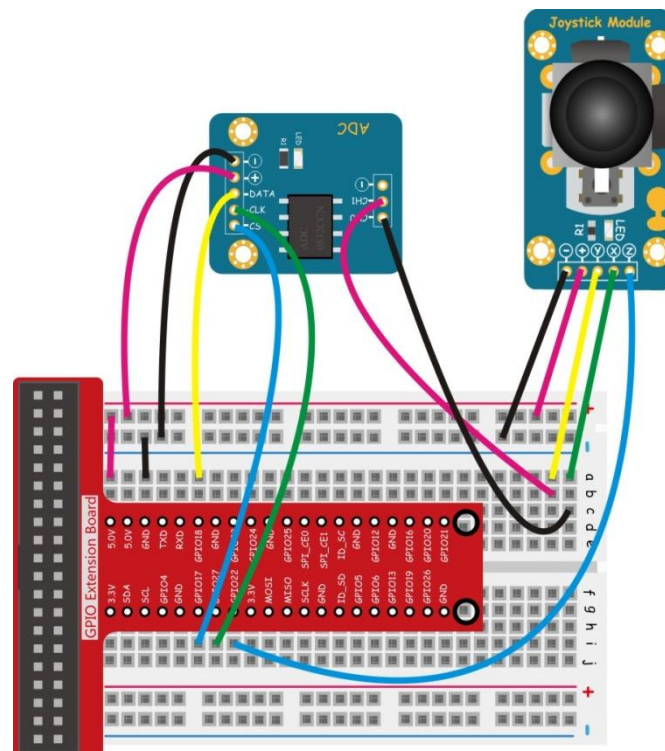
z	Switch Output
x	Analog Output(X)
y	Analog Output(Y)

+	VCC
-	GND

The experiment reads the status of the PS2 Joystick Module, then send the data to and display it on the terminal.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/17_PS2Joystick/joystick.c)

Step 3: Compile

```
$ sudo gcc joystick.c -o joystick -lwiringPi
```

Step 4: Run

```
$ sudo ./joystick
```

For Python users:

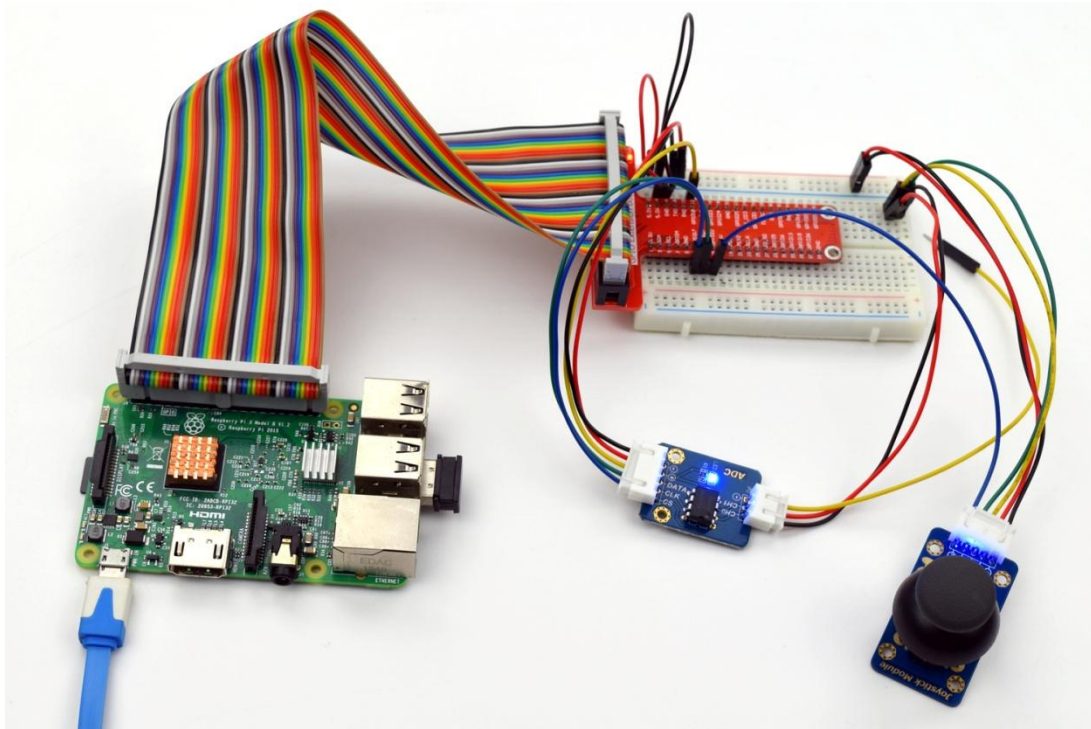
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/17_PS2Joystick.py)

Step 3: Run

```
$ sudo python 17_PS2Joystick.py
```

Press or pull the knob and you will see the value of current status displayed on the terminal.



Lesson 18 LCD1602 Display

Introduction

LCD1602:

1602 crystal, or 1602 character crystal, is a dot-matrix crystal display module used specially to display letters, numbers, symbol, etc. It's composed of several dot-matrix character bits, each of which can display one character. The character bits are separated by one dot pitch and there's a gap between each line. Therefore, characters are spaced within and between lines.

The name 1602 LCD indicates that the display is 16x2, that is, two lines with 16 characters in each.

PCF8574-based I2C Interface Module:

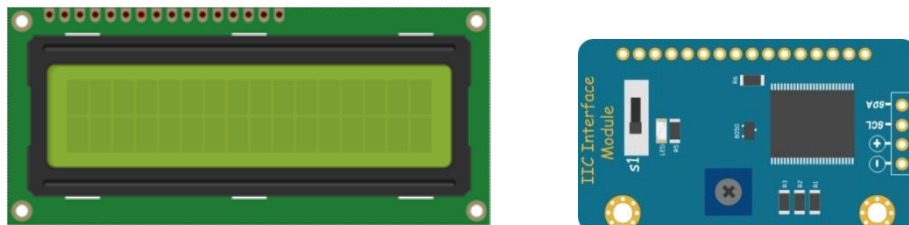
Since there are limited I/O ports on the Raspberry Pi, if you use them to drive the LCD1602, it needs many of the ports and there may be insufficient to connect other devices. To solve this problem, an IIC (or I2C) Interface Module based on PCF8574 is designed to extend the I/O ports of the Raspberry Pi. You only need two wires (SDA and SCL) to control the LCD1602 and save many ports for the board to connect more sensors.

Components

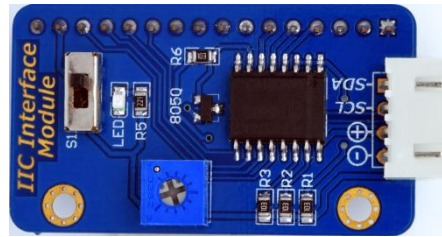
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * LCD1602
- 1 * Potentiometer Module
- 1 * I2C Interface Module
- 1 * 3-Pin Wires
- 1 * 4-Pin Wires
- Several Jumper Wires

Experimental Principle

The Fritzing image:



The Physical picture:

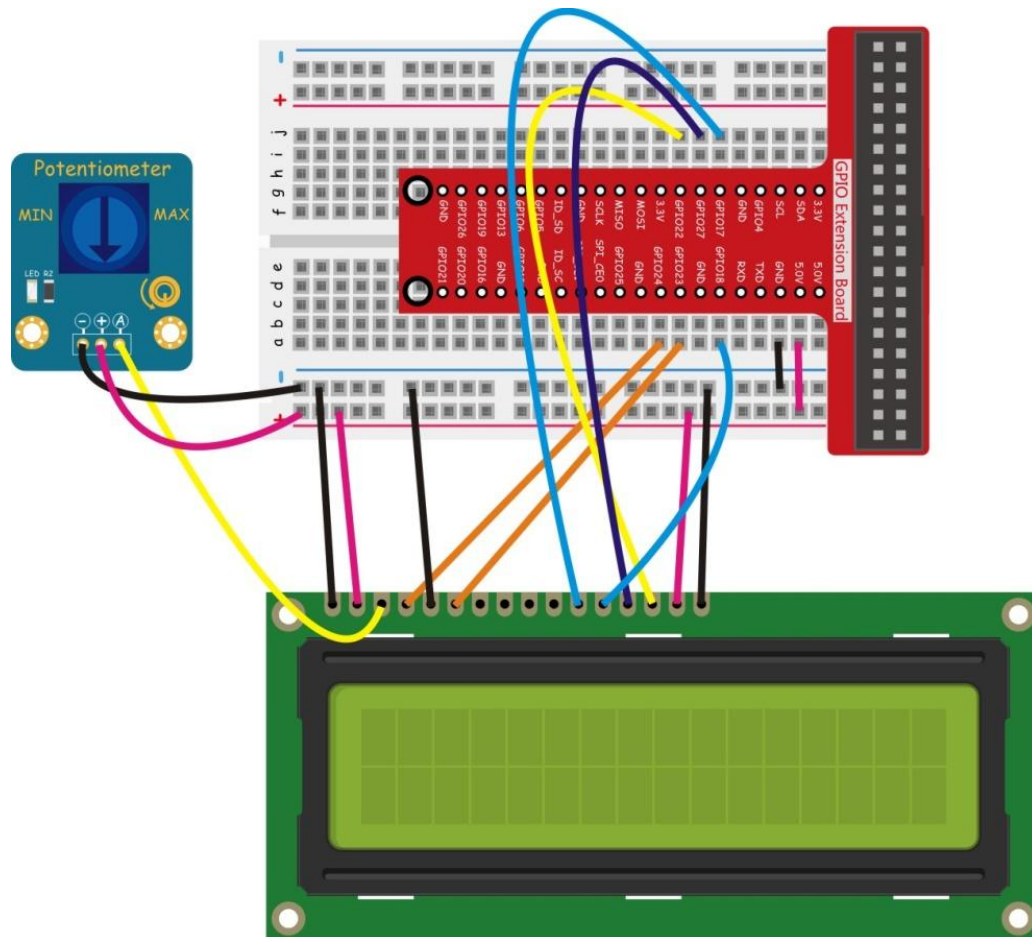


This lesson contains two experiments. The first experiment is controlling LCD1602 directly by the Raspberry Pi's GPIO, the second experiment is controlling via PCF8574-based IIC module.

Experiment 1:

Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/18_LCD1602/lcd1602.c)

Step 3: Compile

```
$ sudo gcc lcd1602.c -o lcd1602 -lwiringPi -lwiringPiDev
```


Step 4: Run

```
$ sudo ./lcd1602
```

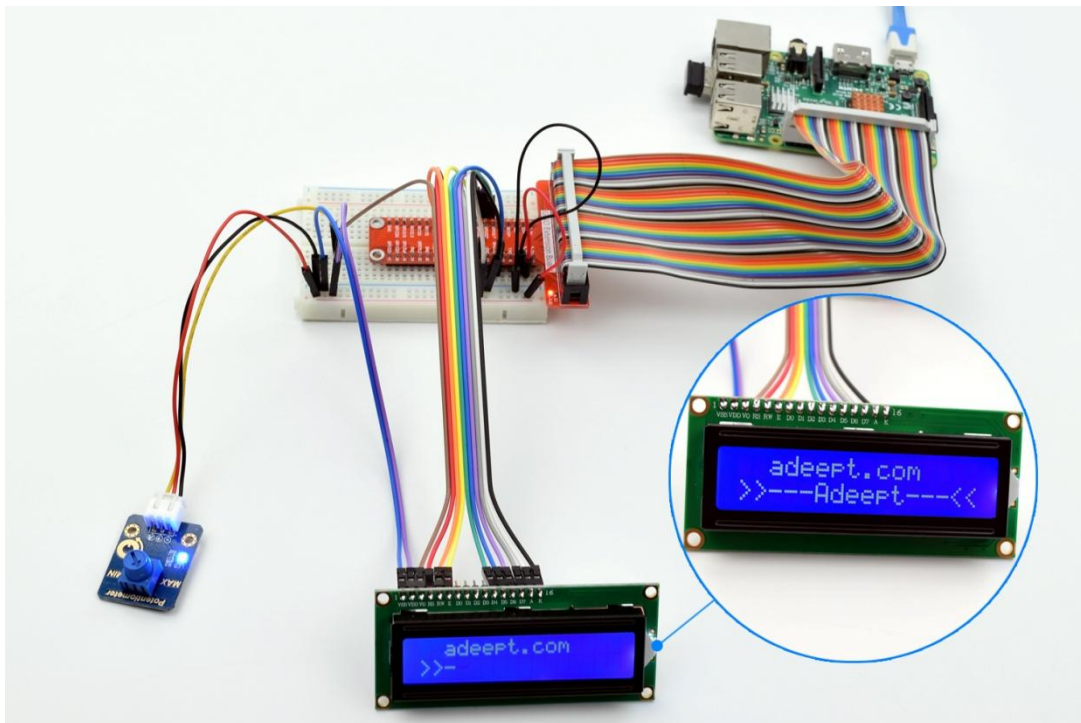
For Python users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/18_LCD1602/lcd1602.py)

Step 3: Run

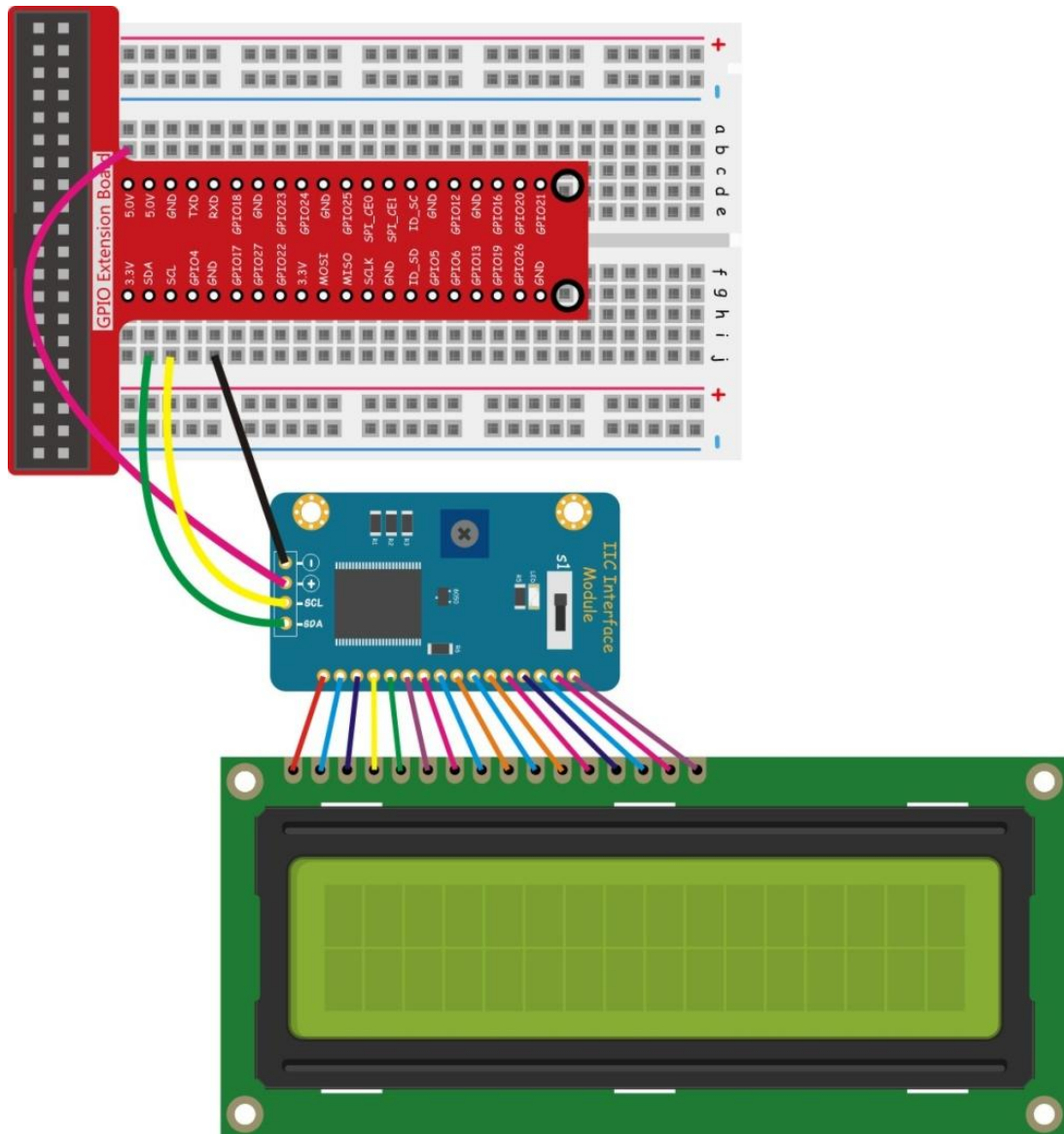
```
$ sudo python lcd1602.py
```



Experiment 2:

Procedures

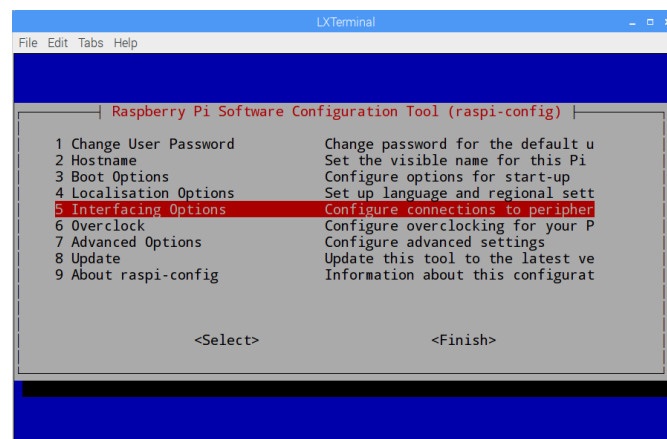
Step 1: Build the circuit



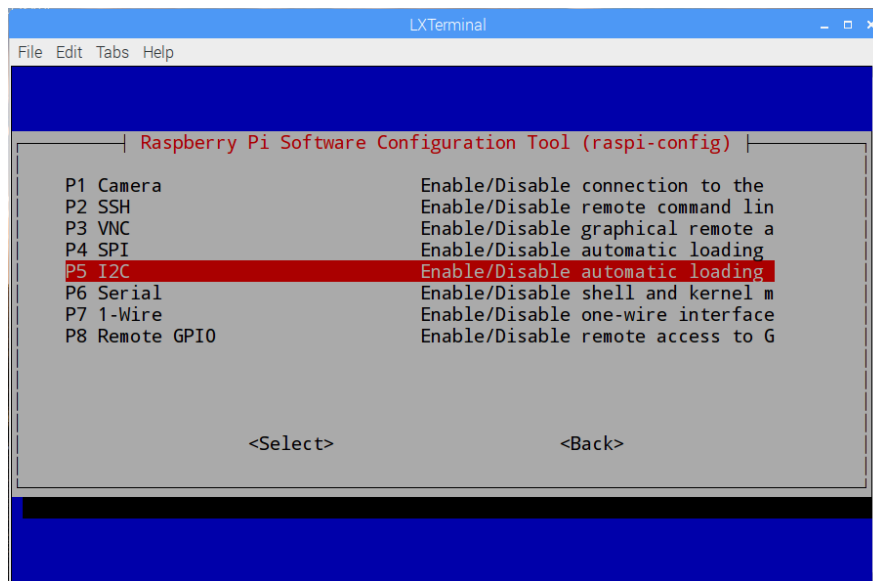
Step 2: The I2C of the Raspberry Pi is not turned on by default, we can use [raspi-config](#) to enable it

```
$ sudo raspi-config
```

Use the down arrow to select **5 Interfacing Options**



Arrow down to **P5 I2C**



Select **yes** when it asks you to enable I2C.

Also select **yes** when it asks about automatically loading the kernel module.

Use the right arrow to select the **<Finish>** button.

Select **yes** when it asks to reboot.

The system will reboot. when it comes back up, log in and enter the following command:

```
$ ls /dev/*i2c*
```

```
root@raspberrypi:/# ls /dev/*i2c*
/dev/i2c-1
```

The Pi should respond with

```
/dev/i2c-1
```

Which represents the user-mode I2C interface.

For C language users:

Step 3: Edit and save the code with vim or nano.

(code path: /home/Adeept_Compact_Sensor_Kit_for_RPi/C/18_LCD1602/i2c_lcd1602.c)

Step 4: Compile

```
$ sudo gcc i2c_lcd1602.c -o i2c_lcd1602 -lwiringPi -lwiringPiDev
```

Step 5: Run

```
$ sudo ./i2c_lcd1602
```

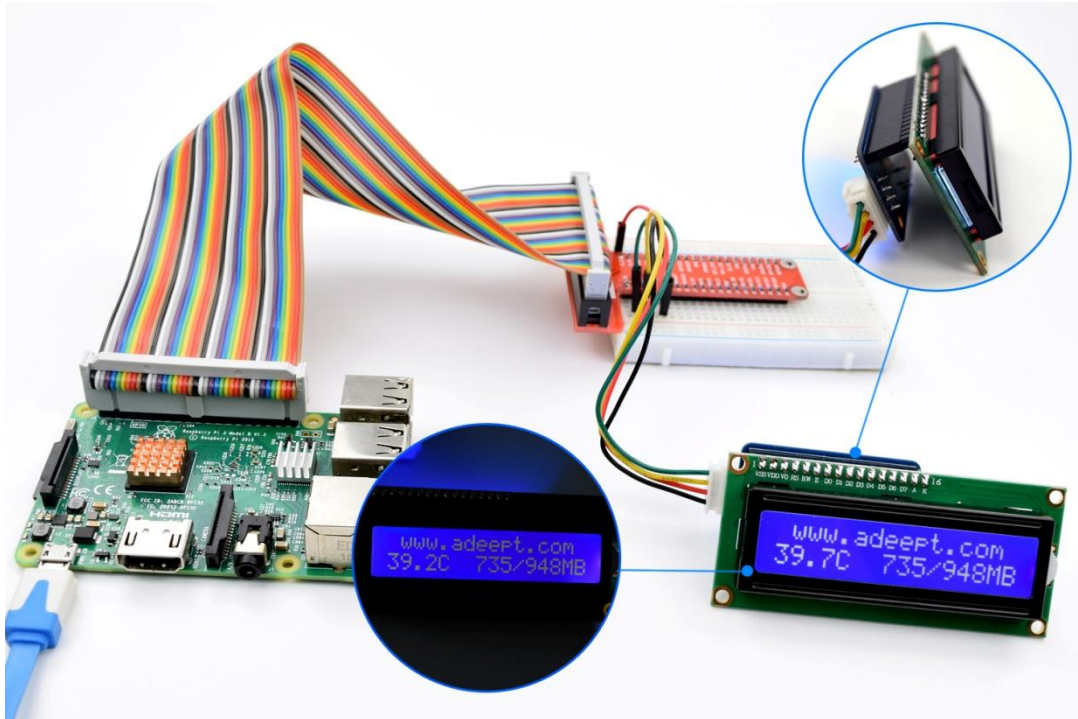
For Python users:

Step 3: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/18_LCD1602/i2c_lcd1602.py)

Step 4: Run

```
$ sudo python i2c_lcd1602.py
```



NOTE:

If characters are not displayed on the LCD, try to turn the potentiometer(a blue knob) for contrast adjustment on the I2C Interface Module. Or, you can toggle the switch on the module to see whether the backlight is on.

Lesson 19 How to Make a Simple Thermometer(1)

Introduction

In this lesson, we will learn how to make a simple thermometer based on DS18B20 and segment display module.

Components

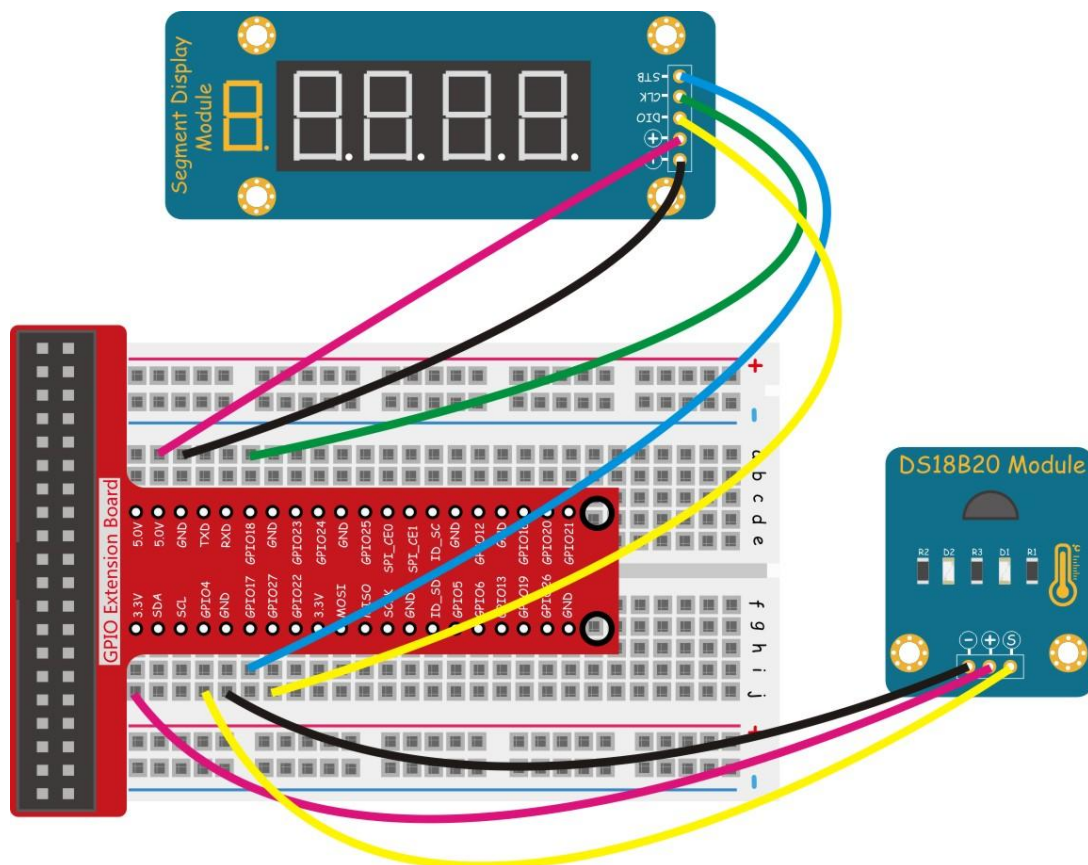
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * DS18B20 Module
- 1 * Segment Display Module
- 1 * 5-Pin Wires
- 1 * 3-Pin Wires

Experimental Principle

In this experiment, we program the Raspberry Pi to read DS18B20 temperature sensor, and then display the temperature on the segment display module.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Compact_Sensor_Kit_for_RPi/C/19_thermometer_1)

Step 3: Compile

```
$ sudo ./build.sh
```

Step 4: Run

```
$ sudo ./main
```

For Python users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Compact_Sensor_Kit_for_RPi/Python/19_thermometer_1/)

Step 3: Run

```
$ sudo python main.py
```

Now you can see the temperature shown on the segment display.



Lesson 20 How to Make a Simple Thermometer(2)

Introduction

In this lesson, we will learn how to make a simple thermometer based on DS18B20 and LCD1602.

Components

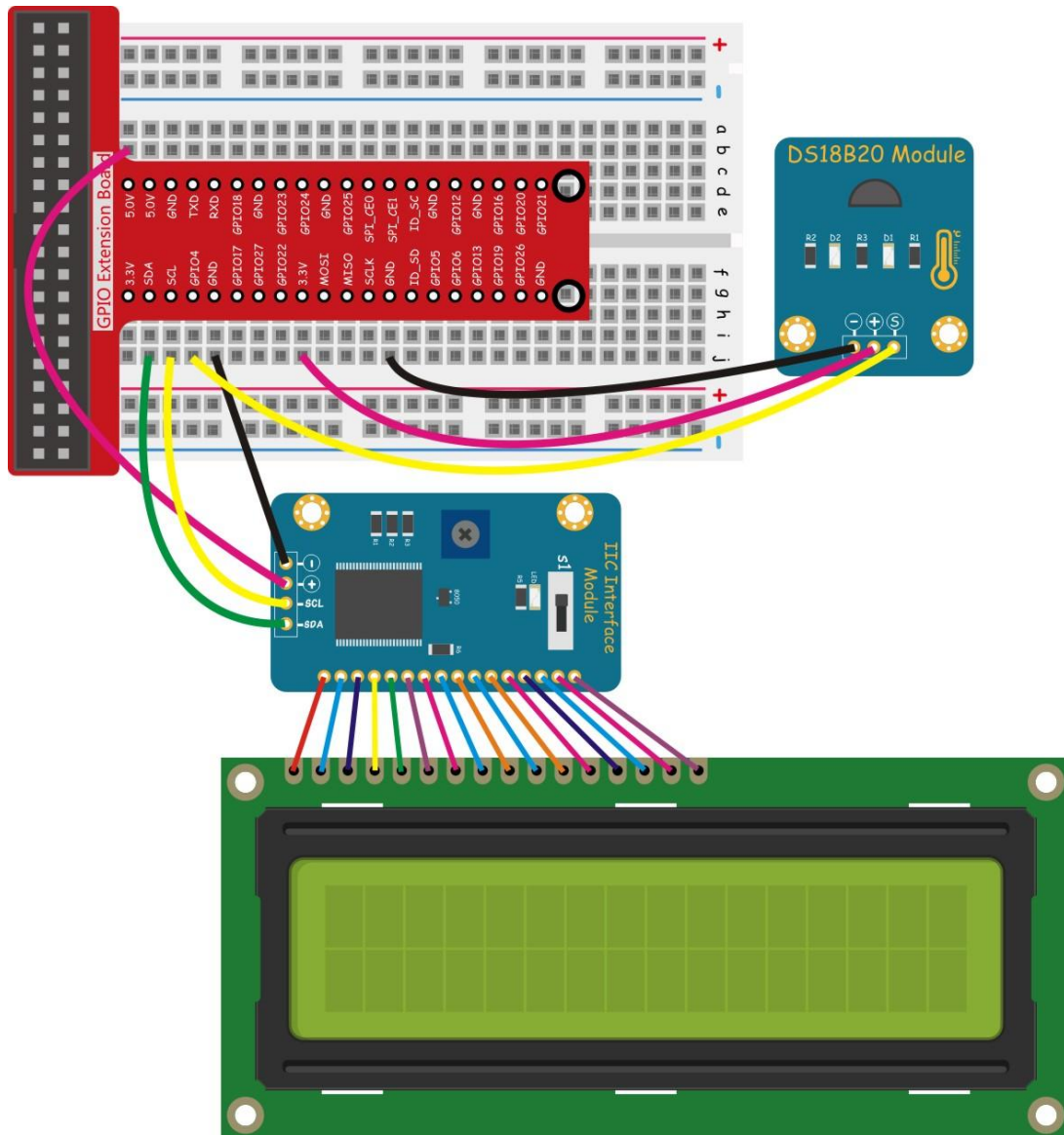
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * DS18B20 Module
- 1 * I2C Interface Module
- 1 * LCD1602
- 1 * 4-Pin Wires
- 1 * 3-Pin Wires

Experimental Principle

In this experiment, we program the Raspberry Pi to read DS18B20 temperature sensor, and then display the temperature on the LCD1602.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/20_thermometer_2)

Step 3: Compile

```
$ sudo ./build.sh
```

Step 4: Run

```
$ sudo ./main
```

For Python users:

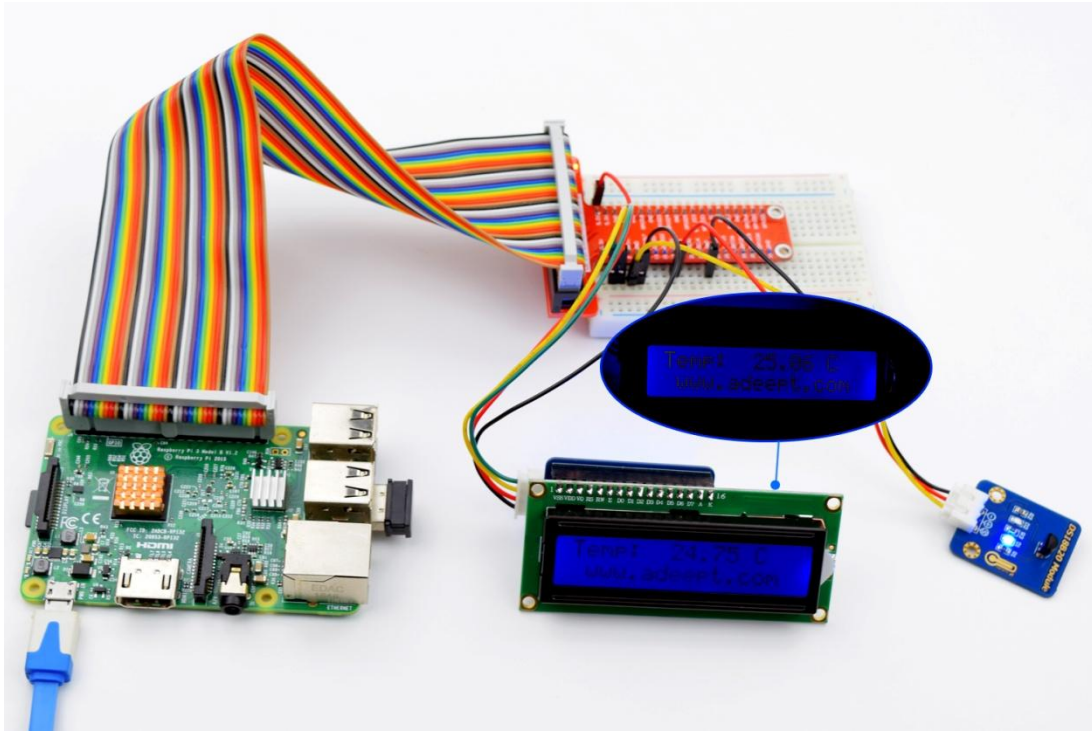
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/20_thermometer_2)

Step 3: Run

```
$ sudo python main.py
```

Now you can see the temperature shown on the LCD1602.



Lesson 21 Make a Distance Measuring Device

Introduction

In this lesson, we will learn how to make a distance measuring device based on ultrasonic module and LCD1602.

Components

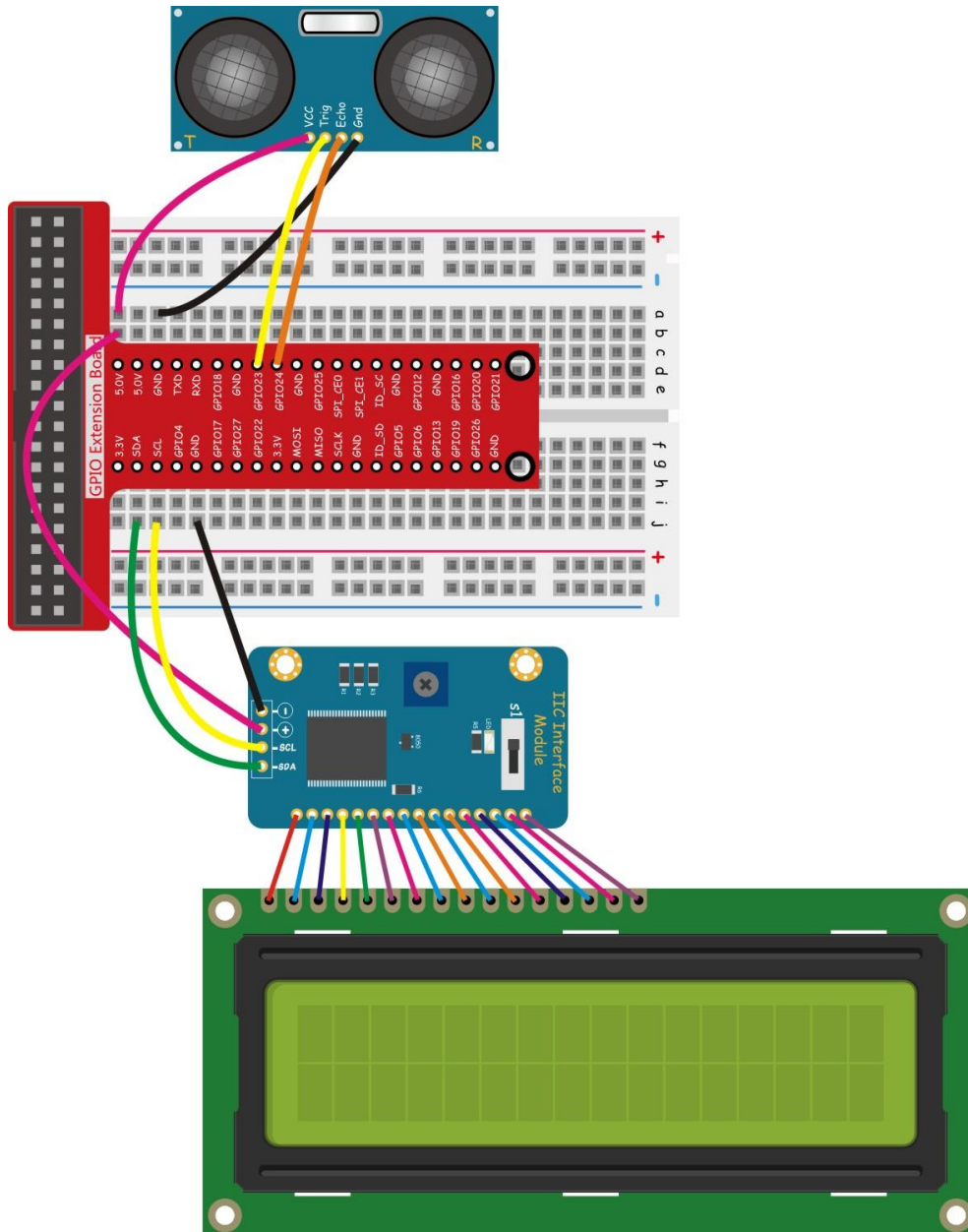
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * LCD1602
- 1 * I2C Interface Module
- 1 * Ultrasonic Sensor Module
- 1 * 4-Pin Wires
- 4 * Jumper Wires

Experimental Principle

In this experiment, we program the Raspberry Pi to detect the distance between the obstacle and ultrasonic module, and then display the data on the LCD1602.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/21_measureDis)

Step 3: Compile

```
$ sudo ./build.sh
```

Step 4: Run

```
$ sudo ./main
```

For Python users:

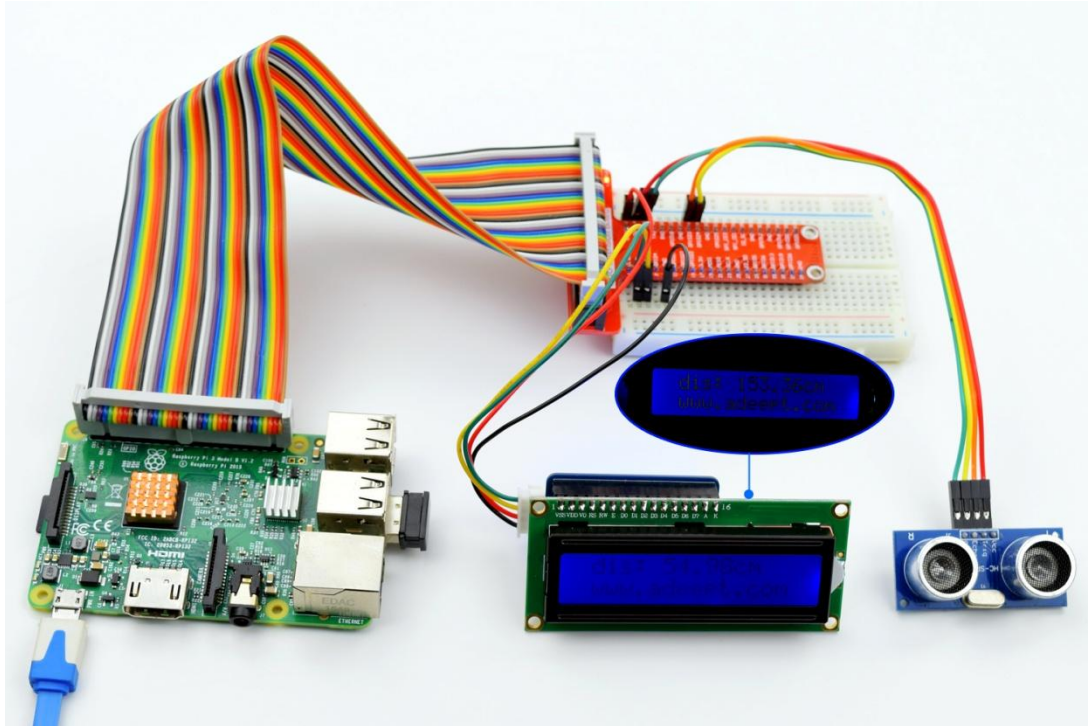
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/21_measureDis/)

Step 3: Run

```
$ sudo python main.py
```

Now, you will see the distance to the obstacle at front of the Ultrasonic Distance Sensor module displayed on the LCD1602.



Lesson 22 How to Make a Simple Voltmeter(1)

Introduction

In this lesson, we will learn how to make a simple voltmeter based on ADC0832 and segment display module.

Components

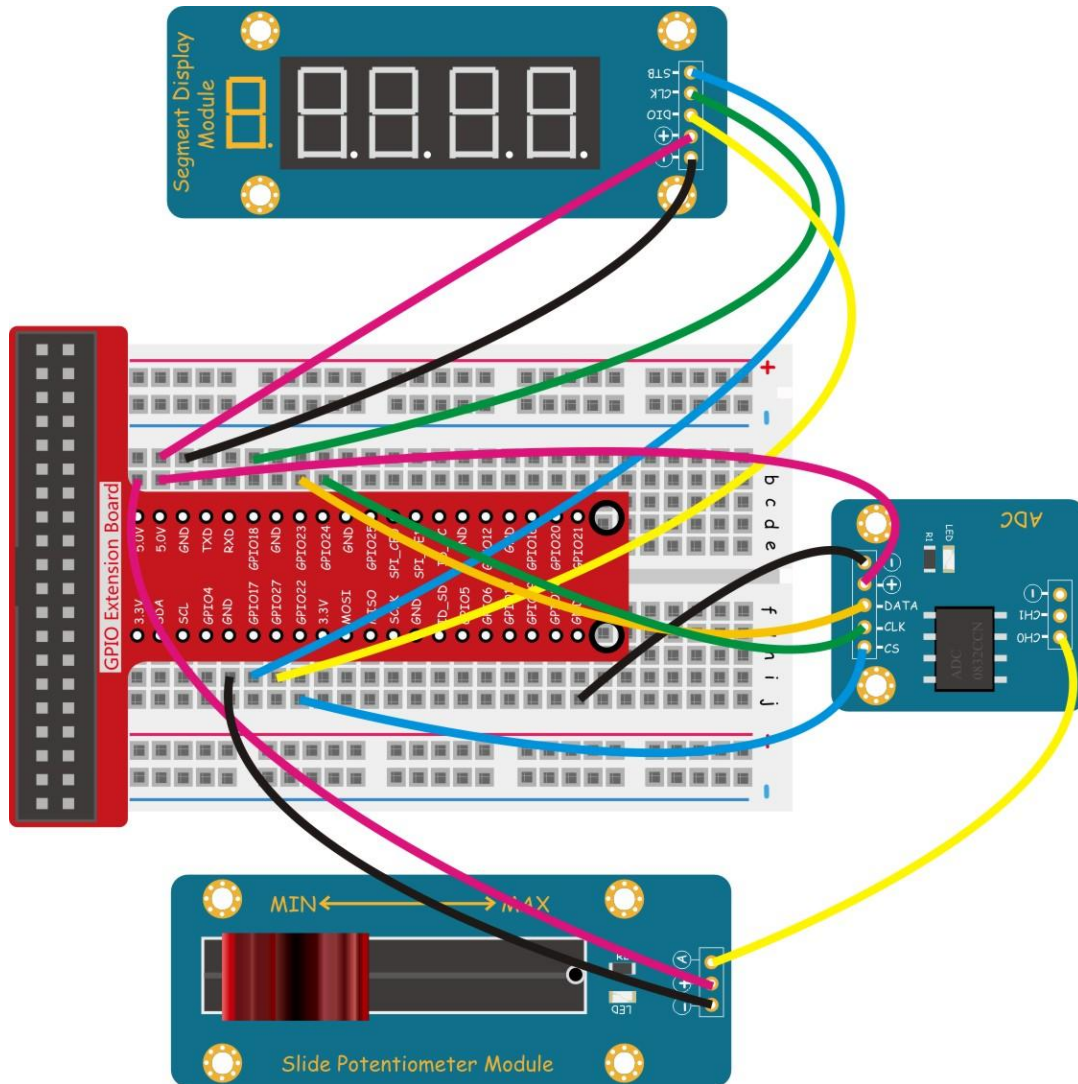
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * ADC0832 Module
- 1 * Segment Display Module
- 1 * Slide Potentiometer Module
- 2 * 5-Pin Wires
- 2 * 3-Pin Wires

Experimental Principle

In this experiment, we program the Raspberry Pi to read voltage(DC: 0-5V) via ADC0832, and then display the voltage value on the segment display module.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/22_voltmeter_1)

Step 3: Compile

```
$ sudo ./build.sh
```

Step 4: Run

```
$ sudo ./main
```

For Python users:

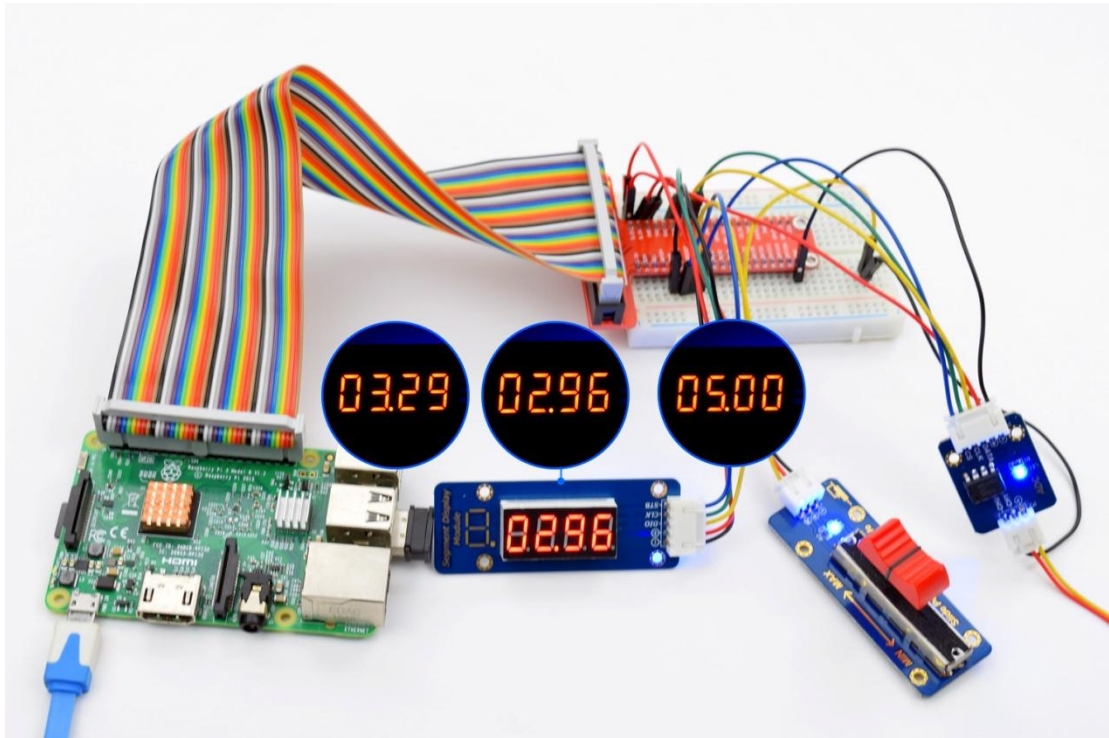
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/22_voltmeter_1)

Step 3: Run

```
$ sudo python main.py
```


Now you can see the voltage value shown on the segment display.



Lesson 23 How to Make a Simple Voltmeter(2)

Introduction

In this lesson, we will learn how to make a simple voltmeter based on ADC0832 and LCD1602.

Components

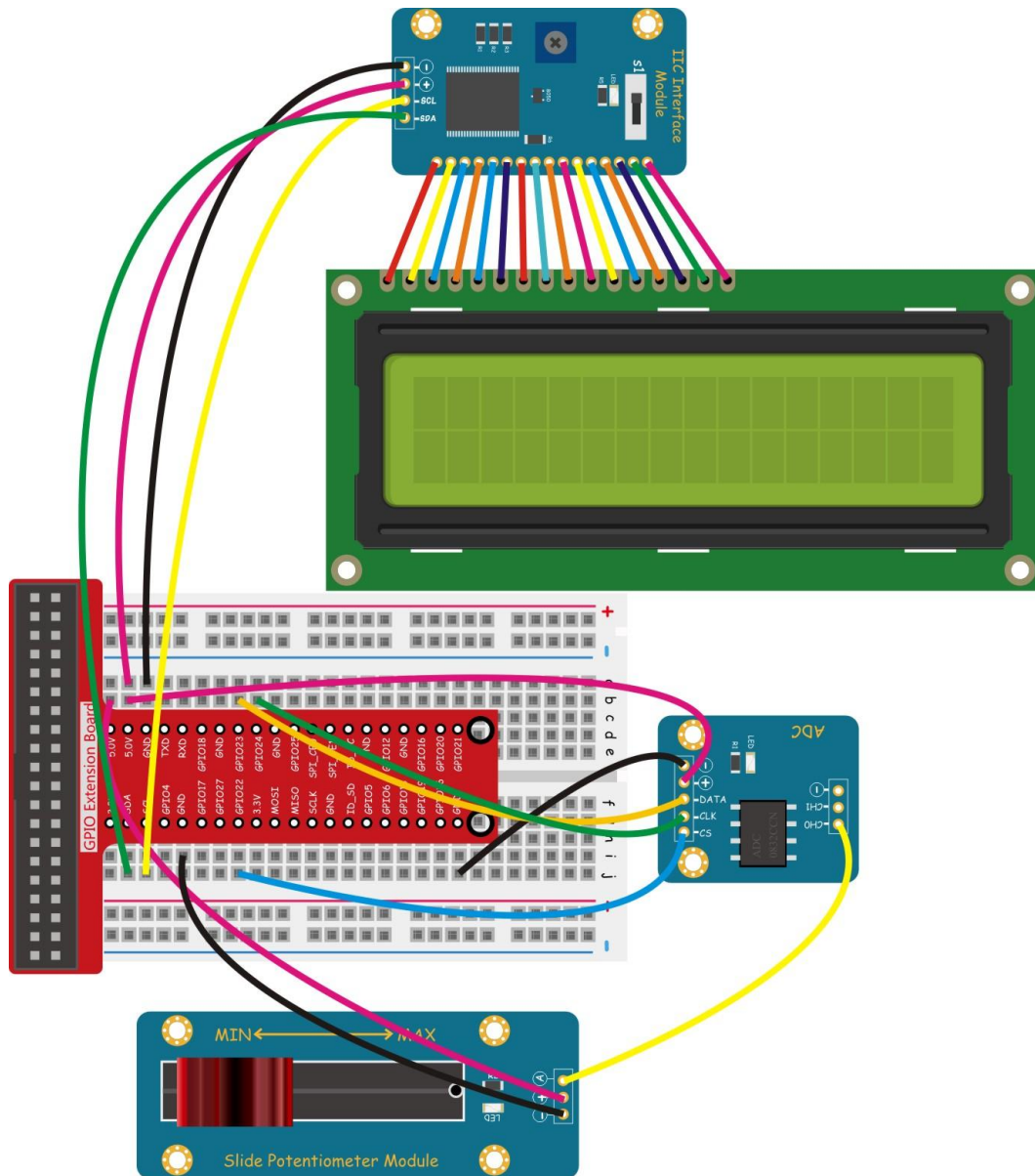
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * ADC0832 Module
- 1 * I2C Interface Module
- 1 * Slide Potentiometer Module
- 1 * LCD1602
- 2 * 3-Pin Wires
- 1 * 4-Pin Wires
- 1 * 5-Pin Wires

Experimental Principle

In this experiment, we program the Raspberry Pi to read voltage(DC0-5V) via ADC0832, and then display the voltage value on the LCD1602.

Experimental Procedures

Step 1: Build the circuit



For C language users:

Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/C/23_voltmeter_2)

Step 3: Compile

```
$ sudo ./build.sh
```

Step 4: Run

```
$ sudo ./main
```

For Python users:

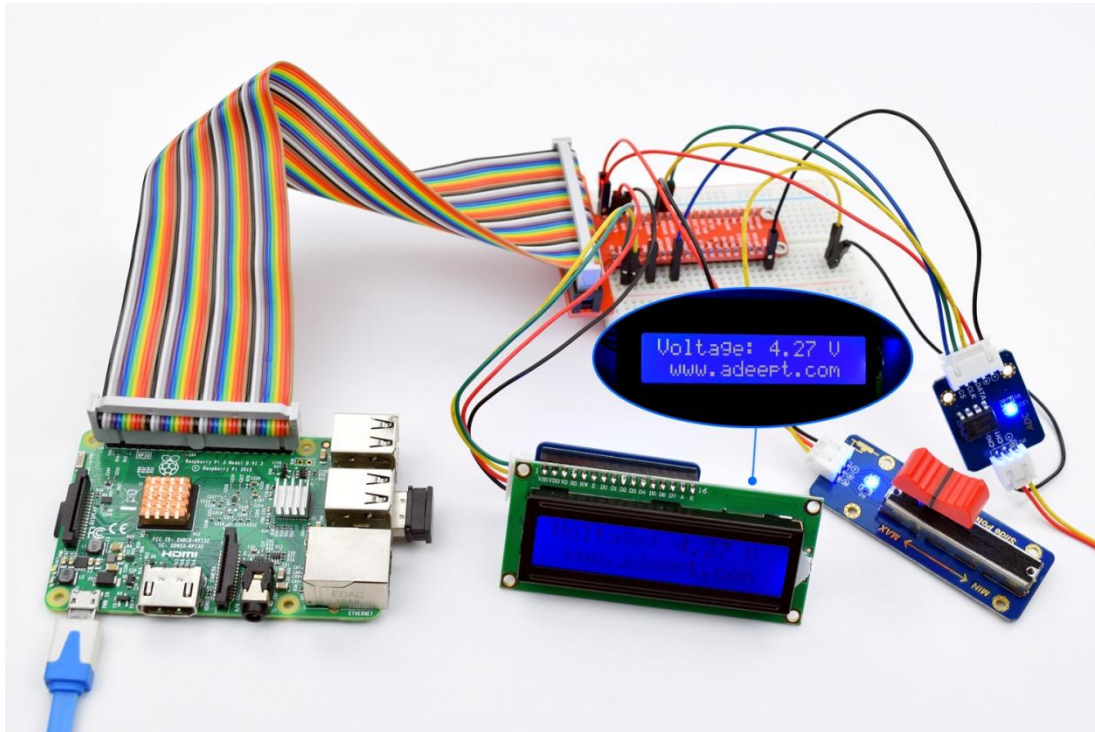
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adept_Compact_Sensor_Kit_for_RPi/Python/23_voltmeter_2)

Step 3: Run

```
$ sudo python main.py
```

Now you can see the voltage value shown on the LCD1602.





www.adeept.com

Adept



Adept



support@adeept.com /



www.adeept.com