Joystick

Overview

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

Experimental Materials:

Raspberry Pi *1

T-type expansion board *1

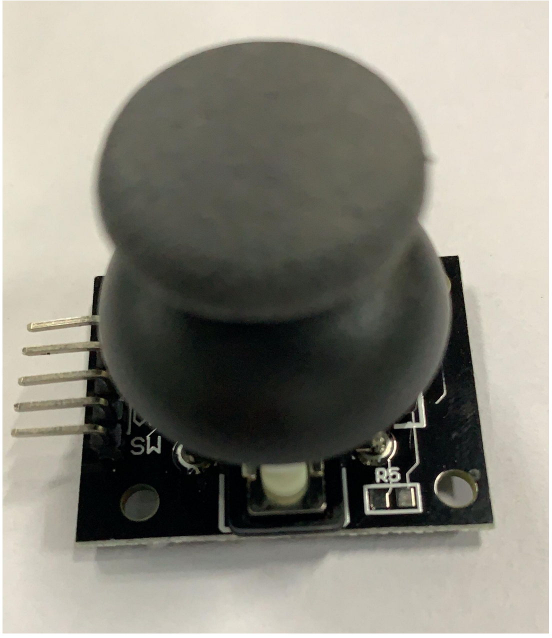
Breadboard*1

Joystick *1

PCF8591 *1

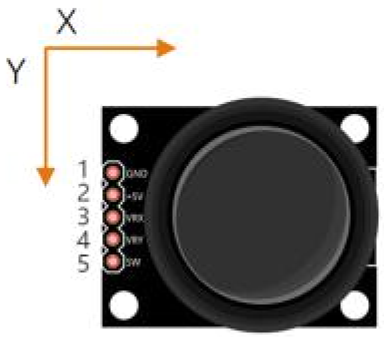
Some DuPont lines

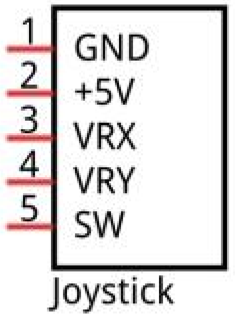
Product description：



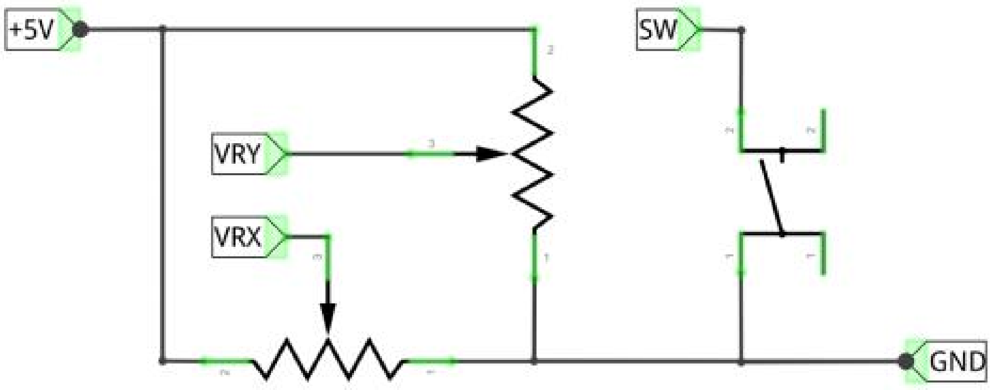
Joystick

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



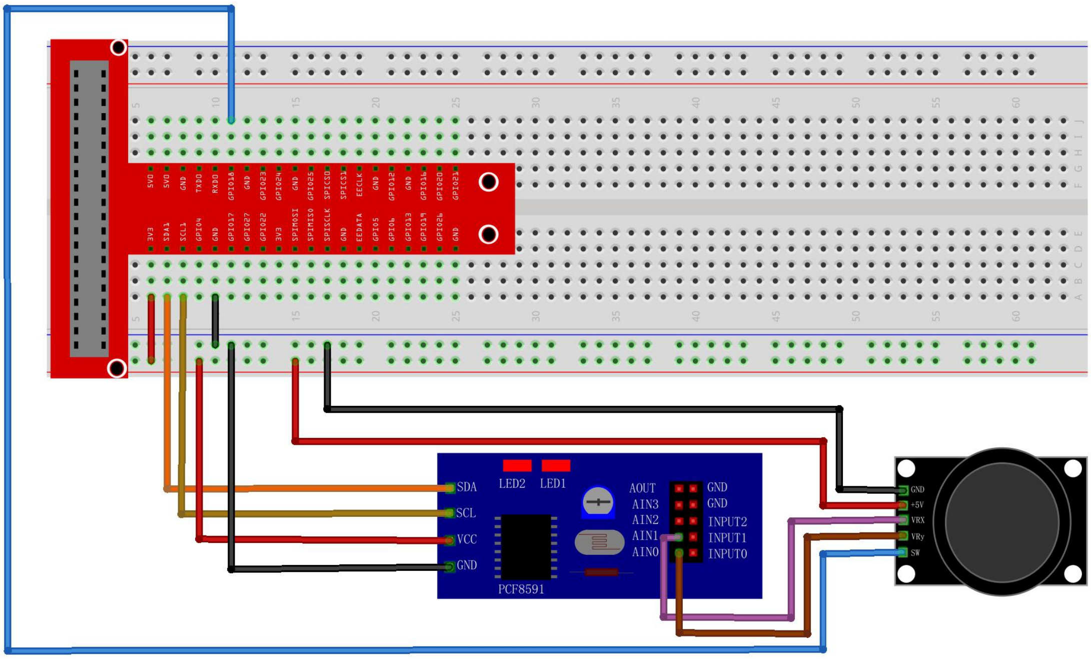


Two rotary potentiometers inside the joystick are set to detect the shift direction of finger, and a push button in vertical direction is set to detect the action of pressing.



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.

Wiring diagram:



C 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 messageto 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;
}
```

Python code：

```
#!/usr/bin/env python3
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():
    global p_Red,p_Green,p_Blue
    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()
```

Experimental results:

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

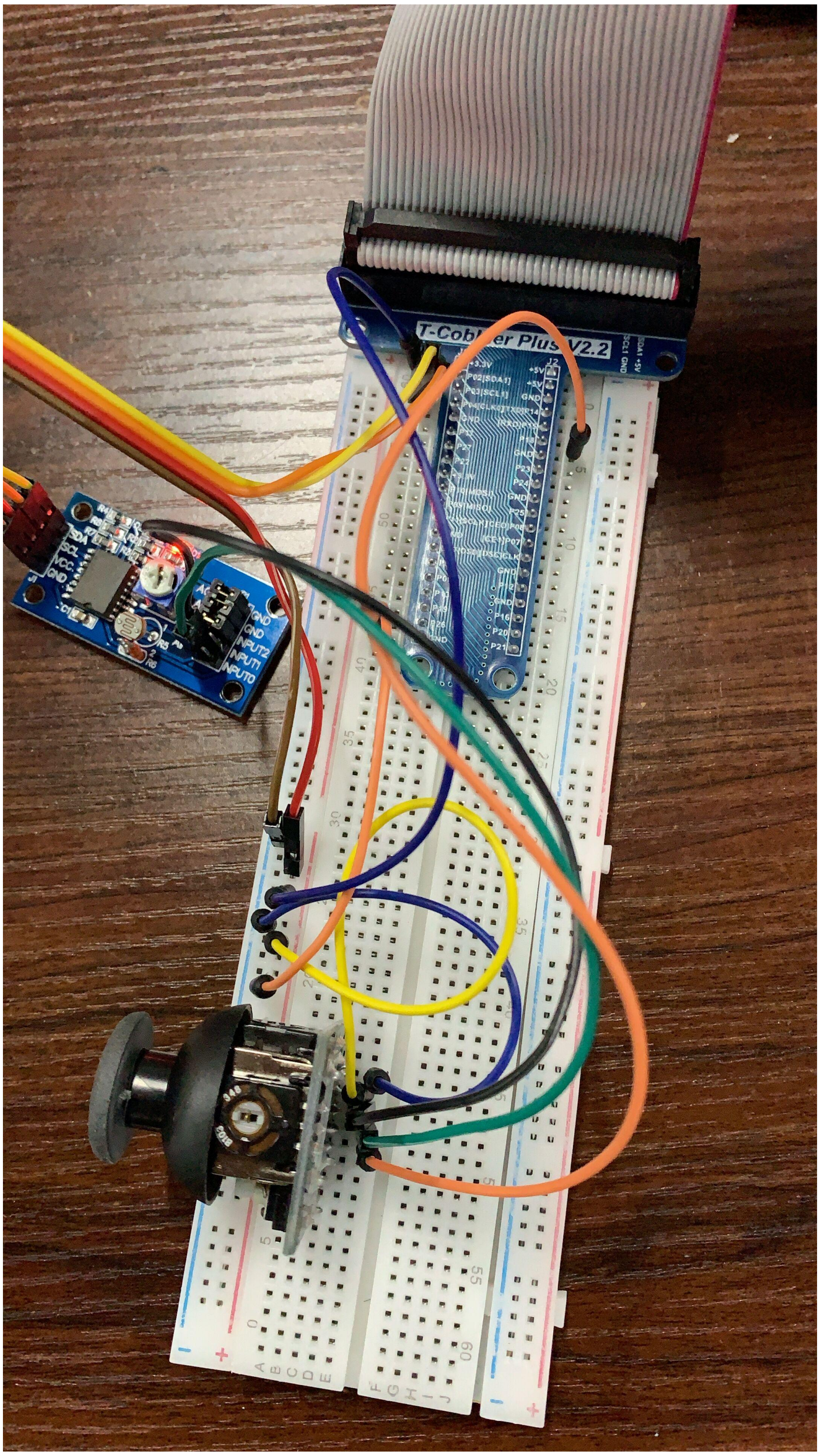
C：

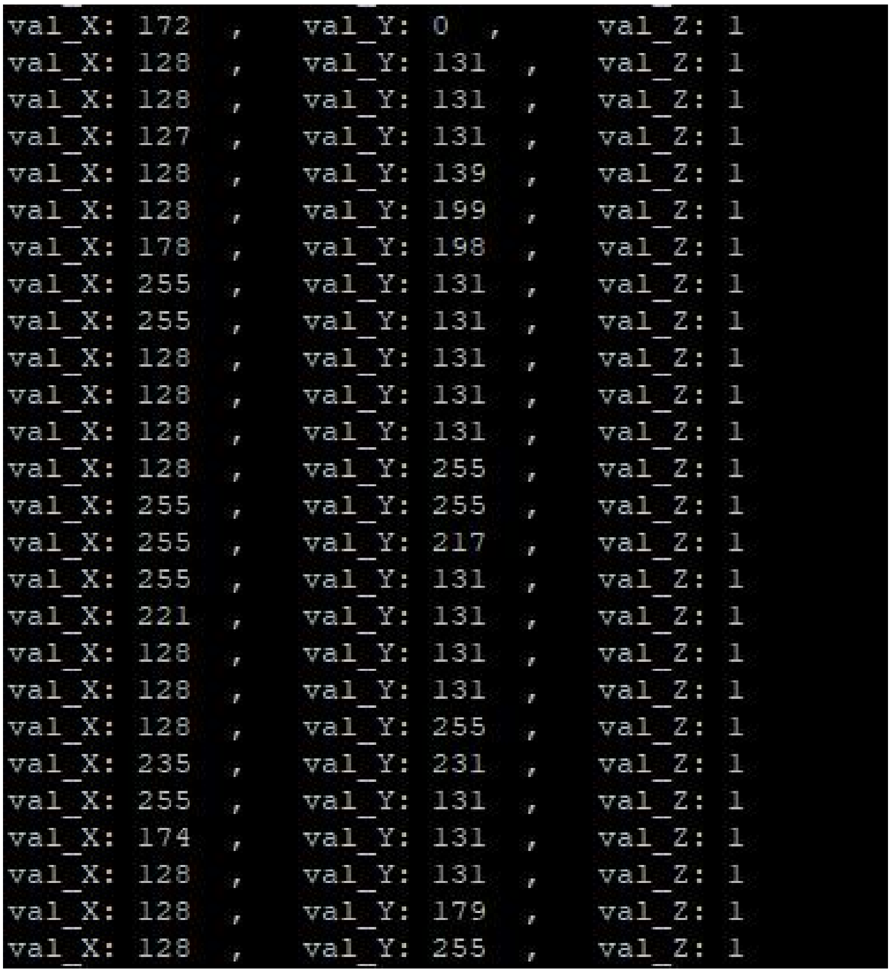
```
gcc -Wall -o Joystick Joystick.c -lwiringPi
sudo ./Joystick
```

Python:

```
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.


T-Cobbler Plus V2.2
+3.3V
P02(SDA1)
P03(SCL1)
+5V
GND
P23
P24
P25
GND
P12
GND
P16
P20
P21
SDA
SCL
VCC
GND
GND
INPUT2
INPUT1
INPUT0



```
val_X: 172  ,    val_Y: 0  ,     val_Z: 1
val_X: 128  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 131  ,   val_Z: 1
val_X: 127  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 139  ,   val_Z: 1
val_X: 128  ,    val_Y: 199  ,   val_Z: 1
val_X: 178  ,    val_Y: 198  ,   val_Z: 1
val_X: 255  ,    val_Y: 131  ,   val_Z: 1
val_X: 255  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 255  ,   val_Z: 1
val_X: 255  ,    val_Y: 255  ,   val_Z: 1
val_X: 255  ,    val_Y: 217  ,   val_Z: 1
val_X: 255  ,    val_Y: 131  ,   val_Z: 1
val_X: 221  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 255  ,   val_Z: 1
val_X: 235  ,    val_Y: 231  ,   val_Z: 1
val_X: 255  ,    val_Y: 131  ,   val_Z: 1
val_X: 174  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 131  ,   val_Z: 1
val_X: 128  ,    val_Y: 179  ,   val_Z: 1
val_X: 128  ,    val_Y: 255  ,   val_Z: 1
```

Note: Before using this tutorial, please open I2C and check the tutorial of PCF8591.