MÓDULO 10: PURE DATA

Aprenderemos qué son las rampas potenciales, filtros y crearemos nuestra primera caja de ritmos

Rampas potenciales: pow y sqrt

¿Os acordáis de las rampas que hemos visto para suavizar los cambios de volumen y los envelopes en las lecciones. Practica 2: nuestro pr... | Librería CATEDU y Envelope | Librería CATEDU? Vamos a ver dos objetos que nos permitirán transformar esas rampas lineales en progresiones potenciales. Abrir el patch *pow-sqrt.pd*

El objeto "**pow~**" **elevara la señal** recibida en el inlet izquierdo a el valor o valores recibidos en el inlet derecho o indicados en su argumento.

El objeto "**sqrt~**" devolverá por su outlet la **raíz cuadra** de la señal recibida en su único inlet. Este objeto no tiene argumento.

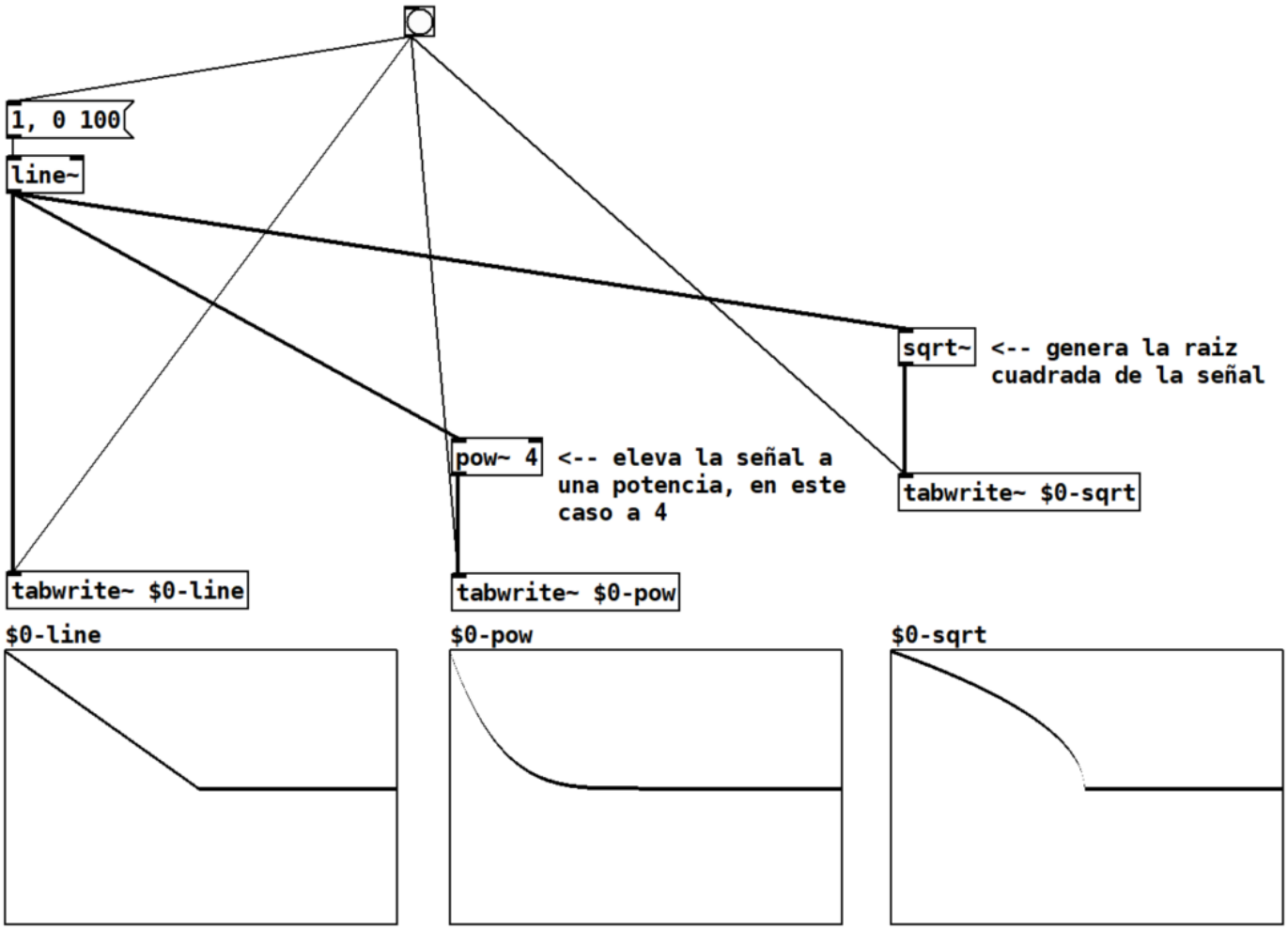


Figura 1. patch *pow-sqrt.pd*

Figuras:

Figura 1. patch *pow-sqrt.pd*

Noise o Ruido

"Los estudiosos de la música definen “ruido” como ondas irregulares sin un tono definido; en ingeniería se define como una señal que interfiere con la detección o calidad de otra señal (ASHA, 2012). En acústica, se entiende por ruido aquel sonido no deseado (Jaramillo, 2007) y comúnmente el ruido, en inglés noise, se utiliza para referirse a un sonido desagradable. Este término aparece también frecuentemente asociado a ideas de comunicación como distorsionador de aquello que se quiere transmitir." (del Río, 2019)

Ruido en el ámbito del Arte Sonoro

"Una de las primeras aproximaciones al concepto de ruido en el contexto artístico es la que da **Luigi Russolo** (1913), que fecha el nacimiento del ruido en el **siglo XIX** con la llegada de la **revolución industrial** y la aparición de las **máquinas**. Russolo queda fascinado por esos nuevos sonidos extraños, disonantes e intensos que tornan exhausta y monótona la tradición musical clásica (Russolo, 1913). El oído quiere más y más extremo, y en este autor se aprecia el anhelo de una experiencia sonora abrumadora." (del Río, 2019)

https://www.youtube.com/embed/IC3KMbSkYNI

Luigi Russolo, 1913-1914

"Si en el siglo XIX el avance tecnológico que revolucionó la economía, sociedad y el modo de producción fue la máquina, en el **siglo XXI** los avances tecnológicos correspondientes serían la **tecnología digital, la computación, Internet**, etc. En el S XIX las máquinas mecánicas reemplazaban las funciones motoras de un cuerpo humano, y en el S XXI las máquinas digitales reemplazan funciones de a la mente humana. Las máquinas del siglo XIX reemplazaron nuestros cuerpos y las del SXXI remplazan también nuestras mentes y nos ofrecen un nuevo espacio virtual en el que habitar. Las tecnologías digitales son para nosotros lo que las máquinas fueron para Luigi Russolo, el más reciente estado de los avances tecnológicos que transformó la sociedad del siglo XIX. Las máquinas de nuestro tiempo son eléctricas, digitales, dispositivos e interfaces omnipresentes conectadas unas con otras a
través de la mega estructura de telecomunicaciones (Andersen, Pold, 2018). Las **máquinas** del **siglo XXI** se han vuelto más **silenciosas e invisibles."** (del Río, 2019)

"Salome Voegelin (2010) apunta que el ruido de las máquinas de Luigi reflejaba el movimiento de masas y el progreso, y que el ruido de hoy refleja una **privada** y **aislada quietud**. La principal noción de ruido que trabaja esta autora es la de
aquel sonido que atrapa al cuerpo que escucha y lo aísla incluso de sí mismo, “Sound is noisy when it deafens my ears to anything but itself” [El sonido es ruidoso cuando ensordece mis oídos de todo excepto de sí mismo] (Voegelin, 2010, p.44)." (del Río, 2019)

"Para Salome Voegelin (2010) existen diferentes tipos de ruido (formas y razones), como “el ruido de lo indeseado” que ejemplifica con la música desagradable de sus vecinos a la que reacciona poniendo música en su casa con la intención de crear una barrera sonora que neutralice el sonido indeseado; “el ruido en el contexto del arte sonoro”; “el ruido de una rave que toma el control del cuerpo” o “el ruido de una voz que renuncia a la comunicación semántica”, entre otros." (del Río, 2019)

"Contextualizando el ruido en la era postmoderna, Voegelin (2010, pp. 62-63) lo presenta como lo que la postmodernidad es a la modernidad, el ruido de la heterogeneidad, trabajando fuera y a través de disciplinas. Una postmodernidad que rechaza las ideas de totalidad y unidad legado de la modernidad, la postmodernidad abraza la fragmentación, la diversidad, la subjetividad del individuo, al cuerpo y su experiencia, integrándolo en el juego artístico." (del Río, 2019)

> “
> Noise breaks with the language base… Noise can only find its way to language in the acknowledgment that it can't. The thing discussed is the body that heard not the work that played. Noise forces the listening subject into the critical ring and turns the work into moments of experience. And that is the true criticality of noise. [El ruido rompe con la base lingüística… El ruido solo puede encontrar su camino en el lenguaje en el reconocimiento de que no puede. Lo que se discute es el cuerpo que escucha no la pieza que suena. El ruido fuerza al sujeto que escucha dentro de un anillo crítico y convierte la pieza en momentos de experiencia. Y esta es la verdadera crítica del ruido.] (Voegelin, 2010, p.65).

Por tanto, ¿podríamos considerar el noise, el ruido, la banda sonora ideal de la era internet, conectividad constante y saturación aislante?

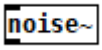
https://www.youtube.com/embed/fR_8gpJCT4I?start=254

Merzbow, 2014

El ruido en pure data

El ruido del que ahora vamos a hablar se ajustaría a la definición de ondas irregulares sin un tono definido. Vamos a ver con que objeto podemos generar ese tipo de señal en Pure data.

En Pd vamos a generar ruido con el objeto "**noise~**". El ruido que nos proporciona este objeto es **ruido blanco**, esto quiere decir, que es un ruido que incluye de manera aleatoria todas las frecuencias por igual. La señal contiene todas las frecuencias y todas ellas muestran la misma potencia.



Existen otros tipos de ruidos en los que determinados rangos de frecuencias tienen más probabilidad de estar presentes ya que la generación de ruido es un proceso aleatorio que tienen una componente estadística.

Vamos a ver un ejemplo para entender mejor esto: Si mi prima la del pueblo compra 10 décimos de lotería para el sorteo del niño va a tener más probabilidades de que le toque algún premio que yo, que solo he comprado 1 décimo. En el **ruido blanco** todas las frecuencias han comprado 10 décimos de lotería y tienen la misma probabilidad de formar parte de la señal generada. En el marron, por ejemplo, cuanto más alta es una frecuencia, aparecerá representada con menor intensidad. Aqui podéis escuchar varios colores de ruidos:

https://www.youtube.com/embed/s3qv_twE6kY

Volvamos al objeto "**noise~**" que como hemos dicho antes va a generar una señal de ruido blanco que saldrá por su único outlet. El inlet de este objeto nos va a permitir configurar la semilla, en inglés "**seed**",

¿Y qué es eso de la semilla? En computación no existe ningún proceso puramente aleatorio y estos objetos que generan patrones aleatorios son en realidad algoritmos que toman un valor inicial a partir del cual calculan una serie de números pseudoaleatorios. La semilla o seed es el valor inicial a partir del cual se calcularán esos valores. Cada vez que enviemos al objeto "noise~" o al objeto "random" la misma semilla, el cálculo comenzara de nuevo, generando la misma secuencia de

val drum ... | Librería CATEDU.

Poc

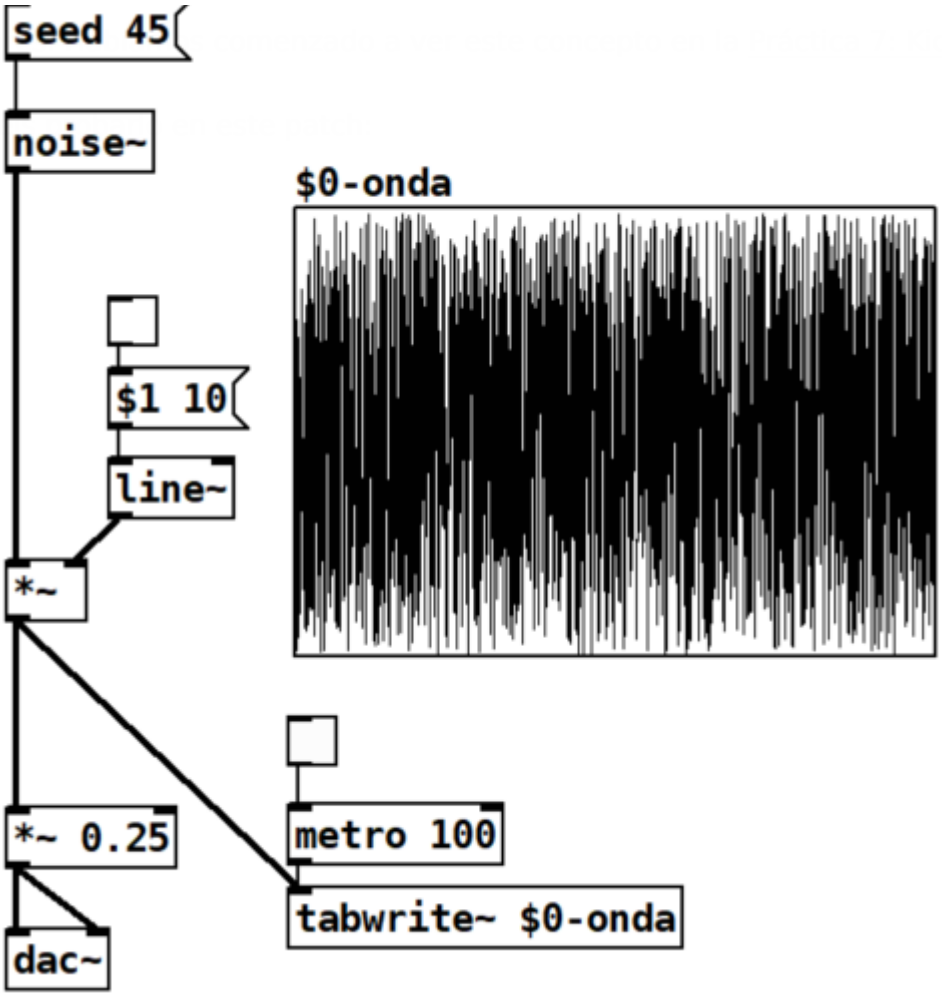


Figura 1. patch *Noise.pd*

Cuando creemos sonidos sintéticos el uso del ruido nos va a permitir crear sonidos más próximos a los sonidos que existen en el mundo acústico/material, como por ejemplo el sonido de los platillos de una batería. Recordar que las ondas sinusoidales puras prácticamente no existen en la naturaleza (Las ondas y el sonido | Librería CATEDU).

Figuras:

Figura 1. patch *Noise.pd*

Referencias:

Andersen, C. U., & Pold, S. B. (2018). *The metainterface: The art of platforms, cities, and clouds*. MIT Press.

ASHA (2012). Serie informativa de audiología. El Ruido. Recuperado de https://www.asha.org/uploadedFiles/AIS-El-Ruido.pdf

del Río, J. (2019) *El ruido como elemento expresivo en la interfaz mediante el coil o inductor como meta-interfaz en el móvil y en el ratón informático. Aplicación práctica performativa (2016-2019)* [Trabajo Final de Máster, Universitat Politècnica de València]. RiuNet. https://riunet.upv.es/handle/10251/129810

Jaramillo, A. M. J. (2007). *Acústica: la ciencia del sonido*. ITM.

Peters, J. D. (1999). *Speaking into the Air: A History of the Idea of Communication*. Chicago: University of Chicago Press.

Russolo, L. (2004). The art of noises: Futurist manifesto. *Audio culture: Readings in modern music*, 10-14.

Voegelin, S. (2010). *Listening to noise and silence: Towards a philosophy of sound art*. USA: Bloomsbury Publishing.

Filtros básicos: low pass, high pass, band pass

Low pass

Este filtro solo deja pasar **frecuencias bajas** a partir de un valor determinado. Este valor se denomina frecuencia cutoff, o **frecuencia de corte**. Las frecuencias superiores al valor del cutoff serán reducidas o eliminadas. En Pure data, es el objeto "**lop~**". Podemos configurar la **frecuencia de corte** el el **argumento** o a traves del **inlet derecho**. En el **inlet izquierdo** recibiremos la **señal** que vamos a filtrar.

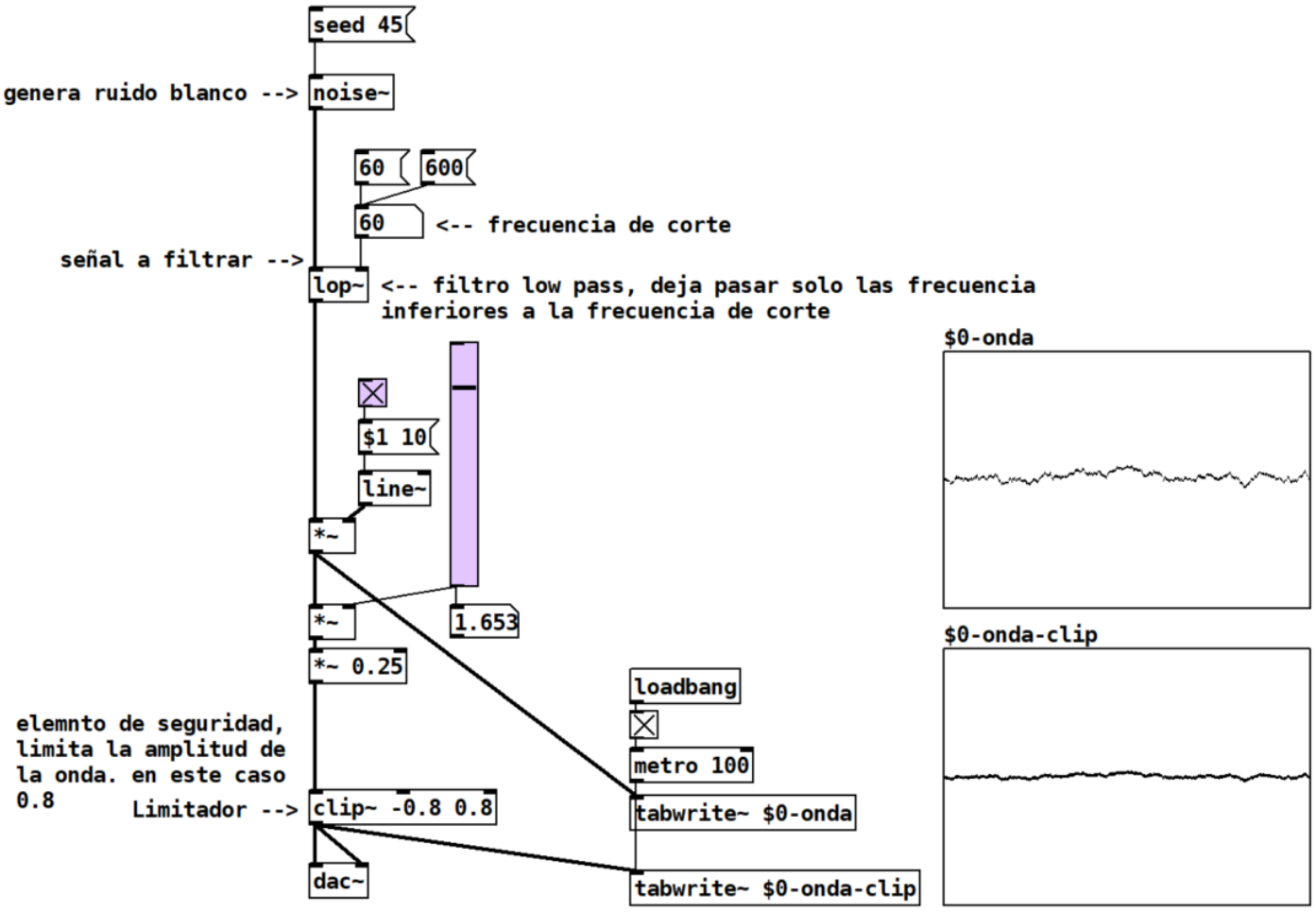


Figura 1. patch *Filtro-Low-pass.pd*

High pass

Este filtro deja **frecuencias altas** a partir de un valor determinado. Al igual que en el low pass, este valor se denomina frecuencia cutoff, o **frecuencia de corte**. Las frecuencias inferiores al valor del cutoff serán reducidas o eliminadas. En pure data es el objeto "**hip~**". Podemos configurar la frecuencia de corte en el argumento o a traves del inlet derecho. En el inlet izquierdo recibiremos la señal que vamos a filtrar.

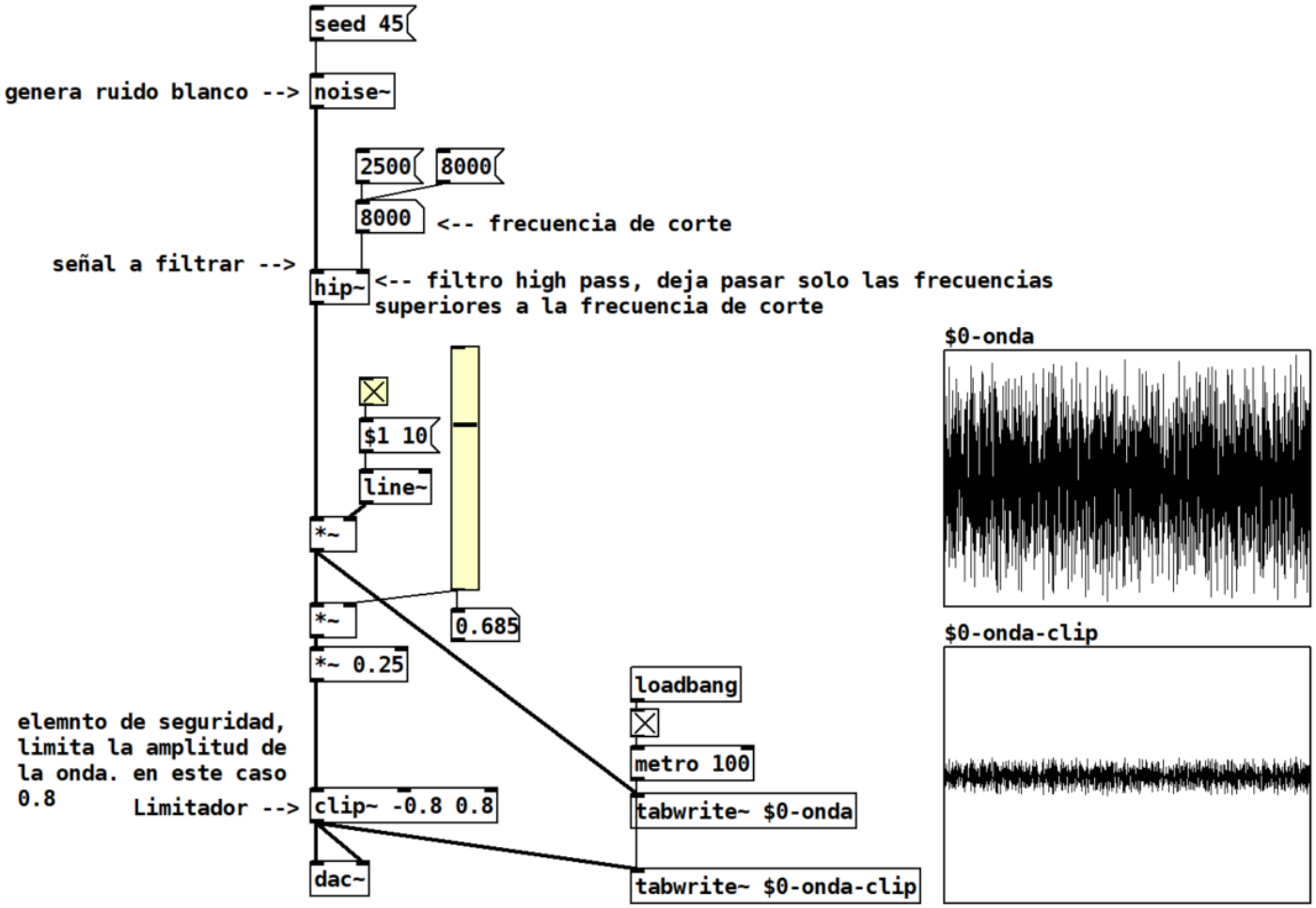


Figura 2. patch *Filtros-High-pass.pd*

Band pass

Podemos describir este filtro como una combinación de un **low pass** y un **high pass**. Dejará pasar las frecuencias comprendidas en un **rango** determinado, reduciendo o eliminando las frecuencias inferiores al valor inferior del rango y las superiores al valor superior del rango. Este rango va a

venir determinado por una **frecuencia central** y la **longitud del rango** de frecuencias que alrededor de esa frecuencia central el filtro dejara pasar, a este rango lo denominaremos **resonancia**. La frecuencia central pasara intacta, las frecuencias alrededor de esta frecuencia central se irán reduciendo progresivamente a medida que nos alejemos de la frecuencia central, hasta ser eliminadas. Este objeto tiene 3 **inlets**, el **primero** recibirá la **señal** de audio a filtrar. El **segundo** la **frecuencia central** y el **tercero** controlara la **resonancia** (en este objeto de pure data se recomienda utilizar unos valores para la de resonancia de entre 0 y 10, pero probar todos los valores que queráis)

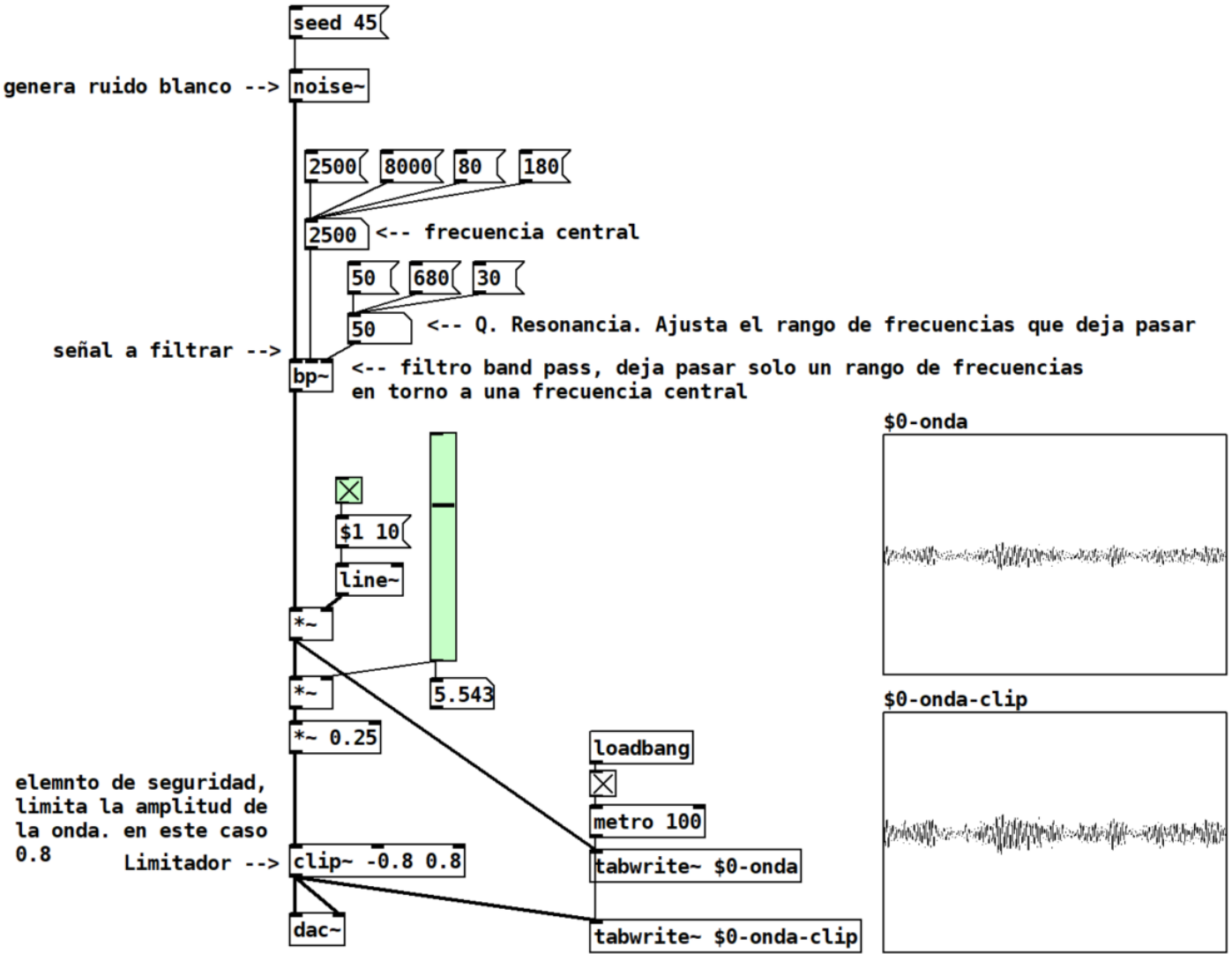


Figura 3. patch *Filtros-Band-pass.pd*

Combinar Filtros

En la carpeta de material, os dejo un patch con todos los filtros para que los probéis, comparéis y mezcléis:

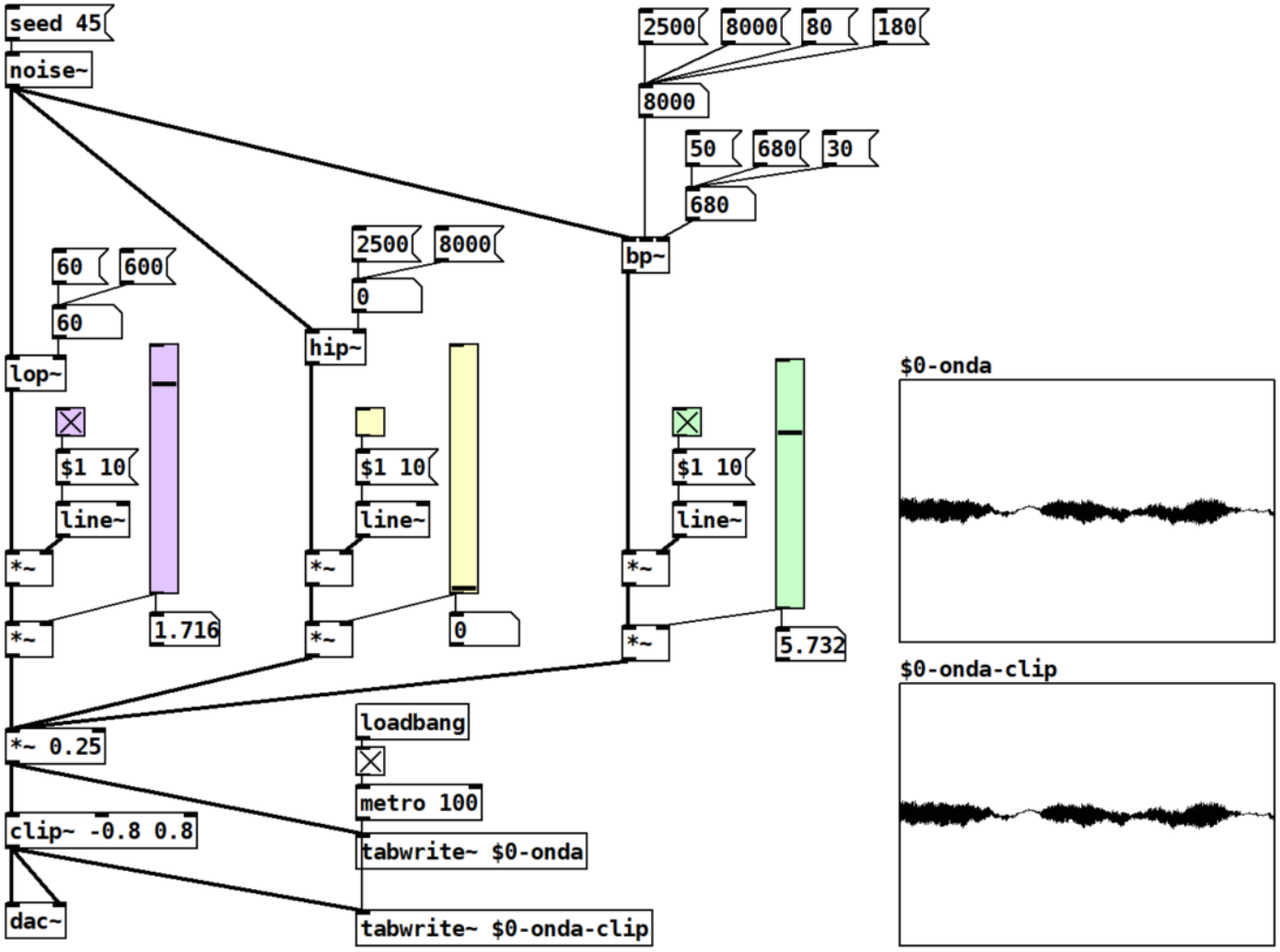


Figura 4. patch *Filtros.pd*

Figuras:

Figura 1. patch *Filtro-Low-pass.pd*

Figura 2. patch *Filtros-High-pass.pd*

Figura 3. patch *Filtros-Band-pass.pd*

Figura 4. patch *Filtros.pd*

Snare drum y Hi hat

Batería

Es un conjunto de instrumentos musicales de percusión. Vamos a ver tres de sus elementos más característicos y recrear su sonido en Pure Data.

https://www.youtube.com/embed/RyXYFtsqym4?start=10

Los elementos que vamos a ver son el kick drum, que vimos anteriormente en la Práctica 4: Kick drum ... | Librería CATEDU, el hi hat y el snare drum:

- **Kick drum:** Es lo que conocemos como **bombo** y produce el sonido más grave y potente de la batería, se percute con una maza que se activa con el pie utilizando un pedal.
- **Hi-hats**: Son los platillos metálicos. Frecuentemente un sistema que consta de **2 platillos** instalados en un soporte con pedal que permite que uno caiga sobre el otro haciéndolos sonar.
- **Snare drum:** En español se conoce como **caja** y es un tambor.

Como vais a ver a continuación lo que vamos a hacer el pure data es emular las ondas generadas por estos instrumentos acústicos y cómo se comportan en el espacio y el tiempo.

Kick drum o bombo

https://www.youtube.com/embed/nhjmB85I3a8

Ahora que conocemos los filtros vamos a añadir un **filtro low pass "lop~"** al Kick drum que va a cortar todas las frecuencias por encima de 1000, esto mejorar su sonido. Y vamos a hacer en el envelope que controla el **volumen**, que el **decay** sea **regulable** utilizando "$1":

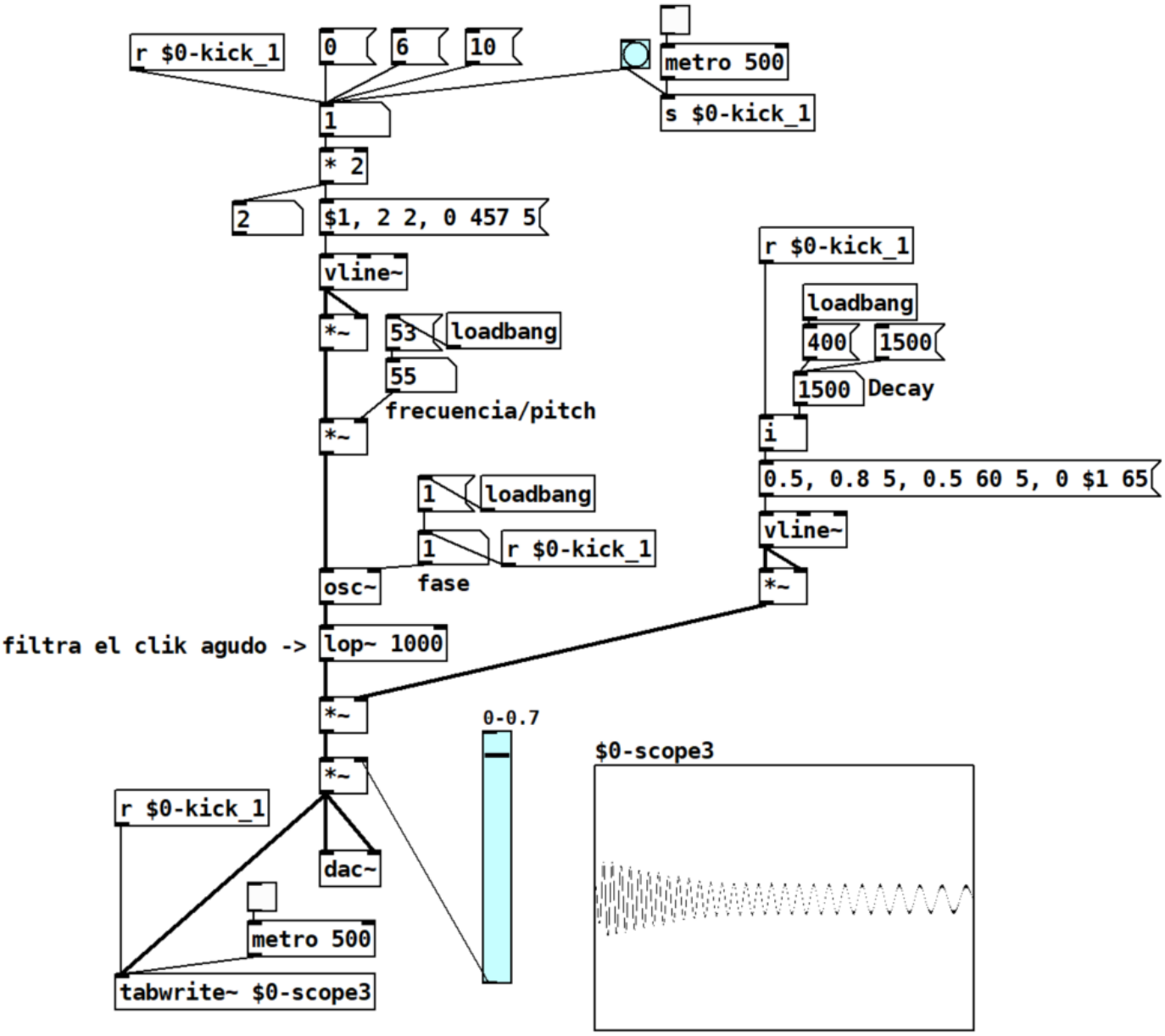


Figura 1. patch *Kick-drum.pd*

Hi hat

https://www.youtube.com/embed/MsrwwmJ6Mts?start=8

De forma similar a como hemos creado el kick drum, vamos a crea un Hi hat. Para ello utilizaremos las **frecuencias más agudas** del **ruido blanco** que crearemos con el objeto **"noise"** y les

daremos forma con un envelope muy sencillo con una progresión exponencial que controle el volumen. Vamos a modificar la rampa que controla el volumen con el objeto "**pow~**" que hemos visto en Rampas potenciales: po... | Librería CATEDU. Tambien haremos que el **Decay** del **volumen** sea **regulable** utilizando **$1**.

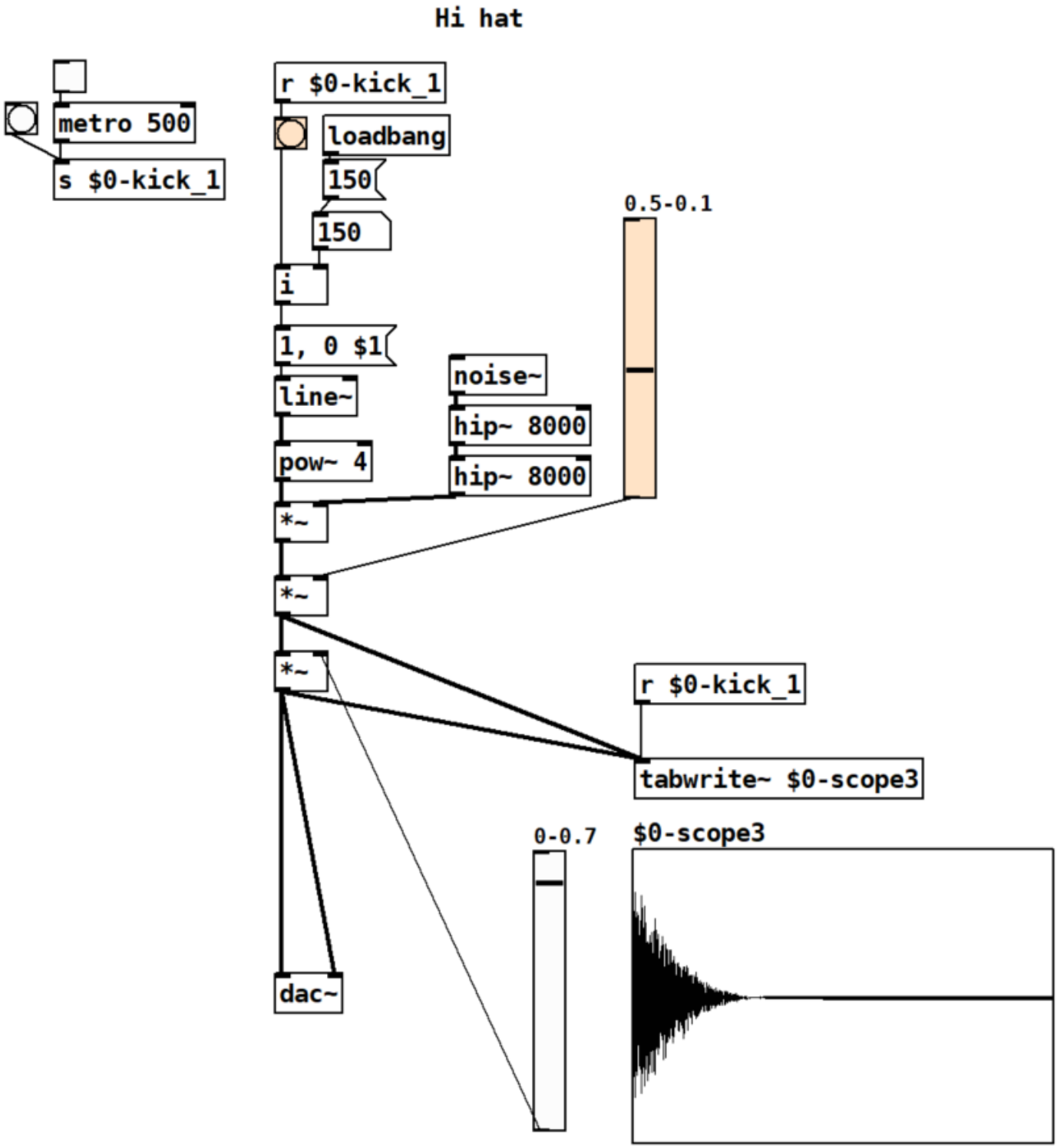


Figura 2. patch *Hi-hat.pd*

> Ejercicio 1: Porque utilizamos una variable integer para las variaciones de la rampa de volumen en los dos ejemplos anteriores? ¿Qué pasaría si omitiéramos el objeto "**i**" y variáramos el valor que llega a la rampa del line?

Snare drum

https://www.youtube.com/embed/KkqsXWg9Gao

Vamos a crea un snare drum muy sencillo en Pure data. Para ello utilizaremos un oscilador cuyo volumen controlaremos con un envelope; las **frecuencias más agudas** del **ruido blanco** que volveremos a filtrar con un **bandpass** y cuyo volumen regularemos con una rampa que dure más que el sonido del oscilador. Probar a cambiar los valores de los envelopes para ver que efecto tiene en el sonido. Probar a cambiar todos los números de los filtros. Experimentar para conocer cómo funciona el patch *Snare-Drum.pd*

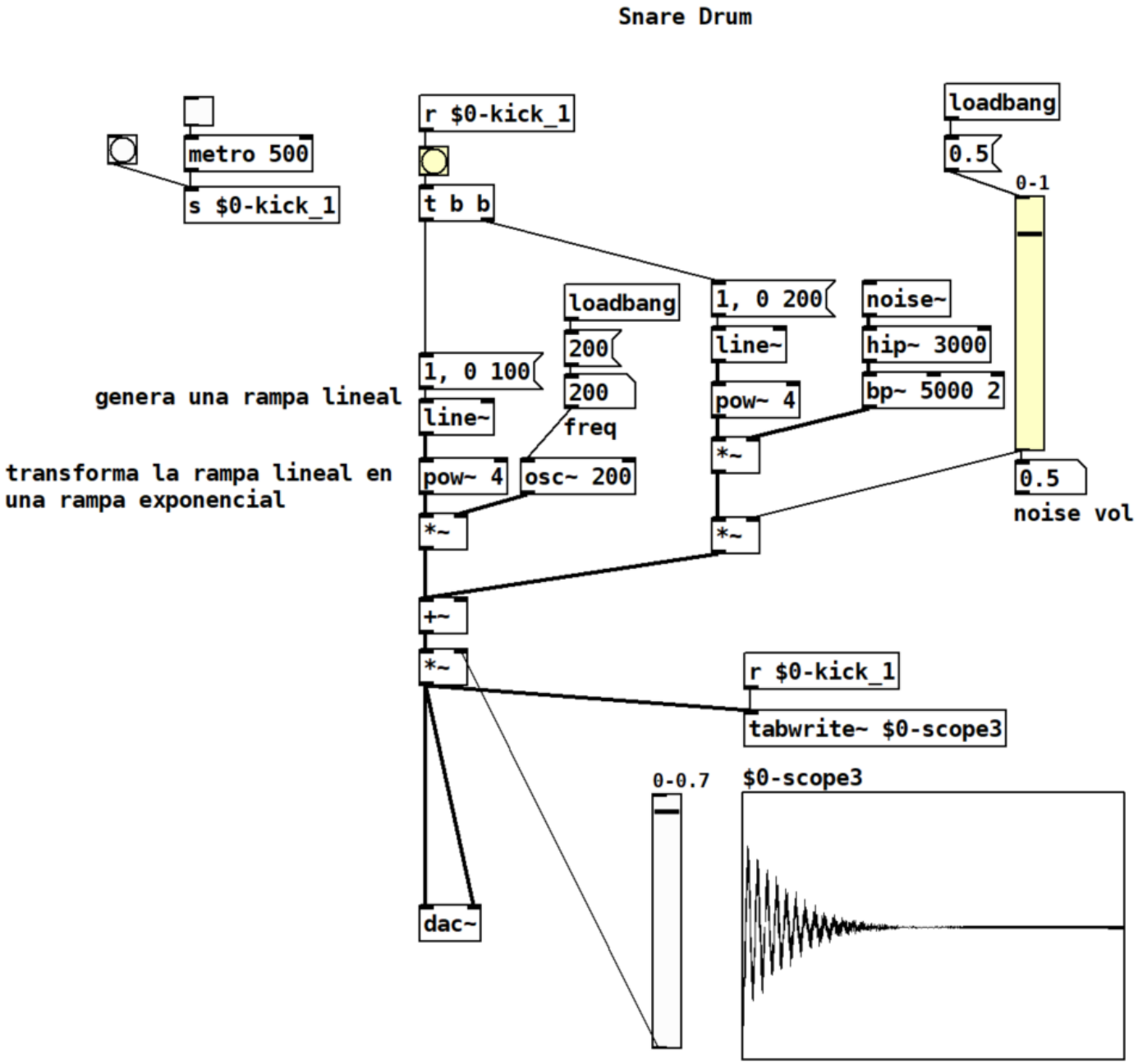


Figura 3. patch *Snare-Drum.pd*

> Ejercicio 2: Crear un **subpatch** para el kick drum, otro para el hi hat y otro para el snare drum. Coloca esos tres subpatches en el mismo patch de pure data. El kickdrum tendrá 3 inlets y 1 outlet. El hi hat 3 inlets y 1 outlet y el snare drum 4 inlets y 1 outlet.

Figuras:

Figura 1. patch *Kick-drum.pd*

Figura 2. patch *Hi-hat.pd*

Figura 3. patch *Snare-Drum.pd*

Práctica 12: Caja de ritmos

Ahora que tenemos los instrumentos de una batería y sabemos hacer un secuenciador vamos a juntarlo todos para crear una caja de ritmos de 8 pasos/tiempos. Os dejo un video de una caja de ritmos de 16 tiempos para que os hagáis una idea de lo que permite:

https://www.youtube.com/embed/gFgITfT2Ihw

¿Qué necesitamos y que estructura va a tener este programa?

Este será nuestro algoritmo general:

1- Necesitamos un **contador** de **8 pasos**.

2- Necesitaremos **enviar** esos **pasos** a cada uno de nuestros instrumentos.

3- Necesitaremos indicar en **cada instrumento** en cual o cuales de los 8 **pasos** queremos que se activen.

4- Necesitaremos que los **instrumentos** generen una señal.

5- Necesitaremos poder **regular el volumen** de **cada instrumento** para ajustar la mezcla.

6- Necesitaremos **mezclar la señal** de los tres instrumentos y **regular el volumen** de ma mezcla (master).

Vamos a ver que tenemos que hacer en cada paso de nuestro algoritmo para conseguir nuestro objetivo:

1- Contador de 8 pasos.

Esto ya lo hemo [illegible] Práctica 3: Contador y... | Librería CATEDU. Y sería algo asi:

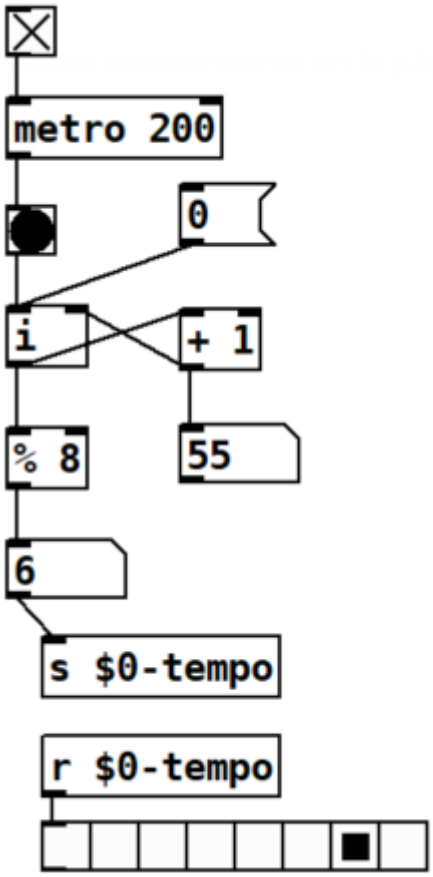


Figura 1. Contador de 8 tiempos.

2- Enviar esos pasos a cada uno de nuestros instrumentos.

Para facilitar el orden y la legibilidad de nuestro patch utilizaremos los objetos "**send**" and "**receive**" para enviar el resultado de nuestro contador (que serán valores del 0 al 7) a cada uno de nuestros instrumentos. Utilizaremos un "**receive**" para cada instrumento

3- Indicar en cada instrumento en cual o cuales de los 8 pasos queremos que se activen.

Este paso va a ser el más novedoso para nostros ya que aún no hemos construido un patch así. ¿Como podemos hacer esto y que necesitamos? escribamos un algoritmo para plantear la solución a nuestro problema.

3.1- Necesitamos saber en qué **paso** está el **contador**.

3.2- Necesitaremos una **configuración** que indique en que **pasos** queremos que se active el **instrumento**.

3.3 - Necesitamos saber **si** el **paso** en que se encuentra el **contador** está **activado** en un **instrumento**.

3.4 - Si el **paso** en el que se encuentra el contador esta **activado** enviaremos un **bang** para activar el instrumento

3.5 - Si el **paso** en el que se encuentra el contador **no** está **activado no** enviaremos un **bang** para activar el instrumento

Vamos a incluir el sub-algoritmo 3 en un subpatch que tendrá 8 inlets y 1 outlets.

3.1- Necesitamos saber en qué paso está el contador.

Habíamos dicho en el punto dos que enviaríamos el resultado del contador con un objeto "send", asique crearemos un objeto "receive" para que a cada uno de nuestros instrumentos les llegue la información de en qué paso está el contador.

3.2- Necesitaremos una configuración que indique en que pasos queremos que se active el instrumento.

En nuestro caso vamos a utilizar 8 objetos **toggle** para cada instrumento, habíamos visto su funcionamiento en la página tal. los colocaremos en fila y cada uno de ellos controlara el estado de un paso del contador. Cuando el toggle este abierto enviara un 1 por su salida y cuando esté cerrado un 0.

Figura 2. Ocho toggles en fila.

3.3 - Necesitamos saber si el paso en que se encuentra el contador está activado en un instrumento.

Para ello vamos a **comparar** la posición en la que se encuentra el contador con el estado de un instrumento para esa posición. Los valores que genera el contador van de 0 a 7 y los valores que genera el toggle que nos indica el estado de cada posición son **0** o **1**. Vamos a buscar una manera

de comparar estos dos valores.

Primero vamos a utilizar el objeto "**select**" o "**sel**" para **separar** cada uno de los **pasos del contador**. ¿Cómo funciona este objeto? En el objeto "sel" podremos introducir tantos argumentos como elementos queramos evaluar, cuando un **elemento recibido** por el sel **coincida** con alguno de su **argumento** enviara un **bang** a traves del outlet correspondiente con la posición del argumento.

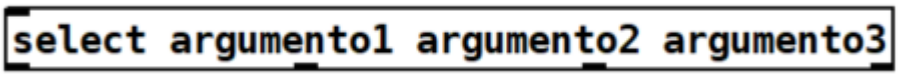


Figura 3. Objeto select.

Esto quiere decir si coincide con el **segundo argumento**, enviara un bang por el **outlet segundo** . "select" se puede utilizar también con símbolos, pero no se pueden mezclar símbolos y numero. En nuestro caso vamos a utilizar números. Pero vemos que para tres argumentos tenemos 4 outlets. Esto es porque por el **ultimo outlet** se enviarán aquellos valores que **no coincidan** con ninguno de los argumentos. Abre el patch *select.pd* para probar el objeto.

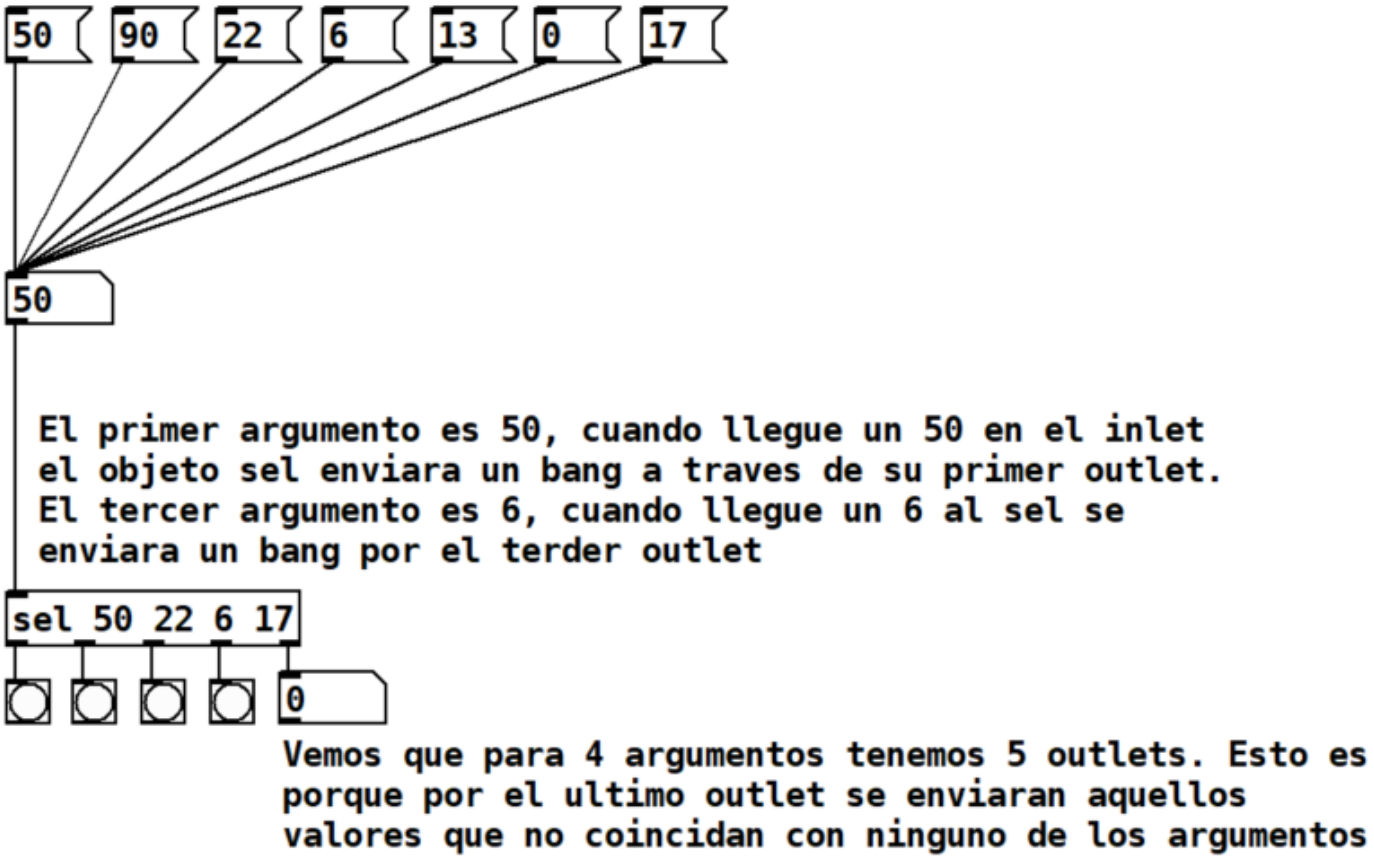


Figura 4. Patch *select.pd.*

Ahora que ya conocemos el objeto select vamos a utilizarlo para separar cada uno de los pasos del contador. Como el contador genera números de 0 a 7 vamos a crear un "select" con 8

argumentos, que **coinciden** con los **valores** que genera nuestro **contador**:

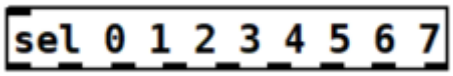


Figura 5. Objeto "select" que vamos a utilizar en esta práctica.

Cada vez que llegue al inlet del "select" de la figura 5, un 6, se enviara un bang por la séptima ; valores que genera el contador.

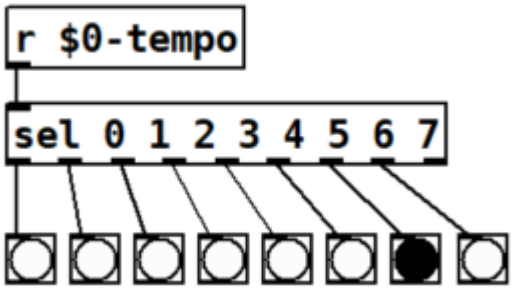


Figura 6. Objeto "sel" que acaba de recibir un 6 en su inlet y envía un bang por el séptimo outlet. (el 6 es el séptimo argumento).

Ahora vamos a crear una **estructura que compare** cada paso del contador con el estado del instrumento en ese paso. Empezamos por el **primer paso**, que en los valores que nos proporciona el **contador** corresponde con el **numero 0**, y que es el primer argumento de nuestro "select". Cada vez que el contador se encuentre en la posición 0 nuestro select enviara un bang por su primer outlet. Habíamos dicho que utilizamos 8 objetos toggle para marcar los pasos en los que queremos que este activado el instrumento. Como el toggle activado envía un **1** vamos a utilizar ese valor como **parámetro de comparación**. Vamos a utilizar el objeto **"=="** para comparar el estado de un instrumento en un paso determinado. Cada vez que el contador este en el paso que estamos evaluando queremos que se realice una comparación con el estado del instrumento en ese paso. Es algo así como responder a la pregunta. Estamos en el paso 1, en qué estado está el instrumento en el paso 1? Si esta activado activa el instrumento.

> Recordar que el los operadores matemáticos el inlet izquierdo introduce valor y da la orden de realizar la operación, por el contrario, el inlet derecho solo introduce valor.

Como queremos que la comparación se realice cuando el **contador** este en el paso que estamos evaluando vamos a enviar un **1** a traves del **inlet izquierdo de "=="** cada vez que el contador se encuentre en el paso que estamos evaluando.

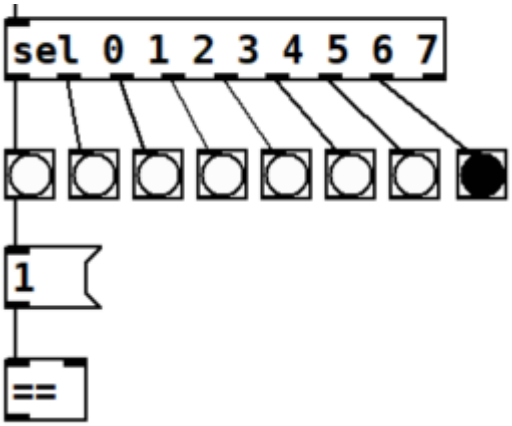


Figura 7. Cuando el objeto "select" reciba un 0 por su inlet,

un bang saldrá por su primer outlet y enviara el mensaje que contiene "1" al inlet izquierdo del objeto "==".

Por el **inlet derecho de "==" vamos a recibir el estado del instrumento** en ese paso. Como es el p ggle, que cuando este **activado** sera un **1** y cua

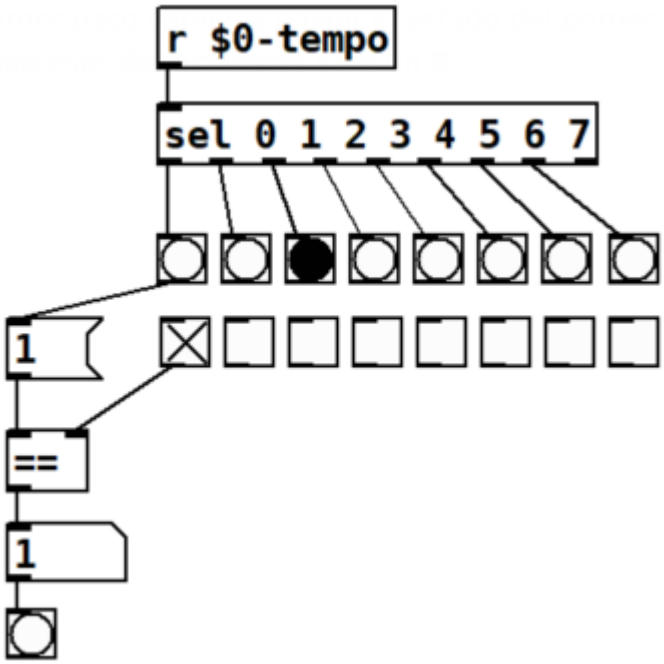


Figura 8. El primer paso del control de este

instrumento esta activado. Cuando el primer toggle este activado se enviará un "1" al inlet derecho del objeto "==", cuando el toggle este desactivado enviará un "0".

Cuando el primer toggle este activado se enviará un "1" al inlet derecho del objeto "==", cuando el primer toggle este desactivado enviará un "0". Cuando el objeto "select" reciba un 0 por su inlet,

un bang saldrá por su primer outlet y se enviara un "1" al inlet izquierdo del objeto "==". Cuando los valores que recibe el objeto "==" sean iguales, enviara un 1 por su outlet, si los valores procedentes del contador y del estado del instrumento no coinciden, enviara un 0.

La operación de comparación en el objeto "==" se realizará cada vez que llegue algo al inlet izquierdo. En nuestro caso al inlet izquierdo llega siempre un 1 y sera el **estado del instrumento** recibido en el inlet derecho el valor que **alterne** entre **0 y 1**, dependiendo de si este activado ese paso o no.

3.4 - Si el paso en el que se encuentra el contador esta activado enviaremos un bang para activar el instrumento + 3.5 - Si el paso en el que se encuentra el contador no está activado no enviaremos un bang para activar el instrumento

Cuando el toggle este activado y el objeto "==" se realize la comparación **"1 ==1"** el objeto "==" emitirá por su **outlet** un **1**, ya que sus dos inlets cumple la condición de igualad que establece el operador matemático. Cuando el toggle este desactivado y el objeto "==" se realize la comparación **"1 ==0"** el objeto "==" emitirá por su **outlet** un **0**, ya que la condición de igualdad no se cumple. Para activar el instrumento nos interesa tener un bang cada vez que esta condición de cumpla por lo que volveremos a utilizar el objeto **"select"** para emitir un **bang** cada vez que el objeto "==" envíe un 1 por su outlet y este **bang** sera el que **active** el **instrumento**.

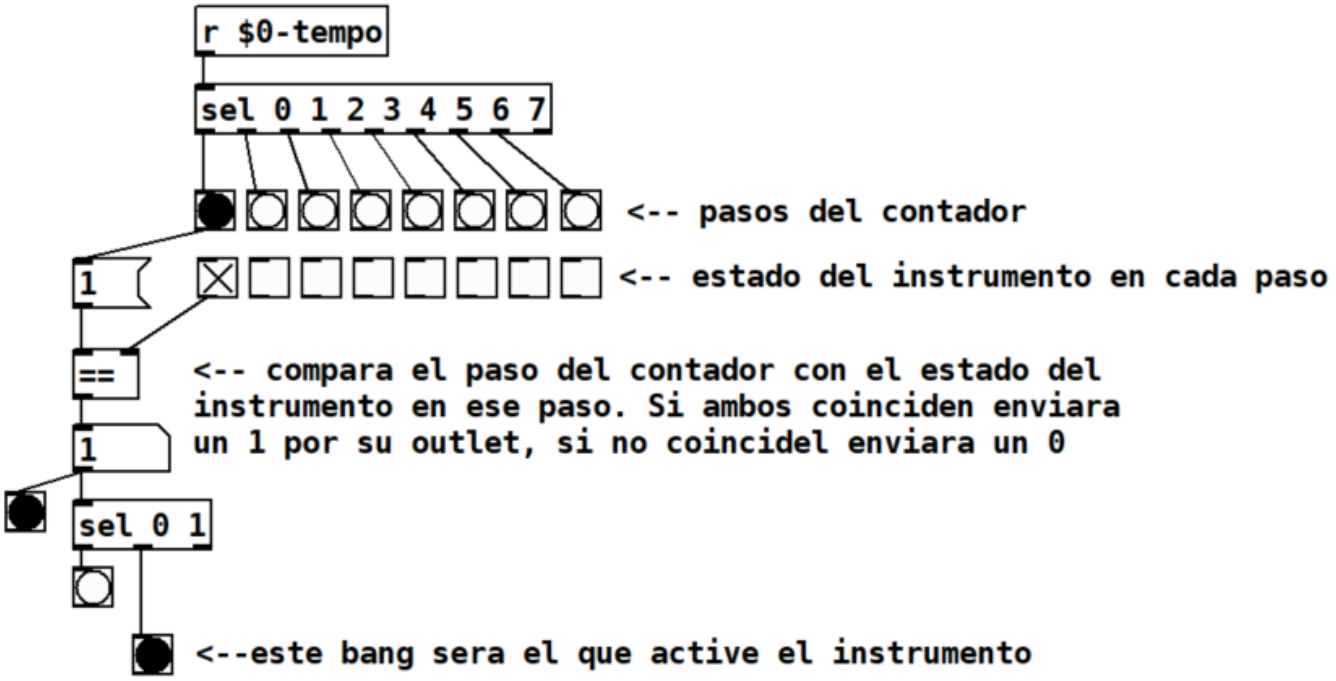


Figura 9. Comprobación del estado del instrumento en el paso en el que se encuentra el contador (primer paso). En ese paso el instrumento esta activado por lo que tras comparar con el "==" y clasificar con el "sel 0 1" obtenemos un bang por el segundo outlet de objeto "sel 0 1".

Lo que hemos hecho hasta el momento evalúa solo el estado del instrumento en el primer paso, tendremos que **repetir** lo mismo **para cada paso**:

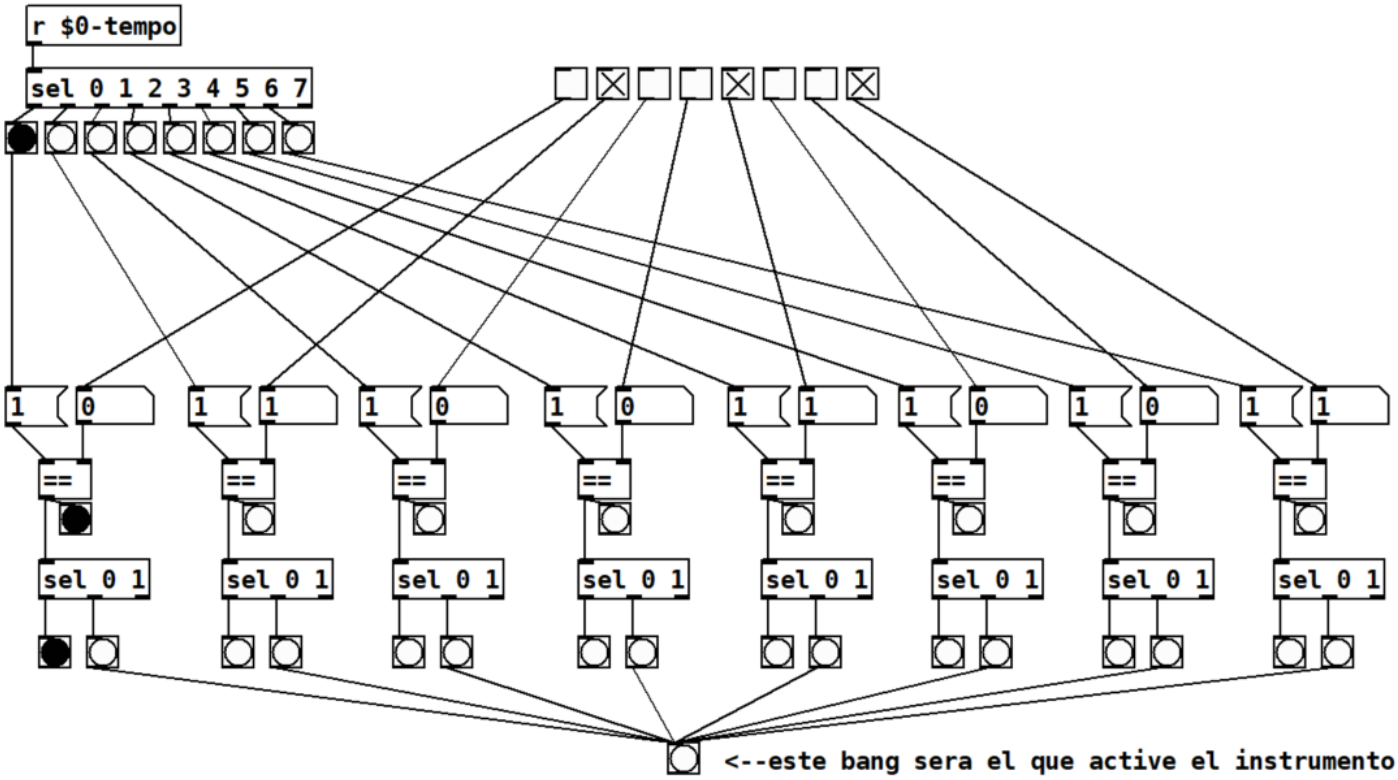


Figura 10. Patch que compara el estado del instrumento en cada uno de los 8 pasos que genera el contador. Cada vez que un paso del instrumento este activado se enviara un bang para activar el sonido.

Una vez lo tengamos hecho para cada paso tendremos completado el control de 1 instrumento, para controlar el resto de los instrumentos, tendremos que **repetir la misma estructura** y lo haremos creando un **subpatch** donde meteremos toda esta estructura de control.

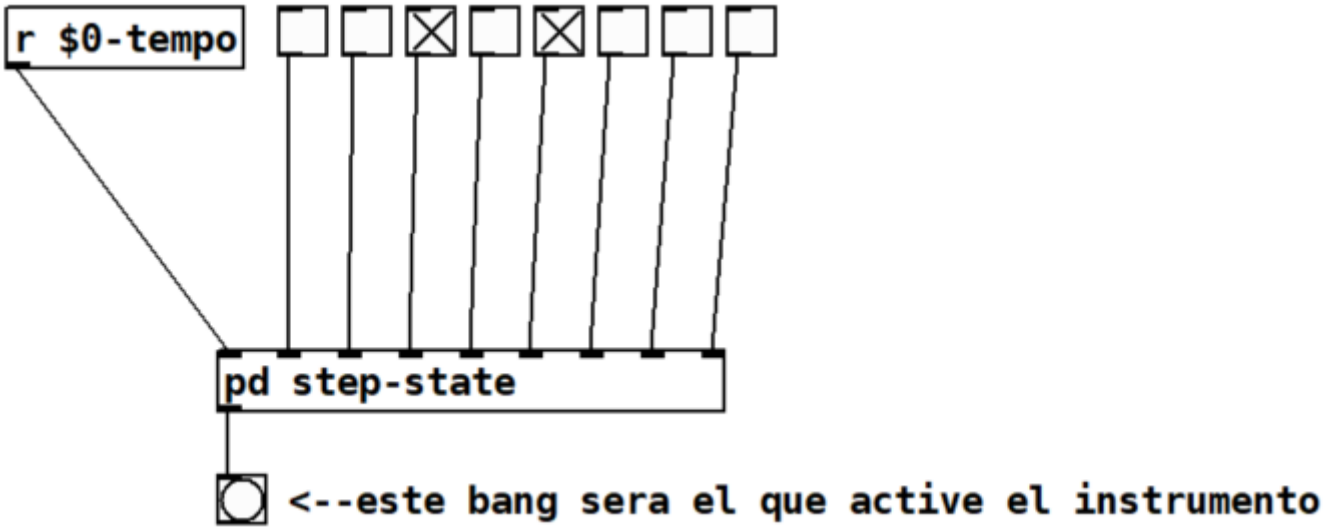


Figura 11. Subpatch que contine gran parte de la estructura de la figura 10.

Este **subpatch** tendra **9 inlets** y **1 outlet**. 8 de los inlets enviaran el estado del instrumento en cada paso que controlan los toggles y 1 de los inlets enviara el valor del contador. En el outlet tendremos el bang que controla el instrumento.

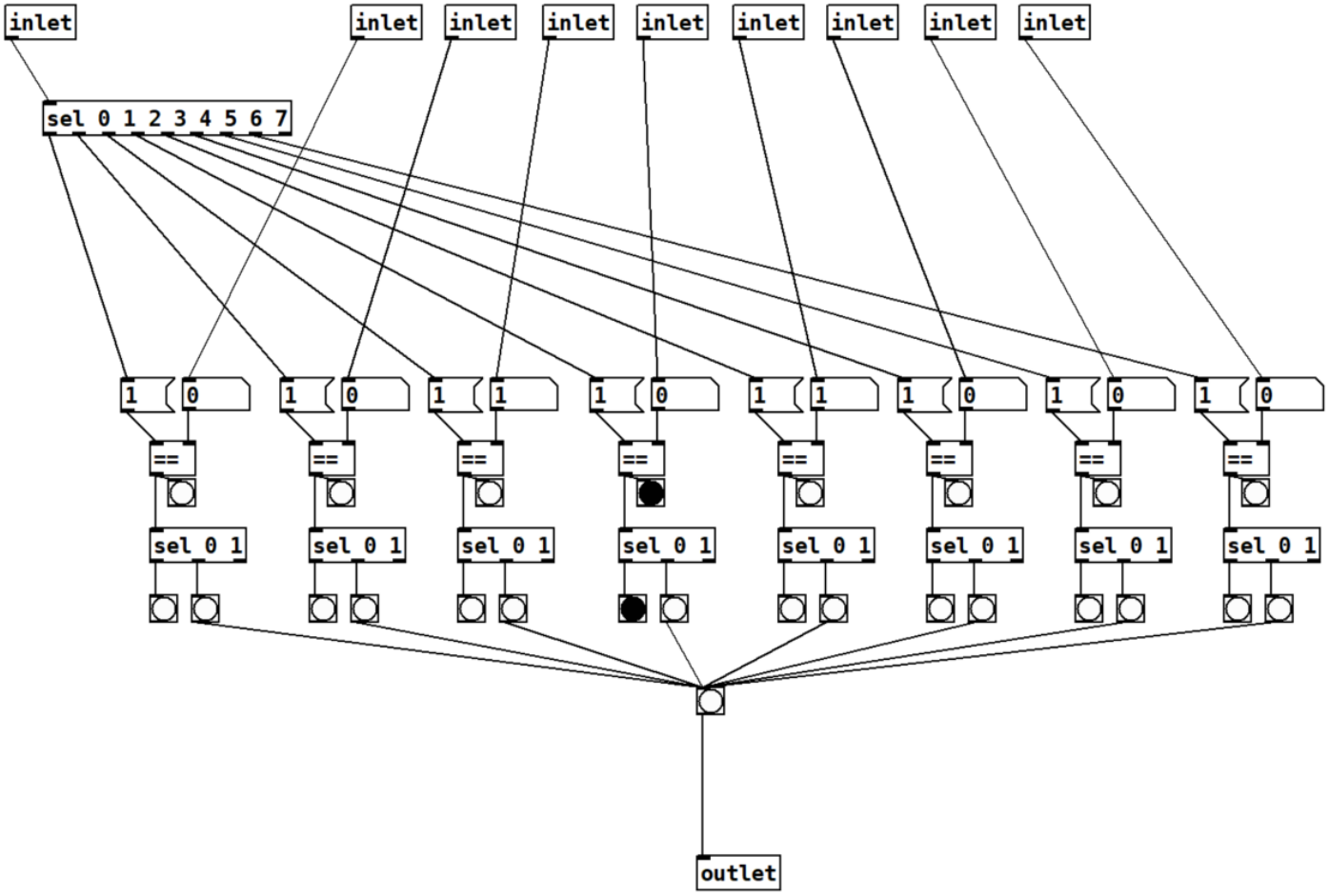


Figura 12. Estructura de control de la figura 10 metida en un el subpatch "pd step-state" de la figura 11.

Ya tenemos nuestro sistema de control para los instrumentos. ¡Traigamos ahora los instrumentos!

https://giphy.com/embed/11MS2pksWxB8SA

Figura 13. La orquesta de camino a vuestro patch de pd.

4- Instrumentos generen una señal, o que la reproduzcan.

Los hemos hecho en lecciones anteriores: Snare drum y Hi hat | Librería CATEDU y Micrófono: Grabar y re... | Librería CATEDU

Podemos utilizar los instrumentos que hemos hecho en la lección anterior o también reproducir sonidos grabados, vamos a combinar estos. Vamos a **meter los instrumentos en subpatches** y dejar fuera del subpatch solo los parámetros que queramos o necesitemos. En previos capítulos hemos creado el **kick drum**, **snare drum** y **hi hats.** Una vez hayamos creado los subpatches en el patch de cada instrumento, vamos a coger esos subpatches y traerlos al patch en el que estamos creando nuestra caja de ritmos.

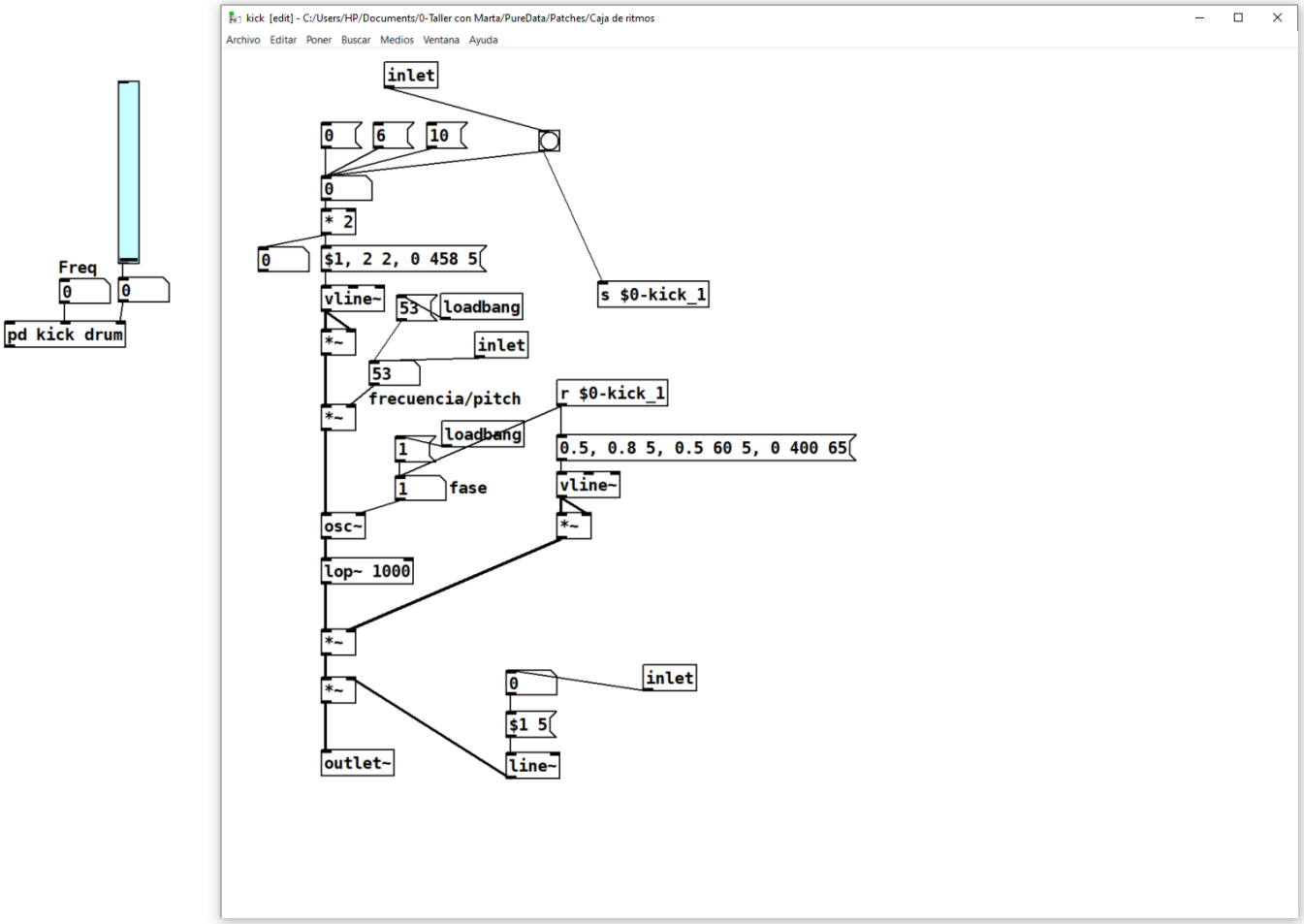


Figura 14. Subpatch para el kick drum o bombo.

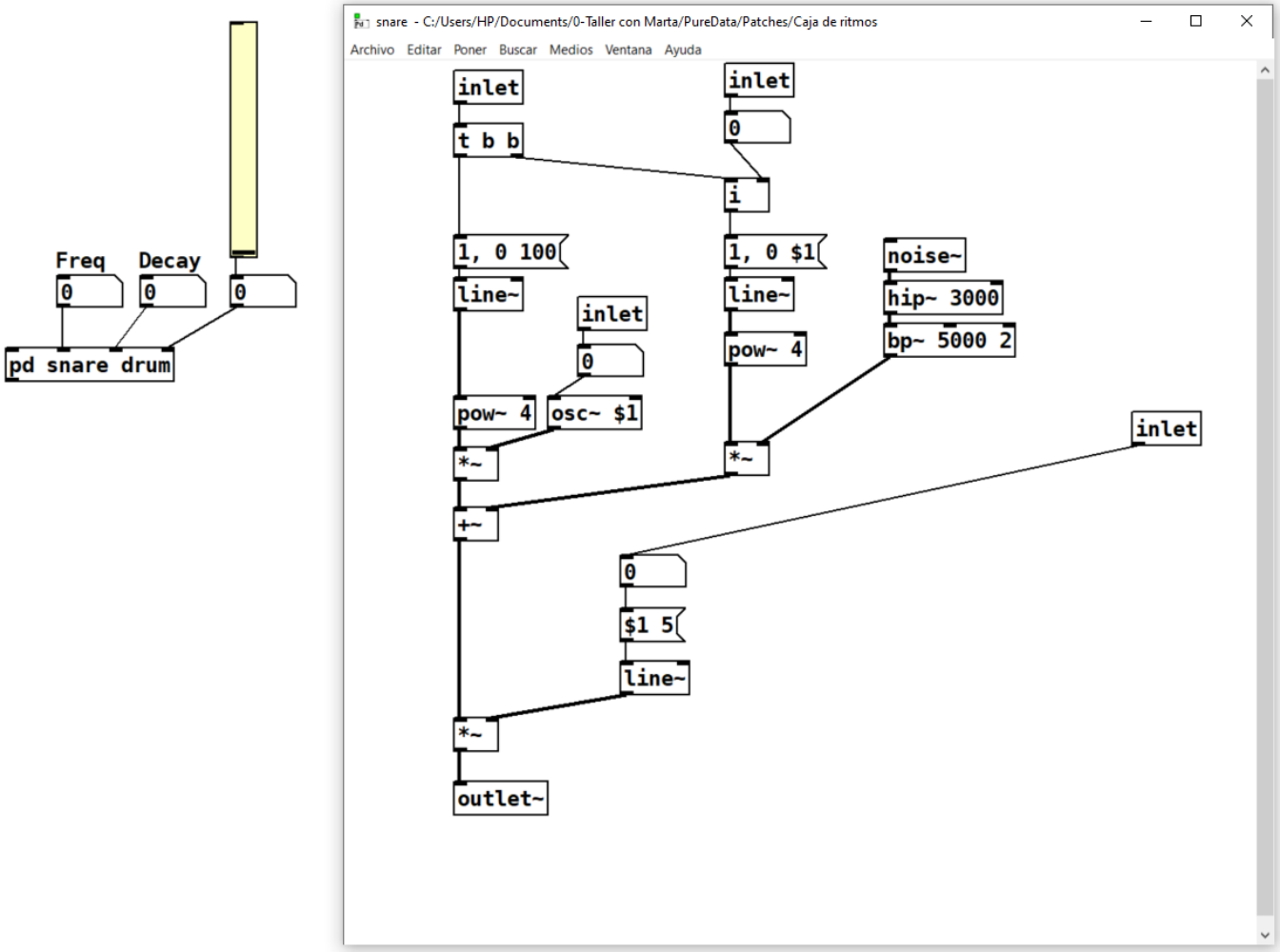


Figura 15. Subpatch para snare drum o caja.

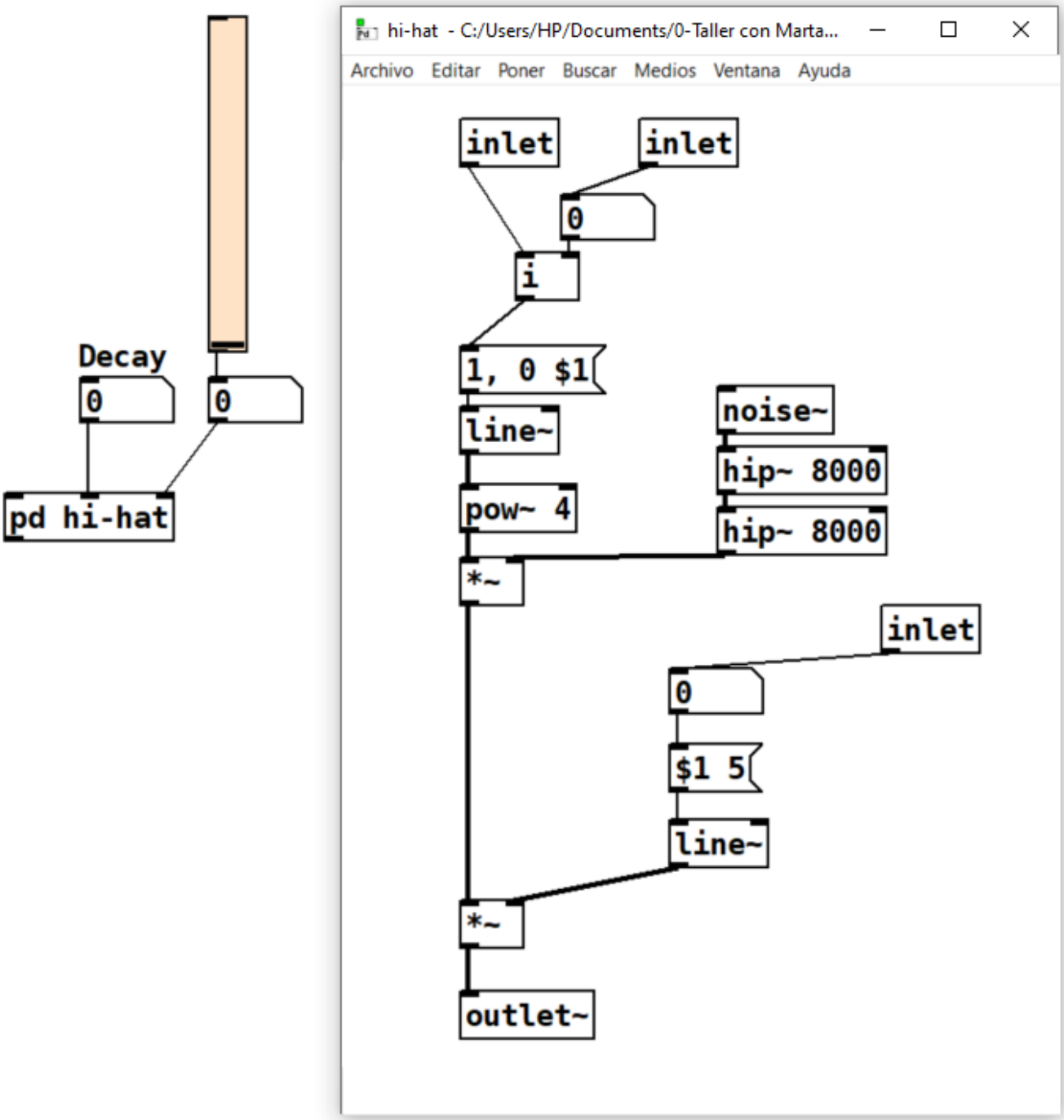


Figura 16. Subpatch para hi hats o platillos.

Copiamos los subpatches de nuestros instrumentos, seleccionándolos en el modo edición, clicamos y sin soltar movemos el rato cubriendo el area donde se encuentren los objetos que queremos seleccionar, una vez seleccionados se pondrán en azul). Los copiámos (Ctrl + C) y los pegamos (Ctrl + V) en el patch que queramos:

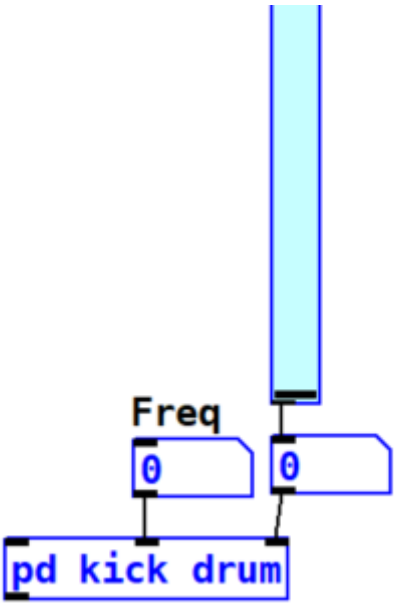


Figura 17. El subpatch del kick drum seleccionado.

Una vez tengamos los instrumentos, conectamos cada uno con un subpatch de control.

Vamos a añadir también un instrumento utilizando un sample, hemos visto cómo hacerlo en Micrófono: Grabar y re... | Librería CATEDU:

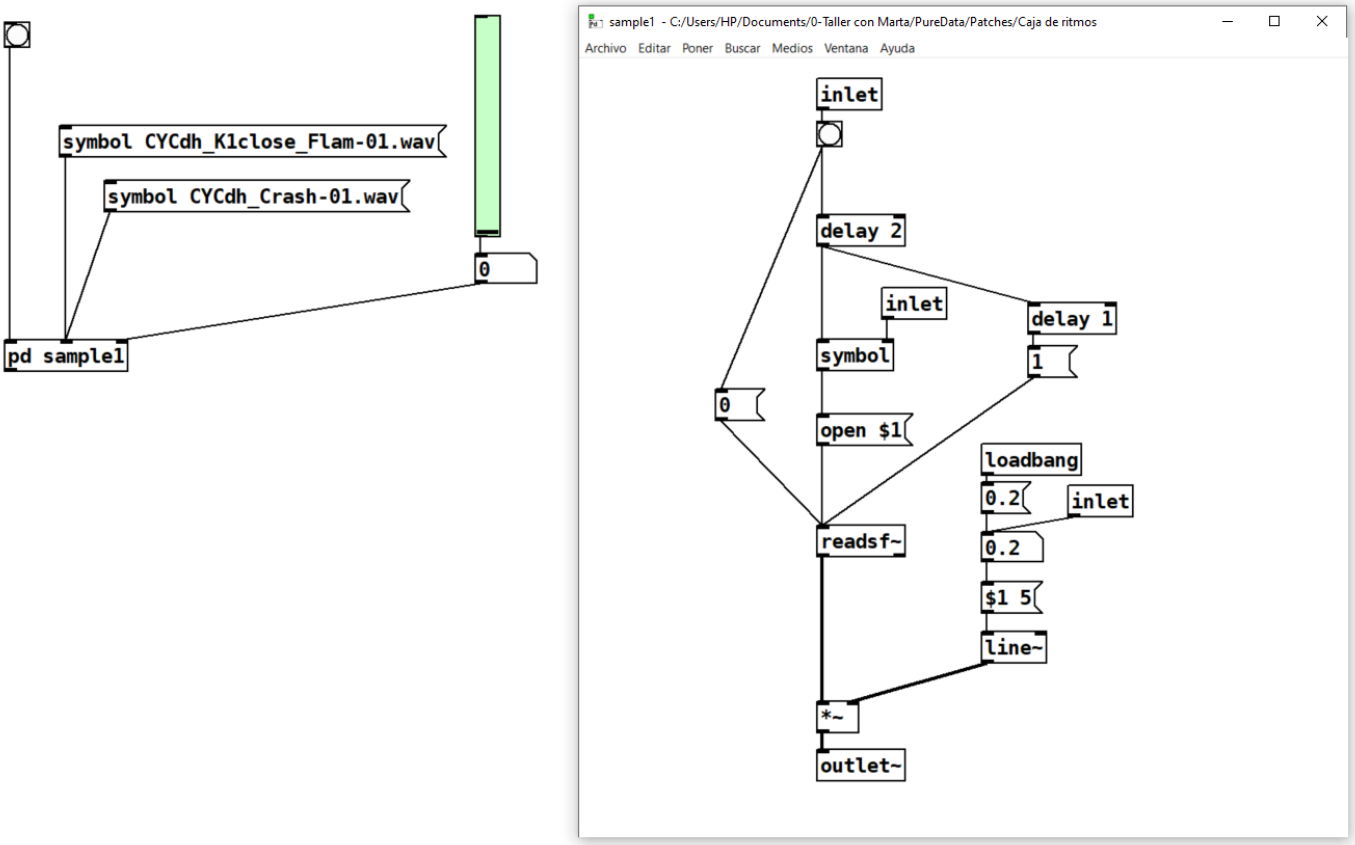


Figura 18. patch *Sample-subpatch.pd*. Subpatch para el sonido procedente de un archivo.

En el mensaje de open hemos sustituido el nombre del archivo por **$1** para poder cargar diferentes archivos enviado un mensaje con el nombre del archivo que queremos reproducir. Hemos añadido al mensaje la palabra symbol para que se identifique como tal. Recordar que los archivos tienen que estar en la misma carpeta que el patch.

> Recordar que el orden en el que conectáis los objetos importa. El orden en que el bang activa en 0 y el delay importan. Para controlar el orden también podéis utilizar un delay muy rápido para aseguraros de que el orden en que se ejecutan los comandos es el que queréis.

5- Regular el volumen de cada instrumento para ajustar la mezcla.

Como podréis observar en las figuras 14, 15, 16, 17 y 18, he añadido un control de volumen que se puede modificar desde fuera del subpatch a todos los instrumentos. Estos nos va a permitir ajustar la mezcla.

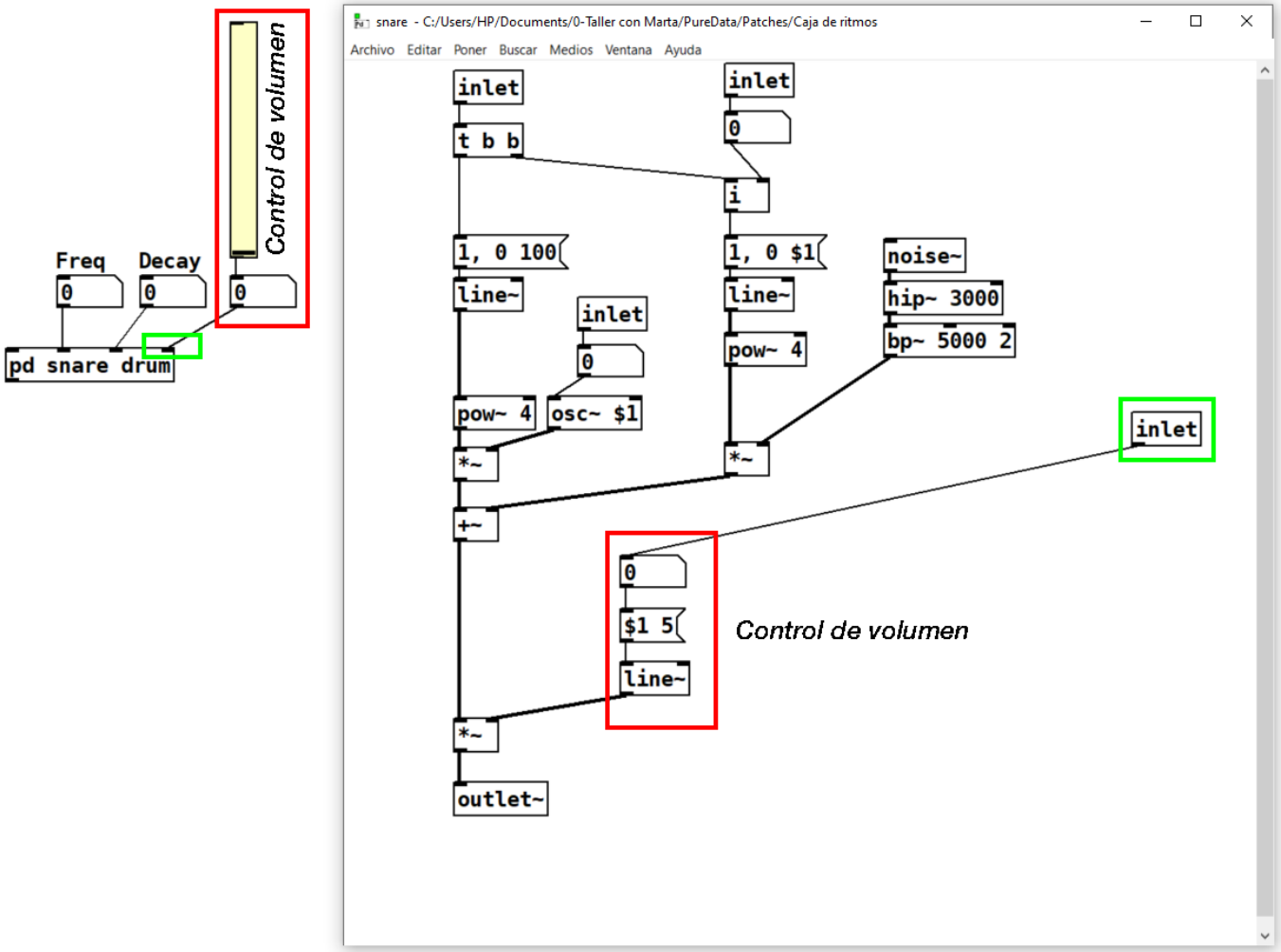


Figura 19. Subpatch para snare drum o caja con el control de volumen general del instrumento marcado en rojo.

6- Mezclar la señal de los tres instrumentos y regular el volumen de la mezcla (master).

Una vez tengamos todos los instrumentos que queramos vamos a **mezclarlos** utilizando el **objeto "*~"** . Enviaremos las señales al inlet izquierdo y a traves del inlet derecho controlaremos el volumen. El **master** es lo que enviamos a nuestros altavoces "**dac~**".

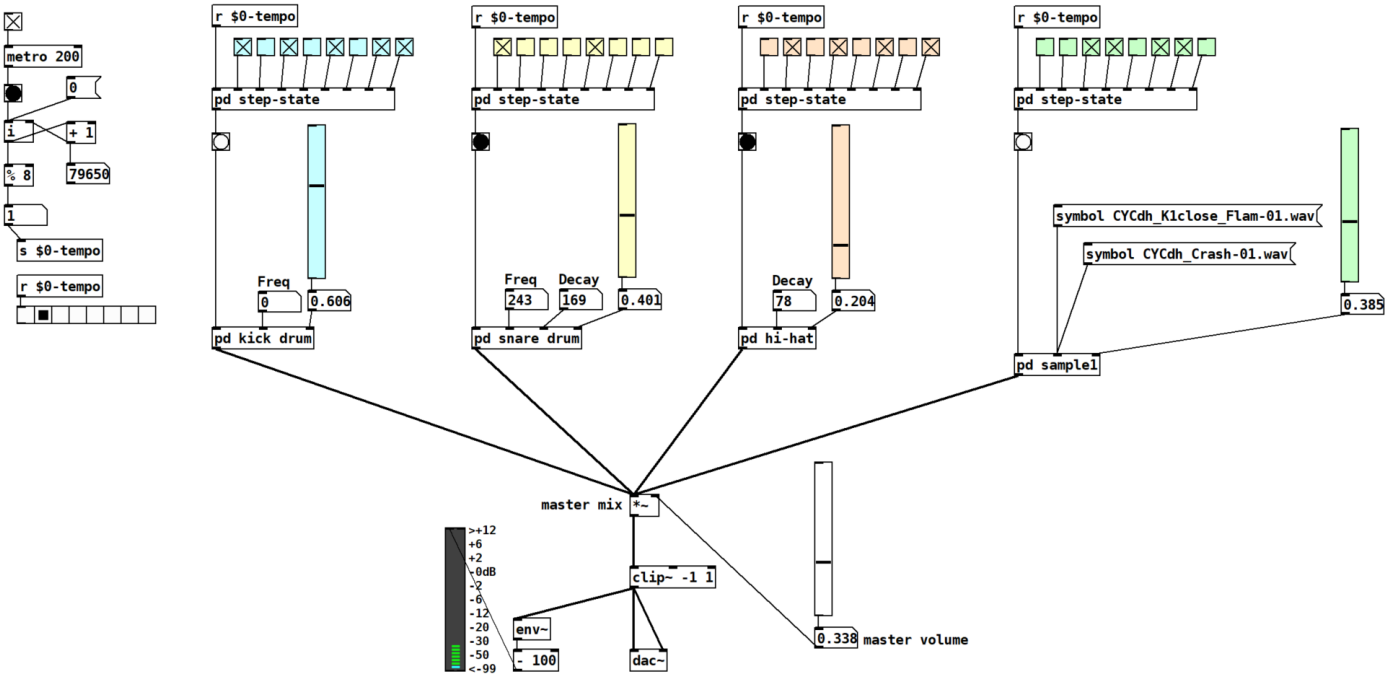


Figura 20. El patch que habéis construido para vuestra caja de ritmos.

> Ejercicio 1: Construye la caja de ritmos como se acabamos de ver. Ha de tener un mínimo de cuatro instrumentos. Puedes incluir más instrumentos o sonidos grabados, incluso una melodía. Explora, **experimenta** y **diviértete**. Una vez tengas tu caja de ritmos graba tres audios utilizando el objeto "**writesf~**" con diferentes patrones de ritmo. Recuerda vimos el objeto "writesf~" en Micrófono: Grabar y re... | Librería CATEDU. Sube tu patch y esas tres grabaciones.

Ejemplos: caja de ritmos de grabaciones de estornudos, de pedos, de palabras grabadas... Explora, **experimenta** y **diviértete**

¿Qué tenemos que entregar?

El ejercicio 1 de la lección. Sube a la carpeta del Moodle de la Practica 12 los siguientes archivos:

- patch ***caja_de_ritmos-vuestronombre_vuestroapellido.pd.***
- grabacion **grabacion_caja_de_ritmos-*vuestronombre_vuestroapellido-1***

- grabacion **grabacion_caja_de_ritmos-*vuestronombre_vuestroapellido-2***
- grabacion **grabacion_caja_de_ritmos-*vuestronombre_vuestroapellido-3***

Si has hecho varios patches diferentes o alguna grabación mas también puedes subirlo. Máximo 3 patches y 6 grabaciones.

Figuras:

Figura 1. Contador de 8 tiempos.

Figura 2. Ocho toggles en fila.

Figura 3. Objeto select.

Figura 4. Patch *select.pd.*

Figura 5. Objeto "select" que vamos a utilizar en esta práctica.

Figura 6. Objeto "sel" que acaba de recibir un 6 en su inlet y envía un bang por el séptimo outlet. (el 6 es el séptimo argumento).

Figura 7. Cuando el objeto "select" reciba un 0 por su inlet, un bang saldrá por su primer outlet y enviara el mensaje que contiene "1" al inlet izquierdo del objeto "==".

Figura 8. El primer paso del control de este instrumento esta activado. Cuando el primer toggle este activado se enviará un "1" al inlet derecho del objeto "==", cuando el toggle este desactivado enviará un "0".

Figura 9. Comprobación del estado del instrumento en el paso en el que se encuentra el contador (primer paso). En ese paso el instrumento esta activado por lo que tras comparar con el "==" y clasificar con el "sel 0 1" obtenemos un bang por el segundo outlet de objeto "sel 0 1".

Figura 10. Patch que compara el estado del instrumento en cada uno de los 8 pasos que genera el contador. Cada vez que un paso del instrumento este activado se enviara un bang para activar el sonido.

Figura 11. Subpatch que contine gran parte de la estructura de la figura 10.

Figura 12. Estructura de control de la figura 10 metida en un el subpatch "pd step-state" de la figura 11.

Figura 13. La orquesta de camino a vuestro patch de pd. https://giphy.com/gifs/perfect-band-member-11MS2pksWxB8SA

Figura 14. Subpatch para el kick drum o bombo.

Figura 15. Subpatch para snare drum o caja.

Figura 16. Subpatch para hi hats o platillos.

Figura 17. El subpatch del kick drum seleccionado.

Figura 18. Subpatch para el sonido procedente de un archivo.

Figura 19. Subpatch para snare drum o caja con el control de volumen general del instrumento marcado en rojo.

Figura 20. El patch que habéis construido para vuestra caja de ritmos.

Efectos