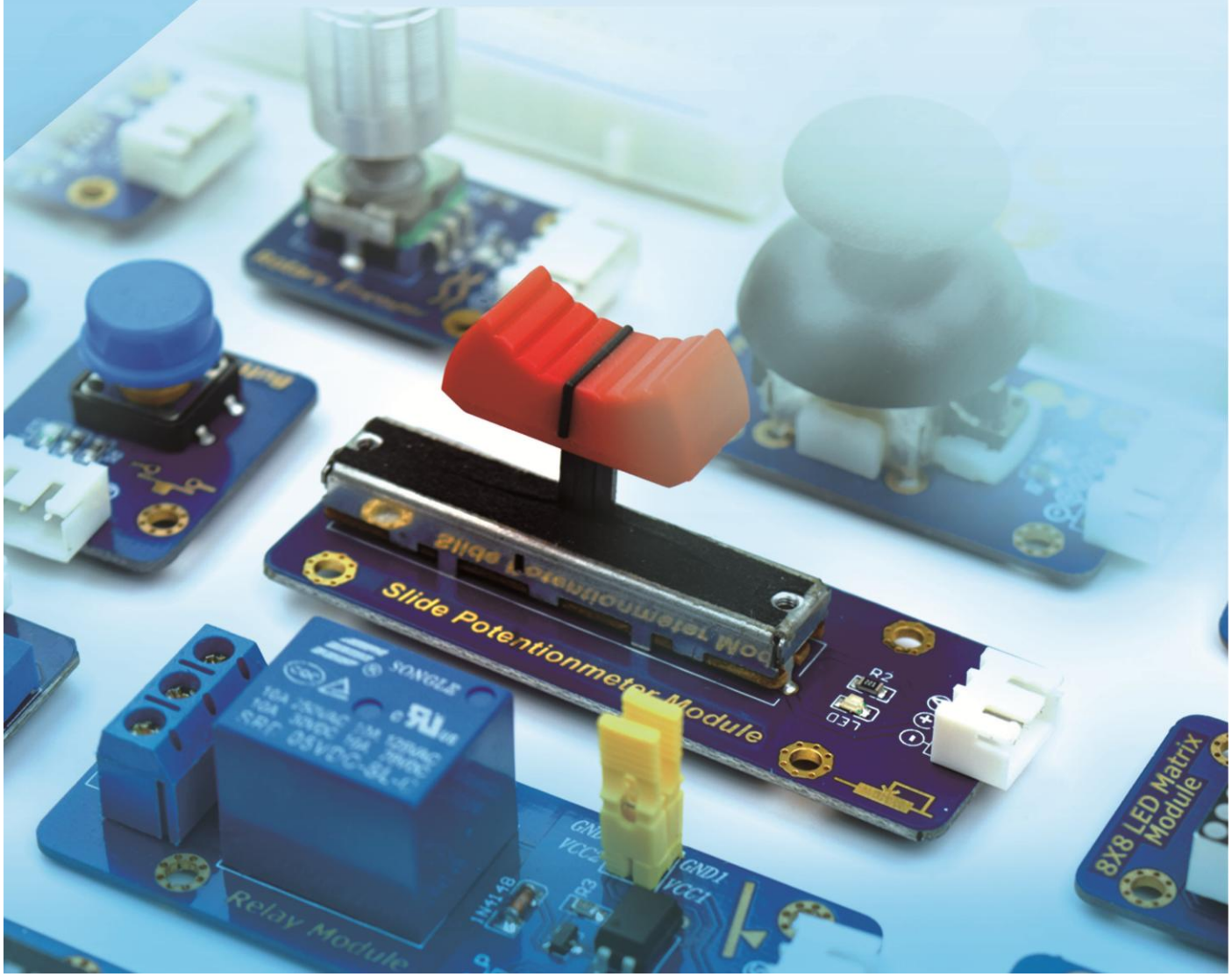
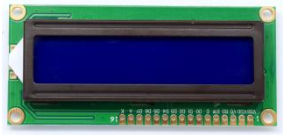
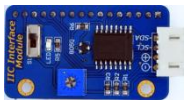
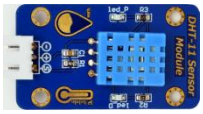
Adeept

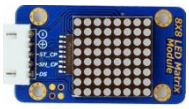
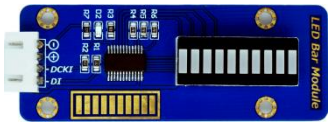
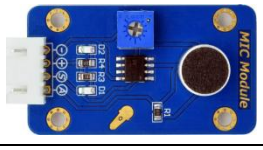
ULTIMATE SENSOR KIT FOR RASPBERRY PI

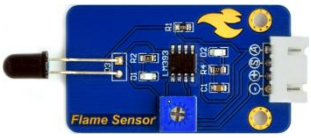
Sharing Perfects Innovation

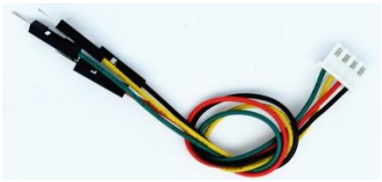


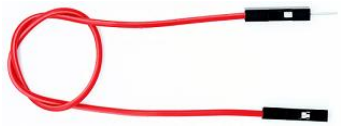
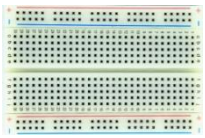
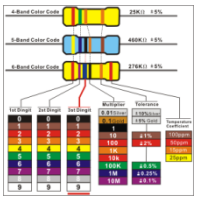
Package List

No.	Name	Picture	Qty
1	LCD1602		1
2	I2C Interface Module		1
3	DHT-11 Temperature and humidity Sensor		1
4	DS18B20 Digital temperature Sensor		1
5	Ultrasonic Distance Sensor Module		1
6	ADC0832 Module		1
7	BMP180 Barometer Sensor		1
8	ADXL345 Accelerometer		1
9	PS2 Joystick Module		1
10	Relay Module		1
11	DC Motor Module		1

12	Segment Display Module		1
13	8x8 LED Matrix Module		1
14	Potentiometer Module		1
15	Slide Potentiometer Module		1
16	Rotary Encoder Module		1
17	PIR Sensor Module		1
18	MQ-2 Gas Sensor Module		1
19	LED Bar Graph Module		1
20	Active Buzzer Module		1
21	Passive Buzzer Module		1
22	MIC Module		1
23	Touch Button Module		1

24	Line Finder Module		1
25	Flame Sensor Module		1
26	Vibration Sensor Module		1
27	CM Module		1
28	Water Level Sensor Module		1
29	Soil Moisture Sensor Module		1
30	Photoresistor Module		1
31	Analog Temperature Sensor(Thermistor Module)		1
32	Hall Sensor Module		1
33	Limit Switch Module		1

34	Reed Module		1
35	RGB LED Module		1
36	Laser Transmitter Module		1
37	Laser Receiver Module		1
38	Button Module		4
39	LED Module		4
40	GPIO Extension Board		1
41	40P GPIO Cable		1
42	3-Pin Wires		8
43	4-Pin Wires		5

44	5-Pin Wires		3
45	Hookup Wire Set		1
46	2-Pin Female to Female Wires		1
47	Male to Female Jumper Wires		20
48	Breadboard		1
49	Band Resistor Card		1

Preface

About Adeept

Adeept 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

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

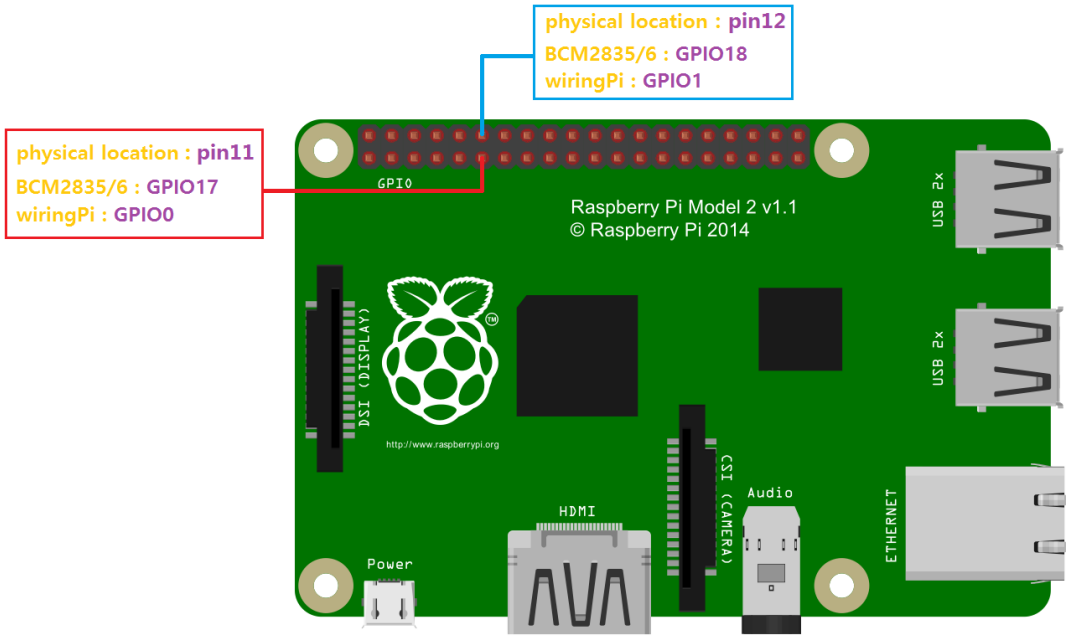
WiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	WiringPi Pin
–	–	3.3v	1 | 2	5v	–	–
8	R1:0/R2:2	SDA1	3 | 4	5v	–	–
9	R1:1/R2:3	SCL1	5 | 6	0V	–	–
7	4	GPIO7	7 | 8	TXD	14	15
–	–	0V	9 | 10	RXD	15	16
0	17	GPIO0	11 | 12	GPIO1	18	1
2	R1:21/R2:27	GPIO2	13 | 14	0V	–	–
3	22	GPIO3	15 | 16	GPIO4	23	4
–	–	3.3v	17 | 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.


physical location : pin12
BCM2835/6 : GPIO18
wiringPi : GPIO1
physical location : pin11
BCM2835/6 : GPIO17
wiringPi : GPIO0
GPIO
Raspberry Pi Model 2 v1.1
© Raspberry Pi 2014
USB 2x
USB 2x
DSI (DISPLAY)
http://www.raspberrypi.org
CSI (CAMERA)
Audio
ETHERNET
HDMI
Power

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

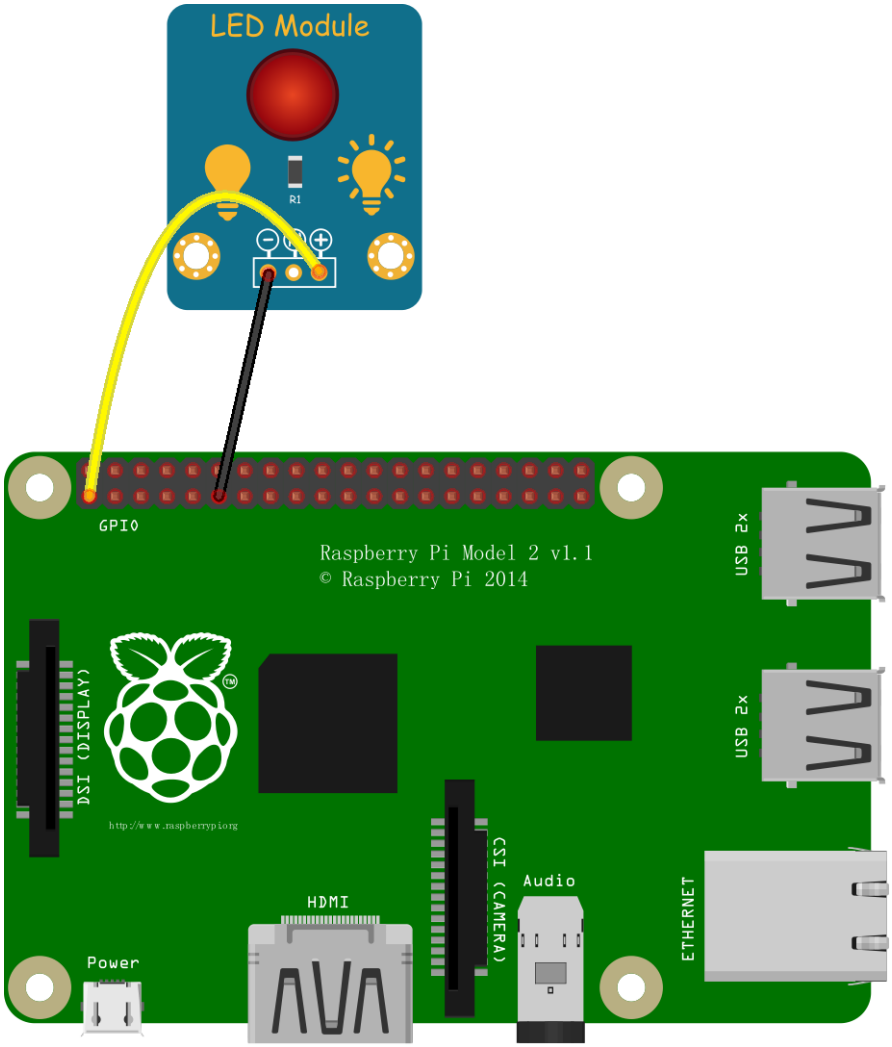
```
pi@raspberrypi ~ $ sudo gpio readall
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   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 |   |      | 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 |   |      | 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 2---+---+------+---------+-----+-----+
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:

C Language Source Code:

$ git clone https://github.com/adeept/Adeept_Sensor_Kit_for_RPi_C_Code

Python Source Code:

$ git clone https://github.com/adeept/Adeept_Sensor_Kit_for_RPi_Python_Code

Lesson 1 Blinking LED

Introduction

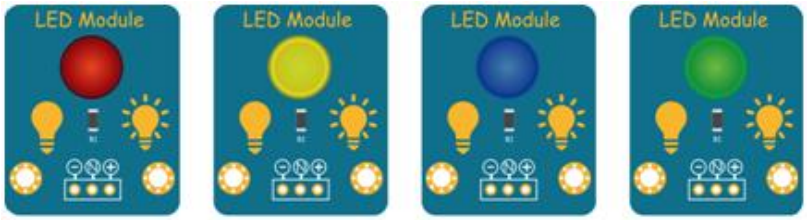
LED is usually used in office lighting, furniture, decoration, sign board, streetlight, etc.

Components

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

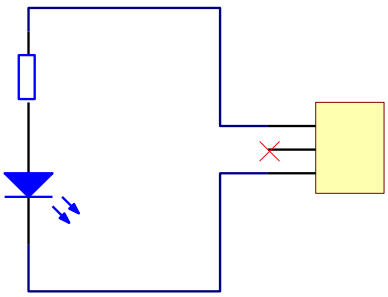
Experimental Principle

The Fritzing images:



The Physical picture:

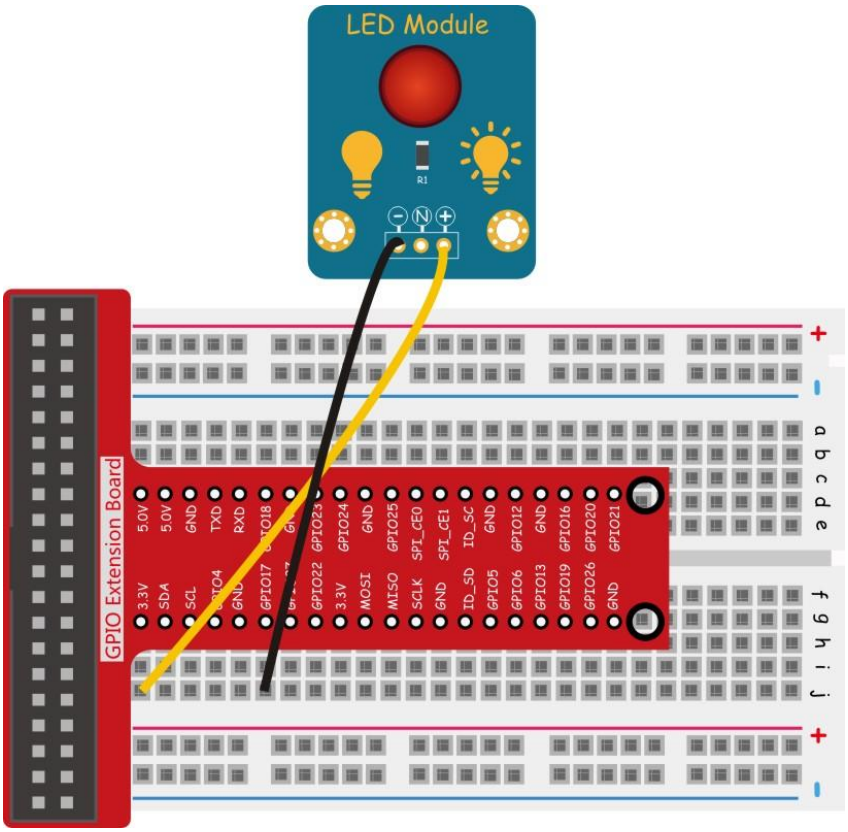
The schematic diagram:



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/Adeept_Sensor_Kit_for_RPi_C_Code/01_LED/blinkingLed.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./led
```

For Python users:

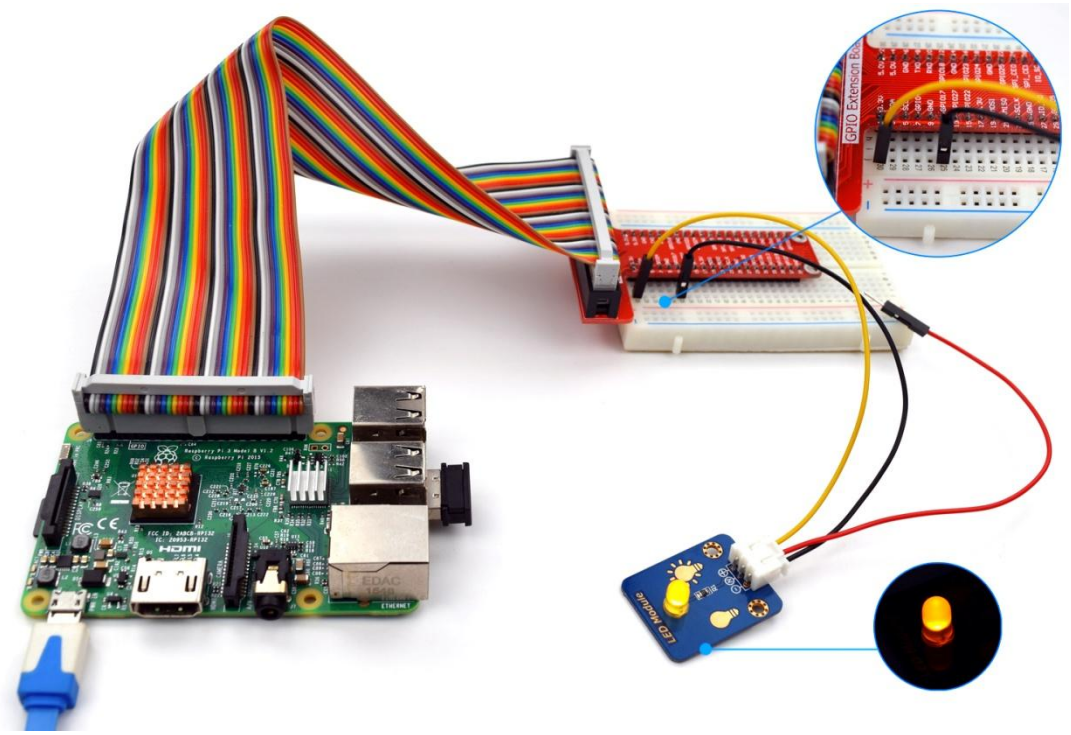
Step 2: Edit and save the code with vim or nano.

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

Step 3: Run

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

Now you can see the LED blinking.


GPIO Extension Board
LED Module

Lesson 2 Controlling an LED by Button

Introduction

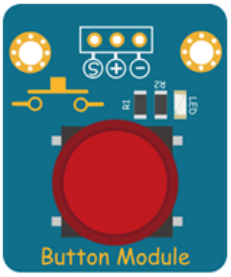
A button is an electronic switch and usually used for device control.

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:

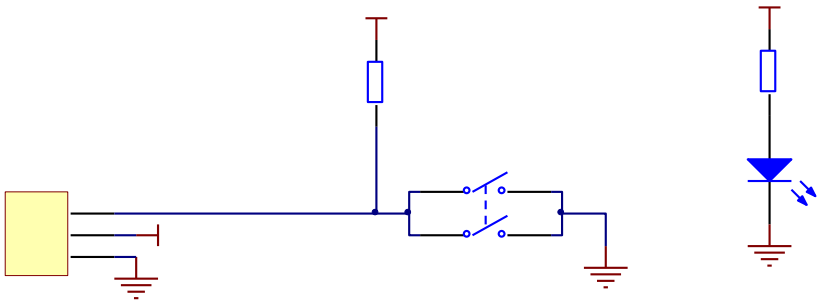


Pin definition:

S	Output
+	VCC
-	GND

The Physical picture:

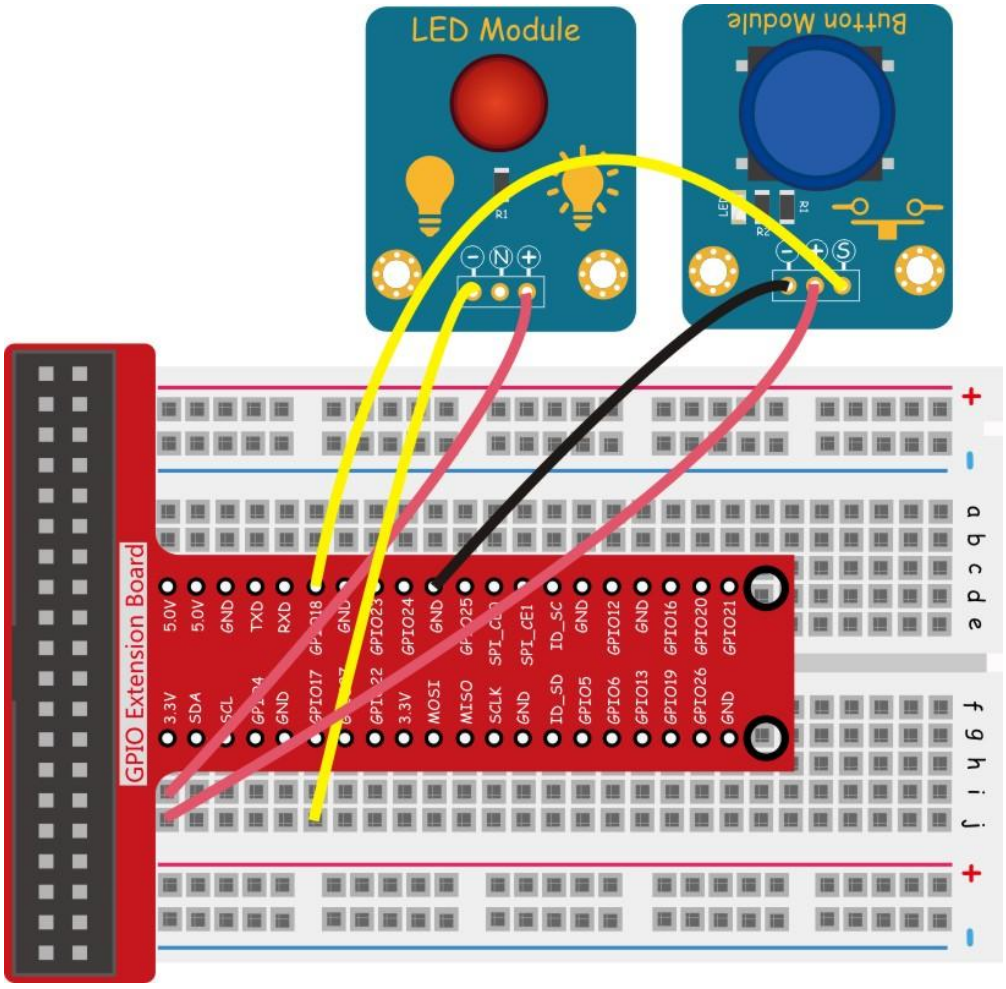
The schematic diagram:



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/Adeept_Sensor_Kit_for_RPi_C_Code/02_btnAndLed/btnAndLed_1.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./btnAndLed
```

For Python users:

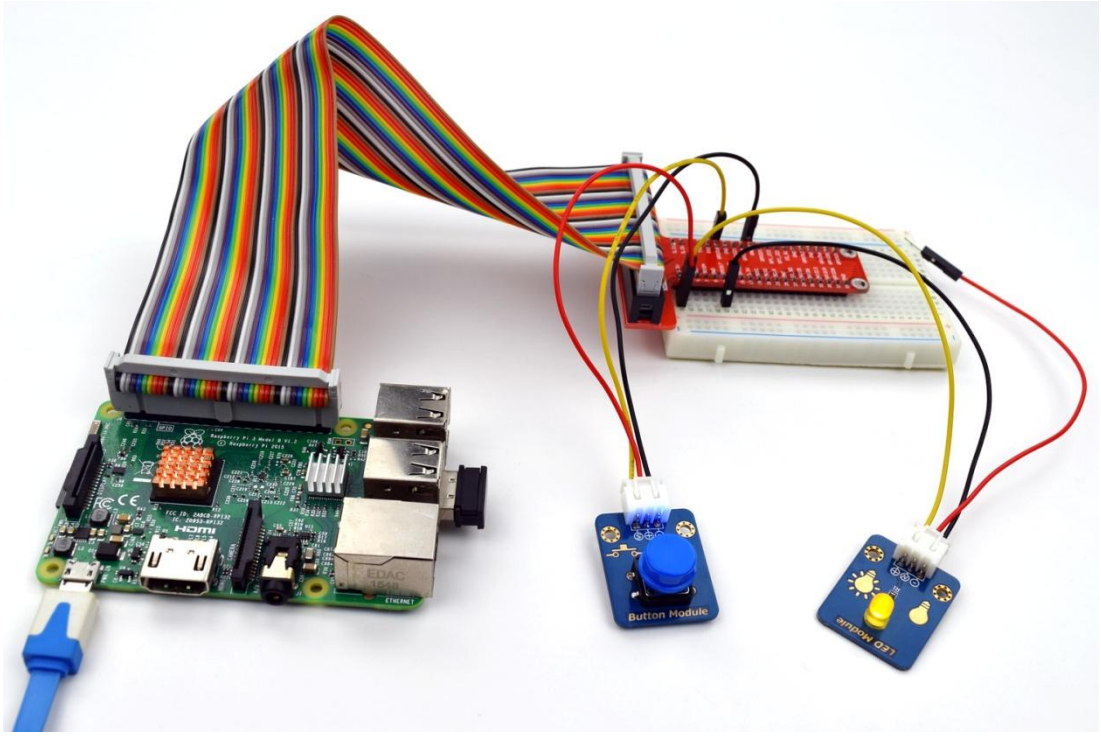
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/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

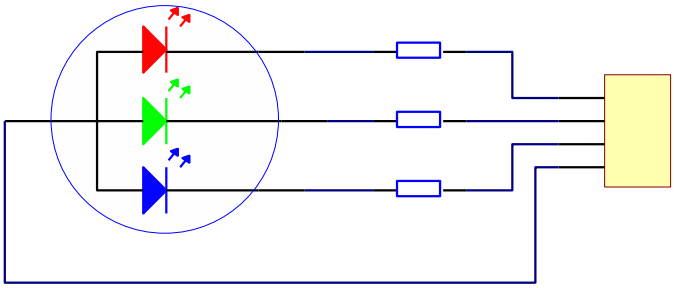
The Fritzing image:

The Physical picture:

Pin definition:

B	Blue Channel
G	Green Channel
R	Red Channel
+	VCC

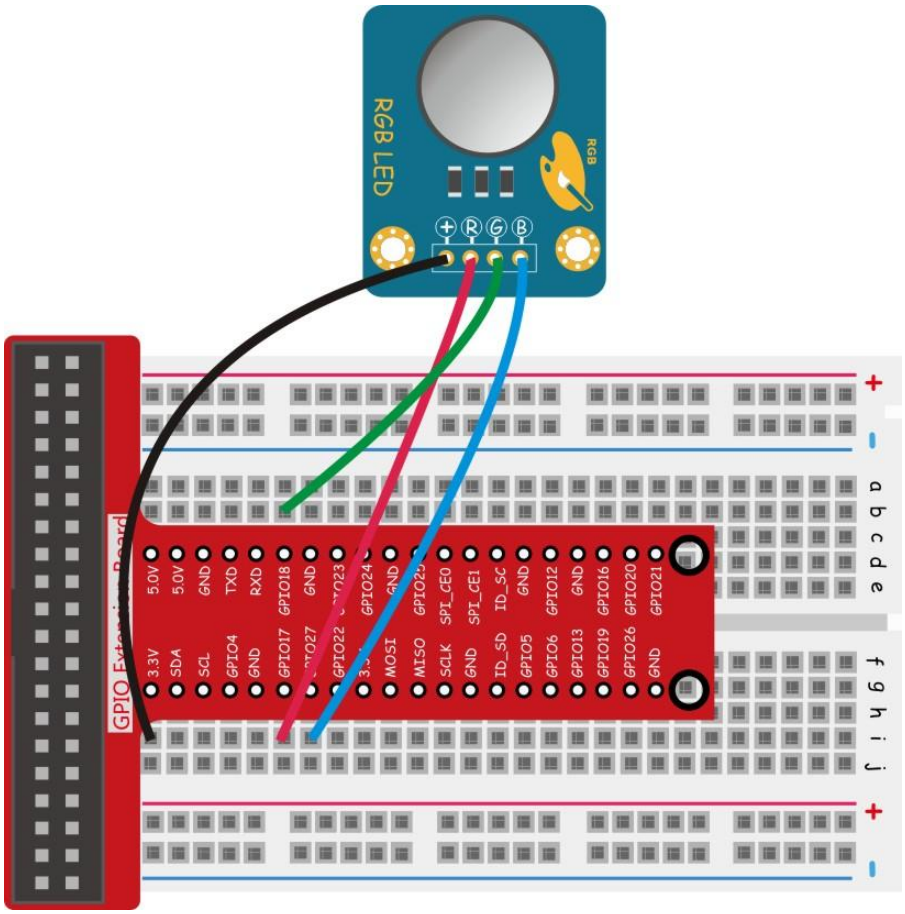
The schematic diagram:



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/Adeept_Sensor_Kit_for_RPi_C_Code/03_rgbLed/rgbLed.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./rgbLed
```

For Python users:

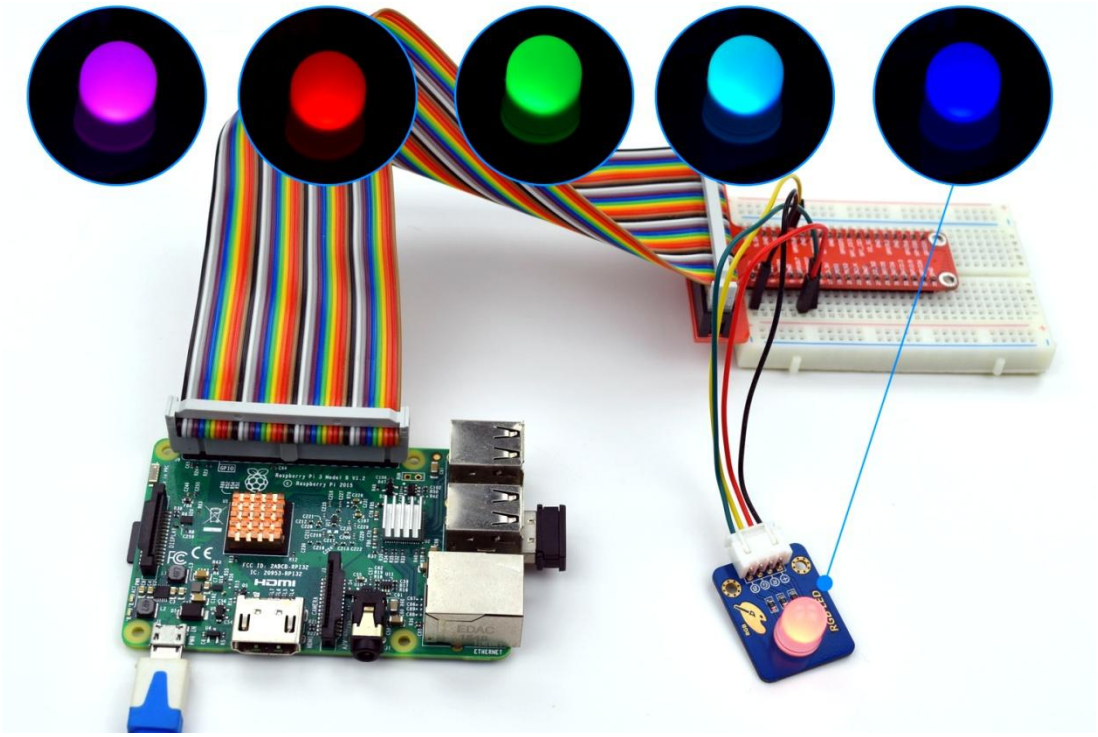
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/03_rgbLed.py)

Step 3: Run

```
$ sudo ./03_rgbLed.py
```

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



Lesson 4 Active Buzzer

Introduction

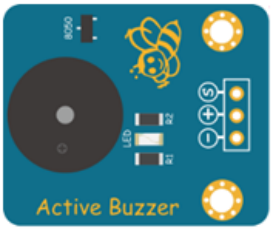
Buzzers are a type of integrated electronic alarm devices and powered by DC supply. They are widely applied for sound producing in devices such as computer, printer, duplicator, alarm, electronic toy, vehicle electronic equipment, phone, and timer and so on. Active buzzers can make sounds constantly 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:

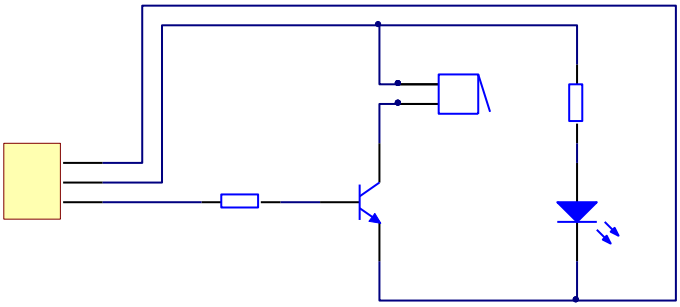


The Physical picture:

Pin definition:

S	Input
+	VCC
-	GND

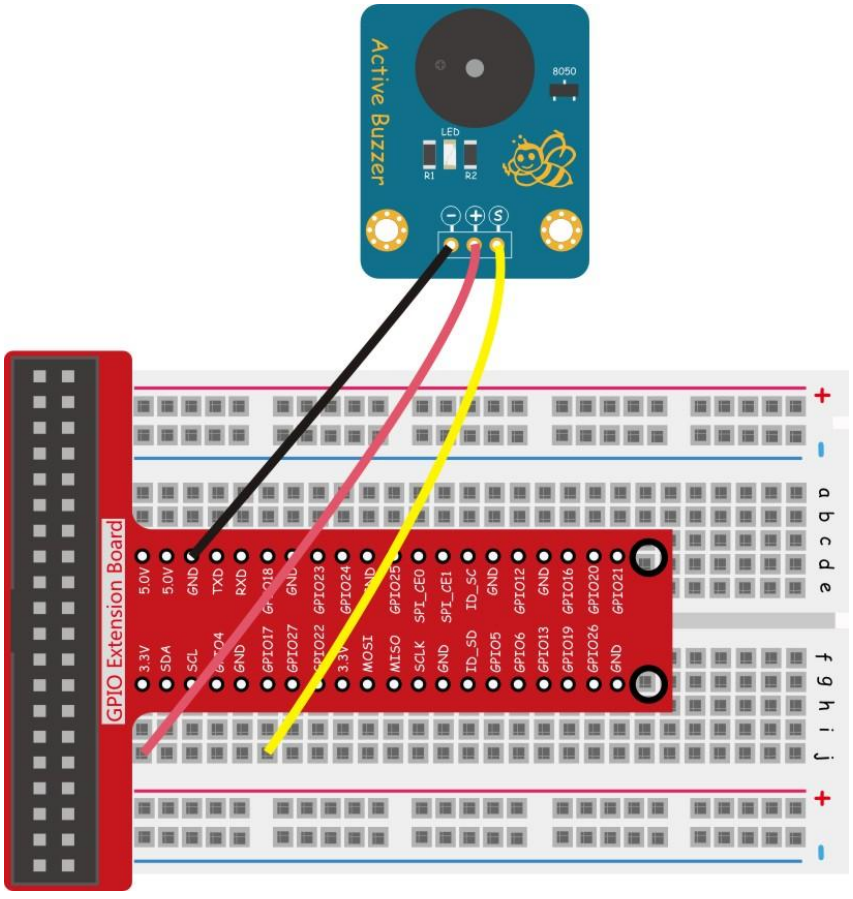
The schematic diagram:



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/Adeept_Sensor_Kit_for_RPi_C_Code/04_activeBuzzer/buzzer.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./buzzer
```

For Python users:

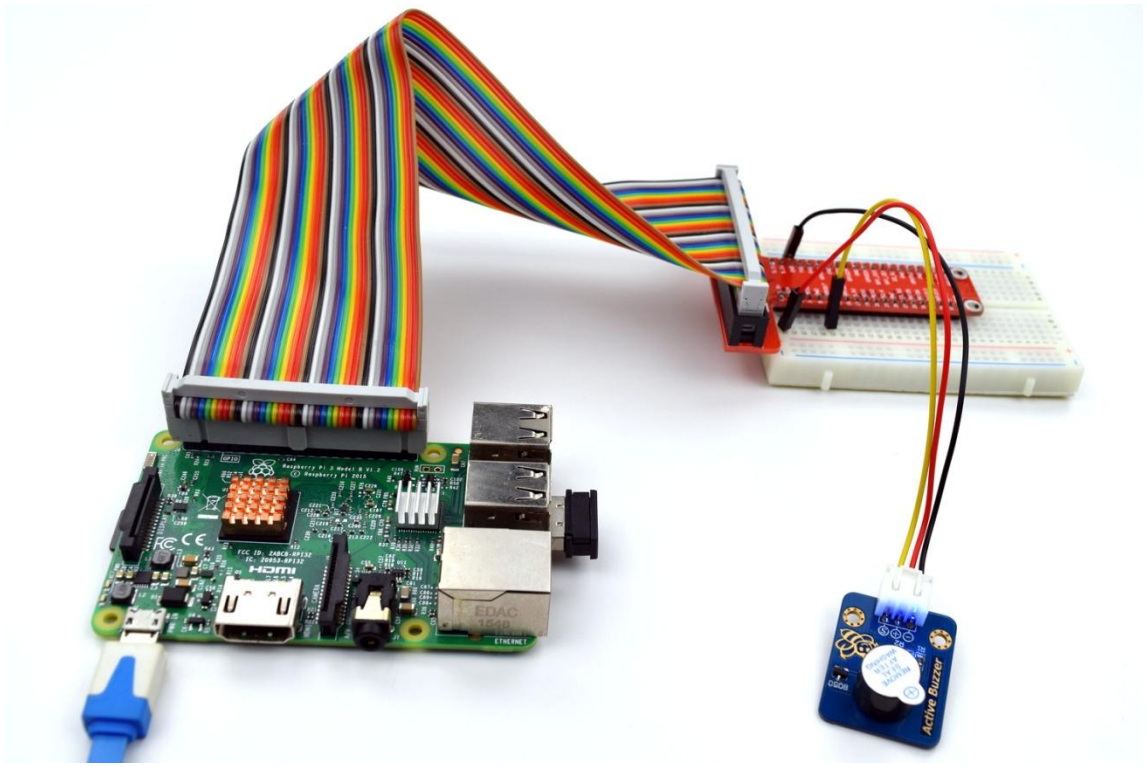
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/04_activeBuzzer.py)

Step 3: Run

```
$ sudo ./04_activeBuzzer.py
```

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



Lesson 5 Passive Buzzer

Introduction

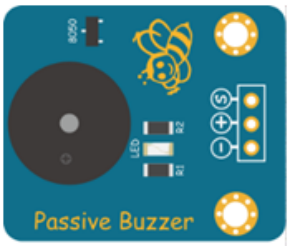
The difference between an active buzzer and a passive one radically lies in the requirement for input signals. The ideal signals for active buzzers are direct currents(DC), usually marked with VCC or VDD. Inside them there are a simple oscillation circuit that can convert constant direct currents into 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:

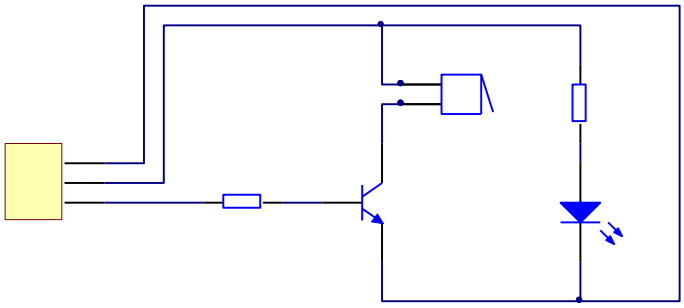


The Physical picture:

Pin definition:

S	Input
+	VCC
-	GND

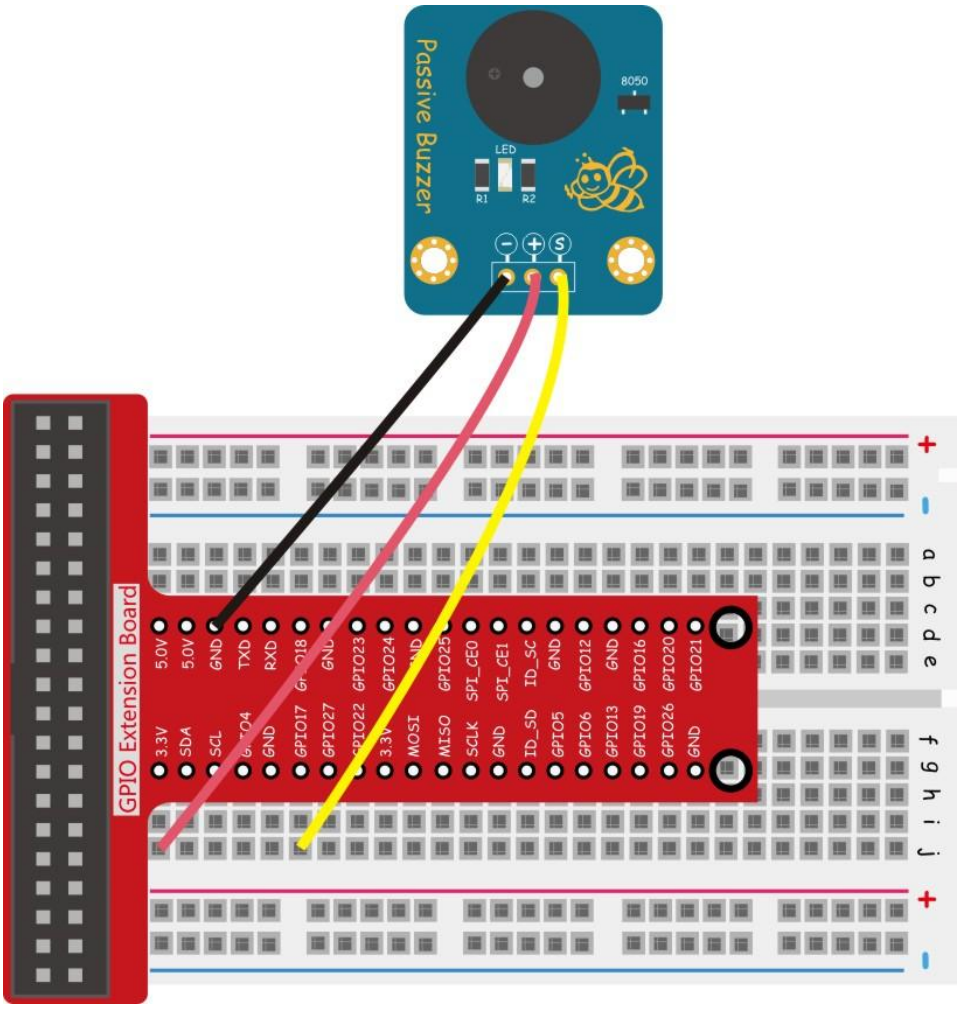
The schematic diagram:



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/Adeept_Sensor_Kit_for_RPi_C_Code/05_passiveBuzzer/passiveBuzzer.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./passiveBuzzer
```

For Python users:

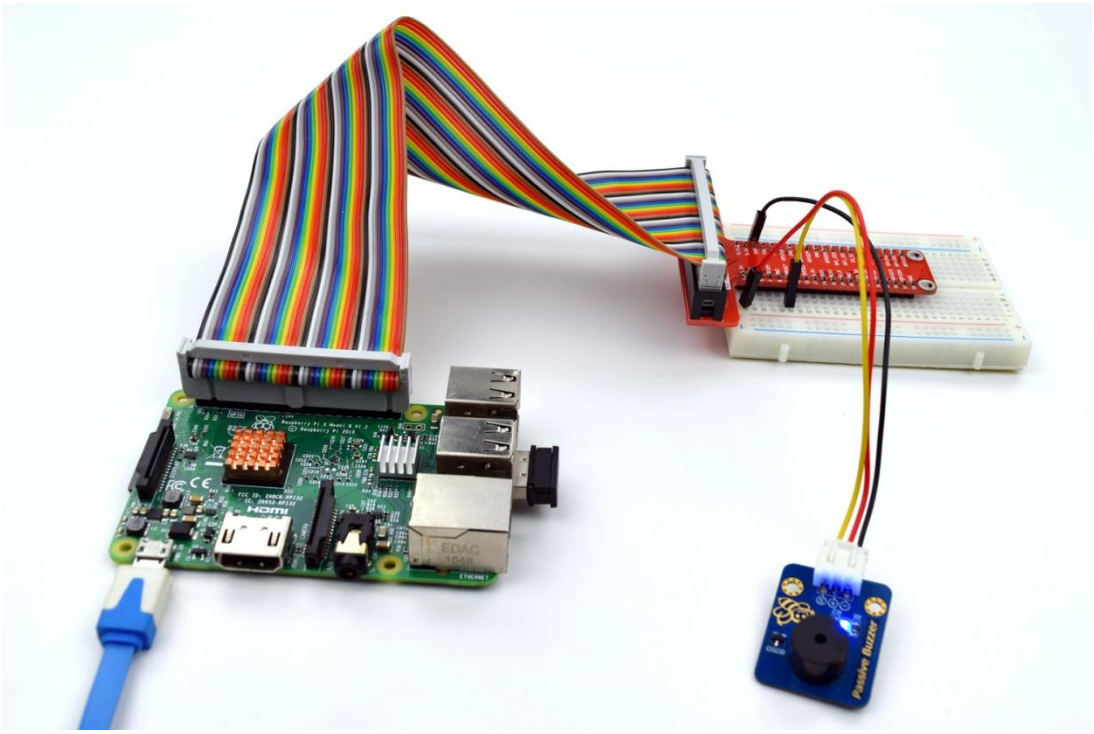
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/05_passiveBuzzer.py)

Step 3: Run

```
$ sudo ./05_passiveBuzzer.py
```

Now you can hear the passive buzzer play music.



Lesson 6 Controlling an LED by Hall Sensor

Introduction

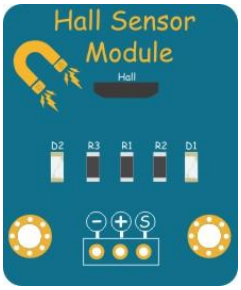
Hall 3144 is a Hall switch circuit. When the N pole of a magnet approaches to its print surface, the switch outputs Low; when the N pole moves away, the switch outputs High.

Components

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

Experimental Principle

The Fritzing image:



The Physical picture:

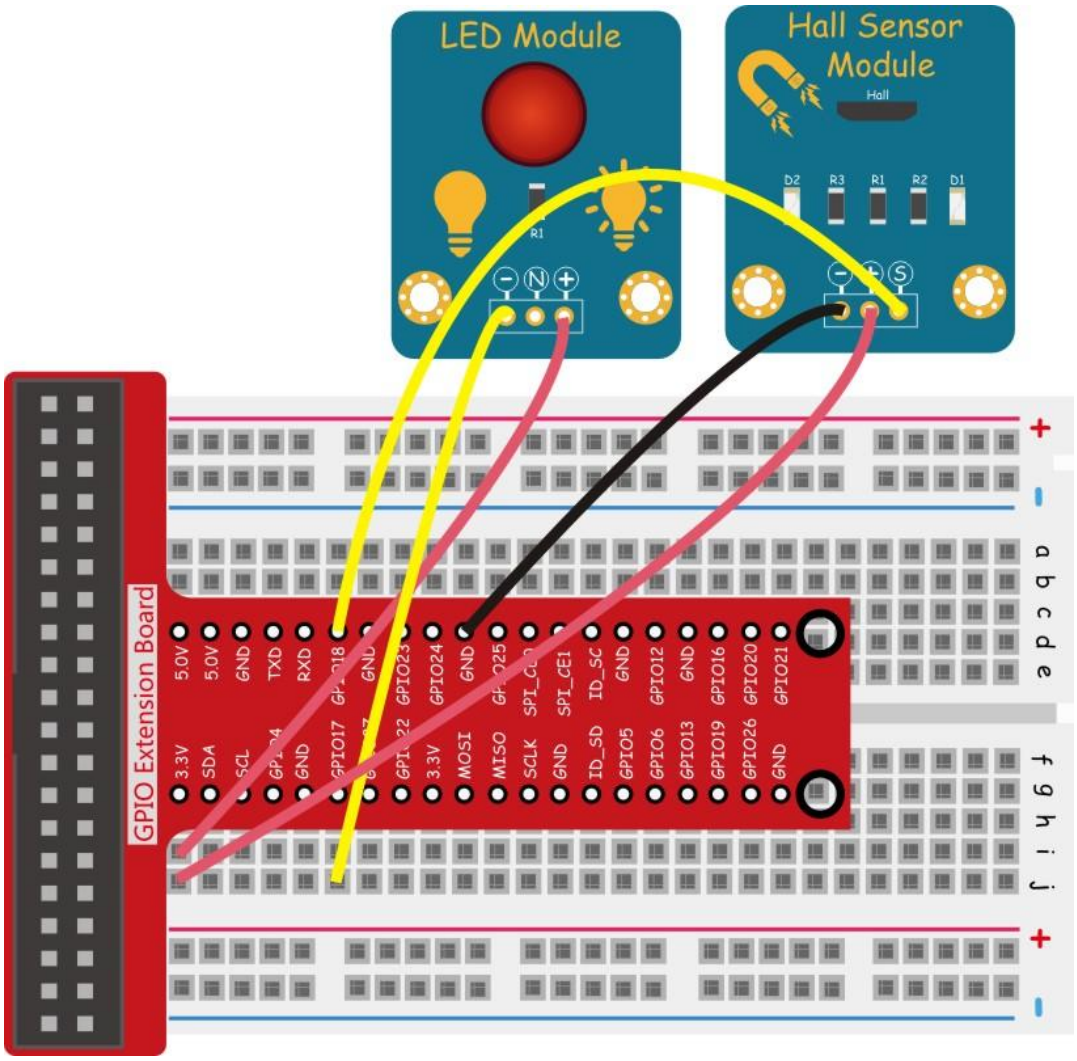
Pin definition:

S	Output
+	VCC
-	GND

In this experiment, by programming the Raspberry Pi, we detect the High or Low level of pin 11 of the Raspberry Pi and then toggle the LED based on the output signal of the hall sensor.

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_Sensor_Kit_for_RPi_C_Code/06_hall/hall.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./hall
```

For Python users:

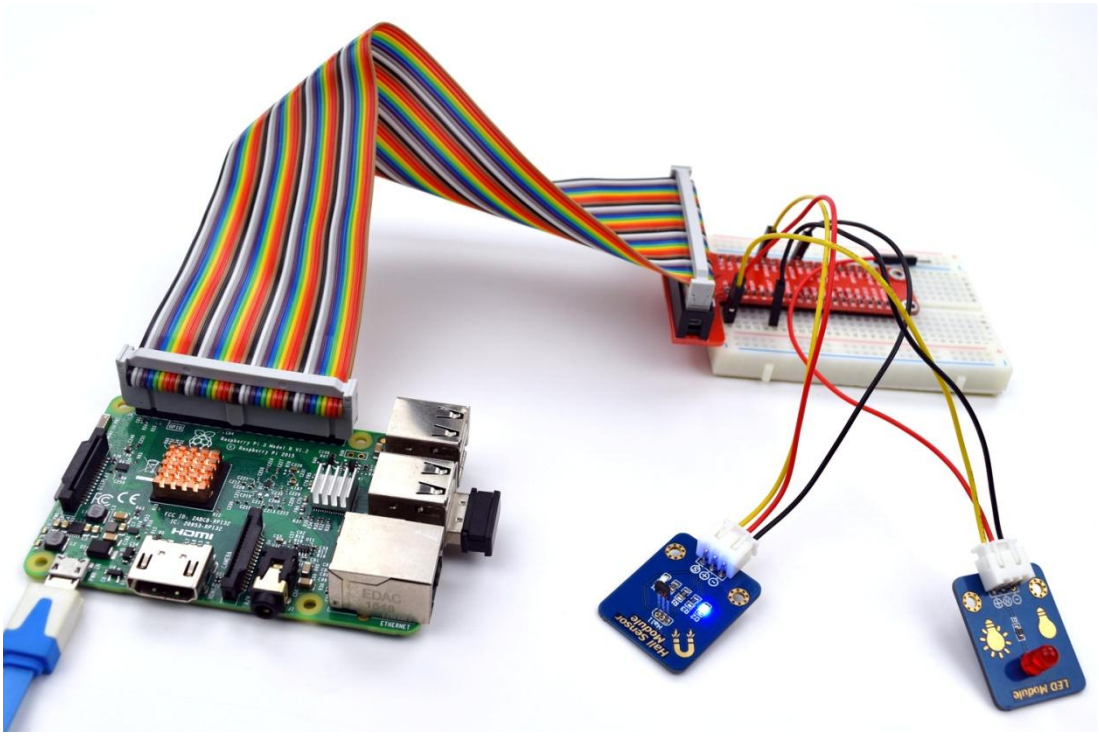
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/06_hall.py)

Step 3: Run

```
$ sudo ./06_hall.py
```

When you place the N pole of a magnet close to or move it away from the Hall Sensor, the state of the LED will be toggled between ON and OFF.



Lesson 7 Controlling an LED by Reed

Introduction

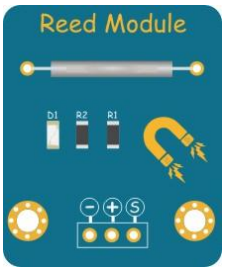
Reed switch is a special magnet-sensitive switch. Inside the glass tube, the reed sheets placed in parallel with a gap between compose the normally-open contact. When a magnet approaches the reed switch, or after the coil wrapped on the reed is electrified and a magnet field comes into existence thus magnetizing the reed, the contact of the reed will sense it and become the opposite pole. Due to the principle that different poles attract each other, when the magnetic force is larger than resistance of the reed, the open contacts (sheets) are closed; when the magnetic force decreases to a certain degree, the resistance takes charge again and the reed will back to the original state.

Components

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

Experimental Principle

The Fritzing image:



The Physical picture:

Pin definition:

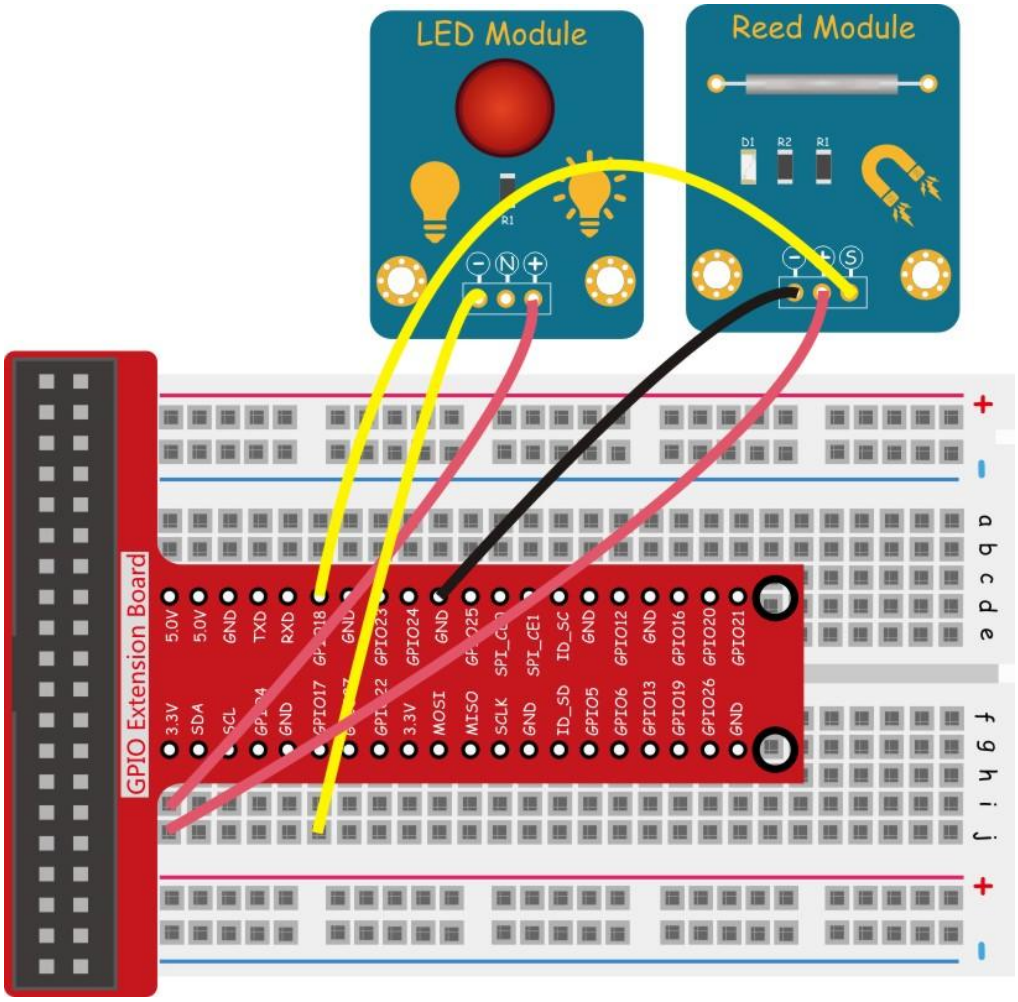
S	Output

+	VCC
-	GND

In this experiment, by programming the Raspberry Pi, we detect pin 12 of the Raspberry Pi and then toggle the LED based on the output signal of the reed switch.

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_Sensor_Kit_for_RPi_C_Code/07_Reed/reed.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./reed
```

For Python users:

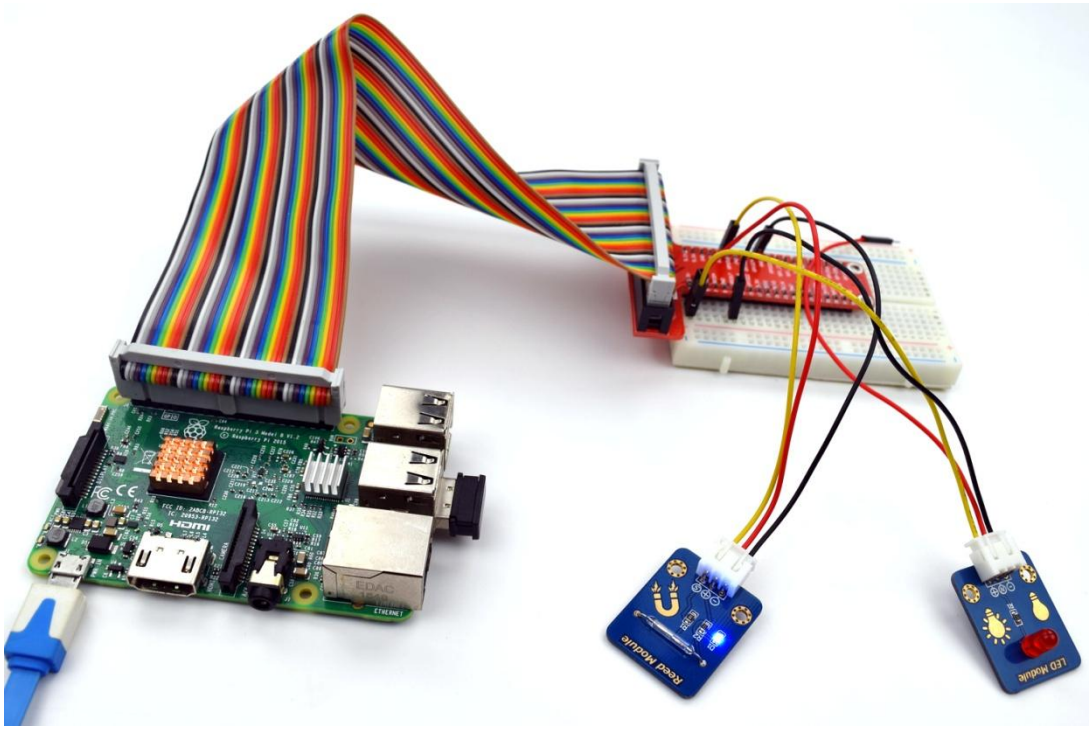
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/07_reed.py)

Step 3: Run

```
$ sudo ./07_reed.py
```

Place the magnet near or away from the Reed Module and you will see the state of the LED will be toggled between ON and OFF.



Lesson 8 How to Use a Relay

Introduction

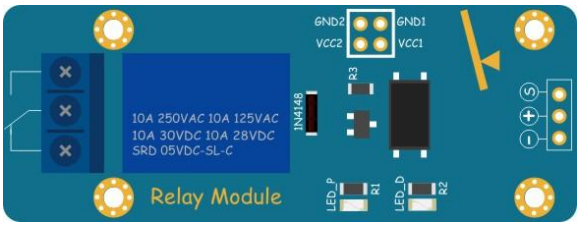
The relay is an electronic and electrical component that controls large currents by small currents. In the course of building a Raspberry Pi project, generally many large current or high volume devices like solenoid valve, lamp and motor cannot be connected directly to digital I/Os of the Raspberry Pi. At this moment, a relay can save your project.

Components

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

Experimental Principle

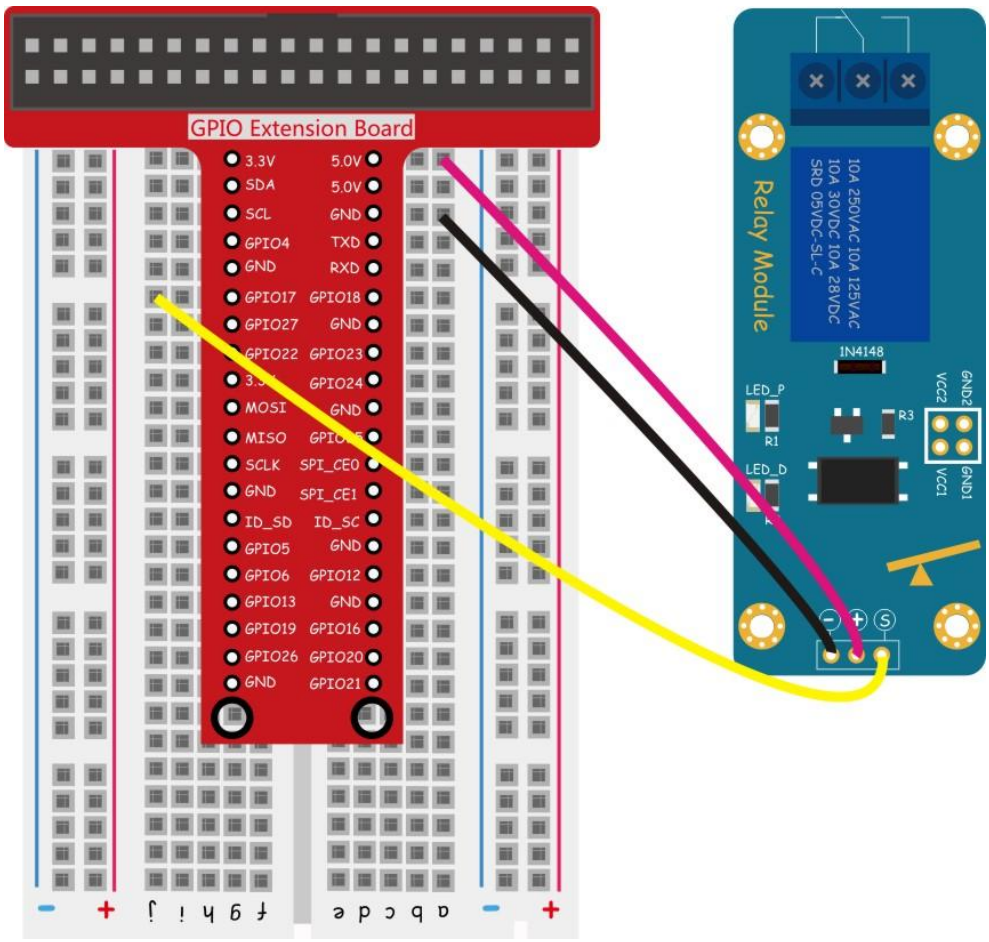
The Fritzing image:



The Physical picture:

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_Sensor_Kit_for_RPi_C_Code/08_relay/relay.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./relay
```

For Python users:

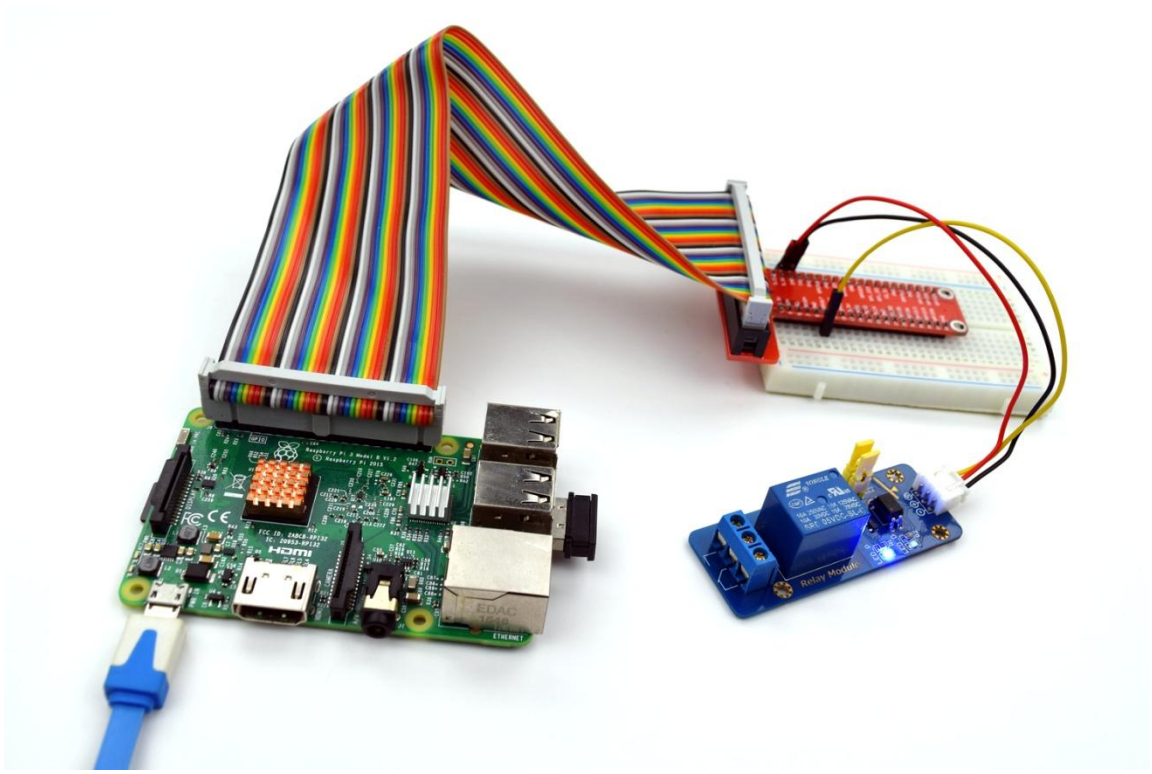
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/08_relay.py)

Step 3: Run

```
$ sudo ./08_relay.py
```

Now you can hear tick-tocks, which are the sounds of relay toggling.


Relay Module

Lesson 9 Laser Transmitter

Introduction

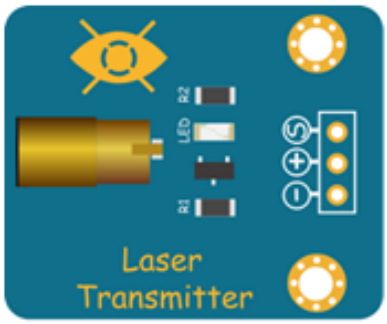
Semiconductor laser modules are widely used in laser communication, ranging, ladar, ignition and blasting, and testing instruments.

Components

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

Experimental Principle

The Fritzing image:

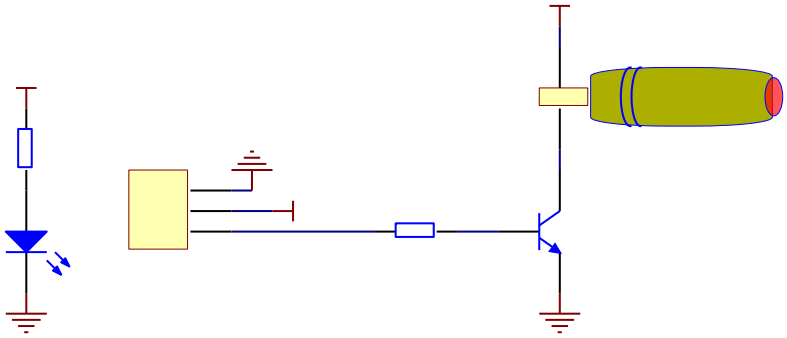


The Physical picture:

Pin definition:

S	Input
+	VCC
-	GND

The schematic diagram:

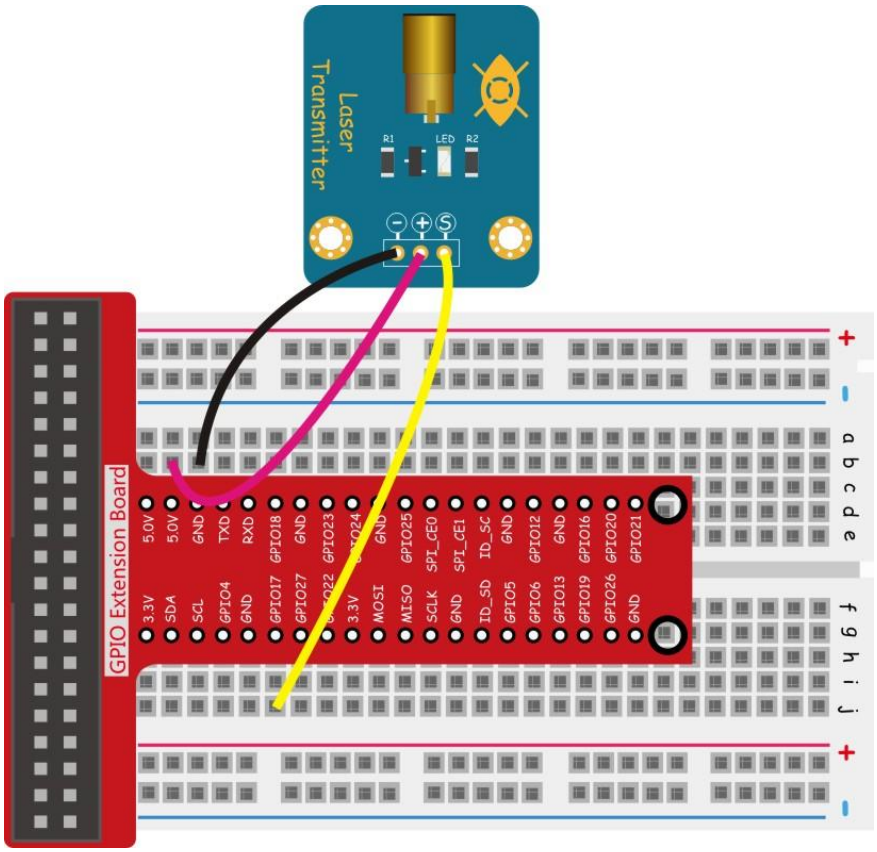


In this experiment, by programming the Raspberry Pi, we control the Laser Transmitter Module to emit laser by pin 11 of the Raspberry Pi.

Note: DO NOT look directly into the laser!

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_Sensor_Kit_for_RPi_C_Code/09_Laser/laser.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./laser
```

For Python users:

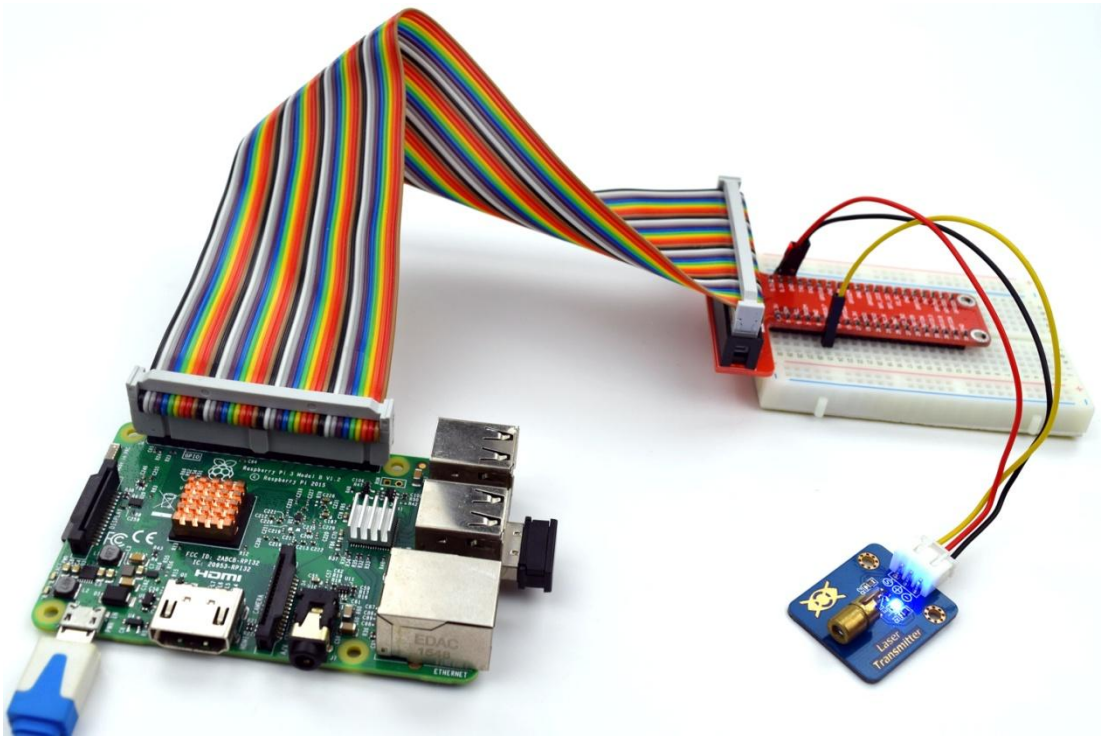
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/09_laser.py)

Step 3: Run

```
$ sudo ./09_laser.py
```

Now you can see the Laser Transmitter Module emit laser and the emission lasts for 1 seconds, and then it stops. After 1s, the cycle repeats.



Lesson 10 Laser Receiver

Introduction

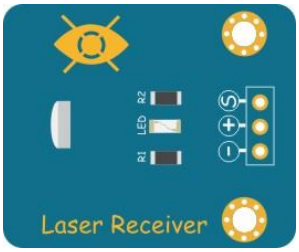
The principle for many laser receiving devices is the same. The laser ray goes through the optical lens and then is received by the photosensitive device, i.e. the photodiode. After receiving the rays, the photodiode will generate currents accordingly (based on the light intensity) which then output electrical signal after running through the amplifier. Then use an I/O port of the Raspberry Pi to detect the output terminal of the Laser Receiver module, and thus we can tell whether there are rays shone on the module.

Components

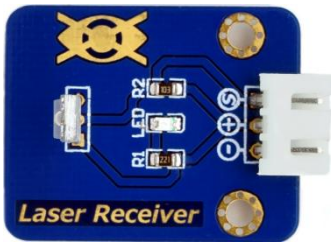
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Laser Transmitter Module
- 1 * Laser Receiver Module
- 2 * 3-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



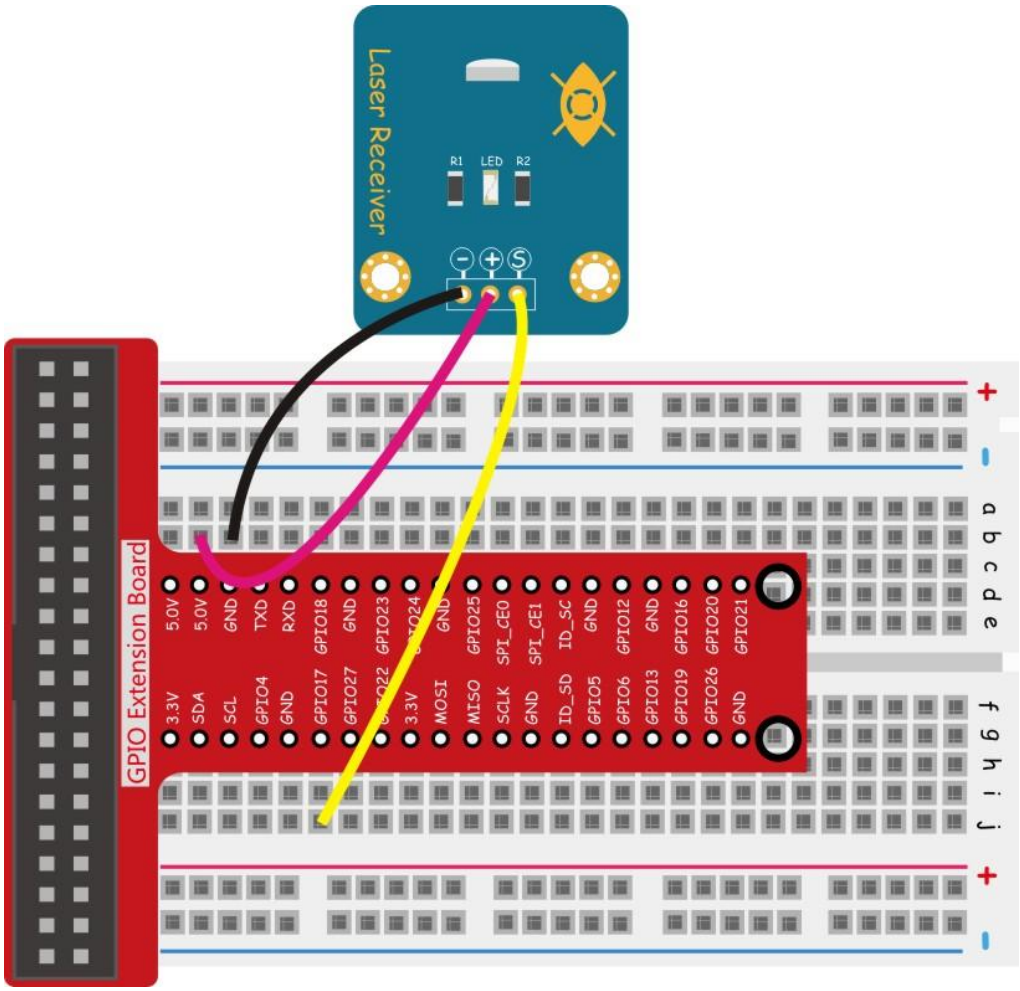
Pin definition:

S	Output
+	VCC
-	GND

In this experiment, we use the Laser Receiver module to detect whether there is laser ray shining on the module. If yes, the output pin (S) of the module will output Low.

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_Sensor_Kit_for_RPi_C_Code/10_LaserRecv/laserRecv.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./laserRecv
```

For Python users:

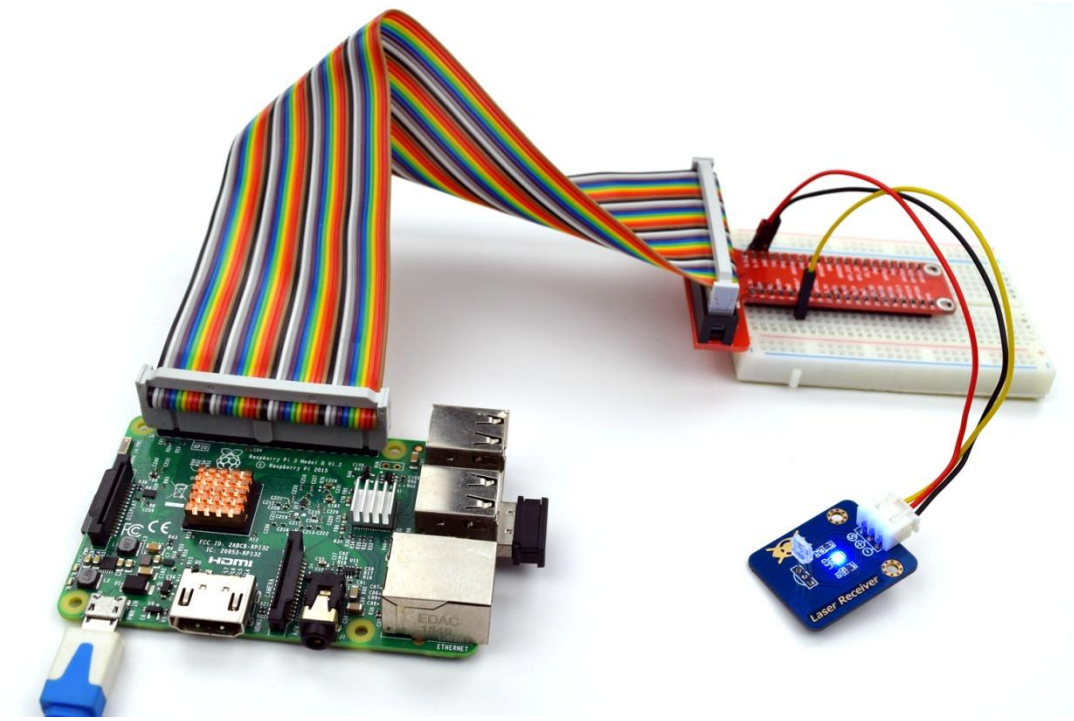
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/10_laserRecv.py)

Step 3: Run

```
$ sudo ./10_laserRecv.py
```

Make the Laser Transmitter module to shoot laser ray onto the Laser Receiver module. Then "Laser received…" will be displayed on the terminal. Remove the transmitter module and "Laser not received" will be shown.



Lesson 11 How to Control a DC Motor

Introduction

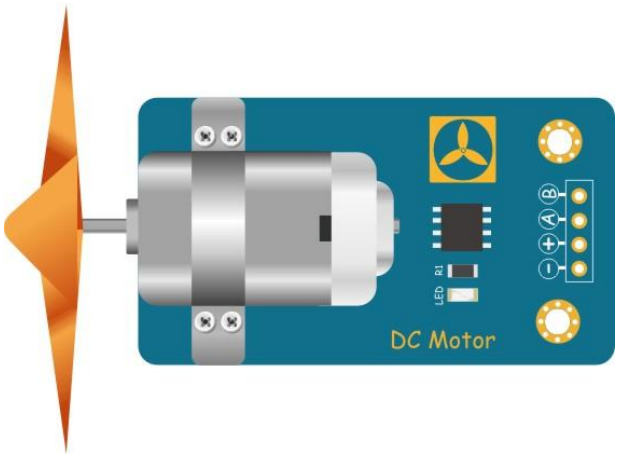
DC motor is a device that converts electrical energy into mechanical energy. Due to the ease of control, it is usually used in fan, electronic toy, shaver, etc.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * DC Motor Module
- 1 * 4-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:

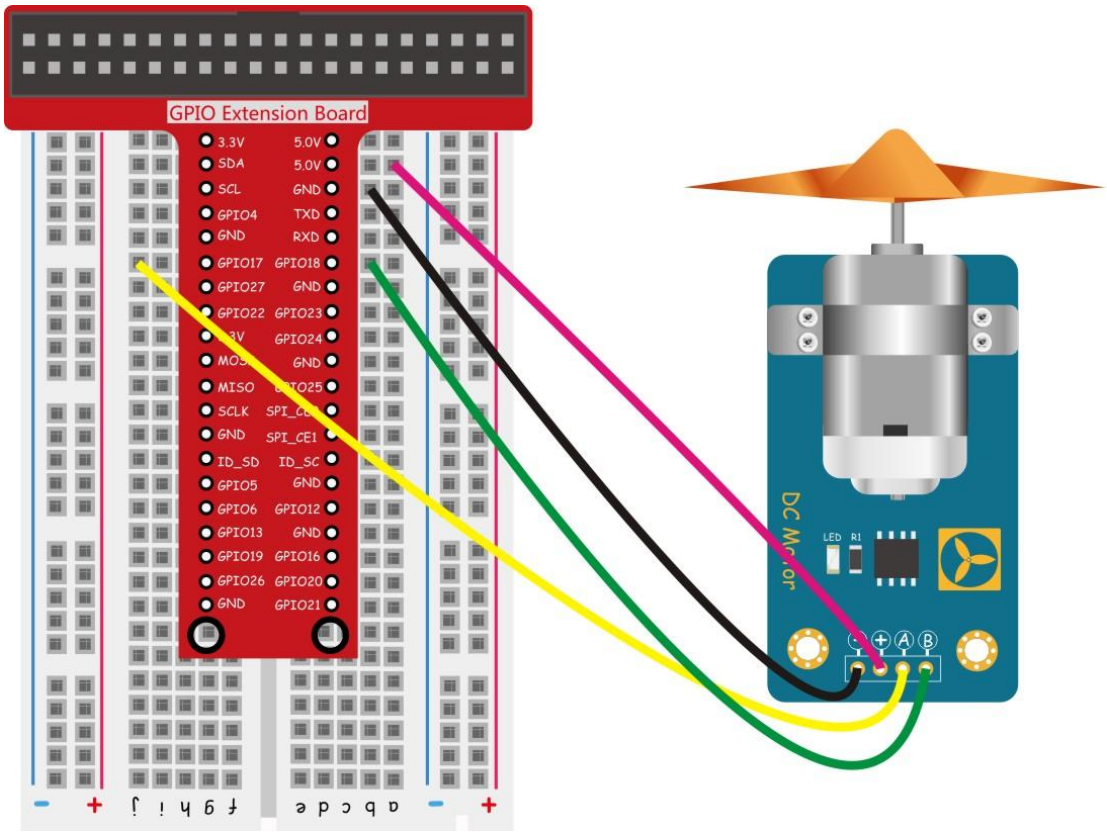
Pin definition:

B	Iutput
A	Iutput
+	VCC
-	GND

This experiment is to control the status of the DC motor via the Raspberry Pi. The statuses include forward, stop and reverse.

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_Sensor_Kit_for_RPi_C_Code/11_motor/motor.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./motor
```

For Python users:

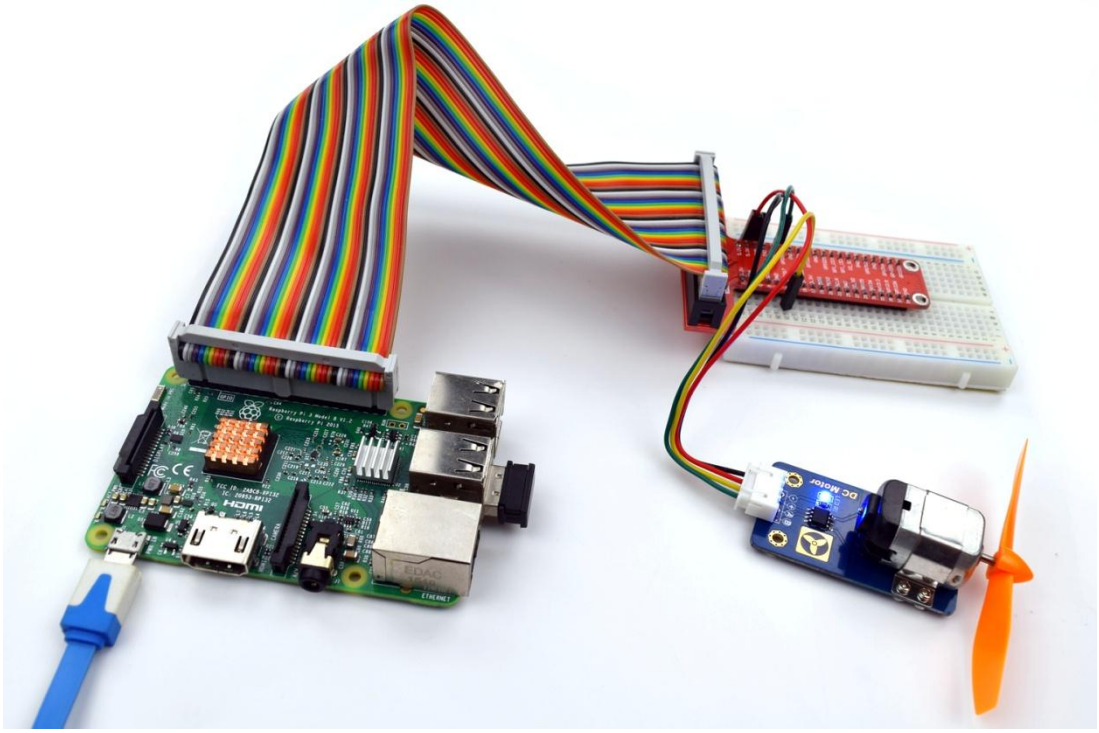
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/11_motor.py)

Step 3: Run

```
$ sudo ./11_motor.py
```

Now you can see the fan rotates forward, stop, reverse.



Lesson 12 Controlling an LED by Limit Switch

Introduction

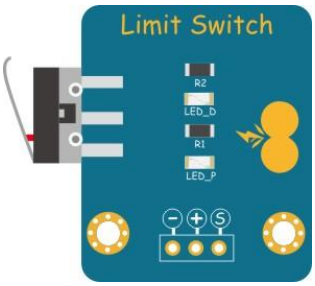
Limit Switch, or travel switch, can be installed on relatively stationary objects such as mounting bracket and door frame, or moving objects like car and door. When the moving object approaches the stationary one, the switch is closed; when the moving one moves away from the static one, the switch is open.

Components

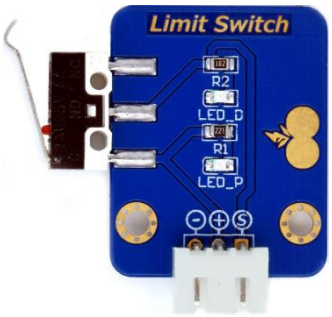
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Limit Switch Module
- 1 * LED Module
- 2 * 3-Pin Wires

Experimental Principle

The Fritzing image:



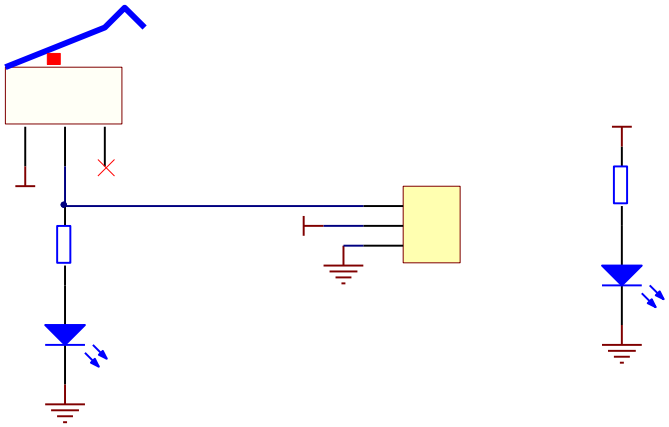
The Physical picture:



Pin definition:

S	Output
+	VCC
-	GND

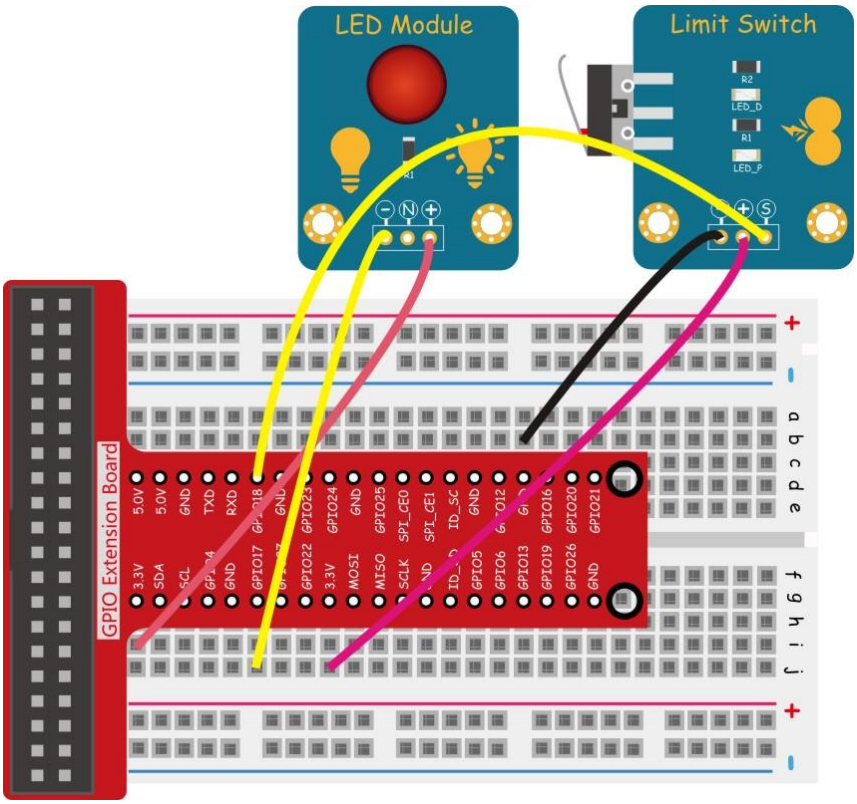
The schematic diagram:



In this experiment, by programming the Raspberry Pi, we detect the status of the Limit Switch module through pin 12 of the Raspberry Pi and then toggle the LED based on the output signal of the limit switch.

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_Sensor_Kit_for_RPi_C_Code/12_limitSwitch/limitSwitch.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./limitSwitch
```

For Python users:

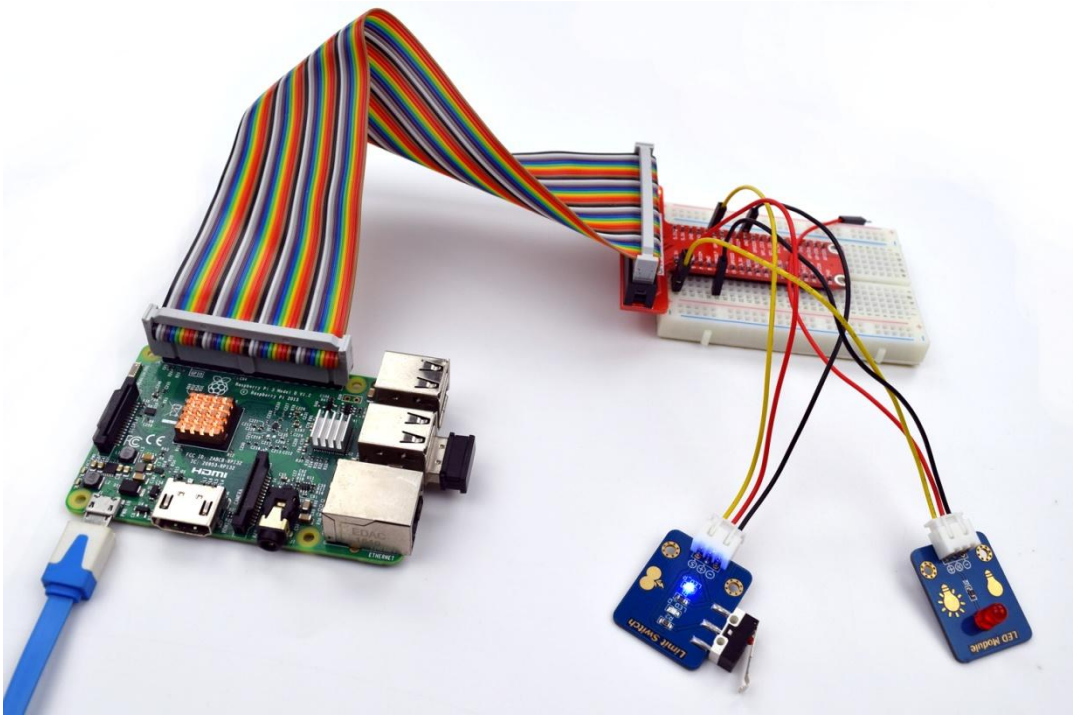
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/12_limitSwitch.py)

Step 3: Run

```
$ sudo ./12_limitSwitch.py
```

Press the Limit Switch and you can see that the state of the LED will be toggled between ON and OFF.



Lesson 13 Controlling an LED by Vibration Switch

Introduction

The vibration digital input module can sense weak vibration signals, thus it can be used for related interaction projects. The core sensor is SW-540, a spring component of no-directional vibration sensing which can be triggered at any angle. The module stays off at any angle when it's still. But when it's hit or knocked by external force, the distorted spring contacts and connects the electrode in the middle thus connecting the two pins and turning the sensor on. When the force disappears, the circuit is back to off.

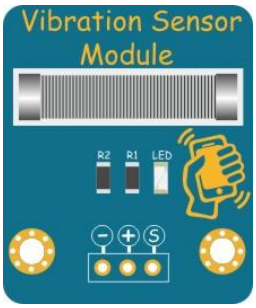
This sensor module is suitable for small-current vibration detection circuits and has been widely used in products such as toys, light shoes, burglar alarms, electronic scales, flash dance shoes, hot wheels, flash balls, etc.

Components

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

Experimental Principle

The Fritzing image:

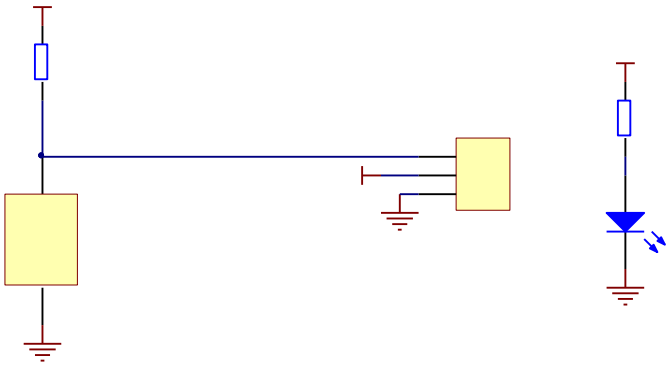


The Physical picture:

Pin definition:

S	Output
+	VCC
-	GND

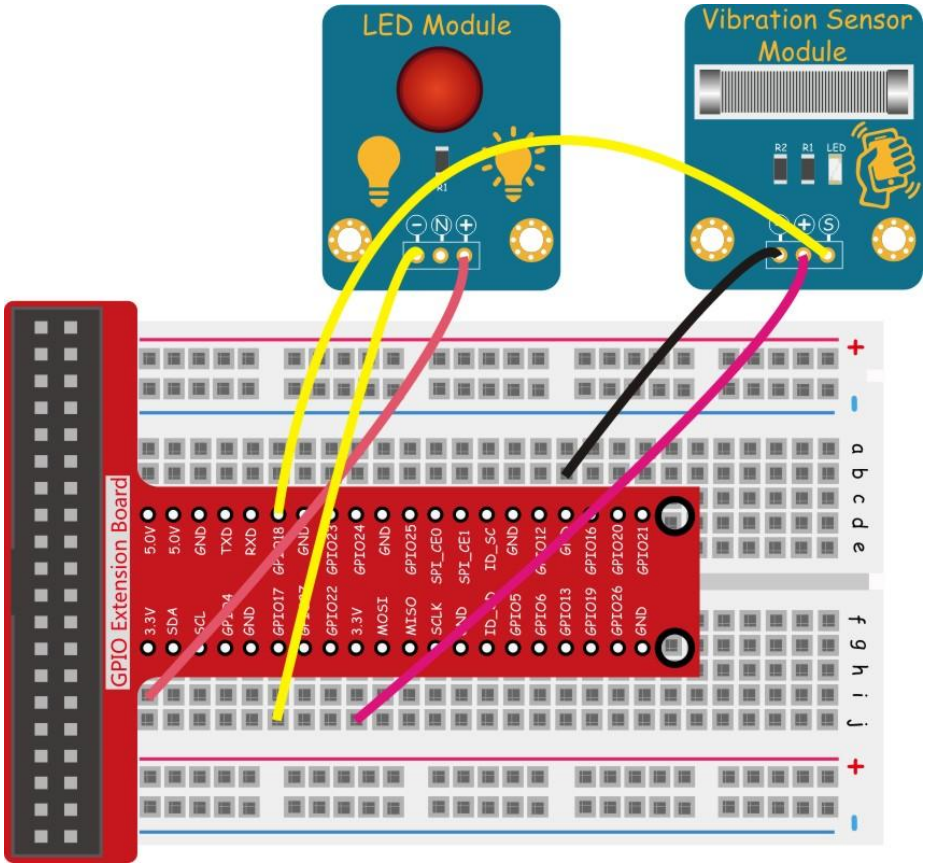
The schematic diagram:



This experiment is to make the LED that connected to the pin 11 of Raspberry Pi flicker with the actions of the Vibration Sensor 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_Sensor_Kit_for_RPi_C_Code/13_vibration/vibration.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./vibration
```

For Python users:

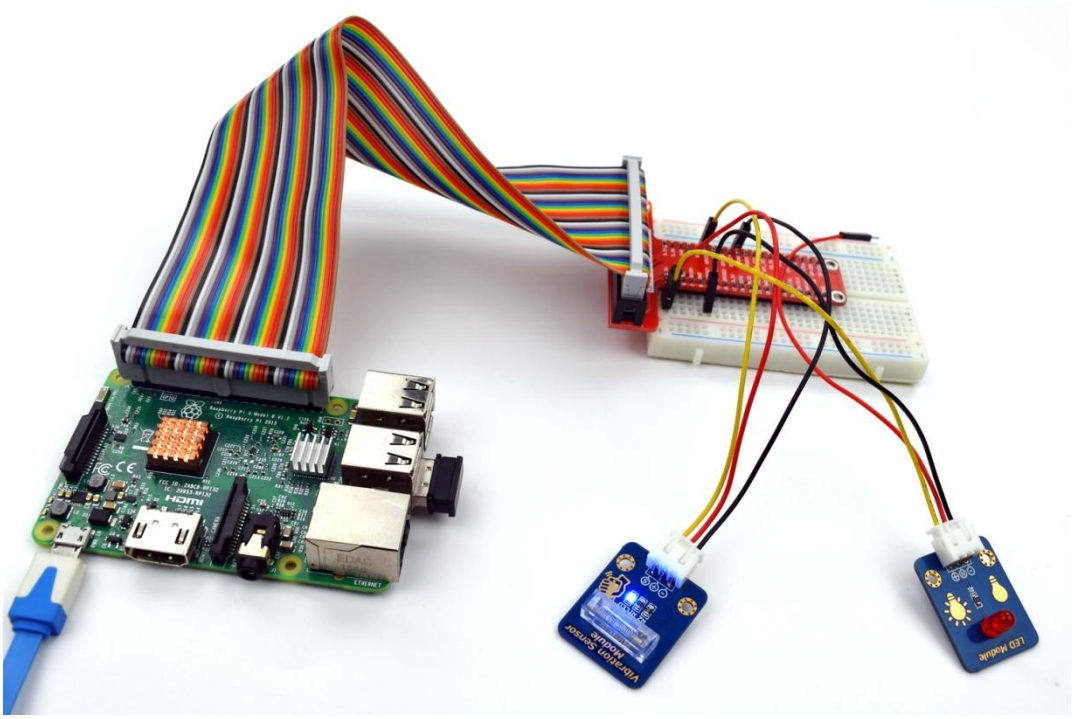
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/13_vibration.py)

Step 3: Run

```
$ sudo ./13_vibration.py
```

Knock or tap the Vibration Sensor module and you'll see the status of LED will be toggled.


Vibration Sensor Module
LED module

Lesson 14 Rotary Encoder

Introduction

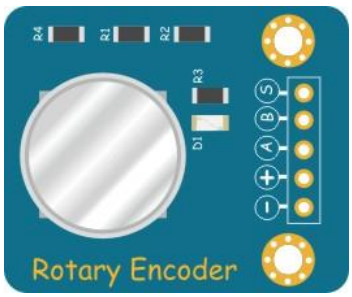
Rotary encoder switch, or small rotary encoder, is a switch electronic component that has a set of regular and strictly-sequenced pulses. The module supports functions such as increase, decrease, turn pages, etc., by collaboration with a microcontroller. For example, in daily life you can see page turning 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:



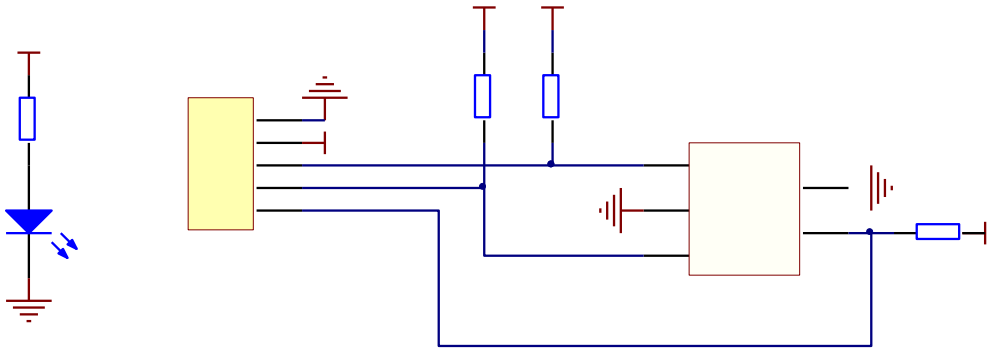
The Physical picture:

Pin definition:

S	Output
B	Output
A	Output

+	VCC
-	GND

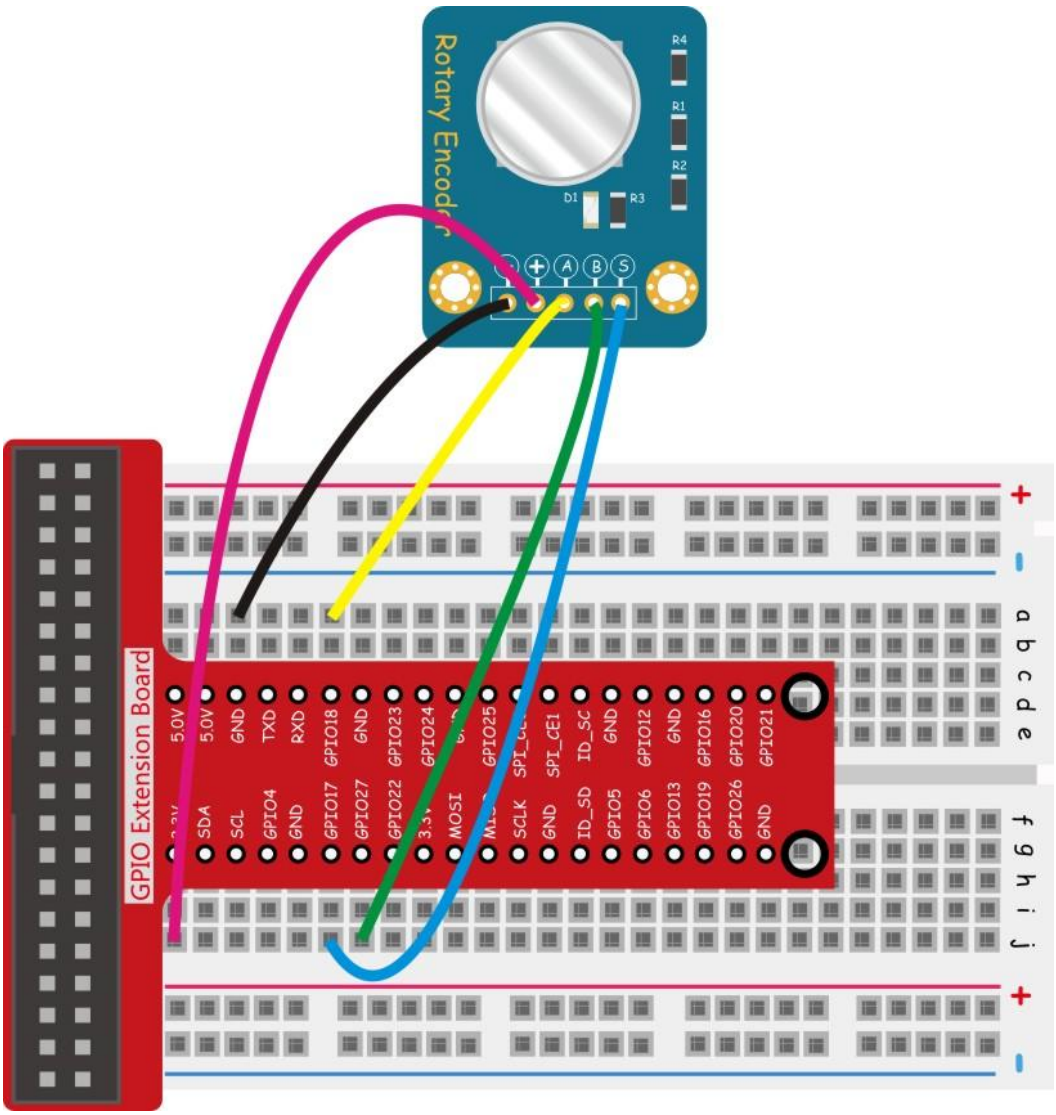
The schematic diagram:



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/Adeept_Sensor_Kit_for_RPi_C_Code/14_RotaryEncoder/rotaryEncoder.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./rotaryEncoder
```

For Python users:

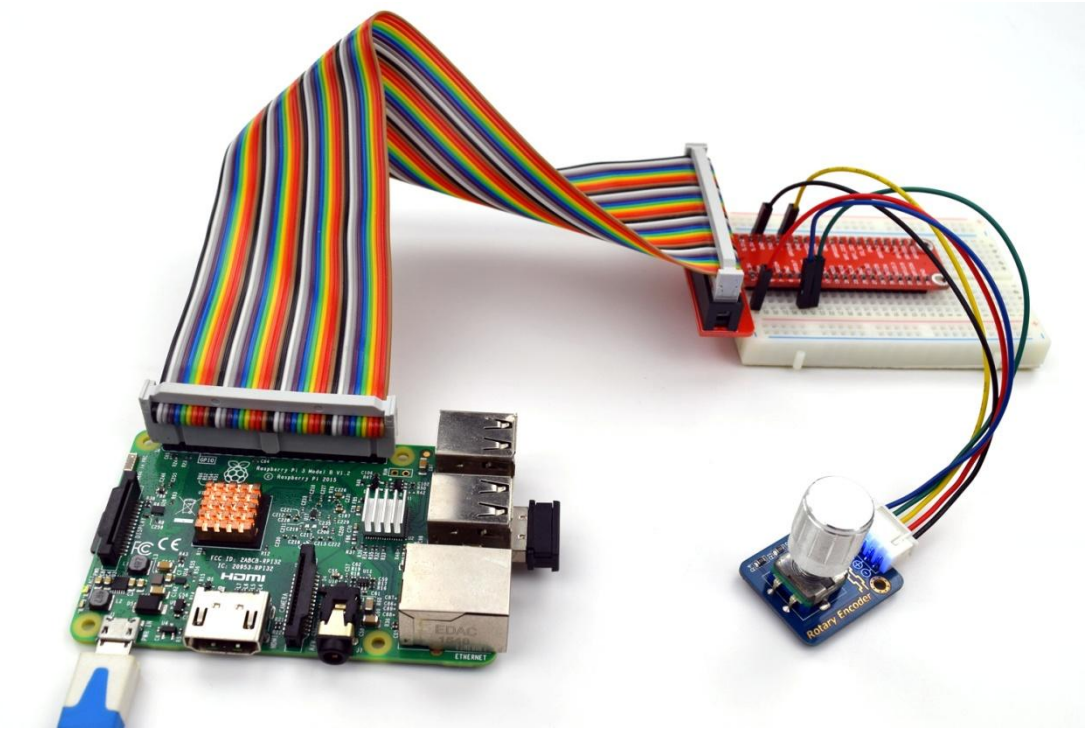
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/14_rotaryEncoder.py)

Step 3: Run

```
$ sudo ./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 15 Controlling an LED by Touch Button

Introduction

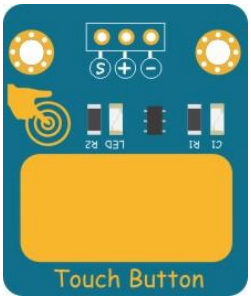
The Touch Button Module is a touch switch module developed based on the principle of capacitive sensing. Touch of human or metal onto the gilded touch surface can be sensed. Besides, it can also detect other such touch with certain materials like plastic and glass between. The sensitivity in these cases 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:



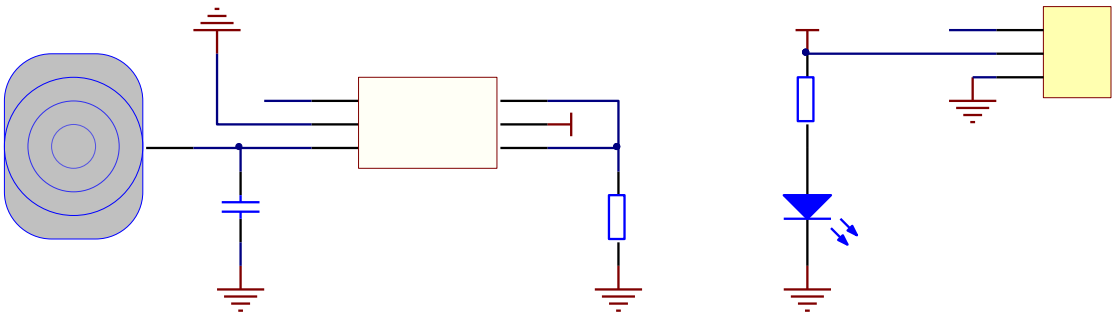
The Physical picture:

Pin definition:

S	Output

+	VCC
-	GND

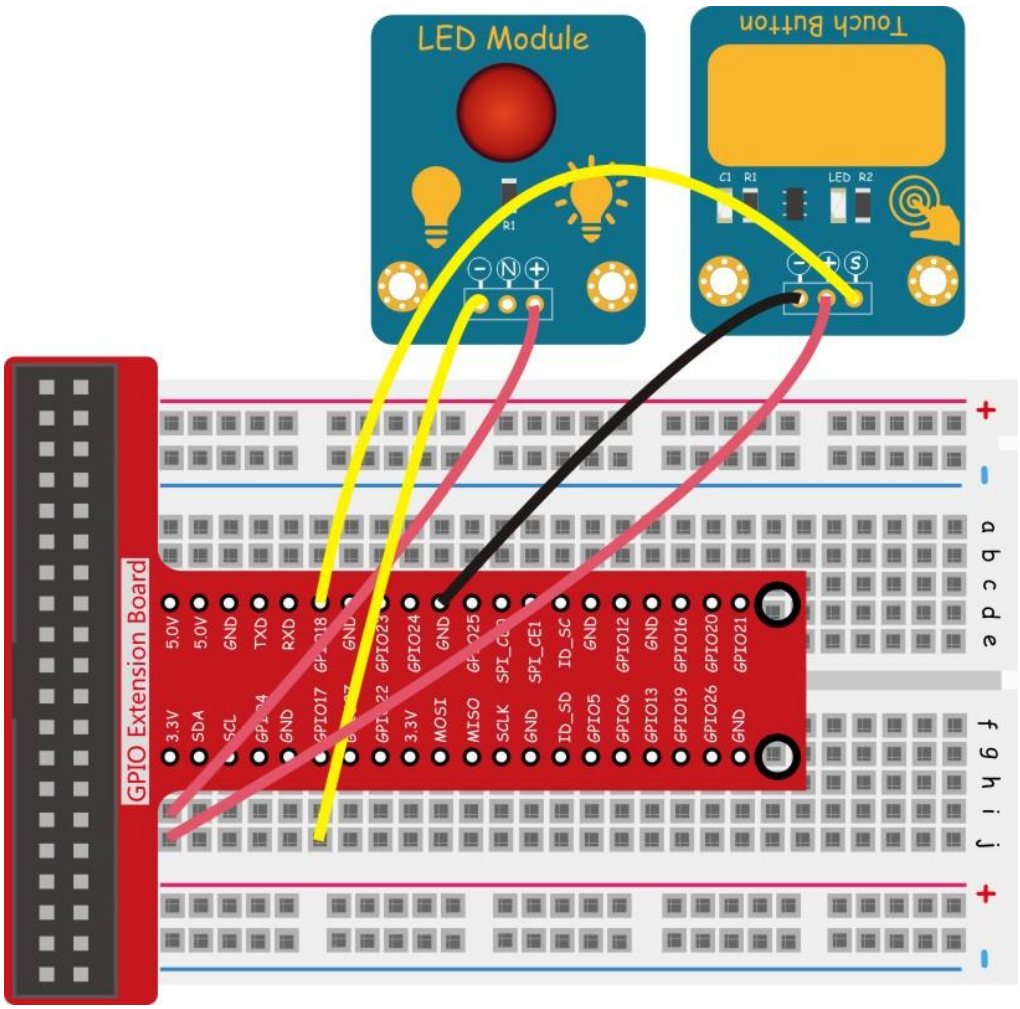
The schematic diagram:



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/Adeept_Sensor_Kit_for_RPi_C_Code/15_touchBtn/touchBtn.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./touchBtn
```

For Python users:

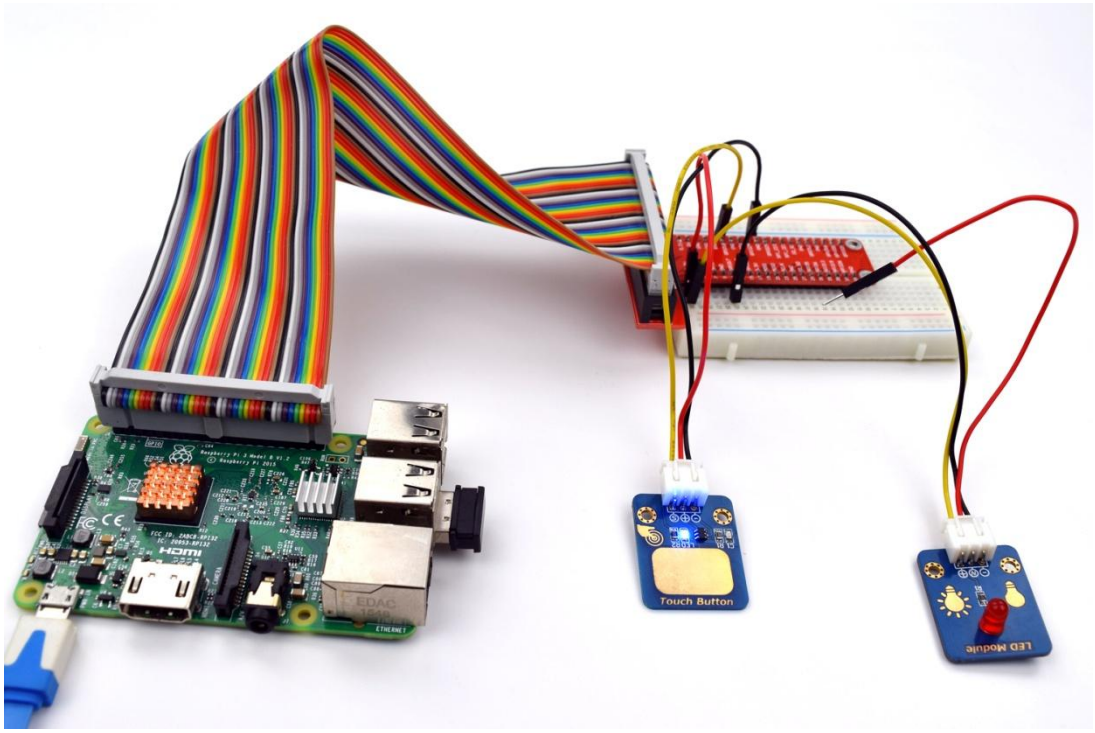
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/15_touchBtn.py)

Step 3: Run

```
$ sudo python 15_touchBtn.py
```

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


Touch Button
LED Module

Lesson 16 Movement Detection Based on PIR

Introduction

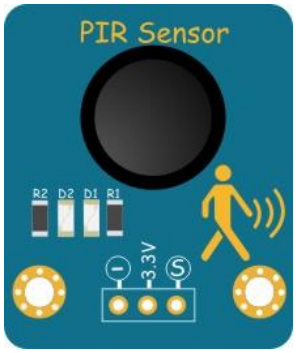
The PIR (passive Infrared) sensor can detect the Infrared rays emitted by human or animals and then output On/Off signals. Traditional pyroelectric PIR sensor needs pyroelectric Infrared probe, special chip and complex circuits to make the effect. In this case, the sensor is a large one with complicated circuits and comparatively less credible. But this Infrared pyroelectric motion sensor adopts the digital integration of the probe, so it boasts high credibility, low power consumption and simple outside circuit with a small size. It can be applied to any cases in detecting moving human or animals.

Components

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

Experimental Principle

The Fritzing image:

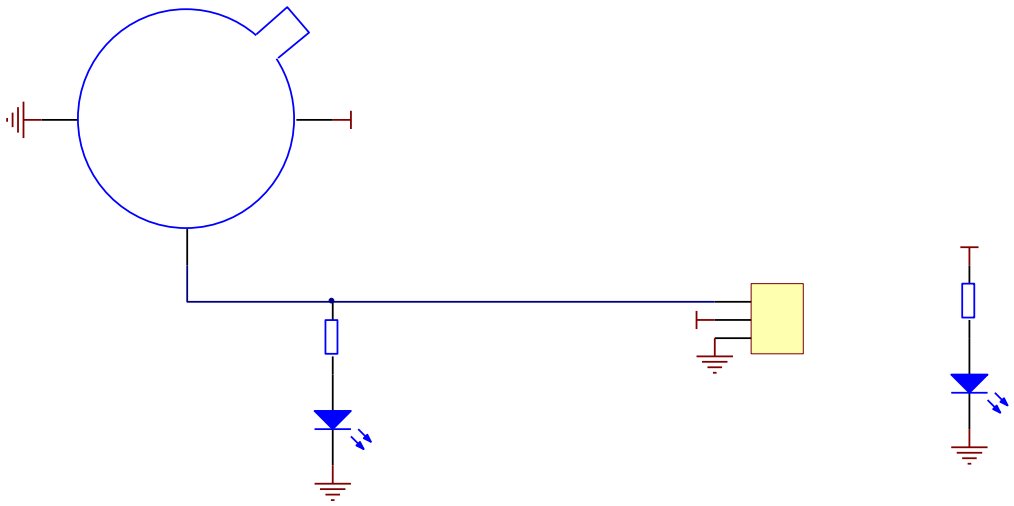


The Physical picture:

Pin definition:

S	Output
3.3V	3.3V
-	GND

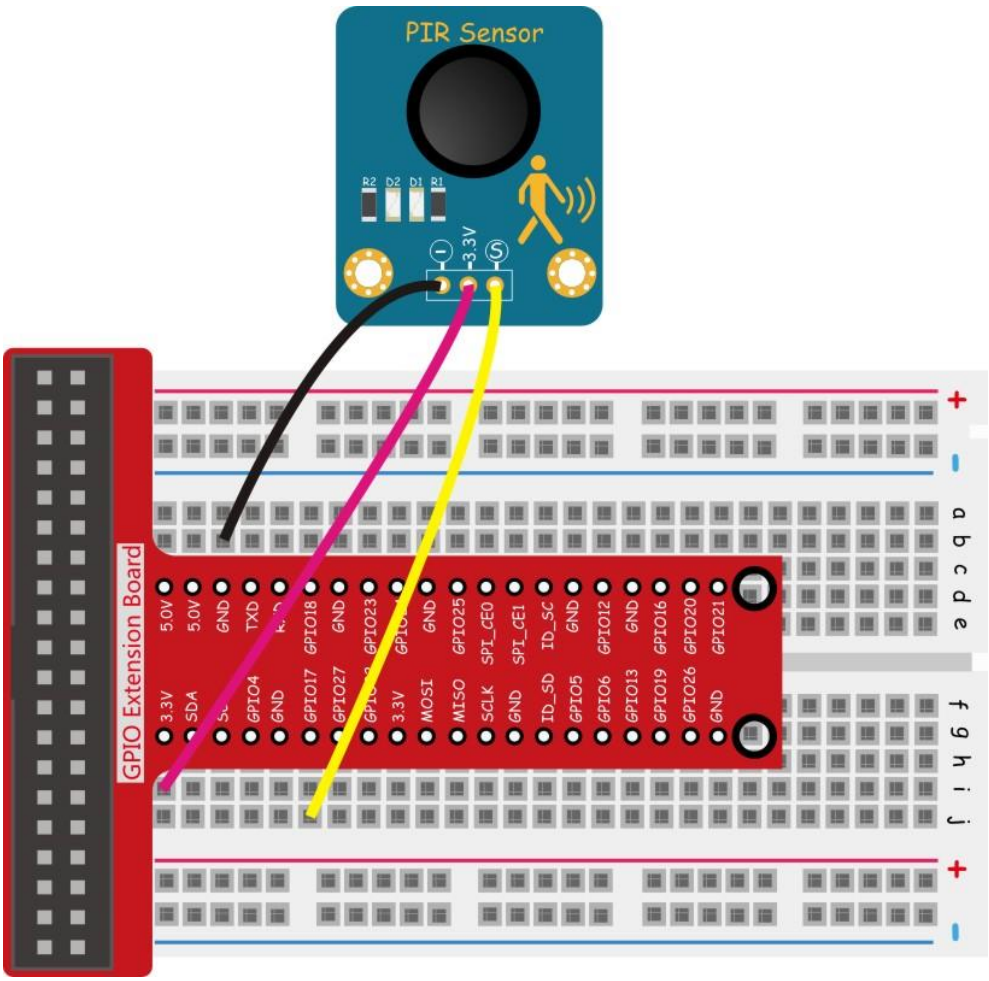
The schematic diagram:



This experiment is to use the PIR Sensor Module to detect whether there is any human activity.

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_Sensor_Kit_for_RPi_C_Code/16_PIR/pir.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./pir
```

For Python users:

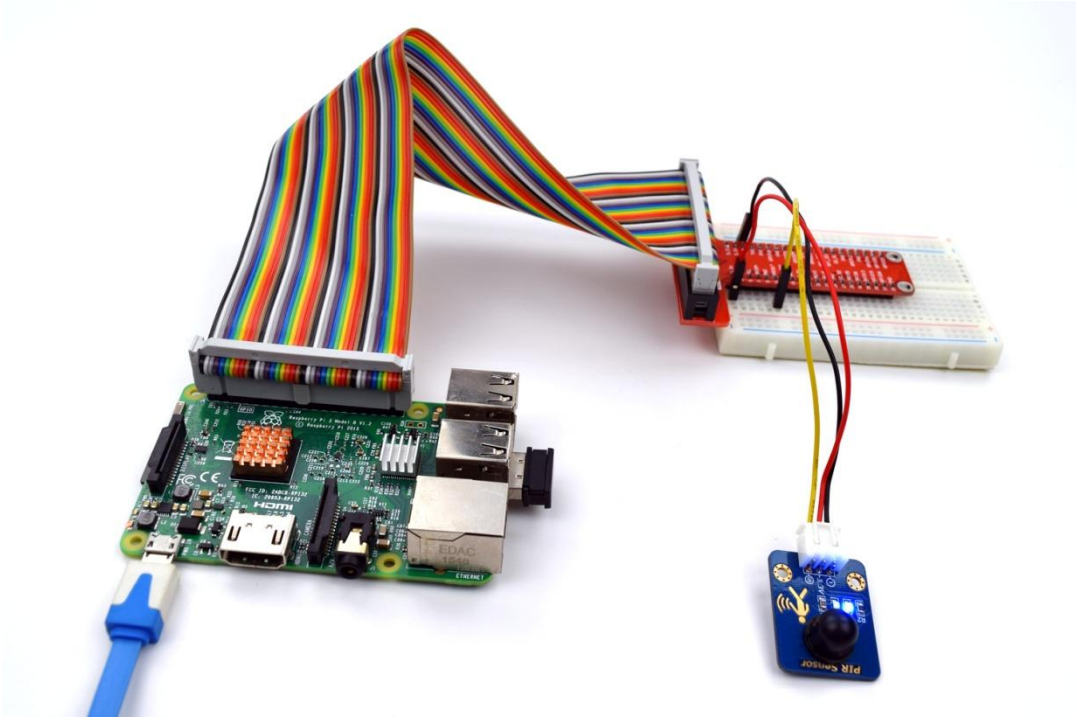
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/16_PIR.py)

Step 3: Run

```
$ sudo python 16_PIR.py
```

When the PIR module detects human movement, " Someone invasion !!!" will be displayed on the terminal.


PIR Sensor

Lesson 17 Flame Sensor

Introduction

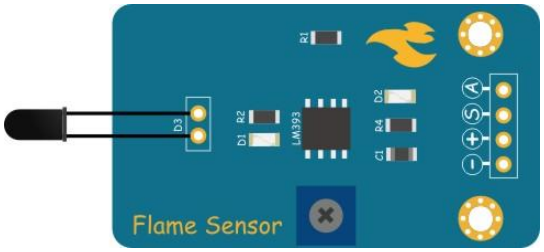
The Flame Sensor detects flames by the special infrared receiver to capture the infrared rays of a specific wavelength in the flames. It supports a detection angle of as high as 60 degrees and works within -25 - 85℃. When in use, you need to pay attention and do not place the probe of the sensor too close to the flames in case of damages. Besides, the sensor can be used to detect light intensity. It can detect a light source with a wavelength of 760 - 1100nm. Pin A outputs the analog data collected by the sensor. You can adjust the potentiometer on the module to set the alarm threshold of the sensor. So when the value reaches the threshold, pin S will switch between High and Low.

Components

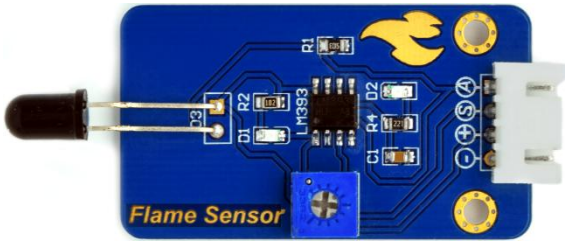
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Flame Sensor Module
- 1 * 4-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



Pin definition:

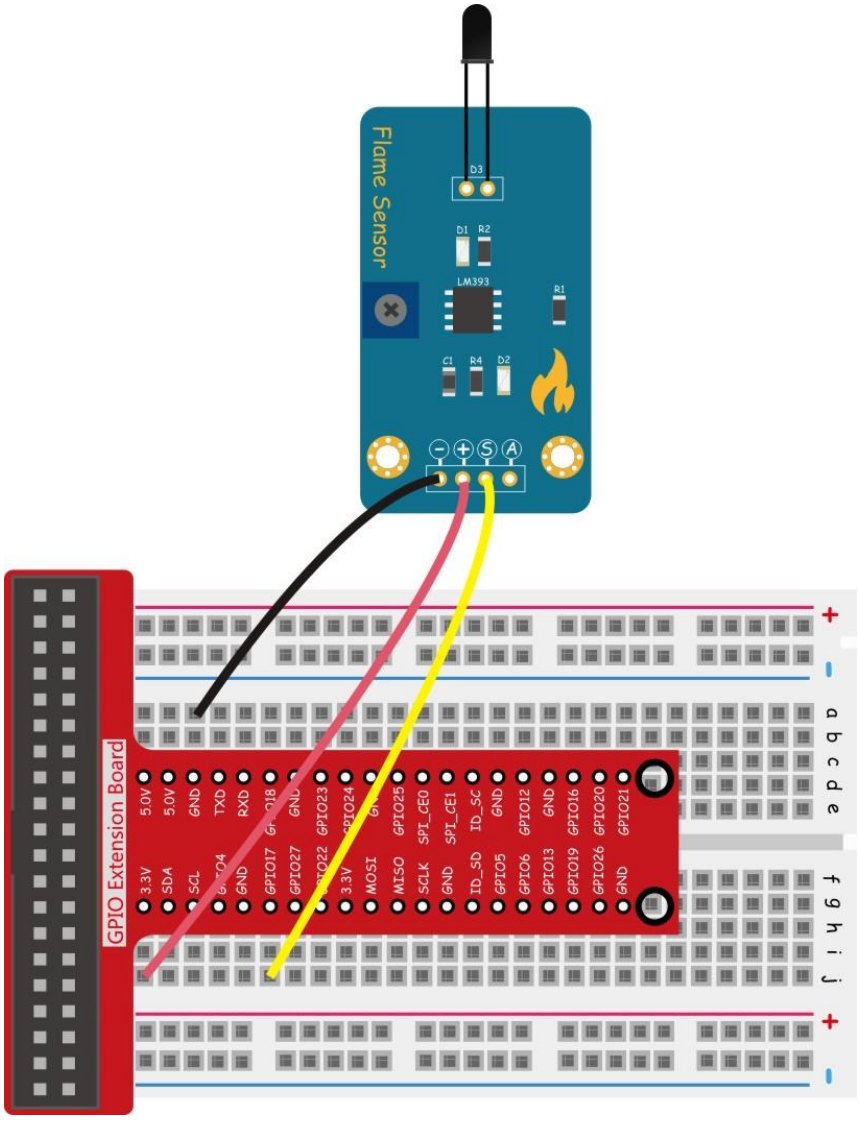
S	Digital Output
A	Analog Output
+	VCC

-	GND

In this experiment, by programming the Raspberry Pi, we detects whether a flame has been encountered.

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_Sensor_Kit_for_RPi_C_Code/17_flame/flame.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./flame
```

For Python users:

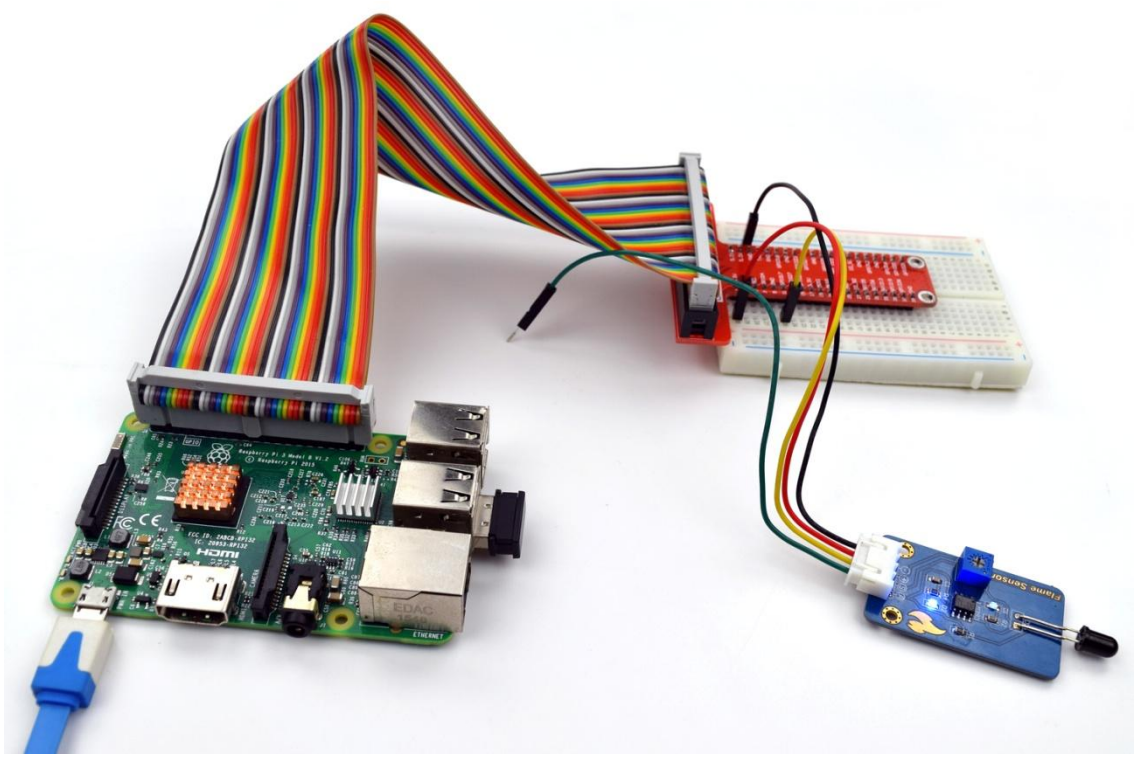
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/17_flame.py)

Step 3: Run

```
$ sudo python 17_flame.py
```

Then you'll see the data detected by the Flame Sensor module on the terminal. The data includes two parts: " Fire detected or not !!! " and "Flame intensity".



Lesson 18 Line Finder

Introduction

The Line Finder Module applies the principle that infrared rays reflect differently on surfaces of different colors. After electrified, the infrared diode on the module sends out infrared rays constantly. When they encounter a white surface, the diffused reflection happens and the reflected rays are received by the receiver on the module. On the other hand, when they come across a black one, the receiver cannot get any infrared.

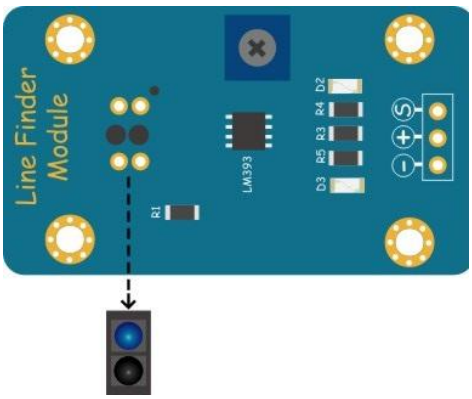
Thus, the processor can tell whether it is a white or black detected surface by receiving the reflected infrared rays or not. Based on this, the module is usually used in line finding on a smart car.

Components

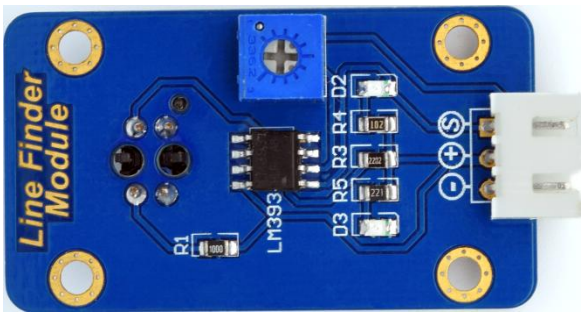
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Line Finder Module
- 1 * 3-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



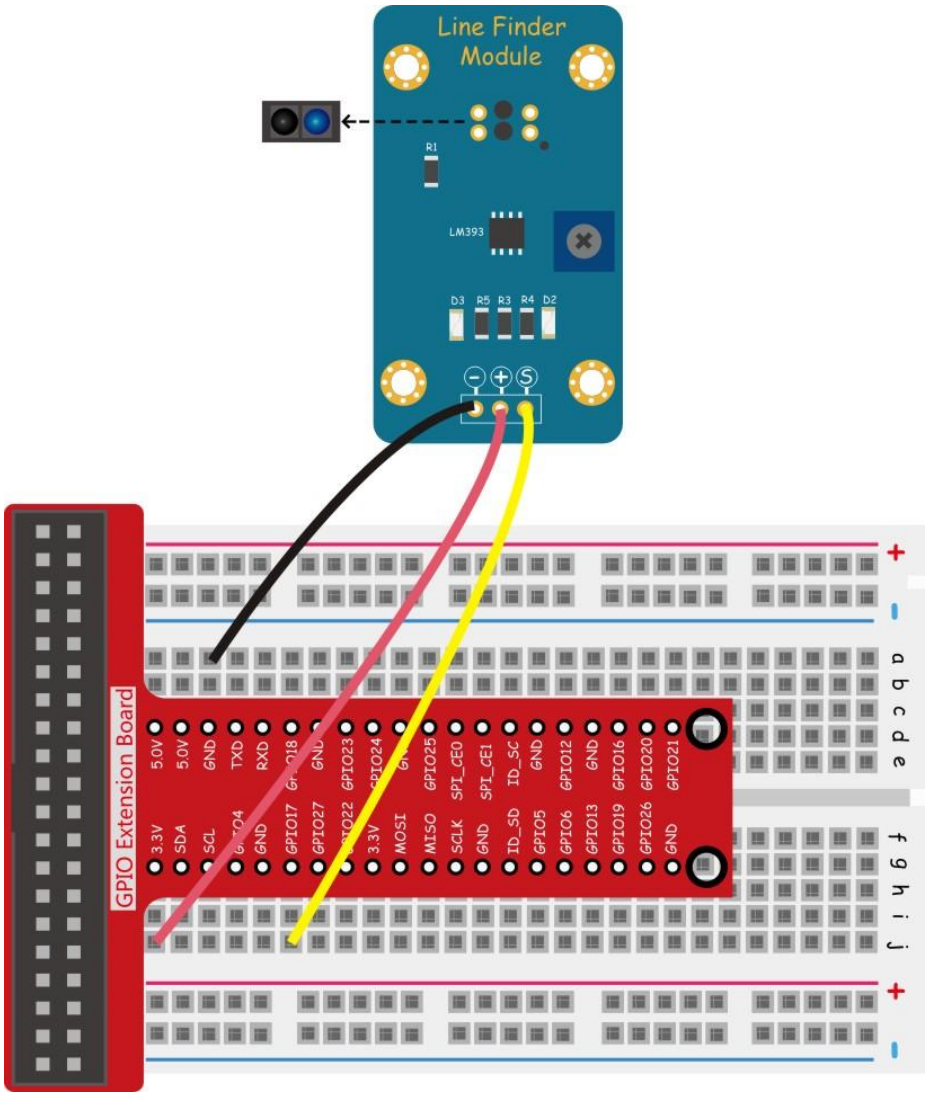
Pin definition:

S	Output
+	VCC
-	GND

In this experiment, we detect a piece of white and another of black paper via the Line Finder 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_Sensor_Kit_for_RPi_C_Code/18_tracking/tracking.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./tracking
```

For Python users:

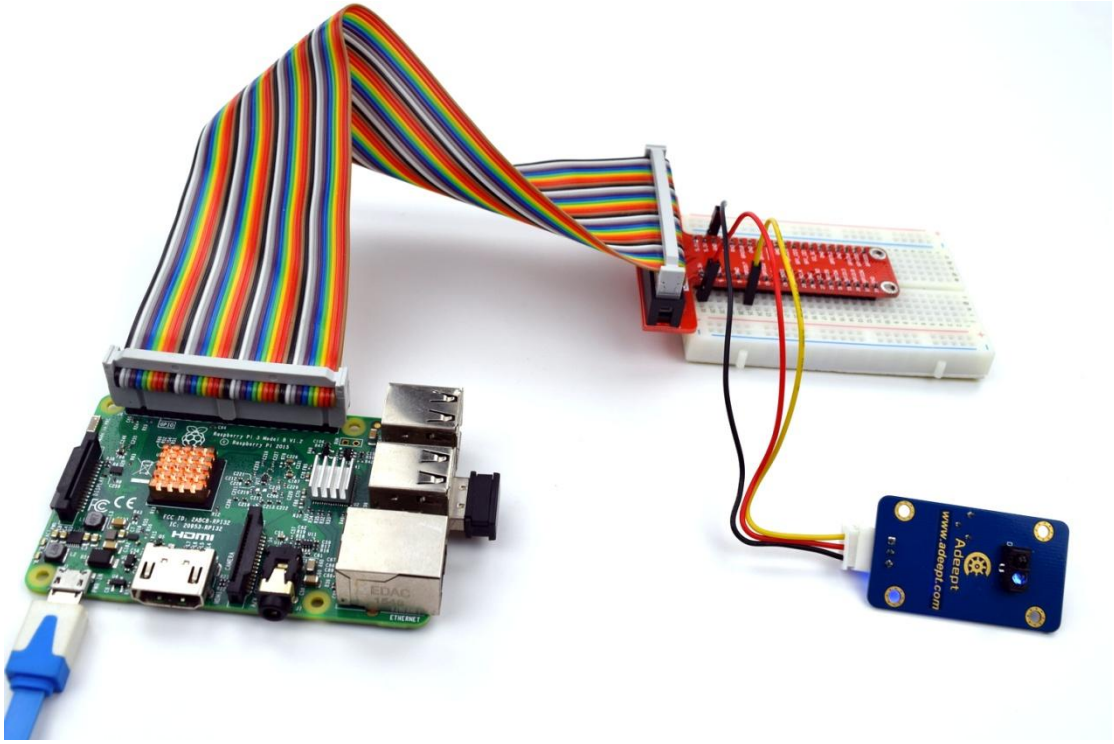
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/18_ tracking.py)

Step 3: Run

```
$ sudo python 18_tracking.py
```

Place the sensor module over a piece of white paper and another of black and you will see the data detected on the terminal. You can adjust the blue potentiometer on the module to change the sensitivity.



Lesson 19 Measuring the Temperature via DS18B20

Introduction

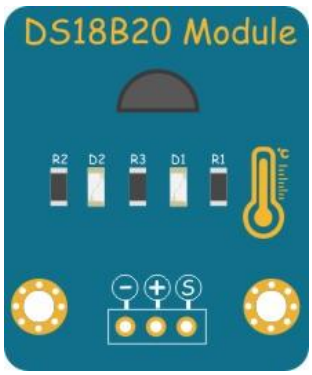
DS18B20 is a single-bus digital temperature sensor of high-precision. The measurement range is - 55℃ - + 125℃ and inherent temperature resolution is 0.5℃. The sensor support multi-point network and multi-point temperature measurement – the measured result is sent to the controller via serial port in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

Components

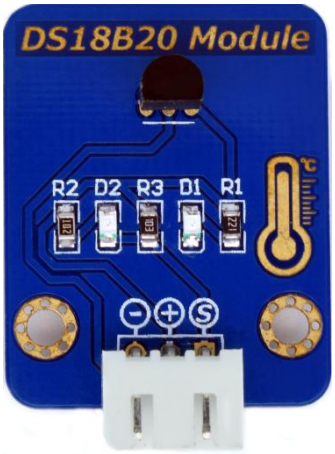
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * DS18B20 Module
- 1 * 3-Pin Wires

Experimental Principle

The Fritzing image:



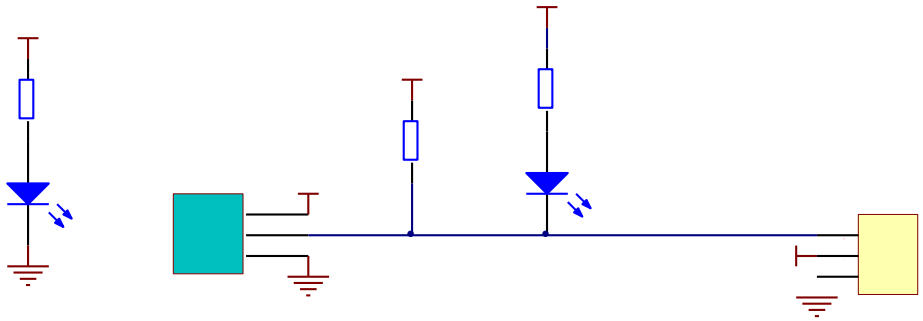
The Physical picture:



Pin definition:

S	Data
+	VCC
-	GND

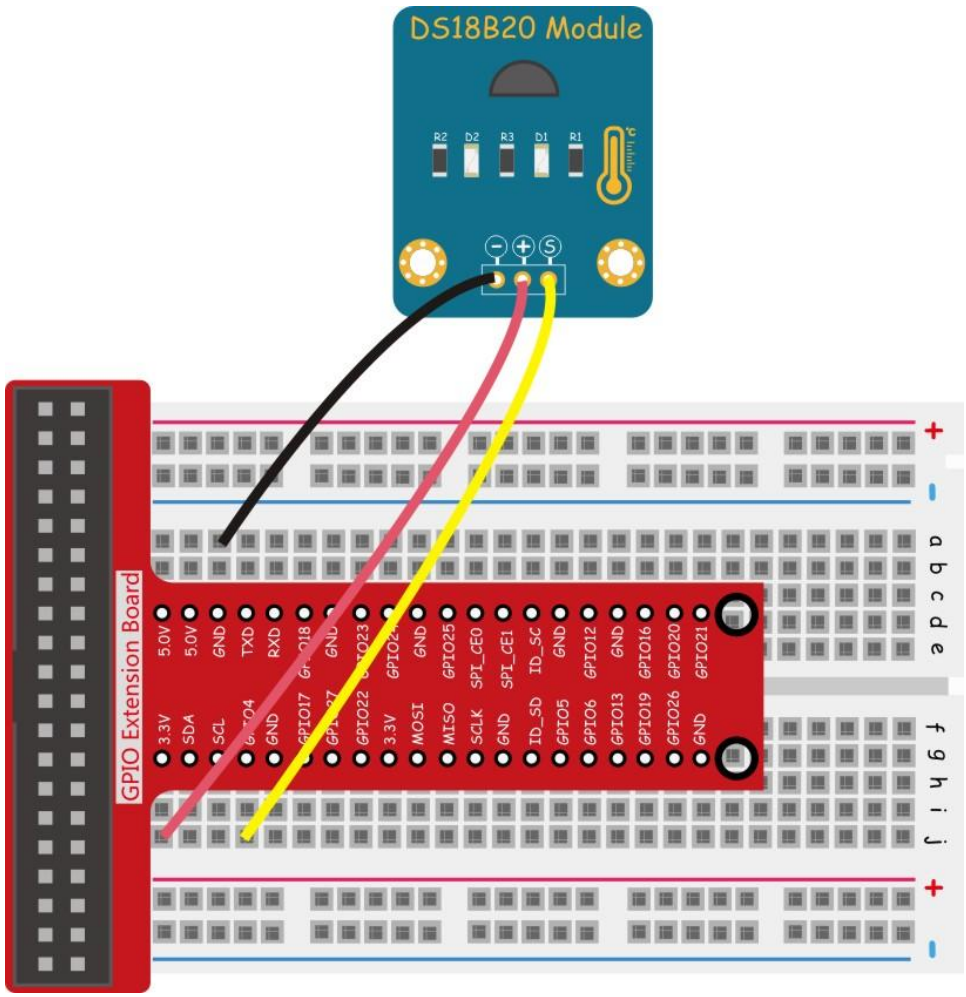
The schematic diagram:



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@rasberrypi:/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@rasberrypi:/sys/bus/w1/devices/28-00000355d573# ls
driver  id  name  power  subsystem  uevent  w1_slave
$ cat w1_slave
The result is as follows:
root@raspberrypi:/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 ℃.

For C language users:

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

(code path: /home/Adeept_Sensor_Kit_for_RPi_C_Code/19_ds18b20/ds18b20_2.c.c)

Step 6: Compile

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

Step 7: Run

```
$ sudo ./ds18b20
```

For Python users:

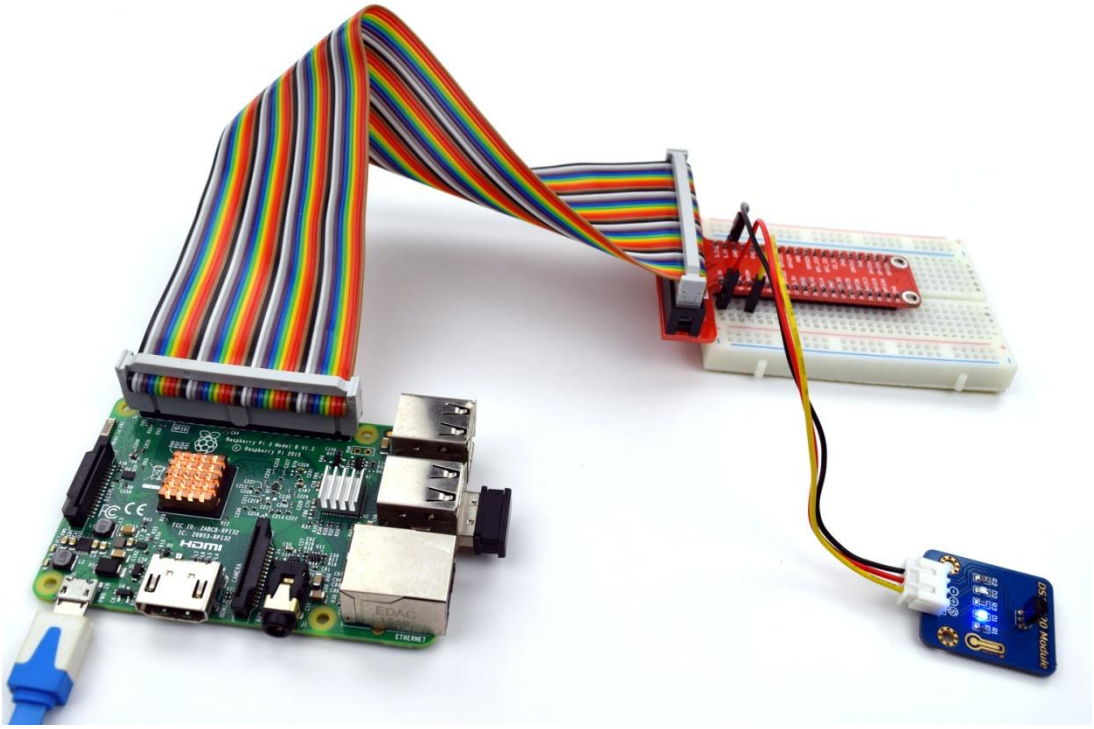
Step 5: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/19_ds18b20.py)

Step 6: Run

```
$ sudo python 19_ds18b20.py
```

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



Lesson 20 Temperature & Humidity Sensor - DHT-11

Introduction

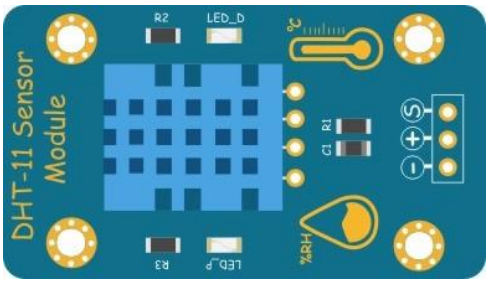
DHT11 is a composite digital thermal sensor that integrates temperature and humidity detection. It can convert the temperature and humidity analog values into digital values via corresponding sensitive components and built-in circuits, which can be directly read by computer or other data collecting devices.

Components

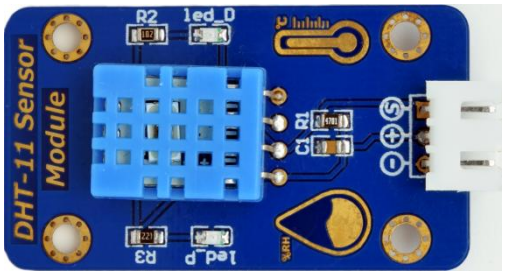
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * DHT-11 Sensor Module
- 1 * 3-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



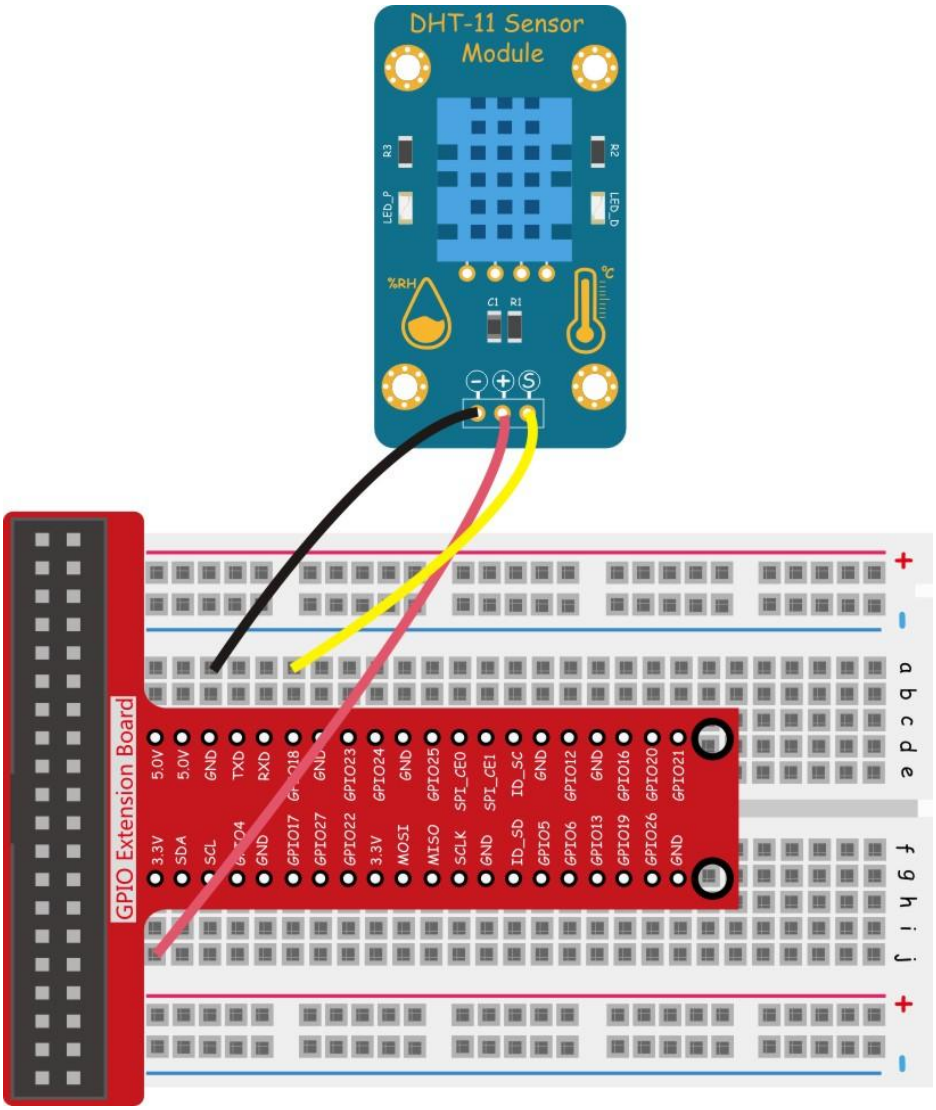
Pin definition:

S	Digital output
+	VCC
-	GND

In this experiment, by programming the Raspberry Pi, we read the temperature and humidity data collected by the DHT11 module by pin 11 of the Raspberry Pi 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/Adeept_Sensor_Kit_for_RPi_C_Code/20_dht11/dht11.c)

Step 3: Compile

```
$ sudo gcc dht11.c -o dht11 -lwiringPi
```

Step 4: Run

```
$ sudo ./dht11
```

For Python users:

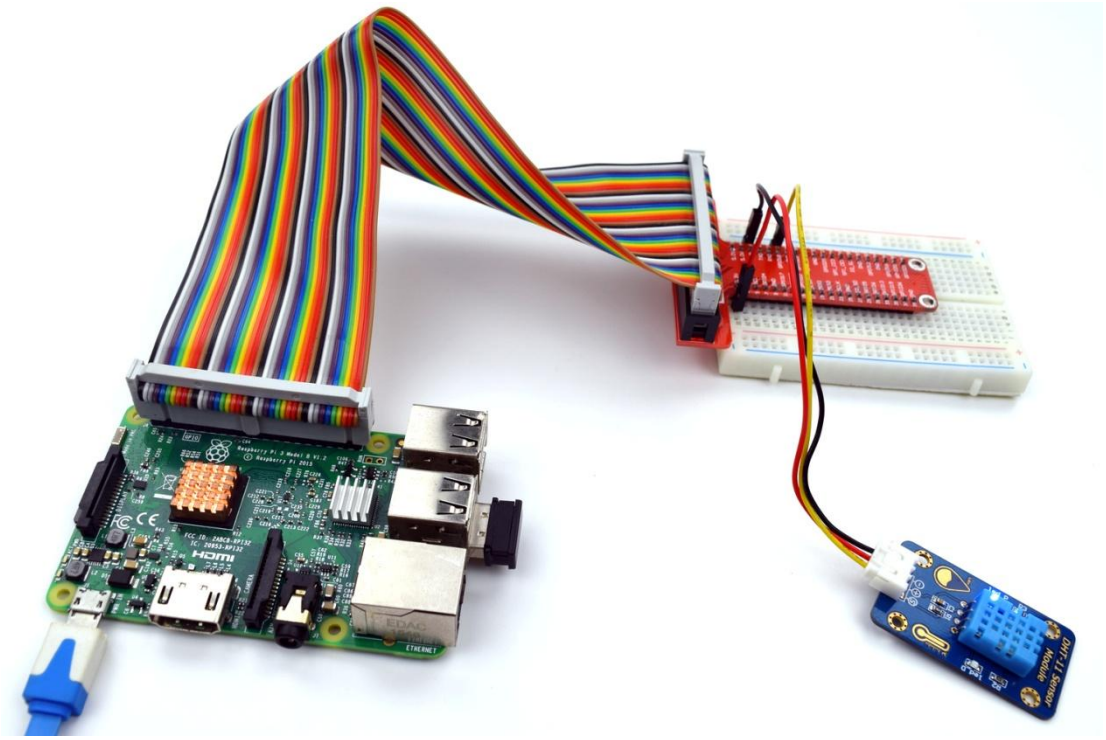
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/20_dht11.py)

Step 3: Run

```
$ sudo python 20_dht11.py
```

Now, you will see the data of current temperature and humidity displayed on the terminal.



Lesson 21 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:

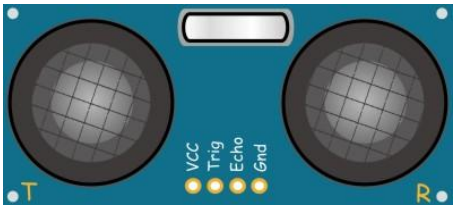
1. The module is not suggested to connect wires when power is on. If you have to do so, please first connect the GND and then other pins; otherwise, the module may not work.
2. During the ranging, the area of the targeted object should be no less than 0.5cm and the surface facing the module should be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

Components

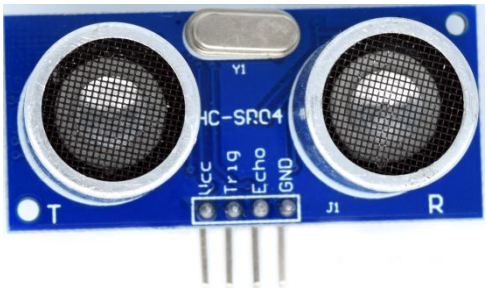
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Ultrasonic Sensor Module
- 4 * Jumper Wires

Experimental Principle

The Fritzing image:



The Physical picture:



Pin definition:

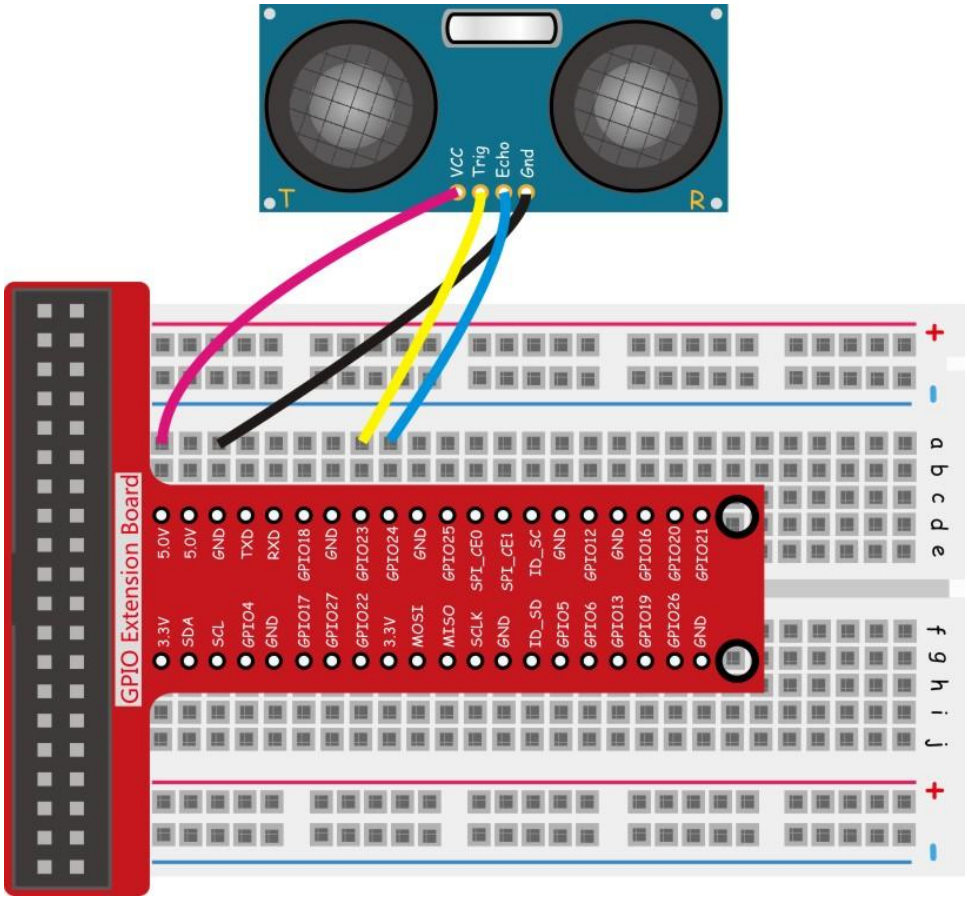
Trig	Iutput
Echo	Output

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/Adeept_Sensor_Kit_for_RPi_C_Code/21_ultrasonicSensor/distance.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./distance
```

For Python users:

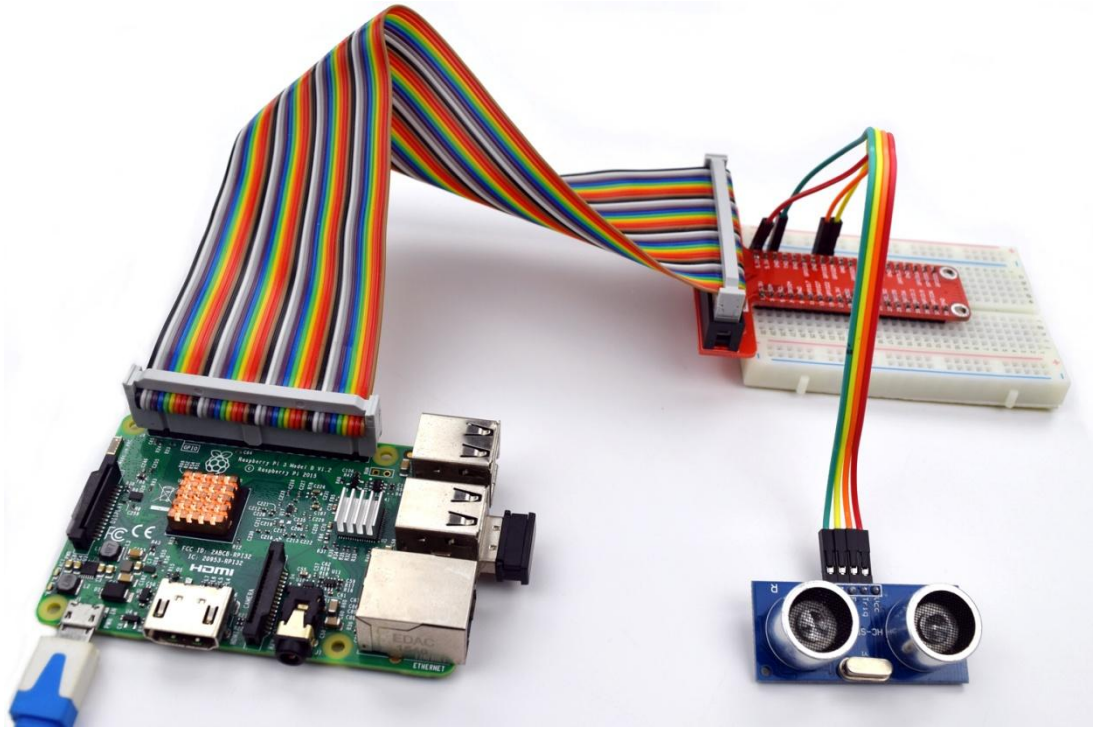
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/21_distance.py)

Step 3: Run

```
$ sudo python 21_distance.py
```

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



Lesson 22 Acceleration Sensor - ADXL345

Introduction

The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3-wire or 4-wire) or I2C digital interface. The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (3.9 mg/LSB) enables measurement of inclination changes less than 1.0°.

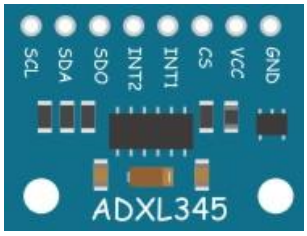
Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * ADXL345 Accelerometer Module
- Several jumper wires

Experimental Principle

The Fritzing image:

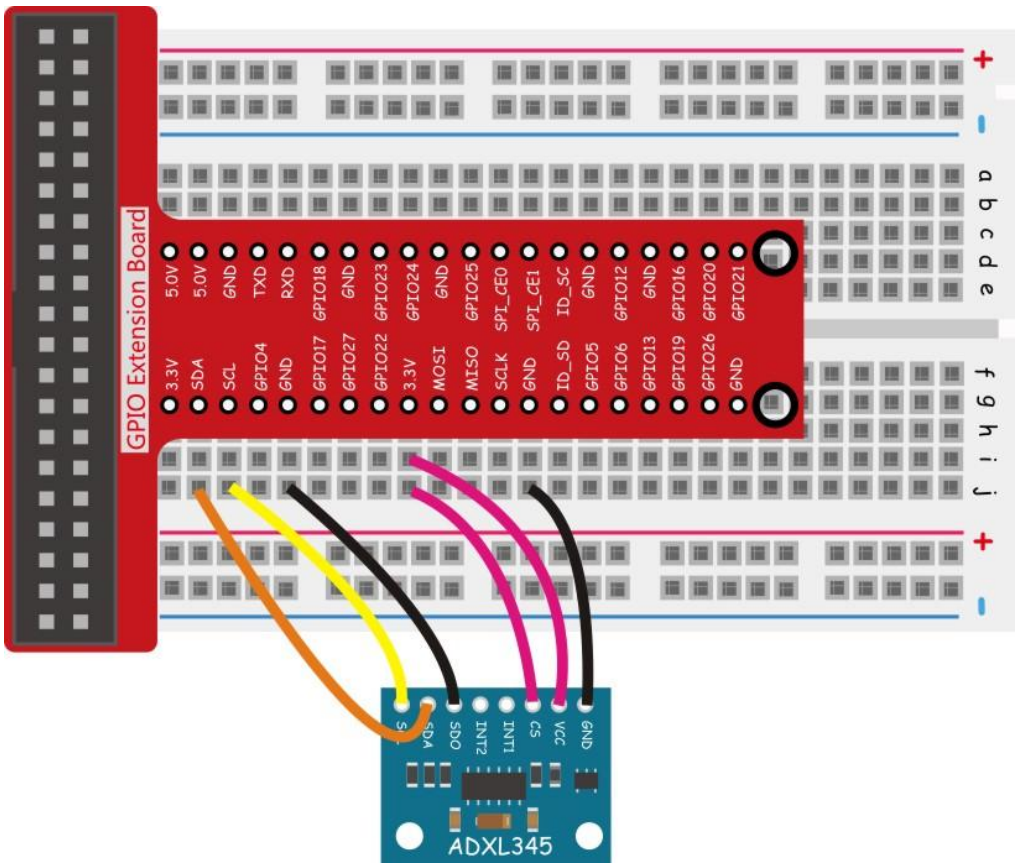


The Physical picture:

In this lesson, we will learn how to use an acceleration sensor ADXL345 to get the acceleration data.

Experimental Procedures

Step 1: Build the circuit



NOTE:

The following program uses an I2C interface. Before running the program, please make sure the I2C driver module of Raspberry Pi has loaded normally.

For C language users:

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

(Code path: /home/Adeept_Sensor_Kit_for_RPi_C_Code/22_ADXL345/adxl345.c)

Step 3: Compile

```
$ gcc adxl345.c -o adxl345 -lwiringPi
```

Step 4: Run

```
$ sudo ./adxl345
```

For Python users:

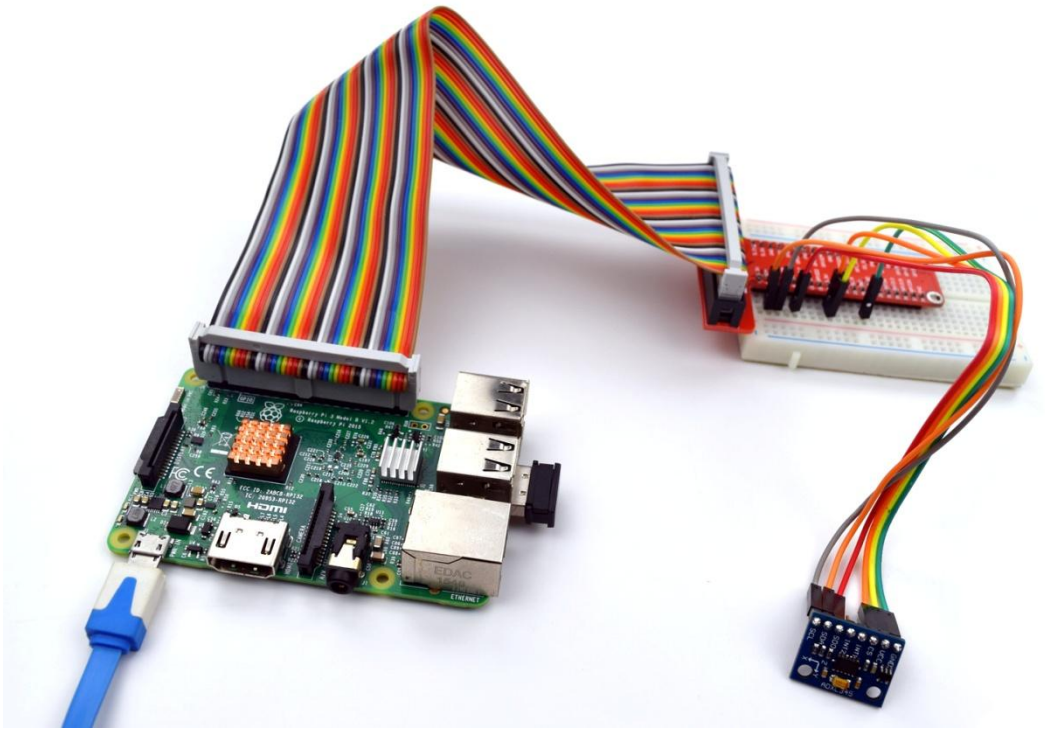
Step 2: Edit and save the code with vim or nano.

(Code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/22_ADXL345.py)

Step 3: Run

```
$ sudo python 22_ADXL345.py
```

Now you should see the acceleration data displayed on the terminal.



Lesson 23 Barometric Sensor - BMP180

Introduction

The BMP180 is the new digital barometric pressure sensor, with a very high performance, which enables applications in advanced mobile devices, such as smart phones, tablet PCs and sports devices. It follows the BMP085 and brings many improvements, like the smaller size and the expansion of digital interfaces.

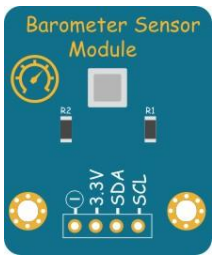
The ultra-low power consumption down to 3uA makes the BMP180 the leader in power saving for your mobile devices. BMP180 is also distinguished by its very stable behavior(performance) with regard to the independency of the supply voltage.

Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * BMP180 Module
- 1 * 4-Pin Wires

Experimental Principle

The Fritzing image:

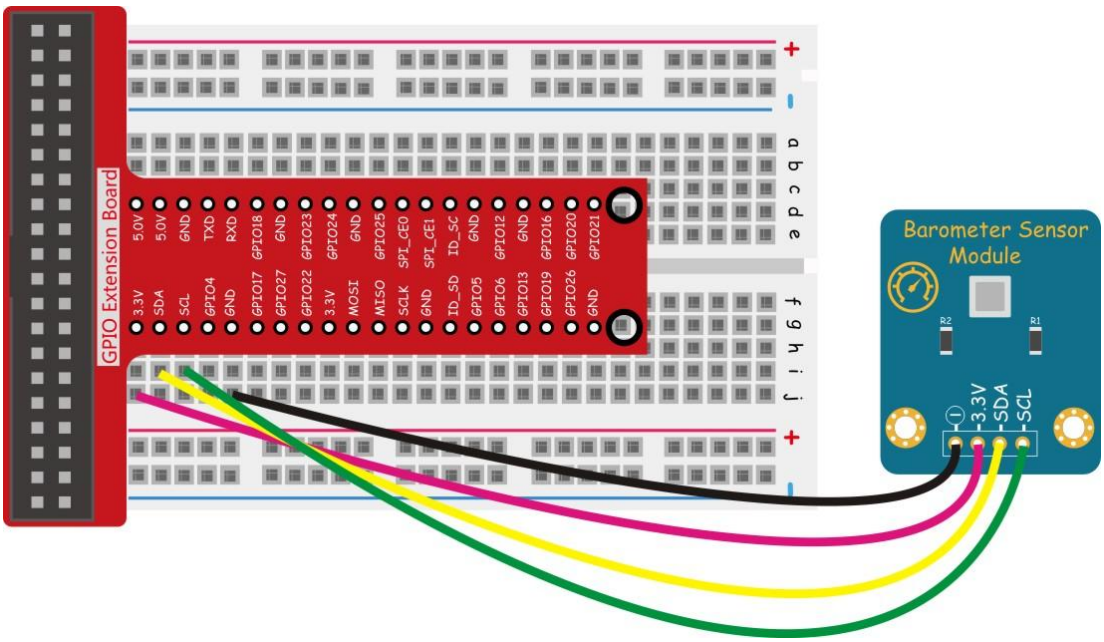


The Physical picture:

In this lesson, we will learn how to use an pressure sensor BMP180 to get the atmospheric pressure and altitude data.

Experimental Procedures

Step 1: Build the circuit



NOTE:

The following program uses an I2C interface. Before running the program, please make sure the I2C driver module of Raspberry Pi has loaded normally.

For C language users:

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

(Code path: /home/Adeept_Sensor_Kit_for_RPi_C_Code/23_bmp180/bmp180dev3.c)

Step 3: Compile

```
$ gcc bmp180dev3.c -o bmp180 -lm
```

Step 4: Run

```
$ sudo ./bmp180
```

For Python users:

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

(Code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/23_bmp180/examples/simpletest.py)

Step 3: Run

```
$ sudo python simpletest.py
```

Now you should see the atmospheric pressure and altitude data displayed on the terminal.


Barometer
Sensor Module

Lesson 24 Dot-matrix Display

Introduction

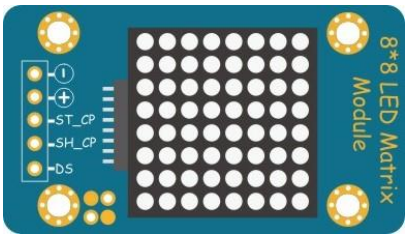
The module drives the 8*8 LED Matrix by cascading two 74HC595 chips. The module communicates with the microcontroller through SPI(Serial Peripheral Interface). It only occupies three I/Os of the Raspberry Pi and save precious ones for connecting other devices.

Components

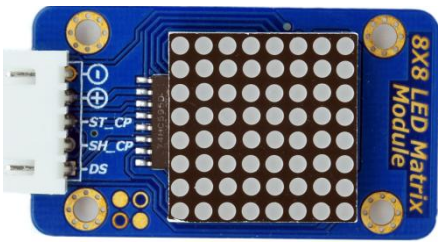
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * 8*8 LED Matrix Module
- 1 * 5-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



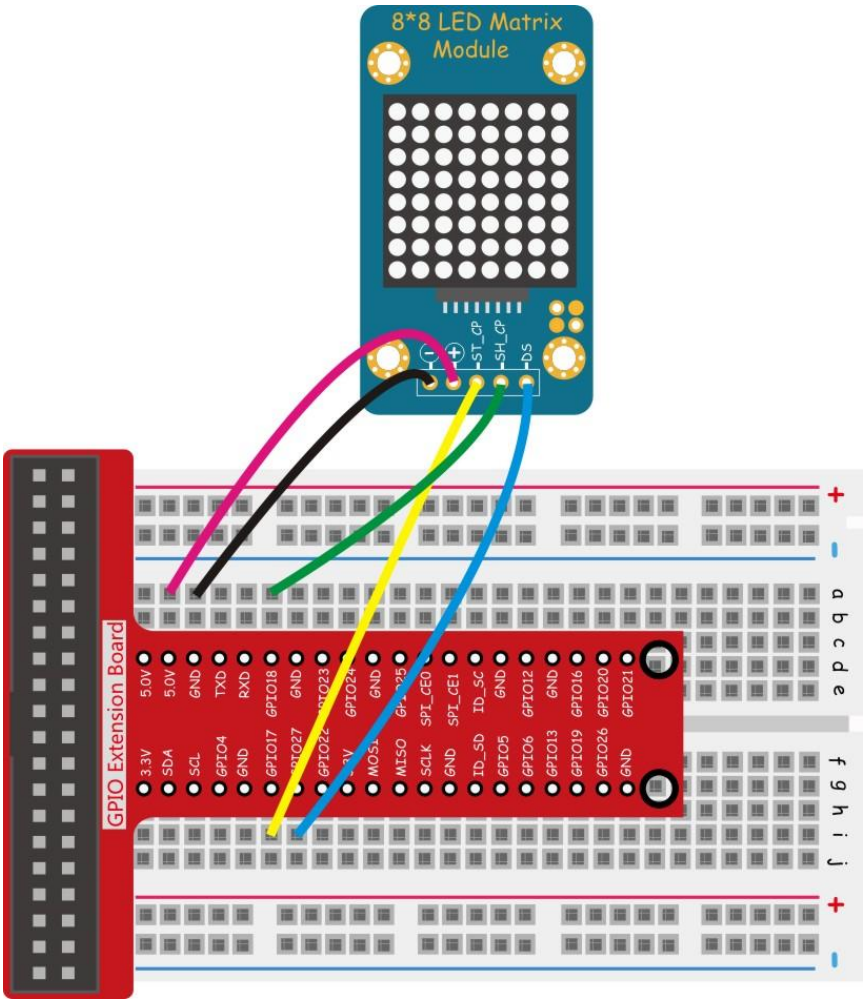
Pin definition:

DS	Input
SH_CP	Input
ST_CP	Input
+	VCC
-	GND

In this experiment, by programming the Raspberry Pi, we send the data to the dot matrix module via the SPI interface and make the display scroll the characters “Adeept”.

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_Sensor_Kit_for_RPi_C_Code/24_ledMatrix/ledMatrix.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./ledMatrix
```

For Python users:

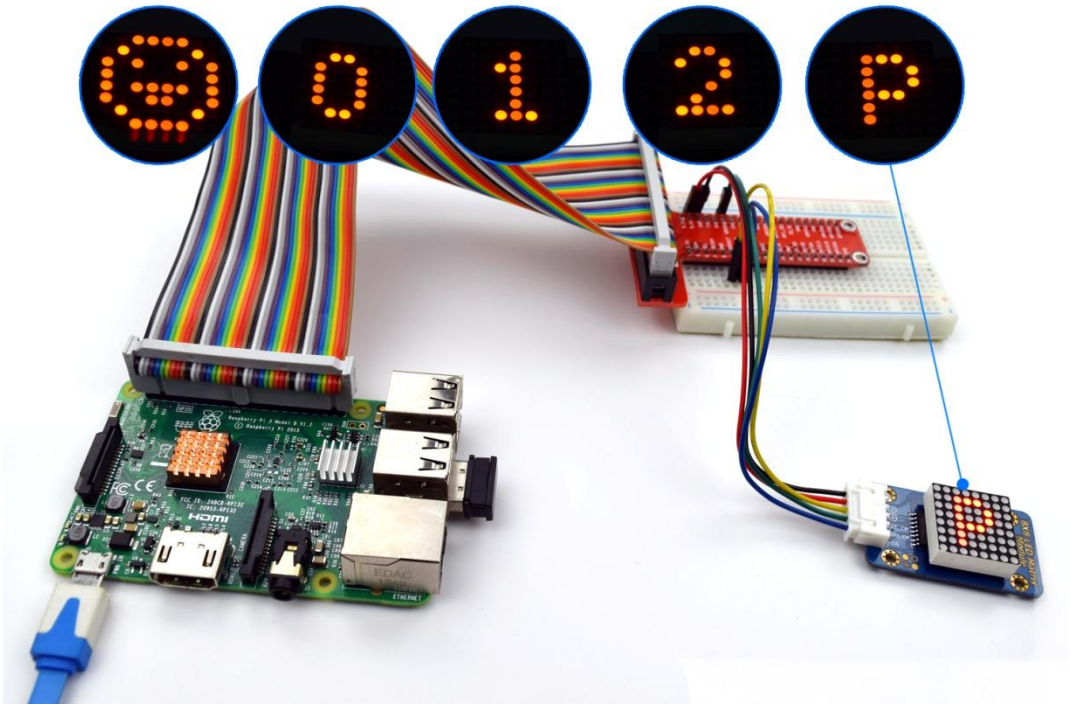
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/24_LedMatrix.py)

Step 3: Run

```
$ sudo python 24_LedMatrix.py
```

Now you can see on the dot matrix module, “Adeept” is displayed in the way of scrolling.



Lesson 25 LED Bar Graph

Introduction

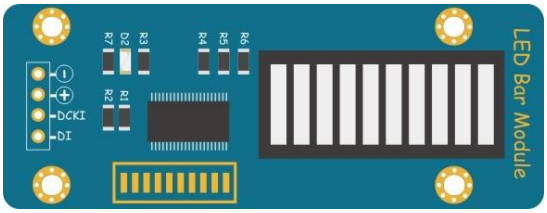
The LED bar is an analog indicating component usually used for volume indication.

Components

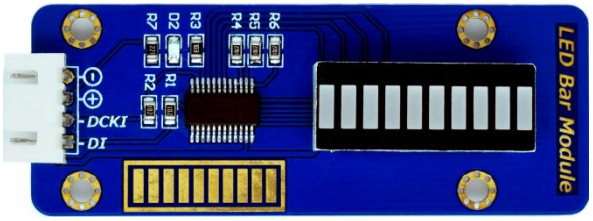
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * LED Bar Module
- 1 * 4-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



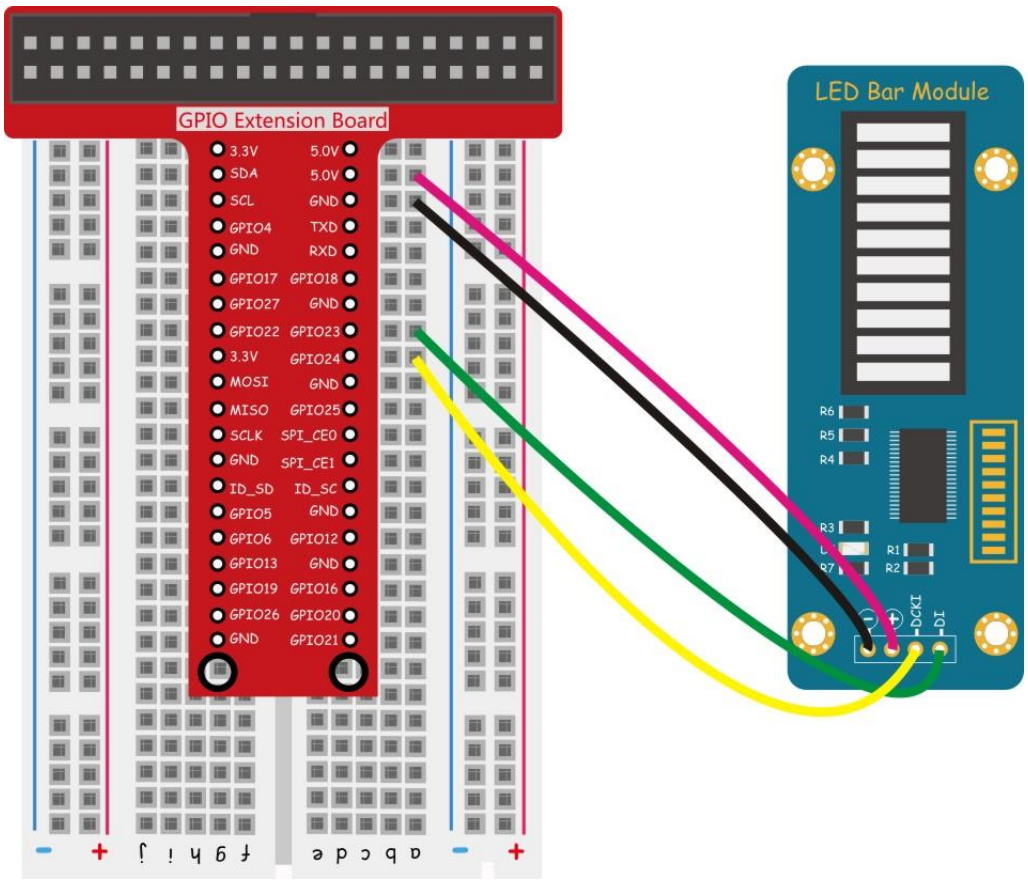
Pin definition:

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/Adeept_Sensor_Kit_for_RPi_C_Code/25_ledBar/ledBar.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./ledBar
```

For Python users:

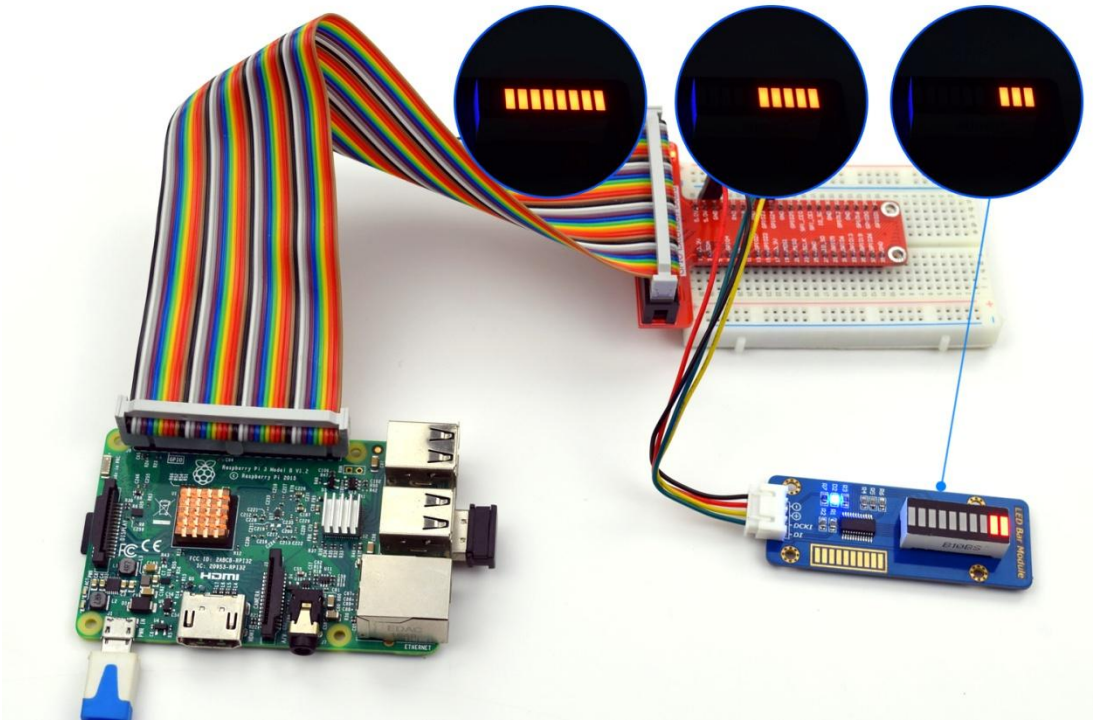
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/25_ledBar.py)

Step 3: Run

```
$ sudo python 25_ledBar.py
```

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



Lesson 26 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:

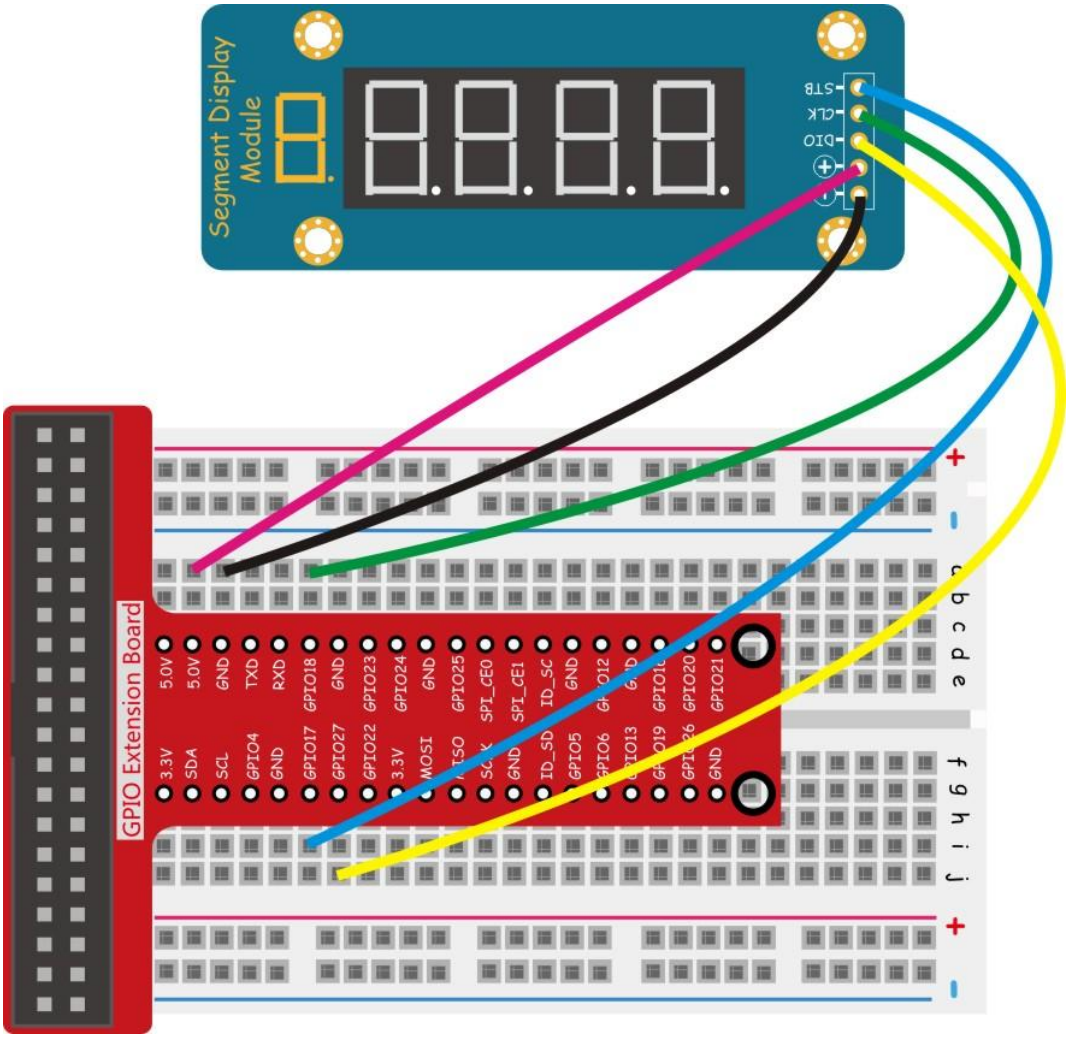
Pin definition:

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/Adeept_Sensor_Kit_for_RPi_C_Code/26_segmentDisplay/segment.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./segment
```

For Python users:

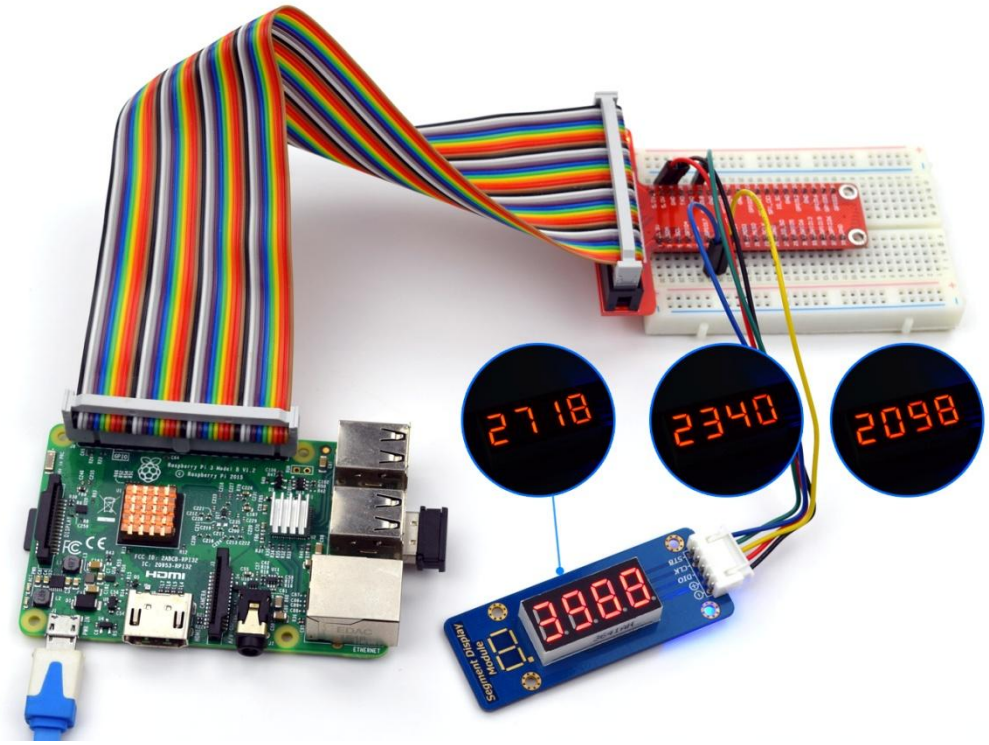
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/26_segment.py)

Step 3: Run

```
$ sudo python 26_segment.py
```

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



Lesson 27 Potentiometer

Introduction

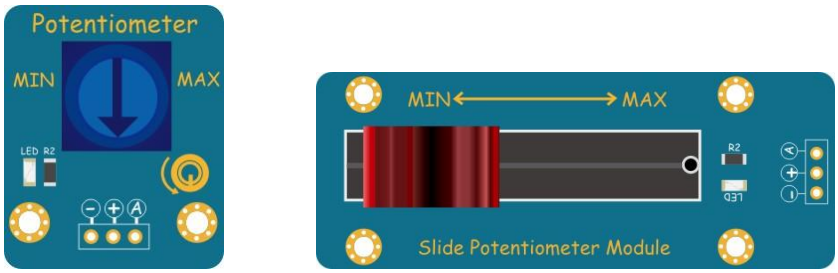
Potentiometer is a resistor whose resistance can be adjusted continuously. When its shaft is turned, the moving contact (or wiper) slides along the resistor.

Components

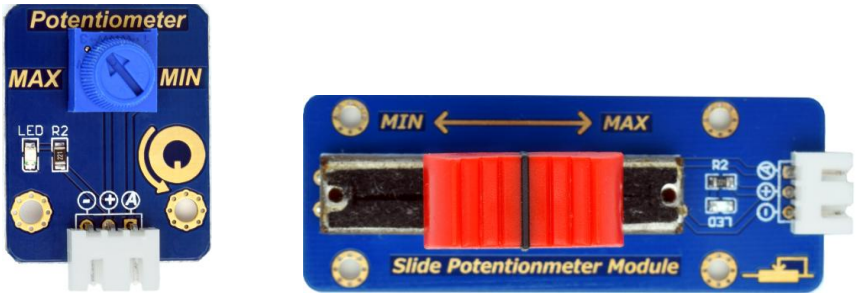
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * ADC0832 Module
- 1 * Potentiometer Module
- 1 * Slide Potentiometer Module
- 2 * 3-Pin Wires
- 1 * 5-Pin Wires

Experimental Principle

The Fritzing image:



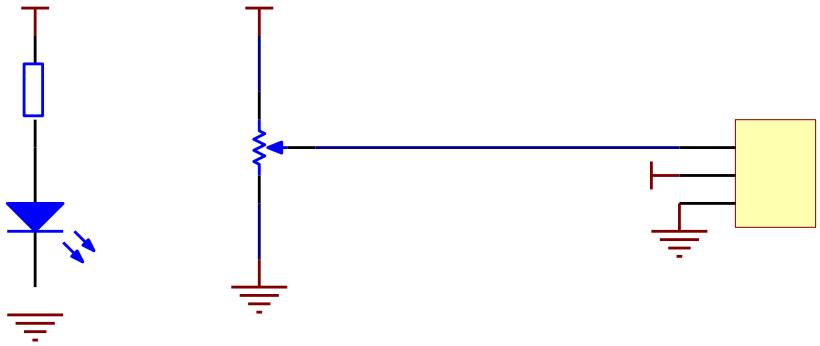
The Physical picture:



Pin definition:

A	Analog Output
+	VCC
-	GND

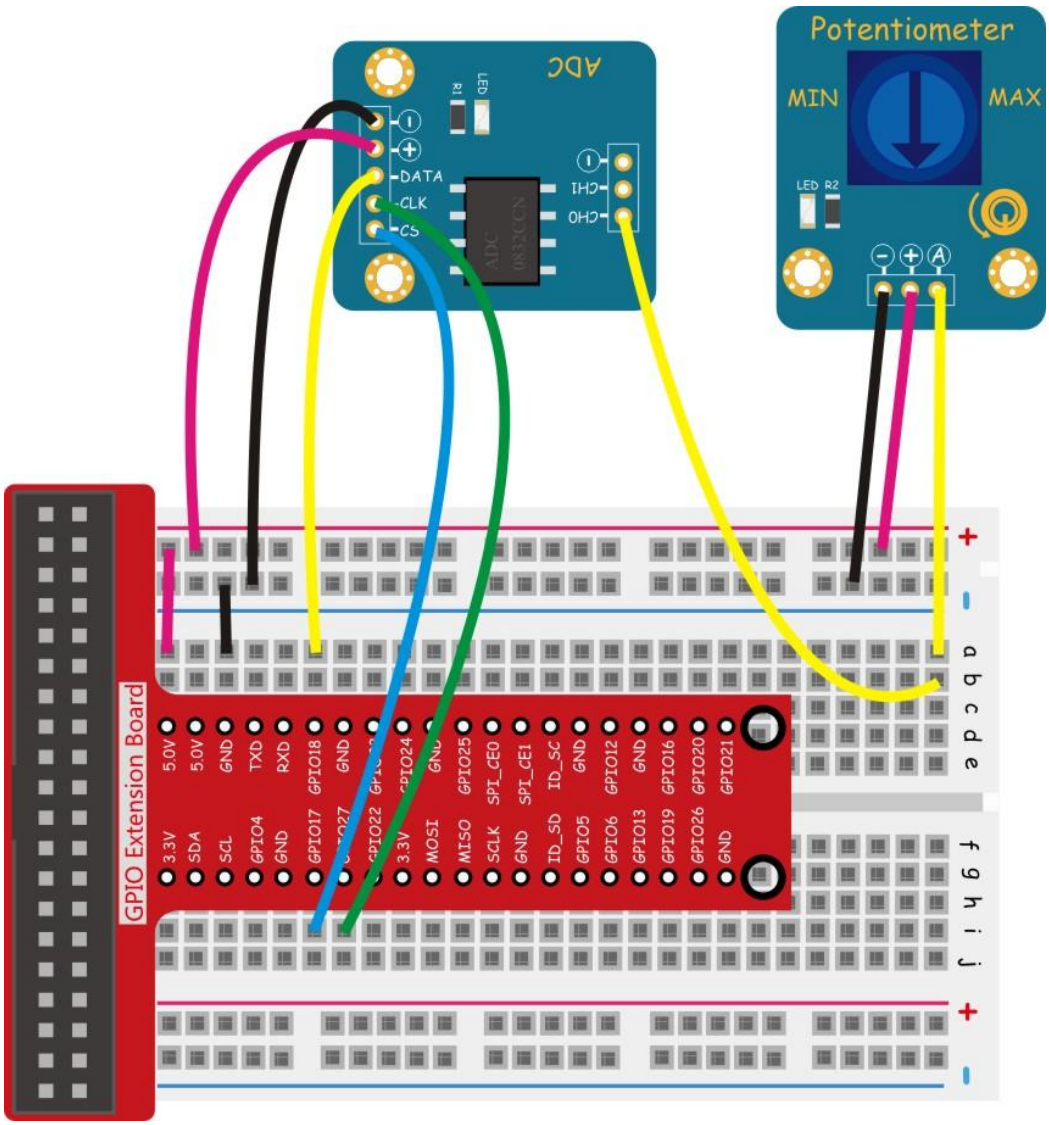
The schematic diagram:

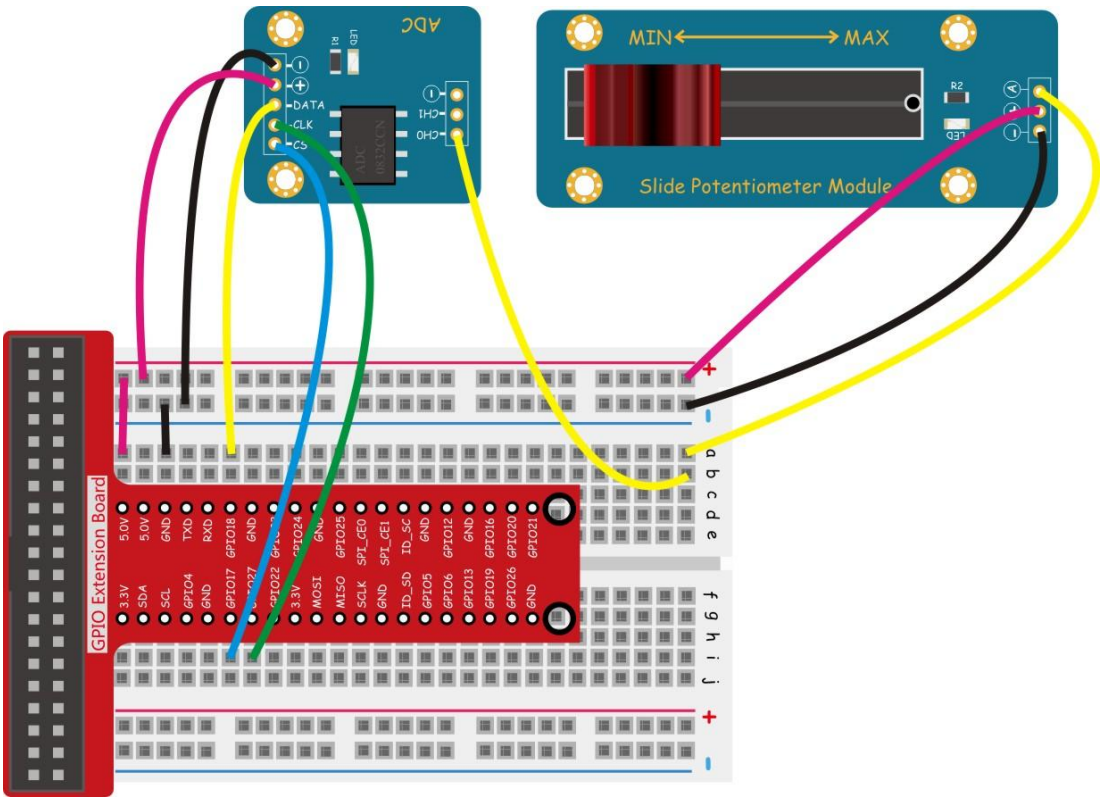


In this experiment, by programming the Raspberry Pi, we collect the analog values output by the Potentiometer module through pin CH0 of the ADC0832, convert it to digital values and display them on the terminal.

Experimental Procedures

Step 1: Build the circuit


Potentiometer
MIN
MAX
LED R2
ADC
DATA
CLK
CS
CH1
CH0
GPIO Extension Board
5.0V
5.0V
GND
TXD
RXD
GPIO18
GND
3.3V
SDA
SCL
GPIO4
GND
GPIO17
MOSI
MISO
SCLK
GND
ID_SD
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
GND
GPIO12
GND
GPIO16
GPIO20
GPIO21
SPI_CE0
SPI_CE1
ID_SC
a b c d e
f g h i j



For C language users:

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

(code path: /home/Adeept_Sensor_Kit_for_RPi_C_Code/27_potentiometer/potentiometer.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./potentiometer
```

For Python users:

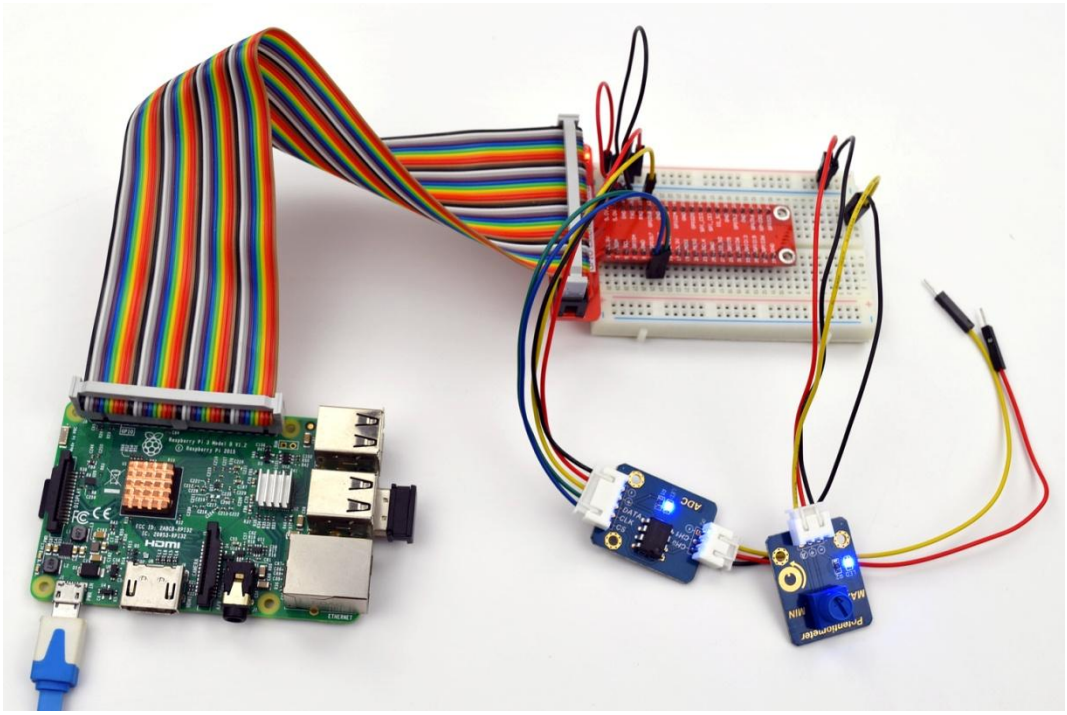
Step 2: Edit and save the code with vim or nano.

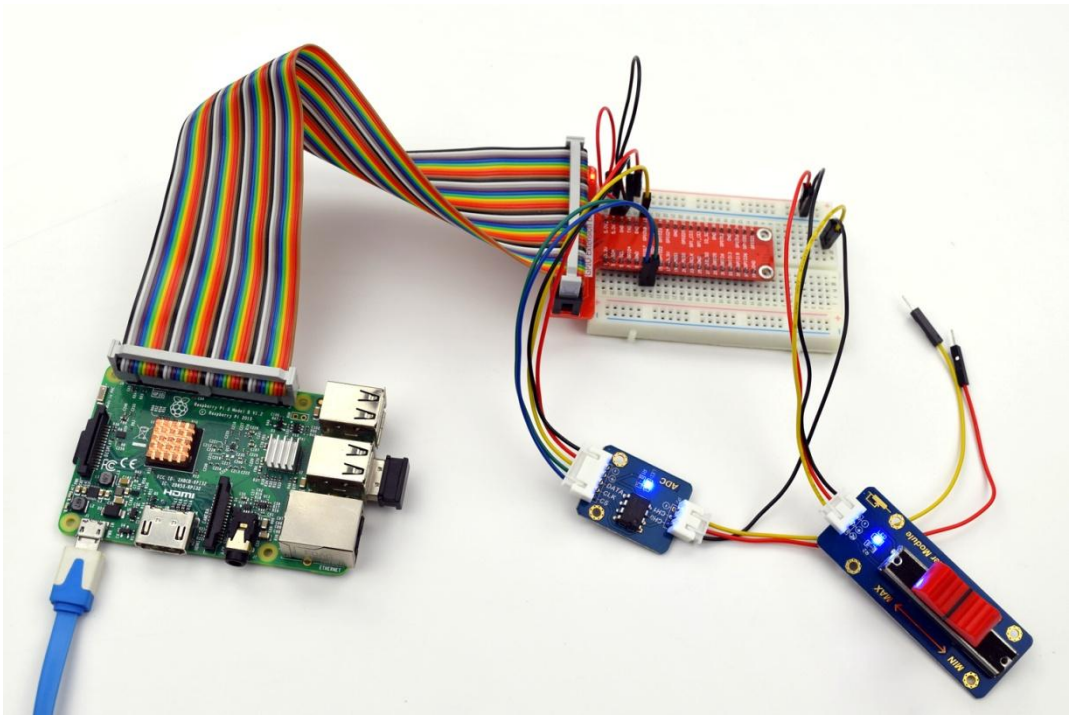
(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/27_potentiometer.py)

Step 3: Run

```
$ sudo python 27_potentiometer.py
```

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





Lesson 28 Photoresistor

Introduction

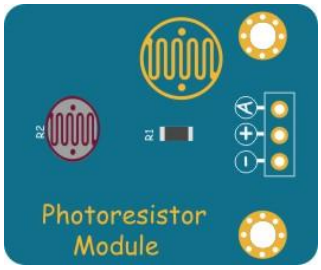
The photoresistor module is a resistor module designed based on the principle of photoconductive effect of semiconductors, of which the resistance varies with the intensity of incident light. The resistance of the photoresistor we use decreases with stronger incident light and increases with weaker light. 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:

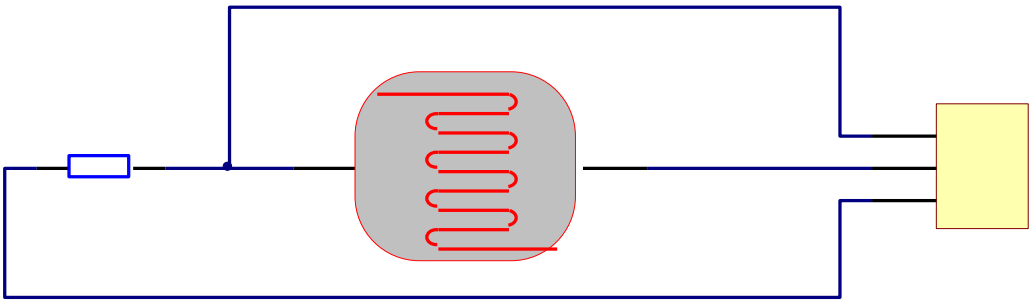


The Physical picture:

Pin definition:

A	Analog Output
+	VCC
-	GND

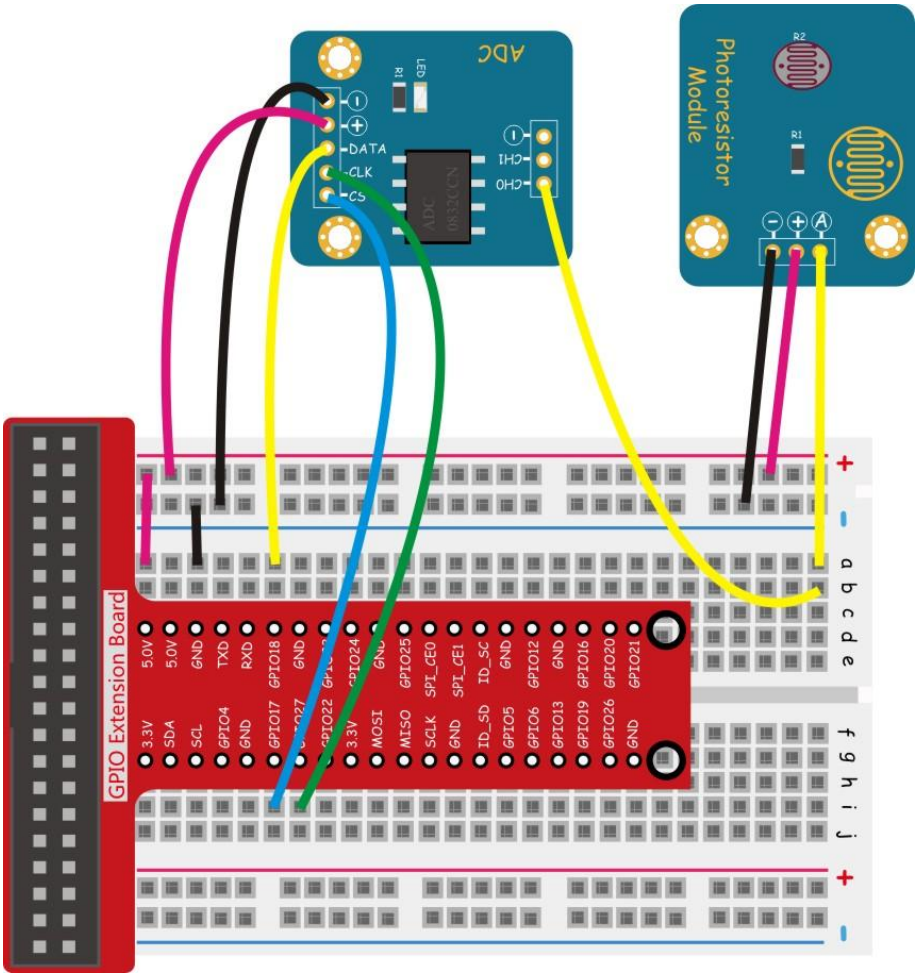
The schematic diagram:



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/Adeept_Sensor_Kit_for_RPi_C_Code/28_photoresistor/photoresistor.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./photoresistor
```

For Python users:

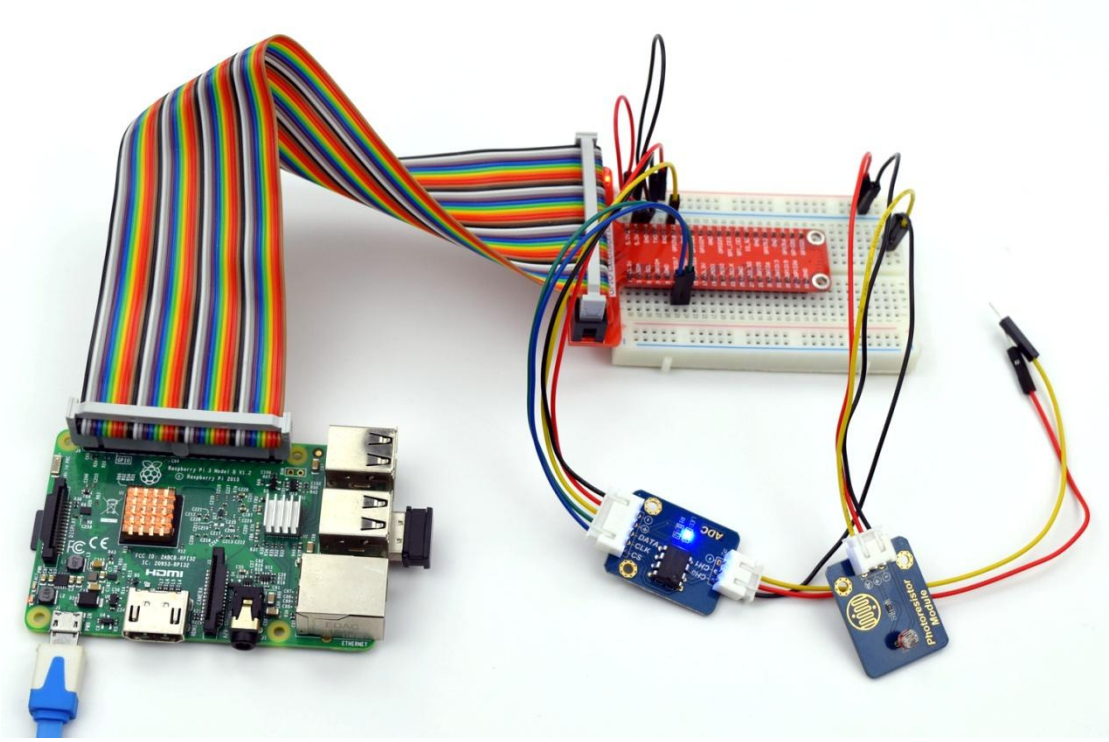
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/28_photoresistor.py)

Step 3: Run

```
$ sudo python 28_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 29 Thermistor

Introduction

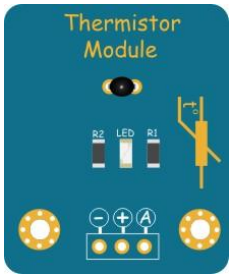
Thermistors can be divided into two types by temperature coefficient: positive temperature coefficient (PTC) and negative temperature coefficient (NTC). The typical feature of a thermistor is sensitive to temperature – its resistance varies with ambient temperature changes. For PTC thermistor, when the 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:



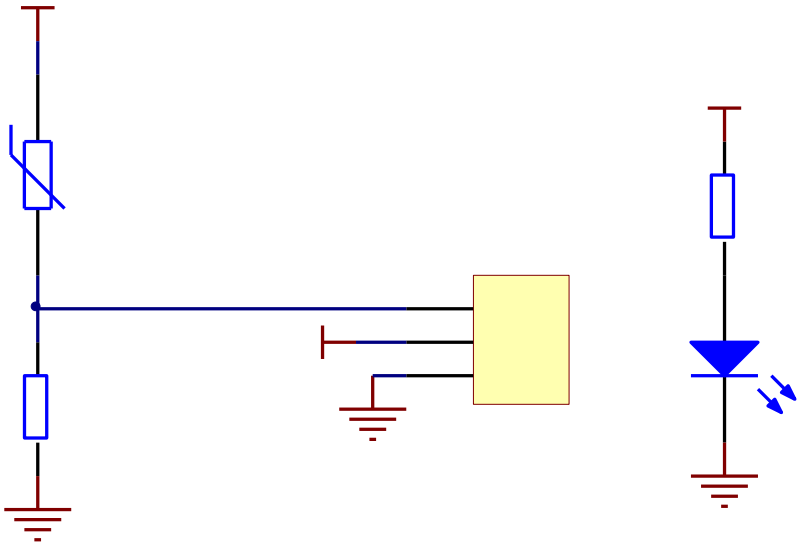
The Physical picture:

Pin definition:

A	Analog Output

+	VCC
-	GND

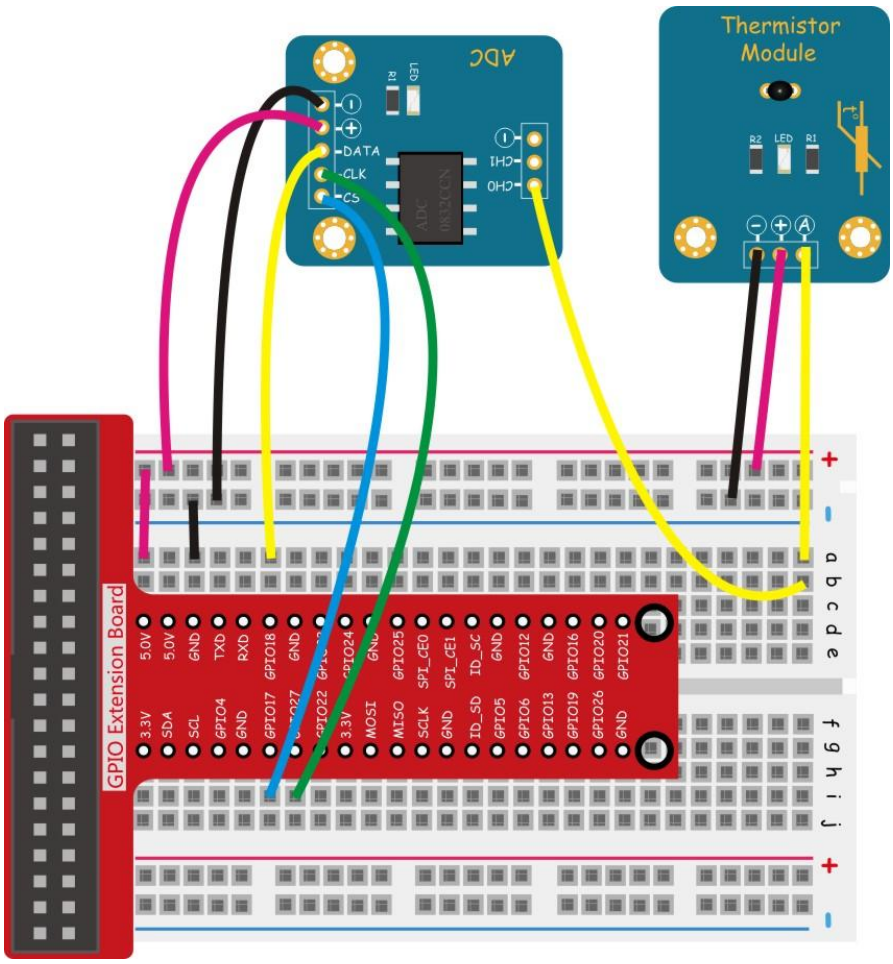
The schematic diagram:



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/Adeept_Sensor_Kit_for_RPi_C_Code/29_thermistor/thermistor.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./thermistor
```

For Python users:

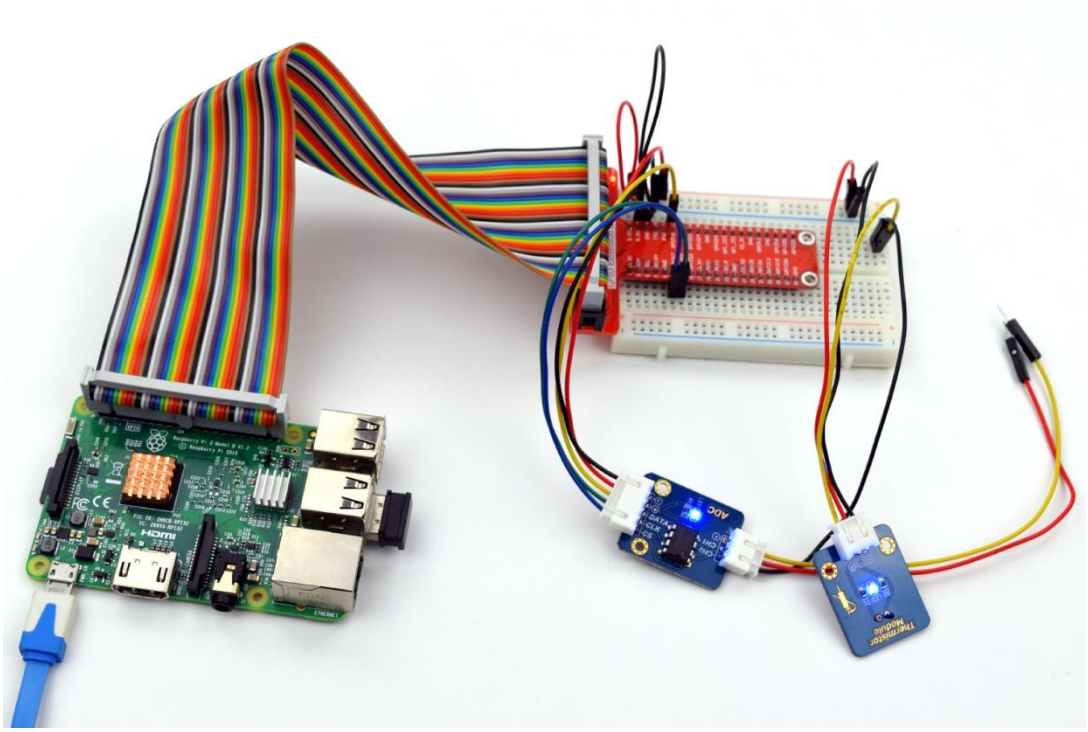
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/29_thermistor.py)

Step 3: Run

```
$ sudo python 29_thermistor.py
```

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


ADC
THERMISTOR
MODULE

Lesson 30 Water Level Detection

Introduction

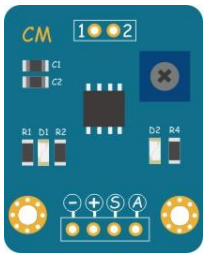
The module is a simple water level sensor. It measures the water volume by the printed wires exposed to the air on the module. The more water on the surface, more wires connected. Thus, the area of electrified wires gets larger, so the output voltage will increase. The surface of the sensor is gilded to prolong its life.

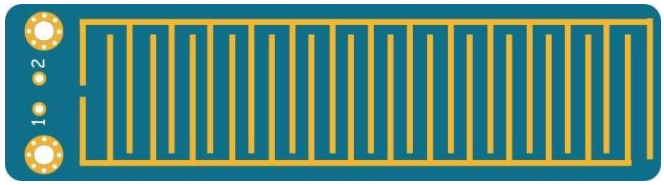
Components

- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * Water Level Module
- 1 * LM393 CM(Comparator) 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:



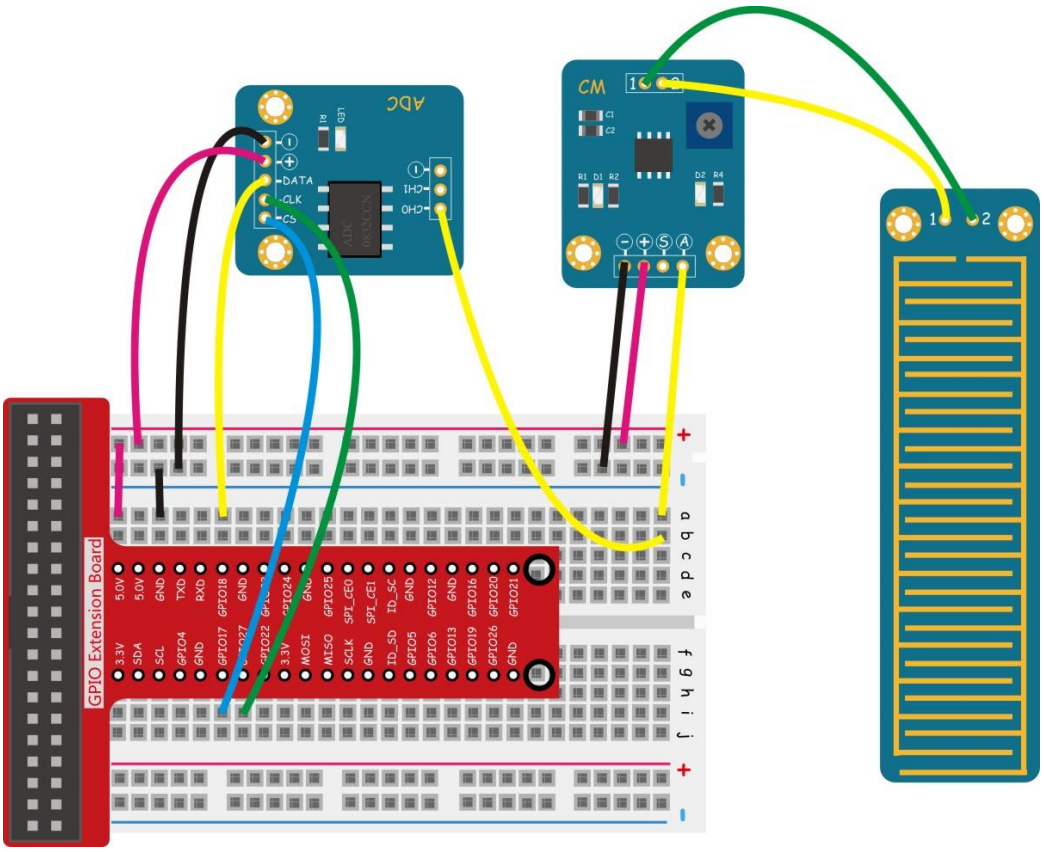


The Physical picture:

This experiment collects the data of water level by the Water Level Sensor module and displays 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/Adeept_Sensor_Kit_for_RPi_C_Code/30_waterLevel/waterLevel.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./waterLevel
```

For Python users:

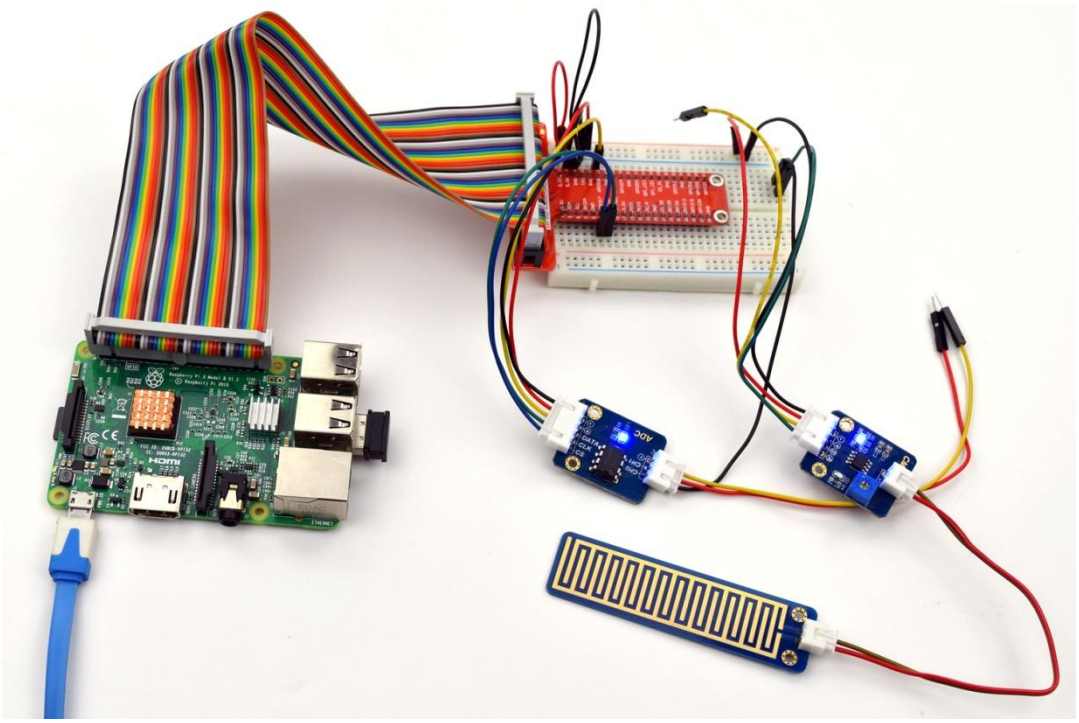
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/30_waterLevel.py)

Step 3: Run

```
$ sudo python 30_waterLevel.py
```

Take a bottle with some water. Place the Water Level Sensor module in the water at end and add water into the bottle. You will see the value of water level displayed on the terminal and changes as you add water.



Lesson 31 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.

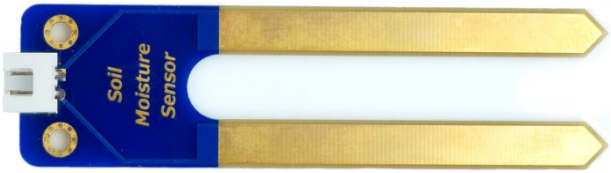
Components

- 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:

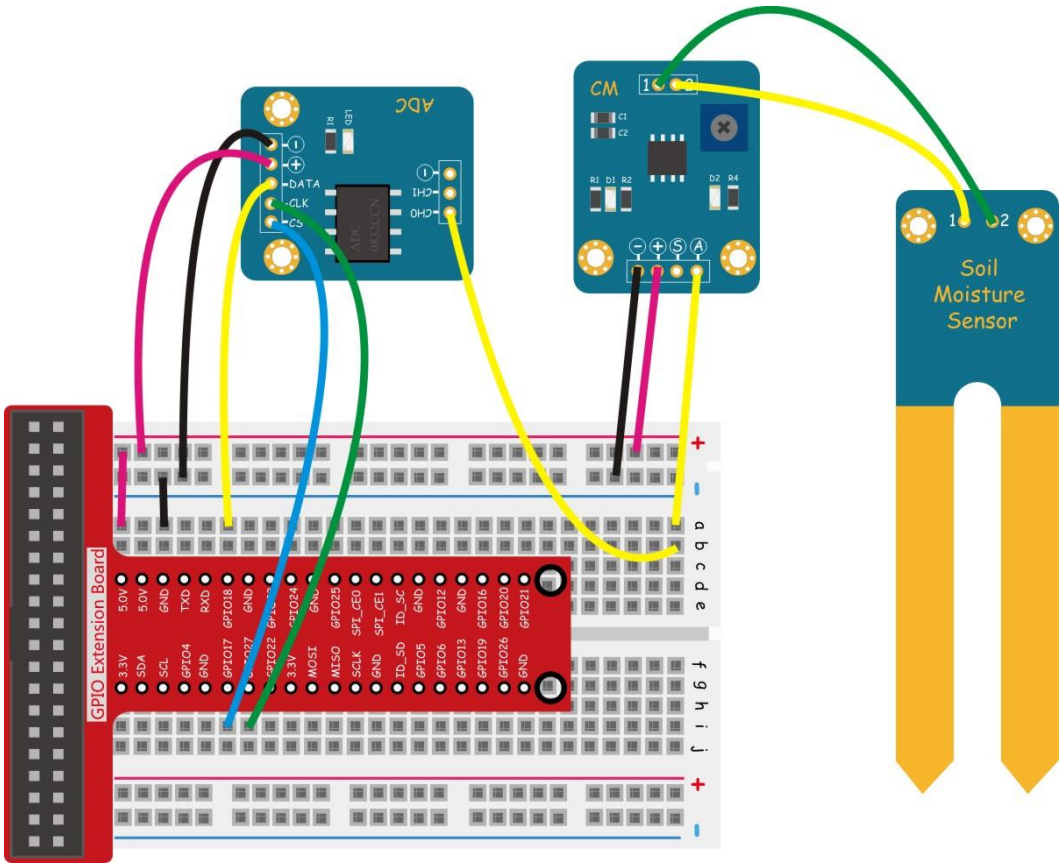
The Physical picture:



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/Adeept_Sensor_Kit_for_RPi_C_Code/31_soilMoisture/soilMoisture.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./soilMoisture
```

For Python users:

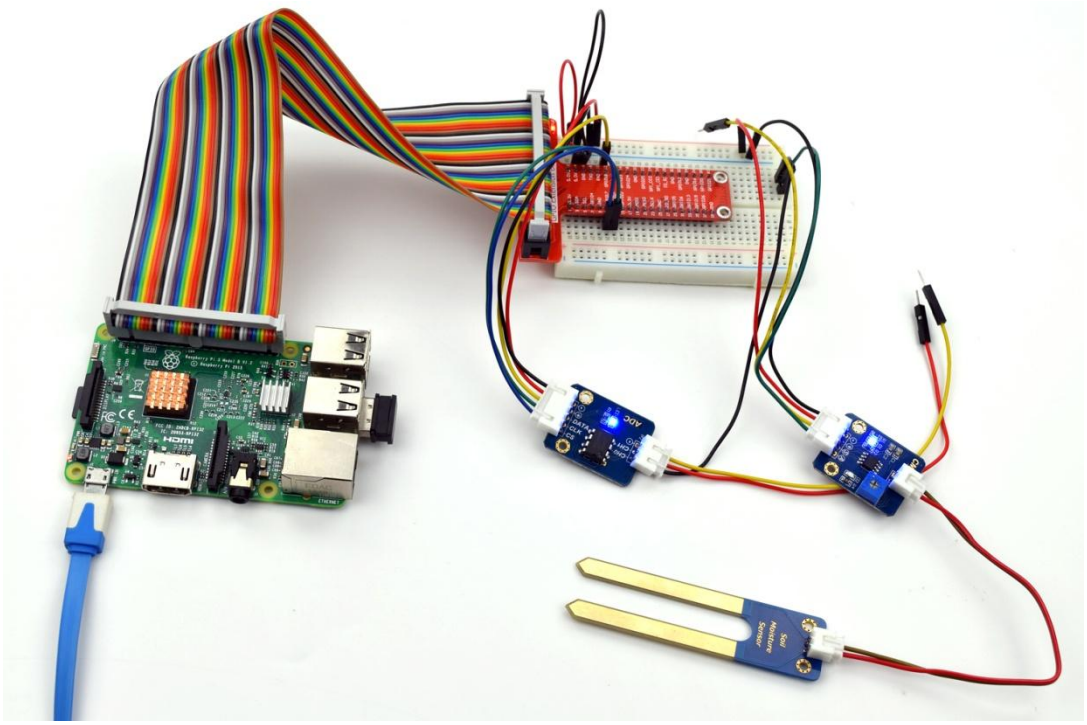
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/31_soilMoisture.py)

Step 3: Run

```
$ sudo python 31_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 32 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.

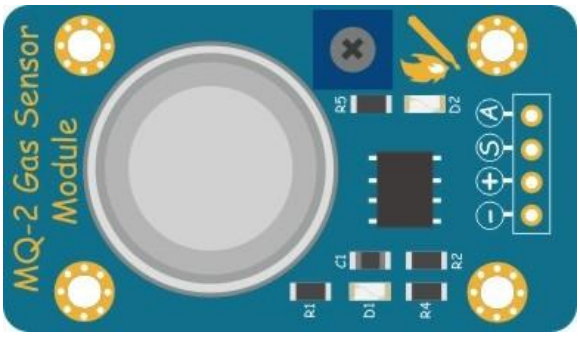
2. The sensor may not output a steady and accurate data immediately; it needs to be preheated for about 1 minute to collect data steadily.

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:

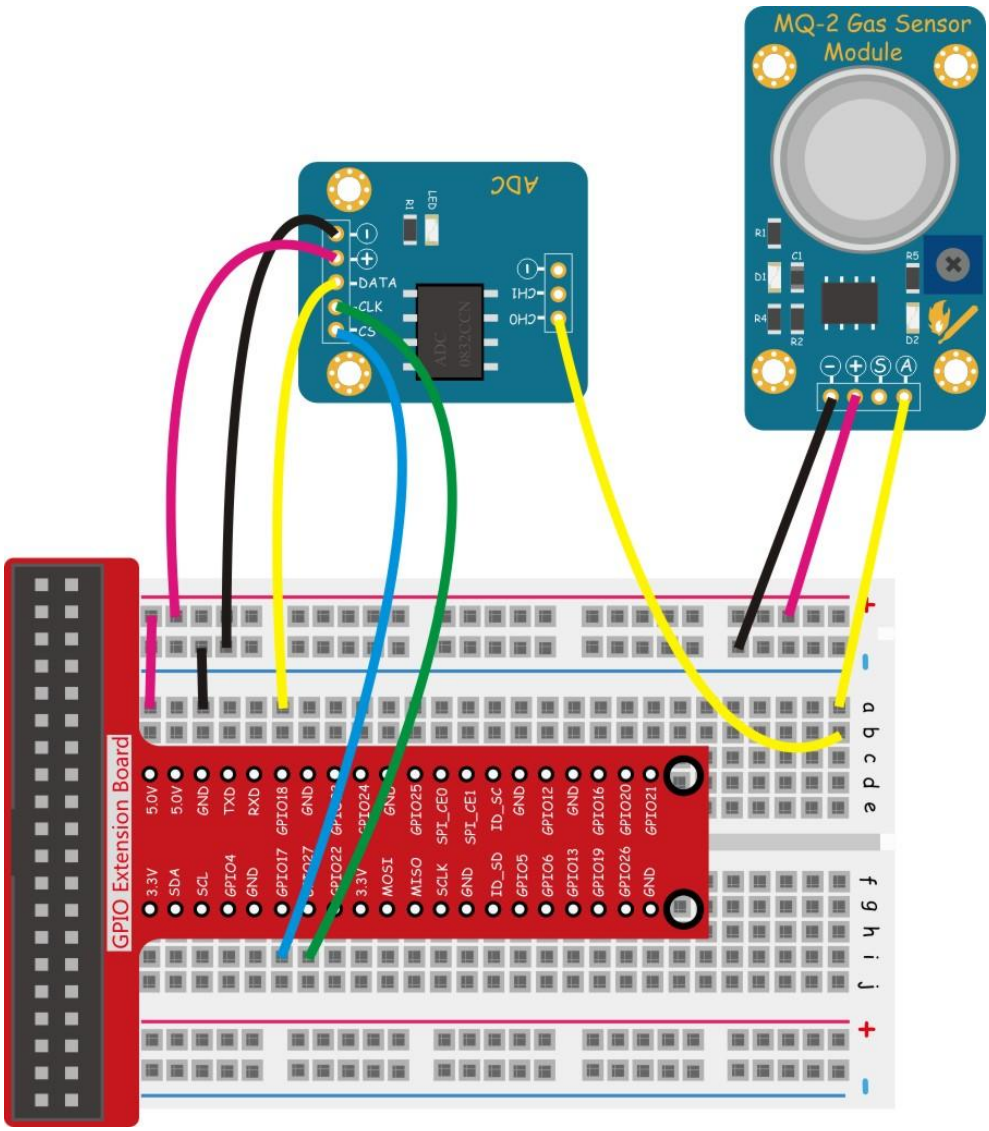
Pin definition:

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/Adeept_Sensor_Kit_for_RPi_C_Code/32_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:

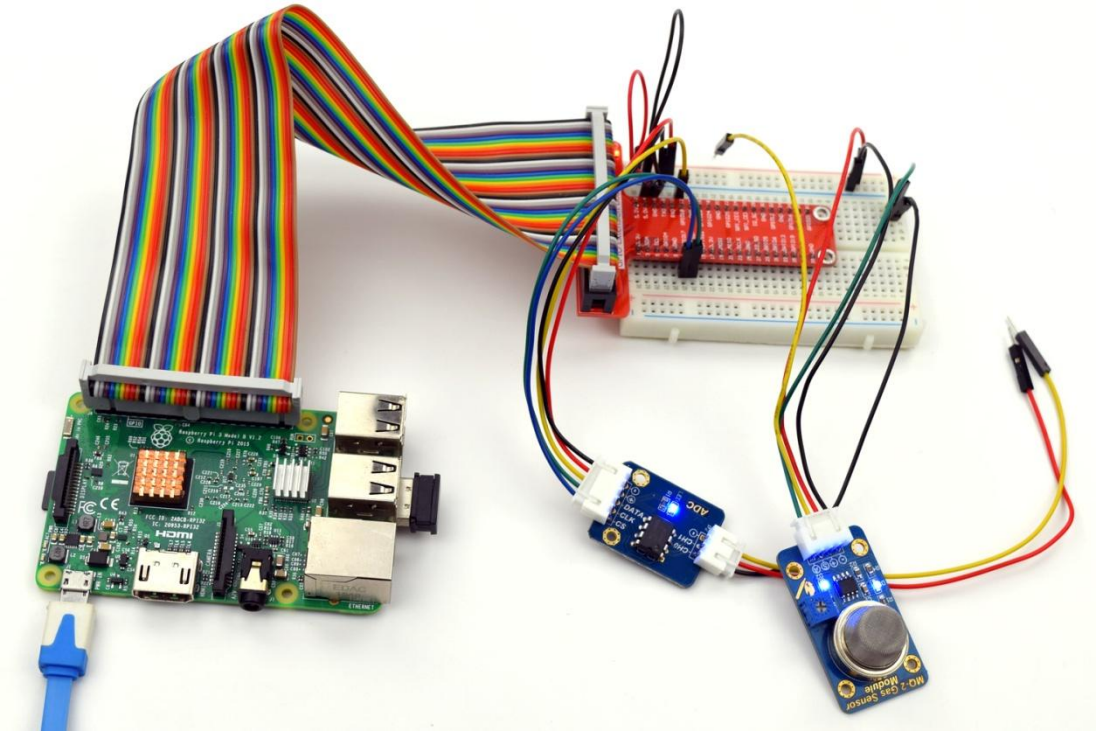
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/32_mq-2.py)

Step 3: Run

```
$ sudo python 32_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 33 Sound Sensor

Introduction

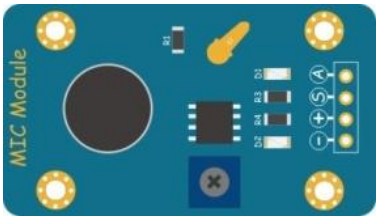
The MIC Module is composed of a small microphone and an LM393 voltage comparator. It can capture minor sound signals and convert them into electric ones. The threshold of the comparator can be adjusted by the blue potentiometer on the module. This module can be applied in sound alarm system.

Components

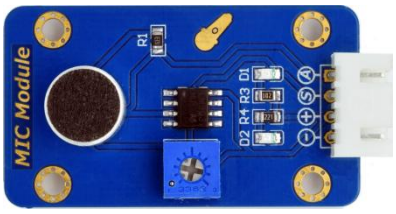
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * MIC Module
- 1 * ADC0832 Module
- 1 * 3-Pin Wires
- 1 * 4-Pin Wires
- 1 * 5-Pin Wires

Experimental Principle

The Fritzing image:



The Physical picture:



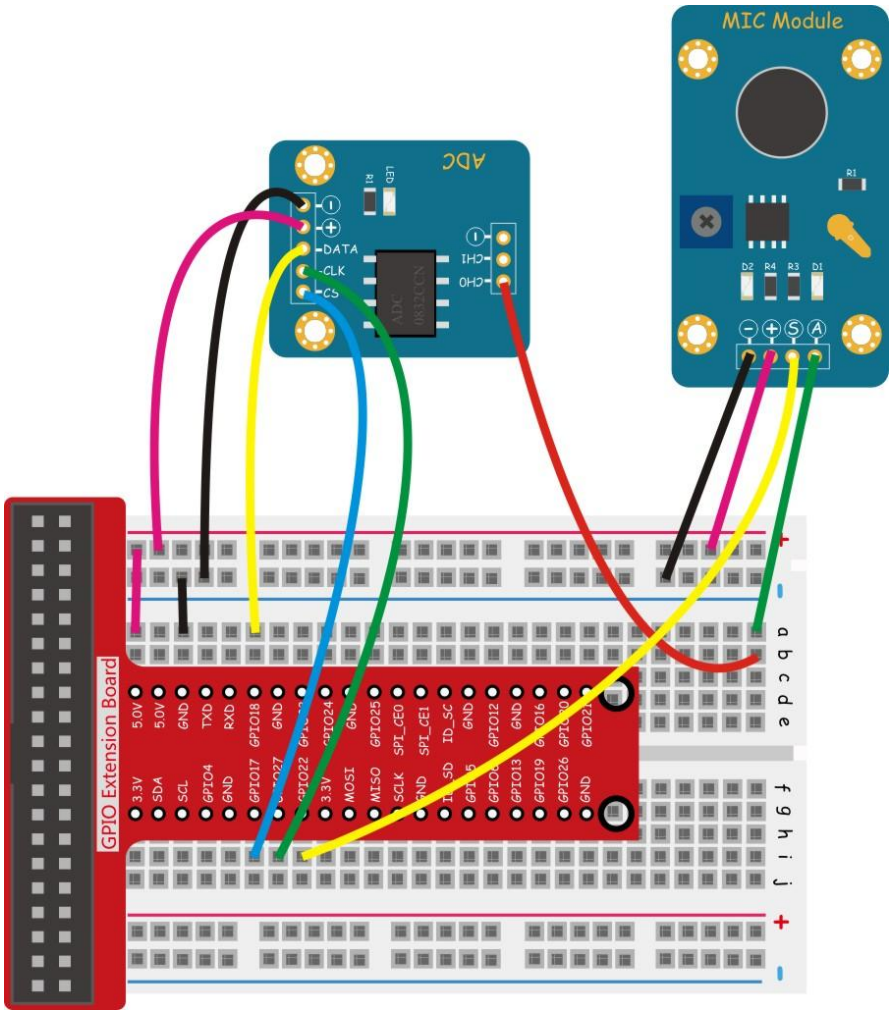
Pin definition:

S	Digital Output
A	Analog Output
+	VCC
-	GND

This experiment uses the MIC Module to detect the sound and display the data 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/Adeept_Sensor_Kit_for_RPi_C_Code/33_mic/mic.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./mic
```

For Python users:

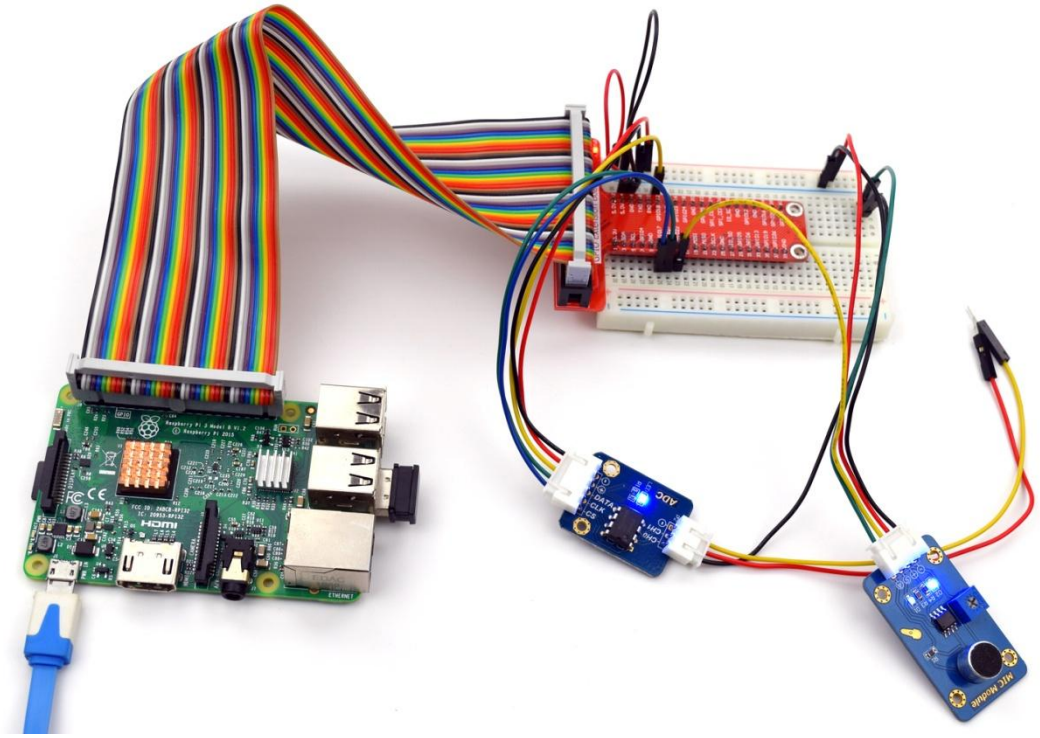
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/33_mic.py)

Step 3: Run

```
$ sudo python 33_mic.py
```

Blow at the MIC Module or make some other sounds near it, and you can see the value on the terminal indicating the sound intensity. The higher the sound volume, the larger value on the terminal; the lower volume, the smaller value.



Lesson 34 PS2 Joystick

Introduction

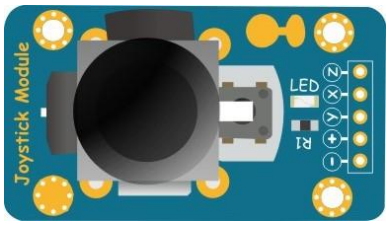
The PS2 Joystick Module is an input device. It consists of a station and the control knob onside. It functions by sending angle or direction signals to the device controlled. The button on the module can also be recognized by the microcontroller. The module supports two-channel analog output, namely, 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:

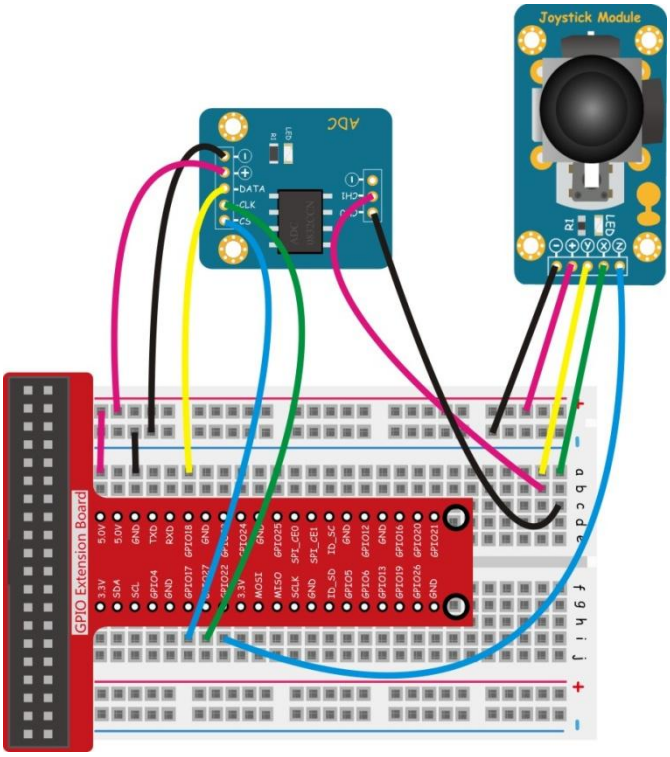
z	Switch Output
x	Analog Output(X)
y	Analog Output(Y)

+	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/Adeept_Sensor_Kit_for_RPi_C_Code/34_PS2Joystick/joystick.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./joystick
```

For Python users:

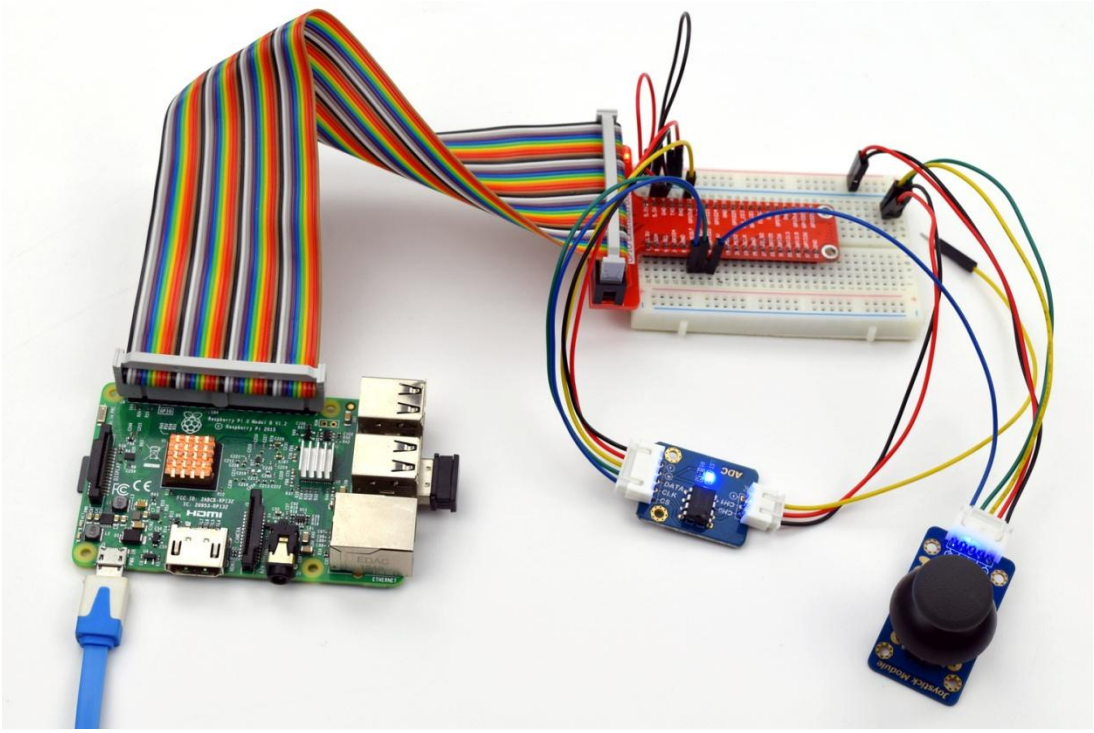
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/34_PS2Joystick.py)

Step 3: Run

```
$ sudo python 34_PS2Joystick.py
```

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



Lesson 35 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:

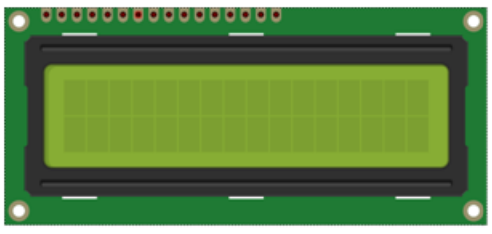
Since there are limited I/O ports on the Raspberry Pi, if you use them to drive the LCD1602, it needs many of the ports and there may be insufficient to connect other devices. To solve this problem, an IIC (or I2C) Interface Module based on PCF8574 is designed to extend the I/O ports of the Raspberry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

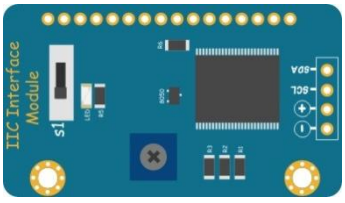
Components

- 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

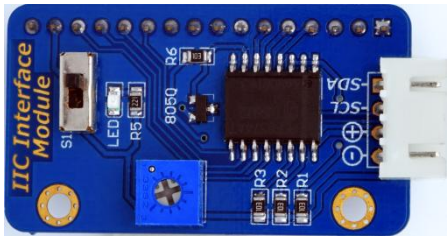
Experimental Principle

The Fritzing image:





The Physical picture:

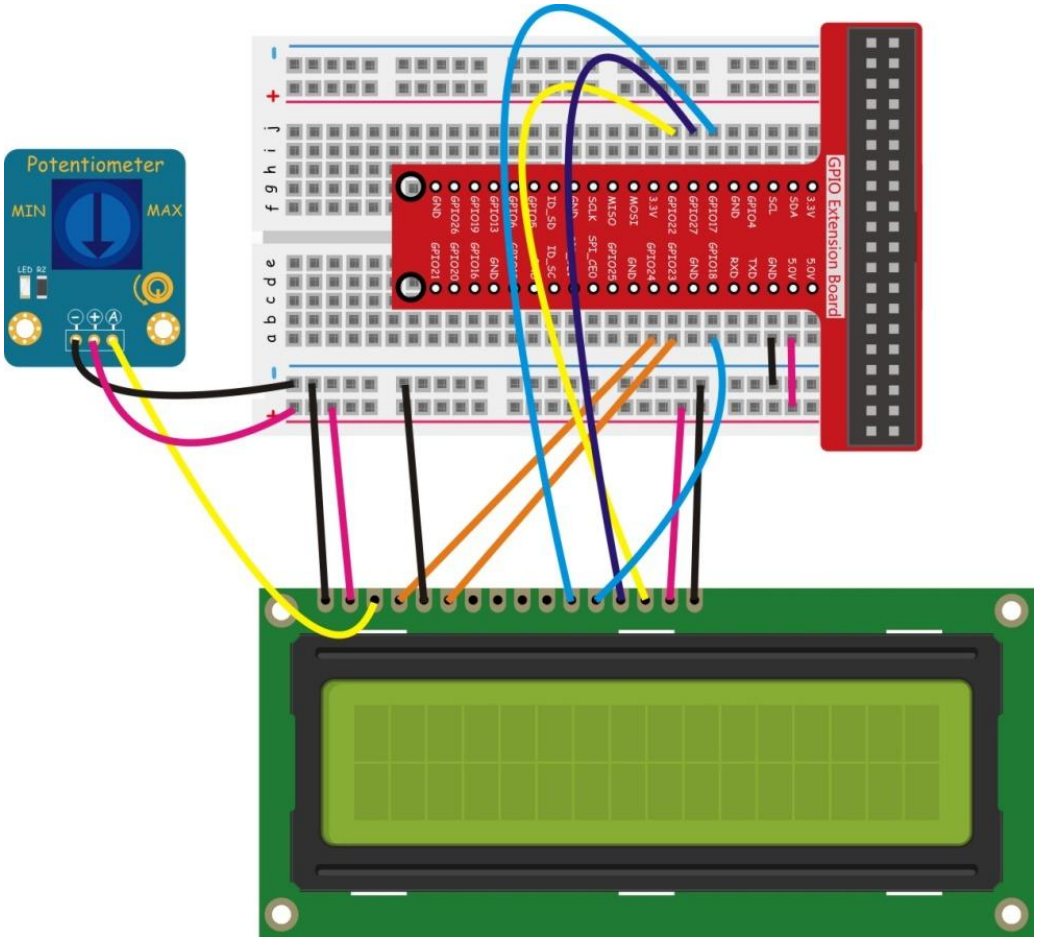


This lesson contains two experiments. The first experiment is controlling LCD1602 directly by the Raspberry Pi's GPIO, the second experiment is controlling via PCF8574-based IIC module.

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/Adeept_Sensor_Kit_for_RPi_C_Code/35_LCD1602/lcd1602.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./lcd1602
```

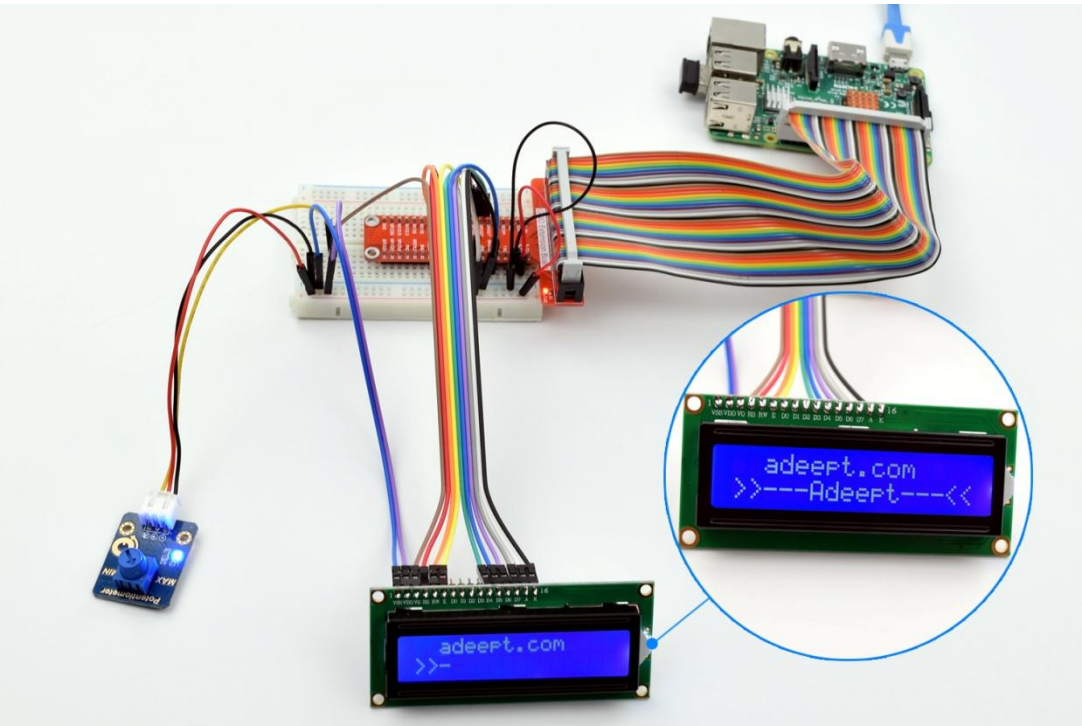
For Python users:

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

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/35_LCD1602/lcd1602.py)

Step 3: Run

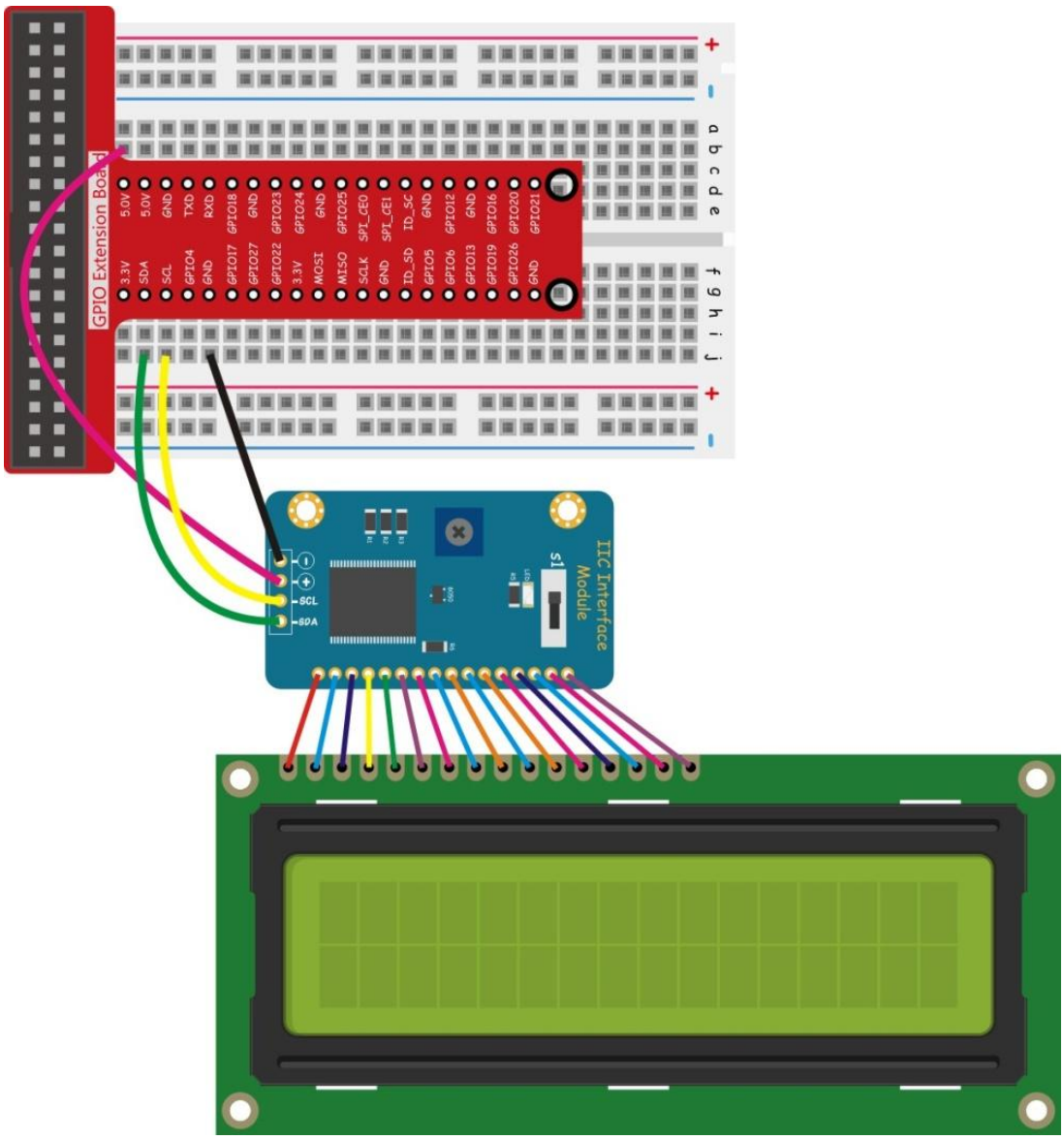
```
$ sudo python lcd1602.py
```



Experiment 2:

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_Sensor_Kit_for_RPi_C_Code/35_LCD1602/i2c_lcd1602.c)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./i2c_lcd1602
```

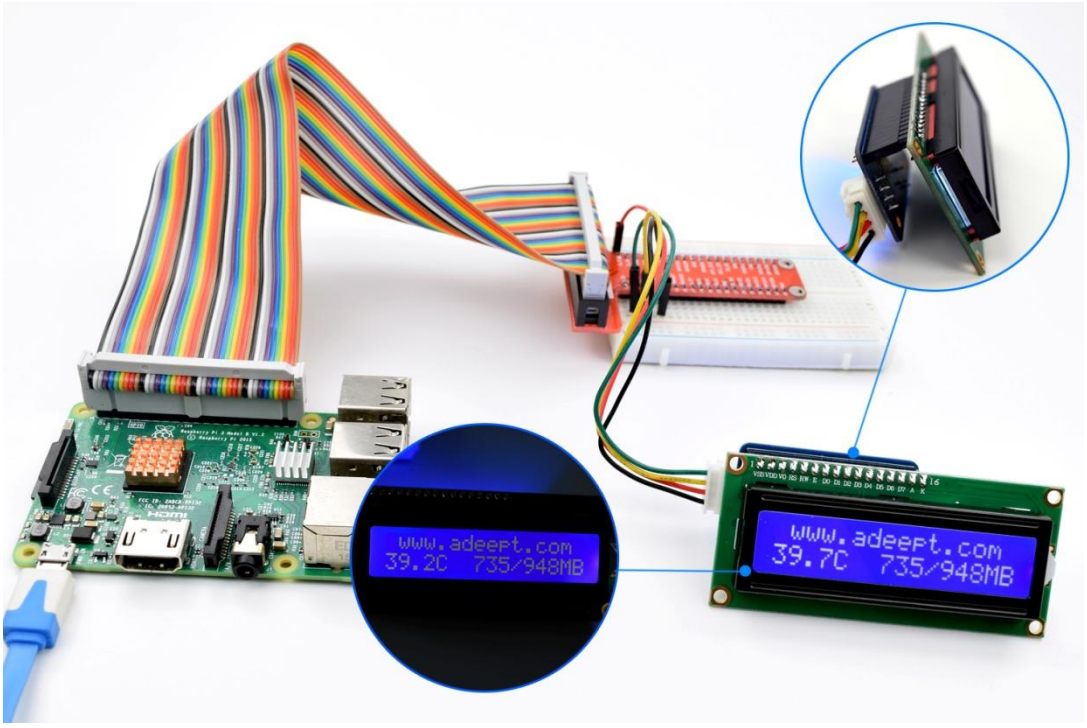
For Python users:

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

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/35_LCD1602/i2c_lcd1602.py)

Step 3: 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 36 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

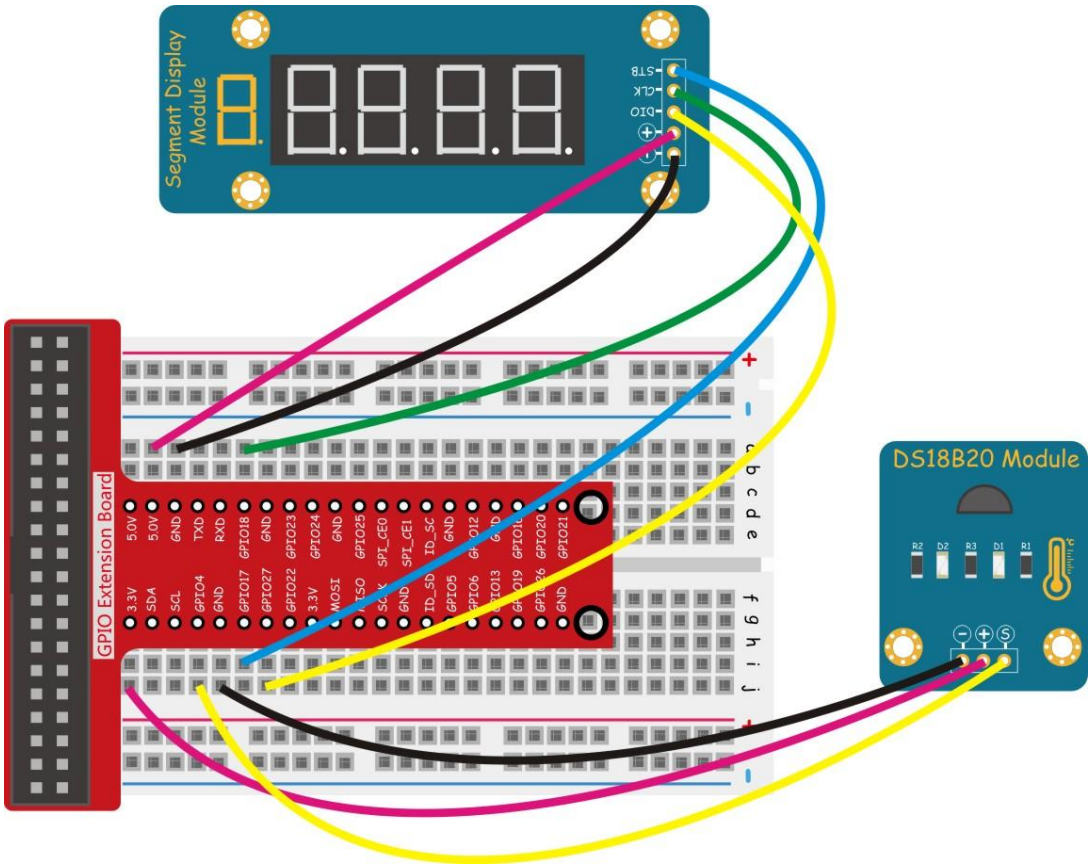
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * DS18b20 Module
- 1 * Segment Display Module
- 1 * 5-Pin Wires
- 1 * 3-Pin Wires

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_Sensor_Kit_for_RPi_C_Code/36_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_Sensor_Kit_for_RPi_Python_Code/36_thermometer_1/)

Step 3: Run

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

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

Lesson 37 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

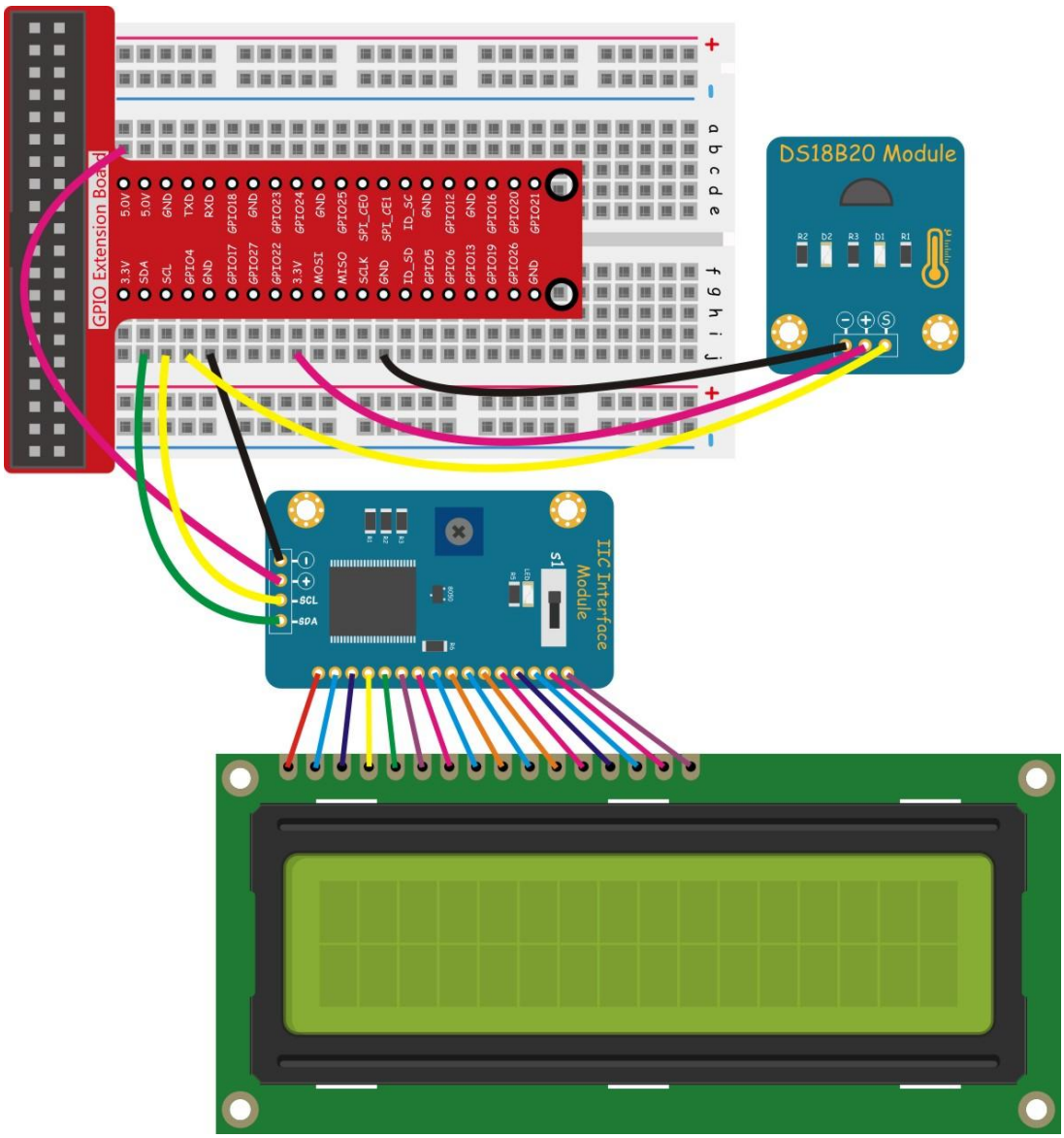
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * DS18b20 Module
- 1 * I2C Interface Module
- 1 * LCD1602
- 1 * 4-Pin Wires
- 1 * 3-Pin Wires

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/Adeept_Sensor_Kit_for_RPi_C_Code/37_thermometer_2)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./main
```

For Python users:

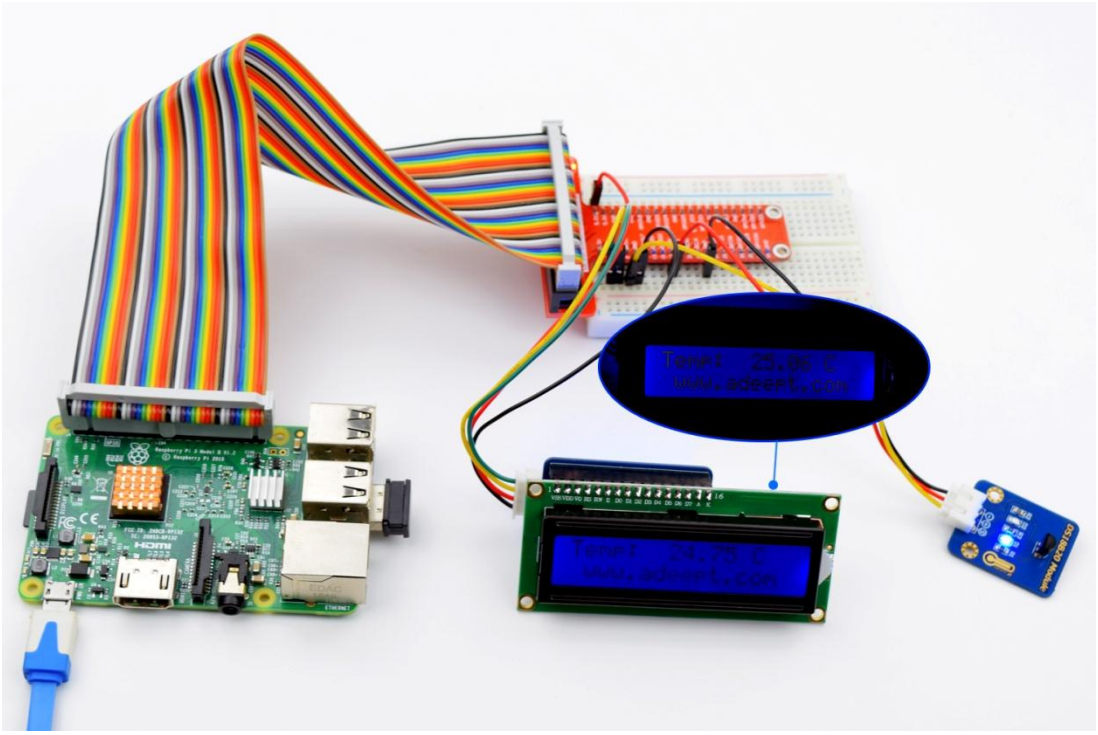
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/37_thermometer_2)

Step 3: Run

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

Now you can see the temperature shown on the LCD1602.



Lesson 38 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

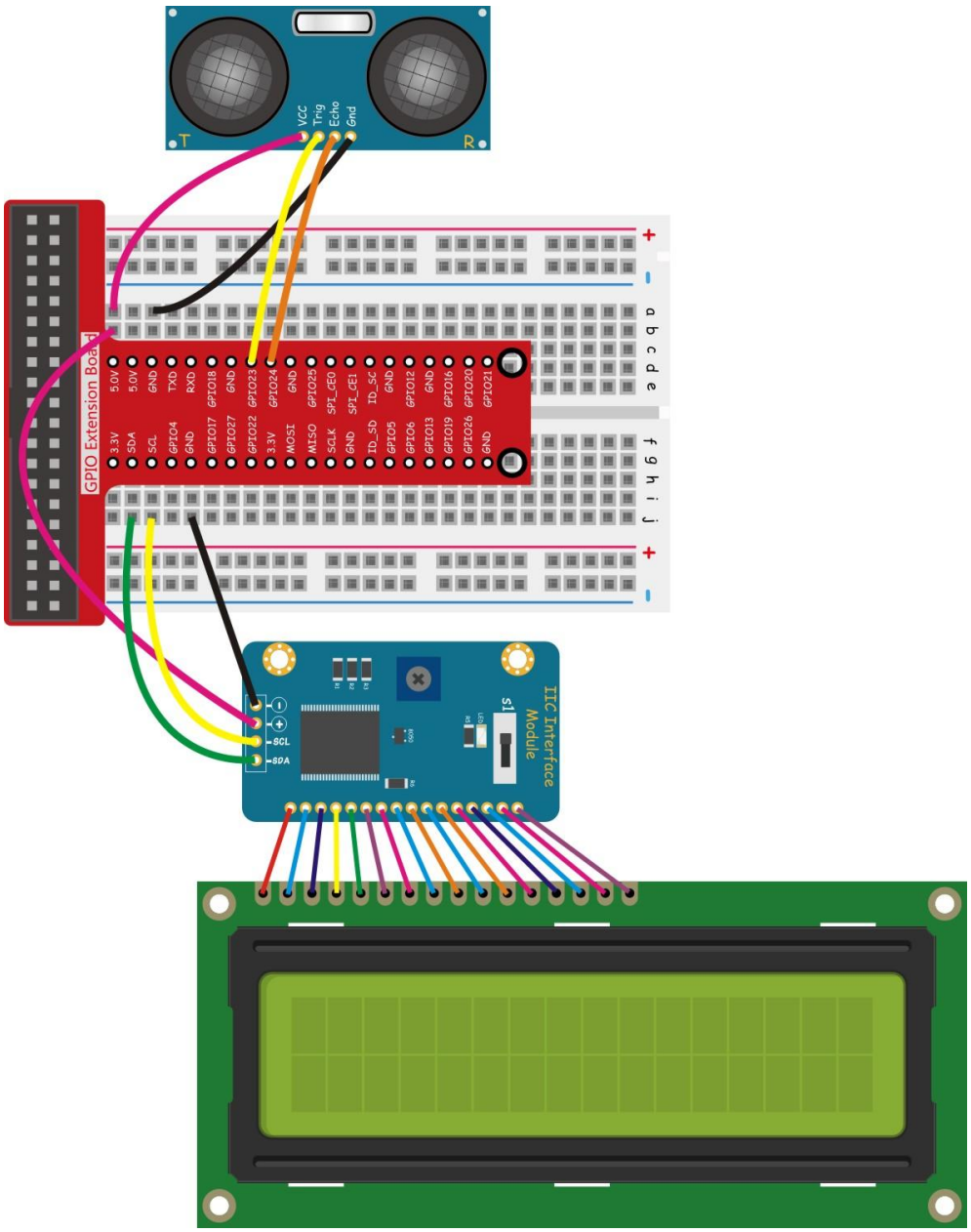
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * LCD1602
- 1 * I2C Interface Module
- 1 * Ultrasonic Sensor Module
- 1 * 4-Pin Wires
- 4 * Jumper Wires

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/Adeept_Sensor_Kit_for_RPi_C_Code/38_measureDis)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./main
```

For Python users:

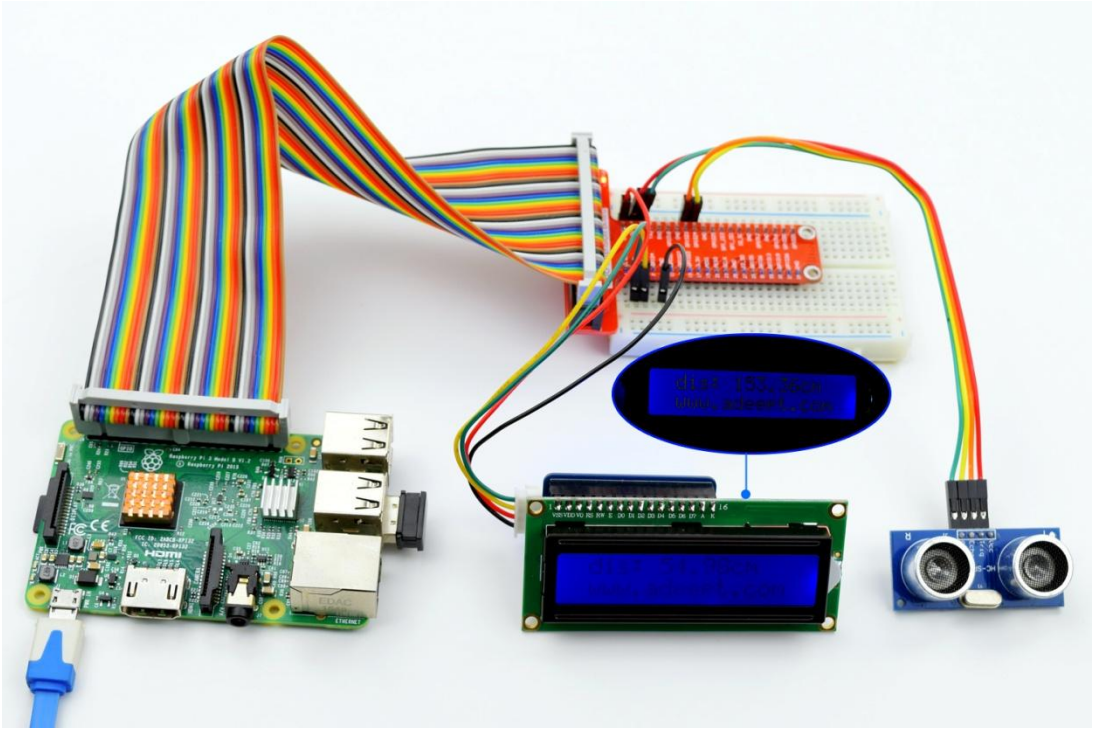
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/38_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 39 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

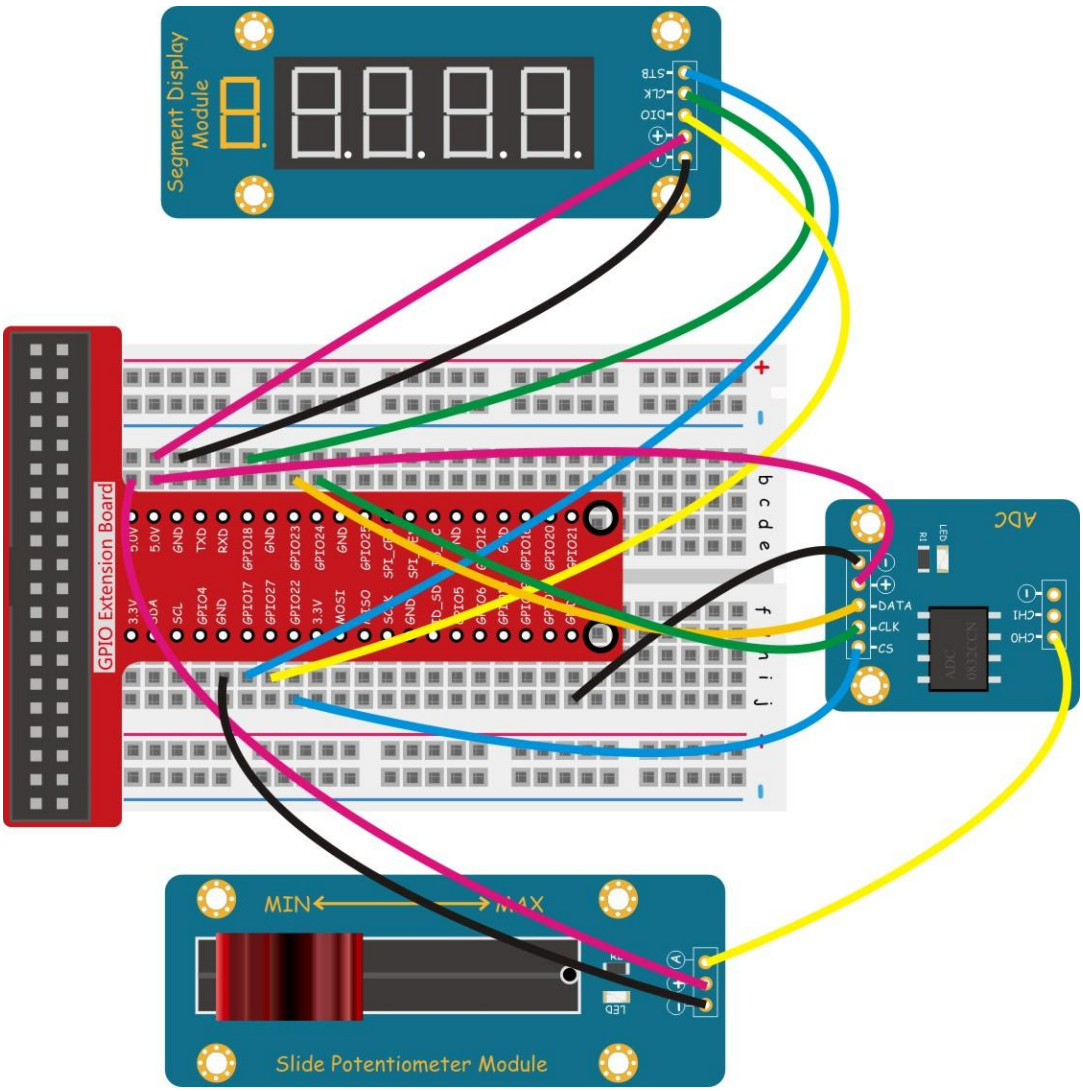
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * ADC0832 Module
- 1 * Segment Display Module
- 1 * Slide Potentiometer Module
- 2 * 5-Pin Wires
- 2 * 3-Pin Wires

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/Adeept_Sensor_Kit_for_RPi_C_Code/39_voltmeter_1)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./main
```

For Python users:

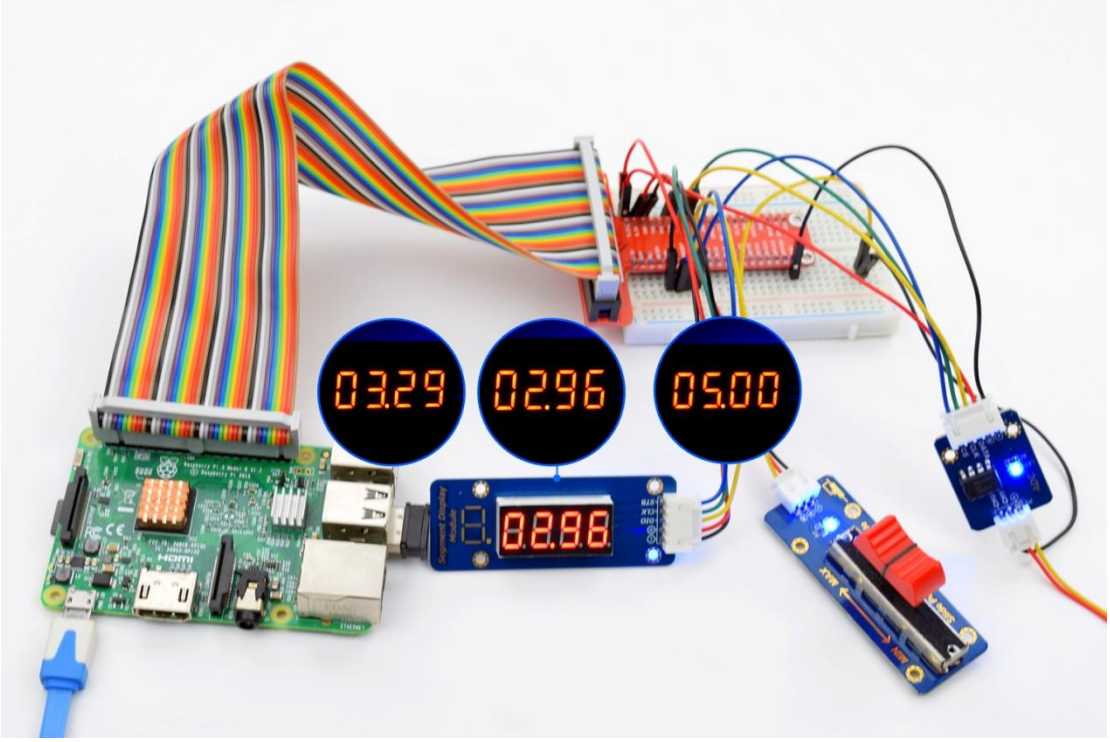
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/39_voltmeter_1)

Step 3: Run

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

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



Lesson 40 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

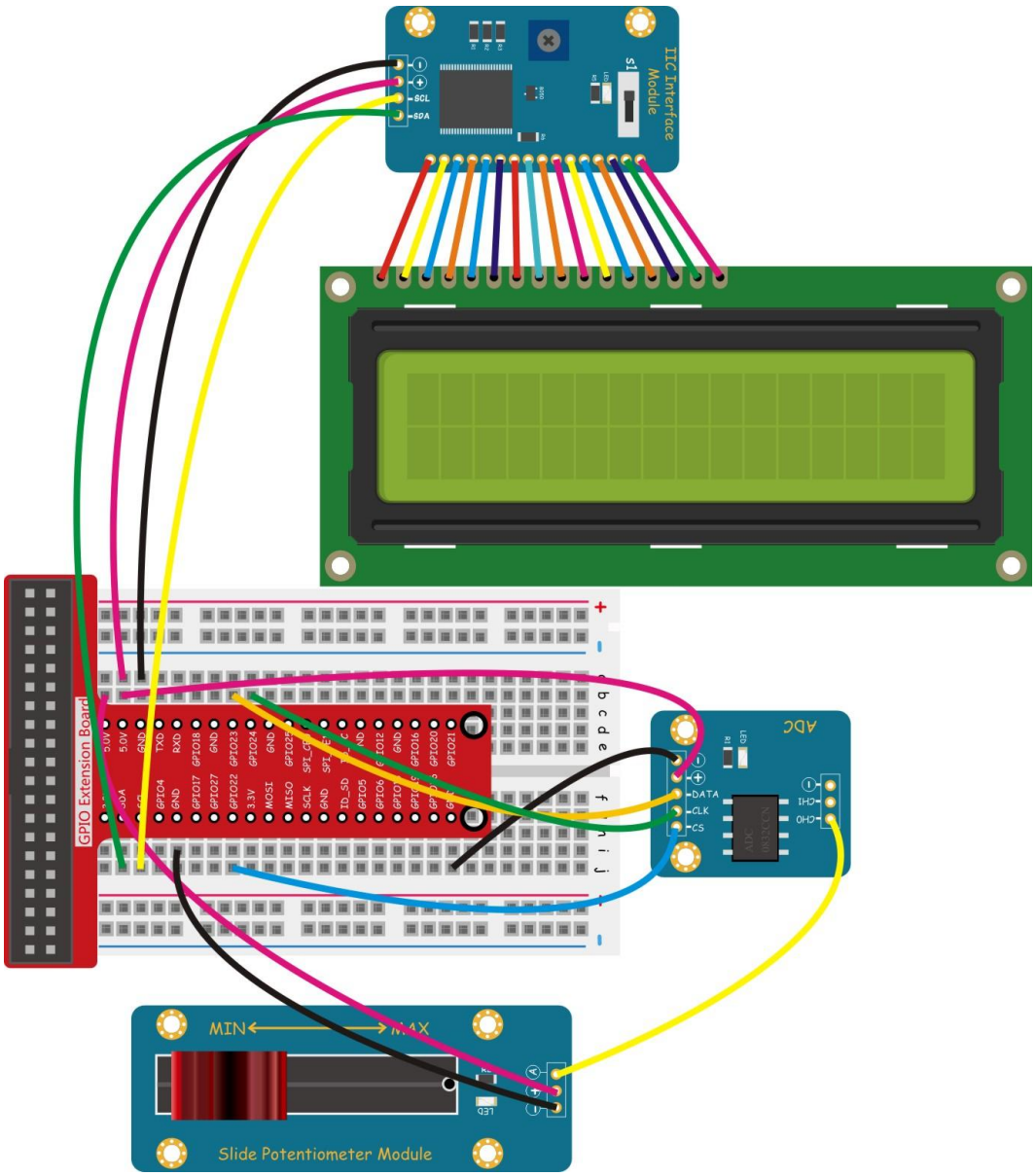
- 1 * Raspberry Pi
- 1 * GPIO Extension Board
- 1 * 40-Pin GPIO Cable
- 1 * Breadboard
- 1 * ADC0832 Module
- 1 * I2C Interface Module
- 1 * Slide Potentiometer Module
- 1 * LCD1602
- 2 * 3-Pin Wires
- 1 * 4-Pin Wires
- 1 * 5-Pin Wires

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/Adeept_Sensor_Kit_for_RPi_C_Code/40_voltmeter_2)

Step 3: Compile

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

Step 4: Run

```
$ sudo ./main
```

For Python users:

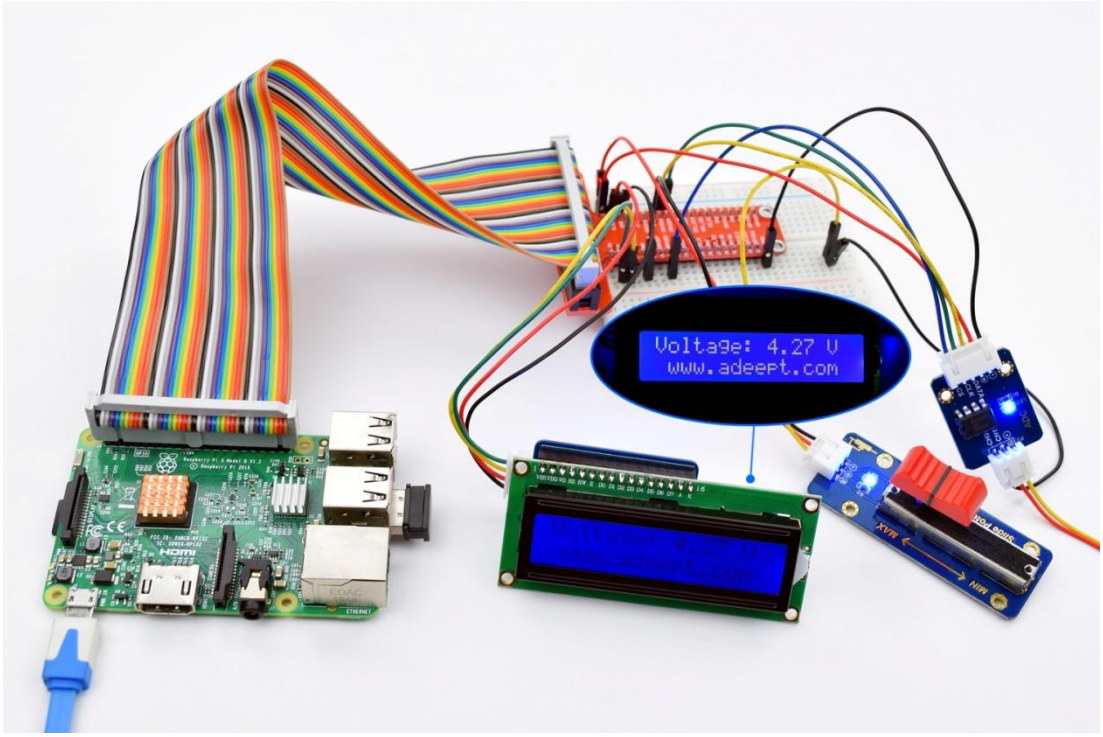
Step 2: Edit and save the code with vim or nano.

(code path: /home/Adeept_Sensor_Kit_for_RPi_Python_Code/40_voltmeter_2)

Step 3: Run

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

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




ULTIMATE SENSOR KIT
FOR RASPBERRY PI
Sharing Perfects Innovation
github
SOCIAL CODING
python
www.adeept.com
support@adeept.com