Introduction

An expansion board with 8-ch relays for Raspberry Pi. It gives your Pi the ability to control high voltage products such as home appliances.

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

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

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

   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

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

   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.

Libraries Installation for RPi

In order to use the API examples we provide, related libraries are required, which should be installed manually.

- bcm2835 libraries
- wiringPi libraries

Install WiringPi Library

Click to download the wiringPi libraries, or you can also obtain the latest version from the WiringPi website:

https://projects.drogon.net/raspberry-pi/wiringpi/download-and-install/

Copy the installation package into your own system via a USB flash drive, enter the WiringPi folder, then do this to install:

```
chmod 777 build
./build
```

Run the following command to check the installation:

```
gpio -v
```

Install C Library bcm2835

Click to download the bcm2835-1.39 libraries or bcm2835-1.45 libraries. You can also obtain the latest version from the bcm2835 website: http://www.airspayce.com/mikem/bcm2835/

Copy the installation package into your own system, enter the bcm2835 libraries folder, then do this to install:

```
./configure
make
sudo make check
sudo make install
```

Install Python Library

Python Libraries for Raspbian (contain RPi.GPIO and spidev installation packages.
See: https://pypi.python.org/pypi/RPi.GPIO https://pypi.python.org/pypi/spidev) get it by apt-get commands.

Please take a note, your Raspberry Pi should be connected to the network when using the command apt-get to install the library. Before the installation, you can run the following command to update your software list.

```
sudo apt-get update
```

1. Run the following command to install the package python-dev

   ```
   sudo apt-get install python-dev
   ```

2. Installing the RPi.GPIO package (GPIO interface functions). Copy the installation package RPi.GPIO to your RPi board, and unzip it. Enter the unzipped file under the terminal, and run the following command to install the library:

   ```
   sudo python setup.py install
   ```

3. Run the following command to install the library smbus (I2C interface functions)

```
sudo apt-get install python-smbus
```

4. Run the following command to install the library serial, which contains UART interface functions

```
sudo apt-get install python-serial
```

5. Installing the library spidev (SPI functions). Copy the installation package spidev to your RPi board, and unzip it. Enter the unzip file under the terminal, and run the following command to install the library:

```
sudo python setup.py install
```

6. Run the following command to install the Python imaging library

```
sudo apt-get install python-imaging
```

Configuring the interfaces

(Before running the API codes we provided, you should start up the corresponding core drivers of the interfaces. In the ready-to-use system image file, both I2C and SPI are set to Enable by default, but the serial port is still in the terminal debugging function mode.)

1. **Enable the I2C function**. Run the following command to configure your Raspberry Pi board:

```
sudo raspi-config
```

Select Advanced Options -> I2C -> yes, to start up the I2C core driver. Then you also need to modify the configuration file. Run the following command to open the configuration file:

```
sudo nano /etc/modules
```

Add the following two lines to the configuration file

i2c-bcm2708

i2c-dev

Press the keys Ctrl+X to exit, and input Y to save the settings. Then, reboot the module to make the settings take effect.

2. **Enable the serial function**. The serial port of RPi is set to serial terminal debugging function mode by default. If you want the serial port services as a common IO, you should modify the settings on the RPi. When the terminal debugging function is disabled, you cannot access RPi board via the serial port any more. If you want to control the RPi, you may need to enable the debugging function of the serial port again.

```
sudo raspi-config
```

Select Advanced Options -> Serial. Select the option no can disable the serial debugging function. And then, the serial port can be used for serial communication. And select the option yes can enable the serial debugging function. You should reboot the module to make the settings take effect.

Note: the serial port on Raspberry Pi 3 Model B is unusable, because Pin 14 and Pin 15 is connected to the on-board Bluetooth model.

1. **Start up the spi function**, run the following command:

```
sudo raspi-config
```

Select Advanced Options -> I2C -> yes, to start up I2C core driver.

File:RPi Relay Board (B) Schematic.pdf

RPi_Relay_Board_(B)_Schematic.pdf (0 × 0 pixels, file size: 520 KB, MIME type: application/pdf)

File:RPi Relay Board B.tar.gz

RPi_Relay_Board_B.tar.gz (file size: 114 KB, MIME type: application/x-gzip)