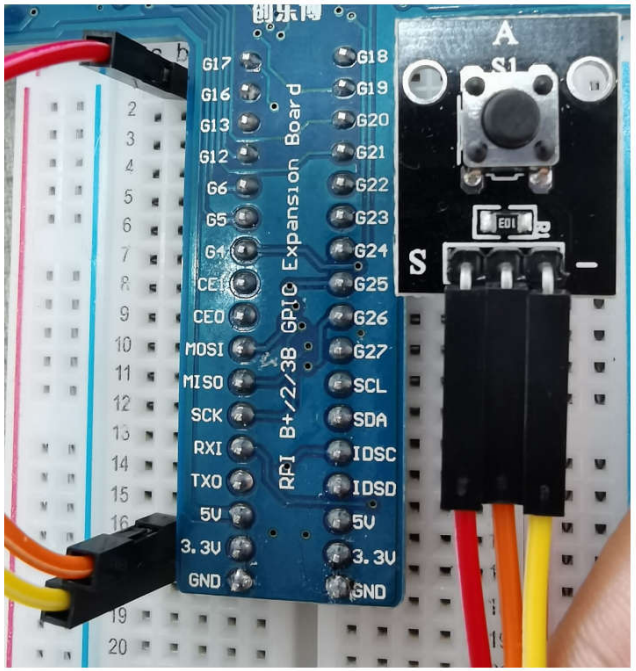
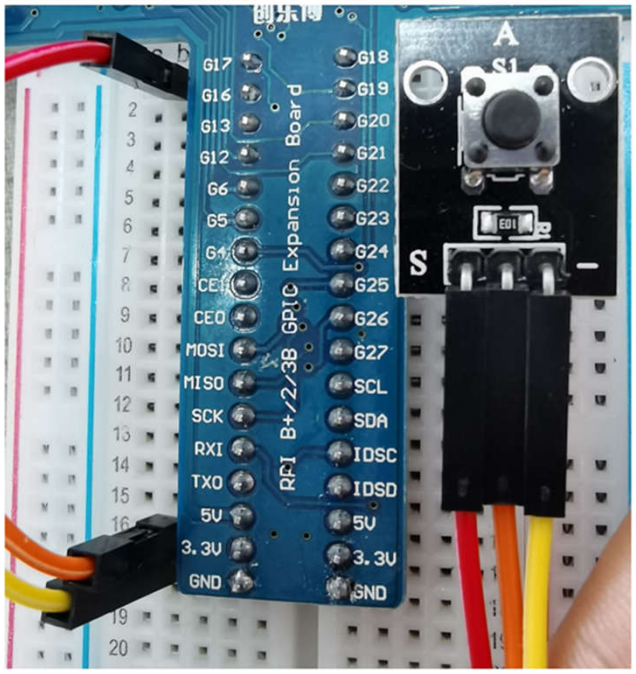
智能系统与控制



树莓派：GPIO-开关量输入

于泓
鲁东大学
信息与电气工程学院
2021.10.22

按键模块是一个使用非常频繁的电子部件，内部由一对轻触拨盘构成，按下时闭合导通，松开时自动断开。

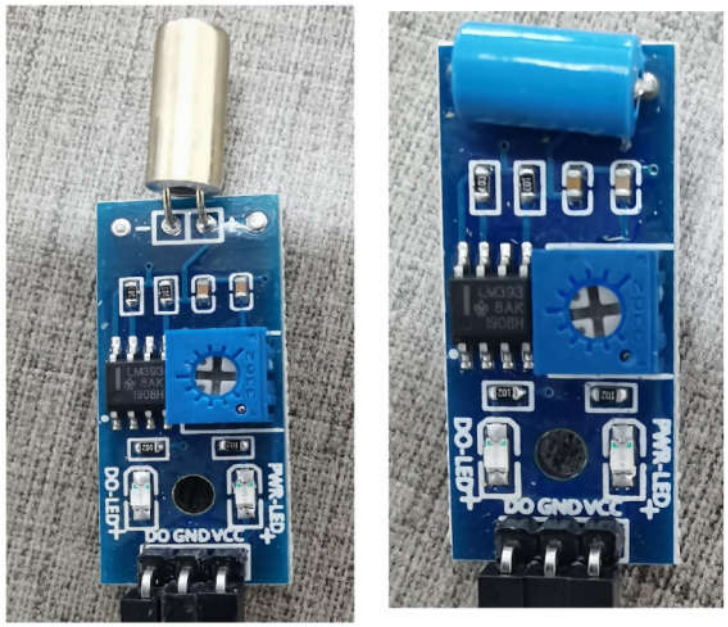


轻触开关模块的 S 端与树莓派扩展板的 GPIO17 相连。
其他两条线分别与 3.3v 和 GND相连接。

当‘-’端与 GND 相连接时， S 端正常状态下是高电平，有按键按下时为低电平

‘-’端也可以与高电平（3.3v）相连，那么 S 端正常状态是低电平，有按键按下时为高电平。

其他开关量传感器:



(a) 倾斜开关　　(b) 振动开关

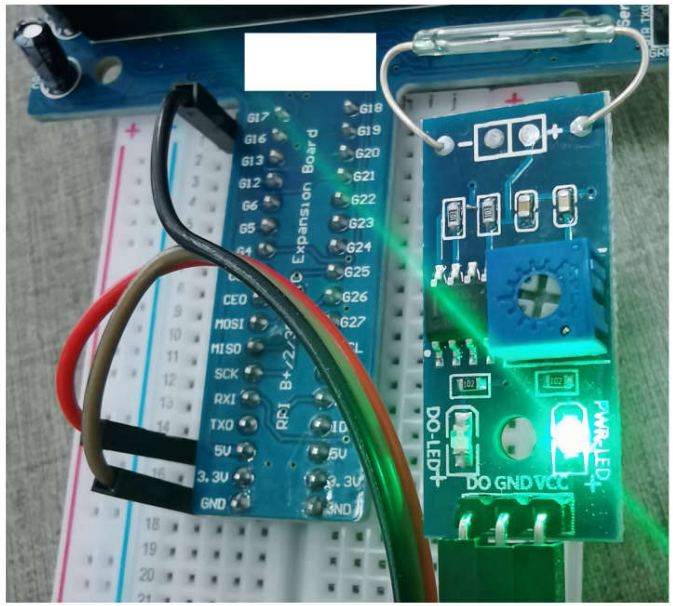


图 3.12: 干簧管传感器

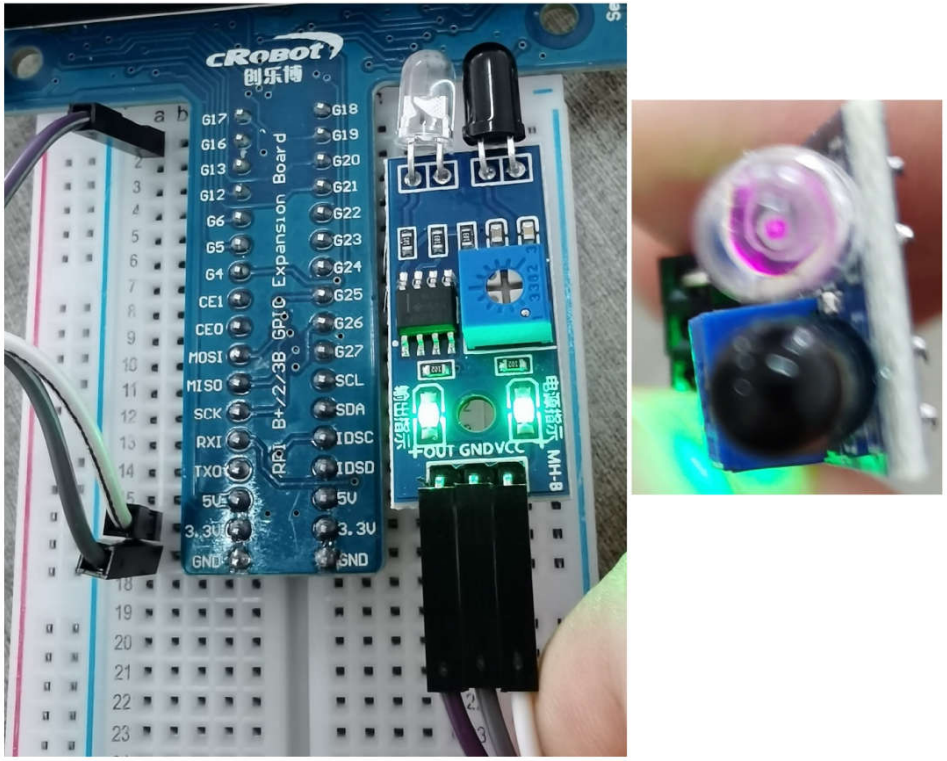


图 3.18: 红外避障传感器

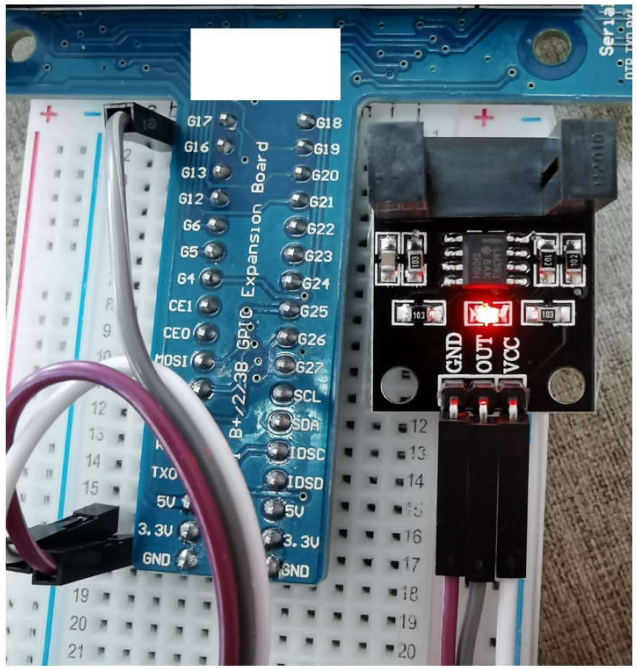


图 3.14: U 型光电开关传感器

对于按键的控制方式有两种：一种为**查询式**，一种为**中断式**。

查询方式的方法需要不断读取信号端传来的数值，当接收到信号触发电平时（低电平），则执行指定的操作

```python
import RPi.GPIO as GPIO
from pin_dic import pin_dic
import time


if __name__ == "__main__":

    # 设置引脚
    pin_Btn = pin_dic['G17']

    # 设置引脚编号模式
    GPIO.setmode(GPIO.BOARD)


    # 设置按键引脚工作方式，
    # 注意 pull_up_down=GPIO.PUD_DOWN 参数表示没有操作时引脚的状态，
    GPIO.setup(pin_Btn,GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

    try:
        while True: # Run forever
            # 如果获取按键是
            if GPIO.input(pin_Btn) == GPIO.HIGH:
                # 消除抖动
                time.sleep(0.2)
                print("按键按下")
    except KeyboardInterrupt:
        print('\n Ctrl + C QUIT')

    finally:

        GPIO.cleanup()
```

GPIO.PUD_DOWN
GPIO.PUD_UP

设置工作状态

消除抖动

```python
import RPi.GPIO as GPIO
from pin_dic import pin_dic
import time

count =0

# 回调函数 必须有一个输入就是引脚编号
def button_push(pin):
    global count
    count = count+1
    print(pin,count)
    print("Button was pushed!")


if __name__ == "__main__":

    # 设置引脚
    pin_Btn = pin_dic['G16']

    # 设置引脚编号模式
    GPIO.setmode(GPIO.BOARD)

    # 设置按键引脚工作方式,
    GPIO.setup(pin_Btn,GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

    # 定义回调函数
    GPIO.add_event_detect(pin_Btn, GPIO.RISING, callback=button_push, bouncetime=200)



    try:
        while True: # Run forever
            pass

    except KeyboardInterrupt:
        print('\n Ctrl + C QUIT')
    finally:
        GPIO.cleanup()
```

回调函数输入为引脚编号

中断式是指在程序中设置一个中断处理函数，当有设定的事件发生时，系统会自动调用
设置好的中断函数来进行处理。利用树莓派的 RPi.GPIO 的软件包，可以很方便的实现中断
处理功能， GPIO 引脚上的上升沿以及下降沿都可以实现中断的触发。

中断发生后延时一段时间后再进行中断响应(ms)

GPIO.FALLING
GPIO.RISING
GPIO.BOTH

在中断时间的检测过程中，经常会出现，上升/下降沿检测不准的情况，因此推荐使用 BOTH 触发方式在，中断函数中再进行，电平的判断

```
def button_push(pin):

    if GPIO.input(pin) == GPIO.HIGH:

        global count
        count = count+1
        print(pin,count)
        print("Button was pushed!")
```

```
if __name__ == "__main__":

    # 设置引脚
    pin_Btn = pin_dic['G16']

    # 设置引脚编号模式
    GPIO.setmode(GPIO.BOARD)

    # 设置按键引脚工作方式，
    GPIO.setup(pin_Btn,GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

    # 定义回调函数
    GPIO.add_event_detect(pin_Btn, GPIO.BOTH, callback=button_push, bouncetime=200)
```

注意：尽量用上升（高电平）触发事件

低电平触发时，参考电压 VCC 尽量选5v