



# How to remote control alarm buzzer with IR device

Infrared has a wide application among all the models of Raspberry Pi. For example, you may use IR to remote control home media center XBMC. This tutorial is to help you learn how to use IR to connect Raspberry Pi with devices.

## Electronic Components:

- 1\* Raspberry Pi 3 Model B
- 1\* 5V 2.5A Raspberry Pi Power Supply
- 1\*VS18338B infrared receiver
- 1\* Raspberry Pi official camera(optional)
- 1\* Buzzer(optional)
- 1\*mini remote control board
- several pieces of DuPont jumpers

## DIY Steps:

- 1, Download the image and burn it into the system
  - 2, Set up the system and install packages
  - 3, Connect wires and adjust the config.txt configuration
  - 4, Connect the remote control panel and test it.
- 1, Download the image and burn it into the system

## Here is the download link for the official image:

<https://www.raspberrypi.org/download>

We recommend Raspbian system. This tutorial is based on Raspbian system. Please search for tutorial if you are using other versions.

```
pi@yoyoRPI:~$ sudo lsb_release -a
No LSB modules are available.
Distributor ID: Raspbian
Description:    Raspbian GNU/Linux 8.0 (jessie)
Release:       8.0
Codename:      jessie
```

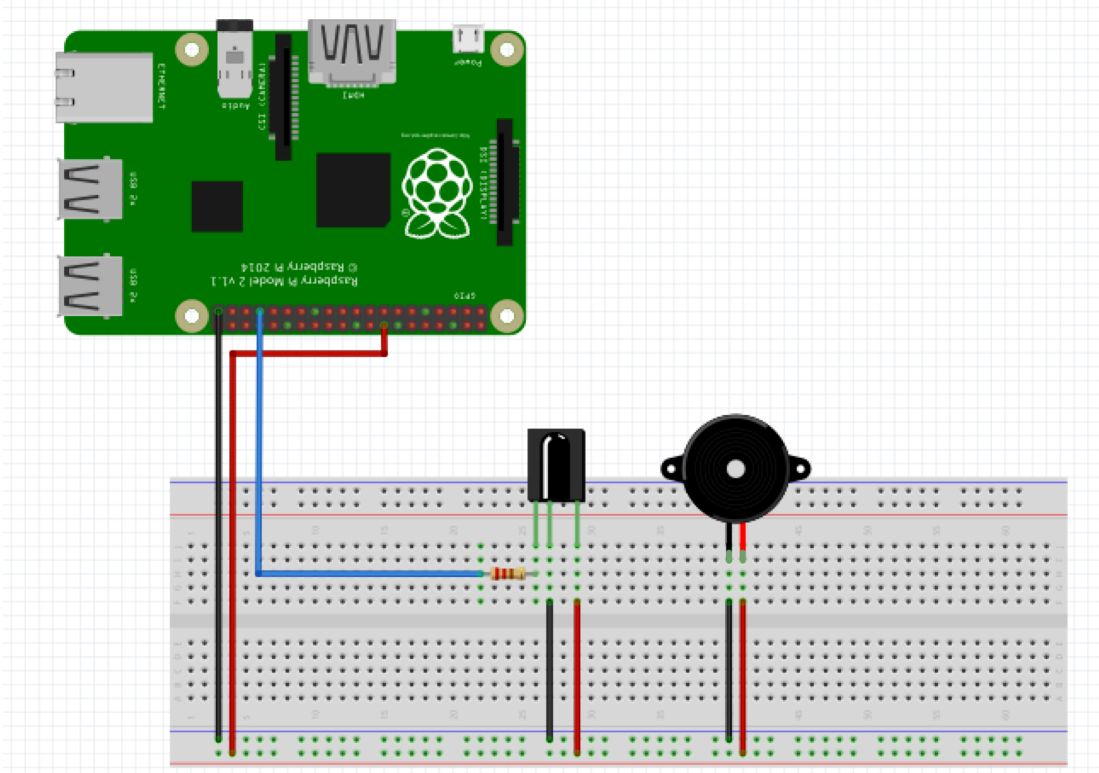
When it is burned, you can run extended file system with `sudo raspi-config`. Edit the host name and some small tuning operations.

- 1, Debug the system and install the packages:

Update the system with following commands and install packages for IR:

```
sudo apt-get update
sudo apt-get upgrade
sudo rpi-update
sudo apt-get -y install lirc
```

2, Connect the cables and adjust the config.txt configuration



To control the buzzer, please connect the power of the buzzer to the 16 Pin as shown in the below picture. It is displayed as No. 4 (BCM23) on WiringPi. Then connect the infrared sensors to GPIO 24 Pin (BCM23)

Pi 3											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA. 1	IN	1	3	4		5V			
3	9	SCL. 1	IN	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	1	TxD	15	14	
		0v			9	10	1	RxD	16	15	
17	0	GPIO. 0	IN	0	11	12	0	GPIO. 1	1	18	
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15	16	0	GPIO. 4	4	23	
		3.3v			17	18	0	GPIO. 5	5	24	
10	12	MOSI	IN	0	19	20		0v			
9	13	MISO	IN	0	21	22	0	GPIO. 6	6	25	
11	14	SCLK	IN	0	23	24	1	CE0	10	8	
		0v			25	26	1	CE1	11	7	
0	30	SDA. 0	IN	1	27	28	1	SCL. 0	31	1	
5	21	GPIO. 21	IN	1	29	30		0v			
6	22	GPIO. 22	IN	1	31	32	0	GPIO. 26	26	12	
13	23	GPIO. 23	IN	1	33	34		0v			
19	24	GPIO. 24	IN	0	35	36	0	GPIO. 27	27	16	
26	25	GPIO. 25	IN	0	37	38	0	GPIO. 28	28	20	
		0v			39	40	0	GPIO. 29	29	21	

The next step is to modify the configuration file: open the file /boot/config.txt with an editor and modify it as follows:

```
device_tree=bcm2710-rpi-3-b.dtb
dtoverlay=lirc-rpi,gpio_in_pin=13
```

Attention: the critical part of this step is to ensure that the 13 in gpio\_in\_pin=13 is the BMC13 Pin.

Then reboot the system and you may start debugging the system. Login and run this command to check if lirc\_rpi module has finished loading. It may be mistaken for the infrared driver.

```
sudo modprobe lirc_rpi
lsmod |grep lirc_rpi
dmesg |grep lirc
```

If what you see is as shown in the below picture, it means that the driver is loaded without problems.

```
pi@yoyoRPI: ~ $ sudo modprobe lirc_rpi
pi@yoyoRPI: ~ $
pi@yoyoRPI: ~ $
pi@yoyoRPI: ~ $ lsmod |grep lirc_rpi
lirc_rpi                6478  0
lirc_dev                8310  1 lirc_rpi
```

```
pi@yoyoRPI: ~ $ dmesg |grep lirc
[  3.919701] lirc_dev: IR Remote Control driver registered, major 244
[  3.958460] lirc_rpi: module is from the staging directory, the quality is unknown, you have been warned.
[  4.919641] lirc_rpi: auto-detected active low receiver on GPIO pin 13
[  4.919965] lirc_rpi lirc_rpi: lirc_dev: driver lirc_rpi registered at minor = 0
[  4.919975] lirc_rpi: driver registered!
```

Then edit the `/etc/lirc/hardware.conf` file into the following:

```
LIRCD_ARGS="--uinput",DRIVER="default"
DEVICE="/dev/lirc0", MODULES="lirc_rpi"
```

Save and exit. Please notice that the `LIRCD_ARGS` configuration is key here.

The final result is shown as below:

```
pi@yoyoRPI: ~ $ grep -v "#" /etc/lirc/hardware.conf |grep -v "^$"
LIRCD_ARGS="--uinput"
LOAD_MODULES=true
DRIVER="default"
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"
LIRCD_CONF=""
LIRCMD_CONF=""
```

Restart lirc service and reload the configuration:

```
sudo /etc/init.d/lirc restart
```

You may test the remote control panel now.

Enter the following command:

```
mode2 -d / dev / lirc0
```

Then press a button on the remote control panel. If you can see similar content, it shows that the interaction is good.



```
pi@yoyoRPI:~$ mode2 -d /dev/lirc0
space 2196610
pulse 9061
space 4541
pulse 569
space 574
pulse 569
space 604
pulse 513
space 598
pulse 546
space 572
pulse 572
space 573
pulse 541
space 581
pulse 563
space 596
pulse 544
space 598
pulse 525
space 1710
pulse 555
space 1709
pulse 570
space 1717
pulse 545
```

You may terminate the process with ctrl+c. Next you need to bind each code to make it work.

First, please review the list of key values can be used and the key value entered must be in this list

**irrecord --list-namespace |more**

Here are a couple of screenshots for your reference. The actual list is very long and daily remote control panel should be available here.



```
pi@yoyoRPI: ~ $ irrecord --list-namespace |more
KEY_0
KEY_102ND
KEY_1
KEY_2
KEY_3
KEY_4
KEY_5
KEY_6
KEY_7
KEY_8
KEY_9
KEY_A
KEY_AB
KEY_ADDRESSBOOK
KEY_AGAIN
KEY_ALTERASE
KEY_ANGLE
KEY_APOSTROPHE
KEY_ARCHIVE
KEY_AUDIO
KEY_AUX
```

Then you can record and generate a configuration file: `lircd.conf`

Before you record, you have to close the service. It is necessary to perform this step:

```
sudo /etc/init.d/lirc stop
```

```
pi@yoyoRPI: ~ $ sudo /etc/init.d/lirc stop
[ ok ] Stopping lirc (via systemctl): lirc.service.
```

When it is completed, execute the record command:

Start recording and you have to pass the 2\*80 test.

It is quite easy to pass the test. After you press the button, you will see a lot of points and please remember to keep pressing the button on the first row for a long time and press the second row one by one:

```
irrecord -n -d/dev/lirc0 lircd.conf
```

```
pi@yoyoRPI:~$ irrecord -n -d /dev/lirc0 lircd.conf

irrecord - application for recording IR-codes for usage with lirc

Copyright (C) 1998,1999 Christoph Bartelmus(lirc@bartelmus.de)

This program will record the signals from your remote control
and create a config file for lircd.

A proper config file for lircd is maybe the most vital part of this
package, so you should invest some time to create a working config
file. Although I put a good deal of effort in this program it is often
not possible to automatically recognize all features of a remote
control. Often short-comings of the receiver hardware make it nearly
impossible. If you have problems to create a config file READ THE
DOCUMENTATION of this package, especially section "Adding new remote
controls" for how to get help.

If there already is a remote control of the same brand available at
http://www.lirc.org/remotes/ you might also want to try using such a
remote as a template. The config files already contain all
parameters of the protocol used by remotes of a certain brand and
knowing these parameters makes the job of this program much
easier. There are also template files for the most common protocols
available in the remotes/generic/ directory of the source
distribution of this package. You can use a template files by
providing the path of the file as command line parameter.

Please send the finished config files to <lirc@bartelmus.de> so that I
can make them available to others. Don't forget to put all information
that you can get about the remote control in the header of the file.

Press RETURN to continue.
```

Press Enter to continue and you will see the below picture:

```
Now start pressing buttons on your remote control.

It is very important that you press many different buttons and hold them
down for approximately one second. Each button should generate at least one
dot but in no case more than ten dots of output.
Don't stop pressing buttons until two lines of dots (2x80) have been
generated.

Press RETURN now to start recording. █
```

Press RETURN now to start recording:

```
Press RETURN now to start recording.
.....
Found const length: 109484
Please keep on pressing buttons like described above.
█
```

Then follow the instruction and type in the name of the key:





```
Press RETURN now to start recording.
.....
Found const length: 109614
Please keep on pressing buttons like described above.
.....
Space/pulse encoded remote control found.
Signal length is 67.
No header found.
Found trail pulse: 565
No repeat header found.
Signals are space encoded.
Signal length is 33
Now enter the names for the buttons.

Please enter the name for the next button (press <ENTER> to finish recording)
```

Here I entered the KEY\_CHANNELDOWN.

You will be prompted to enter the name of the button you want to record. For example, I would like to enter the Number 1. I have to type in KEY\_NUMERIC\_1 and press Enter. Press and hold the corresponding button. Keep pressing 1 until you are prompted to type in the name of next button. Be patient and finish all the testing. You will be prompted to press some random buttons. If there are points, it shows the interaction works fine.

It will automatically exit after finished.

It will generate a configuration file called Lircd.conf in the current directory, and need to be copied to / etc / lirc / directory. Copy it to /etc/lirc/:

```
sudo cp lircd.conf /etc/lirc/lircd.conf
```

Start lirc service:

```
sudo /etc/init.d/lirc start
```

Run the irw command for testing:

```
sudo irw
```

Press the button and you will see the corresponding key value on the screen. If it is right, then it is fine. if it is wrong, repeat the above recording process.

When the recording is done, i want to test if i can control the buzzer with remote control panel. The principle is to check if buzzer can gain power supply from GPIO.

Next write down the following configuration /etc/lirc/lircrc file to receive the infrared signal and execute the command:

```
sudo vim.tiny /etc/lirc/lircrc
```

Type in the following parameters:

```
begin
```





```
prog = irexec  
button = KEY_NUMERIC_1 #name of the button  
config = sudo gpio mode 4 OUT && sudo gpio write 4 1 #command execution  
end
```

If want to bind more, just repeat the begin to end modules. Do not forget to restart lirc service after the configuration is done:

```
sudo/etc/init.d/lirc restart
```

Then you can hear the buzzer when you press No.1 button with the remote control. Of course, you can DIY more interesting applications. Feel free to explore it.