Aplicaciones de audio para tarjetas de desarrollo

Procesador de efectos de código abierto

Introducción

La introducción de nuevas plataformas para el desarrollo tecnológico abre la posibilidad de usar dichas herramientas para la creación de proyectos artísticos con nuevas capacidades técnicas, es por ello que ante la creciente oferta de microcomputadoras a manera de tarjetas de desarrollo es posible utilizar dichos dispositivos para la generación y/o procesamiento de audio mediante herramientas abiertas a su distribución y libre modificación

Objetivos

Por medio de este tutorial se detalla en los procesos técnicos necesarios para la generación de un proyecto de audio con la tarjeta de desarrollo Beaglebone Black, se implementará una solución simple a necesidades de entrada y salida de audio, así como de la creación de una interfaz física para el control de parámetros del audio resultante.

Descripción

Para cubrir los aspectos clave que nos permiten trabajar con audio en tarjetas de desarrollo usaremos un proyecto sencillo planteando un procesador de efectos de audio en donde se pueda conectar un instrumento musical, procesar su sonido y modificar la forma en que se está afectando la entrada de sonido; para esta implementación se usará principalmente el lenguaje gráfico Pure Data (PD) ya que debido a su portabilidad se puede incorporar a esta plataforma de una manera sencilla, además se usarán scripts en el lenguaje Python para crear una interconexión con los controles externos a nuestro procesador de audio en PD, como última instancia se generarán las rutinas necesarias para que el dispositivo se autónomo y realice el procesamiento sin necesidad de tener una pantalla o estar conectado a otra computadora.Resulta importante aclarar que este dispositivo tiene como característica el poder modificarse hacia las necesidades particulares de otros proyectos, debido a su naturaleza de software y hardware abierto.

Materiales

Beaglebone Black
Tarjeta SD Clase 10 16 GB Kingston
Tarjeta de audio USB Syba
Protoboard
Resistencia 1kOhm
Potenciometro 10kOhm
Switch

Desarrollo

Inicializando BeagleBone Black

La tarjeta BeagleBone Black funciona mediante un sistema operativo que inicializa desde una tarjeta SD, por lo que es necesario descargar una imagen de un sistema operativo que en nuestro caso será Debian, este puede ser descargado de la siguiente página donde buscaremos la última versión de Debian for Beaglebone:

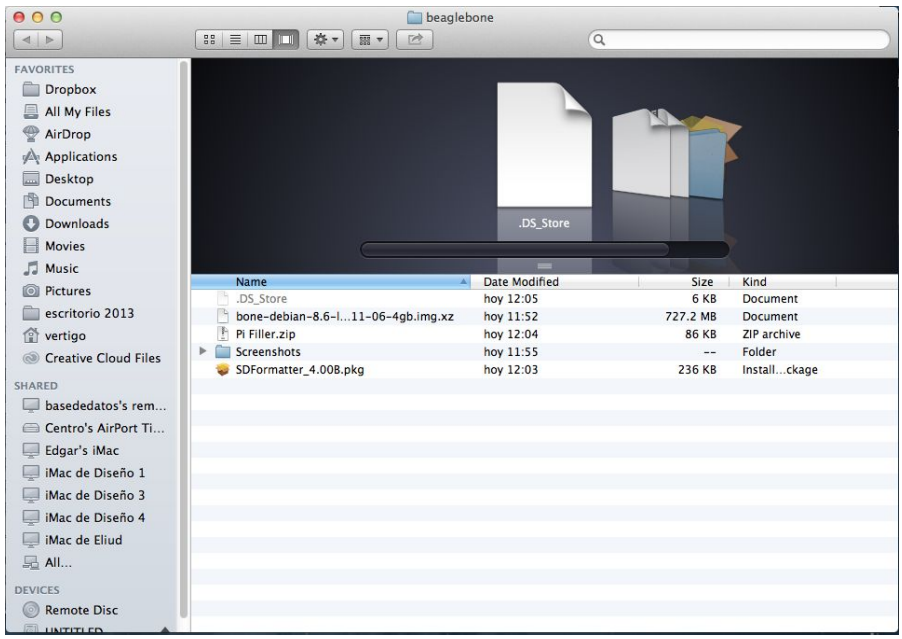
https://beagleboard.org/latest-images

Debian es una distribución de Linux que mediante un programa llamado Pi Filler se quemará en la tarjeta SD como imagen de disco, en el siguiente link puede conseguirse dicho programa:

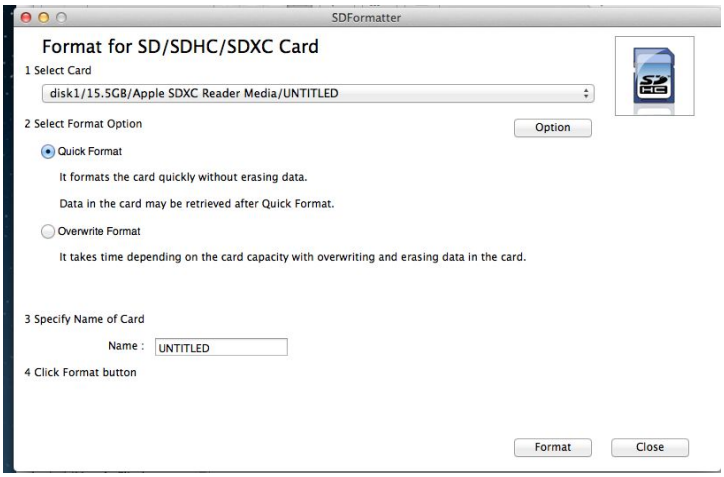
http://ivanx.com/raspberrypi/

Una recomendación es formatear la tarjeta SD con el programa SD Formatter para asegurarnos que se encuentre en el formato correcto, este se puede descargar de:

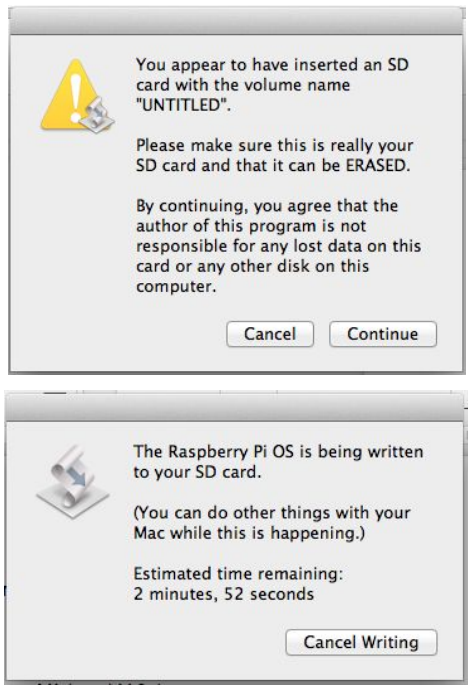
https://www.sdcard.org/downloads/formatter_4/



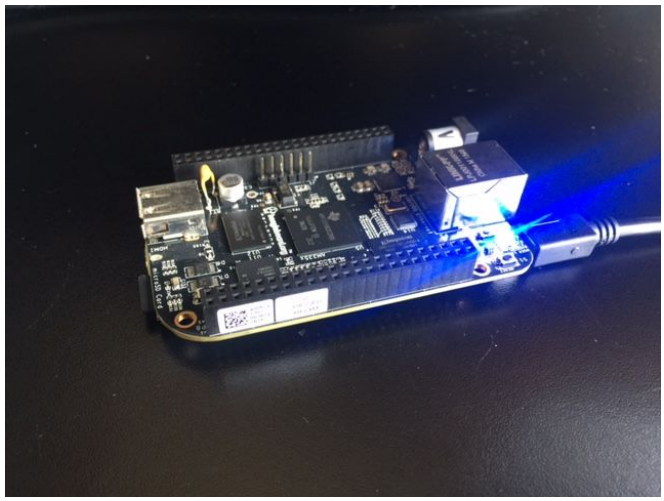
Antes de comenzar formatear la tarjeta SD con SD Formatter utilizando la opción de Quick Format



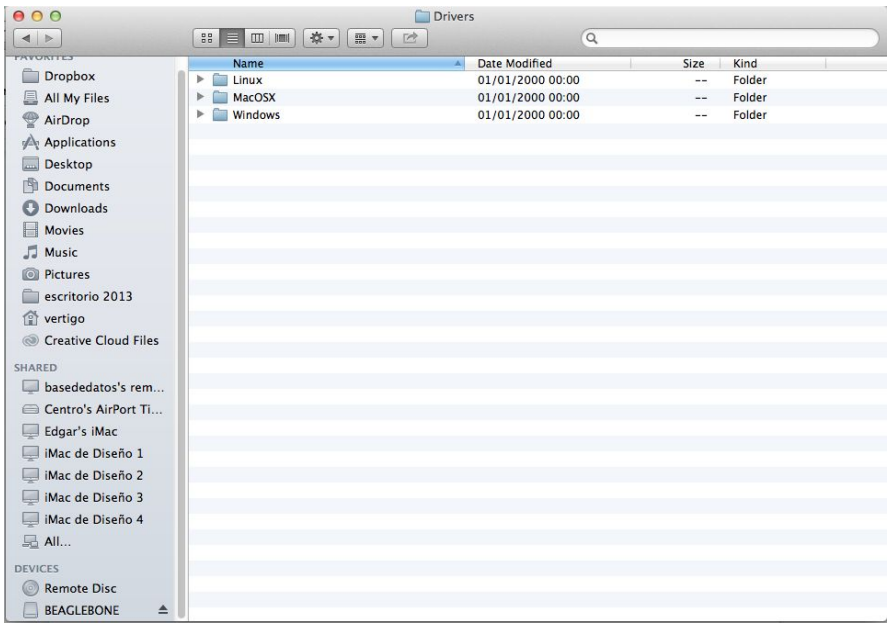
Procede instalando y abriendo Pi Filler, dónde nos pedirá seleccionar la imagen de Debian que hemos descargado previamente, una vez seleccionada seguiremos las instrucciones que nos va presentando en pantalla Pi Filler



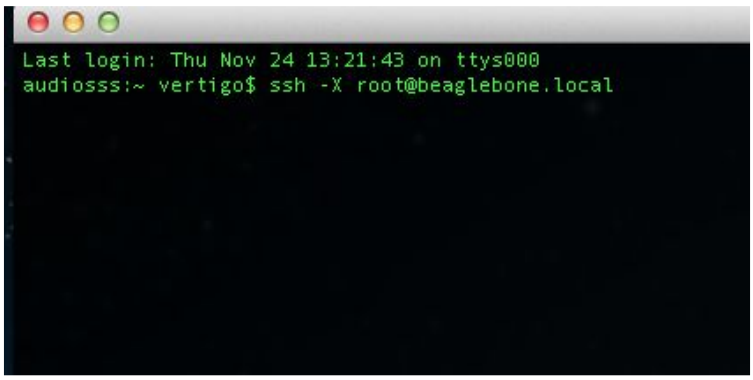
Una vez lista la tarjeta de memoria insertar a la BeagleBone y conectar a una computadora vía USB



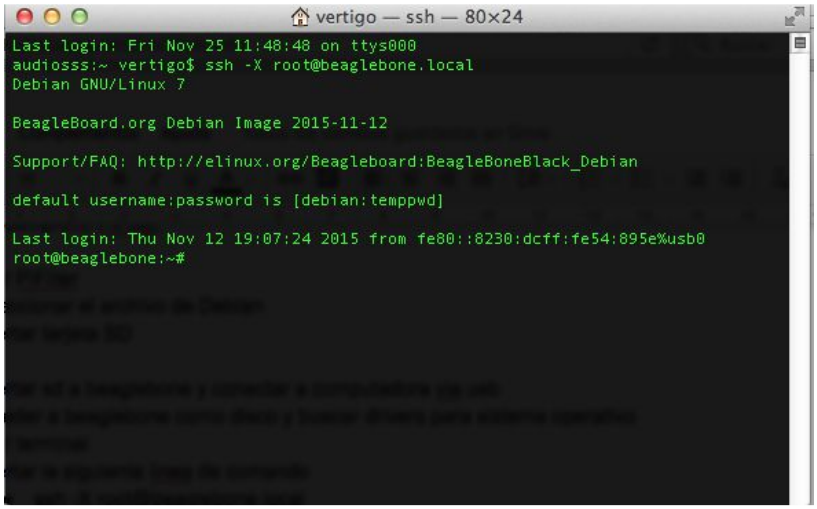
Accedemos a la Beaglebone como un disco extraible donde encontraremos los drivers necesarios para nuestro sistema operativo, en este caso MacOSX



Una vez instalados dichos drivers se procederá a abrir la terminal e insertar la siguiente línea de comando la cual nos permitirá comunicarnos vía remota a la BeagleBone



Seguramente se pedirá autorización para el acceso a este dispositivo, después habremos ingresado al mismo sin necesidad de una contraseña



Dependiendo de cuanta memoria posea la tarjeta SD que usamos, tendremos que expandir el sistema de archivos accediendo a la tabla de particiones mediante:

```
fdisk /dev/mmcblk0
```

Una vez dentro de esta utilidad presionaremos `d` para borrar la partición y luego `2` para seleccionar la segunda, a continuación agregaremos una nueva partición con `n` y `p` indicando que es primaria, usamos `2` para crear dicho número de partición y acto seguido nos pedirá ingresar el tamaño de los bloques de memoria para los que dejaremos los que se indican como preestablecidos, guardar los cambios presionando `w`.

Hecha la partición reiniciamos el dispositivo con el siguiente comando:

```
shutdown -r now
```

Ahora debemos llenar la imagen con la tabla de partición extendida:

```
resize2fs /dev/mmcblk0p2
```

Y reiniciamos una vez más. Una vez reiniciada tendremos una Beaglebone funcional para usar como una microcomputadora.

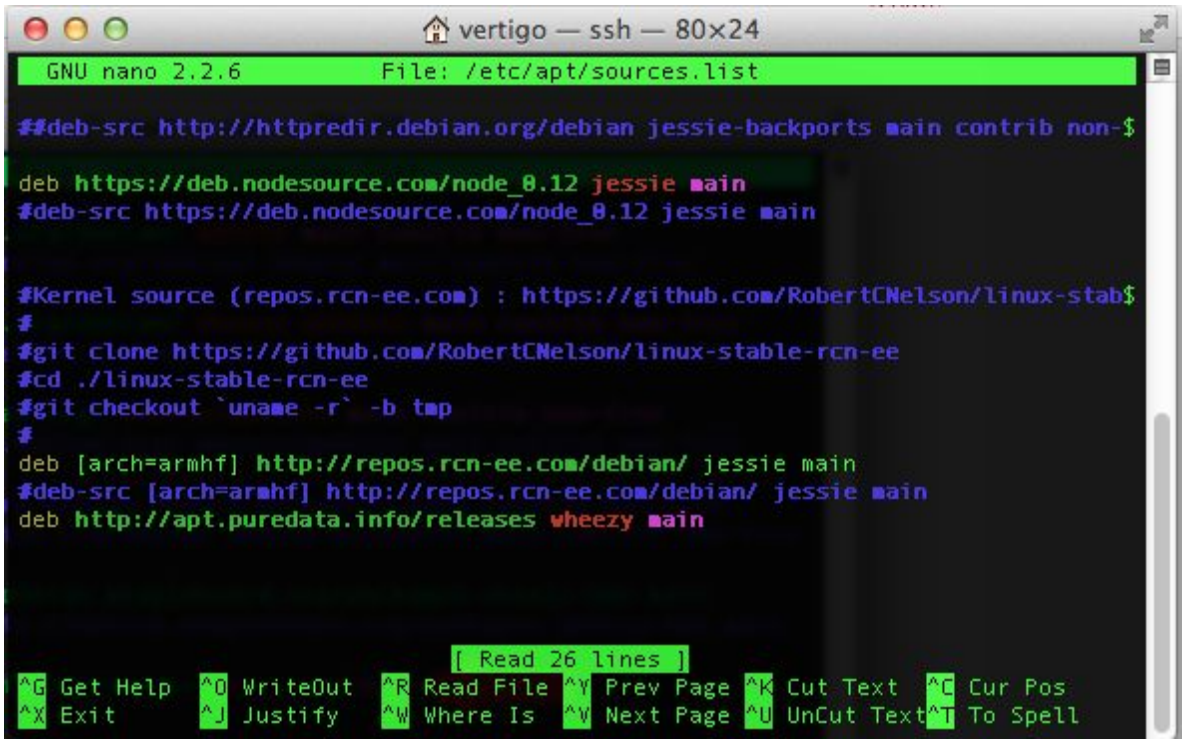
Instalando programas a usar

Como se mencionó anteriormente se usará principalmente Pure Data para el audio y scripts de Python para la interfaz física.
El sistema de actualización e instalación principal en Debian es apt-get por lo cual debemos editar la lista de fuentes por medio de un editor de texto sencillo que desde este momento usaremos de forma recurrente llamado nano.

Con la siguiente línea abrimos la lista de fuentes:
`nano /etc/apt/sources.list`

Agregamos al final del archivo:

```
deb http://apt.puredata.info/releases wheezy main
```



Para guardar los cambios se usa Ctrl-O y Ctrl-X para salir, una vez guardado este documento procederemos a actualizar la hora y fecha de nuestra Beaglebone mediante el comando:

```
sudo ntpdate pool.ntp.org
```

Ahora actualizaremos el repositorio de apt-get para después instalar PD y herramientas de Python y ALSA que es el servidor de audio para la BeagleBone:

```
apt-get update
```

```
apt-get install puredata pd-extended build-essential python-dev
python-pip -y --force-yes
```

```
sudo apt-get install alsa-base alsa-utils
```

Y finalmente instalaremos la biblioteca de GPIO para BeagleBone de Adafruit

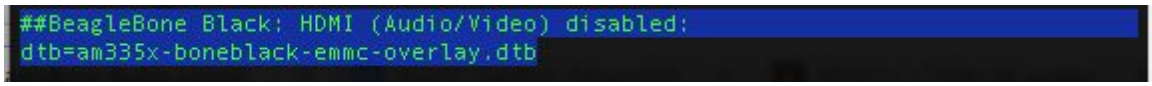
```
sudo pip install Adafruit_BBIO
```

Configuraciones de audio y video

La tarjeta BeagleBone Black no cuenta con una salida analógica de sonido, sin embargo lo manda por medio de su puerto de HDMI, para poder acceder a él de manera separada deshabilitamos este puerto y configuramos la salida de sonido por la tarjeta USB
Para lo anterior accederemos a un archivo de configuración; para poder desplazarnos entre carpetas de nuestro sistema usaremos el comando `cd` seguido de la dirección de la carpeta que queramos acceder, en este caso es:

```
cd /boot/nano
```

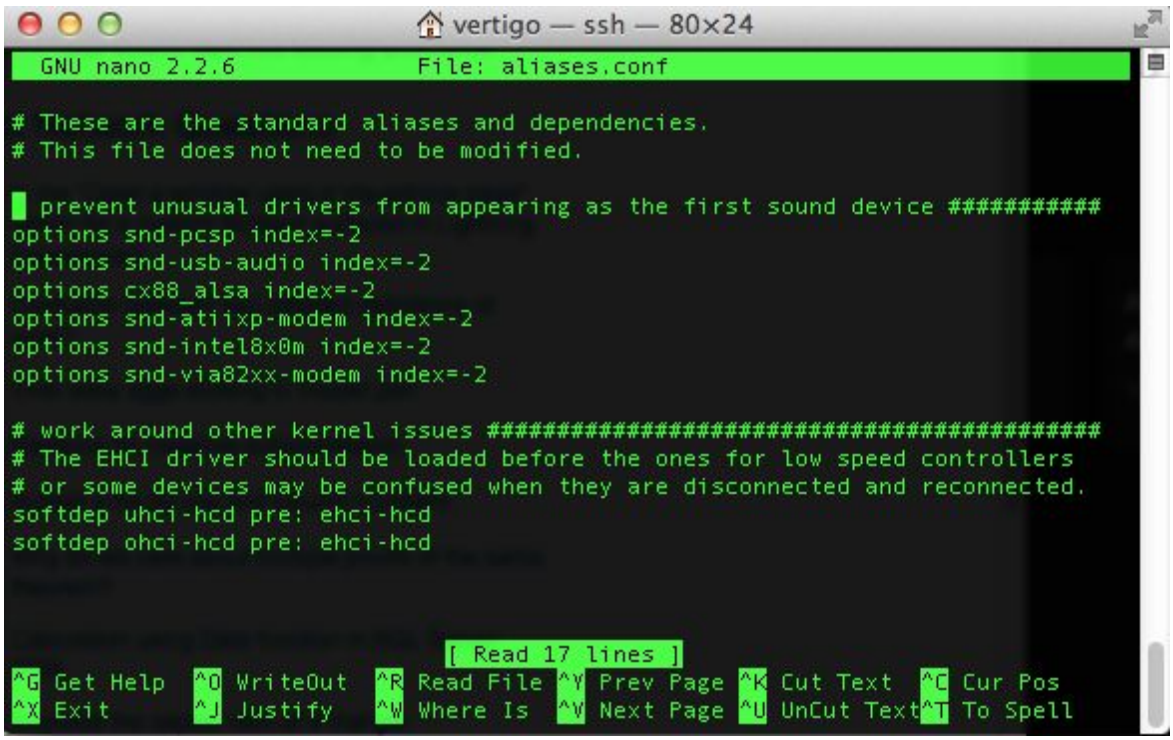
Donde se encuentra alojado el archivo `uEnv.txt` que abriremos con `nano`, una vez en el editor vamos a quitar el signo `#` a la línea de código indicada en la imágen lo que habilita la ejecución de dicho comando



Ahora buscaremos el archivo de configuración para dispositivos de sonido

```
/lib/modprobe.d/aliases.conf
```

Al abrirlo con nuestro editor `nano` encontraremos lo siguiente:



Cambiamos el índice de la siguiente línea a `-1`

```
options snd-usb-audio index=-2
```

Guardamos y cerramos el archivo para reiniciar la BeagleBone con la tarjeta de audio ya conectada; una vez reiniciada podemos hacer pruebas para saber si la tarjeta está siendo detectada con:

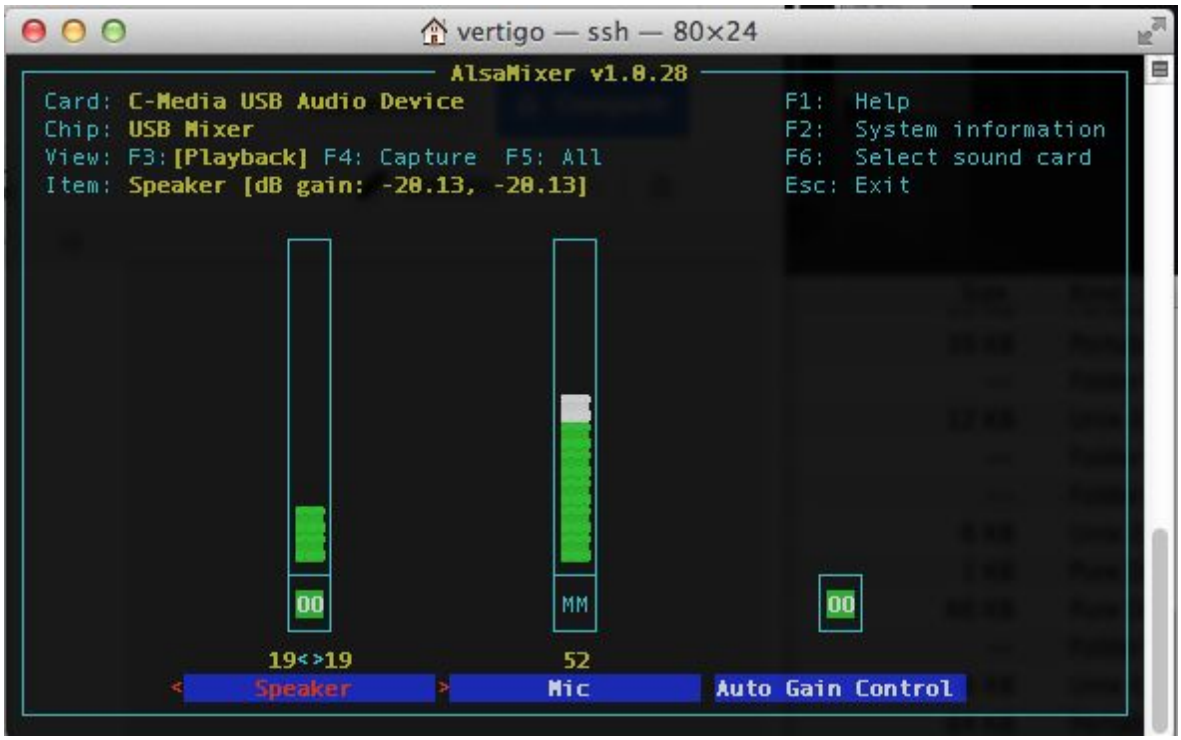
```
aplay -l
```

Que enlistarà todos los dispositivos de audio

```
root@beaglebone:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: Device [C-Media USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Es posible ajustar los niveles de entrada y salida mediante

```
alsamixer
```

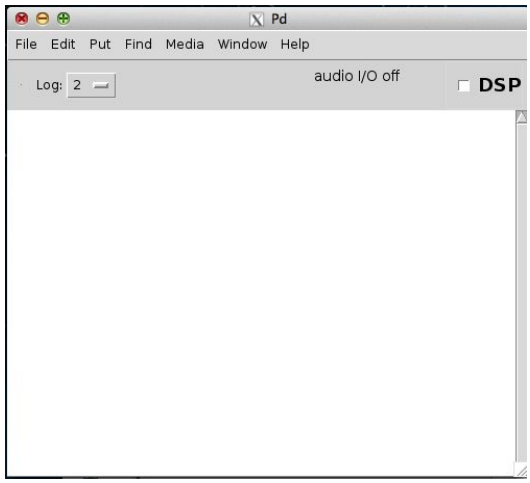


Y guardar los valores con la línea

```
alsactl store
```

Creando procesamiento de audio

Para el manejo de la salida y entrada de audio abriremos PD, guardando en un patch el proceso que se busca realizar con el mismo; en la consola ejecutaremos `puredata` y una interfaz gráfica aparecerá en pantalla, para esto se debe instalar en la computadora XQuartz.



Esta es la interfaz principal de PD y la consola del mismo dónde podemos monitorear los procesos y errores que ejecutemos de manera gráfica dentro de nuestro patch.

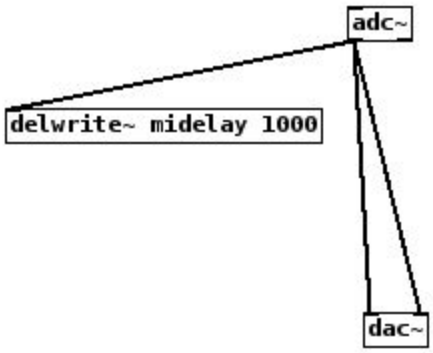
Se puede iniciar un nuevo patch tecleando Ctrl+N, donde se programará todo nuestro procesamiento de audio.

De manera muy general PD es un entorno en el cuál objetos interaccionan con mensajes u otros objetos principalmente para el manejo y modificación de audio y datos, desde el menú Media/Test audio and MIDI es posible hacer una prueba conectando audífonos o un sistema de sonido a la tarjeta usb para comprobar que la configuración fue correcta.

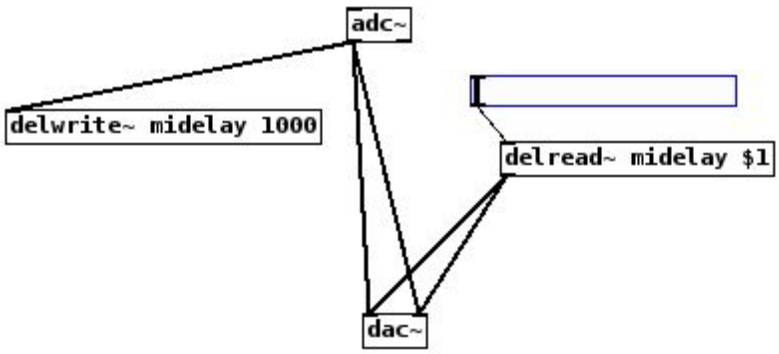
A continuación se detallará sobre un programa simple para procesar sonido con nuestro dispositivo, sin embargo está abierto a complejizarse o cambiar la funcionalidad basándose en los principios básicos que se mostrarán a continuación:

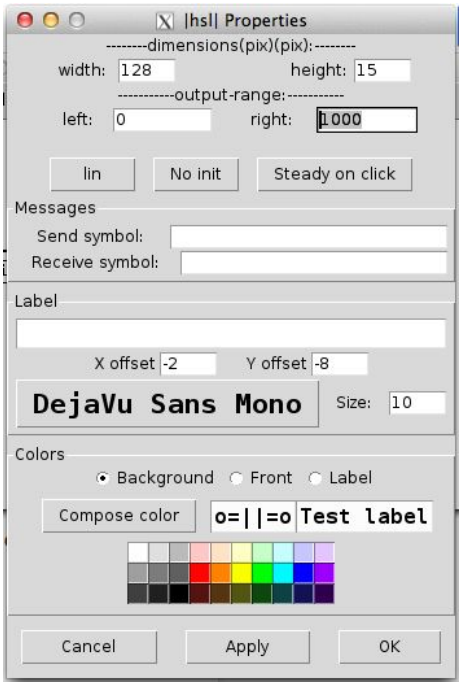
Nuestro primer objeto es la entrada del convertidor analógico a digital que en sus salidas nos dará la línea de audio que conectemos a la tarjeta USB

Agregaremos el objeto que reproducirá en nuestra salida de audio toda señal que mandemos, en este caso lo conectaremos directo a nuestra entrada.

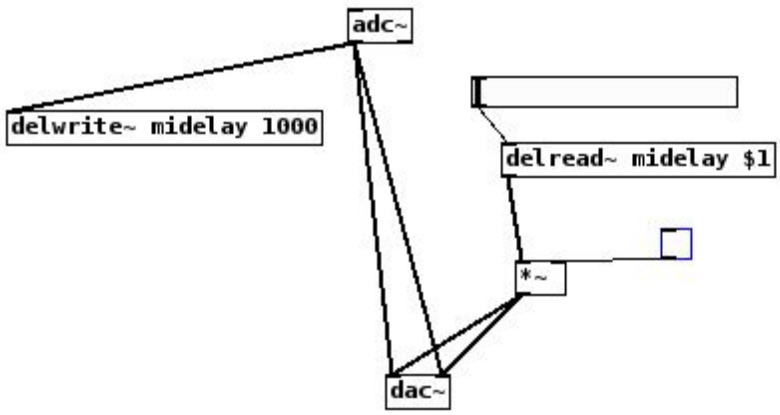


Por medio de este nuevo objeto escribiremos una línea de delay a partir de la señal entrante

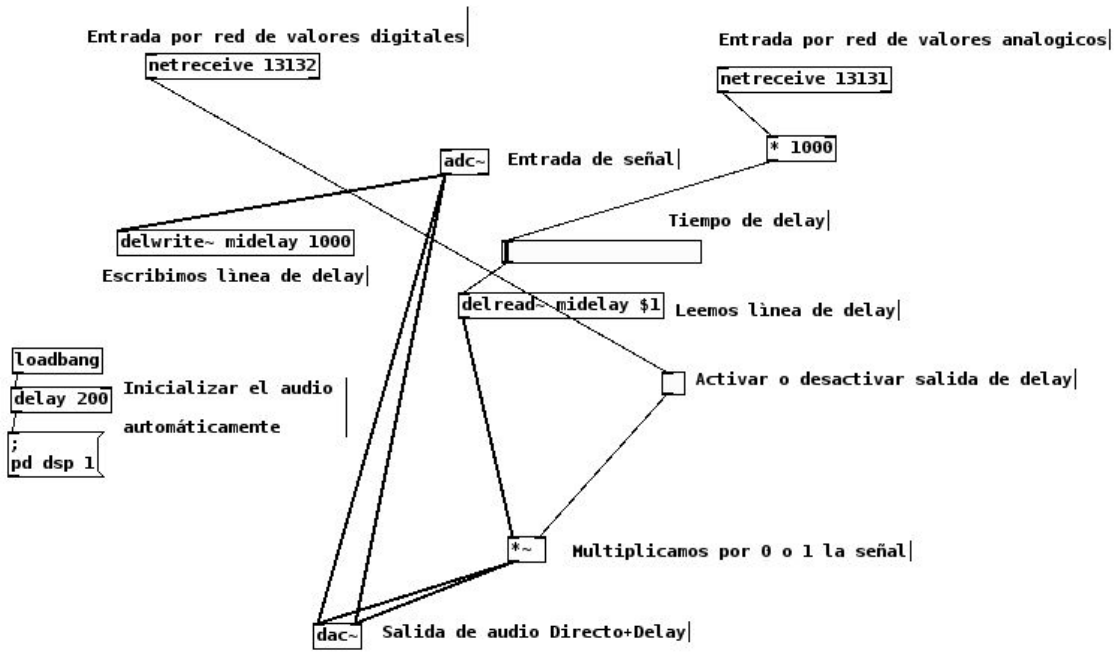




Hacemos la lectura de esta línea de delay que mandamos hacía la salida de audio, el tiempo de retraso lo modulamos a partir de un slider que mapeamos de 0 a 1000 milisegundos por medio de su ventana de propiedades al hacer click derecho sobre el.



Añadiremos una multiplicación antes de la salida de audio para poder multiplicar por 0 o 1 con un objeto toggle, de esta forma podremos apagar o prender la salida del delay

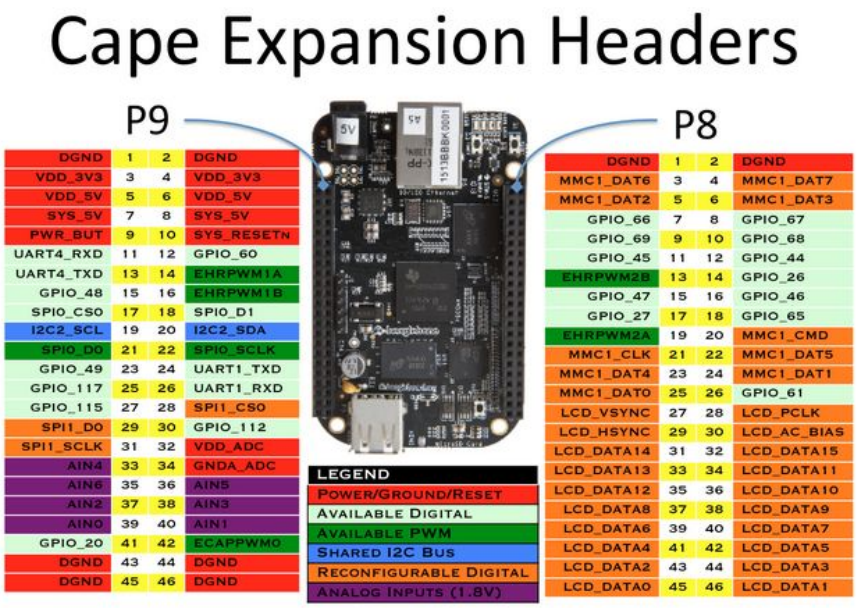


Nuestro patch final contiene los objetos que hacen la conexión por red a la interfaz que a continuación esquematizamos, estos hacen una conexión por TCP y se ha dejado una para los valores digitales en el puerto 13132 que va directo al toggle para activar o desactivar y otra para lecturas analógicas en el 13131 que se multiplica para mapear el valor a nuestro slider de tiempo de retraso. También se agrega un inicializador del audio de PD con un pequeño retraso para hacer el proceso en automático.

Guardaremos este patch bajo el nombre delay.pd para acceder a él más adelante.

Construyendo la interfaz física

La interfaz física aquí mostrada es muy básica pero demuestra los conceptos fundamentales para tener una señal analógica y otra discreta que pueden modificar parámetros del patch de PD; para generarla se accede al GPIO de la BeagleBone por medio de Python y se hará la lectura de dos puertos. A continuación podemos ver un esquema de los puertos disponibles y sus funciones:



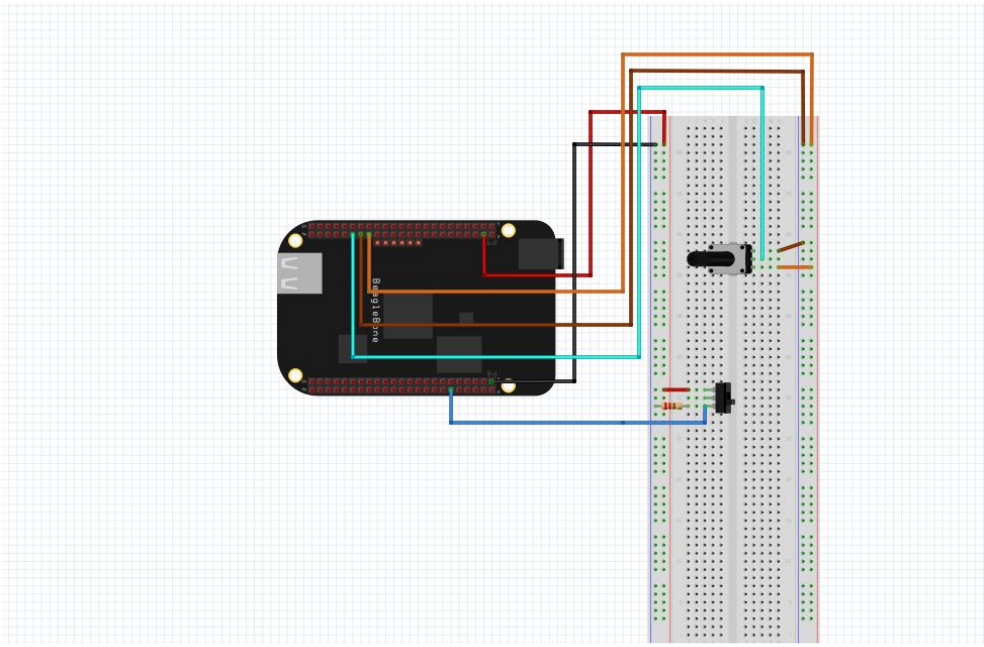
Para generar los scripts usaremos el editor `nano` para crear un nuevo archivo, a continuación se muestran los códigos para cada una de las lecturas:

```
#Importamos bibliotecas a usar
import subprocess
import time
import socket
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.ADC as ADC
import time

#Inicializamos el convertidor analogico a digital
ADC.setup()

#Definimos el puerto de comunicacion por red
TCP_IP='127.0.0.1'
TCP_PORT=13131
BUFFER_SIZE=1024
#Se hace la conexion
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((TCP_IP,TCP_PORT))

#Definimos la funcion para mandar valores
def mandar(val):
        s.send('%s;'%(val))


#Hacemos un ciclo para mandar constantemente los valores leidos
while True:
#Se lee el valor
        valor=ADC.read("P9_36")
#Mandamos el valor
        mandar(valor)
#Esperamos un segundo
        time.sleep(.5)

#Se cierra la conexion
s.close()
```

```
#Importamos bibliotecas a usar
import subprocess
import time
import socket
import Adafruit_BBIO.GPIO as GPIO

#Inicializamos el pin de GPIO
GPIO.setup("GPIO1_12",GPIO.IN)
GPIO.add_event_detect("GPIO1_12",GPIO.BOTH)

#Definimos el puerto para conexion de red
TCP_IP='127.0.0.1'
TCP_PORT=13132
BUFFER_SIZE=1024
#Realizamos conexion
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((TCP_IP,TCP_PORT))

#Definimos funcion para mandar bang
def mandarbang():
        s.send("bang;")

#Hacemos un loop para enviar cada que se detecta un cambio en el GPIO
while True:
        if GPIO.event_detected("GPIO1_12"):
                mandarbang()

#sE CIERRA LA CONEXION
s.close()
```

El primer script lo guardaremos con Ctrl+O nombrandolo pyADC.py y el segundo pyGPIO.py

Una vez guardados procederemos a implementar el circuito que se comunicará con los puertos GPIO, a continuación un esquemático realizado en Fritzing:

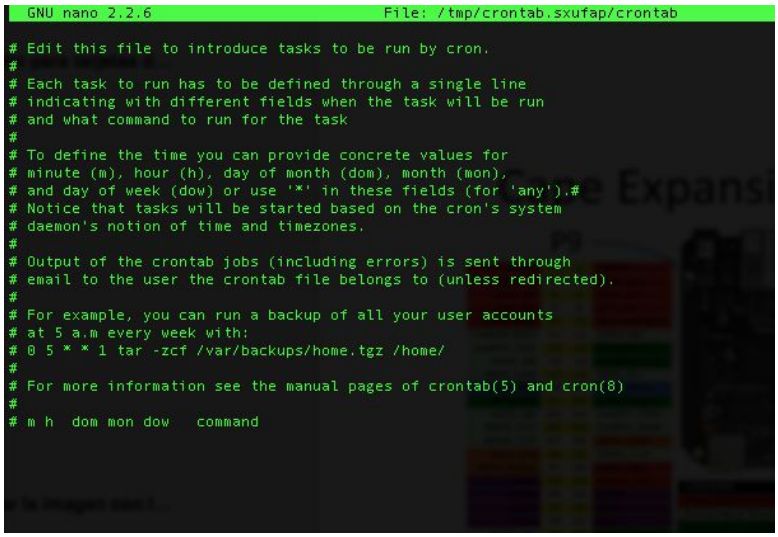


Para la lectura analógica usaremos el Pin 9-36, el cual irá conectado a la salida de un potenciómetro que a su vez está alimentado por la tierra y el voltaje directo del ADC; a su vez se utiliza el Pin 8-12 para la salida de un switch que a su vez está conectado a una resistencia pull-down y alimentado por la tierra y voltaje del GPIO.

Automatizando procesos

Para finalizar realizaremos los scripts que se ejecutarán para inicializar el dispositivo sin necesidad de una computadora externa, esto se va a realizar por medio del programa `crontab` que nos permite agendar ejecución de líneas de comandos , procederemos a abrir la herramienta:

```
crontab -e
```



Al final de este archivo escribiremos:

```
@reboot sh /root/startup.sh
```

Este archivo va a ejecutar cada que reinicie la BeagleBone un programa sh que a continuación crearemos, dentro de este programa daremos inicio a PD y los scripts de Python necesarios para nuestro proyecto; iremos a el directorio principal mediante `cd` y abriremos nano para crear un nuevo archivo que guardaremos como startup.sh

Una vez guardado este archivo podemos probar a reiniciar la tarjeta y conectar un instrumento de entrada vía la tarjeta de sonido y un amplificador a la salida,recordar que podemos alimentar la BeagleBone desde un cargador externo dando independencia el procesador de efectos, quedando concluido nuestro proyecto.