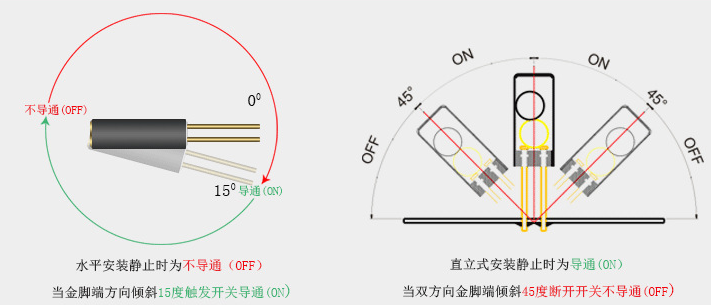
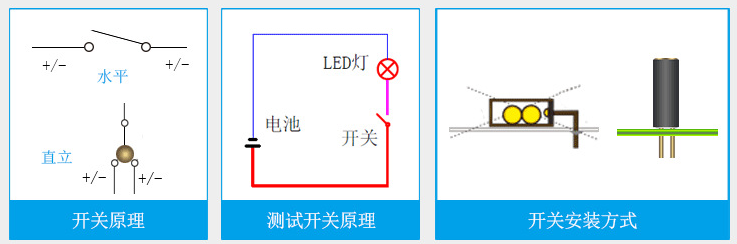
# Tilt Switch Experiment

## Introduction

The tile switch is also called a steel ball switch which is actually a vibration switch. The ball control the connection or disconnection of the circuit by touching or not touching the pin. Simply explaining this principle is like turning a light on or off. If the switch touches the inner metal plate, the light will turn on and if the switch does not touch the inner metal plate, the light will go out. Changing the path of light by touching a metal terminal or using a small bead in this switch will produce a conduction effect.

The tile switches are extensively used, such as: tire pressure monitoring system (**TPMS**), pedal lights, digital photo frame rotation, flip cameras, anti-theft systems and so on. The common tile switches in the market are **SW-200D, SW-300DA** and **SW-520D** models, etc. The switch used in this experiment is **SW-520D** which is OFF in a static state. While the external touch force meets the proper vibration conditions or the moving speed produces proper centrifugal force, the conductive needle will instantly be in the ON state and change its electrical characteristics. While the external force disappears and the electrical characteristics will return to the OFF state.





## Tilt Switch Parameters

1. Non-directional and it can be triggered at any angle.

2. Fully sealed package&waterproof and dustproof.

3. It is suitable for triggering in small current circuits.。

4. Double the pin and the touch point is more stable.

5. Sensitive, sealed, 12V 0.1mA, conduction time 0.1ms, open circuit impedance 10M, temperature tolerance 105. (Vibration switches are divided into high sensitivity, sensitivity, standard type and slow type. This switch is a sensitivity type.).

6. Its diameter is 4.5mm and its length is 11mm.



## Experimental Purpose

This experiment requires us to understand the working principle of the **SW-520D** shock sensor and how to use **Raspberry Pi** to control the **SW-520D** shock sensor to realize the anti-theft alarm function.

## Component List

* Raspberry Pi main board
* T-Cobbler Plus expansion board
* Breadboard
* Cable
* SW-520D Vibration sensor \* 1
* LED \*1
* Active Buzzer \*1
* 10kΩ Resistor \*1
* Several jumper wires

## Experimental Principle

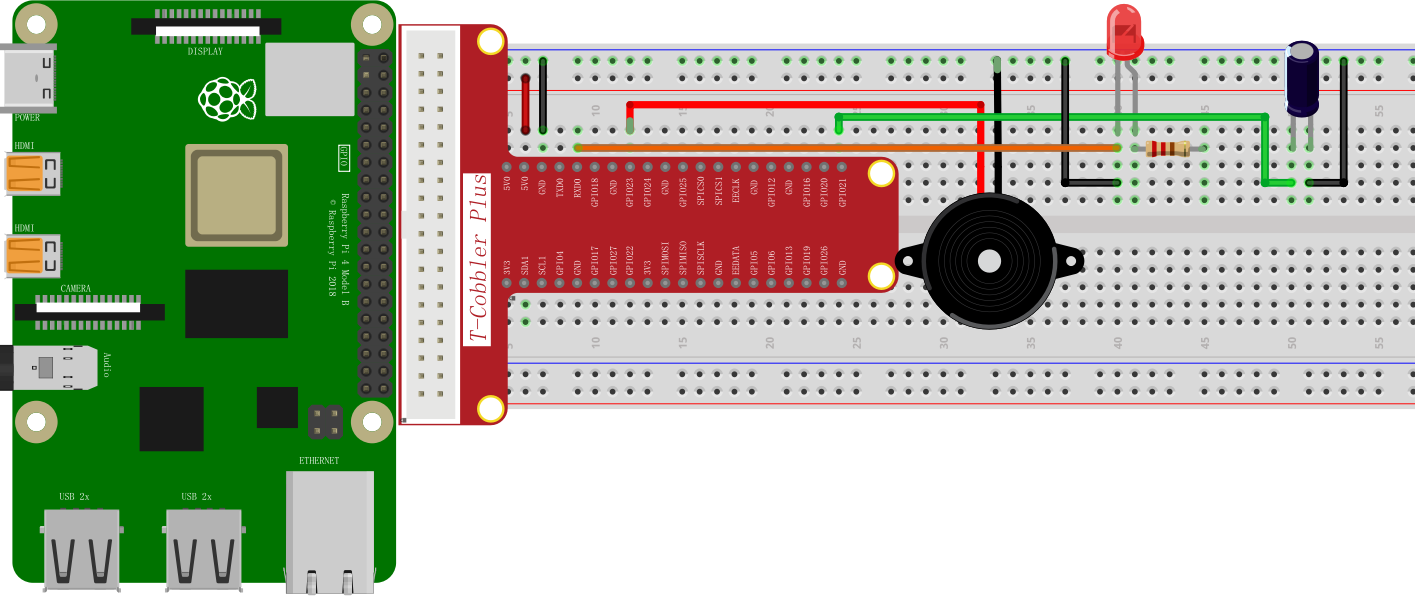
Using the working characteristics of the tile switch, when the circuit board is in a static state, the tile switch is in a conducting state. While an external force touches the circuit board and causes it to vibrate and cause the ball switch to disconnect, an alarm is triggered at this time, the LED is on and the buzzer sounds. While the ball switch returns to the on state, the LED goes out and the buzzer does not sound.

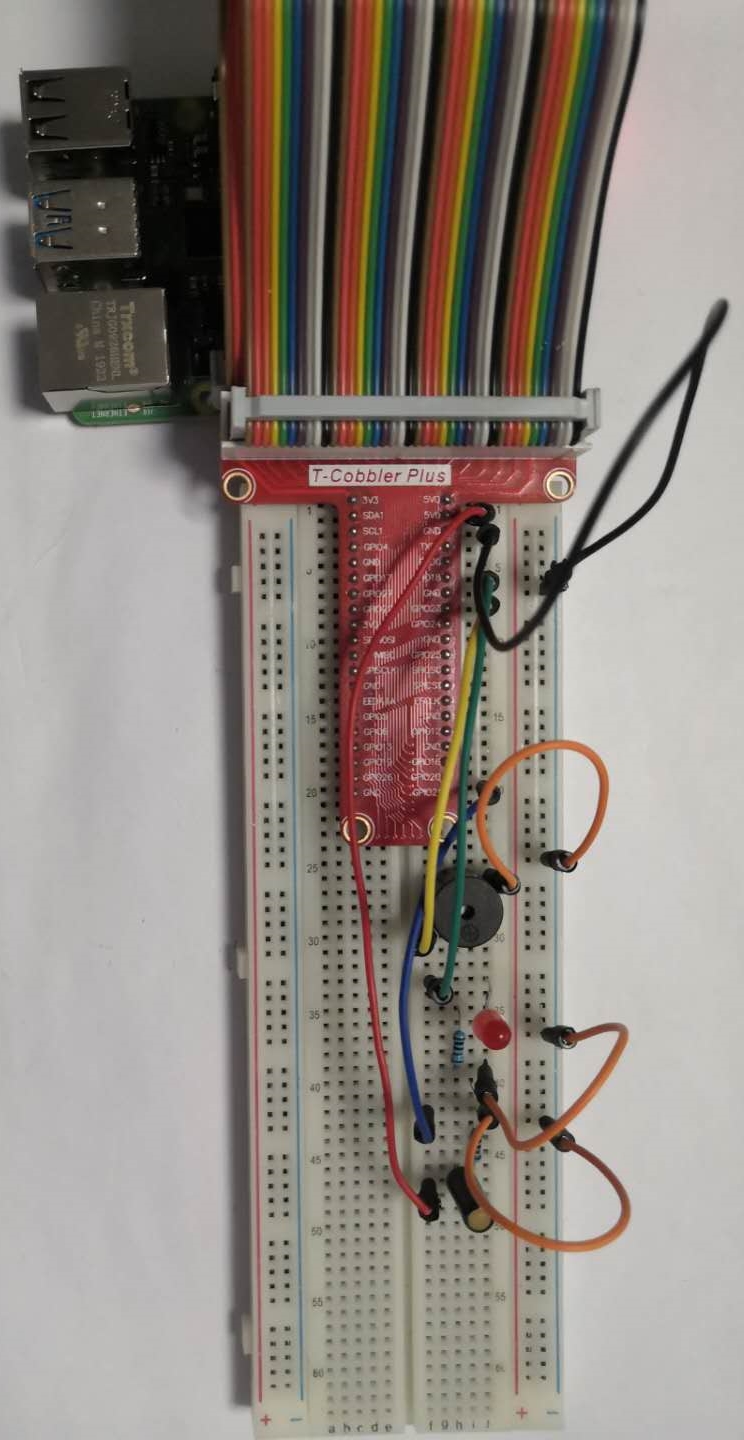
## Wiring

|  |  |
| --- | --- |
| Raspberry Pi | Active Buzzer |
| IO4(wiringPi)/23(BCM) | + |
| GND | - |

|  |  |
| --- | --- |
| Raspberry Pi | SW520D |
| IO29(wiringPi)/21(BCM) | 1 |
| GND | 2 |

|  |  |
| --- | --- |
| Raspberry Pi | LED |
| IO18(wiringPi)/1(BCM) | + |
| GND | — |





## C++ program

|  |
| --- |
| #include <stdio.h>  #include <wiringPi.h>  int main **()**  **{**  int buzzer **=** 4**;**  int ledpin **=** 1**;**  int sw **=** 29**;**  wiringPiSetup**();**  pinMode**(**buzzer**,** OUTPUT**);**  pinMode**(**ledpin**,** OUTPUT**);**  pinMode**(**sw**,** INPUT**);**  **while(**1**)**  **{**printf**(**"digitalRead(sw): %d\n"**,**digitalRead**(**sw**));**  **if(**digitalRead**(**sw**))**  **{**  digitalWrite**(**buzzer**,** HIGH**);**  digitalWrite**(**ledpin**,** HIGH**);**  delay**(**1000**);**  digitalWrite**(**buzzer**,** LOW**);**  digitalWrite**(**ledpin**,** LOW**);**  delay**(**10**);**  **}**  **}**  **}** |

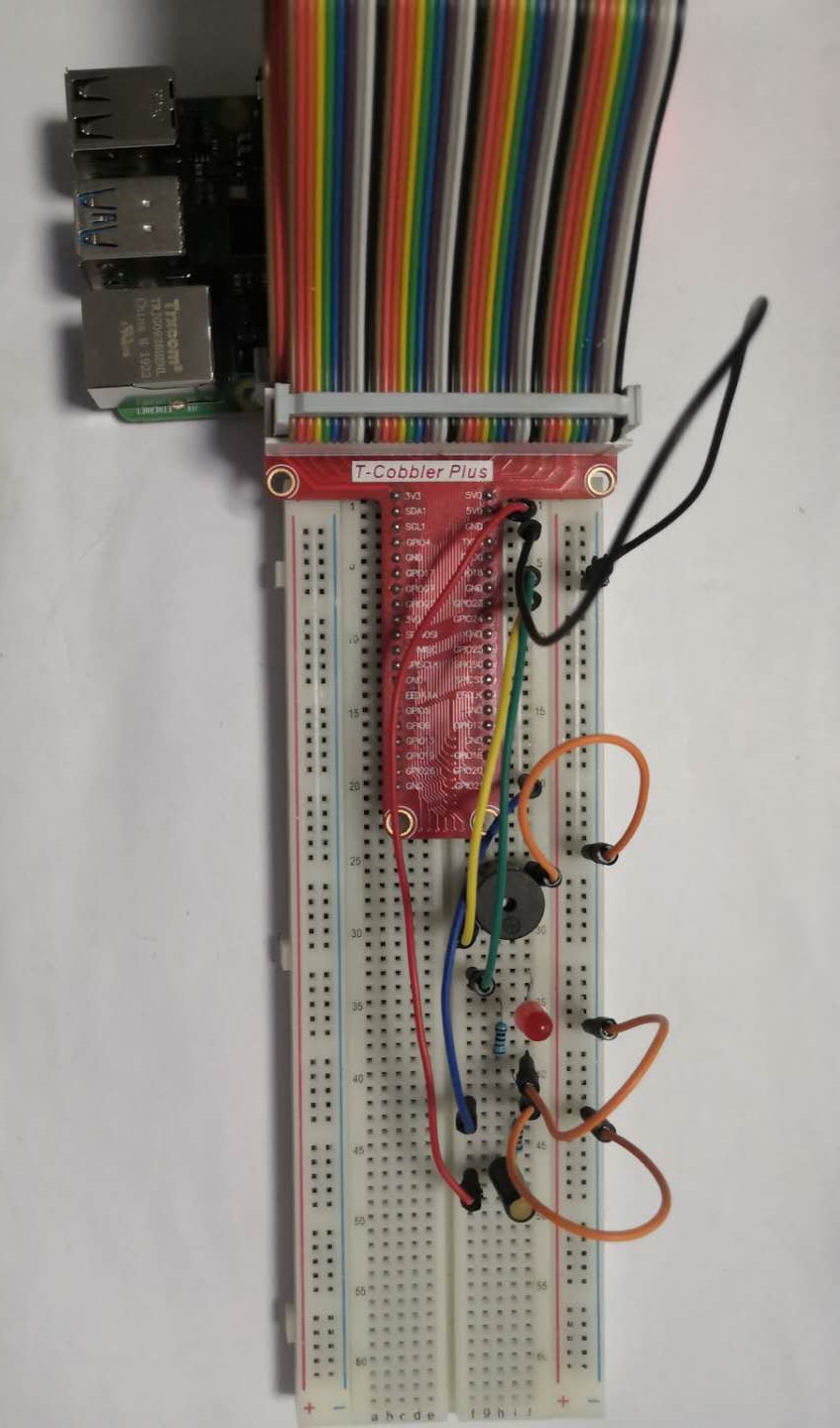
## Python program

|  |
| --- |
| import RPi**.**GPIO as GPIO  import time  buzzer **=** 23  ledpin **=** 18  sw **=** 21  GPIO**.**setmode**(**GPIO**.**BCM**)**  GPIO**.**setup**(**ledpin**,** GPIO**.**OUT**)**  GPIO**.**setup**(**buzzer**,** GPIO**.**OUT**)**  GPIO**.**setup**(**sw**,** GPIO**.**IN**)**  **while** True**:**  **if** GPIO**.**input**(**sw**):**  GPIO**.**output**(**ledpin**,** GPIO**.**HIGH**)**  GPIO**.**output**(**buzzer**,** GPIO**.**HIGH**)**  time**.**sleep**(**1**)**  GPIO**.**output**(**ledpin**,** GPIO**.**LOW**)**  GPIO**.**output**(**buzzer**,** GPIO**.**LOW**)**  time**.**sleep**(**0.02**)**  GPIO**.**cleanup**()** |

## 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 Tilt **{**    public static void main**(**String**[]** args**)** **throws** InterruptedException **{**  // create gpio controller  final GpioController gpio **=** GpioFactory**.**getInstance**();**  // provision gpio pin as an input pin with its internal pull down resistor enabled  final GpioPinDigitalInput sw **=** gpio**.**provisionDigitalInputPin**(**RaspiPin**.**GPIO\_03**,**  PinPullResistance**.**PULL\_DOWN**);**  // setup gpio pins #04, #05, #06 as an output pins and make sure they are all LOW at startup  GpioPinDigitalOutput TAB**[]** **=** **{**  gpio**.**provisionDigitalOutputPin**(**RaspiPin**.**GPIO\_04**,** "Buzzer"**,** PinState**.**LOW**),**  gpio**.**provisionDigitalOutputPin**(**RaspiPin**.**GPIO\_01**,** "led"**,** PinState**.**LOW**),**  **};**  // create a gpio control trigger on the input pin ; when the input goes HIGH, also set gpio pin to HIGH  sw**.**addTrigger**(new** GpioSetStateTrigger**(**PinState**.**HIGH**,** TAB**[**0**],** PinState**.**HIGH**));**  sw**.**addTrigger**(new** GpioSetStateTrigger**(**PinState**.**HIGH**,** TAB**[**1**],** PinState**.**HIGH**));**  // create a gpio control trigger on the input pin ; when the input goes LOW, also set gpio pin to LOW  sw**.**addTrigger**(new** GpioSetStateTrigger**(**PinState**.**LOW**,** TAB**[**0**],** PinState**.**LOW**));**  sw**.**addTrigger**(new** GpioSetStateTrigger**(**PinState**.**LOW**,** TAB**[**1**],** PinState**.**LOW**));**    **for** **(;;)** **{**  Thread**.**sleep**(**500**);**  **}**  **}**  **}** |

## Experimental Effect



Use the Raspberry Pi to control this tilt switch sensor to control a buzzer and a LED. While this switch is tilted, this circuit is connected, the buzzer sounds once and the LED flashes once.。