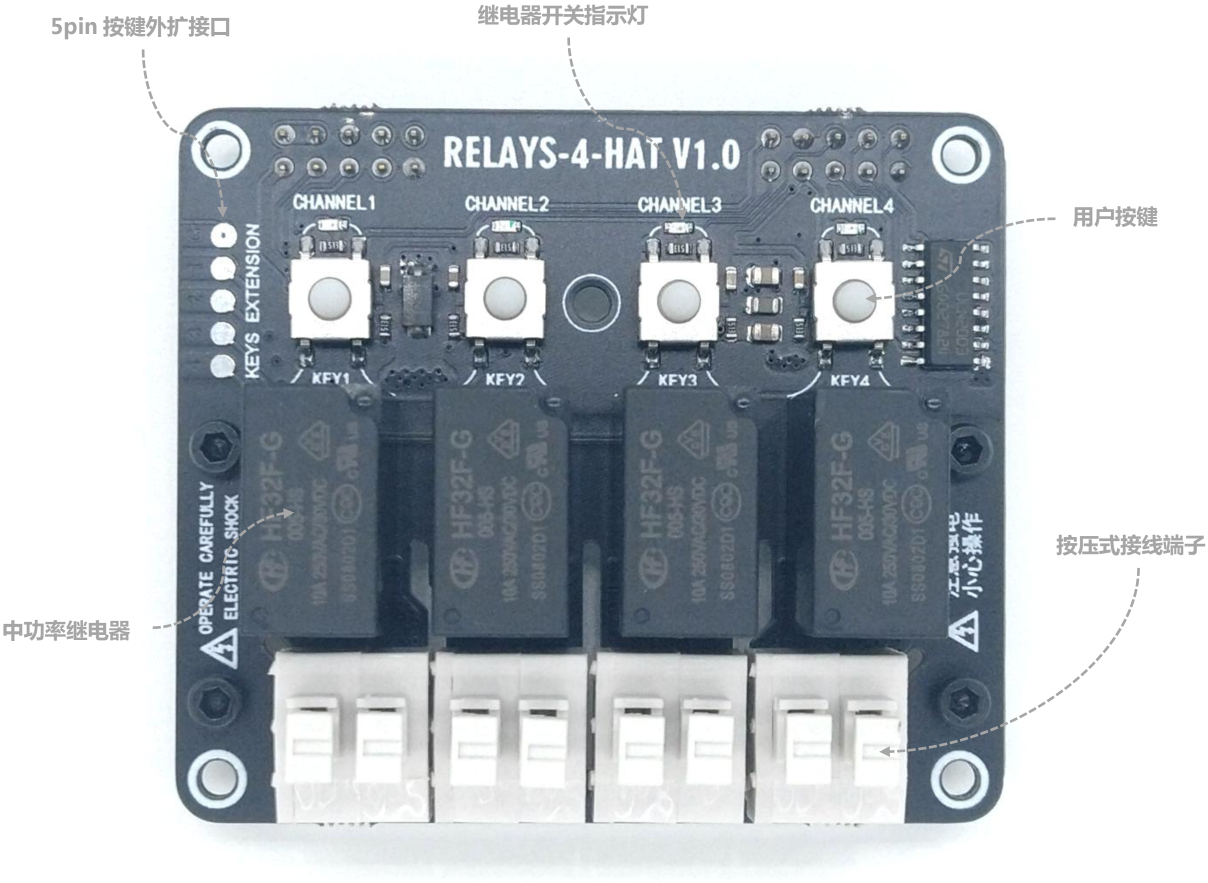
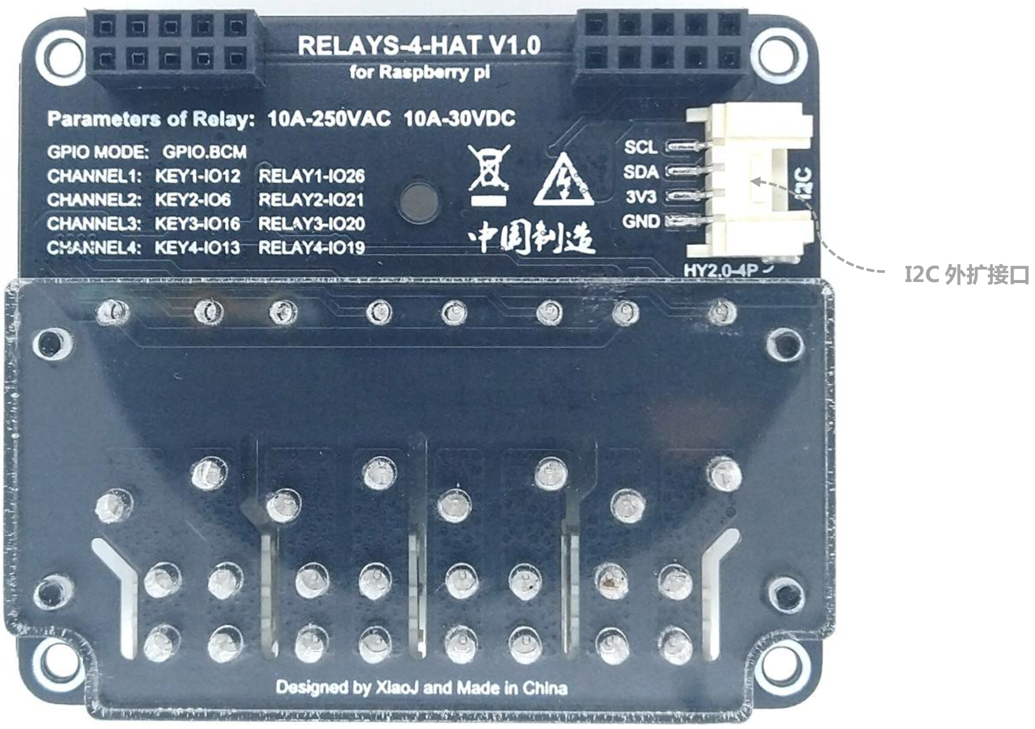
RELAYS-4-HAT 使用说明

4 路开关继电器扩展板

1. 简介

RELAYS-4-HAT 是一款专门为树莓派 A/B 型主板设计的继电器开关控制扩展板。该扩展板提供了 4 路中功率的开关继电器、4 个用户按键、5pin 按键外扩接口、1 路 I2C 外扩接口（兼容 GROVE 接口），采用按压式的接线端子，方便快捷。其中 4 路开关继电器可用于各类中小功率用电设备的开关控制，当继电器闭合时，板上对应的红灯会亮起，表示该继电器已经导通。4 个用户按键可通过编写相应程序设置为继电器开关控制按键，进而实现手动控制功能。5pin 按键外扩接口可以通过连线对 4 个用户按键进行延伸外扩。I2C 外扩接口采用 4pin HY2.0 接线座，可以外接各种 I2C 接口的传感器。




RELAYS-4-HAT V1.0
for Raspberry pi
Parameters of Relay: 10A-250VAC 10A-30VDC
GPIO MODE: GPIO.BCM
CHANNEL1: KEY1-IO12 RELAY1-IO26
CHANNEL2: KEY2-IO6 RELAY2-IO21
CHANNEL3: KEY3-IO16 RELAY3-IO20
CHANNEL4: KEY4-IO13 RELAY4-IO19
中国制造
SCL
SDA
3V3
GND
I2C
HY2.0-4P
Designed by XiaoJ and Made in China

I2C 外扩接口

电气参数：

总功率：**1.5W (4 路继电器全开，300mA@5V)**

继电器负载功率：**10A-250VAC 10A-30VDC**

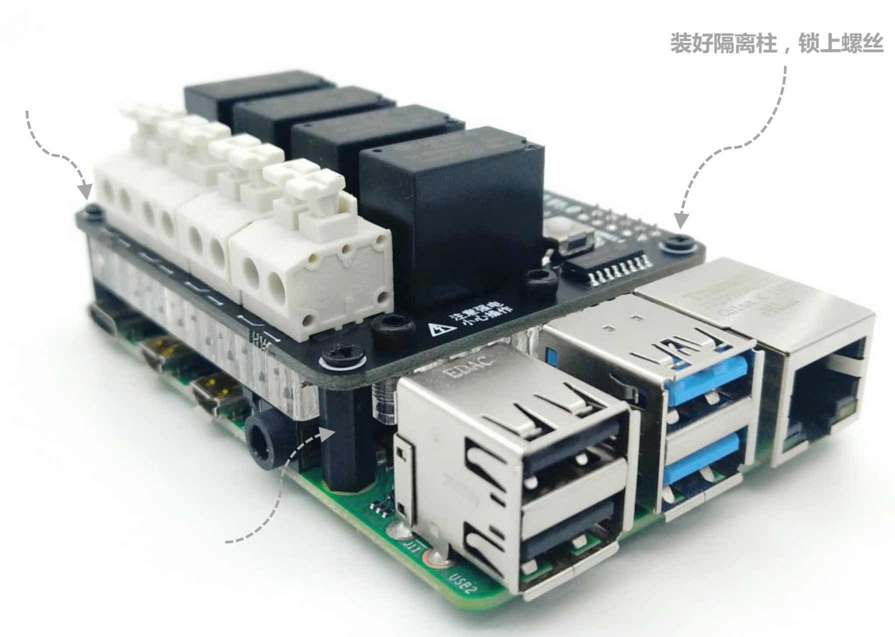
接线线径：**22~14 AWG**

GPIO 接线定义：

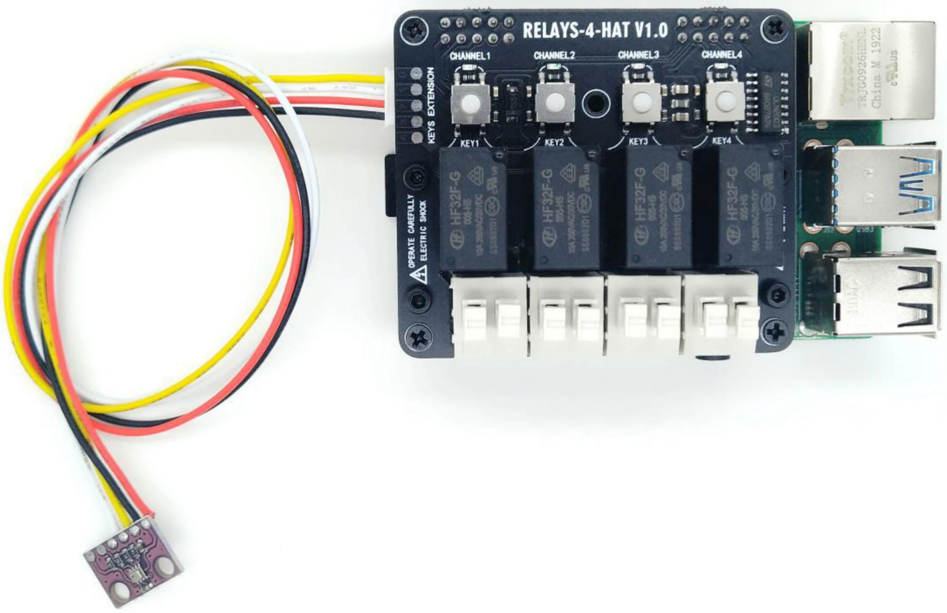
开关通道 1	继电器 1--IO26	按键 1--IO12
开关通道 2	继电器 2--IO21	按键 2--IO6
开关通道 3	继电器 3--IO20	按键 3--IO16
开关通道 4	继电器 4--IO19	按键 4--IO13
备注：GPIO 模式为 BCM。继电器高电平导通。按键外接了上拉电阻，按下低电平。		

2.安装

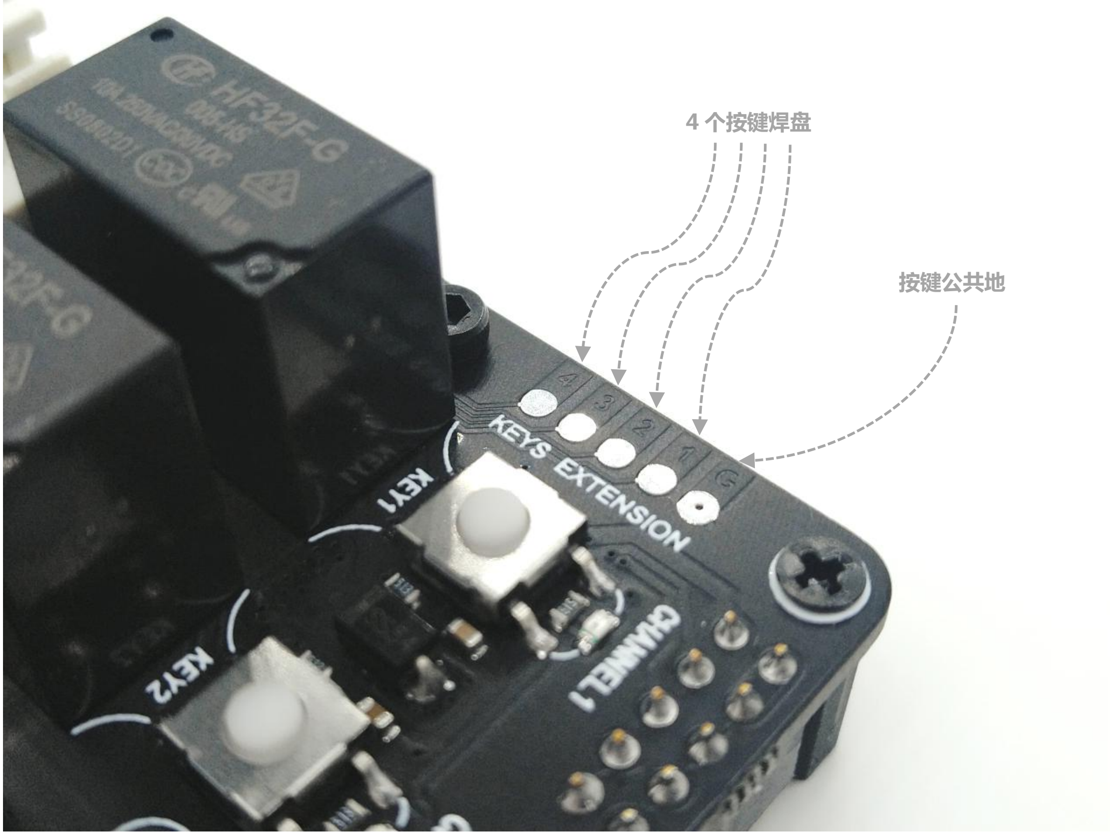
如图所示组装好配套的螺丝及隔离柱即可。



I2C 外扩接口如图所示可外接类似 BMP280 这样的 I2C 接口传感器



5pin 按键外扩接口，焊盘 1~4 对应按键 1~4，焊盘 G 对应 GND，当短接 1 跟 G 时，相当于按下了按键 1，IO 口检测到低电平。其他按键以此类推。



附录：

参考程序

```python
#!/usr/bin/python
# -*- coding:utf-8 -*-
import RPi.GPIO as GPIO
import time


Relay_Ch1 = 26
Relay_Ch2 = 21
Relay_Ch3 = 20
Relay_Ch4 = 19

Relay_Key1 = 12
Relay_Key2 = 6
Relay_Key3 = 16
Relay_Key4 = 13


Key1_state=0
Key2_state=0
Key3_state=0
Key4_state=0


def Key1_Interrupt(Relay_Key1):
    global Key1_state
    Key1_state=not Key1_state
    #print(Key1_state)
    time.sleep(0.1)
    if (Key1_state == 0):
        GPIO.output(Relay_Ch1,GPIO.LOW)
        print("Channel 1 : OFF")
    elif(Key1_state == 1):
        GPIO.output(Relay_Ch1,GPIO.HIGH)
        print("Channel 1 : ON")


def Key2_Interrupt(Relay_Key2):
    global Key2_state
    Key2_state=not Key2_state
    #print(Key2_state)
    time.sleep(0.1)
    if (Key2_state == 0):
        GPIO.output(Relay_Ch2,GPIO.LOW)
        print("Channel 2 : OFF")
    elif(Key2_state == 1):
        GPIO.output(Relay_Ch2,GPIO.HIGH)
        print("Channel 2 : ON")


def Key3_Interrupt(Relay_Key3):
    global Key3_state
    Key3_state=not Key3_state
    #print(Key3_state)
    time.sleep(0.1)
    if (Key3_state == 0):
        GPIO.output(Relay_Ch3,GPIO.LOW)
        print("Channel 3 : OFF")
    elif(Key3_state == 1):
        GPIO.output(Relay_Ch3,GPIO.HIGH)
        print("Channel 3 : ON")


def Key4_Interrupt(Relay_Key4):
    global Key4_state
    Key4_state=not Key4_state
    #print(Key4_state)
    time.sleep(0.1)
    if (Key4_state == 0):
        GPIO.output(Relay_Ch4,GPIO.LOW)
        print("Channel 4 : OFF")
    elif(Key4_state == 1):
        GPIO.output(Relay_Ch4,GPIO.HIGH)
        print("Channel 4 : ON")




GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

GPIO.setup(Relay_Ch1,GPIO.OUT)
GPIO.setup(Relay_Ch2,GPIO.OUT)
GPIO.setup(Relay_Ch3,GPIO.OUT)
GPIO.setup(Relay_Ch4,GPIO.OUT)

GPIO.setup(Relay_Key1,GPIO.IN)
GPIO.setup(Relay_Key2,GPIO.IN)
GPIO.setup(Relay_Key3,GPIO.IN)
GPIO.setup(Relay_Key4,GPIO.IN)


GPIO.add_event_detect(Relay_Key1,GPIO.FALLING,Key1_Interrupt,200)
GPIO.add_event_detect(Relay_Key2,GPIO.FALLING,Key2_Interrupt,200)
GPIO.add_event_detect(Relay_Key3,GPIO.FALLING,Key3_Interrupt,200)
GPIO.add_event_detect(Relay_Key4,GPIO.FALLING,Key4_Interrupt,200)




GPIO.output(Relay_Ch1,GPIO.HIGH)
time.sleep(0.4)
GPIO.output(Relay_Ch2,GPIO.HIGH)
time.sleep(0.4)
GPIO.output(Relay_Ch3,GPIO.HIGH)
time.sleep(0.4)
GPIO.output(Relay_Ch4,GPIO.HIGH)
time.sleep(0.4)
GPIO.output(Relay_Ch4,GPIO.LOW)
time.sleep(0.4)
GPIO.output(Relay_Ch3,GPIO.LOW)
time.sleep(0.4)
GPIO.output(Relay_Ch2,GPIO.LOW)
time.sleep(0.4)
GPIO.output(Relay_Ch1,GPIO.LOW)
time.sleep(0.4)




while True:

    print("Channel1: %d    Channel2: %d    Channel3: %d    Channel4: %d" %(Key1_state,Key2_state,Key3_state,Key4_state))
    time.sleep(1.5)

GPIO.cleanup()
```

参考资料：

https://github.com/linshuqin329/RELAYS_4_HAT

联系我：

416386001@qq.com