RPi Relay Board (B)
User Manual

OVERVIEW

This is 8-channel relay module, has both terminals and Raspberry Pi compatible socket, can be switched by jumpers. You can also cut the PCB along the "dotted line" to reduce size if you only want to use terminals.

FEATURES

- Compatible with Raspberry Pi A+/B+/2B/3B/3B+
- High quantity relays
- Load up to 5A 250V AC or 5A 30V DC
- Photo coupling isolation, prevent interference from high voltage circuit
- Onboard LEDs for indicating relays status
- Relay control jumper, allows to control the relays by custom pins other than the default pins
- Comes with development resources (wiringPi, bcm2835, python, python-bottle and crontab)

INTERFACE

Channel	RPI pin	wiringPi	BCM	Description
CH1	29	P21	5	Channel 1
CH2	31	P22	6	Channel 2
CH3	33	P23	13	Channel 3
CH4	36	P27	16	Channel 4
CH5	35	P24	19	Channel 5
CH6	38	P28	20	Channel 6
CH7	40	P29	21	Channel 7
CH8	37	P25	26	Channel 8

【Note】The silk printing on PCB are BCM2835 codes

HOW TO USE

We only provide demo code for Raspberry Pi. The codes we provide are bcm2835, WiringPi, python, python-bottle and contab.

Development board: Raspberry Pi

For all codes, hardware connection is same. Which you can refer to Interface part.

BCM2835 CODE

1. Files description

   Execute command **ls** to list the files on demo code (downloaded from Waveshare Wiki)

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

   Makefile: You need to execute **sudo make clean** and then **sudo make** to recompile code if you change codes.

   Relay_demo: Executable files

   Relay_demo.c: Sources code of this project.

   Relay_demo.o: Object files

2. Running code with command: **sudo ./Relay_demo**

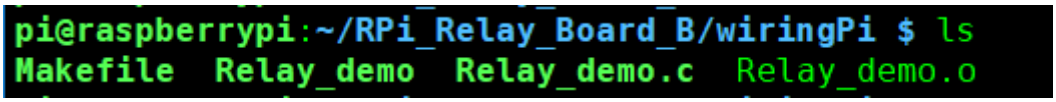
3. Expected result:

   The relays will close one by one, and then open. Every relay has one indicator, you can judge their states by the indicatos. Press Ctrl+C to stop process.

WIRINGPI CODE

1. Files description

   Execute command **ls** to list the files



   Makefile: You should execute command sudo make clean and then sudo make to generate new executable file if you modify codes.

   Relay_demo: Executable files

   Relay_demo.c: Sources code of this project.

   Relay_demo.o: Object files

2. Running code with command: **sudo ./Relay_demo**

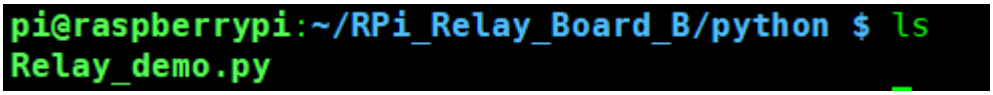
3. Expected result:

   The relays will close one by one, and then open. Every relay has one indicator, you can judge their states by the indicatos. Press Ctrl+C to stop process.

PYTHON CODE

1. Files description

   Execute command **ls** to list the files



   relay_demo.py: Sources code, includes all the control codes

2. Expected result:

   The relays will close one by one, and then open. Every relay has one indicator, you can judge their states by the indicatos. Press Ctrl+C to stop process.

PYTHON-BOTTLE CODE

1. python-bottle

   Bottle is a lightweight, efficient micro Python Web framework. It is distributed as a single file module and has no dependencies other than the Python Standard Library.

2. Install python-bottle

   sudo apt-get install python-bottle

3. Files description

   Execute command ls to list files

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

   index.html: HTML file, source code of web page

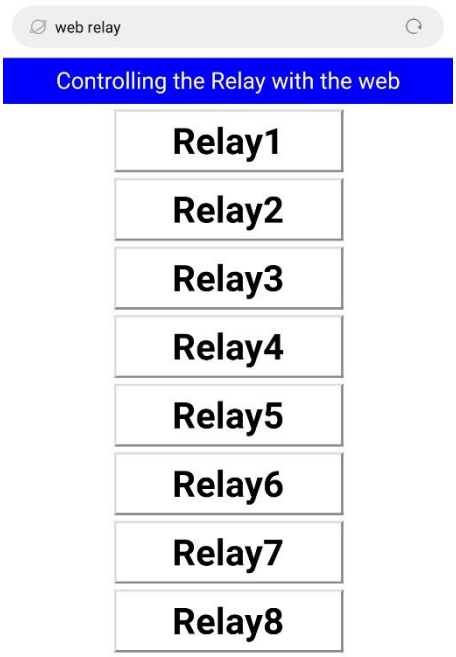
   jquery-3.3.1.js: source file of jquery. jquery is a JavaScript library, it makes JavaScript programming much simpler with many function modules which could be directly called when using.

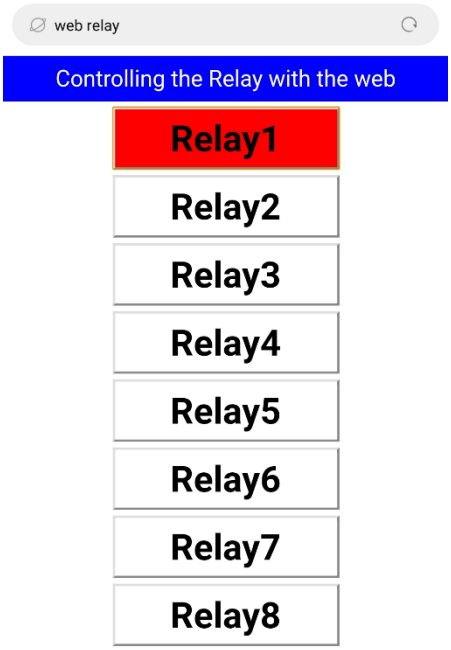
   main.py: Source code of controlling. It receives data from web page and control IO to control relays according to these data.

4. Running project: **sudo python mian.py**

5. Expected result:

   Running code, and type IP address of Raspberry Pi to browser to open the web page, port is 8080. Then you can open the web page which has control buttons for 8 relays as below, you can press these buttons to control relays.





CRONTAB CODE

1. crontab

   crontab are Unix command, used to create periodically executed crontab commands. Such command will read command from standard input devices and save to “crontab” file for further use.

2. Files description

   Execute command **ls** to list the files

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

mian.py: source file which includes all control codes. Its main function is to read last relay data from Relay_status.txt file, control replay according to the data and then save current register data to Relay_status.txt

Relay_status.txt: Files for saving status data of every relay

3. running code

Enter crontab directory, use **pwd** command to confirm the current path of directory. Change parameter "dir" to current path on main.py. Don't forget to add Relay_status.txt at the end of path

Execute command **sudo crontab -e** to open crontab configure file. Append statement to the end of file:

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

【Note】don't forget to change the path to correct one.

save and exit

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

This statement is used to run the main.py every minutes

Execute command **sudo /etc/init.d/cron restart** to restart crontab service

4. Expected result:

After restarting, crontab service is go to effect, and module will open one relay every minute, if all relays were opened, it turn to close one relay every minute, keep looping.

If you want to stop crontab service, you just need to open crontab configure file and comment the command (we added before) and restart crontab.