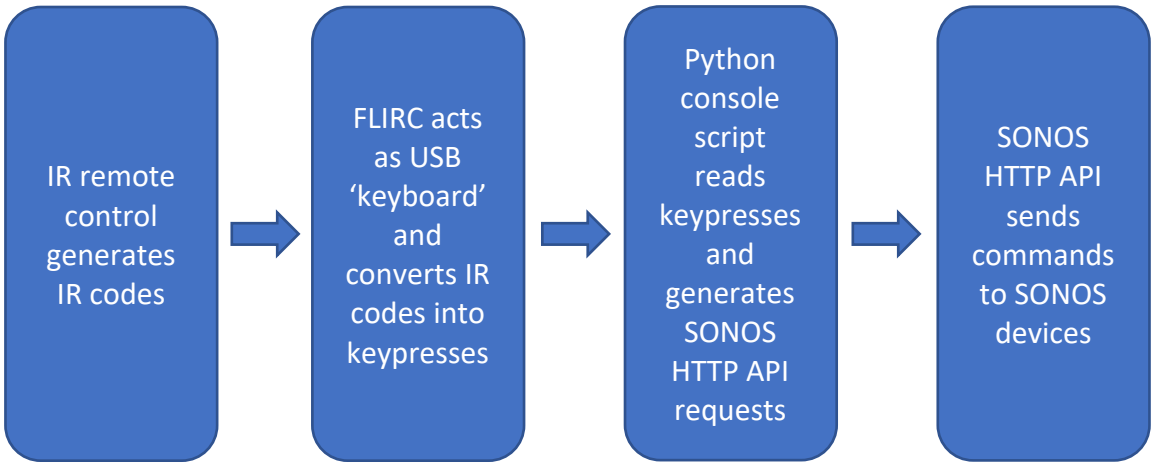
SONOS Infrared Control using FLIRC

Peter Toft <pwtoft@gmail.com>
** DRAFT ** V0.3 2018-08-27

Basic Setup



FLIRC

A FLIRC [1] is a small, programmable USB dongle that receives infrared (IR) signals from IR remote controls, and translates them into keyboard keypresses. It retails for $22.95 in the US and £22 in the UK.

The FLIRC behaves like an attached USB keyboard and is inserted into the computer that will control your SONOS device(s). When programmed IR codes are sent to the FLIRC, it generates keyboard characters. Obviously, the FLIRC needs to be within line of sight of the remote control(s) you're going to use. A USB extension cable can be used if required.

The FLIRC can be used with any standard IR remote control, as far as I know. Since it learns the codes via programming, there is no requirement for a remote control to be specifically 'supported'.

Programming the FLIRC

FLIRC programming consists of teaching the FLIRC the IR codes you want to use from your remote control(s), using the supplied FLIRC software. Program the FLIRC to generate unique keyboard characters for every button on the remote that you wish to use with SONOS. For example, you may want to map the 'Play' button on a remote to the keyboard character 'p'.

The FLIRC has persistent memory, and programming only needs to be performed once unless any changes are subsequently required.

Since the FLIRC behaves like a keyboard it's simple to test whether you have programmed it correctly. Just send the relevant IR codes to the FLIRC and the associated characters should appear on the computer console as if they were typed on a keyboard.

Configuring the Python Keypress Capture Script

A simple Python script runs as a console program and is used to read FLIRC 'keypresses' and to generate the appropriate HTTP requests to the SONOS HTTP API [2].

The script is shown below, and is available for download at [3]. In the configuration section, you need to edit the script to suit your local SONOS environment and the IR commands you wish to enable:

1. Edit the **room name** you want to control. This needs to match the room name in SONOS exactly.
2. Edit the SONOS HTTP API **hostname** or IP address.
3. Set up any **favourites** you want to start via IR control.

```
# SONOS keypress capture script. Converts keypresses into
# requests to the SONOS HTTP API.

import sys, os, requests, datetime

# Configuration Section ##############################################

# Set up Room Name(s)
study = 'study'
front_reception = 'front%20reception'

# Set up the SONOS HTTP API Host. Can be 'localhost'.
hostname = '192.168.0.36'

# Assemble the root URL to use for SONOS HTTP API Requests
root_url = 'http://' + hostname + ':5005/' + front_reception + '/'

# Set up SONOS Favourites (if required). These must be identical to
# the favourite names set within SONOS.

# 0 : Radio Paradise
# 1 : Jazz24
# 2 : BBC Radio 2
# 3 : BBC Radio 3
# 4 : BBC Radio 4
# 5 : Classic FM
# 6 : BBC Radio 6 Music

zero = 'favorite/Radio%20Paradise%20UK%20AAC%20320kbps'
one = 'favorite/Jazz24%20-%20KNKX-HD2'
two = 'favorite/BBC%20Radio%202'
three = 'favorite/BBC%20Radio%203'
four = 'favorite/BBC%20Radio%204'
five = 'favorite/Classic%20FM'
six = 'favorite/BBC%20Radio%206%20Music'

# Command Matrix. Maps IR 'keypresses' to HTTP command URLs.

commands = { 'r'  : ('PLAY', root_url + 'play'),
             'f'  : ('PAUSE', root_url + 'pause'),
             'x'  : ('STOP', root_url + 'pause'),
             '\'' : ('OFF', root_url + 'pause'),
             '.'  : ('NEXT', root_url + 'next'),
             '1'  : ('ONE', root_url + one),
             '2'  : ('TWO', root_url + two),
             '3'  : ('THREE', root_url + three),
             '4'  : ('FOUR', root_url + four),
             '5'  : ('FIVE', root_url + five),
             '6'  : ('SIX', root_url + six),
             '7'  : ('SEVEN', root_url + ''),
             '8'  : ('EIGHT', root_url + ''),
             '9'  : ('NINE', root_url + ''),
             '0'  : ('ZERO', root_url + zero),
             '='  : ('VOLUME_UP', root_url + 'volume/+10'),
             '-'  : ('VOLUME_DOWN', root_url + 'volume/-10'),
```

```
            ','  : ('PREVIOUS', root_url + 'previous') }

# End Configuration Section ########################################

# Wait for a keypress and return
def wait_key():
    ''' Wait for a key press on the console and return it. '''
    result = None
    if os.name == 'nt': # Windows
        import msvcrt
        result = msvcrt.getch()
    else: # Unix
        import termios
        fd = sys.stdin.fileno()
        oldterm = termios.tcgetattr(fd)
        newattr = termios.tcgetattr(fd)
        newattr[3] = newattr[3] & ~termios.ICANON & ~termios.ECHO
        termios.tcsetattr(fd, termios.TCSANOW, newattr)
        try:
            result = sys.stdin.read(1)
        except IOError:
            pass
        finally:
            termios.tcsetattr(fd, termios.TCSAFLUSH, oldterm)
    return result

# Main Loop
print "Ready ..."
while True:
    code = wait_key()
    try:
        timestamp = str(datetime.datetime.now())[:-7]
        if code in commands:
            requests.get(commands[code][1])
            print timestamp + ' ' + commands[code][0] + ': ' + commands[code][1]
        else:
            print timestamp + ' ' + code + ": Unmapped Keycode, ASCII = " + str(ord(code))
    # Catch the exception  and continue if the Request fails
    except:
        print 'Error'
        pass

# End of Script
```

Note that the SONOS HTTP API can run on the same computer as the Python script, or it can run elsewhere on the network as long as it is reachable.

Running the Keypress Capture Script

The script should work under Python 2.7+ or Python 3+. It requires the ‘datetime’ and ‘requests’ packages. It has only been tested on Linux systems (Raspbian and MacOS), but in principle it should work on Windows – the only difference is in the ‘wait_key()’ function.

Since the FLIRC is like a directly-attached keyboard, the keypress capture script needs to run on the console and not in a graphical environment (in which it would be difficult to guarantee that the keyboard input goes to the correct window).

To start the script, run: **`python sonos-keypress-script.py`**

The script runs in an infinite loop, waiting for console keypresses generated by the FLIRC or by any other attached keyboard device. The latter makes it very easy to test just using a regular keyboard.

Starting the Keypress Capture Script Automatically on Reboot

It’s optional but convenient to start the keypress script automatically on reboot. On Linux systems this comprises of:

1. Automatically logging in a user at the console. Don't do this if you're worried about any security implications if folks have access to the console of the computer, and choose a user without sudo privileges.
2. Detecting whether the current shell session is a console or not, and starting the Python script if it's a console session.

Logging in Automatically at the Console

For systems using **`systemd`**, one can automatically log in a user at the console prompt as follows:

As root, create a file: **`/etc/systemd/system/getty@.service.d/customexec.conf`**

Add the following lines to the file, replacing **`<user>`** with the user to be logged in

```
[Service]
ExecStart=
ExecStart=-/sbin/agetty --noclear --autologin <user> %I $TERM
```

Starting the Python Script

Detect the console session by testing if the terminal is a TTY. At the end of the **`.bashrc`** file, insert:

```
case $(tty) in /dev/tty[0-9]*)
  echo Console Detected
  python wait-for-keypress.py
esac
```

References

[1] FLIRC Website: https://flirc.tv/more/flirc-usb
[2] SONOS HTTP API: https://github.com/jishi/node-sonos-http-api
[3] Download link for Python script: https://www.dropbox.com/s/kewzig0rw7xya7f/sonos-keypress-script.py?dl=0