# Raspberry Pi Operated Remote Control

Tabitha Lee

November 03, 2016

## 1 Introduction

The purpose of this project was to control a pair of IR LEDs with a Raspberry Pi. The IR commands sent can then be automated for convenient testing of the IPTV box.

## 2 Materials

Below are the materials used for this project:

- Raspberry Pi
- SD Card
- Breadboard
- 2 IR LEDs
- NPN transistor
- 10K resistor
- Wires
- Male to Female Breadboard Jumper Wires
- IR Receiver
- Aluminum foil (optional)

**Notes on Materials** Any version of Raspberry Pi can be used as long as there is a working network port. An SD card of 8GB or higher is preferred as the Raspbian operating system requires at least 4 GB. Any IR LED or NPN transistor will do. Any IR receiver should work, but some work better than others and are listed on the LIRC website. For this project, VS838 was used.Aluminum foil greatly reduces the effects of interference present at the time of recording.

# 3 Procedure

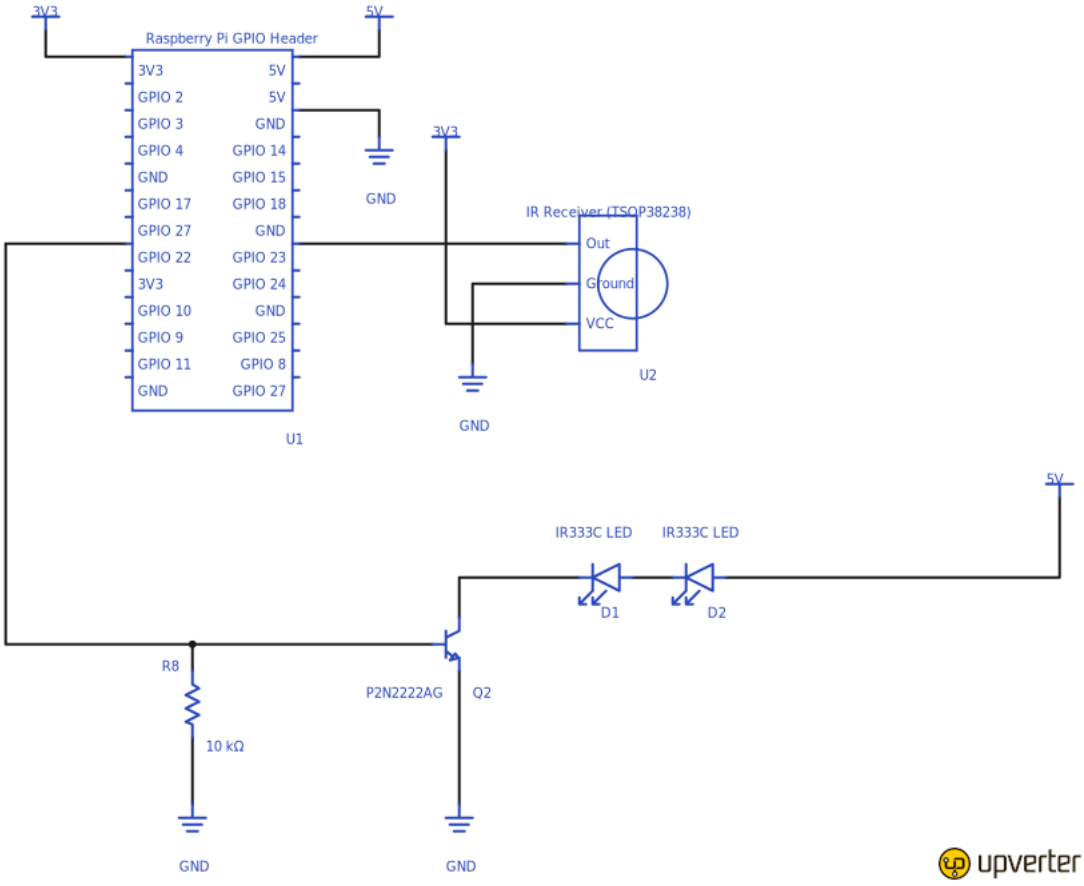
## 3.1 Background Information

Information from the remote is usually sent via a sequence of pulses and spaces, at a frequency of 38kHz. The gap length is the length of the trailing space, which is the space after the last pulse. This is the key to how LIRC records the remote control signals. It is generally longer than the rest of the signals, and is unique to each remote and even between different buttons.

LIRC will ask the user to press as many buttons as possible before it starts recording. This will enable it to calculate a gap length based on the average gap length of all the buttons pressed.

## 3.2 Recording

A simple receiver and transmitter circuit was built according to the diagram below.



The Raspberry Pi was then configured for LIRC according to link [1]

Although the choice of pin to use for input and output is arbitrary, to be consistent with this tutorial, the **input pin is 23** and the **output pin is 22**

After installing LIRC on the Raspberry Pi and connecting the receiver circuit (see Fig.), the following commands were executed by the Raspberry Pi. Accessing the Pi remotely via SSH (GitBash) was more convenient in the event of having to terminate a process.

```
sudo /etc/init.d/lirc stop
irrecord -f -d /dev/lirc0  /filename.conf
```

After following instructions from LIRC, the following command can be used to replace the default config file with the new file. LIRC only runs from its own directory so replacing the original .conf file is a necessary step.

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

LIRC must be started again before it can transmit via the LEDs.

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

The following command is useful for debugging as it allows one to see the stream of pulses and spaces that the IR receiver is detecting in real-time.

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

Finally, use the command below to see a list of allowed names for signals to be recorded under. This command is only used if a new config file needs to be recorded.

```
irrecord –list-namespace
```

At this point, some signals should be transmitting properly, however, not all will work. This is due to the different gap lengths. For this particular Arris remote, it was discovered that no one gap length was perfect in ensuring that all the commands worked.

Once a config file has been generated, a Python script can be used to send signals from the IR LEDs. See section 3.3 for more details on the Python code.

Although a bit heuristic, the following strategy was effective for recording most signals off the remote.

1. Stop LIRC and start the recording process
2. Press only a few buttons when prompted (preferably in the same section eg, only the number keys) instead of pressing as many buttons as possible so as to tailor the gap length to those specific buttons.
3. Turn off all fluorescent ligthting screens, and any lights that could be flickering. Cover LEDs on the Raspberry Pi with tinfoil or a sheet of paper or both.
4. Record commands according to the instructions displayed on the screen.
5. Copy the contents of the new config file generated and append to the master config file
6. Change the remote control name under 'name' in the header of the config file. This is the line right after 'begin remote'. This will be what the python command refers to as the remote name.
7. Replace the config file with the command in the previous section.
8. Connect the IR LED circuit and test all buttons, carefully identifying which signals are not working properly
9. Repeat all the steps above as needed, tailoring the gap length for the commands that are not working properly

### 3.3 Sending Commands with Python

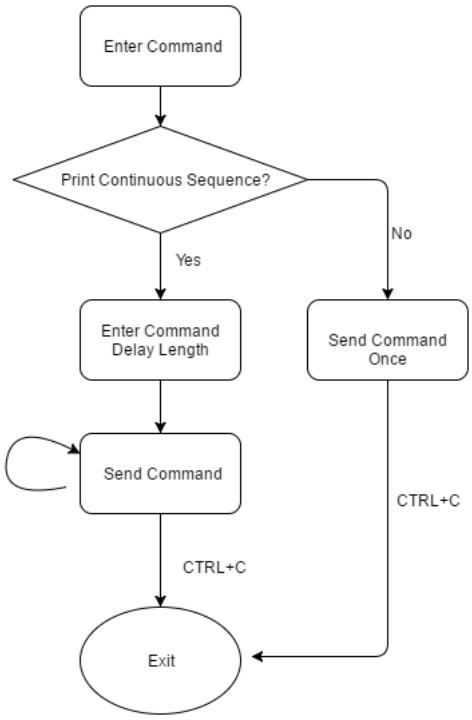
After a satisfactory config file had been generated, a Python script was used to send commands from the two IR LEDs. Two were used instead of one for a stronger signal and to provide a slightly larger range.

In the script, the main commands are

```
toEnter = ('irsend SEND_ONCE ' + getRemoteName(command[i]) +
' ' + keys[command[i]])
os.system(toEnter)
```

in which getRemoteName refers to the name of the remote control as chosen earlier and keys refers to a mapping file. The mapping file matches each button on the remote control with a number. The labelled picture is included at the end of this file. When prompted to enter the sequence of commands, simply enter the number corresponding to the remote button. For a sequnce, ensure that the numbers are separated by a space.

The logic flow of the Python Script runs as in the diagram below.




# 4 Discussion

## 4.1 Sources of Error

The two largest sources of error for this project were interference and different gap lengths.

The main source of interference stemmed from the fact that the fluorescent lights overhead flickered at around the same frequency as an IR signal. Other sources of pulsating light nearby also affected the sequence of pulses and spaces. Once mode2 was activated, it was easy to see that the stream of pulses and spaces were flickering too. The ideal stream one can see in mode2 should be one where minimal amounts of pulses and spaces are detected. This can be achieved in a room with all fluorescent lights turned off and with aluminum foil covering the LED light from the Raspberry Pi

The fact that there were different gap lengths for different sections on the remote led to the recording of multiple config files in which not all remote control commands were able to be transmitted. The solution to this was to first record an initial config file and then figure out which buttons did not work. Then a new config file with a different name was recorded, but when prompted by LIRC, only pressing the buttons that did not work. This enabled LIRC to detect a new gap length which is suitable for recording those specific buttons. The new config file is then appended to the original config file. After testing the combination of config files, the process is repeated again until all button commands can be transmitted.

However, there are some limitations to the LIRC software, and not all buttons can be recorded. This will be further discussed in the next section.

### 4.2 Internal Commands and Multiple Identities

There are some buttons used for internal commands, like setup, stb, dvd, aux, which are used to configure the remote and therefore no actual IR signals are sent out from the remote. In this case, it is impossible and not necessary to record the button command.

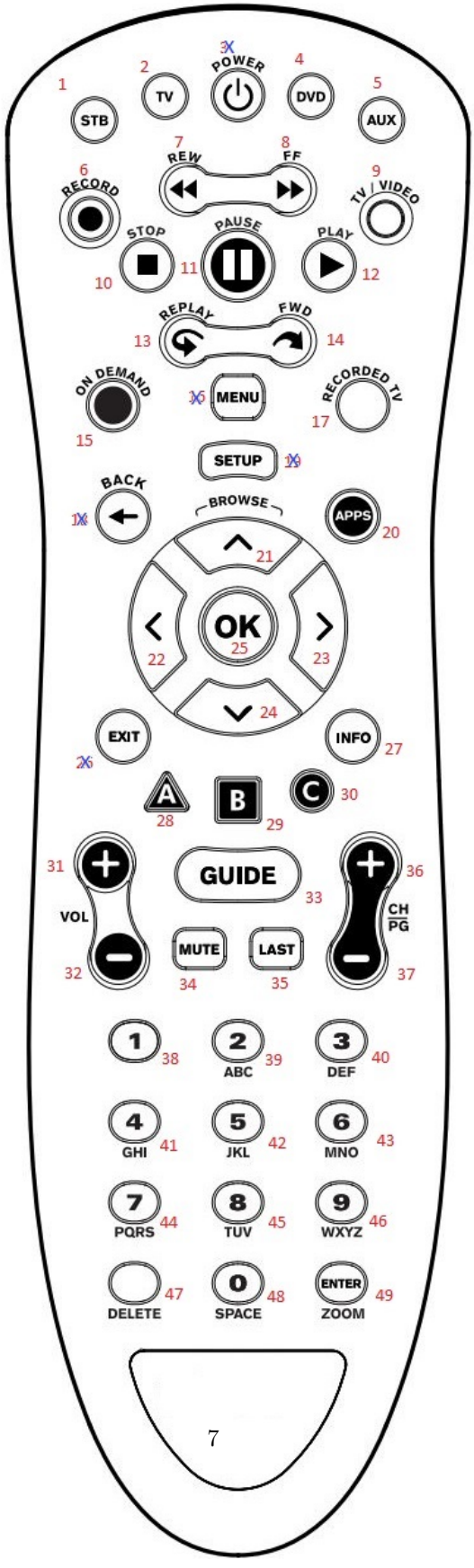
Other buttons, such as poweron, back, exit, and menu have been recorded many times but as such do not seem to be transmittable via the IR LEDs. It is speculated that since they control multiple devices, LIRC may not be able to record a specific gap length. However, the same command can be replicated with other keys, eg. the menu key can be replicated by the A key or sometimes the left arrow key.

## 5 Conclusion

This project was intended to be completed within a short period of time. As such, further improvements could be made. These include deeper investigation on the protocols of the remote so that a functioning power, back, and exit command can be sent out. This project was also intended to test IPTV boxes for potential crashes, but a specific testing regime has not been laid out yet. The ideal use for the Raspberry Pi controlled remote would be continuous looping of a sequence of signals.

## 6 Sources

[1] http://alexba.in/blog/2013/01/06/setting-up-lirc-on-the-raspberrypi/
[2] http://www.lirc.org/
[3] http://alexba.in/blog/2013/03/09/raspberrypi-ir-schematic-for-lirc/


1
STB
2
TV
3
POWER
4
DVD
5
AUX
6
RECORD
7
REW
8
FF
9
TV / VIDEO
10
STOP
11
PAUSE
12
PLAY
13
REPLAY
14
FWD
15
ON DEMAND
16
MENU
17
RECORDED TV
18
SETUP
19
BACK
20
APPS
BROWSE
21
22
OK
25
23
24
26
EXIT
27
INFO
28
A
29
B
30
C
31
VOL
32
33
GUIDE
34
MUTE
35
LAST
36
CH
PG
37
1
38
2
ABC
39
3
DEF
40
4
GHI
41
5
JKL
42
6
MNO
43
7
PQRS
44
8
TUV
45
9
WXYZ
46
DELETE
47
0
SPACE
48
ENTER
ZOOM
49