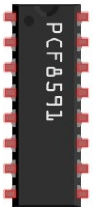
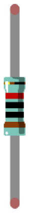
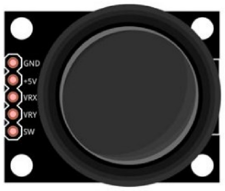
Chapter 12 Joystick

In the previous chapter, we have learned how to use rotary potentiometer. Now, let's learn a new electronic module Joystick which working on the same principle as rotary potentiometer.

Project 12.1 Joystick

In this project, we will read the output data of Joystick and print it to the screen.

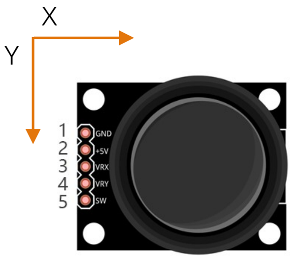
Component List

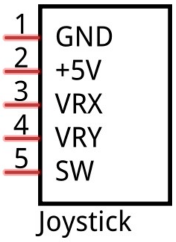
<table>
<tr><td colspan="2">Raspberry Pi 3B x1
GPIO Expansion Board & Wire x1
BreadBoard x1</td><td colspan="2">Jumper
 </td></tr>
<tr><td>PCF8591 x1
 </td><td colspan="2">Resistor 10kΩ x2
</td><td>Joystick x1
 </td></tr>
</table>

Component knowledge

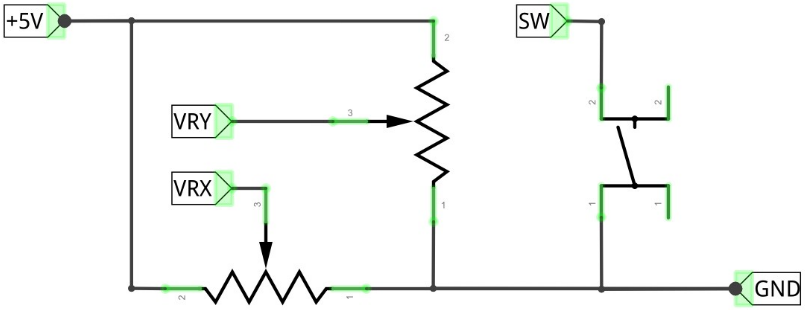
Joystick

Joystick is a kind of sensor used with your fingers, which is widely used in gamepad and remote controller. It can shift in direction Y or direction X at the same time. And it can also be pressed in direction Z.





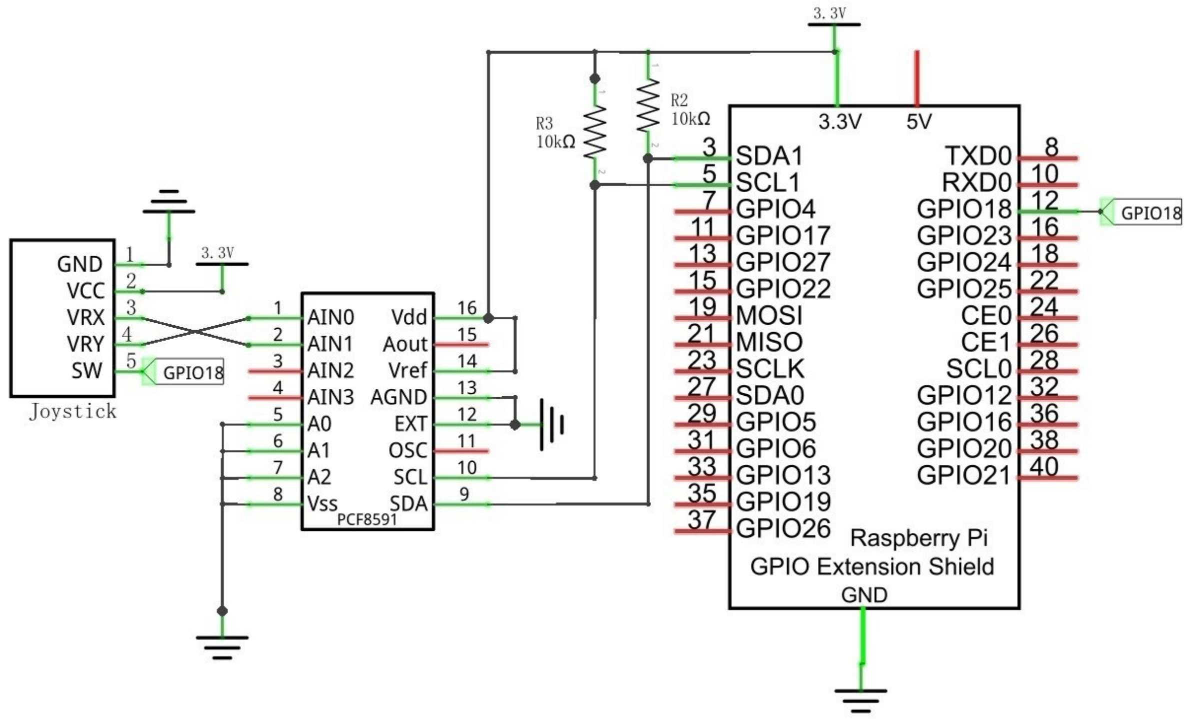
Two rotary potentiometers inside the joystick are set to detect the shift direction of finger, and a push button in vertical direction is set to detect the action of pressing.



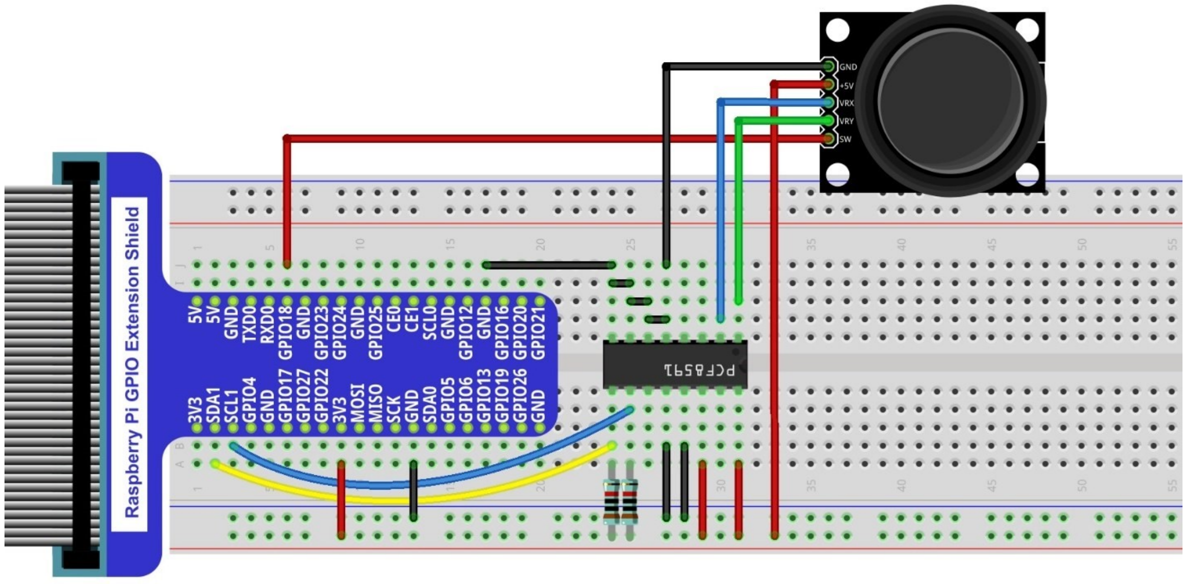
When read the data of joystick, there are some different between axis: data of X and Y axis is analog, which need to use ADC. Data of Z axis is digital, so you can directly use the GPIO to read, or you can also use ADC to read.

Circuit

Schematic diagram



Hardware connection. If you need any support, pleasefeel free to contact us via: support@freenove.com



Code

In this project code, we will read ADC value of X and Y axis of Joystick, and read digital quality of Z axis, then print these data out.

C Code 12.1.1 Joystick

First observe the project result, and then analyze the code.

1. Use cd command to enter 12.1.1_Joystick directory of C code.

```
cd ~/Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi/Code/C_Code/12.1.1_ Joystick
```

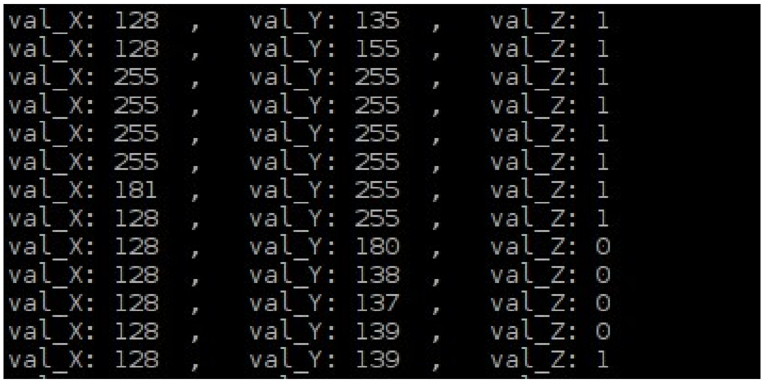
2. Use following command to compile "Joystick.c" and generate executable file "Joystick.c". "-lm" option is needed.

```
gcc Joystick.c -o Joystick -lwiringPi -lm
```

3. Then run the generated file "Joystick".

```
sudo ./Joystick
```

After Program is executed, the terminal window will print out the data of 3 axes X, Y, Z. And shifting the Joystick or pressing it will make those data change.



```
val_X: 128  ,   val_Y: 135  ,   val_Z: 1
val_X: 128  ,   val_Y: 155  ,   val_Z: 1
val_X: 255  ,   val_Y: 255  ,   val_Z: 1
val_X: 255  ,   val_Y: 255  ,   val_Z: 1
val_X: 255  ,   val_Y: 255  ,   val_Z: 1
val_X: 255  ,   val_Y: 255  ,   val_Z: 1
val_X: 181  ,   val_Y: 255  ,   val_Z: 1
val_X: 128  ,   val_Y: 255  ,   val_Z: 1
val_X: 128  ,   val_Y: 180  ,   val_Z: 0
val_X: 128  ,   val_Y: 138  ,   val_Z: 0
val_X: 128  ,   val_Y: 137  ,   val_Z: 0
val_X: 128  ,   val_Y: 139  ,   val_Z: 0
val_X: 128  ,   val_Y: 139  ,   val_Z: 1
```

The flowing is the code:

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

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

#define Z_Pin 1     //define pin for axis Z

int main(void){
    int val_X,val_Y,val_Z;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
```

```
        printf("setup wiringPi failed !");
        return 1;
    }
    pinMode(Z_Pin,INPUT);       //set Z_Pin as input pin and pull-up mode
    pullUpDnControl(Z_Pin,PUD_UP);
    pcf8591Setup(pinbase,address);      //initialize PCF8591

    while(1){
        val_Z = digitalRead(Z_Pin);  //read digital quality of axis Z
        val_Y = analogRead(A0);      //read analog quality of axis X and Y
        val_X = analogRead(A1);
        printf("val_X: %d  ,\tval_Y: %d  ,\tval_Z: %d \n",val_X,val_Y,val_Z);
        delay(100);
    }
    return 0;
}
```

In the code, configure Z_Pin to pull-up input mode. In while cycle of main function, use **analogRead** () to read the value of axis X and Y and use **digitalRead** () to read the value of axis Z, then print them out.

```
    while(1){
        val_Z = digitalRead(Z_Pin);  //read digital quality of axis Z
        val_Y = analogRead(A0);      //read analog quality of axis X and Y

        val_X = analogRead(A1);
        printf("val_X: %d  ,\tval_Y: %d  ,\tval_Z: %d \n",val_X,val_Y,val_Z);
        delay(100);
    }
```

Python Code 12.1.1 Joystick

First observe the project result, and then analyze the code.

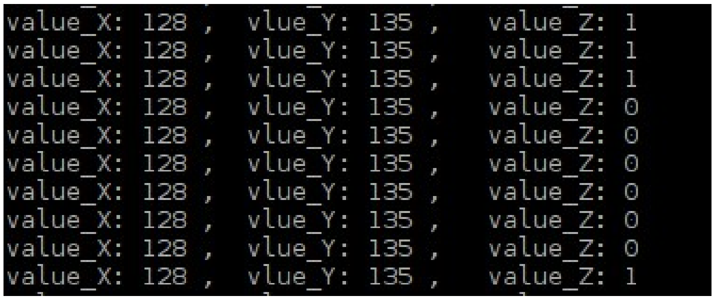
1. Use cd command to enter 12.1.1_Joystick directory of Python code.

```
cd ~/Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi/Code/Python_Code/12.1.1_ Joystick
```

2. Use python command to execute python code "Joystick.py".

```
python Joystick.py
```

After Program is executed, the terminal window will print out the data of 3 axes X, Y, Z. And shifting the Joystick or pressing it will make those data change.



```
value_X: 128 ,  vlue_Y: 135 ,   value_Z: 1
value_X: 128 ,  vlue_Y: 135 ,   value_Z: 1
value_X: 128 ,  vlue_Y: 135 ,   value_Z: 1
value_X: 128 ,  vlue_Y: 135 ,   value_Z: 0
value_X: 128 ,  vlue_Y: 135 ,   value_Z: 0
value_X: 128 ,  vlue_Y: 135 ,   value_Z: 0
value_X: 128 ,  vlue_Y: 135 ,   value_Z: 0
value_X: 128 ,  vlue_Y: 135 ,   value_Z: 0
value_X: 128 ,  vlue_Y: 135 ,   value_Z: 0
value_X: 128 ,  vlue_Y: 135 ,   value_Z: 1
```

The following is the program code:

```python
import RPi.GPIO as GPIO
import smbus
import time

address = 0x48
bus=smbus.SMBus(1)
cmd=0x40
Z_Pin = 12      #define pin for Z_Pin
def analogRead(chn):        #read ADC value
    bus.write_byte(address,cmd+chn)
    value = bus.read_byte(address)
    value = bus.read_byte(address)
    #value = bus.read_byte_data(address,cmd+chn)
    return value

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

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(Z_Pin,GPIO.IN,GPIO.PUD_UP)    #set Z_Pin to pull-up mode
def loop():
    while True:
        val_Z = GPIO.input(Z_Pin)       #read digital quality of axis Z
        val_Y = analogRead(0)           #read analog quality of axis X and Y
        val_X = analogRead(1)
        print ('value_X: %d ,\tvlue_Y: %d ,\tvalue_Z: %d'%(val_X,val_Y,val_Z))
        time.sleep(0.01)

def destroy():
    bus.close()
    GPIO.cleanup()

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

```

In the code, configure Z_Pin to pull-up input mode. In while cycle of loop, use **analogRead** () to read the value of axis X and Y and use **GPIO.input** () to read the value of axis Z, then print them out.

```
    while True:
        val_Z = GPIO.input(Z_Pin)       #read digital quality of axis Z
        val_Y = analogRead(0)           #read analog quality of axis X and Y
        val_X = analogRead(1)
        print ('value_X: %d ,\tvlue_Y: %d ,\tvalue_Z: %d'%(val_X,val_Y,val_Z))
        time.sleep(0.01)
```