# RPi Relay Board(B)
# 用户手册

## 产品概述

本产品是 8 路继电器模块，提供接线端子和树莓派两种控制接口，可通过跳线帽选择，在只使用接线端子的情况下，也可将板子沿着中间的虚线折断，以减少板子的面积。

### 产品特性

- 适用于树莓派 A+/B+/2B/3B/3B+
- 采用高质量的继电器
- 继电器允许接入高达 5A 250V AC 或 5A 30V DC
- 带光耦隔离，避免高电压电路干扰
- 带继电器指示灯，方便查看继电器的工作状态
- 带继电器选择跳线，方便选择树莓派其他引脚控制继电器
- 提供完善的配套资料手册（包括 wiringPi、bcm2835、python、python-bottle 和 crontab 等例程）

### 接口说明

继电器通道和树莓派管脚的对应关系：

| 功能引脚 | RPi 引脚号 | wiringPi | BCM | 描述 |
|---|---|---|---|---|
| CH1 | 29 | P21 | 5 | 通道 1 |
| CH2 | 31 | P22 | 6 | 通道 2 |
| CH3 | 33 | P23 | 13 | 通道 3 |
| CH4 | 36 | P27 | 16 | 通道 4 |
| CH5 | 35 | P24 | 19 | 通道 5 |
| CH6 | 38 | P28 | 20 | 通道 6 |
| CH7 | 40 | P29 | 21 | 通道 7 |
| CH8 | 37 | P25 | 26 | 通道 8 |

【注意】PCB 上的丝印标识对应 BCM 编码。

## 使用说明

该模块仅提供树莓派的例程，支持使用 BCM2835、WiringPi、python、python-bottle、crontab 等方式来控制。

硬件配置

所有例程使用的主控板为：RaspberryPi

不论使用何种方式控制，模块和树莓派之间的硬件配置都是不变的。继电器和树莓派的管脚对应关系表在前面已经列出来了，以下不再赘述。

### BCM2835 例程的使用

1. 文件说明

   运行 ls 命令，可见如下文件

```
pi@raspberrypi:~/RPi_Relay_Board_B/bcm2835 $ ls
Makefile  Relay_demo  Relay_demo.c  Relay_demo.o
```

   其中：

   Makefile：为源代码文件交叉编译脚本，用于编译代码并生成可执行文件。若您修改了源代码，则需要先运行 sudo make clean 来删除所有已经生成的文件，再运行 sudo make 命令来生成可执行文件。

   Relay_demo：为编译后生成的可执行文件。

   Relay_demo.c：为该例程的源代码文件。

   Relay_demo.o：为编译时生成的中间文件，通常不需要理会。

   运行程序：`sudo ./Relay_demo`

2. 现象

   运行 sudo ./Relay_demo 命令之后，模块上的继电器依次闭合，然后依次断开，每个继电器带有响应的指示灯，通过观察指示灯的状态可知。当用户想要停止运行程序时，按 Ctrl+C 键可以停止程序的运行。

### WIRINGPI 例程的使用

1. 文件说明

   运行 ls 命令，可见如下文件

```
pi@raspberrypi:~/RPi_Relay_Board_B/wiringPi $ ls
Makefile  Relay_demo  Relay_demo.c  Relay_demo.o
```

   其中：

   Makefile：为源代码文件交叉编译脚本，用于编译代码并生成可执行文件。您若修改了源代码，则需要先运行 sudo make clean 来删除所有已经生成的文件，再运行 sudo make 命令来生成可执行文件。

Relay_demo：为编译后生成的可执行文件。
Relay_demo.c：为该例程的源代码文件。
Relay_demo.o：为编译时生成的中间文件，通常不需要理会。

运行程序：sudo ./Relay_demo

2. 现象

运行 sudo ./Relay_demo 命令之后，模块上的继电器依次闭合，然后依次断开，每个继电器带有响应的指示灯，通过观察指示灯的状态可知。当用户想要停止运行程序时，按 Ctrl+C 键可以停止程序的运行。

## PYTHON 例程的使用

1. 文件说明

运行 ls 命令，可见如下文件：

```
pi@raspberrypi:~/RPi_Relay_Board_B/python $ ls
Relay_demo.py
```

其中：

relay_demo.py：Python 源代码文件，包含了该模块控制的所有代码。

运行程序：sudo python relay_demo.py

2. 现象

运行 sudo python relay_demo.py 命令之后，模块上的继电器依次闭合，然后依次断开，每个继电器带有响应的指示灯，通过观察指示灯的状态可知。当用户想要停止运行程序时，按 Ctrl+C 键可以停止程序的运行。

## PYTHON-BOTTLE 的使用

1. python-bottle 简介

Bottle 是一个非常小巧但高效的微型 Python Web 框架，它被设计为仅仅只有一个文件的 Python 模块，并且除 Python 标准库外，它不依赖于任何第三方模块。详细教程见微雪课堂：http://www.waveshare.net/study/article-770-1.html

2. python-bottle 库的安装

运行命令：sudo apt-get install python-bottle

3. 文件说明

运行 ls 命令，可见如下文件：

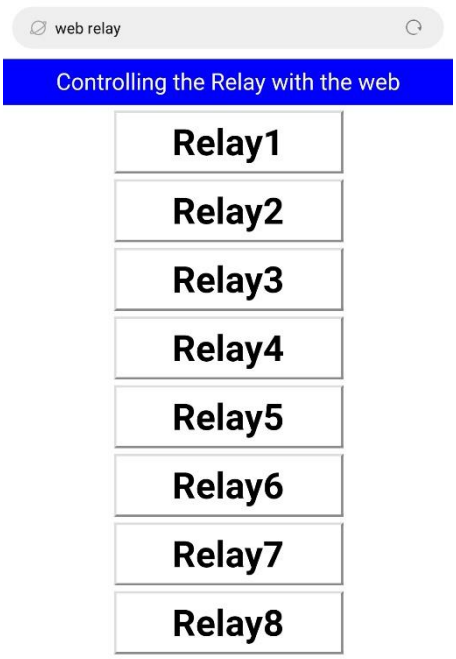
```
pi@raspberrypi:~/RPi_Relay_Board_B/python-bottle $ ls
index.html  jquery-3.3.1.js  main.py
```

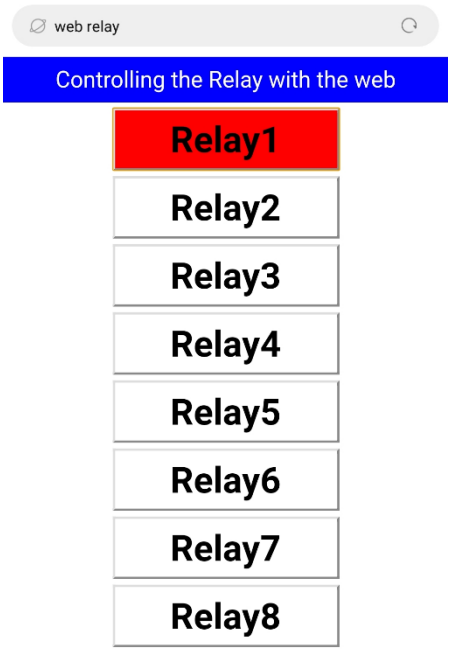
index.html：HTML 文件，为网页的源文件。
jquery-3.3.1.js：jquery 的源文件。jquery 是一个 JavaScript 库，极大地简化了 JavaScript 编程。其内部提供了大量的功能模块，在使用的时候直接调用即可。
main.py：控制部分源代码，接收网页传来的数据，并根据数据来控制 IO 电平，从而完成对继电器的控制。。

运行程序：sudo python main.py

4. 现象

运行 sudo python main.py 命令，在浏览器地址栏内输入树莓派 IP 地址，端口号 8080。即可看到有 8 个继电器的控制按钮，点击对应的按钮即可控制每个继电器的开断。网页显示内容如下：




## CRONTAB 的使用

1. crontab 简介

crontab 命令常见于 Unix 和类 Unix 的操作系统之中，用于设置周期性被执行的指令。该命令从标准输入设备读取指令，并将其存放于"crontab"文件中，以供之后读取和执行。

2. 文件说明

运行 ls 命令，可见如下文件：

```
pi@raspberrypi:~/RPi_Relay_Board_B/crontab $ ls
main.py  Relay_status.txt
```

其中：

main.py：为源文件，包含所有的控制代码，主要功能是从 Relay_status.txt 文件中读取上次的继电器数据，然后根据上次的数据执行该次的控制，最后再将这次的寄存器数据存入 Relay_status.txt 文件中，以便下次读取。

Relay_status.txt：存储每个继电器开断数据的文本文件。

3. 使用步骤

打开 crontab 文件夹，运行命令 pwd 以确认当前目录的路径，将 main.py 中的“dir”变量的路径修改为当前路径，在路径的最后加上 Relay_status.txt，使目录的路径变为该文件的路径。

运行命令 sudo crontab -e，打开 crontab 配置文件。在文件的结尾新建一行，输入：

*/1 * * * * sudo python /home/pi/RPi_Relay_Board_B/crontab/main.py

注意：命令中的 main.py 路径需要根据实际情况修改。

然后保存并关闭该文件。

```
# m h  dom mon dow   command
*/1 * * * * sudo python /home/pi/RPi_Relay_Board_B/crontab/main.py
```

该命令表示每隔一分钟执行一次该路径下的 main.py 文件，命令中的文件路径需要根据实际情况修改。

运行命令 sudo /etc/init.d/cron restart，重启 crontab 服务。

4. 现象

在重启 crontab 服务之后，crontab 服务便开始生效，继电器模块将会每隔一分钟断开一个继电器，当所有继电器断开之后，每隔一分钟则会闭合一个继电器，如此反复循环。

若想停止该 crontab 服务，只需要打开 crontab 配置文件，将相应的命令注释或者删除，然后再次重启 crontab 服务即可。