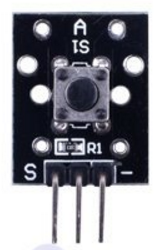
Button Switch  


Overview

Buttons are widely used controls in audio and video products, communications products, medical equipment, security products, toys, digital products, fitness equipment and other fields. In the button switch sensor, when the button is not pressed, voltage flows from the +5V input pin through a resistor to the output signal (S), which therefore reads high. Pressing the button momentarily switches the output to GND, which reads low. In this experiment, the Raspberry Pi monitors this output signal and switches an LED on and off with the button.

Experimental Materials

Raspberry Pi x1

Breadboard x1

Button switch x1

LED (3-pin) x1

Resistor(330Ω) x1

Dupont jumper wires

Experimental Procedure

1. If you have not done so already, prepare your development system by installing the Python interpreter, RPi.GIO library, and wiringPi library as described in READ\_ME.TXT.
2. Install the button and LED in your breadboard, and use Dupont jumper wires to connect them to your Raspberry Pi as illustrated in the Wiring Diagram below. (The two-color LED modules in this kit includes onboard series resistors, so no additional resistors are needed.)
3. Execute the sample stored in this experiment’s subfolder.

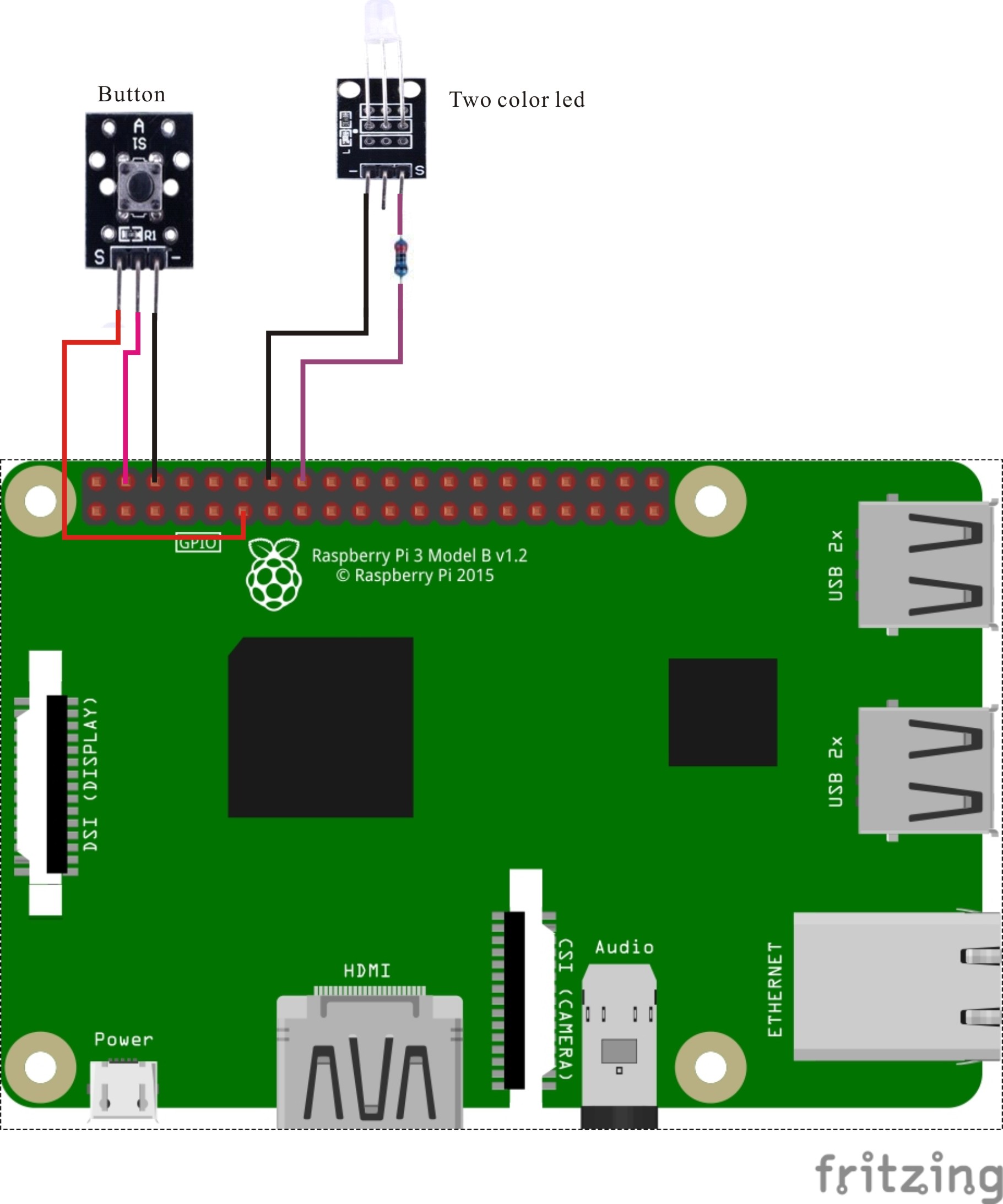
If using C, compile and execute the C code:  
cd Code/C  
gcc buttonSwitch.c -o buttonSwitch.out

–lwiringPi  
./buttonSwitch.out

If using Python, launch the Python script:  
cd Code/Python  
python buttonSwitch.py

1. Make experimental observations. Each time you press the button, the LED changes status.

Wiring Diagram



Button Switch pin position:

"S" ↔ Raspberry Pi pin 11

"-" ↔ Raspberry Pi GND

"+" ↔ Raspberry Pi +5V

LED pin position:

"S" ↔ Raspberry Pi pin 16

"-" ↔ Raspberry Pi GND

Sample code

Python code

#!/usr/bin/env python

import RPi.GPIO as GPIO

BtnPin = 11

LedPin = 16

Led\_status = 0

def setup():

GPIO.setmode(GPIO.BOARD) # Numbers GPIOs by physical location

GPIO.setup(LedPin, GPIO.OUT) # Set LedPin's mode is output

GPIO.setup(BtnPin, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

GPIO.output(LedPin, GPIO.LOW) # Set LedPin low to off led

def swLed(ev=None):

global Led\_status

Led\_status = not Led\_status

GPIO.output(LedPin, Led\_status)

print "LED: on " if Led\_status else "LED: off"

def loop():

GPIO.add\_event\_detect(BtnPin, GPIO.FALLING, callback=swLed, bouncetime=200) # wait for falling

while True:

pass # Don't do anything

def destroy():

GPIO.output(LedPin, GPIO.LOW) # led off

GPIO.cleanup() # Release resource

if \_\_name\_\_ == '\_\_main\_\_': # Program start from here

setup()

try:

loop()

except KeyboardInterrupt:

destroy()

C code

#include <wiringPi.h>

#include <stdio.h>

#define BtnPin 0

#define LedPin 4

void myBtnISR(void)

{

digitalWrite(LedPin, !digitalRead(LedPin));

printf("Button is pressed\n");

}

int main(void)

{

if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto screen

printf("setup wiringPi failed !");

return 1;

}

if(wiringPiISR(BtnPin, INT\_EDGE\_FALLING, myBtnISR)){

printf("setup ISR failed !");

return 1;

}

pinMode(LedPin, OUTPUT);

while(1);

return 0;

}

Characteristic Parameters

◆ Rated Range: 50mA 12VDC

◆ Contact resistance: 50mΩ max(initial)

◆ Insulation resistance: 100MΩ (DC250V)

◆ [Voltage](D:/%E7%BD%91%E6%98%93%E6%9C%89%E9%81%93%E8%AF%8D%E5%85%B8/Dict/8.0.0.0/resultui/html/index.html" \l "/javascript:;) [limit](D:/%E7%BD%91%E6%98%93%E6%9C%89%E9%81%93%E8%AF%8D%E5%85%B8/Dict/8.0.0.0/resultui/html/index.html" \l "/javascript:;): AC 250V(50/60Hz for 1 minute)

◆ Environmental temperature: -25°C~+105°C

◆ Heat distortion temperature: 250°C~280°C