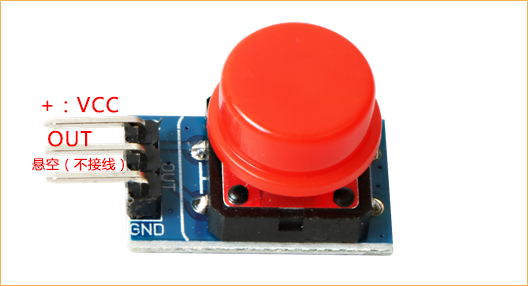
# Key Switch Experiment

## Introduction

The button is also called a light touch switch. In order to meet the operating force conditions when the use of pressure on the direction of the switch operation. If the switch is pressed, the circuit is connected; if the switch is released, the circuit is disconnected. Its internal structure relies on the force change of the metal shrapnel to achieve on-off. From the four-pin circuit diagram below, we can clearly find that under normal circumstances, the 1, 2, 3 and 4 pins of the key are connected; while we press the button, the four pins are connected to each other; while we release the button, the four pins are disconnected. The digital I/O can be used to achieve the effect of button control. The definition of the digital I/O port is the INPUT and OUTPUT interfaces. In the previous LED experiment, we only used the OUTPUT function of GPIO. Now we try to use the INPUT function of I/O in Arduino which means it needs to read the output value from an external device. We use a button and an LED to complete the experiment of INPUT and OUTPUT combination.

## Experimental Purpose

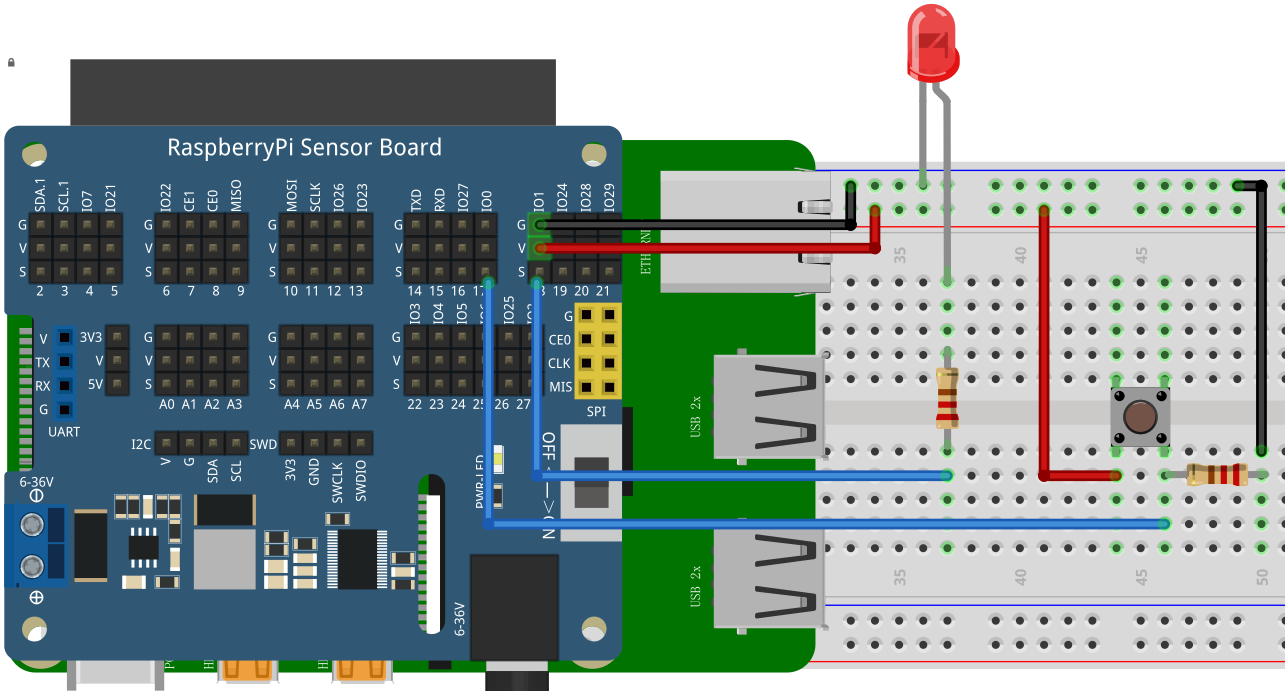
We connect the button to the IO0 interface and the red LED to the IO1 interface.

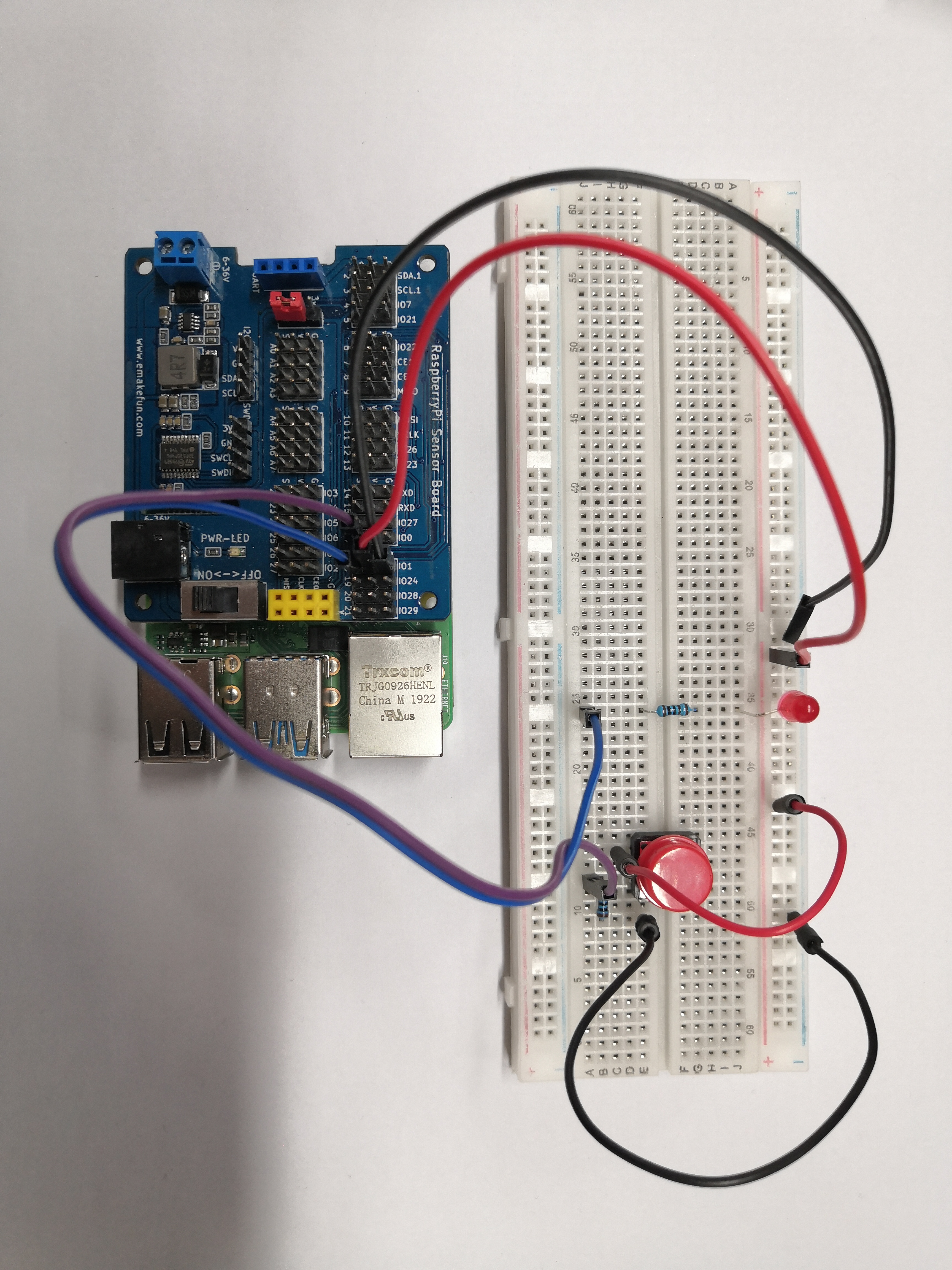
## Component List

* Raspberry Pi main board
* Breadboard
* USB Data cable
* Key Switch \* 1
* LED Module \* 1
* Several jumper wires

## Wiring

|  |  |  |
| --- | --- | --- |
| Raspberry Pi | Key Module Screen Printing | LED |
| IO0(wiringPi)/17(BCM) | OUT |  |
| IO1(wiringPi)/18(BCM) |  | VCC |
| GND | GND（can without） |  |
| +5V | VCC | GND |



****

## Experimental Purpose

Via analysis the circuit schematic diagram, we know that while the button is pressed, and the IO0 interface is low level voltage and the IO1 output pin is set to high level voltage, so that the light can be lit. While the button is released, and the IO0 interface is a high level voltage and the IO1 output pin is set to low level voltage, and then the light will go out at this time. The principle is the same as above.

## C++ program

|  |
| --- |
| #include <wiringPi.h>  #include <stdio.h>  #define BtnPin 0  #define LEDpin 1  int main**(**void**)**  **{**  **if(**wiringPiSetup**()** **==** **-**1**){** //when initialize wiring failed,print messageto screen  printf**(**"setup wiringPi failed !"**);**  **return** 1**;**  **}**  pinMode**(**BtnPin**,** INPUT**);**  pinMode**(**LEDpin**,** OUTPUT**);**  **while(**1**)**  **{**  **if(**1 **==** digitalRead**(**BtnPin**))**  **{**  delay**(**20**);**  **if(**1 **==** digitalRead**(**BtnPin**))**  **{**  digitalWrite**(**LEDpin**,** HIGH**);**  **}**  **}**  **else** **{**  digitalWrite**(**LEDpin**,** LOW**);**  **}**  **}**  **return** 0**;**  **}** |

## Python program

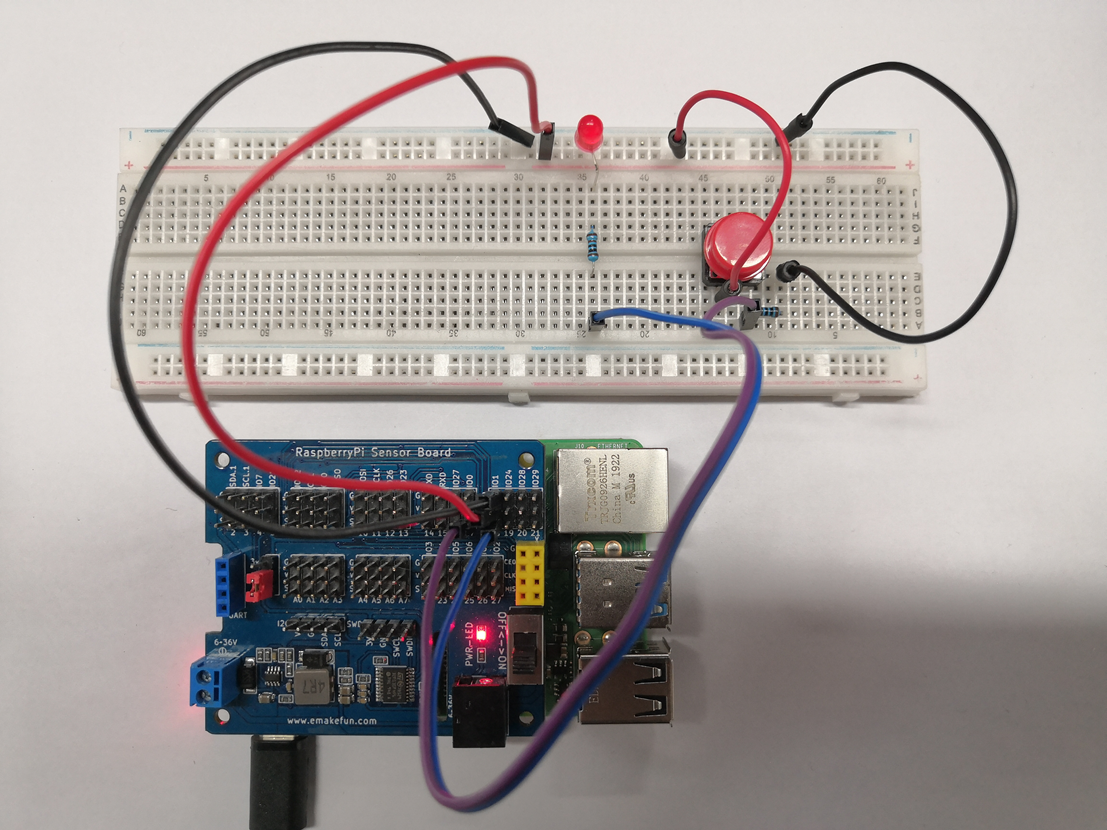
|  |
| --- |
| import RPi**.**GPIO as GPIO  button **=** 17  led **=** 18  GPIO**.**setmode**(**GPIO**.**BCM**)**  GPIO**.**setup**(**led**,**GPIO**.**OUT**)**  GPIO**.**setup**(**button**,**GPIO**.**IN**)**  **while** True**:**  **if** GPIO**.**input**(**button**):**  GPIO**.**output**(**led**,**GPIO**.**HIGH**)**  **else:**  GPIO**.**output**(**led**,**GPIO**.**LOW**)**  GPIO**.**cleanup**()** |

Run and compile these programs to complete this key switch experiment. The experimental principle of the LED module we use is very simple and it is extensively used in various circuits and electrical appliances. We can easily find it in various devices in real life. For instance, if we randomly press the buttons on the mobile phone, the backlight of the mobile phone will turn on, and if we click the elevator button, the indicator light on the elevator will light up and so on.

## Java program

|  |
| --- |
| **import** java**.**util**.**concurrent**.**Callable**;**  **import** com**.**pi4j**.**io**.**gpio**.**GpioController**;**  **import** com**.**pi4j**.**io**.**gpio**.**GpioFactory**;**  **import** com**.**pi4j**.**io**.**gpio**.**GpioPinDigitalInput**;**  **import** com**.**pi4j**.**io**.**gpio**.**GpioPinDigitalOutput**;**  **import** com**.**pi4j**.**io**.**gpio**.**PinPullResistance**;**  **import** com**.**pi4j**.**io**.**gpio**.**PinState**;**  **import** com**.**pi4j**.**io**.**gpio**.**RaspiPin**;**  **import** com**.**pi4j**.**io**.**gpio**.**trigger**.**GpioCallbackTrigger**;**  **import** com**.**pi4j**.**io**.**gpio**.**trigger**.**GpioPulseStateTrigger**;**  **import** com**.**pi4j**.**io**.**gpio**.**trigger**.**GpioSetStateTrigger**;**  **import** com**.**pi4j**.**io**.**gpio**.**trigger**.**GpioSyncStateTrigger**;**  public class Button **{**    public static void main**(**String**[]** args**)** **throws** InterruptedException **{**  // create gpio controller  final GpioController gpio **=** GpioFactory**.**getInstance**();**  // provision gpio pin #02 as an input pin with its internal pull down resistor enabled  final GpioPinDigitalInput myButton **=** gpio**.**provisionDigitalInputPin**(**RaspiPin**.**GPIO\_00**,**  PinPullResistance**.**PULL\_DOWN**);**  // setup gpio pins #01 as an output pins and make sure they are all LOW at startup  GpioPinDigitalOutput myLed**[]** **=** **{**  gpio**.**provisionDigitalOutputPin**(**RaspiPin**.**GPIO\_01**,** "LED #1"**,** PinState**.**LOW**),**  **};**    // create a gpio control trigger on the input pin ; when the input goes LOW, also set gpio pin #01 to LOW  myButton**.**addTrigger**(new** GpioSetStateTrigger**(**PinState**.**LOW**,** myLed**[**0**],** PinState**.**LOW**));**  // create a gpio control trigger on the input pin ; when the input goes HIGH, also set gpio pin #01 to HIGH  myButton**.**addTrigger**(new** GpioSetStateTrigger**(**PinState**.**HIGH**,** myLed**[**0**],** PinState**.**HIGH**));**  // keep program running until user aborts (CTRL-C)  **for** **(;;)** **{**  Thread**.**sleep**(**500**);**  **}**  // stop all GPIO activity/threads by shutting down the GPIO controller  // (this method will forcefully shutdown all GPIO monitoring threads and scheduled tasks)  // gpio.shutdown(); <--- implement this method call if you wish to terminate the Pi4J GPIO controller  **}**  **}** |

## Experimental Effect



The LED light is on when the button is pressed and it goes out when the button is released.