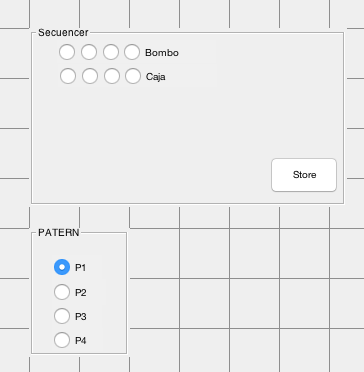
**Un secuenciador con interfaz grafica**

****Se desea crear un secuenciador básico que tenga una interfaz grafica que permita 4 botones para bombo y 4 para caja (uno por cada negra) agrupados en un panel llamado Secuencer los que el usuario puede colocar en cualquier configuración y que serán guardados al presionar un botón en 4 paterns distintos que a su vez son llamados por otros 4 botones de radio agrupados en un panel de botones llamado Patern, donde solo uno puede estar presionado a la vez.

Al iniciar todos los patrones están en blanco y se van llenando de datos mientras el usuario crea las secuencias de bombo-caja y usa el botón Store.

Por el momento no ejecutara sonidos, solo se almacenaran y llamaran los paterns.

Creamos una variable global P que será el numero de Patern en uso (que corresponde al botón apretado) luego ponemos en el callback de cada botón de radio correspondiente a un patern el código siguiente cambiando el valor de P según corresponda.

global P

P=1;

Además conviene que en el **callback OpeningFcn** que se crea de manera automática al guardar el .fig coloquemos el valor inicial la variable global P en 1.

global PP P

PP(1,P)=get(handles.radiobutton1,'Value');

PP(2,P)=get(handles.radiobutton2,'Value');

PP(3,P)=get(handles.radiobutton3,'Value');

PP(4,P)=get(handles.radiobutton4,'Value');

PP(5,P)=get(handles.radiobutton14,'Value');

PP(6,P)=get(handles.radiobutton15,'Value');

PP(7,P)=get(handles.radiobutton16,'Value');

PP(8,P)=get(handles.radiobutton17,'Value');

En el botón Store creamos un código que lea el estado de cada botón y lo almacene en una variable que llamaremos PP que es en este caso será una matriz de 8X4 donde las 8 filas serán el estado de los botones de radio del secuenciador y las 4 columnas serán cada una un patern.

Además debemos inicializar el valor de PP en el **callback OpeningFcn** en este caso como una matriz de 8X4 llena de ceros usando:

global PP P

PP=zeros(8,4);

P=1;

Finalmente agrandamos el código dentro de cada botón de patern para que lea el estado en que esta anteriormente

global P PP

P=1;

set(handles.radiobutton1,'Value',PP(1,P));

set(handles.radiobutton2,'Value',PP(2,P));

set(handles.radiobutton3,'Value',PP(3,P));

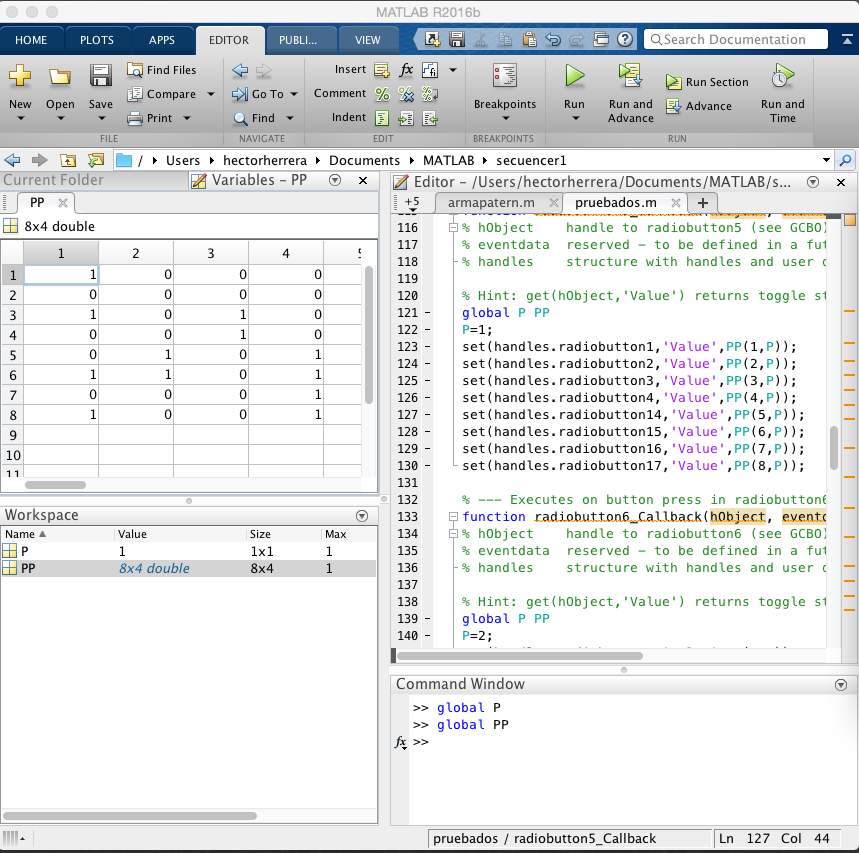
set(handles.radiobutton4,'Value',PP(4,P));

set(handles.radiobutton14,'Value',PP(5,P));

set(handles.radiobutton15,'Value',PP(6,P));

set(handles.radiobutton16,'Value',PP(7,P));

set(handles.radiobutton17,'Value',PP(8,P));

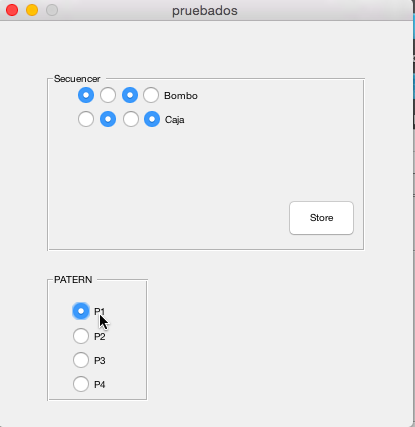
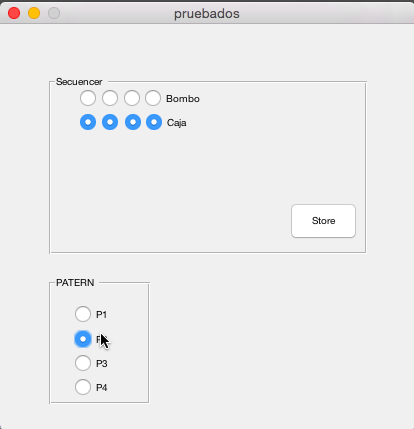
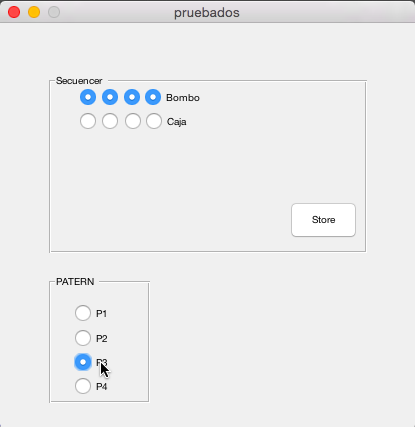
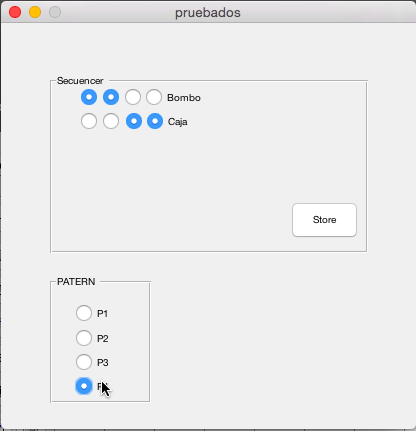
Podemos ir comprobando el avance de las operaciones usando global PP en la ventana de comando,

Y viendo que ahora PP aparece en workspace, de donde podemos abrirla para inspeccionar su contenido en tiempo real y podemos ver como cambia al apretar el botón Store en nuestra interfaz grafica.

En PP las primeras 4 filas son el bombo y las ultimas 4 la caja, cada columna es un patern distinto.

Si todo esta OK podemos probar el funcionamiento de nuestro secuenciador guardando 4 paterns distintos y luego llamándolos.

Aquí podemos ver una prueba:

Ahora haremos que suenen los paterns creados.

Cargamos un sonido de bombo y caja previamente ajustados a una duración estándar de unos 4000 samples (100 ms aprox)

Usar sonidos en <https://todoelectroacustica.blogspot.com/search?updated-max=2018-05-25T19:13:00-04:00&max-results=7>

Se deben bajar y colocar en el mismo directorio del .fig y .m luego formatear con este script:

Este script que lee los audios y luego los lleva a mono de 4000 samples en total con un fade-out de 1000 samples.

clc,clear

[bombo,fs]=audioread('Bd 5.wav');

[caja,fs]=audioread('Sn 4.wav');

env=[ones(3000,1); linspace(1,0,1000)'];

env=env.^2;

bombo=bombo(1:4000,1).\*env;

caja=caja(1:4000,1).\*env;

subplot(211);plot(bombo)

subplot(212);plot(caja)

audiowrite('bombo.wav',bombo,fs)

audiowrite('caja.wav',caja,fs)

Luego los guarda como ‘bombo.wav’ y ‘caja.wav’

Estos audios serán cargados en nuestra interfaz grafica.

global PP P bombo caja fs

PP=zeros(8,4);

P=1;

[bombo,fs]=audioread('bombo.wav');

[caja,fs]=audioread('caja.wav');

Se recomienda hacerlo en el **callback OpeningFcn** para que así estén disponibles desde el inicio.

Usaremos

pb=[]; pc=[]; T=15000; N=4000;

% N es la duracion en samples de cada audio

% T es el tempo del patern

for i=1:4 % armado de cada pista del partern

if PP(i,P)==1

pb=[pb;bombo;zeros(T-N,1)];

else

pb=[pb;zeros(T,1)];

end

if PP(i+4,P)==1

pc=[pc;caja;zeros(T-N,1)];

else

pc=[pc;zeros(T,1)];

end

end

sound(pb+pc,fs)

Luego hacemos que el botón Store tenga además la funcionalidad de crear el patern de audio usando la misma estructura vista en la guía sobre secuencias y funciones publicada anteriormente. Este código se agrega al que ya existe en el botón Store

**Ya funciona nuestro secuenciador básico!!!**

1. Agregue ajuste de TEMPO mediante ingreso del BPM
2. Se deben tener 8 tiempos por cada pista (el ejemplo es con 4)
3. Se deben poder generar 8 paterns distintos. (en el ejemplo son 4)
4. Agregue pistas para hihat cerrado y hihat abierto
5. Agregue gráficos de la forma de onda de cada patern una vez generado
6. Agregue faders para controlar el nivel de cada instrumento.
7. Agregue la posibilidad de elegir entre 3 tipos de bombo, 3 de caja y 3 de hihat.
8. Agregue pequeños gráficos para ver la forma de onda cargada en cada pista.
9. Agregue un sistema para poder crear un song que permita enlazar 8 paterns a gusto del usuario.
10. Agregue gráficos de la forma de onda del song completo una vez generado.
11. Agregue un botón para salvar como audio .wav el song creado.
12. Agregue un cursor que se mueva sobre la forma de onda a medida que se reproduce el audio de cada patern o del song

**Este trabajo se debe entregar el próximo jueves 14 de junio a las 18:40 como EPR2.**

**Será revisada ese mismo día.**