www.kumantech.com

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 pull-up 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.GPIO library, and wiringPi library as described in READ_ME_FIRST.TXT.
2. Install the button and LED in your breadboard, and use resistors and Dupont jumper wires as illustrated in the Wiring Diagram below. Note you will connect only two of the three pins on the LED.
3. Execute the sample stored in this experiment's subfolder.
   If using C, compile and execute the C code:

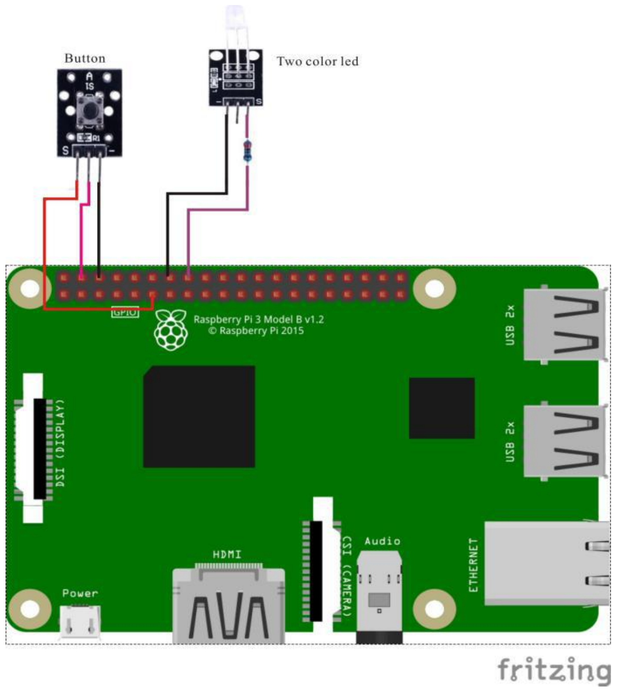
```
cd Code/C
gcc buttonSwitch.c -o buttonSwitch.out
 -lwiringPi
./buttonSwitch.out
```

If using Python, launch the Python script:

```
cd Code/Python
python buttonSwitch.py
```

4. 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 (through resistor)
"-"	↔	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 limit: AC 250V(50/60Hz for 1 minute)
- Environmental temperature: -25°C~+105°C
- Heat distortion temperature: 250°C~280°C