Descripción

Un FIRtro simplificado para máquinas pequeñas.

Con motivo de un encargo, se pretende que sirva en un salón para oir la TV y para reproducir Spotify en modo black-box controlado desde un teléfono o tablet, con las ventajas de FIRtro: control remoto, volumen con loudness, DRC.

- Sin selector de entradas
- Las fuentes serán aplicaciones ALSA o bien la entrada analógica de la tarjeta de sonido:
    - Raspotify (librespot) → ALSA → Jack
    - Receptor de audio BlueTooth → ALSA → Jack
    - LineIN → Jack

Se descarta el uso directo de Brutefir sobre ALSA (sin Jack), el pequeño consumo de CPU de Jack compensa los posibles probemas de I/O observados con el uso de Brutefir sobre ALSA (Loopbak + tarjeta física).

DRC de bajo %CPU basado en Ecasound.

ALSA

Se configura `.asoundrc` con el plugin jack apuntando a Ecasound.

Jack

Hay que ajustar el buffer de jack al mínimo posible. OjO que el buffer de JACK debe ser igual o menor que la particion de Brutefir.

Target: latencia de la entrada analógica para ver la tele.

Brutefir

Particionado

Hay que adaptarlo al buffer de Jack que seamos capaces de conseguir.

Dither: OFF

Si se deseara y se dispusiera de una CPU suficiente, quizás convenga activar dither en Jack (audio/config).

Jack I/O

https://www.ludd.ltu.se/~torger/brutefir.html#bfio_jack

> *To avoid putting I/O-delay into the JACK graph, the JACK buffer size should be set to the same as the BruteFIR partition size. It is however possible to set the JACK buffer size to a smaller value. The I/O-delay in number of JACK buffers as seen by following JACK clients will be:*
>
> *2 * <BruteFIR partition size> / <JACK buffer size> - 2*

Note that both the JACK buffer size and BruteFIR period size is always a power of two.

Swap de disco

Desactivar el montaje de swap de disco en **`/etc/fstab`** .

O bien en sistemas Raspian modernos

`/etc/dphys-swapfile`

```
CONF_SWAPSIZE=0
```

O mejor ejecutar `sudo /sbin/dphys-swapfile uninstall`

Configurar en **`brutefir_config`**

```
lock_memory: false;
```

Versión 1.0m vs 1.0o

Parece mejor la 1.0o, aunque no es algo claro.

https://github.com/AudioHumLab/FIRtro/wiki/911-Brutefir---versiones

Ecasound

Se usa una plantilla sencilla de 4 paramétricos por canal para minimizar la carga de CPU, queda configurado en `audio/config`.

Raspotify (librespot)

Para raspberry pi (sirve en cualquier placa arm) se dispone de una “well known distro” con paquete Debian usado por muchos (Hifiberry, etc..)

https://github.com/dtcooper/raspotify

```
$ curl -sL https://dtcooper.github.io/raspotify/install.sh | sh
```

Esto instala automágicamente un paquete Debian, nos interesa solo el ejecutable que se instalará en `/usr/bin/librespot`

La configuración en `/etc/default/raspotify`

OjO se instala un servicio de sistema que desactivaremos:

```
$ sudo apt-get remove -y raspotify
```

Nosotros usaremos un watchdog a nivel de usuario `firtro` para reiniciar `librespot` en caso de fallos

`bin_custom/librespot_watchdog.sh 160 &`

```
/usr/bin/librespot --name $(hostname) --disable-audio-cache \
                   --backend alsa --device jack --bitrate $1
```

Al tercer fallo consecutivo, se optará por reiniciar la máquina.

librespot saca el audio por ALSA por el plugin jack de `.asoundrc`. Se observa que hay xruns y cortes en la conexión emergente “alsa-jack.rawjackPxxx” cuando la CPU está en uso por librespot (consume tanto como brutefir). Hay que encontrar un buffer en Jack que resulte viable para el uso de librespot.

En un FIRtro Raspberry Pi 1 puede que bitrate 320 sea difícil de descodificar sin que ocurran under runs.

Receptor de audio Bluetooth

En versiones anteriores de Raspbian se proporcionaba soporte audio BT con Pulseaudio. Afortunadamente esto ha cambiado y el soporte ahora está basado en ALSA.

El audio recibido por BT se captura con **`bluealsa`** que proporciona una tarjeta virtual ALSA. No hay un cliente nativo BT para Jack.

Se ha preparado un watchdog para establecer la escucha automágicamente cuando haya una conexión BT

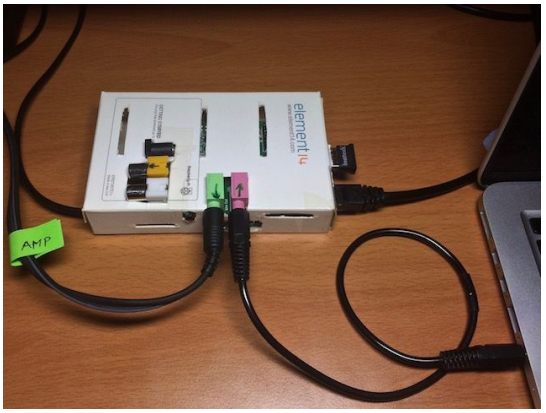
```
bin_custom/escuchaBT.py &

...
arecord -D bluealsa:HCI=hci0,DEV=54:E4:3A:1E:FC:92 \
        -r44100 -c2 -f S16_LE | aplay -D jack
...
```

Web de control

Por lo explicado más abajo, se podría inhibir la página de control de entradas en favor de unos botones en la página ‘custom’.

También conviene desactivar las gráficas de las curvas de DRC ya que los cálculos FFT son muy costosos en una Raspberry Pi 1.

Raspberry Pi 1 con Tarjeta de sonido I2S Wolfson Audio Card



Overclock

$ raspi-config → Overclock: Turbo

```
Turbo  1000MHz ARM, 500MHz core, 600MHz SDRAM, 6 overvolt
```

```
--------------------
    Thermal Zone 0
      57.838 °C
--------------------
```

Wolfson Audio Sound Card / Cirrus Logic Audio Card para Raspberry Pi

http://www.horus.com/~hias/cirrus-driver.html

1. Actualizar Raspbian `apt update && apt upgrade` para tener un kernel >= 4.x

2. Instalar drivers oficiales `sudo rpi-update`

3. Y activarlos:

   `/boot/config.txt` to enable the Cirrus Logic card driver

   ```
   dtoverlay=rpi-cirrus-wm5102
   ```

   `/etc/modprobe.d/cirrus.conf` with the following content:

   ```
   softdep arizona-spi pre: arizona-ldo1
   ```

4. Reiniciar:

   ```
   $ sudo reboot
   ```

5. Comprobar:

```
$ aplay -l
```

6. Instalar las utilidades de configuración de la tarjeta:

   ```
   $ mkdir /home/pi/bin_cirrus/
   $ wget http://www.horus.com/~hias/tmp/cirrus/cirrus-ng-scripts.tgz
   $ cd /home/pi/bin_cirrus/
   $ tar zxf ../cirrus-ng-scripts.tgz
   ```

7. Ajustes de la tarjeta

   El `alsamixer` de las tarjetas Wolfson/CirrusLogic es muy complicado, dispone de un DSP interno.

   Para seleccionar la entrada, salida, o resetar ajustes ejecutar los scripts del fabricante:

   ```
   /home/pi/bin_cirrus/
         Cirrus_listen.sh
         Playback_to_Headset.sh
         Playback_to_Lineout.sh
         Playback_to_SPDIF.sh
         Playback_to_Speakers.sh
         Record_from_DMIC.sh
   ```

```
Record_from_Headset.sh
Record_from_Linein_Micbias.sh
Record_from_Linein.sh
Record_from_SPDIF.sh
Reset_paths.sh
```

Para ajustar los niveles de entrada/salida y guardarlos para el arraque de FIRtro podemos usar estos scripts:

```
/home/pi/bin_clac_custom/clac_dBlevel.py
/home/pi/bin_clac_custom/clac_dBlevel_headset.py
```

Guardamos los ajustes para el arranque de FIRtro

```
$ alsactl -f /home/audio/asound.RPiCirrus store RPiCirrus
```

Software necesario:

Bluetooth

```
$ sudo apt install bluealsa
```

Archivos ad-hoc:

Ver en https://github.com/Rsantct/FIRtro-light

```
/etc/sudoers.d/020_firtro

/etc/rc.local #con el patch y los scripts de spotify y bluetooth

/home/firtro/.asoundrc

/home/firtro/bin_custom/

      patch_firtro.sh
      restart_jack_brutefir.sh
      restart_jack_brutefir_ecasound.sh
      librespot_watchdog.sh
      escuchaBT.py

/home/firtro/bin/

      webcustombutton_7.sh
      webcustombutton_8.sh
      webcustombutton_8.sh

/home/www/config/config.ini

/home/audio/inputs
```

Prestaciones:

librespot OK aunque en RPI1 funciona mejor con --bitrate 160

LineIn necesita ajustar jackd -p 1024 o menos

bluetooth OK

Si el ajuste --period (frames per period) de jackd es inferior a 4096, se observan underruns desde el plugin jack de .asoundrc cuando es usado por librespot debido a su alto consumo CPU. Otros usos como por ejemplo el receptor BT (bluealsa) no presentan problemas.

Como se necesita --period pequeño para escuchar material audiovisual por la entrada analógica LineIn, preparamos unos botones de la web de control para relanzar jack/brutefir/ecasound, ajustando --period en Jack y filter_lenght en Brutefir.

`bin_custom/restart_jack_brutefir_ecasound.sh tamaño_buffer`

Alternativamente, se ha intentado ajustar el slave del pcm.jack de `.asoundrc` con un valor period_time alto, pero no funciona :-/

```
pcm.jack {
    type plug
    slave {
        pcm         "rawjack"
        period_time 8192
    }
```