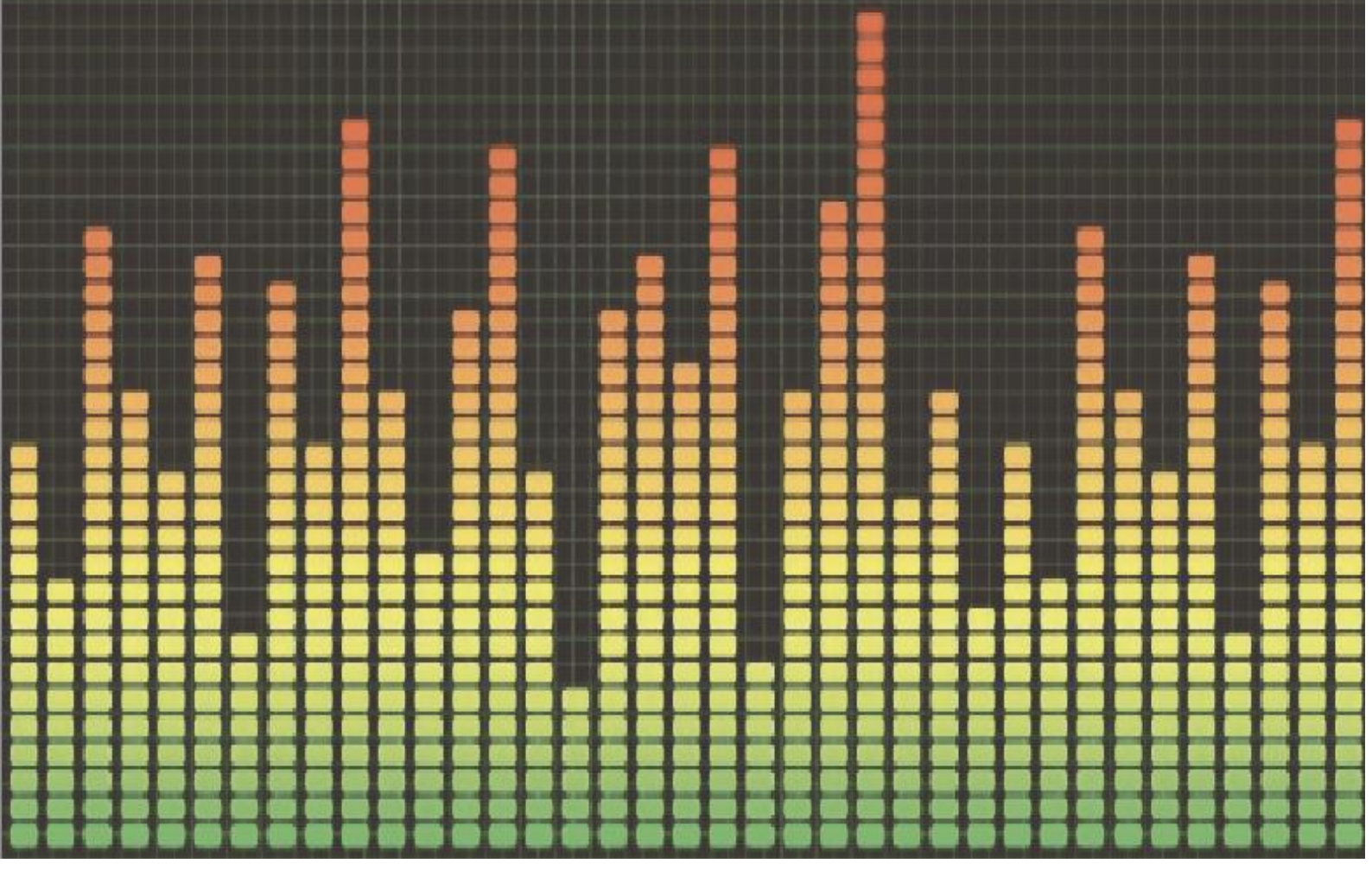


# laSalle

## LSTools

Pràctica PDS – Fase 2  
2023-24



## Índex

<b>1. INTRODUCCIÓ.....</b>	<b>2</b>
<b>2. DESCRIPCIÓ DE LA PRÀCTICA.....</b>	<b>2</b>
<b>3. INTRODUCCIÓ A APP DESIGNER .....</b>	<b>3</b>
<b>4. FASE II.....</b>	<b>8</b>
4.1 GUI .....	13
4.2 Processament a temps real dels efectes .....	15
<b>5. ENTREGA.....</b>	<b>17</b>

## 1. INTRODUCCIÓ

L'estudi de gravació de La Salle Campus Barcelona vol crear un plugin nou per l'eina d'edició sonora LSAudioTool que permeti equalitzar o aplicar reverberació al senyal de les produccions realitzades pels alumnes i es pugui utilitzar en temps real en les entrevistes del canal de televisió SalleVisión. Per aquest motiu, s'han posat en contacte amb els estudiants d'enginyeria que cursen Processat de la Senyal perquè els ajudin a aconseguir-ho fent el primer prototip en Matlab.



**Per la realització de la Fase 2 de la pràctica és imprescindible que verifiqueu que disposeu de les llibreries següents: Signal Processing Toolbox, Audio Toolbox, DSP System Toolbox.**

## 2. DESCRIPCIÓ DE LA PRÀCTICA

La pràctica es dividirà en dues fases que s'han d'implementar de manera seqüencial pel correcte funcionament d'aquesta. Aquesta pràctica s'ha de realitzar en grup de 3 persones de la mateixa classe.

- **FASE I: Disseny dels diferents efectes (equalitzador basat en un banc de filtres IIR, i reverberació).**
- **FASE II: Implementació del plugin d'efectes amb una Graphical User Interface (GUI) que permeti veure i escoltar a temps real el resultat.**

A l'hora de qualificar la pràctica es valorarà cadascuna de les fases per separat i la pràctica en conjunt tenint en compte que cada part es pondera amb un 50% de la pràctica. Es molt important complir amb els requeriments de cada fase i els requeriments finals de la pràctica.

Fase I	50%
Fase II	50%

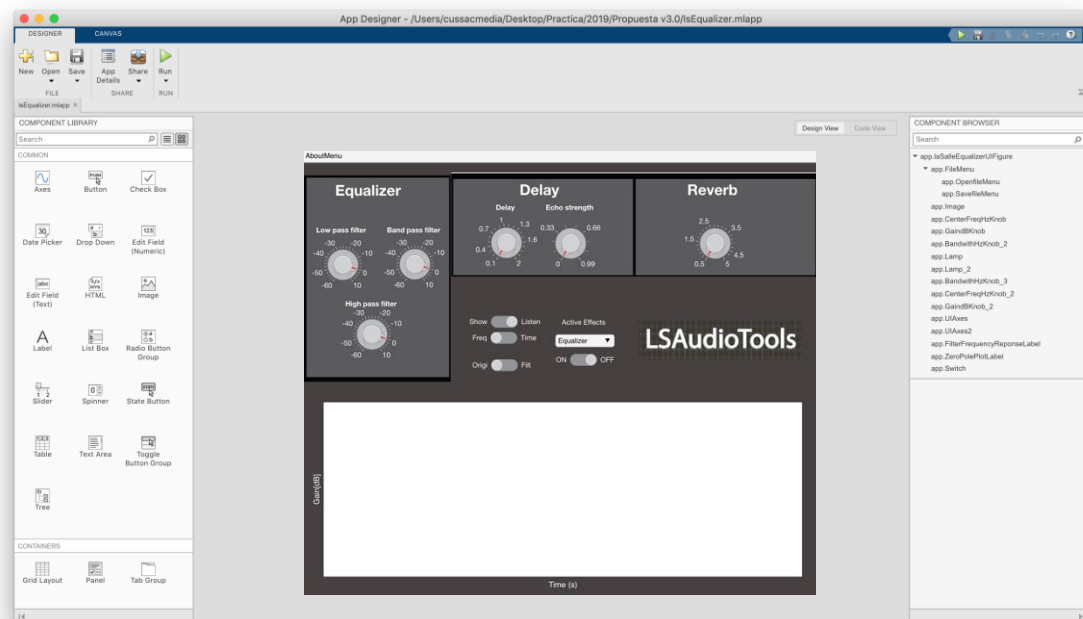
La entrega de la **primera fase** serà el dia **22/12/23 sobre 10**, o bé **14/1/23 sobre 7**.

La **segona fase** s'entregarà el dia **14/1/23**.

### 3. INTRODUCCIÓ A APP DESIGNER

App Designer es una nova eina que ofereix el software el software MATLAB i que integra les dos principals funcions que implica la creació d'una app: l'organització dels components visuals i interactius en una interfície gràfica d'usuari (GUI) i la programació del comportament de la app i de cada component que aquesta inclou. Per tant, es l'entorn recomanat per crear apps en MATLAB.

Per obrir App Designer podem fer-ho de dos formes: des del menú APPS de Matlab seleccionant l'opció Design App, o escrivint **appdesigner** des de la línia de comandes.



*Figura 1: Entorn de treball de l'App Designer.*

Aquesta és la interfície principal de l'App Designer. En la cantonada superior dreta podem observar que podem treballar de dos maneres diferents:

- **Design View:** Ens permet veure la versió gràfica de la interfície incorporant diferents elements arrossegant els components de la Component Library a la interfície.

- **Code View:** És la vista de programador, a on podem programar el codi que s'executarà a l'interactuar amb els diferents elements de la interfície.

Si està activada la pestanya Design View, podem trobar els següents panells:

1. **Design Editor.** Àrea sobre la qual es distribueixen els diferents elements de control i que constitueix una representació del que es veurà en pantalla a l'executar la interfície.
2. **Component Library.** Llista dels diferents components disponibles pel disseny de la aplicació. Aquests es troben classificats en *common components*, *containers*, *figure tools* i *instrumentation*. Per utilitzar-los, només cal arrossegar els elements elegits per l'aplicació al Design Editor.

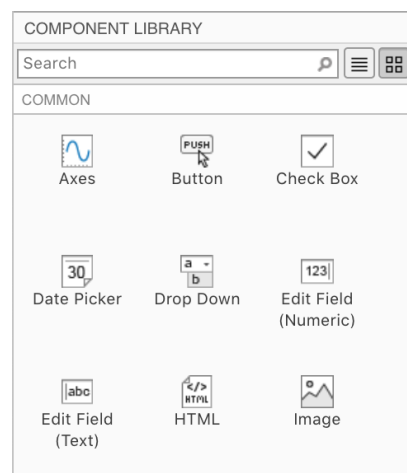











Figura 2: Component Library.

Pel cas de la nostra aplicació, ens interessa conèixer els següents components. No obstant, es convida als alumnes a que jugueu amb els altres components que ofereix el programa.

ICONO	COMPONENTE	DESCRIPCIÓN
	Axes	Permet crear uns eixos o gràfiques. En el nostre cas utilitzarem una gràfica.
	Image	Permet incorporar una imatge. La nostra interfície ja té una imatge de fons.
	Button	Permet incorporar botons e interactuar amb ells.
	Label	Permet crear una etiqueta amb text.
	Knob	Potenciòmetre que determina el valor d'un paràmetre de control (p.ex. el guany d'una banda de l'equalitzador). En el nostre cas en



		tindrem 6 per poder modificar els paràmetres de cada efecte (igual que a la fase 1), els knobs han de tenir com a valors extrems els mateixos que apareixen a la figura 1.
	Menu Bar	Menú desplegable que ens permet obrir i guardar arxius.
	Switch	Permet seleccionar entre dos o més opcions. En el nostre cas utilitzarem només dos.
	Drop Down	Desplegable per escollir entre varies opcions predeterminades d'un menú.
	Edit Field	És un camp d' introducció de caràcters numèrics.

3. **Button Properties.** Finestra amb les propietats de cadascun dels objectes utilitzats en el Design Editor. Un cop seleccionats el components es mostren les característiques (diferents per cada tipus d'objecte) que poden ser modificades.

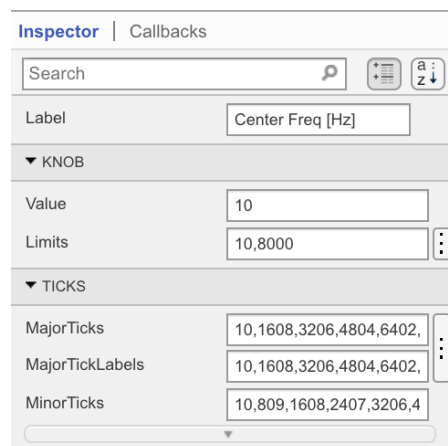


Figura 3: Button Properties.

Si està activada la pestanya Code View, podem trobar els següents panells:

1. **Code Editor.** Editor que conté tant el codi generat automàticament després d'afegir objectes com el dels diferents *callbacks*, funcions i propietats implementades. Aquest codi és el que haurem de modificar/ampliar perquè la nostra interfície acabi realitzant les funcionalitats del nostre plugin d'àudio.

2. **Code Browser.** Llista amb els diferents *callbacks*, funcions d'utilitat i propietats creades en la aplicació.

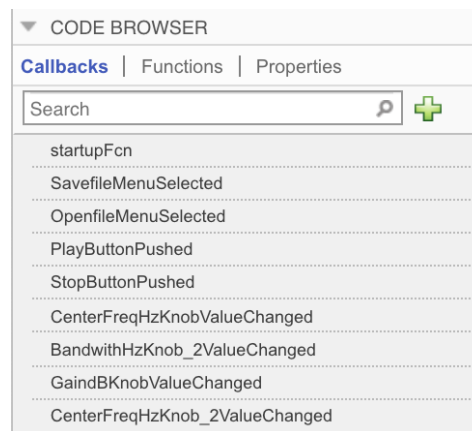


Figura 4: Code browser.

3. **Toolbar.** Correspon a la barra d'eines. Aquesta es divideix en dos pestanyes. Els principals elements d'aquesta barra d'eines son:

ICONO	COMPONENTE	DESCRIPCIÓN
	Save	Guarda la aplicació amb la que s'està treballant.
	Callback	Crea un <i>callback</i> per un determinat element de la interfície. Es tracta d'un petit codi que s'executarà cada cop que l'usuari modifiqui l'estat de l'element (p.ex. si modifiquem el guany volem veure un canvi en la funció de transferència).
	Function	Crea una funció d'utilitat.
	Property	Crea una propietat.
	Run	Executa l'aplicació.

A continuació us indiquem el workflow que haureu de complir dins de la aplicació del vostre plugin d'àudio creat amb l'eina App Designer, que seria el següent:

1. Crear la finestra base de la nostra interfície del plugin d'àudio. En el cas de la nostra aplicació, es podria afegir una imatge de fons equalitzador que donés un cert estil (skin) per aquest tipus d'aplicacions d'àudio.

2. Afegir els diferents components a la interfície. Podeu canviar el nom dels components per identificar-los de manera més ràpida quan editeu les *callback function* de l'aplicació..
3. Afegir una funció *callback* per a cada component de la nostra interfície dins del codi del Code Editor que s'executarà cada cop que interactuem amb aquest component.

En l'apartat següent se us explica amb més detall com portar a la pràctica aquest workflow genèric pel cas pràctic del nostre plugin d'àudio.

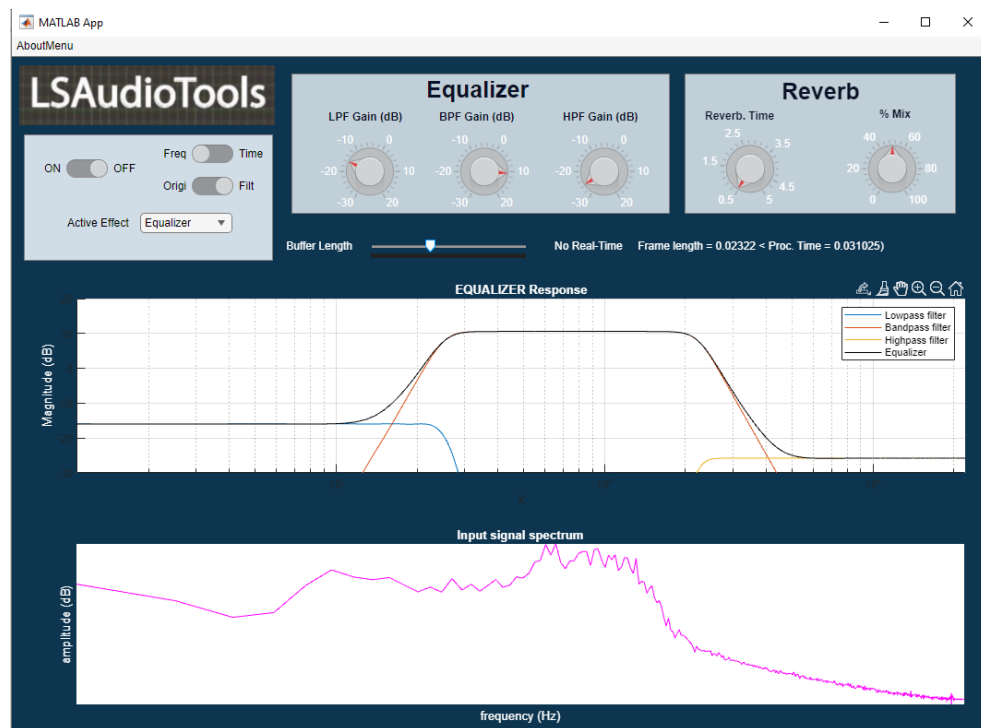
En el següent vídeo podreu veure un petit tutorial de funcionament de l'App Designer, com a material complementari a aquesta memòria, molt recomanable per abans de realitzar aquesta pràctica:

[Link al vídeo sobre App Designer](#)



## 4. FASE II

La Fase II consisteix en la realització d'una interfície gràfica per controlar els efectes d'àudio i poder-los aplicar al senyal d'un micròfon en temps real (plugin d'àudio): equalitzador de tres bandes i reverberació, ja implementats i testejats a la fase I. La freqüència de mostratge que usareu serà la de àudio de 44100 Hz (usada pel disseny dels dos efectes). A la figura 5 es mostra una possible forma que podria tenir la interfície.



*Figura 5. Exemple d'interfície d'usuari de l'aplicació dissenyada.*

Poseu els límits de -20 i 30 dB pels controls dels guanys de l'equalitzador, 0.5 i 5 pel temps de reverberació i de 0 a 100 pel factor de mescla del reverberador.

Per poder realitzar el funcionament dels efectes a temps real, serà necessari que useu un micròfon i uns auriculars per tal d'evitar que l'àudio que prové del micròfon processat es realimenti a través de l'escolta del mateix senyal. Si no ho feu així, el més probable és que l'àudio acabi saturant degut a aquest efecte de realimentació positiva, que és un efecte bastant desagradable i molest.

Matlab disposa d'una llibreria nativa (funcions `audioDeviceReader` i `audioDeviceWriter`) que permet utilitzar la targeta de so de l'ordinador per poder processar l'àudio a temps real (llegir trames del convertidor A/D que connecta amb el micròfon del PC i escriure trames cap al convertidor D/A que connecta amb el dispositiu de sortida – auriculars connectats al PC en el nostre cas). Degut als alts recursos que necessita el software per

realitzar el processament gràfic, es treballarà en dos modes de forma separada, fent ús d'un switch anomenat "onoff". Per tant, per evitar saturar de càlculs la CPU de l'ordinador, el funcionament que ha de realitzar la aplicació és el següent:

- a) Switch "onoff" igual a "OFF". Només abans d'activar el processament d'àudio, s'ha de poder modificar els paràmetres de control del comportament dels dos efectes (guanys en dB de les 3 subbandes de l'equalitzador, i temps de reverberació i factor de mescla de l'efecte de reverberació). En aquest mode podrem escollir un o altre efecte amb un component de tipus Drop Down anomenat EffectDropDown (vegeu etiqueta Active Effect a la figura 5), amb les opcions "Equalizer" i "Reverb", de forma que un cop escollit l'efecte podrem modificar els seus paràmetres (usant un Knob per cadascun d'ells) i en un eix o figura gràfica se'ns mostrarà de forma interactiva el disseny realitzat: per l'equalitzador se'ns mostrarà el guany en dB del filtre i de les 3 subbandes en el domini de la freqüència (en eix logarítmic), mentre que per la reverberació se'ns mostrarà la resposta impulsional en un eix de temps que vagi de 0 a 5 segons (en aquest cas, el factor de mescla no afectarà a la visualització en el mode "OFF").
- b) Un cop activem el processament a temps real (valor del switch a ON) ja no podrem modificar cap dels paràmetres i aquests apareixeran de manera bloquejada fins que apaguem l'efecte (OFF). En aquest mode ON podrem sentir pels auriculars el senyal del micròfon processat per l'efecte que haurem prèviament seleccionat i particularitzat (els seus paràmetres propis) en el mode OFF, i alhora també se'ns mostrarà per un segon eix o figura gràfica un dels dos senyals (original o filtrat) en un dels dos dominis (temporal o en freqüència). Per tant, durant el mode OFF haurem hagut abans de seleccionar la posició de dos switchos que anomenarem Switch\_freq\_time (per escollir si volem visualitzar el senyal en un o altre domini) i Switch\_original\_filt (per escollir si volem veure el senyal del micròfon o el senyal ja processat amb l'efecte).

A més, també caldrà incloure un darrer component que permetrà seleccionar la mida del buffer que usareu a l'hora de realitzar el processament a temps real: un slider amb nom BufferLengthSlider. Escolliu com a valors mínim 1024 i màxim 10240. El processament a temps real es basa en agafar la trama de mostres del buffer i processar-la abans que s'esgoti el temps que dura la mateixa trama. En cas que això no s'acompleixi el procés no funciona a temps real, de forma que abans d'acabar l'execució del processament de la trama el sistema passarà a tornar a omplir de nou el buffer, provocant discontinuïtats en la sortida. En el processament de la trama, usarem la crida

a la funció `filter.m`, la qual permet accedir a les condicions inicials i finals del filtre per a poder donar continuïtat d'una trama a l'altra.

A continuació us indiquem les variables globals que s'han de generar (podeu generar-ne més si voleu). Són les següents:

- `deviceReader` → objecte que extreu les trames d'àudio del conversor A/D.
- `deviceWriter` → objecte que envia la trama d'àudio amb efecte cap al conversor D/A.
- `fs` → freqüència de mostratge a la que han de treballar els conversors A/D i D/A (valor 44100 Hz).
- `audio` → vector que conté una trama d'àudio capturada pel conversor A/D.
- `audioEffect` → vector que conté una trama d'àudio processada amb l'efecte corresponent per a enviar al conversor D/A.
- `audiof` → vector que conté l'àudio d'entrada o amb efecte en el domini de la freqüència (per a fer la seva visualització en la segona figura de la interfície)
- `SOS_LP`, `SOS_BP`, `SOS_HP` → Matius de coeficients de les seccions d'ordre 2 corresponents al disseny les 3 subbandes de l'equalitzador (disseny original).
- `G_LP`, `G_BP`, `G_HP` → Vectors de guanys de les seccions d'ordre 2 corresponents al disseny les 3 subbandes de l'equalitzador (disseny original).
- `G_LPF`, `G_BPF`, `G_HPF` → Valor dels guanys lineals que apliquem a les 3 subbandes de l'equalitzador (s'hauran de convertir de dB a valors lineals).
- `heq_length` → longitud en mostres de la resposta impulsional de l'equalitzador.
- `zief` → registre de condicions inicials/finals del filtre equalitzador.
- `zirev` → registre de condicions inicials/finals del filtre reverberador.
- `h_reverb` → resposta impulsional del filtre reverberador.

Com a ajuda per a la implementació d'aquest prototip, a continuació es detalla el codi que cal posar en la funció de callback del switch "onoff", tant quan s'activa la posició de *switch* ON per activar el processament en temps real del so, com per parar el seu processament al prémer el OFF.



```
% Value changed function: onoff
function onoffValueChanged(app, event)
if strcmp(app.onoff.Value,'ON')
    % Inicialitzem els buffers de la reverb i l'equalitzador
    % per bandes
    app.zirev = zeros(1,length(app.h_reverb)-1);
    app.zieq = zeros(1,app.heq_length);
    % Inicialitzem la comunicació amb els conversors AD i DA, i
    % fixem la freqüència de mostreig a 44100 Hz
    app.deviceReader = audioDeviceReader(app.fs, app.BufferLengthSlider.Value);
    app.deviceWriter = audioDeviceWriter('SampleRate',app.deviceReader.SampleRate);
    % Bloquegem tots els altres controls de la interfície menys
    % l'onoff
    app.blockbuttons(false)
    % Inicializem comptador de temps
    tic
    try
        while strcmp(app.onoff.Value,'ON')
            % Llegim una trama del conversor AD
            app.audio = app.deviceReader();
            % Processem una trama
            processAudio(app)
            % Escrivim la trama processada al conversor DA
            app.deviceWriter(app.audioEffect);
            % Dibuixem el senyal d'entrada/sortida en el domini
            % temporal/freqüencial
            freq_temp(app);
            % Parem comptador de temps
            totalTime = toc;
            tic
            pause(0.001);
            % Treiem missatge en la etiqueta de la interfície
            % RTLabel sobre si el procés va a temps real o no
            if (totalTime < (app.BufferLengthSlider.Value/app.fs))
                app.RTLabel.Text = [' Real-Time   Frame length = ',num2str(app.BufferLengthSlider.Value/app.fs)];
            else
                app.RTLabel.Text = [' No Real-Time   Frame length = ',num2str(app.BufferLengthSlider.Value/app.fs),' < Proc. Time = ',num2str(totalTime),'];
            end
        end
    catch
        app.RTLabel.Text = ['Real-time effect stopped'];
    end
    % Desbloquegem tots els controls de la interfície
    app.blockbuttons(true)
    % Finalitzem la comunicació amb els conversors AD i DA
    release(app.deviceReader);
    release(app.deviceWriter);
end
```

Figura 6. Codi del callback del switch “onoff”

En el codi anterior es comença inicialitzant els buffers del reverberador i de l'equalitzador així com els objectes associats amb els conversors A/D i D/A. Seguidament es bloquegen tots els objectes interactius llevat del botó “onoff” amb la crida a *blockbuttons()*, i es crida a un procés iteratiu (*while*) a on es llegeix el buffer del conversor A/D, aquest es processa amb l'efecte, i el resultat s'envia al conversor D/A i es visualitza el senyal escollit prèviament. A més, també es visualitza en la interfície gràfica (etiqueta amb nom RTLabel) un missatge que indica si el temps invertit en el processament de la trama d'àudio és inferior a la seva durada (en aquest cas s'indica que és possible el processament a temps real). Quan això no passi, la etiqueta ho indicarà i es podrà sentir l'àudio entretallat. Per evitar aquests talls, es podrà modificar la mida del buffer amb el valor de l'slider BufferLengthSlider. Normalment mides de buffer grans permeten que el processament a temps real sigui més factible, però en contrapartida, el retard que s'introdueix és major (aquest és com a mínim la mida d'aquest buffer passada a segons).

Del codi anterior, es pot veure que hi ha tres funcions privades que s'han d'implementar, més una quarta que es cridarà des d'una d'aquestes:

- 1) *blockbuttons(flag)* → funció que bloqueja o desbloqueja en funció del valor del boolean *flag* tots els botons i interruptors que permeten modificar el filtre o gràfiques. Hi ha un atribut, per cada knob/switch, que fa d'enable de cadascun dels controls de la interfície, l'atribut és el següent, per exemple, knob.enable = true.
- 2) *processAudio()* → funció que processa cada trama d'àudio que llegeix el FileReader en funció del efecte seleccionat. Aquesta funció serà molt semblant a la ja implementada a la Fase 1, però amb algunes diferències que se us expliquen més endavant, degudes al fet que estem processant una trama d'un flux continu de àudio, i cal garantir la continuïtat del procés de filtratge digital. Dins d'aquesta el primer de tot serà cridar a una altra funció amb nom *getCoeficients()* per actualitzar algunes variables globals. En segon lloc, en funció de l'efecte a aplicar (valor del component EffectDropDown), es realitzarà un o altre filtrat<sup>1</sup>.
- 3) *freq\_temp()* → funció que s'encarrega de visualitzar en temps o en freqüència el senyal filtrat o original per cada trama d'àudio.
- 4) *getCoeficients()* → funció que s'encarrega d'actualitzar el valor de les variables globals G\_LPF, G\_BPF, G\_HPF, i h\_reverb a partir dels valors que l'usuari ha modificat en els components de la interfície.

Durant el mode ON, per la visualització de el senyal d'entrada o processat en el domini de la freqüència dins la funció *freq\_temp()*, es proposa que apliqueu un suavitzat temporal de la transformada de Fourier fent ús de la següent equació:

$$\text{app.audiof} = \text{app.audiof} * 0.7 + 0.3 * \text{abs}(\text{audiof}(1:\text{nf})).^2;$$

a on *nf* és el número de punts que usem en la visualització i *audiof* és la representació del buffer d'entrada o amb efecte (segons el valor de Switch\_original\_filt) en el domini de la freqüència. D'aquesta forma, *app.audiof* s'haurà d'inicialitzar a zero abans d'iniciar el filtratge i amb aquesta equació s'aconseguirà que en la visualització de la visió en freqüència els canvis temporals siguin més suaus. Per a la visualització tenir en compte que fer la crida a  $10 * \log_{10}(\text{app.audiof})$  per convertir-la a un guany en dB.

---

<sup>1</sup> Vegeu més detalls a l'apartat 4.2.

## 4.1 GUI

Com ja s'ha explicat, l'objectiu serà la creació de una interfície d'usuari (GUI) que permeti capturar l'àudio del micròfon, aplicar un efecte dels 2 ja treballats a la Fase 1, seleccionar per l'usuari els paràmetres dels efectes i finalment escoltar i visualitzar l'àudio amb l'efecte seleccionat. El disseny de la interfície és lliure, però ha de complir les següents restriccions:

- L'arxiu que crea App Designer ha de dir-se `IsAudioTool.mlapp` i ha d'estar a la carpeta del projecte amb el resta dels elements.
- Tres **Discrete Knobs** o **Sliders** que ens permetin ajustar el guany de cadascun dels filtres de l'equalitzador.
- Dos **Discrete Knobs** o **Sliders** que ajustin el temps de reverberació i el factor de mescla en % de l'efecte de reverberació.
- Un **Slider** que ajusti la mida del buffer per realitzar el processament a temps real del senyal d'àudio.
- Heu de presentar **dues gràfiques (Axes)**. La primera haurà de mostrar la interactivitat en el mode OFF, quan l'usuari modifica els guanys en dB de l'equalitzador (la resposta en freqüència de les 3 subbandes i la de l'equalitzador global canviarà cada cop que es modifiqui un dels 3 valors) o el temps de reverberació de l'efecte reverb (la resposta impulsional en un eix temporal de 0 a 5 seg. s'actualitzarà cada cop que es modifiqui aquest valor). La segona, en canvi, ha de mostrar el senyal d'entrada o bé el senyal filtrat, en el domini temporal o bé en el freqüencial, mentre es processa el senyal en el mode ON. Les gràfiques en el domini freqüencial han de tenir un eix logarítmic en Hz, mentre que la temporal ha d'estar en segons i correspondre a la trama actual. En el domini de la freqüència s'aplicarà el suavitzat temporal de la transformada de Fourier explicat al final de l'apartat anterior.
- Un **Switch** que permeti canviar entre representació en domini temporal o domini freqüencial (tot i que aquest només es pot modificar durant el mode OFF, el seu valor afectarà a la representació del senyal escollit en la gràfica segona durant el mode ON).
- Un **Switch** que ens permeti visualitzar el senyal d'àudio original o el senyal d'àudio amb efecte (tot i que aquest només es pot modificar durant el mode OFF, el seu valor afectarà a la representació del senyal escollit en la gràfica segona durant el mode ON).
- Les següents funcions privades amb el funcionament anteriorment descrit: **`blockbuttons(flag)`**, **`processAudio()`**, **`freq_temp()`**, **`getCoefficients()`**
- La funció provada **`grafica1()`** que s'explica en la pàgina següent.



- Tots els **callbacks()** corresponents als diferents elements de la interfície i la corresponent interacció. Cal recordar que durant el mode ON la interfície ha de permetre experimentar visualment en la figura 1 els canvis realitzats quan modifiquem qualsevol dels paràmetres de l'efecte d'àudio.
- Definir una **Property()** privada amb totes les variables globals a emprar.

**NOTA: És important que visualitzeu el vídeo sencer d'introducció a l'app designer ja que s'expliquen punts molt importants i de gran ajuda, per exemple a partir del minut 5 s'explica com crear els callbacks i propietats.**

Un cop incorporats tots els elements a la interfície, passarem a treballar a nivell de codi. El primer pas serà crear la property privada amb totes les variables globals del sistema.

Per accedir a una variable global del sistema o a un element de la interfície, posem sempre la paraula app.(i el nom de la variable) per tenir el seu valor.

Un cop realitzat aquest pas, a la funció privada inicial (*function startupFcn(app)*) llegir els fitxers .mat generats a la Fase 1 i guardar els coeficients en les variables globals corresponents de la app, ja que així podem accedir a aquestes en tot moment.

Per llegir o modificar el valor d'un component de la interfície, podem fer servir la comanda "app.nom\_element.Value".

A partir d'aquí, cal implementar les funcions privades i els callbacks que permetin una interacció completa amb tots els elements de la interfície. Pel mode OFF, la interactivitat dels diferents elements que afecten a la resposta dels efectes d'àudio implicarà que cada cop que aquests elements es modifiquin s'haurà de cridar a una cinquena funció que anomenarem **grafica1()**. Dins d'aquesta funció es realitzarà la visualització de la resposta freqüencial del banc de filtres així com dels filtres individuals quan EffectDropDown.Value sigui 'Equalizer'<sup>2</sup>, i la visualització de la resposta impulsional del reverberador quan el seu valor sigui 'Reverb'.

---

<sup>2</sup> Es recomana usar la crida a la funció *strcmp.m* per a comparar dos strings.

## 4.2 Processament a temps real dels efectes

En aquest apartat s'explica gran part del codi que heu de posar dins de la funció *processAudio()*, especialment el que es refereix al procés de filtrat de la trama d'àudio d'entrada.

En la Fase I, disposàvem de tot el vector de l'àudio seguit i junt, ja que provenia d'un fitxer .wav emmagatzemat i, per tant, només aplicàvem el filtre de l'efecte un cop sobre tot el fitxer. Ara, en canvi, al tenir un codi que funciona en temps real, quan apliquem l'efecte a cada trama d'àudio haurem de guardar les condicions dels registres de memòria del filtre per a que aquests siguin els mateixos a l'inici del processament de la següent trama, perquè no es produeixi cap discontinuïtat.

Recordem una funció que permet filtrar de Matlab, i la forma de la crida que ens permet accedir al valor del registre del filtre abans i després de filtrar un vector o trama d'entrada:

$$[y,zf] = \text{filter}(B,A,x,zi)$$

A on:

**zi:** són les condicions inicials per als registres interns de memòria del filtre, abans d'aplicar el filtre a la trama x, especificades com a vector. La seva longitud ha de ser  $\max(\text{longitud}(A), \text{longitud}(B))-1$ .

**zf:** són les condicions finals per als registres interns de memòria del filtre (un cop processada la trama). La longitud és la mateixa que la de zi, però els valors que aquest tindrà seran probablement diferents dels d'aquest.

Per a la primera trama (inicialització) posarem a zi el valor per defecte amb un vector de retard igual a zeros. A partir de la segona trama farem que les condicions inicials d'una trama (Zi) siguin iguals a les condicions finals de la trama anterior (zf), que ens haurem d'haver guardat en alguna variable de la interfície. D'aquesta manera podrem filtrar correctament els tres efectes en temps real, sense cap tipus de discontinuïtat a l'àudio de sortida.

Per poder aplicar aquesta filosofia de treball als nostres 2 efectes (banc de filtres + reverb) abans caldrà disposar de la seva resposta impulsional h. Pel banc de filtres, el filtrat amb seccions d'ordre dos no ens permet treballar amb aquesta filosofia a no ser que ho fem dividint el filtrat en tants filtres com seccions tinguem, i usem les condicions inicials i finals de cada subfiltre. En comptes d'això, us recomanem que calculeu la resposta impulsional de l'equalitzador (recordeu que l'efecte tindrà uns valors dels tres

guanys prèviament escollits per l'usuari, pel que caldrà calcular aquesta resposta cada cop que canviïn aquests paràmetres), i en la crida anterior igualeu:

$$B = h$$

$$A = 1$$

Cal tenir en compte que per l'equalitzador la resposta impulsional no ha de ser massa llarga, així que mireu de verificar visualment que us quedeu amb la longitud tal que agafeu les mostres que son significativament majors o menors que zero (al ser un filtre estable, la seva resposta impulsional ha de tendir a zero per la dreta). Podeu fixar aquesta longitud a un valor constant visualitzant també les respostes de les 3 subbandes per separat.

## 5. ENTREGA

Com a resultat d'aquesta segona fase de la pràctica cal adjuntar en una carpeta comprimida en un .zip ( el qual ha de tenir el següent nom: PDS\_FaseII\_N.zip, essent N el vostre número de grup, p.ex. A3 o B6), els següents ítems:

- **Vídeo.** Enllaç al núvol d'un vídeo on es demostrí el correcte funcionament de l'aplicació dissenyada. Per tal afecte, ens haureu d'adjuntar un fitxer amb nom vídeo\_demo.txt amb el link al vídeo realitzat.
- **Recursos.** Incorporar en una subcarpeta tot el material extra que heu emprat per al disseny de la interfície (p.ex. imatge de fons, etc.).
- **IsAudioTool.mlapp.** Fitxer d'AppDesginer que funcioni sense errors.
- **filtersSOS.mat** Fitxers amb els coeficients dels filtres de l'equalitzador dissenyat.

A l'hora de realitzar la pràctica s'han de tenir en compte les consideracions següents:

- Cal complir totes les restriccions que marca cada apartat de l'enunciat.
- El codi de MATLAB s'ha d'executar per si mateix. No s'accepten codis que s'executin a trossos, o que tinguin errors d'execució.