Universitat Oberta de Catalunya

uoc.edu

Creació d'un multiefectes d'àudio sobre Raspberry Pi i Simulink

Arnau Mañosa i Martí
Grau en Enginyeria de Tecnologies i Serveis de Telecomunicació
Aplicacions multimèdia basades en processament del senyal

Lourdes Meler Corretjé
Jose Antonio Moran Moreno

Maig 2022

FITXA DEL TREBALL FINAL

Títol del treball:	*Creació d'un multiefectes d'àudio sobre Raspberry Pi i Simulink*
Nom de l'autor:	*Arnau Mañosa i Martí*
Nom del consultor/a:	*Lourdes Meler Corretjé*
Nom del PRA:	*Jose Antonio Moran Moreno*
Data de lliurament (mm/aaaa):	*06/2022*
Titulació o programa:	*Grau en Enginyeria de Tecnologies i Serveis de Telecomunicació*
Àrea del Treball Final:	*Aplicacions multimèdia basades en processament del senyal*
Idioma del treball:	*Català*
Paraules clau	*Processament d'àudio, Raspberry Pi, Simulink, Efectes, Temps real*

Resum del Treball (màxim 250 paraules):

El processament de senyal és un dels temes específics de la menció d'Audiovisuals dins el GETiST. Conjuntament amb el processament de vídeo, el processament d'àudio és l'aplicació concreta de processos i manipulacions sobre els senyals rebuts.

En el cas concret de l'àudio, el processat es pot fer mitjançant efectes temporals o freqüencials que afectin el senyal original de forma lineal o no lineal. D'aquesta forma aconseguim manipular el senyal i n'obtenim un altre en funció dels paràmetres que nosaltres desitgem.

Actualment, trobem dispositius hardware programats per fer aquesta tasca (tant destinats a senyals de veu com d'instruments) i també trobem opcions software. Però en ambdós casos és difícil trobar solucions que permetin la programació dels diversos efectes.

Aquest TFG planteja, doncs, la creació d'un dispositiu que permeti l'execució en temps real de diversos efectes per àudio. La programació es realitza sobre Simulink (un entorn de programació visual basat en MATLAB) i sobre un ordinador de baix cost (concretament un Raspberry Pi), de forma que es poden utilitzar en actuacions en viu sense necessitat d'un ordinador "convencional". A més a més, per tal d'ajustar els efectes, s'afegeixen diferents potenciòmetres per a la manipulació dels efectes en temps real.

Abstract (in English, 250 words or less):

Signal Processing is one of the specific subjects studied during the Audiovisual Mention in our Telecommunications Degree. Along with video processing, audio processing is the actual application of processes and manipulations on received signals.

In the specific case of audio, processing can be done by temporary or frequency effects that affect the original signal linearly on nonlinearly. Thus, we can manipulate the signal in order to obtain another resulting signal according to the desired parameters.

We currently have hardware devices built to do this task (for both Voice and musical instruments signals) and we also find software options, but in both cases it is difficult to find solutions that allow the programming and personalization of the various effects.

This TFG, therefore, proposes the creation of a Device that allows the real-time execution of various audio effects. Programming is done on Simulink (a visual programming environment based on MATLAB) and on a low-cost computer (specifically, a Raspberry Pi) so that they can be used in live performances without the need for a conventional computer.

In addition, and in order to adjust the effects parameters, different potentiometers are added for real-time manipulation.

Agraïments

Si tot té un inici i un final, aleshores diuen, l'important és confirmar l'interval. I si aquesta memòria és el final del viatge dins el Grau, l'interval ha sigut possible també a moltes persones que han ajudat a fer-lo més lleuger.

En primer lloc, gràcies a la tutora d'aquest Treball final, Lourdes Meler, per la paciència, per aguantar els fils de correu mal escrits i sempre a deshora. El procés ha sigut complex, però ben segur que ha valgut la pena.

Gràcies a tots els companys amb qui hem compartit entregues, exercicis i neguits. Alguns teniu cara, els altres sou completament virtuals, però tots heu sigut xarxa durant molt de temps.

Gràcies al Pare i a la Mare, a la Marta, al Lluc, a l'Alguer, al Juli i a l'Anna per ser-hi sempre, fins i tot quan no anava res com tocava. A la Marta, també, per les correccions d'aquesta memòria.

Gràcies a tots els que heu patit algun renec en forma de "p*** U*C", als Ovidi4 i als Despers per ser la vàlvula d'escapament durant, sobretot, aquests dos últims anys. Gràcies a la Irene i al Berni per tantes hores de conversa i al Rai també per la primera lectura de la memòria.

Gràcies a tu Guim, per arribar i trastocar-ho tot. Segurament no recordaràs aquests dos anys amb la mateixa intensitat que ho faré jo, però ha sigut un luxe poder-te acompanyar de tant a prop.

I gràcies a la Mireia, per confiar més en mi que jo mateix; per fer aquest camí (i tants d'altres!) més fàcil; per posar-hi il·lusió, energia i estructura també des del primer dia. Si les hores han estat meves, la responsabilitat de ben segur que és compartida. Gràcies sempre.

Índex

Llista de figures

Llista de taules

Llista d'equacions

1. Introducció

1.1. Context i justificació del treball

La motivació principal del TFG és la de posar en conjunt tots els coneixements adquirits durant el Grau i vincular-los amb el meu àmbit laboral mitjançant la creació d'un producte que sigui utilitzable en un entorn de producció musical en viu.

Amb la construcció del multiefectes es vol proveir d'una plataforma programable per al processat d'àudio a temps real que pugui ser utilitzable en entorns d'actuacions musicals en viu. Tot i requerir d'una llicència de Simulink, la majoria de components són de baix cost (el mateix ordinador on s'executarà), de forma que és fàcilment reproduïble.

Es podria haver escollit un altre llenguatge de programació, però no s'haurien aprofitat els coneixements acumulats al llarg del Grau amb el programari de MATLAB i també s'hauria vist incrementat el temps de programació enlloc d'aprofitar el potencial de l'entorn gràfic.

Per aconseguir el multiefectes es programaran els diferents efectes seleccionats sobre Simulink (recollint alguna de les activitats proposades durant l'assignatura de Processament d'àudio), es construirà l'electrònica per a la seva manipulació (recollint els coneixements tant de Teoria de Circuits com de Circuits Electrònics) i es configurarà el Raspberry Pi per a poder utilitzar-la de forma autònoma.

1.2. Objectius del Treball

L'objectiu principal del TFG és aconseguir un producte funcional que sigui capaç de processar àudio en temps real de forma autònoma, com s'ha comentat anteriorment, es vol proveir d'una plataforma programable per al processat d'àudio que pugui ser utilitzable en entorns d'actuacions musicals en viu i també, en el mon educatiu.

Un segon objectiu és la realització de la memòria del TFG on es documenti el procés, l'estat de l'art, la investigació i les conclusions realitzades per l'estudiant.

1.3. Enfocament i mètode seguit

Per tal d'acomplir el primer objectiu plantejat en l'apartat anterior, s'escullen la plataforma de programació visual Simulink i un ordinador de baix cost com el Raspberry Pi per a tal de portar-lo a terme.

D'una banda, la plataforma de programació visual Simulink ens permet utilitzar blocs preprogramats per a la manipulació de l'àudio i també comunicar-nos amb el hardware on s'executi. Així d'una forma més transparent, ens estalviem una gran quantitat d'hores de programació que hauríem de dur a terme si el llenguatge fos un altre (Python, per exemple).

D’altra banda s’escull la plataforma de maquinari Raspberry Pi per diversos factors, el primer i més important, perquè és de baix cost (uns 40€ per unitat), perquè és fàcilment programable amb Simulink i perquè és fàcil d’afegir-hi parts hardware per poder modificar els paràmetres del multiefectes.

Finalment, es farà l’anàlisi de l’estructura teòrica dels efectes, se n’analitzaran les principals característiques i els blocs necessaris per a implementar-los (de forma independent al llenguatge de programació escollit). Un cop analitzats es modelaran amb el Simulink per a poder-los utilitzar correctament.

1.4. Planificació del Treball

S’ajunta la figura del Diagrama de Gantt que es crea per a la realització completa del TFG. Aquest inclou tant el calendari de les entregues de l’avaluació continua (PACS) com les tasques que s’han de seguir pel correcte desenvolupament de del producte final.

Il·lustració 1 – Planificació del TFG sobre un diagrama de Gantt

Pel que fa a la descripció de les tasques, es descriuen les de l’apartat “Desenvolupament”:

Proves RP2: Configuració de la Raspberry Pi2 amb la imatge que ofereix MathWorks (l'empresa responsable de MATLAB i Simulink) per el desenvolupament de projectes amb Simulink sobre la Raspberry Pi.

Proves RP4: Configuració del Raspberry Pi 4 – una unitat amb més capacitat de processament que l'anterior- amb la imatge que ofereix MathWorks (l'empresa responsable de MATLAB i Simulink) per el desenvolupament de projectes amb Simulink sobre el Raspberry Pi.

Programació d'efectes: El gruix de la programació del TFG. S'han de modelitzar els efectes que es vulguin implementar tot fent un anàlisi dels models teòrics que es volen reproduir. Inclou la part d'anàlisi de l'estructura que els formen per a la posterior implementació sobre Simulink.

Proves micròfon extern: Si volem realitzar la modificació a temps real, necessitem una font sonora per captar el so exterior. En aquest cas escollim un micròfon tot i que podríem treballar amb senyal de línia.

Configuració targeta externa: La targeta de so inclosa en la Raspberry Pi és de baixa qualitat. S'opta per una targeta externa per tal de poder capturar i generar l'àudio amb més fidelitat.

Muntatge de l'entorn de treball: Instal·lació i adequació de l'espai per el correcte desenvolupament del projecte i poder, entre altres coses, tenir un monitor i altaveus per el Raspberry Pi i no haver de treballar sempre en mode escriptori remot.

Proves encoders: Per la modificació dels paràmetres dels efectes es vol utilitzar uns potenciòmetres giratoris que s'han de connectar al Raspberry Pi mitjançant la seva placa GPIO. Es volen realitzar proves prèvies per tal de tenir el circuit prototipat abans de la creació dels efectes.

Disseny capsa final: Es tracta d'una fase gairebé estètica on, un cop vist i provat el prototip final, s'intentarà buscar la forma de presentar el producte d'una forma el més comercial possible.

1.5. Planificació pressupostària

Per a l'inici del projecte es requereix una quantitat de material per a poder desenvolupar el producte final.

Quantitat	Nom Producte	Descripció	Preu unitat
1	Raspberry Pi 4B 2Gb	Per a poder desenvolupar la plataforma	52.90€
1	Kit Ventilació RPI4	Conjunt de ventiladors opcional	19.30€
5	Potenciòmetres rotatoris	Per a poder manipular els efectes	4.49€

1	Cablejat i protoboard	Cablejat per a poder realitzar les connexions sense necessitat de soldadura	9.80€
1	Llicència MATLAB personal	Com a estudiant del GETiST tenim accés a la versió estudiantil.	0€
1	Targeta so externa	Per a millorar la qualitat de l'àudio generat	5€
		TOTAL	**87**€

Taula 1 - Planificació pressupostària

1.6. Breu sumari dels productes obtinguts

Un cop finalitzat aquest TFG, el producte final és un multiefectes d'àudio basat en un RPI 4 i programat sobre Simulink. Tot i no complir les premisses inicials (capacitat d'execució dels efectes per un entorn de música en viu), és un molt bon banc de proves i un bon inici per a portar a terme noves investigacions al respecte. Els efectes modelats, per altra banda són plenament funcionals i exportables a altres models de *hardware*.

El producte és el resultat de totes les investigacions portades a terme durant el semestre que s'ha cursat durant la realització d'aquest TFG.

1.7. Breu descripció dels altres capítols de la memòria

En el capítol 2 "Estat de l'Art", es fa un breu repàs a què entenem per efecte digital d'àudio. Se n'observen les seves aplicacions comercials i també uns quants dels projectes DIY que han servit per a establir les bases per aquest projecte concret.

En el capítol 3 "Disseny del producte", es porta a terme la construcció de tots els efectes d'àudio modelats. Per a cadascun d'ells es fa un repàs teòric i se n'explica la modelització construïda durant el TFG. En el cas de l'equalitzador, una mica més complex que la resta, es fa un anàlisi de què són els filtres i es plantegen dos modelats diferents.

Aquest apartat 3, conté l'apartat 3.4 "Problemes de disseny" on s'enumeren, analitzen i es proposen solucions per als diversos problemes que han anat sorgint durant aquest procés.

Per acabar aquesta memòria, a l'apartat 4, "Conclusions i línies futures", es plantegen les conclusions a les que s'ha arribat durant el transcurs d'aquest projecte i es plantegen possibles línies futures, també en base dels objectius no acomplerts d'aquest projecte.

2. Estat de l'art

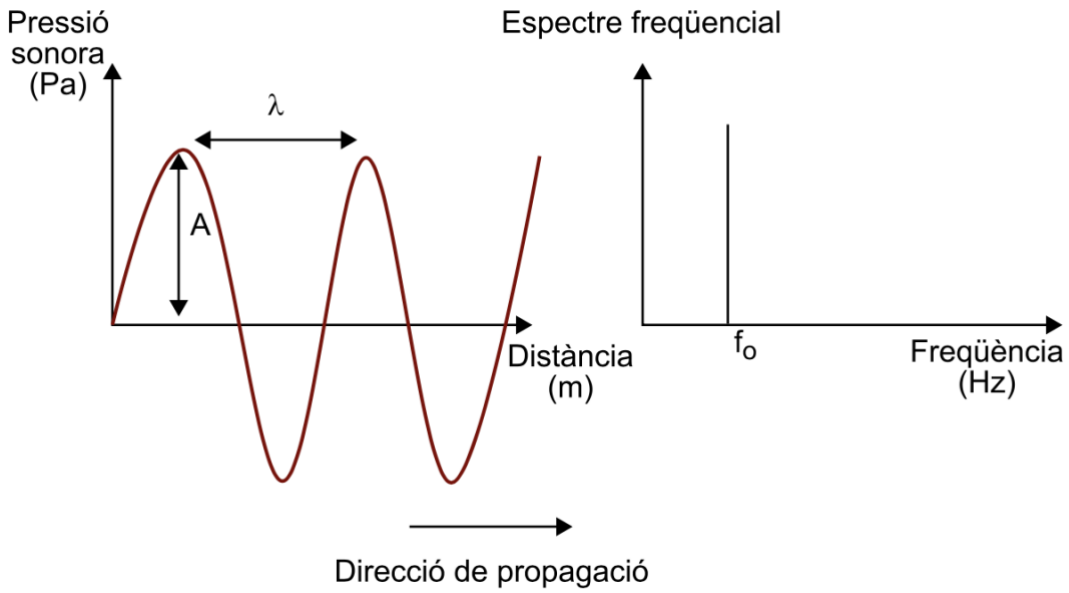
2.1. Què és un efecte d'àudio?

Segons els materials de l'assignatura de Processament d'àudio, un efecte és "qualsevol modificació que es fa sobre un senyal d'àudio que provoca un canvi en la percepció del so"[1]. Així, podem entendre que la modificació no només afecta a la durada de l'àudio sinó que també pot modificar el to o el timbre del què s'està escoltant.

Necessitem doncs saber quins són els paràmetres que ens defineixen una ona sonora per poder analitzar quines modificacions els podem fer.

Una ona sonora es defineix per diferents paràmetres:

- **L'amplitud (A):** definit com el desplaçament màxim respecte la seva posició d'equilibri;
- **El to:** definit com el contingut freqüencial de la senyal, en el cas de tos purs, parlem directament de freqüència;
- **El període:** si és un senyal cíclic, és definit com el temps entre repeticions. Matemàticament, és la inversa de la freqüència



Il·lustració 2 - Esquema d'ona sonora. Representació temporal i freqüencial. Extret de [1]

Altres paràmetres que es valoren en ones sonores són la velocitat de propagació (que variarà en funció del medi on es propagui) o bé la longitud d'ona (la distància entre punts en el mateix estat d'excitació, λ) o també la direcció de propagació.

Per acabar, podem parlar de l'ample de banda de l'ona. Sabem que les ones poden tenir freqüències molt diferents, però que només unes bandes determinades són perceptibles per la oïda humana. En el nostre cas, i per estandardització, treballem amb l'interval dels 20 als 20kHz com a zona audible.

Una vegada tenim definida l'ona sonora, necessitem processar aquesta ona i poder tenir-la en el domini digital.

El procés per a obtenir-ho és la conversió analògica-digital i, tot i que es pot tractar com un sol bloc, consta de tres fases diferenciades:

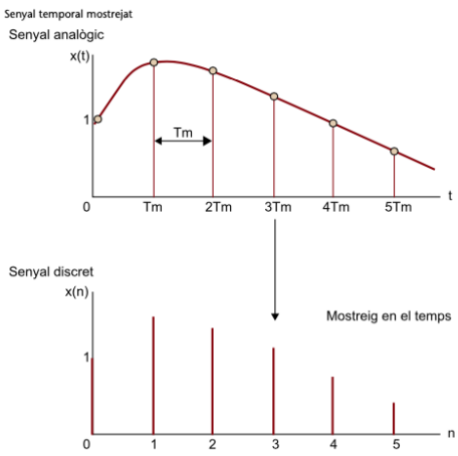
- **Mostratge:**

El primer pas és el mostratge del senyal. En aquest cas agafem els diferents valors d'amplitud del senyal de forma periòdica. Podem dir que és el moment on el senyal deixa de ser un senyal continu i tenim un senyal discret, ja que només obtenim mostres en funció de la freqüència de mostreig determinada. Aquests valors, tot i que discrets, poden prendre qualsevol valor dins el rang d'amplitud del senyal original.

L'expressió matemàtica d'aquest pas és:

$$\boldsymbol{x[n] = x(t)|_{nTm} = x(nTm)}$$

Equació 1 - Expressió per el procés de mostratge



Il·lustració 3 - Esquema representatiu del pas de discretització. Extret de [1]

On $x[n]$ és el senyal discretitzat provinent d'un senyal $x(t)$ que es mostreja amb un període de mostreig Tm que és pot determinar amb la freqüència de mostreig mitjançant la relació:

$$\boldsymbol{Tm = \frac{1}{Fm}}$$

Equació 2 - Relació entre període de mostreig i freqüència de mostreig

- **Quantificació:**

El segon pas de la cadena és el pas de la quantificació. En aquest pas, els valors discrets anteriors es quantifiquen per tenir una senyal digital quantificada en uns nivells definits en funció del nivell de quantificació i el nivell de profunditat de bits amb què dividim el marge dinàmic.

Per obtenir aquests diferents nivells, dividim el marge dinàmic en diferents nivells de quantificació i assignarem els valors discrets anteriors als valors quantificats més

propers. Existeixen diferents tipus de quantificació aplicables (uniforme, logarítmica, entre d'altres) que ens ofereixen diferents avantatges i inconvenients.

Per determinar els nivells de quantificació necessitem saber el valor màxim i mínim del senyal a quantificar. D'aquest manera, podem definir el pas de quantificació mitjançant el marge dinàmic del senyal:

$$\Delta = \frac{\boldsymbol{MD}}{\boldsymbol{2^b}}$$

Equació 3 - Pas de quantificació

On el marge dinàmic, MD, es defineix com:

$$\boldsymbol{MD = màxim\ valor\ d'amplitud\ a\ quantificar - mínim\ valor\ d'amplitud\ a\ quantificar}$$

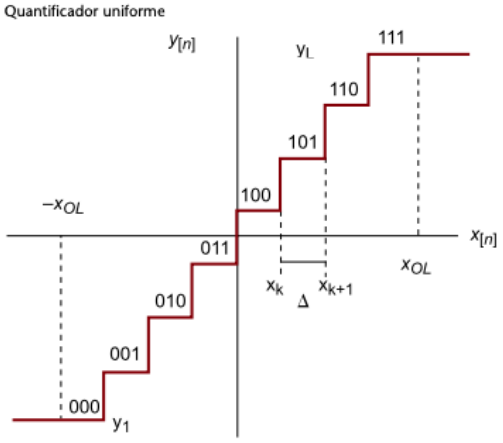
Equació 4 - Relació del marge dinàmic

I on b és el nombre de la profunditat de bits. Amb aquesta profunditat de bits també obtindrem els nivells de quantificació amb la relació:

$$\boldsymbol{N = 2^b}$$

Equació 5 - Càlcul dels nivells de quantització

En el cas d'un quantificador uniforme, una representació podria ser:

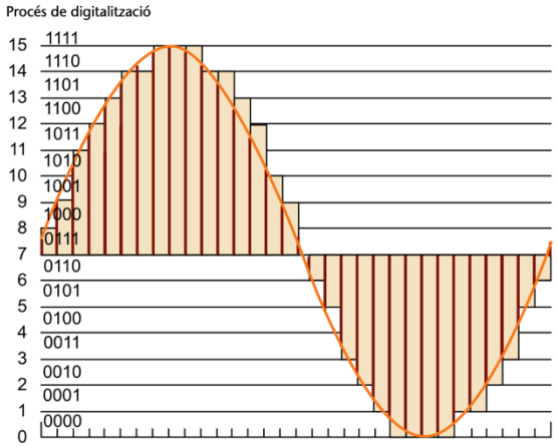


Il·lustració 4 - Esquema del pas de quantització d'un senyal discret. Extret de [1]

- **Codificació:**

Finalment, aquests valors quantificats es codifiquen al sistema binari de forma que obtenim un tren de zeros i uns. Aquest tren de zeros i uns no ens donarà mai el senyal original sinó la reconstrucció del senyal quantificat.

Una vegada realitzats aquests passos, podem afirmar que el senyal ja és un senyal digital i binari i que per tant, ens trobem en el domini digital.



Il·lustració 5 - Esquema de la codificació del senyal discret quantitzat. Extret de [1]

És imprescindible, parlant de processament d'àudio, mencionar el Teorema de mostreig de Nyquist. Aquest teorema ens diu que per poder realitzar una digitalització del senyal de qualitat, la freqüència mínima de mostreig que necessitarem és de, com a mínim, el doble de la freqüència d'àudio que té el senyal analògic original. Per tant, per poder treballar amb contingut sonor audible, necessitem treballar amb freqüències iguals o superiors als 40kHz.

Pel que fa als passos de quantització, és habitual treballar amb valors de 8, 16 o 24 bits. Per posar un exemple, els paràmetres de conversió analògic-digital d'un reproductor de CD és de 44.1KHZ com a freqüència de mostreig i 16 bits de profunditat.

En aquest TFG ens centrarem en els efectes en el domini de processat digital, tot i que òbviament, la majoria d'efectes que s'utilitzen provenen del món analògic. Arribats en aquest punt, una possible classificació d'aquests efectes és:

- **Efectes modificadors del temps:** Són els efectes que afegeixen una o varies versions retardades del senyal original. En aquesta categoria trobem els efectes de reverberació o *delays*.

- **Efectes modificadors del contingut freqüencial:** Són els efectes que ens modificaran el contingut freqüencial de la senyal d'àudio. En aquesta categoria trobem tant els equalitzadors com els efectes coneguts com *flanger* o *phaser.*

- **Efectes modificadors de la dinàmica:** Són els efectes que ens modificaran l'amplitud de la senyal d'àudio. Poden ser efectes de *tremolo* o també compressors i expansors.

Una altra possible classificació seria segons la linealitat dels efectes i la seva invariabilitat en el temps. Podem parlar de sistemes lineals o sistemes no-lineals i sistemes variables i no variables en el temps.

El principi de linealitat ens diu que si sumem dues senyals sortides del mateix filtre obtindrem el mateix resultat que si apliquem l'efecte a les dues senyals originals i les escalem.

Pel que fa a la variabilitat temporal, podem veure que un efecte és invariable en el temps quan retardar el senyal d'entrada és igual que retardar el senyal de sortida.

En cas que no es compleixin aquestes condicions, parlarem de sistemes variables en el temps o sistemes no-lineals.

També cal dir que, en moltíssimes ocasions, els efectes són una combinació de diversos efectes explicats. Per exemple, en efectes de reverberació, és molt habitual trobar paràmetres per modificar el contingut freqüencial del senyal resultant mitjançant un equalitzador, per exemple.

2.2. Usos dels efectes d'àudio en entorns de producció musical

És fàcil afirmar que tota la música que consumim ha estat processada en algun moment (fins i tot en el domini analògic) i per tant, ha pogut ser modificada mitjançant efectes.

En el nostre cas concret, entenem entorns de producció musical com poden ser un estudi de gravació o una actuació en directe. En ambdós, trobem inserits en la cadena d'àudio, diferents components que poden aplicar efectes sobre l'àudio. Tot i això, podem veure diferències entre els dos entorns. En el cas de les actuacions en directe el processat ha de ser a temps real (la gent ha de sentir el que es toca i el que s'està modificant) i per contra, els estudis de gravació poden, en la majoria d'ocasions, processar tantes vegades com vulguin i amb temps de latència més elevats.

Introduïm aquí el concepte de latència ja que és important per poder quantificar si el processament que realitzem és satisfactori per aplicar-lo a temps real o no. La **latència** és, segons el Termcat [2], el període de temps entre l'entrada i la sortida d'un senyal d'un sistema. Per tant, la diferència que tenim entre la captura i la reproducció de l'àudio un cop s'ha processat dins el sistema. Dins aquest processat, pot incloure o no l'aplicació de l'efecte dissenyat.

Així, podem veure que en les produccions en viu, l´ús d'efectes d'àudio és imprescindible si, per exemple, volem reduir les freqüències més greus de la captació d'un micròfon; o volem simular que els nostres instruments es troben en un entorn molt més reflectant del que estem (o en una altra sala en el cas de les reverberacions més complexes); o volem incrementar o reduir la dinàmica d'una senyal captada. Per contra, en entorns de postproducció, no és tant rellevant ja que podem capturar el senyal, aplicar l'efecte i reproduir-lo sense preocupar-nos de la latència del procés.

Actualment, i parlant del cas concret de les actuacions musicals, tenim molts dispositius que permeten el processament d'àudio. De fet, un cop captades les ones sonores, les taules de mescles digitals ja incorporen mòduls dedicats al processament d'àudio. Anteriorment abans de treballar amb àudio digital, les taules analògiques només

disposaven d'efectes modificadors del contingut freqüencial, de forma habitual. Ara però, qualsevol altaveu té un mòdul equalitzador o un limitador per a protegir electrònicament els seus components. A més a més, molts dels instruments que s'utilitzen ja tenen mòduls de processat per a modificar el so que generen. Per exemple, un teclat electrònic pot incorporar un mòdul de reverberació al final de la seva cadena d'àudio.

Per acabar d'acotar l'àmbit del TFG, direm que el projecte del multiefectes s'inclou en l'àmbit dels aparells que poden utilitzar guitarristes o cantants, que capten un senyal d'àudio, el digitalitzen i mitjançant certs algoritmes processen aquest àudio i el retornen ja sigui cap a un altaveu o cap a un altre aparell de la cadena d'àudio. Per a simplificar, i utilitzant l'argot de l'entorn musical, estem parlant d'una *pedalera d'efectes.*

Aquesta diferenciació no és banal, ja que com s'ha comentat, hi ha molts aparells capaços de processar l'àudio rebut. En l'àmbit temporal del TFG no és possible desenvolupar un processador d'àudio per processar el senyal d'una actuació complerta (amb varis micròfons i molta més capacitat de processament) o un sistema de processat per un sistema complert d'altaveus tot i aquest projecte podria tractar-se d'un punt de partida per a altres projectes d'aquesta envergadura.

2.3. Multiefectes comercials

Al mercat trobem moltíssimes opcions comercials encarregades de processar el senyal d'àudio. Podem distingir entre els que es centren en aplicar un sol efecte (i aquest és modificable amb potenciòmetres) o els que tenen diversos efectes programats, els multiefectes. El catàleg i les característiques són extenses però al final, tots han de tenir, com a mínim:

- **Una entrada d`àudio** per poder captar la font sonora, ja sigui un instrument o un micròfon;
- **Una sortida d'àudio** per poder entregar el senyal modificat;
- **Un convertidor** de senyal analògica a digital i un convertidor de senyal digital a analògica que pugui mostrejar i quantificar l'àudio que captem
- **Controladors** per poder, per exemple, activar o no l'efecte i poder modificar-ne els paràmetres.

Il·lustració 6 - Efecte digital de delay de la marca Boss. Extret de [48]

Per altra banda, trobem els aparells que permeten l'aplicació de més d'un efecte a l'àudio. Com abans, el catàleg comercial és inabastable però, com abans, les parts són similars. Aquesta vegada i com a principal diferència dels anteriors aparells, al necessitar més capacitat de processament, acostumem a trobar xips específics per al processament de senyal, un DSP o un FPGA. També, al ser una mica més complexes, acostumen també a disposar de pantalles LCD o LED per poder saber amb quin programa o efecte estem treballant.

Il·lustració 7 - Multiefectes comercial de la marca TC Electronics per guitarra. Extret de [49]

Tot i això, no hi ha gaires projectes comercials que permetin la reprogramació dels efectes preestablerts més enllà dels paràmetres modificables o, en cas de ser diversos efectes, l'ordre de la mateixa cadena d'efectes. Només hem trobat un producte similar, el Line 6 ToneCore DSP Developer Kit (TCDDK) [3] de line6 va arribar al mercat i, segons la pàgina web del fabricant, està descatalogat.

Il·lustració 8 - Pedal d'efecte programable de la marca Line6. Extret de [50]

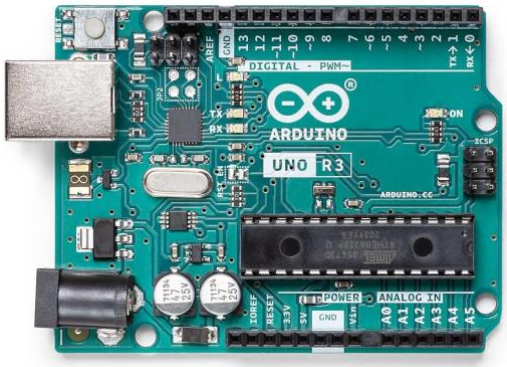
2.4. Projectes DIY

Per acabar aquest Estat de l'art, es creu necessari fer un repàs a diferents projectes desenvolupats fora de les marques comercials.

En cas de processament analògic de senyal, existeixen molts tutorials, *kits* prefabricats i fins i tot instruccions per modificar efectes comercials pe poder fer-los modificacions personalitzades. Al no tractar amb àudio digital, podem considerar aquests projectes plenament de filtrat analògic de la senyal elèctrica i per tant, queden fora de l'àmbit d'aquest TFG.

En el cas dels efectes o multiefectes digitals trobem persones que han experimentat sobre plaques de tipus Arduino, Raspberry Pi o plaques FPGA com les Xilinix.

La plataforma Arduino és una plataforma que agrupa solucions hardware (plaques de circuit imprès amb un microcontolador) i software (entorns de programació) per acostar l'accés a les noves tecnologies a interactuar amb el món físic[4]. La programació i depuració del codi del microcontrolador s'ha de realitzar des d'un altre ordinador.

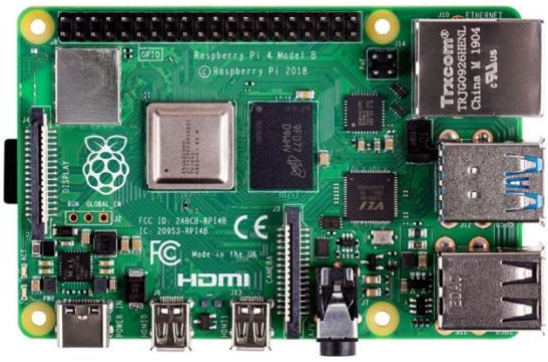


Il·lustració 9 - Placa Arduino Uno R3. Extret de [4]

Trobem una gran comunitat que desenvolupa projectes sobre Arduino, però en el cas de l'àudio acostumem a trobar limitacions en quant a hardware i és fàcil que necessitin algun complement (una *shield* que pugui gestionar un convertidor analògic/digital) específic per al processament d'àudio. Tot i això, la facilitat de connectar sensors i potenciòmetres als pins de l'Arduino i el reduït cost de les plaques fa que sigui una opció a considerar per realitzar projectes similars.

Podem citar, per exemple, el projecte echoTrek[5]. Un multiefectes digital que permet variar entre diferents estils d'efectes de retard sobre un senyal d'entrada. És important remarcar que el projecte genera una latència important sobre el senyal d'entrada.

Una altra plataforma on trobem molts projectes DIY és el RaspberryPi[6]. El RaspberryPi és un ordinador monoplaca basat en processadors ARM. Aquí trobem la primera diferència amb les plaques d'Arduino ja que es tracta d'un ordinador funcional, no d'un microcontrolador. Al tractar-se d'una plataforma ARM, el sistema operatiu ha estat històricament basat en Linux.



Il·lustració 10 - Placa de la RasberryPi4. Extret de [6]

Existeixen molts projectes per a aquesta plataforma; la majoria, però, acostumen a ser adaptacions de software existents per a distribucions de Linux que, instal·lades sobre un Raspberry Pi, permeten l'ús d'efectes d'àudio. És el cas d'aquest anomenat Pi-FX: A Raspberry Pi-Based Pedal Board[7]. També trobem aquest projecte anomenat Raspberry Pi PC: DIY Guitar Pedal using NeuralPi que utilitza xarxes neuronals per a l'emulació d'efectes coneguts[8] com poden ser una distorsió o una reverberació existents com el Delay [48] que hem anomenat anteriorment.

Una altra vegada però, no podem personalitzar els efectes tal i com voldríem. A més a més, en segons quina generació de Rasberry Pi necessitem també una targeta de so dedicada per millorar la qualitat del processament d'àudio.

Per acabar fem un repàs als projectes amb FPGA. L'arquitectura FPGA (Field Programable Gate Array[9]) és una arquitectura on el dispositiu també és programable de forma que podem programar tant el codi que executarà com la configuració dels diversos blocs del xip. Es tracta d'una arquitectura molt estesa però molt menys a l'abast que les dues comentades anteriorment. I per aquest motiu, només trobem algun projecte i normalment relacionat amb centres d'estudis o investigació.

Il·lustració 11 - Placa FPGA Xilinx. Extret de [9]

En tot cas, existeixen projectes de multiefectes com aquest treball final de la EE de Tel-Aviv[10] o aquest projecte de multiefectes del MIT[11] programat sobre Zedboard[12], programats mitjançant VHDL o Verilog que aprofiten la potència de la placa anomenada anteriorment i el seu xip dedicat a àudio per poder realitzar el producte dins una sola placa. En aquest cas, disposem de xip dedicat a la conversió d'àudio en funció de la placa i del seu preu, però els temps de latència acostumen a ser molt bons.

Per acabar aquest repàs, trobem un producte entre el DIY i els productes comercials. Són els productes de la marca ElectroSmash[13], que ven els kits de construcció pels seus pedals (també els ven sencers). Disposen d'un producte anomenat *Time Manipulator - Arduino Delay/Echo/Reverb*[14] que ells defineixen com a reprogramable ja que es basa en el xip de l'Arduino (ATMEGA328P-PU). Tot i això, la programació ha de ser dins l'IDE de l'Arduino, amb els avantatges i inconvenients que això significa.

Per tant, una vegada anomenats alguns dels antecedents disponibles, podem parlar que el nostre producte ha de diferenciar-se de la resta en dos grans aspectes: la

personalització de l'efecte aplicat (tipus d'efecte, ordre en cas de voler aplicar diversos efectes o els paràmetres a modificar) i el cost, que ha de ser reduït.

3. Disseny del producte

3.1. Raspberry Pi

3.1.1. Especificacions

El Raspberry Pi (RPi, a partir d'ara) és un ordinador de la categoria dels ordinadors de placa única (SBC, *single-board computer*, en les seves sigles en anglès) desenvolupat al Regne Unit per la fundació Raspberry Pi. El primer model va ser llançat al mercat durant l'any 2012 i n'han aparegut diferents iteracions (altres SBC, amb perifèrics incorporats, més potència, etc)[6].

Il·lustració 12 - Diverses versions de Raspberry Pi

Tot i que en un primer moment estava enfocat a democratitzar l'ensenyament de les ciències de la computació, el seu cost va fer que també fos utilitzat per a dissenys d'aficionats i usos generals. Per aquest motiu, existeixen molts recursos i tutorials per a projectes emmarcats en el moviment DIY.

En tots els models existents, però, l'arquitectura es basa en processadors ARM i el seu sistema operatiu *de facto* han estat adaptacions de Debian distribuïdes des de la mateixa Fundació Raspberry Pi sota el nom de Raspberry Pi OS (anteriorment conegut com a Raspbian). L'emmagatzematge es realitza sobre targetes SD o micro-SD en funció de la iteració amb què treballem. A nivell de connectivitat, les plaques han evolucionat de forma que en el model actual disposen de fins a dos connectors micro-HDMI, diversos ports USB i connectivitat Wifi. A més a més, disposa de pins GPIO (*General Purpose Input Output)* per la connectivitat de sensors o actuadors externs.

El primer inconvenient detectat és que el RPi 4 no disposa ni de micròfon integrat ni d'entrada de línia d'àudio. Per aquest motiu, en cas de voler capturar àudio amb aquest ordinador, haurem de buscar una opció que ens permeti fer-ho ja sigui via una targeta externa USB o algun altre protocol.

3.1.2. Justificació de la tria

En el cas d'aquest TFG el model escollit és un model del RPi 4B+ (apareguda al mercat el juny del 2019) que disposa de 2Gb de RAM i un processador quad-core Cortex-A72 de 64 bits. S'ha de dir que actualment costa molt trobar unitats de model 4B+ de més de 2Gb degut a la crisi dels semiconductors que viu la indústria. Tot i que inicialment la idea

era treballar amb un model de 4 o 8 Gb ha estat impossible trobar-ne alguna unitat a un preu raonable.

Il·lustració 13 - Placa de la Raspberry Pi 4 utilitzada durant el projecte. Extret de [4]

Aquest model ens proporciona un total de 4 ports USB (una parella sota l'especificació 2.0 i dos ports més seguint l'especificació 3.0), connectivitat wifi de doble banda (2.4 i 5Ghz) i Bluetooth, alimentació sobre USB-C i dues sortides micro-HDMI (que suporten resolucions de fins a 4K a 60fps) [15].

Paral·lelament, ens ofereix un *header* de GPIO de 40 pins per a la connectivitat de perifèrics que no utilitzin USB.

Per acabar, el sistema operatiu escollit en el moment inicial és la distribució completa i vigent en aquell moment, la versió 11, *Bullseye* (amb el *kernel* en versió 5.10.63). Com es veurà més endavant amb més detall, el producte final, però, no s'implementa sobre aquesta distribució concreta.

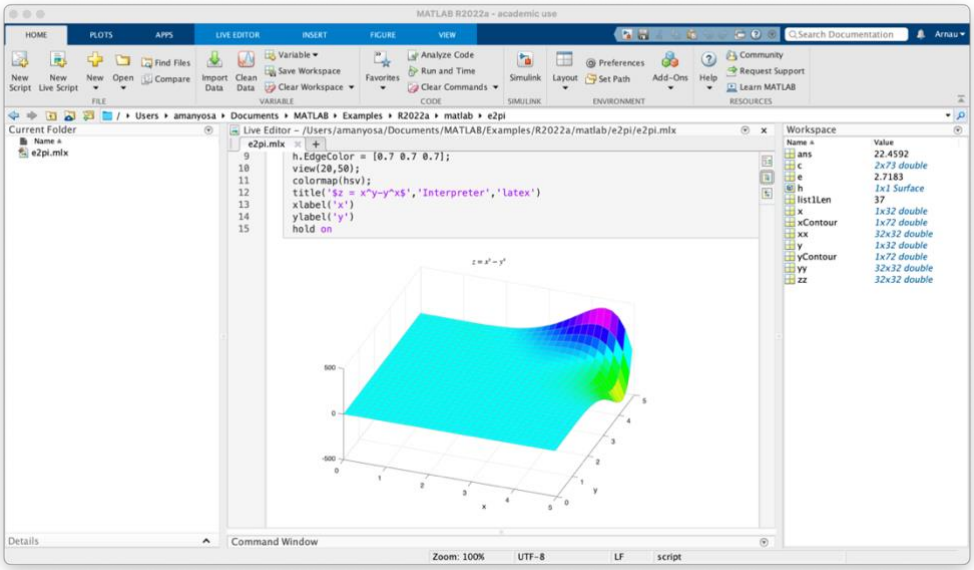
En el moment d'escriure aquesta memòria, la última versió de *Bullseye* es va alliberar el 4 d'abril del 2022[16] i, entre d'altres, canvia el *kernel* de Linux a la versió 5.15.30. Com es veurà més endavant, aquests canvis són crucials en el desenvolupament del producte final.

3.2. MATLAB i Simulink

3.2.1. Especificacions

MATLAB i Simulink són paquets de programari privatiu desenvolupats per l'empresa MathWorks a partir de l'any 1984. A l'hora d'escriure aquesta memòria la última versió del software és la R2022a, publicada al març del 2022. El model desenvolupat per a l'entrega final també és compilat amb aquesta versió – tot i que es va començar a treballar sobre la versió 2021. És compatible amb els sistemes operatius Windows, Mac OSX i Linux[17].

A la il·lustració 14 podem veure una captura de pantalla d'un projecte d'exemple del MATLAB.

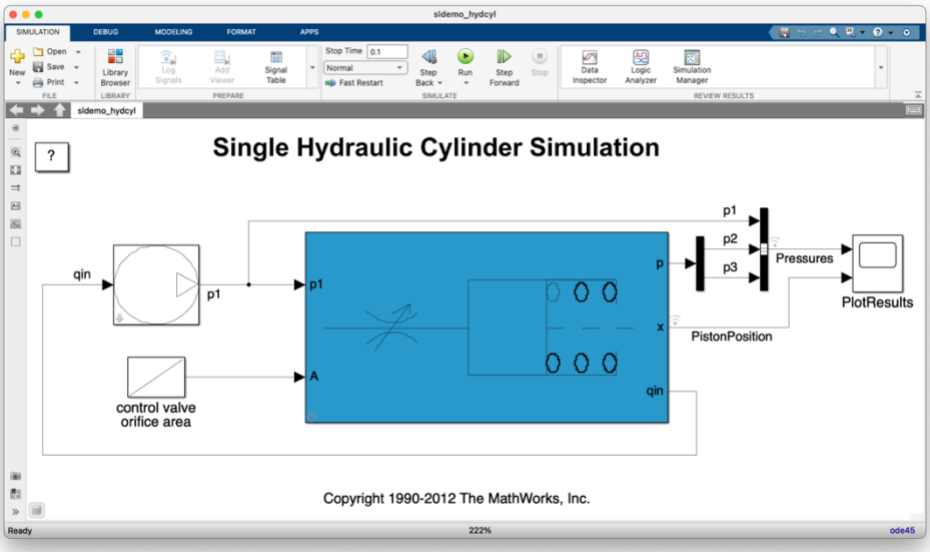


Il·lustració 14 - Captura de pantalla del MATLAB amb un projecte d'exemple

L'entorn del MATLAB és altament conegut en entorns d'enginyeria ja que es tracta d'un sistema de computació numèrica amb un llenguatge de programació propi (el llenguatge M) i un entorn de programació integrat (IDE, *Integrated Development Enviroment*, en les seves sigles en anglès) amb capacitat per manipular dades, implementar algorismes, comunicació amb maquinaris propis o capacitat per interactuar amb altres llenguatges de programació. És àmpliament utilitzat en multitud d'entorns d'enginyeries i és un dels programaris més utilitats al Grau d'Enginyeria de Tecnologies i Serveis de Telecomunicació de la UOC (a partir d'ara GETiST).

El programari disposa de complements (anomenats *add-on* o *Toolboxes*) de tot tipus (desenvolupats tant per l'empresa com per la comunitat d'usuaris) que van des de solucions dedicades a treballar amb VHDL, eines dedicades al processament digital de senyal o al modelat de vehicles.

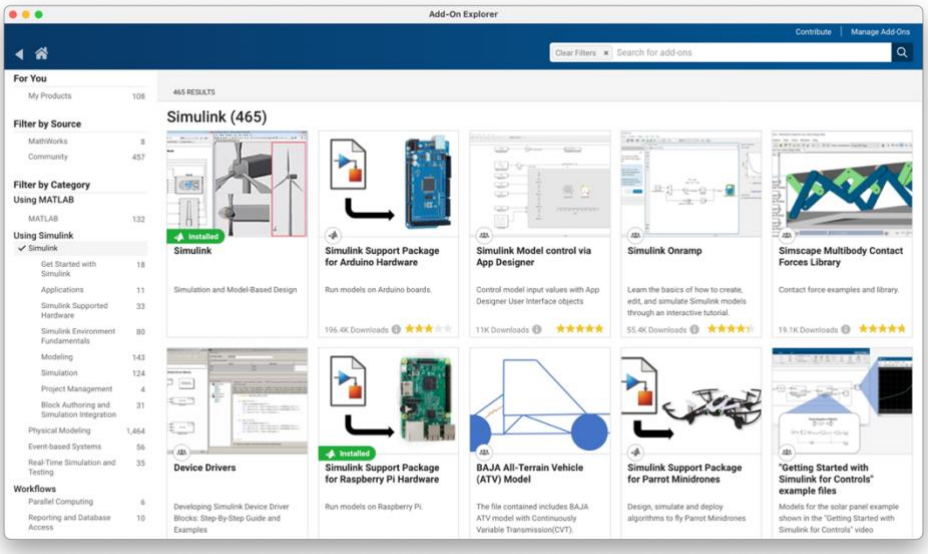
El Simulink, per altra banda, és un d'aquests *add-on* del programa principal que ens permet, mitjançant una interfície visual basada en diagrames de blocs, programar, modelar i analitzar sistemes dinàmics (tant analògics com discrets)[18].



Il·lustració 15 - Captura de pantalla del Simulink amb un projecte d'exemple

Tot i ser un *add-on* del programa principal es diferencia dels altres complements per oferir un entorn visual per a la seva programació. En aquest sentit, si l'usuari ho desitja, es pot dedicar només a unir blocs sense programar ni una sola línia de codi.

Per la seva banda, el Simulink també té complements dedicats a desenvolupaments específics. En la majoria de casos són equivalents als complements del MATLAB, però també poden ser complements específics del Simulink com ara paquets de nous blocs pels models que es realitzen.



Il·lustració 16 - Captura de pantalla del catàleg dels complements per al Simulink

3.2.2. Justificació de la tria

Com ja s'ha comentat a l'apartat anterior, tant el MATLAB com el Simulink són àmpliament utilitzats en diferents assignatures durant el GETiST. En el cas concret del Simulink, és a l'assignatura de Processament d'Àudio on la majoria d'exercicis pràctics que requereixen realitzar tractaments de senyal es realitzen amb aquest programari.

A més a més, per tal de poder treballar amb plataformes de hardware extern, la mateixa empresa MathWorks ha desenvolupat més complements per al Simulink per generar codi tant pel RPi com per la plataforma Arduino. En el nostre cas, per treballar amb el RPi, necessitem imprescindiblement el complement "Simulink Suport Package for Raspberry Pi Hardware"[19].

Tot i tractar-se d'un software privatiu, fet que ens allunya de l'objectiu de fer la plataforma a més baix cost possible, és fàcil obtenir una llicència d'estudiant per poder treballar de forma gratuïta amb tots els components necessaris o poder treballar amb la llicència de prova per realitzar un projecte prou concret.

3.2.3. Plataforma final

Així doncs, la plataforma per desenvolupar el multiefectes es concreta en una RPi 4 i el programari utilitzat pel seu disseny serà el programari Simulink (en la versió 2021 durant l'inici de la programació i posteriorment actualitzat al 2022). El model de Simulink es desenvolupa sobre un ordinador que executa Mac OSX i posteriorment es fa el *deploy* sobre el RPi.

Pel que fa a l'entrada d`àudio, es comença treballant amb una targeta de so amb una entrada monofònica i una sortida estereofònica connectada per USB.

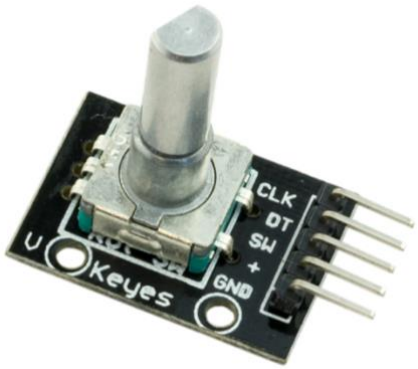
El procés d'instal·lació del Simulink és senzill (i la majoria de problemes que els usuaris poden trobar-se es resolen amb els tutorials de la mateixa MathWorks). El mateix és aplicable a la instal·lació del complement *"Simulink Suport Package for Raspberry Pi Hardware"* que ens instal·la les llibreries perquè Simulink es pugui comunicar amb el nostre hardware. Tot i això, és en aquest pas que s'observen els primers problemes de compatibilitat entre el complement i el sistema operatiu del RPi 4.

Sense entrar en detalls que ja es desenvolupen a la secció 3.4.1 "Compatibilitat del Sistema Operatiu" d'aquest mateix TFG, la solució final passa per no utilitzar la última versió del Raspberry Pi OS, sinó una versió anterior on podem instal·lar les llibreries necessàries per a la connectivitat entre l'ordinador de desenvolupament i el RPi 4.

Acabem treballant, doncs sobre la versió 10 de Raspberry Pi OS anomenada "Buster" (amb *kernel* 5.10.63) que s'executa sobre un RPi 4 de 2Gbs de memòria RAM.

Per acabar, els paràmetres del efectes es controlaran amb uns potenciòmetres digitals del model EC11[20]. Aquest model ofereix tres sortides (A,B i C, que ofereixen sentit horari, sentit antihorari i un botó central); es poden alimentar o bé amb 3 o 5 volts i per

tant, poden funcionar amb els ports GPIO del RPi 4. També disposen d'un port de connexió a terra (GND), tal i com es pot veure a la il·lustració 17.

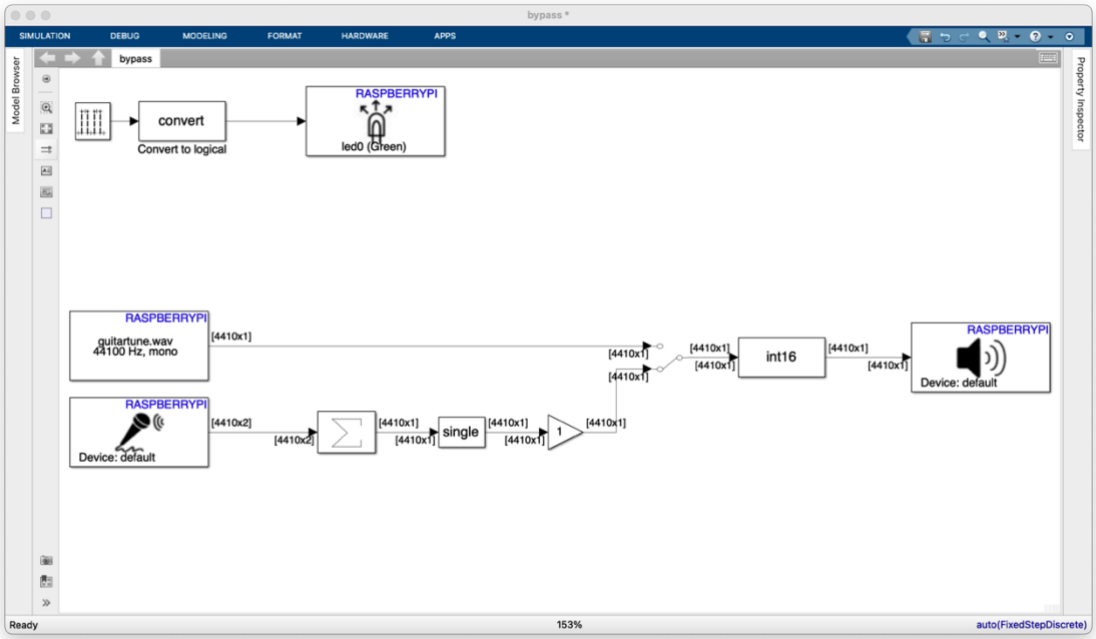


Il·lustració 17 - Potenciòmetre model EC11 utilitzat durant el projecte [20]

3.3. Els efectes

3.3.1. Bypass i control de latència

El primer element que es modela és la cadena d'àudio que utilitzarà el sistema. És el disseny més senzill i es concreta en molts pocs blocs. A la part esquerra del bloc trobem la part de la captació (que inclou capacitació, mostreig i quantificació del senyal analògic) i a la part dreta, trobem la part de la reproducció (que inclou la descodificació i reproducció del senyal). En aquest bypass, l'àudio no és tractat de cap manera, només és captat (per tant, digitalitzat) i reproduït. El model més senzill és el que es mostra a la il·lustració 18.



Il·lustració 18 - Captura de pantalla corresponent al model de Bypass

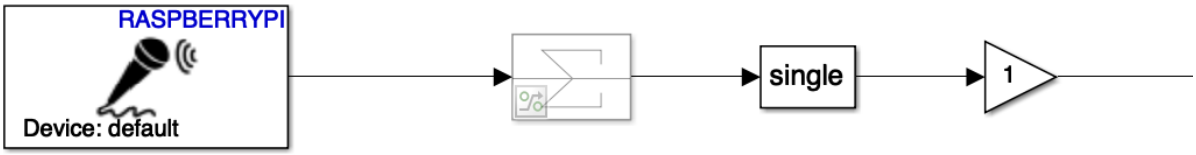
En aquest model ja podem observar certs blocs i alguns paràmetres que ens acompanyaran fins al desenvolupament final. Per començar, a la part superior trobem una seqüència intermitent del led de control del RPi que s'hereda d'un dels tutorial que ofereix MathWorks[21].

Aquesta part del led es modela per poder comprovar directament al hardware si el model s'està executant correctament. També tenim la possibilitat d'utilitzar la línia de comandes del mateix MATLAB (amb la sentència *isModelRunning*), però aquest bloc ens ofereix un retorn visual utilitzant el led situat al mateix hardware.

A la part central del model, trobem un bloc de reproducció d'un arxiu en local al RPi que ens condueix a un selector on podem escollir si escoltem l'arxiu o la captació que fa l'entrada d'àudio. Aquest selector s'implementa al principi del desenvolupament ja que existien problemes amb la captació i la targeta d'àudio del RPi. Aquests problemes s'analitzen i se'n presenten les solucions a l'apartat 3.4.2 "Captació i reproducció de l'àudio". Amb la inclusió d'aquest *switch* és possible comprovar possibles errors en la reproducció de la cadena (errors en la freqüència de mostreig o errors en el nombre de

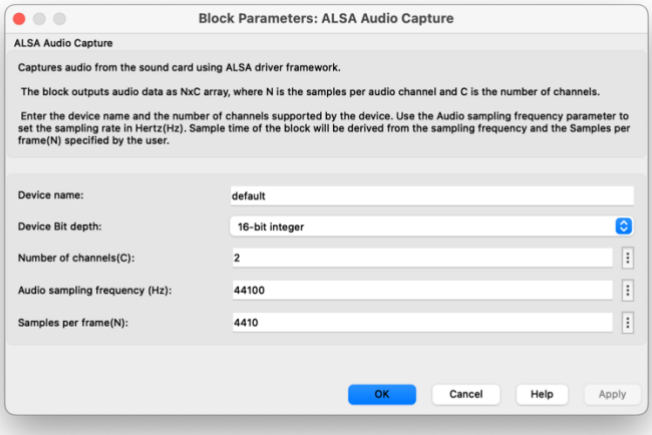
canals, per exemple) o bé en el dispositiu de captació (manca de configuració o errors en la mateixa).

A la il·lustració 19, es pot veure el model de captació que s'utilitzarà posteriorment en tots els models:



Il·lustració 19 - Diagrama de blocs del sistema de captació del projecte

El primer bloc de l'esquerra és la captació del RPi4, que treballa a una freqüència de mostreig de 44.1kHz i 16 bits de profunditat. La seva configuració es mostra a la il·lustració 20.



Il·lustració 20 - Configuració del bloc de captació d'àudio

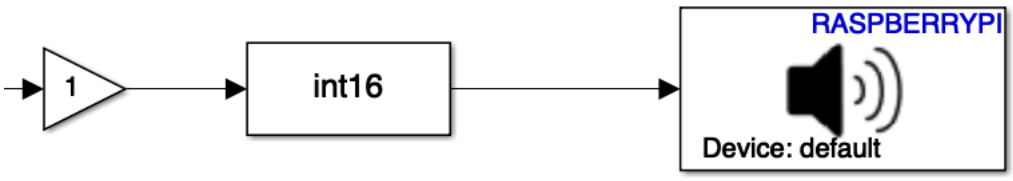
Ens ofereix un total de 4410 *mostres per frame* i, com que es traca d'un dispositiu estereofònic, dos canals. A la figura 20, per exemple, es pot observar que la magnitud dels *frames* i el nombre de canals es poden visualitzar sobre els models del Simulink com a part de les opcions de depuració.

Aquest senyal circula per un bloc sumador que converteix els dos canals en un de sol. L'origen d'aquest sumador és la necessitat de fer la comparativa entre l'àudio captat i l'arxiu reproduït en local, que ja s'ha comentat en aquest mateix apartat. Es manté per poder-se activar o no en funció de l'efecte que modelem posteriorment, tot i que en la majoria d'ocasions en trobarem desactivat, és a dir, comentat.

Finalment, un convertidor de tipus de 16 bits a tipus *single* i un guany per aplicar una normalització a l'àudio d'entrada.

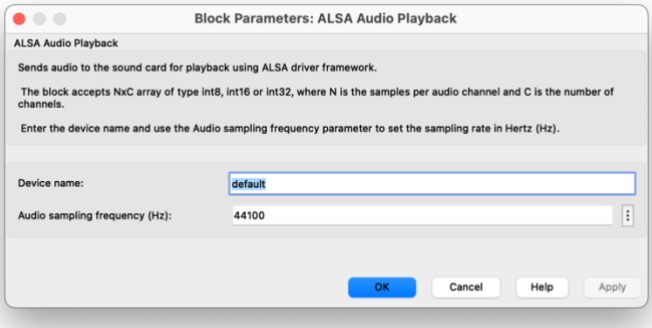
Pel que fa a la cadena de sortida bàsica mostrada a la il·lustració 21, hi trobem menys blocs. Concretament un guany normalitzador a 1 per no saturar la sortida i un

convertidor del senyal a 16 bits que requereix la sortida de la targeta d'àudio del RPi 4. Aquest model, que s'utilitza durant tots els efectes, és el que es mostra a la il·lustració 21:



Il·lustració 21 - Diagrama de blocs del sistema de reproducció del projecte

La targeta de sortida es configura, òbviament, a la mateixa freqüència de mostreig que l'entrada. Aquesta configuració és la que ens mostra la il·lustració 22:

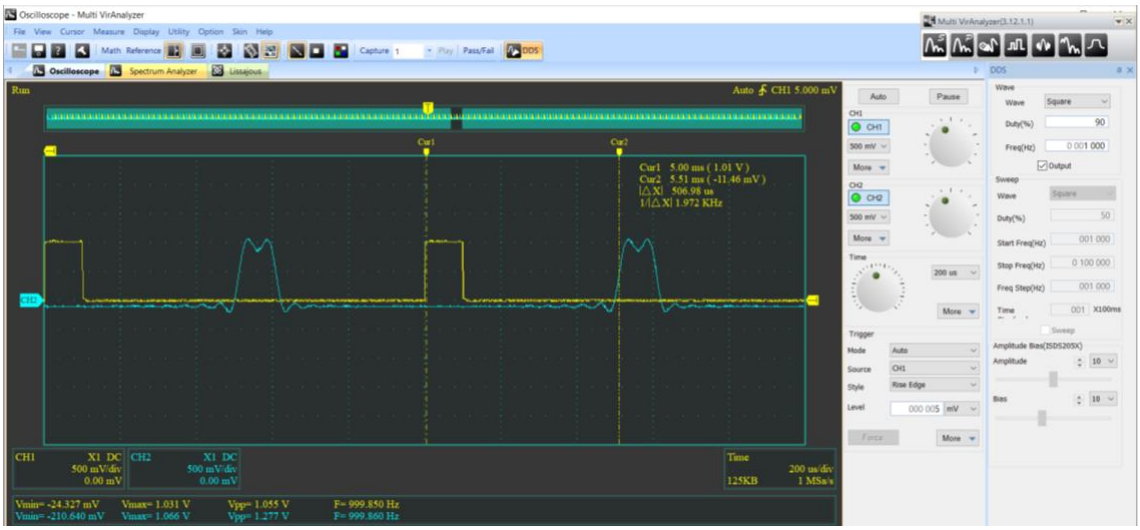


Il·lustració 22 - Configuració del bloc de reproducció d'àudio

Una vegada es disposa de la cadena d'àudio funcional, s'ha de realitzar un càlcul de la latència [23] que ofereix el sistema per saber si és apte per a aplicacions de treball en temps real. Durant el procés es va intentar generar un sistema que fos autocontingut, és a dir, que s'executés dins el RPi 4 i mostrés els resultats a l'ordinador de desenvolupament.

Malauradament no fou possible i finalment el càlcul de la latència es va fer amb el Kit de Pràctiques de les assignatures de Teoria de Circuits i Circuits Electrònics del GETiST. Amb el model anterior plantejat realitzem un muntatge com el que es mostra a la il·lustració 24, en el que utilitzem el generador de funcions del mateix programari.

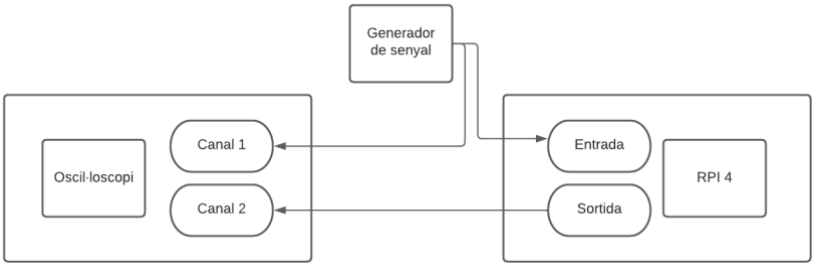
La captura obtinguda és la que es mostra a la il·lustració 23, on tenim el canal 1, visualitzat en groc, mesurant l'entrada i el canal 2, visualitzat en blau, mesurant la sortida del sistema:



Il·lustració 23 - Captura de pantalla de la mesura de latència amb l'oscil·loscopi

En aquesta mesura de la il·lustració 23, es pot observar una diferència temporal de 506 us. El problema utilitzant aquest sistema és que no tenim manera de saber si aquest pols que considerem inicial és el pols equivalent on posem el primer marcador un cop passat el sistema. Les proves auditives que fem ens ofereixen un resultat molt més elevat. S'observa que el resultat no és coherent i, per tant, hem de mesurar d'alguna altra manera.

Per aquest motiu passem a utilitzar l'eina *Data Recorder* del mateix programari que ens permet enregistrar el que passa als dos canals d'anàlisi. D'aquesta manera podrem visualitzar en quin instant comencem a enregistrar dades al segon canal i comparar-ho amb les registrades al primer canal. En aquest cas, però, necessitem utilitzar un senyal extern que ens generarà l'ordinador de desenvolupament i que podem *disparar* quan ho necessitem. El diagrama de blocs que realitzem és el mostrat a la il·lustració 24:



Il·lustració 24 - Diagrama de connexions per les mesures de latència

Una vegada fet el muntatge que correspon al diagrama de blocs de la figura 25 obtenim aquesta captura. (Recordem el canal 1 –entrada– en groc i el canal 2 –sortida– en blau):

Il·lustració 25 - Captura de pantalla de la mesura de latència amb el DataRecorder

Aquí sí que gràcies als cursors integrats veiem que tenim una latència de 502,41 mil·lisegons, que ja correspon més a la realitat que escoltem.

A mode de resum i sense entrar en detalls, l'oïda humana pot percebre un so diferenciat si es produeix més enllà dels 13 o 14 ms. Abans d'aquest període, el fenomen de l'emmascarament produït per les nostres pròpies oïdes ens ajuda a percebre-ho com un sol so[22].

Amb aquests valors de latència queda clar que no podem utilitzar el sistema per treballar en actuacions en viu ja que notarem auditivament diferències temporals entre el senyal capturat per el micròfon d'entrada i el senyal que obtenim a la sortida[23].

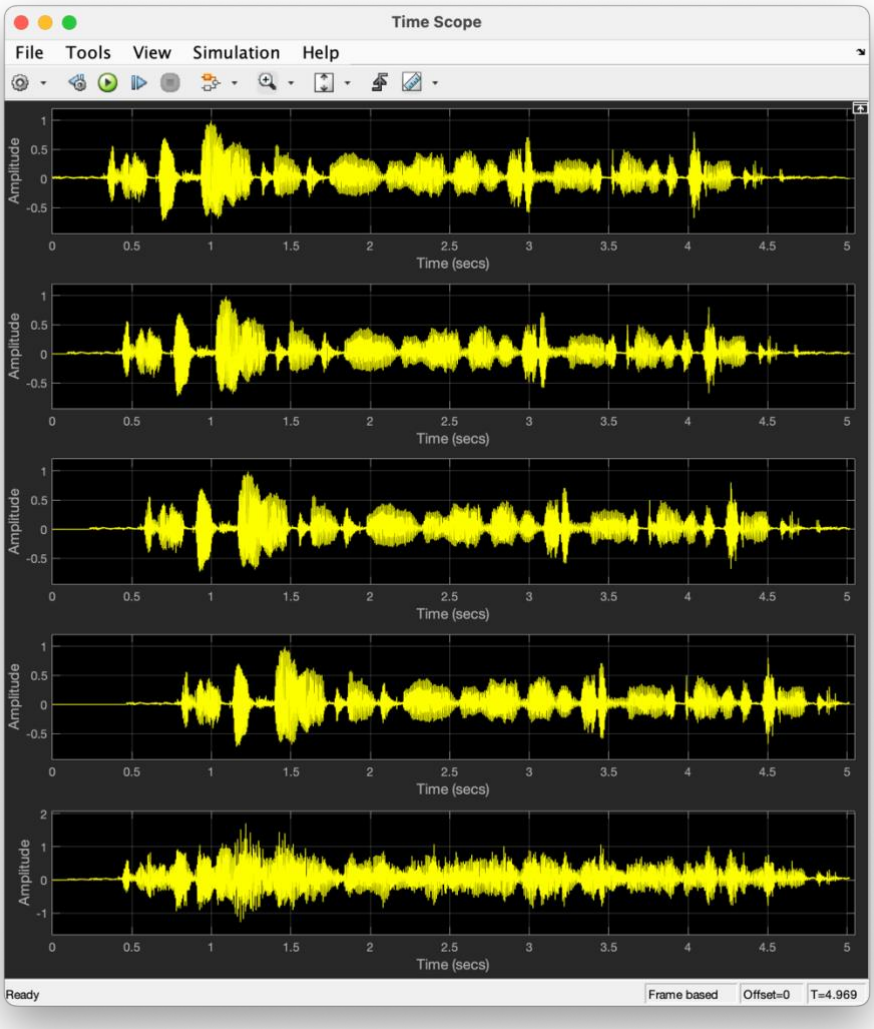
Les investigacions i possibles solucions respecte a la latència es tracten a l'apartat 3.4.3 "Temps real i latència" d'aquest mateix TFG.

3.3.2. Retardador (Delay)

3.3.2.1. Fonaments teòrics

Un efecte retardador (*delay* en el seu nom en anglès) és segurament l'efecte d'àudio més simple que podem trobar. Una vegada tenim el senyal digital, el desplacem temporalment un numero determinat de mostres (i per tant, de temps) de forma que el sentim més tard en el temps.

Amb aquest senyal retardat, si el sumem al senyal inicial, obtenim un senyal final on podem veure i escoltar el senyal original i el senyal retardat.



Il·lustració 26 - Captura del visualitzador de forma d'ona del Simulink amb diversos retards

A la il·lustració 26 podem veure una captura d'un *Time Scope* on es pot veure el mateix senyal original (primera fila), el senyal retardat dues mostres (segona fila), un retard de 5 mostres (tercera fila), un retard de 10 mostres (quarta fila) i a la part inferior, la suma de tots els senyals.

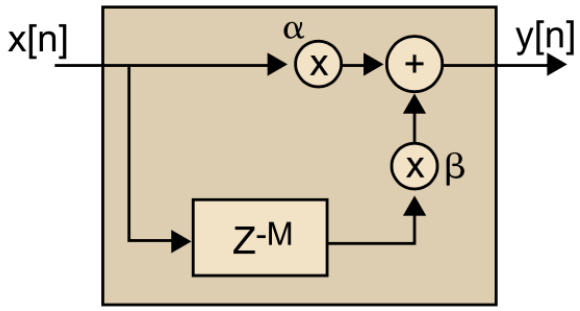
També podem veure que en cas de no escoltar el senyal original i només reproduir el desplaçat, la durada de l'àudio serà igual que la del l'original. En el cas d'escoltar la suma de l'original i el retard sí que en variarem la durada.

L'expressió algebraica del retard és:

$$\boldsymbol{y[n] = x[n] + x[n - m]}$$

Equació 6 - Expressió algebraica per un efecte de retard bàsic

On m és el numero de mostres que desplacem respecte el senyal original. I un possible diagrama de blocs podria ser el mostrat a la il·lustració 27, on veiem unes constants α i β que corresponen al guany que aplicaríem al retard original i al senyal retardat.



Il·lustració 27 - Diagrama de blocs per un sistema retardador. Extret de [1]

En funció del temps de retard que tinguem entre el senyal original i el senyal retardat i també del numero de senyals retardats que tinguem podem parlar de diferents tipus d' efectes basats en retards. Per citar-ne uns quants exemples:

- **Delay doubling**: es caracteritza per tenir una sola repetició de menys de 10ms de diferència amb el primer senyal.

- **Delay slapback**: es caracteritza per treballar amb temps entre 25 i 50 mil·lisegons de diferència.

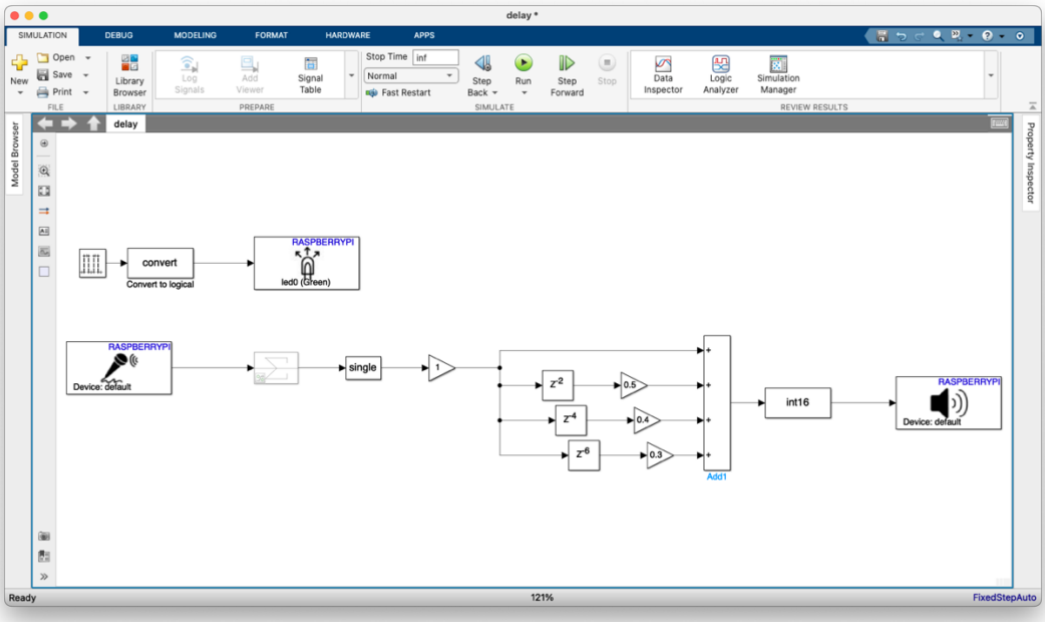
- **Eco:** es caracteritza per tenir una o vàries repeticions del senyal original atenuades amb un temps de retard de més de 50 mil·lisegons.

- **Reverberació:** amb temps de retard de menys de 50 mil·lisegons tenim infinites còpies del senyal original. Aquest tipus d'efecte s'analitza a l'apartat 3.3.3 d'aquesta memòria.

Per acabar, podem afirmar que el retard és un sistema lineal i invariant en el temps.

3.3.2.2. Disseny de blocs de Simulink

El nostre model generat correspon al que es mostra a la il·lustració 28, on trobem modelats similars i comentats anteriorment (captació i reproducció) i una secció central amb un sumador de 4 entrades on sumem el senyal original, un senyal retardat 2 mostres, un senyal retardat 4 mostres i un senyal retardat 6 mostres. Després de cada mòdul de retard s'aplica un guany específic per simular que el senyal retardat perd potència respecte el senyal original.



Il·lustració 28 - Captura de pantalla corresponent al model de Delay

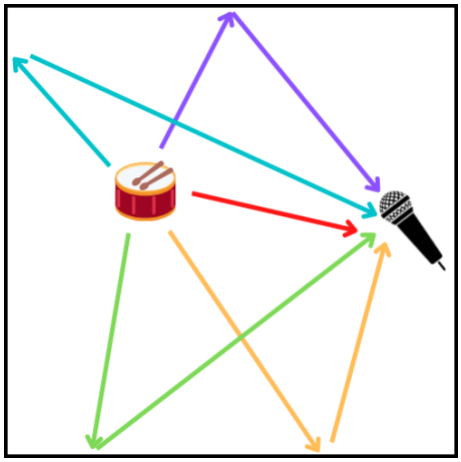
El senyal que es genera al sumatori es dirigeix cap al bloc de reproducció conegut que ja s'ha comentat a l'apartat anterior.

3.3.3. Reverberació

3.3.3.1. Fonaments teòrics

La reverberació és, segons el DIEC, "una reflexió del so a les parets d'un local tancat en virtut de la qual, un cop sentit el so que s'ha propagat directament des del lloc on s'ha produït, hom encara el continua sentint a causa de les ones que es van reflectint successivament a les parets"[24].

També podem definir-la com el conjunt d'ones sonores que no arriben directament pel camí més curt que uneix l'emissor i el receptor i per tant, arriben amb un cert retard i una certa atenuació. Sabem també, que per percebre aquestes diferents reflexions, han d'arribar més tard de 40 mil·lisegons respecte la primera ona.



Il·lustració 29 - Esquema d'algunes reflexions es un espai tancat. Elaboració pròpia.

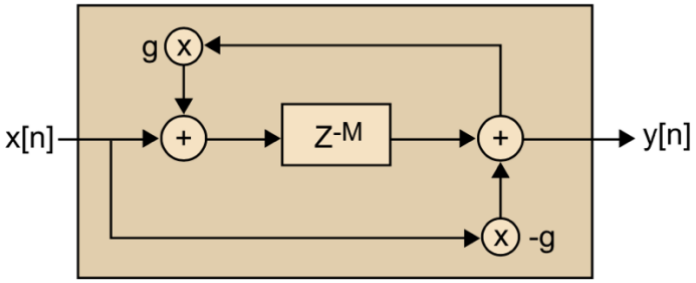
Podem doncs veure que aquest conjunt de reflexions són les que ens donen la sensació espacial de la percepció sonora. Qualsevol espai on es produeixi un so ofereix una reverberació determinada, amb uns paràmetres concrets en funció de l'espai, els materials on rebotin les ones i la climatologia de l'espai.

Tot i que actualment la majoria de reverberacions digitals parteixen de simulacions espacials, és directament possible expressar una reverberació simple seguint l'expressió matemàtica que expressa una reverberació molt simple de la forma:

$$\boldsymbol{y[n] = -gx[n] + x[n-m] + gy[n-m]}$$

Equació 7 - Expressió algebraica per un efecte de reverberació bàsica

Un possible diagrama de blocs per aquest model és el que es mostra a la figura 30:

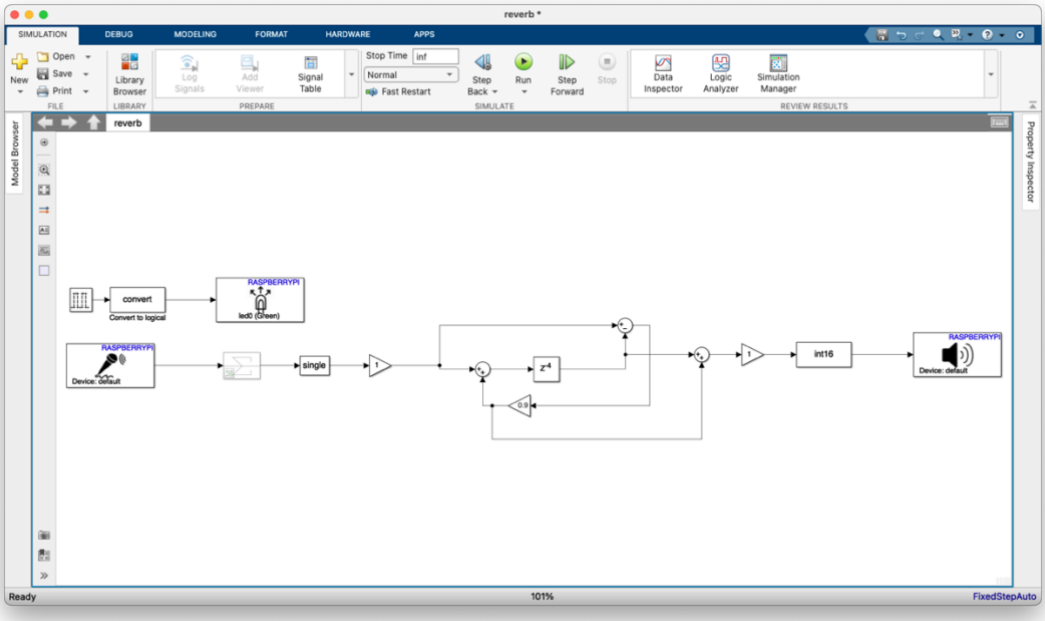


Il·lustració 30 - Diagrama de blocs d'un sistema reverberador. Extret de [1]

Per acabar, podem afirmar que la reverberació és un sistema lineal i invariant en el temps.

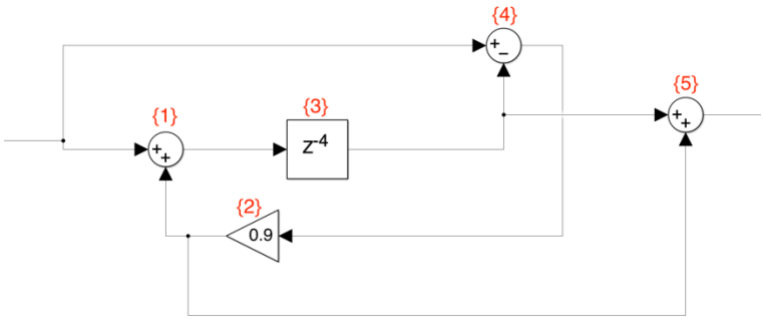
3.3.3.2. Disseny de blocs de Simulink

El nostre model dissenyat correspon al model mostrat a la il·lustració 31:



Il·lustració 31 - Captura de pantalla corresponent al model de reverberació

Podem observar el mòdul de captació que ofereix un senyal que serà bifurcat en diverses iteracions.



Il·lustració 32 - Detall del model de reverberació

Tal i com es mostra a la il·lustració 32, després de la primera bifurcació, el senyal original es troba amb un sumador {1}; posteriorment, un bloc retardador {3} que entrega el senyal a un altre sumador {4} que suma el senyal original amb aquest senyal retardat i invertit (es tracta d'una resta). Aquest senyal resultant del bloc {4} té un guany aplicat i es suma tant al senyal original sense retard i al senyal retardat. Aquesta suma dels tres llaços produeix el nostre efecte de reverberació.

3.3.4. Flanger

3.3.4.1. Fonaments teòrics

L'efecte *flanger* torna a ser un efecte basat en retardadors com els dos anteriors. En aquest cas però, el retard de la segona versió no és constant, sinó que és variable en el temps. El valor d'aquest retard depèn d'una ona generada amb una freqüència concreta que ens permeti diferenciar les dues versions del senyal. És habitual doncs, que els efectes de *flanger* treballin amb ones sinusoidals de freqüències baixes(entre 100 i 1000Hz), generades per oscil·ladors de baixa freqüència (*Low Frequency Generators, LFO*).

En el cas més simple d'un efecte de *flanger* l'expressió algebraica serà:

$$\boldsymbol{y[n] = x[n] + g[x[n - M(n)]]}$$

Equació 8 - Expressió algebraica per un efecte de flanger

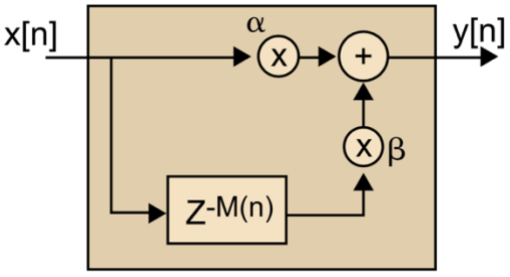
A l'equació número 8 M(n) és l'ona responsable del retard variable i seguirà la forma, en cas d'una sinusoidal com:

$$\boldsymbol{M(n) = M_0(1 + A \cdot sin(2\pi \cdot f \cdot n \cdot T)]}$$

Equació 9 - Expressió algebraica per una ona sinusoidal per un LFO

Els paràmetres de l'equació 9 són l'amplitud de l'ona sinusoidal (A), la seva freqüència (f) i el seu període (T). A més a més sabem que haurem de discretitzar i quantificar l'ona i per tant, es produiran arrodoniments en funció dels paràmetres d'aquest procés.

I un possible diagrama de blocs per aquest model es mostra a la il·lustració 33:

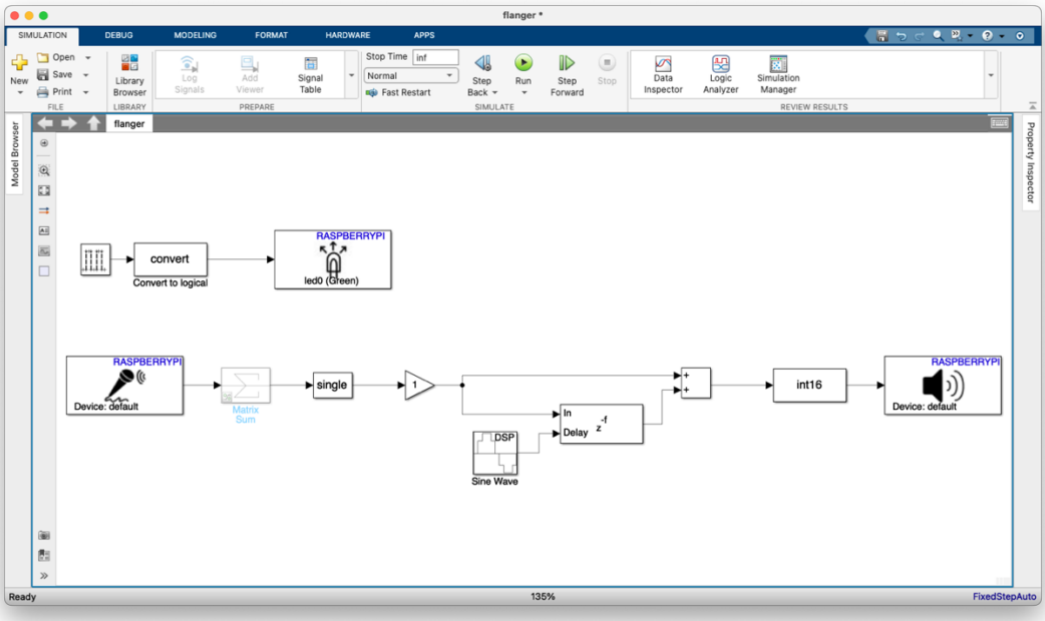


Il·lustració 33 - Diagrama de blocs per un sistema de flanger. Extret de [1]

Per acabar, podem afirmar que el *flanger* és un sistema lineal però variant en el temps.

3.3.4.2. Disseny de blocs de Simulink

La nostra proposta de disseny és la que es mostra a la il·lustració 34:

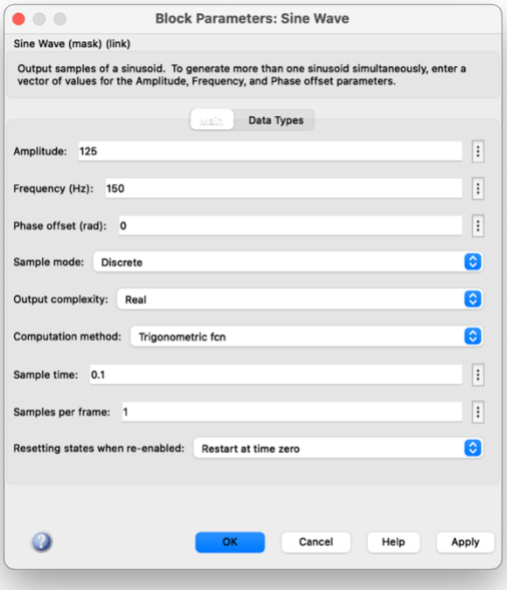


Il·lustració 34 - Captura de pantalla corresponent al model de flanger

En aquest model trobem un bloc de retard variable controlat per un generador d'ones sinusoidals que retarda el senyal original en funció dels paràmetres d'aquest segon.

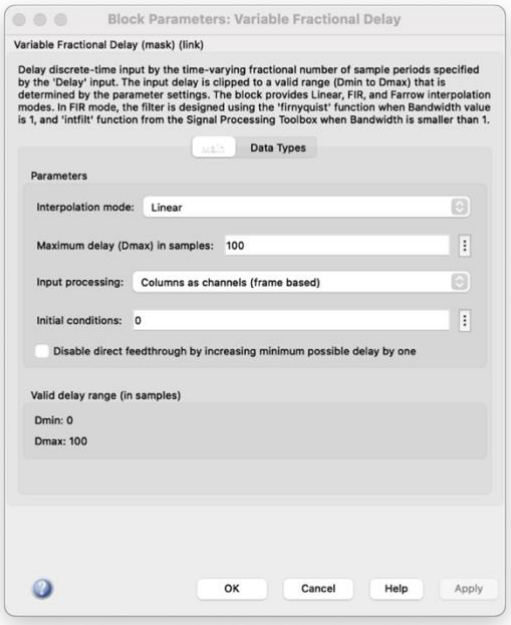
Tal i com s'ha explicat anteriorment, la freqüència d'aquest generador ha d'estar entre els entre 100 i 1000Hz de forma que es pugui percebre l'efecte. L'amplitud de l'ona generada ens determinarà en quina quantitat s'aplicarà al senyal i per tant, també varia en funció de l'amplitud del senyal original (i del seu procés de conversió analògic-digital). Amb el nostre sistema final, els valors aproximats d'amplitud de treball es troben entre 75 i 175 de mitjana.

La configuració del generador és la que apareix a la il·lustració 35:



Il·lustració 35 - Configuració del bloc del generador d'ona sinusoidal

El bloc de retard variable s'ha de configurar per treballar amb el mode *frame based*[52] (l'adequat per a treballar amb senyals d'àudio) tal i com es pot observar a la il·lustració 36:



Il·lustració 36 - Configuració del bloc de retard variable

Posteriorment, el senyal retardat es suma al senyal original i es reprodueix mitjançant el mòdul comú per a la sortida.

3.3.5. Tremolo

3.3.5.1. Fonaments teòrics

L'efecte de tremolo es basa en la modulació de l'amplitud del senyal original. És un efecte que no es basa en retards (com els treballats en els apartats anteriors), sinó que es basa en la variació de l'amplitud del senyal original mitjançant ones sinusoidals de freqüències baixes (normalment es treballa per sota dels 20Hz).

És un efecte que no modifica el contingut freqüencial del senyal original sinó que només en modifica l'amplitud en funció de la freqüència de l'ona sinusoidal.

La seva expressió algebraica en el cas més senzill serà:

$$\boldsymbol{y[n] = x[n] \cdot M[n]}$$

Equació 10 - Expressió algebraica per un efecte de tremolo

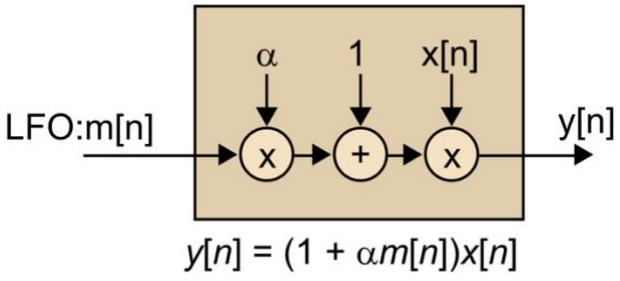
A l'expressió 10, M(n) és l'ona moduladora i seguirà la forma, en cas d'una sinusoidal com:

$$\boldsymbol{M(n) = (1 + A \cdot sin(2\pi \cdot f \cdot n \cdot T)]}$$

Equació 11 - Expressió algebraica per una ona sinusoidal per un LFO

Els paràmetres de l'equació 9 són l'amplitud de l'ona sinusoidal (A), la seva freqüència (f) i el seu període (T). A més a més, sabem que haurem de discretitzar i quantificar l'ona i per tant, es produiran arrodoniments en funció dels paràmetres obtinguts d'aquest procés.

Un possible diagrama de blocs per aquest model es mostra a la il·lustració 37:

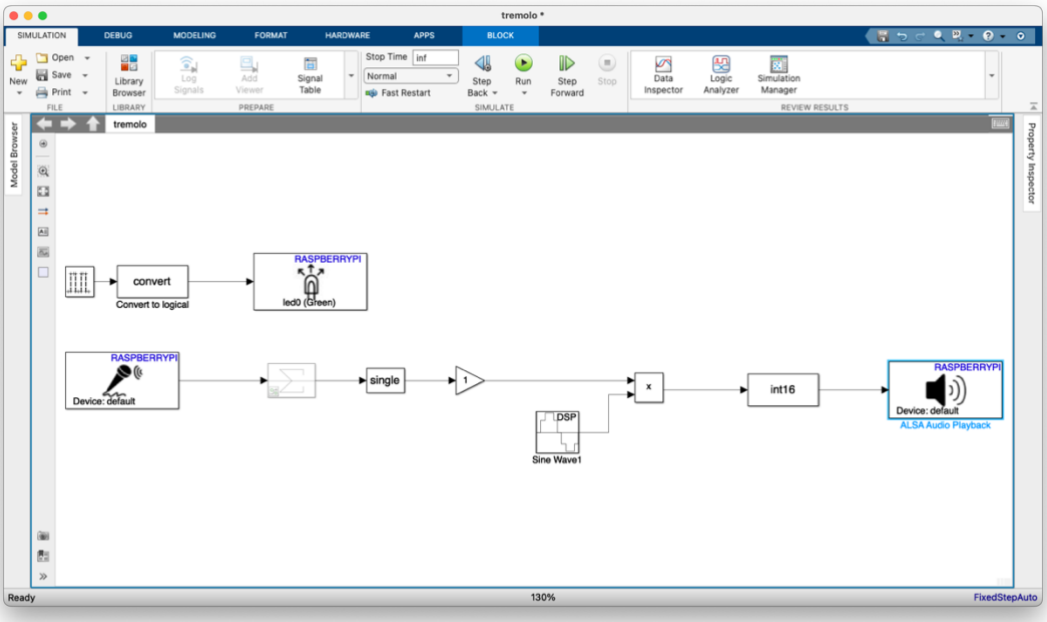


Il·lustració 37 - Diagrama de blocs per un sistema de tremolo. Extret de [1]

Per acabar, podem afirmar que un efecte tremolo és un sistema lineal i variant en el temps.

3.3.5.2. Disseny de blocs del Simulink

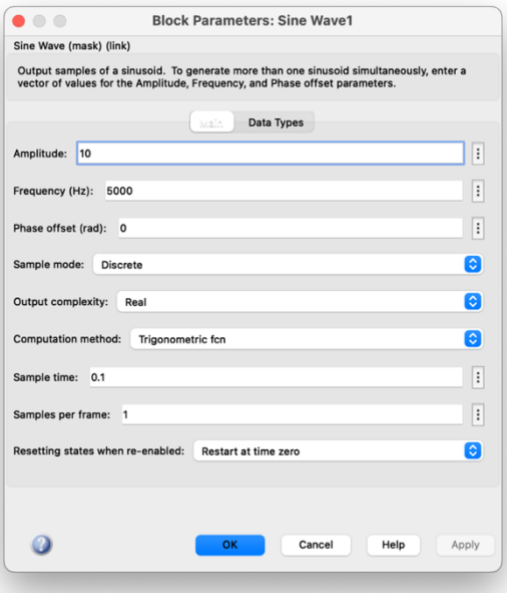
El nostre disseny de l'efecte es modela de la forma que apareix a la il·lustració 38:



Il·lustració 38 - Captura de pantalla corresponent al model de tremolo

A la il·lustració 38 hi trobem el bloc de captació que condueix el senyal a un bloc multiplicador. Aquest bloc multiplicador disposa de dues entrades; el senyal original i una entrada per un generador de senyals sinusoidal que és l'encarregat de modular l'amplitud. La sortida del bloc multiplicador es dirigeix cap al bloc reproductor habitual.

Els paràmetres utilitzats per al LFO que genera el bloc del generador de senyals es configura com apareix a la il·lustració 39:



Il·lustració 39 - Configuració del bloc generador de l'ona sinusoidal

3.3.6. Equalitzador

3.3.6.1. Fonaments teòrics

L'equalització de senyals és un dels tractament més habituals dins els efectes d'àudio. És el procés d'ajustar el contingut freqüencial d'un senyal variant modificant la relació que tenen les diferents bandes de freqüència dins el mateixa senyal. Aquest procés es pot produir tant per motius acústics (compensar l'acústica del recinte), per motius artístics (per ressaltar uns instruments per sobre d'altres en una mescla) o bé per la optimització dels mateixos equips (entregant als altaveus les freqüències adequades per la seva reproducció).

Tots els equalitzadors es basen en filtres, i en funció del tipus i el nombre de filtres que tinguin ofereixen unes prestacions o altres. Podem trobar filtres tipus:

- **Passa-baixos:** filtren l'espectre en altes freqüències. Només deixen passar les freqüències per sota de la freqüència de tall.

- **Passa-banda**: deixen passar una banda en concret i atenua la resta de freqüències.

- **Passa-alts:** filtren l'espectre en baixes freqüències. Només deixen passar les freqüències per sobre de la freqüència de tall.

- **Band-stop:** filtren una banda concreta de freqüències. Les parts superior i inferior de la banda ajustada, passen el filtre.

- **Low Shelving i High Shelving:** Tornen a ser filtres passa alts o passa baixos, però en aquest cas podem ajustar com afecten la banda en qüestió. Podem ajustar el guany o atenuació de la banda afectada.

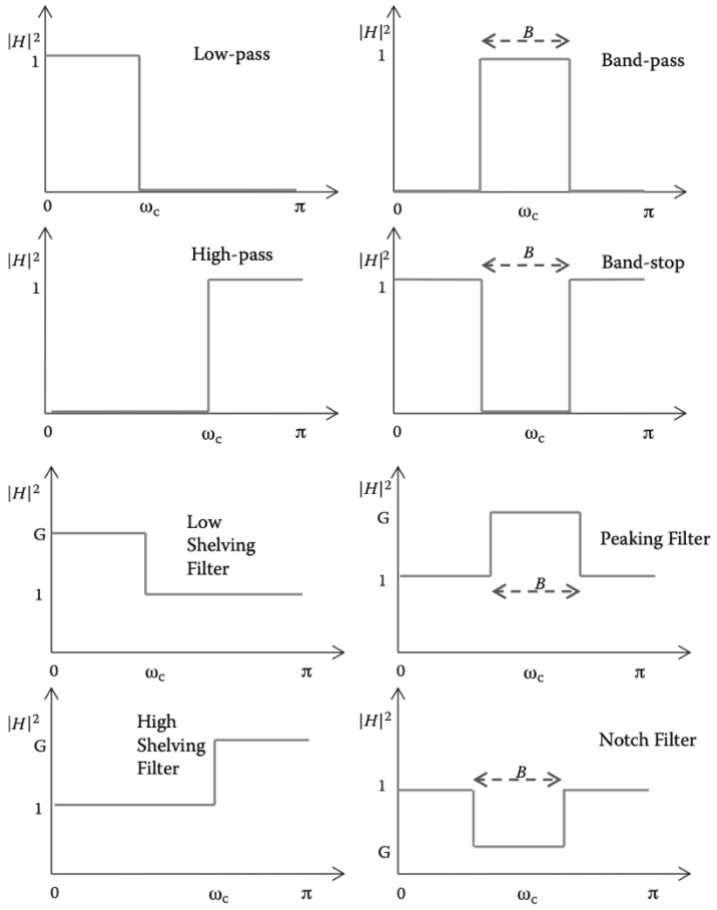
- **Notch o de banda eliminada:** Deixa passar tot l'espectre menys una banda en concret.

- **Peak o de banda amplificada:** Modifica una banda en concret i no modifica la resta de l'espectre.

Podem veure una representació esquemàtica de tots aquests filtres a la il·lustració 40 d'aquesta mateixa memòria.

Els paràmetres que ens ofereixen els filtres són la freqüència de tall -on comença a actuar; l'ample de banda -quin ample de l'espectre afecta; i l'ordre del mateix -la complexitat matemàtica del filtre i també el nivell i la forma de l'atenuació amb què afectarà el senyal.

Queda fora dels objectius d'aquesta memòria l'anàlisi de les diferents funcions de transferència així com dels diferents diagrames de pols i zeros que ens ofereixen, tot i que són consultables a la font bibliogràfica [48]

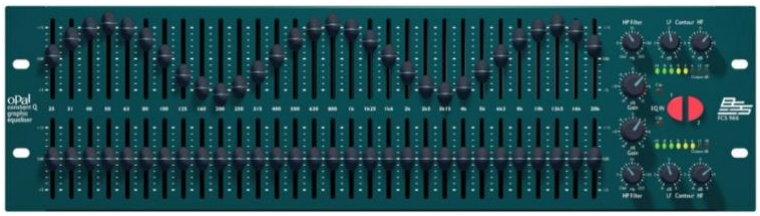


Il·lustració 40 - Representació esquemàtica dels diferents tipus de filtres. Extret de [48]

Amb aquest conjunt de filtres, i en funció dels paràmetres de cadascú que se'n poden manipular, podem construir diversos tipus d'equalitzadors. Els paràmetres bàsics de tots els equalitzadors són el control de guany, el selector de freqüència i l'ample de banda. Aquests paràmetres existiran per a cada banda que es pugui modificar i determinaran quin tipus d'equalitzador, serà. El número de bandes depèn de les característiques de l'equalitzador tot i que normalment poden equivaldre a les octaves de l'espectre (o als seus submúltiples).

En funció dels paràmetres que permeten modificar doncs, podem llistar diferents tipus d'equalitzadors:

- **Equalitzador gràfic** (il·lustració 41)**:** permet modificar l'amplitud d'unes determinades bandes de freqüència fixes. No permet el nombre de bandes ni el seu ample.

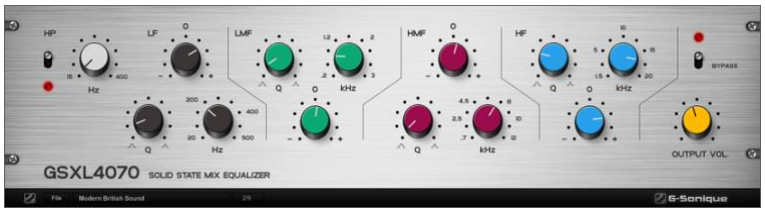


Il·lustració 41 - Equalitzador gràfic de BSSAudio model FCS-966[25]

- **Equalitzador semi-paramètric** (il·lustració 42)**:** permet modificar l'amplitud i la freqüència d'un numero de bandes determinat. No permet modificar l'ample de banda del filtre on actua.

Il·lustració 42 – Equalitzador semi-paramètric VT1 [26]

- **Equalitzador paramètric** (il·lustració 43)**:** permet modificar tots tres paràmetres per a cada banda de freqüència on actua.



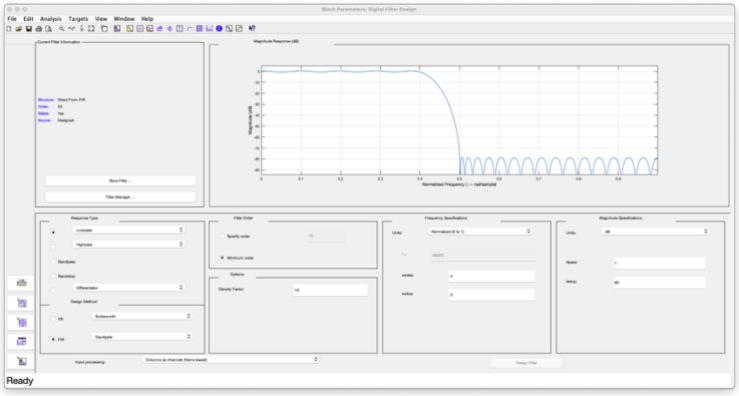
Il·lustració 43 - Equalitzador paramètric model GSXL4070 [27]

Per tots tres tipus d'equalitzador trobem des d'esquemes més senzills (menys nombre de bandes de freqüència d'actuació) fins a esquemes més complexes (fins a 31 bandes d'actuació en alguns equalitzadors gràfics o combinacions de diversos filtres en equalitzadors de taules de mescla digital).

3.3.6.2. Disseny de blocs del Simulink

En el nostre cas, el modelat de l'equalitzador inclou el disseny dels filtres necessaris per a la modificació de l'àudio que captem i busca ser un equalitzador de tipus semi paramètric.

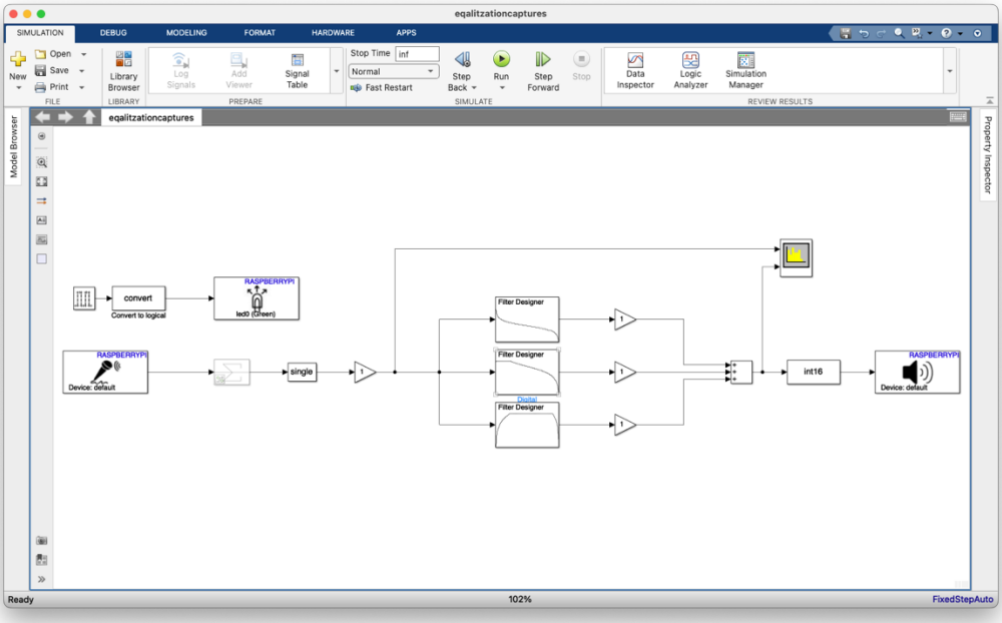
Per fer-ho, el Simulink ofereix dues opcions; la primera, utilitzant l'eina *Digital Filter Design*[28] que ens permet, mitjançant una interfície dedicada, el disseny de filtres i la possible exportació com a codi al mateix MATLAB.



Il·lustració 44 - Captura del DFDesign amb un filtre d'exemple

Aquesta eina permet veure els diagrames de magnitud i fase, el diagrama de pols i zeros i també el retard de grup que aplica el filtre. Com s'ha comentat anteriorment, tot i la importància d'aquests paràmetres en l'estudi dels filtres, queden fora de l'estudi d'aquest TFG.

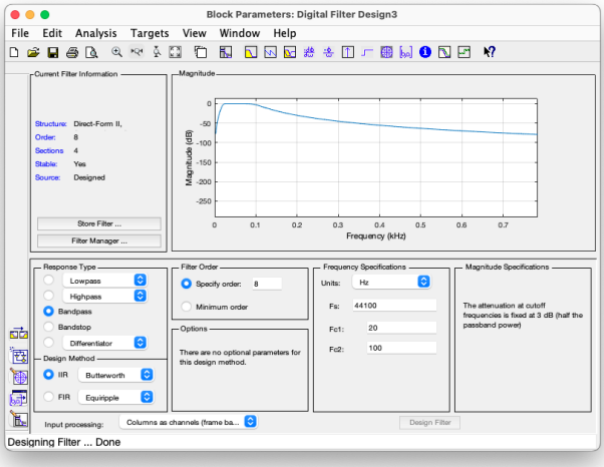
El primer model es genera amb aquesta eina, però s'observen mancances com ara el control de guany que s'ha de fer amb blocs diferenciats. Un primer model d'equalitzador el que es mostra a la il·lustració 45:



Il·lustració 45 - Captura de pantalla corresponent al primer model d'equalitzador

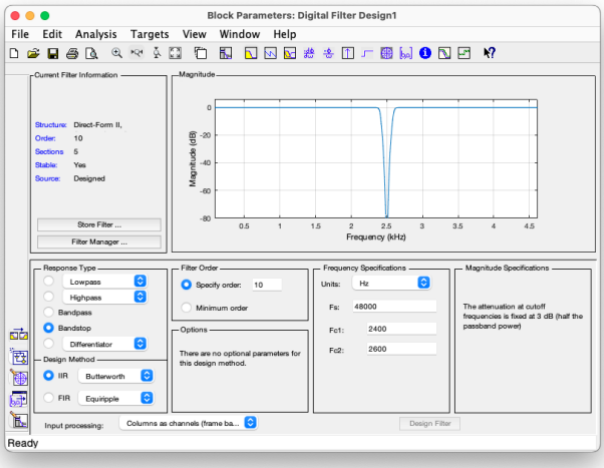
En aquest model es poden observar les tres bandes amb què es separa el senyal original per poder ser filtrat. Un cop superat cada filtre, trobem el guany per a cada branca (el que ens modificarà l'amplitud de la banda desitjada), un bloc sumador i, finalment, el bloc de reproducció.

El filtre superior és un filtre passa banda però centrat entre els 20 i 100Hz. A partir d'aquí, gràcies a l'increment de l'ordre, tenim una atenuació de -50 dB al voltant dels 200Hz. La captura de la resposta en magnitud és:



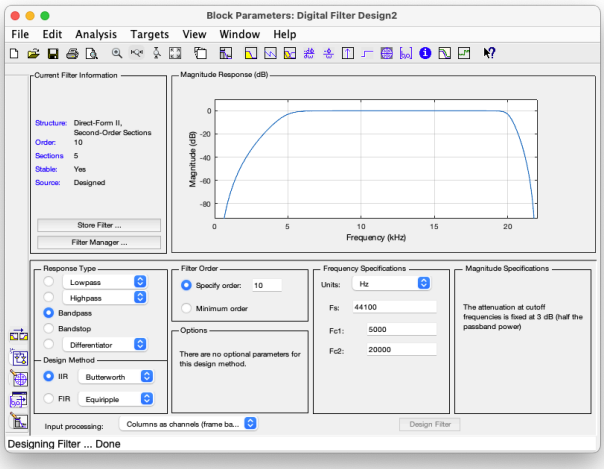
Il·lustració 46 - Captura del DFD amb el filtre per les freqüències greus

El segon filtre també és un band stop però aquesta vegada centrat a 2500Hz, en aquest cas d'ordre 10. La seva resposta en magnitud és:



Il·lustració 47 - Captura del DFD amb el filtre per les freqüències mitges

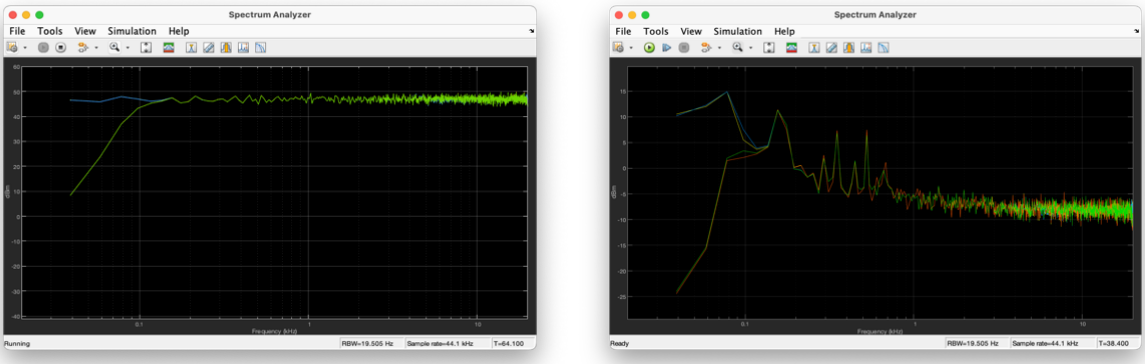
Finalment, la tercera branca es filtra amb un últim filtre de banda de pas amb les especificacions que segueixen:



Il·lustració 48 - Captura del DFD amb el filtre per les freqüències altes

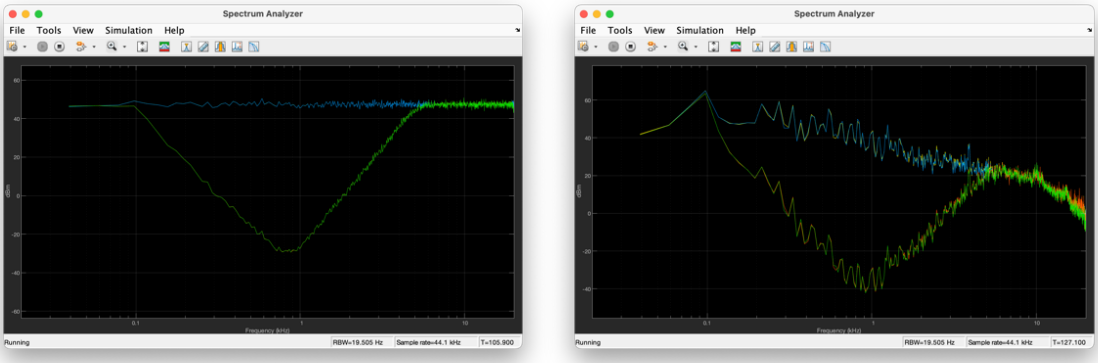
En aquest model s'introdueix el bloc de l'analitzador d'espectre que ens permet veure quin és el contingut freqüencial del senyal que estem manipulant. Les il·lustracions de la 49 i fins la 51 corresponen a l'execució en temps real de l'equalitzador:

Per un guany de 0 a la sortida del filtre de greus, il·lustració 49:



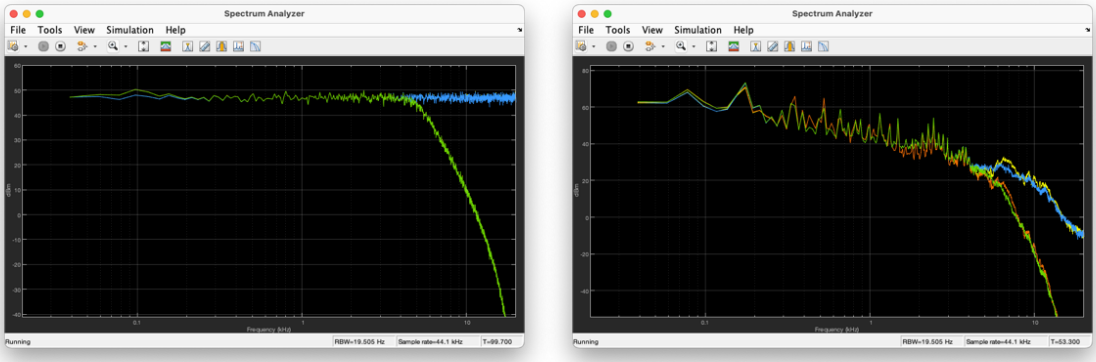
Il·lustració 49 - Analitzador d'espectre amb el filtre greu atenuant.

Per un guany de 0 a la sortida del filtre de mitjos, il·lustració 50:



Il·lustració 50 - Analitzador d'espectre amb el filtre de mitjos atenuant

Per un guany de 0 a la sortida del filtre d'aguts, il·lustració 51:



Il·lustració 51 - Analitzador d'espectre amb el filtre d'aguts atenuant

A banda de les diferències evidents entre les atenuacions que fa de l'àudio i que corresponen als diferents filtres analitzats anteriorment, podem veure com en les diferents captures situades a la dreta apareixen fins a 4 senyals ja que el tractament és d'un senyal estèreo.

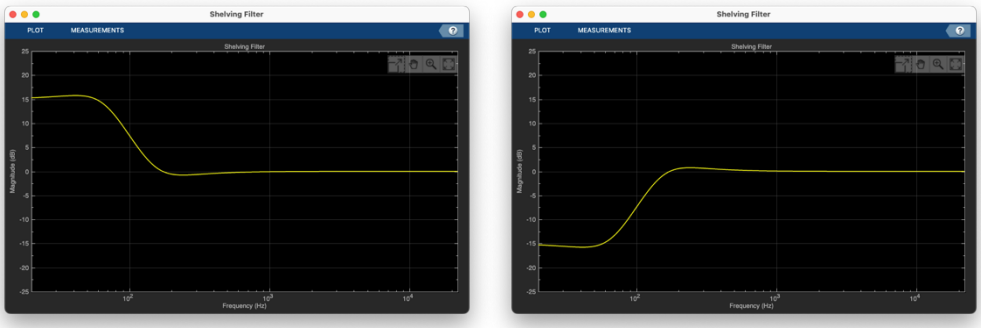
La segona proposta d'equalitzador es basa en blocs de filtres del complement *Audio Toolbox*[29] que es pot afegir al Simulink. En aquest banc de filtres trobem des de filtres Shelving ja dissenyats fins a blocs d'equalitzador paramètrics complerts on podem escollir el numero de bandes. Aquest segon disseny, amb tres filtres diferenciats per tot l'espectre, és la còpia d'una secció d'equalització d'una taula de mescles Yamaha Mg12, que inclou tres bandes d'equalització amb les següents especificacions[30]:

-Banda Low: guany de -15 a +15 dB, tipus *Shelving,* freqüència central 100Hz
-Banda Mid: guany de -15 a +15 dB, tipus *Peak,* freqüència central 2500Hz
-Banda Hi: guany de -15 a +15 dB, tipus *Shelving,* freqüència central 10kHz

Aquesta vegada, però, el diagrama és lleugerament diferent ja que no bifurquem el senyal inicial en cada filtre i el sumem després sinó que circula per tota la cadena de forma seqüencial. Així, el model té un major control dels paràmetres que necessitem modificar ja que els blocs poden ser controlats per constants externes que al seu temps, poden variar-se mitjançant controls físics.

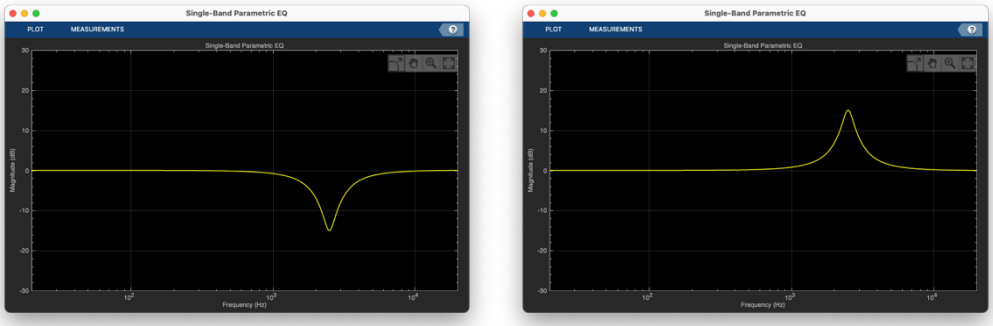
Els filtres del model per imitar els filtres de la taula de mescles Yamaha Mg12 són els que segueixen:

Pel filtre de la banda Low, il·lustració 52:



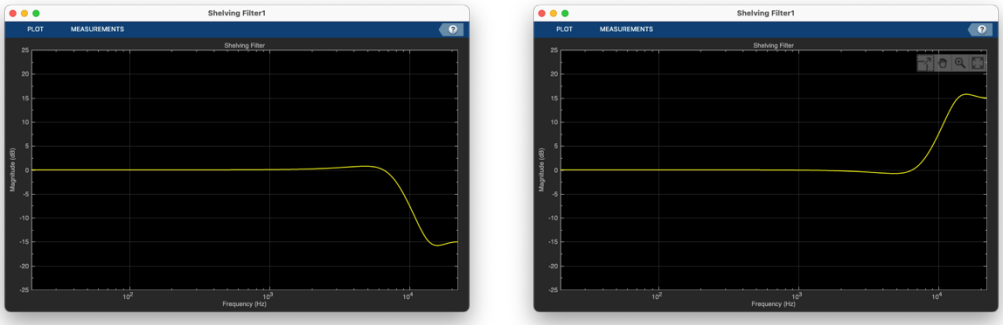
Il·lustració 52 - Gràfic de magnitud amb el màxim i mínim guany aplicat a la banda Low

Pel filtre de la banda Mid, il·lustració 53:



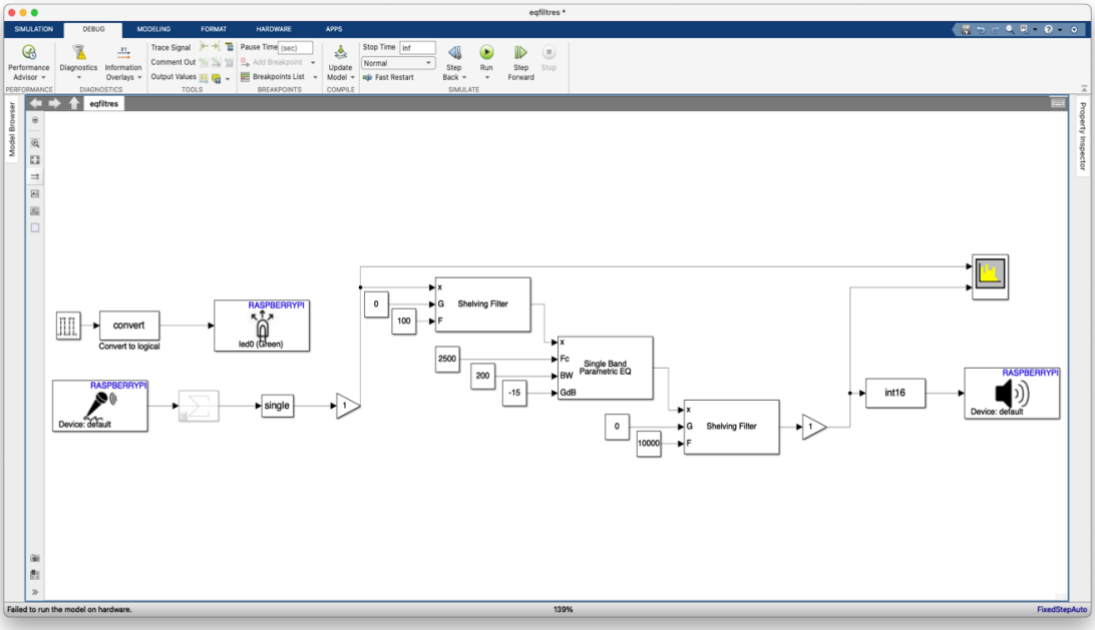
Il·lustració 53 - Gràfic de magnitud amb el màxim i mínim guany aplicat a la banda Mid

Pel filtre de la banda High, il·lustració 54:



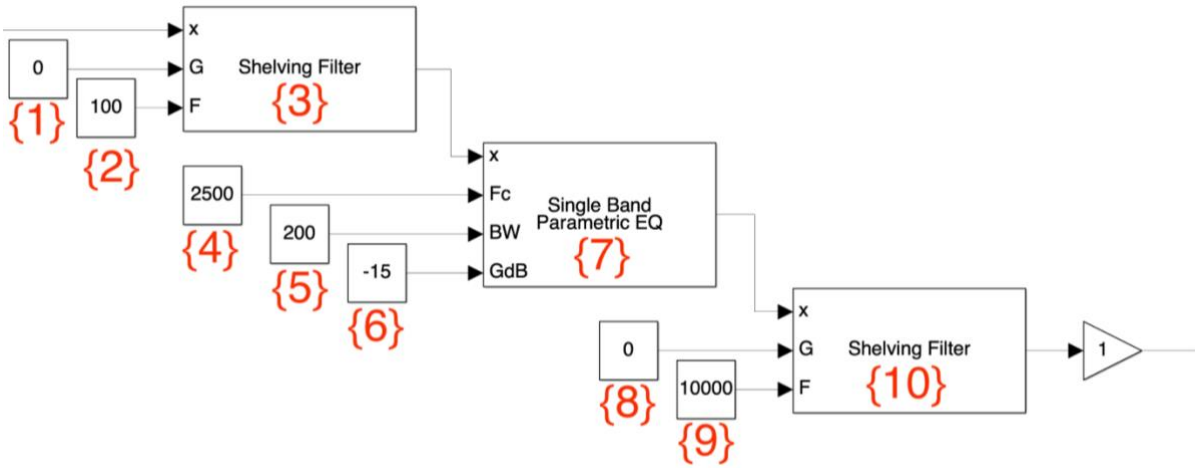
Il·lustració 54 - Gràfic de magnitud amb el màxim i mínim guany aplicat a la banda High

I el model dissenyat per aquest equalitzador és el que es mostra a la il·lustració 55:



Il·lustració 55 - Captura de pantalla corresponent al segon model d'equalitzador

En aquest model de la il·lustració es pot veure que el bloc de captació habitual condueix la senyal cap a un bloc d'equalització format pels tres filtres en sèrie. El bloc central amb més detall queda així:



Il·lustració 56 - Detall del segon model d'equalitzador

Podem veure-hi que els valors de les constants {1} i {2} controlen el guany i la freqüència de tall del primer filtre tipus *Shelving* que actua com a passa baixes. Els blocs de constats {4}, {5} i {6} controlen els paràmetres de la banda mitja (que està implementada sobre un bloc paramètric). Finalment, les constants {8} i {9} controlen el guany i la freqüència de tall del segon filtre tipus *Shelving* que actua com a passa altes.

Els valors que apareixen a les constants són els valors que han d'aparèixer en les configuracions dels mòduls en cas de no voler fer-los variables mitjançant els potenciòmetres externs.

Finalment, el senyal modificat apareix a un mòdul de guany que controla que no estiguem generant una senyal sobresaturada.

Les captures de l'espectre que ens ofereix aquest modelat són les corresponents a les il·lustracions 57 fins la 59.

Per un guany de -15 a la sortida del filtre Low, il·lustració 57:

Il·lustració 57 - Analitzador d'espectre amb el filtre greu actuant

Per un guany de -15 a la sortida del filtre Mid, il·lustració 58:

Il·lustració 58 - Analitzador d'espectre amb el filtre de mitjos actuant

S'ofereixen dues parelles de captures amb variació de l'amplitud del filtre. En el cas superior s'intenta imitar l'ample de banda que ofereix el filtre dissenyat amb la *Digital Filter Design.* En la segona parella, el filtre actua com actuaria amb l'ample del filtre de la taula de mescles Yamaha Mg12.

Per un guany de -15 a la sortida del filtre High, il·lustració 59:

Il·lustració 59 - Analitzador d'espectre amb el filtre high atenuant

Finalment, una vegada existeixen els dos models funcionals, es planteja quin és més adient pel modelat final. Més enllà de la musicalitat que poden tenir els models, el primer, implementat amb el *Digital Filter Design,* és molt més agressiu (els valors de guany són molt més elevats) i més difícil de controlar mitjançant blocs externs. Com que un dels objectius plantejats és poder controlar els efectes amb els potenciòmetres externs, decidim fer la implementació del model final amb la segona versió, la que és implementada amb les eines de l'*Audio Toolbox*.

3.3.7. Disseny del hardware

Una vegada dissenyats els efectes, es necessiten dissenyar els controls hardware per els efectes. La primera idea va ser utilitzar un total de 5 potenciòmetres de forma que el primer actués de selector de l'efecte desitjat i que els altres servissin per a la modificació dels paràmetres dels efectes. Així, el plantejament inicial va ser el proposat a la taula 2:

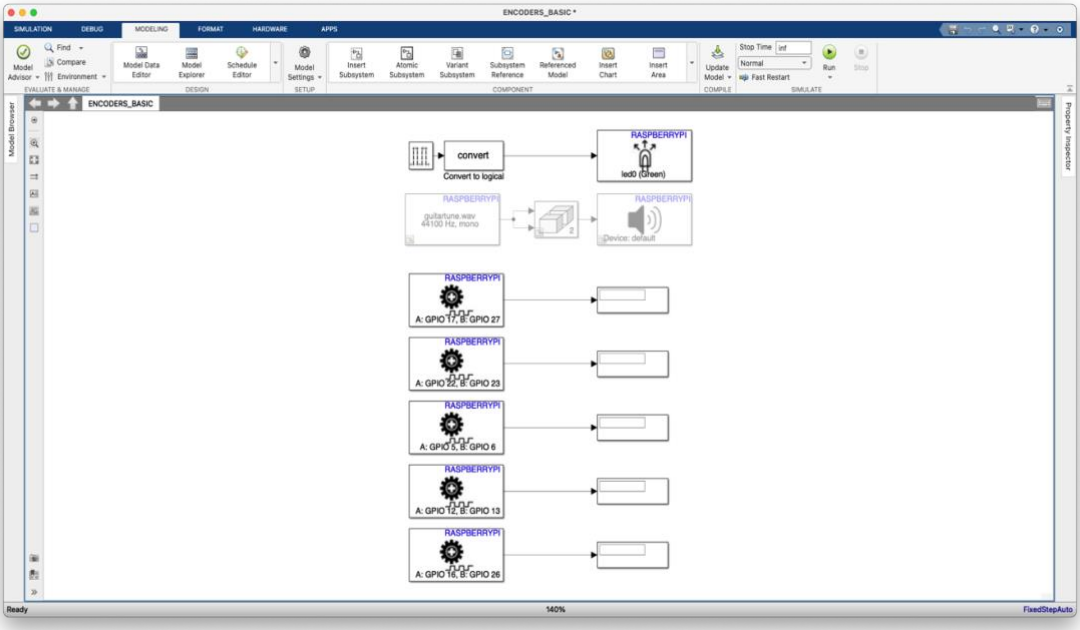
Pot.	Delay	Reverberació	Flanger	Tremolo	Equalitzador
1	Selector				
2	Número de mostres de retard del primer retard	Número de mostres de retard	Amplitud LFO	Amplitud LFO	Guany banda baixa
3	Guany del primer retard	Guany de la realimentació	Freqüència del LFO	Freqüència del LFO	Freqüència central banda mitja
4	Número de mostres de retard del segon retard				Guany banda mitja
5	Guany del segon retard				Guany banda alta

Taula 2 - Resum del disseny del hardware per el sistema

Amb aquest plantejament es genera un model pel control dels potenciòmetres al Simulink per testejar-lo abans d'implementar-lo amb el model definitiu que contingui tots els efectes modelats.

Al començar a treballar amb el bloc anomenat *Encoder* dissenyat per controlar el RPi, es van observar els primers canvis en els espais de treball del Simulink. A mode de resum, ja que els problemes es tractaran a l'apartat 3.4.4 "XCP i GPIO" d'aquesta mateixa memòria, el fet d'utilitzar aquest bloc modifica automàticament el *Solver* de l'espai de treball i és imprescindible funcionar amb *fixed-step* i un temps concret de *sample time*. Aquest fet va implicar refer tots els models d'efectes usats fins al moment.

Un cop solucionat el problema, el model generat és el que es mostra a la il·lustració 60:

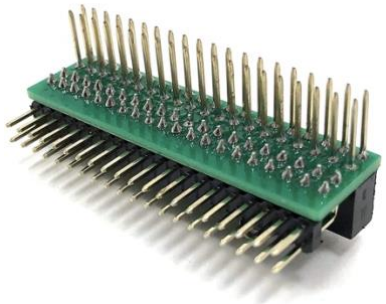


Il·lustració 60 - Captura de pantalla corresponent al disseny de hardware

Simplement trobem 5 blocs de potenciòmetre amb un display per veure el valor del comptador, el bloc de control a la part superior i un bloc de reproducció en local al RPi. Als diferents potenciòmetres se'ls ha d'assignar un dels pins del *GPIO* de què disposa el RPi 4.

Això no és un problema sinó fos perquè, com veurem posteriorment, la targeta d'àudio amb què estem treballant en aquest moment es comunica al RPI 4 mitjançant GPIO i per tant és imprescindible veure quins pins concrets utilitza. A més a més, aquesta targeta concreta no disposa d'un *splitter* dels GPIO sinó que s'han de soldar a la placa (fet que anul·la la garantia de la targeta) o buscar una alternativa.

L'alternativa arriba en forma de *splitter* de GPIO, mostrat a la il·lustració 61, que encareix una mica el preu del producte final.

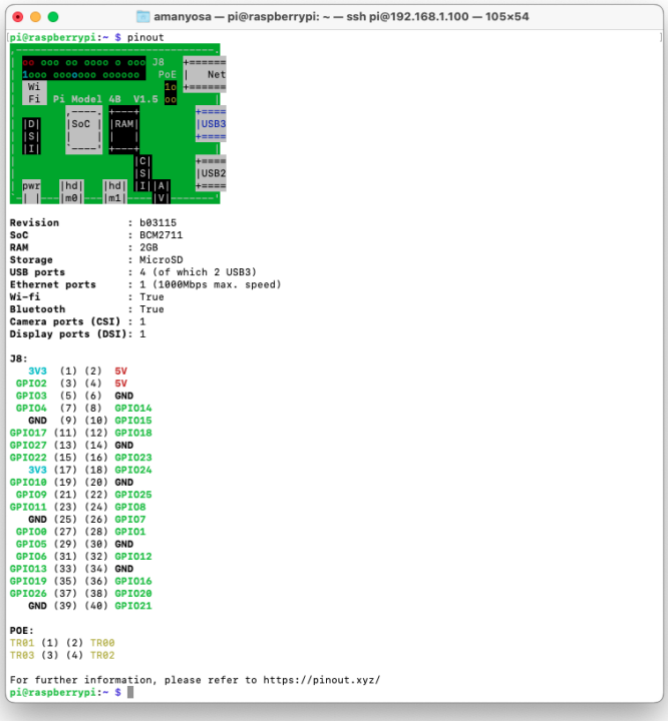


Il·lustració 61 - Splitter GPIO utilitzat al projecte [31]

El muntatge sobre la placa del RPi 4 queda com es mostra a la il·lustració 62:

Il·lustració 62 - Placa del RPi4 amb l'splitter de GPIO

En aquest punt comencem a investigar quins són els pins que tenim disponibles. Primer de tot, mitjançant la comanda pinout a la línia de comandes del RPI obtenim la captura següent:

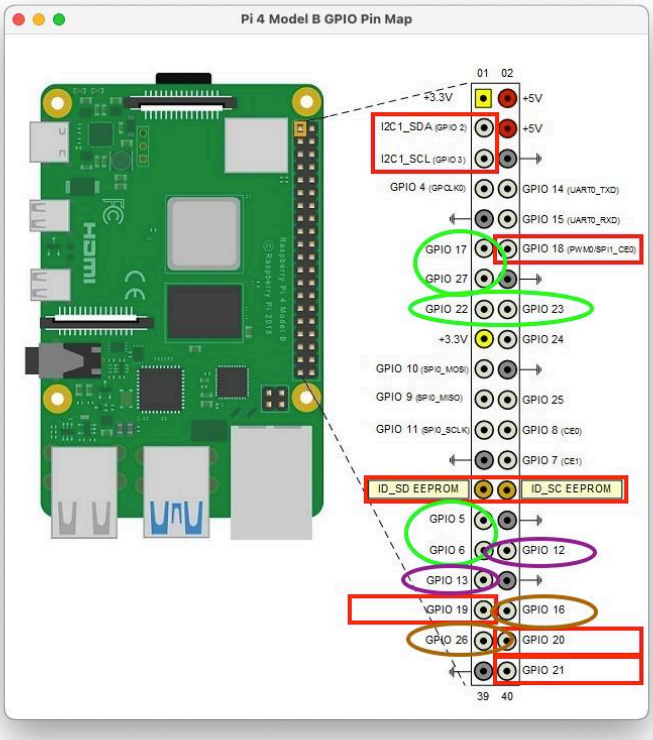


Il·lustració 63 - Captura de la sortida a la comanda pinout en una sessió de SSH amb el RPi 4

En aquesta il·lustració 63, podem observar quins pins tenim a la placa. Aquest esquema l'hem de contrastar amb les indicacions que ens ofereix el fabricant de la targeta d'àudio, que ens indica que la targeta es comunica amb el RPI 4 mitjançant els pins següents:

Il·lustració 64 - Especificacions de funcionament dels GPIO segons el fabricant HiFiBerry

Per tant, sobre l'esquemàtic que ens ofereix el mateix Simulink, l'esquema de connexions -sabent que necessitem dos pins per potenciòmetre (un per augmentar el valor i un per restar el valor)- és el que segueix a la il·lustració 65:



Il·lustració 65 - Esquema de connexions definitiu sobre un model del RPi 4

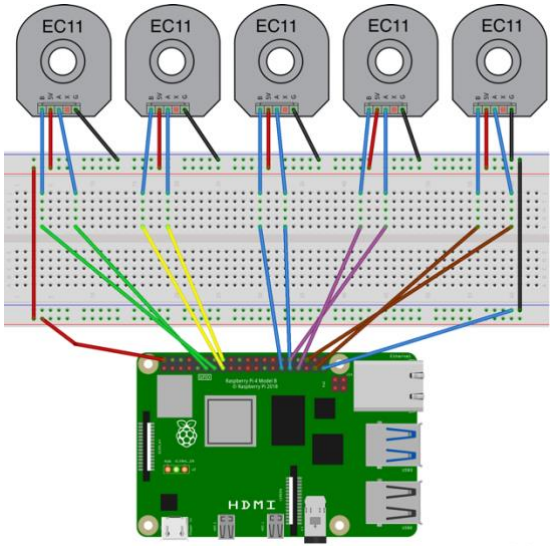
A mode de llegenda sobre la il·lustració 65; trobem en vermell els pins reservats per a la targeta d'àudio, en verd, lila i marró les parelles de pins que podem utilitzar per als *encoders*. Com es pot veure al model de la figura 61, els blocs dels *encoders* ja estan configurats amb aquesta nomenclatura. La taula final, doncs, pel disseny del hardware és el que es mostra a la taula 3:

Pot.	GPIO	Reverberació	Delay	Flanger	Tremolo	Equalitzador
1	17/27	Selector				
2	22/23	Número de mostres de retard	Número de mostres de retard del	Amplitud LFO	Amplitud LFO	Guany banda baixa

			primer retard			
3	5/6	Guany de la realimentació	Guany del primer retard	Freqüència del LFO	Freqüència del LFO	Freqüència central banda mitja
4	12/13		Número de mostres de retard del segon retard			Guany banda mitja
5	16/26		Guany del segon retard			Guany banda alta

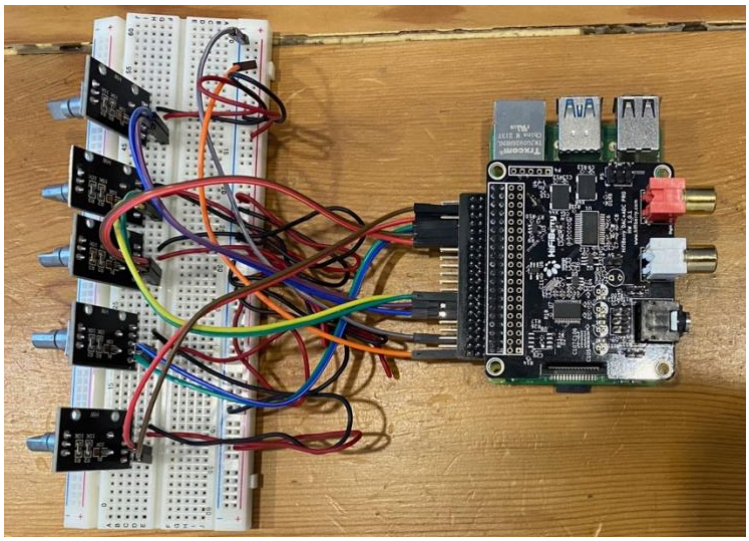
Taula 3 - Disseny definitiu de les connexions hardware del sistema

L'esquema del muntatge és el que es representa a la il·lustració 66:



Il·lustració 66 - Esquema de les connexions GPIO i els potenciòmetres.

El muntatge obtingut és el que es mostra a la il·lustració 67:



Il·lustració 67 - Fotografies detall del muntatge del hardware

Per acabar les connexions, falta anomenar també la línia d'alimentació, que pot ser tant al pin 1 (3,3 volts) com al pin 2 (5 volts), ja que els potenciòmetres poden treballar als dos voltatges. Finalment necessitem fer les connexions de la terra (*ground, gnd)* i utilitzem diversos dels pins destinats a aquest propòsit que ens ofereix el RPI 4.

Una vegada muntat es procedeix a comprovar el correcte funcionament dels potenciòmetres. Tots funcionen de forma individual (mostren el valor concret al seu *display* vinculat al bloc) però quan s'intenta fer-los funcionar de forma simultània amb la targeta d'àudio, el Simulink és capaç de compilar i carregar el codi al RPi 4 però el protocol XCP provoca talls en l'àudio.

Aquest problema es desenvolupa i s'analitza a l'apartat 3.4.4. "XCP i GPIO".

La solució final pel control dels efectes acaba sent un apartat de *dashboard* (via controls gràfics) al mateix Simulink on, sense necessitat de *hardware* afegit, es poden controlar els paràmetres dels efectes.

3.4. Problemes amb el disseny

3.4.1. Compatibilitat del Sistema Operatiu

El primer problema de disseny es produeix a l'hora de configurar la plataforma inicial. Malauradament, el complement per comunicar-nos amb el hardware del RPi (el "Simulink Suport Package for Raspberry Pi Hardware" que hi havia disponible en aquell moment) no era plenament compatible amb la versió del Raspberry Pi OS que s'havia instal·lat al RPI4 i el procés d'instal·lació fallava sense donar cap retorn.

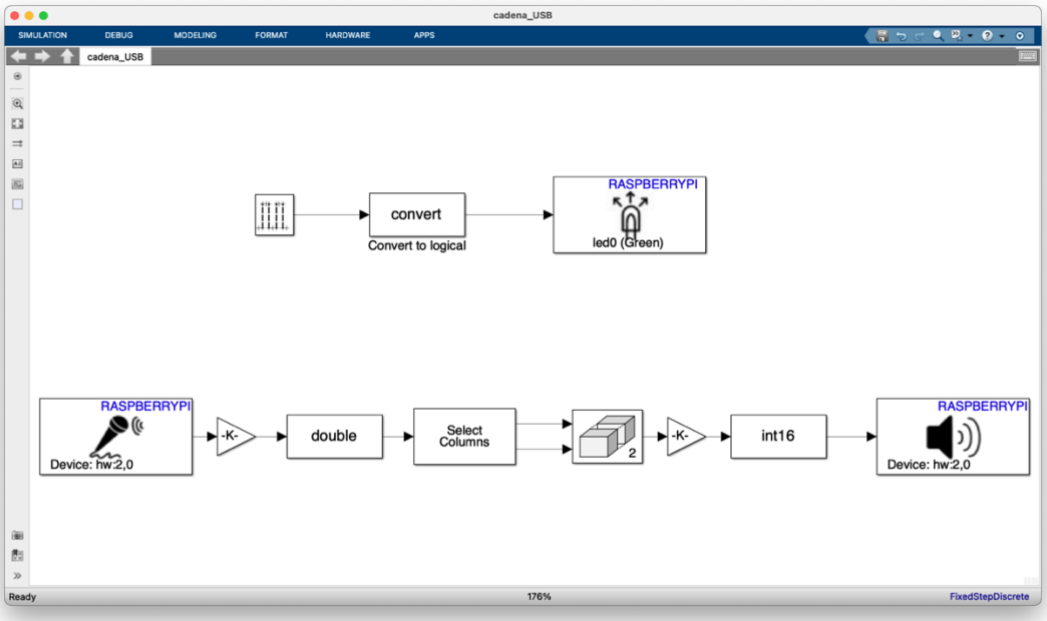
Per intentar solucionar-ho, es prova un altre mètode ofert per la mateixa MathWorks. En lloc d'instal·lar el complement sobre un sistema operatiu existent, l'empresa ofereix una imatge personalitzada de Raspbian per a l'ús amb RPi.

Un cop descarregada la versió que oferien quan es va començar el disseny (la versió 2021), aquesta imatge no era compatible amb el hardware de la RPi4. Aquesta limitació es produïa perquè estava basada en una distribució de Raspberry Pi OS anterior (segurament amb un kernel 4.19 o anterior, que no es podien executar sobre la RPi4). Les proves que es van fer sobre una RPi model 2 així ho indicaven, ja que amb aquest hardware sí que es podia fer funcionar la imatge original.

Per tant, la disjuntiva va ser si s'implementava el model sobre una RPI 2 o si es podien implementar les llibreries sobre una versió diferent de Raspbian. La limitació de potència de la RPi 2 davant la RPi 4 fa que es decideixi treballar amb una versió no tan actualitzada del sistema operatiu Raspbian, però amb un hardware més potent[32].

3.4.2. Captació i reproducció d'àudio

Al moment de començar el projecte, observem la necessitat d'obtenir una entrada d'àudio al RPi 4 que es soluciona amb la targeta externa per USB. Els primers models de *bypass* es construeixen sobre aquesta targeta i, malauradament, no s'obtenen els resultats amb qualitat adequada.

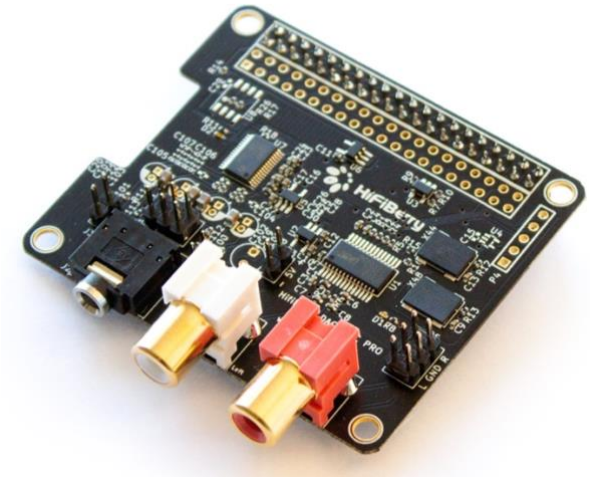


Il·lustració 68 - Captura de pantalla d'una cadena de captació i reproducció plantejada

Tot i tenir una cadena més complexa que la utilitzada finalment, a la reproducció es produeixen microtalls, saturació en cas de senyals molt panoramitzats i en general, una mala reproducció.

Paral·lelament, és una targeta que afegeix una latència molt exagerada al sistema.

Després de fer recerca en altres opcions, afegim la que es considera una de les millors targetes per als RPi i que s'ha utilitzat en algun dels projectes presentats anteriorment. Concretament, es tracta de la HifiBerry DAC+ ADC Pro [33] que es mostra a la imatge 69 i que ens ofereix una entrada i sortida estereofònica i guany ajustable via *jumper* a l'entrada.



Il·lustració 69 - Targeta de so externa HifiBerry DAC-ADC Pro [33]

A més a més, i en comparació a la targeta USB o a la targeta integrada al RPi 4, pot treballar a freqüències de mostreig de fins a 192kHz. Finalment, sabem que es comunica

amb el RPi 4 via els ports GPIO, i concretament, amb el protocol I2S que s'utilitza per a connectar dispositius d'àudio digital[34][35].

La conseqüència directa de treballar amb aquesta targeta, deixant de banda l'increment de preu del sistema, és que hem d'inhabilitar la targeta integrada al sistema. Aquest procediment es realitza mitjançant el fitxer de configuració (situat a /boot/config.txt de la targeta micro-SD) del RPi 4.

El fitxer s'ha d'editar afegint la línies següent al final del document:

```
Dtoverlay=hifiberry-dacplusadcpro
```

També s'han de treure els comentaris de les línies:

```
dtparam=i2c_arm=on
dtparam=i2s=on
dtparam=spi=on
```

I finalment, comentar les línies referents a la targeta integrada:

```
#dtparam=audio=on
```

Per comprovar que la targeta externa està ben instal·lada, podem consultar, mitjançant la línia de comandes, els recursos de què disposa el RPi4. El resultat de la comanda aplay -l i arecord -l és la que segueix:

Il·lustració 70 - Captura de la sortida a la comanda aplay -l i arecord -l al RPi 4

Una vegada es realitza aquest procediment, dins el Simulink, les referències als recursos de hardware disponibles canvien de nomenclatura, tal i com es mostra a la il·lustració 71.

Il·lustració 71 - Blocs de captura referenciats de forma diferent després de la configuració

A la part esquerra veiem com abans es referenciava amb la nomenclatura del sistema i ara es pot referenciar com a targeta predeterminada (bloc situat a la dreta). Amb aquest canvi de mètode, resolem els problemes amb el tractament del nombre de canals

d'entrada o en la gestió de la freqüència de mostreig amb què es realitza la captació. Amb el canvi de targeta doncs, es guanya en definició i facilitat de gestió del senyal captat.

3.4.3. Temps Real i latència

El tractament d'àudio mitjançant dispositius electrònics té el gran inconvenient que necessita treballar en temps molt curts per poder percebre'l d'una forma natural i no com a resultes d'un procés extern.

De fet, el nostre dispositiu ja treballa a temps real; el que no fa és treballar amb baixa latència (o una latència que sigui imperceptible per a l'oïda humana)[23].

Tot i treballar a temps real (capturem l'àudio, li apliquem l'efecte i el reproduïm una altra vegada), amb els valors obtinguts durant el procés descrit a l'apartat 3.3.1 "Bypass", som plenament conscients que qualsevol persona pot percebre els canvis d'una forma poc natural. Així, i com a conclusió el nostre dispositiu, amb les condicions de latència que es van mesurar, no es podia inserir en una cadena d'àudio per treballar en entorns de producció en viu.

Arribats en aquest punt es va començar una investigació sobre possibles solucions per intentar reduir aquests 500ms acumulats.

La primera opció va ser mirar si existien distribucions de Linux per poder treballar amb menys latència. Primer de tot, es van revisar els projectes presentats a l'apartat 2.4 de l'Estat de l'Art, per conèixer sobre quines distribucions treballaven.

Per la seva banda, el *projecte Raspberry Pi PC: DIY Guitar Pedal using NeuralPi*[8] sí que utilitza una altra distribució de Linux anomenada Elk Auido OS[36]. Una vegada descarregada aquesta distribució, es veu clarament que és enfocada al tractament d'àudio (no desplega entorn gràfic per treballar-hi) i només es pot accedir a treballar-hi mitjançant línia de comandes tal i com es veu a la il·lustració 72.

Aquesta distribució assegura 1 mil·lisegon de latència en els seus productes (també desenvolupa i ven *hardware* especialitzat), ja que utilitza programari propi fora dels divers habituals en altres distribucions Linux. També s'ha de dir que en cas de no utilitzar el seu hardware, la mateixa empresa Elk aconsella i de fet, ven part del seu programari per treballar amb targetes d'àudio HiFiBerry[37].

Per tant, es va descarregar la imatge d'ElkAudio OS i es va preparar una targeta SD per treballar-hi. La primera sorpresa fou que la imatge no esta optimitzada pel RPi 4 (tot i que al repositori del programari així ho indica) i per tant, es va executar sobre el RPI 2.

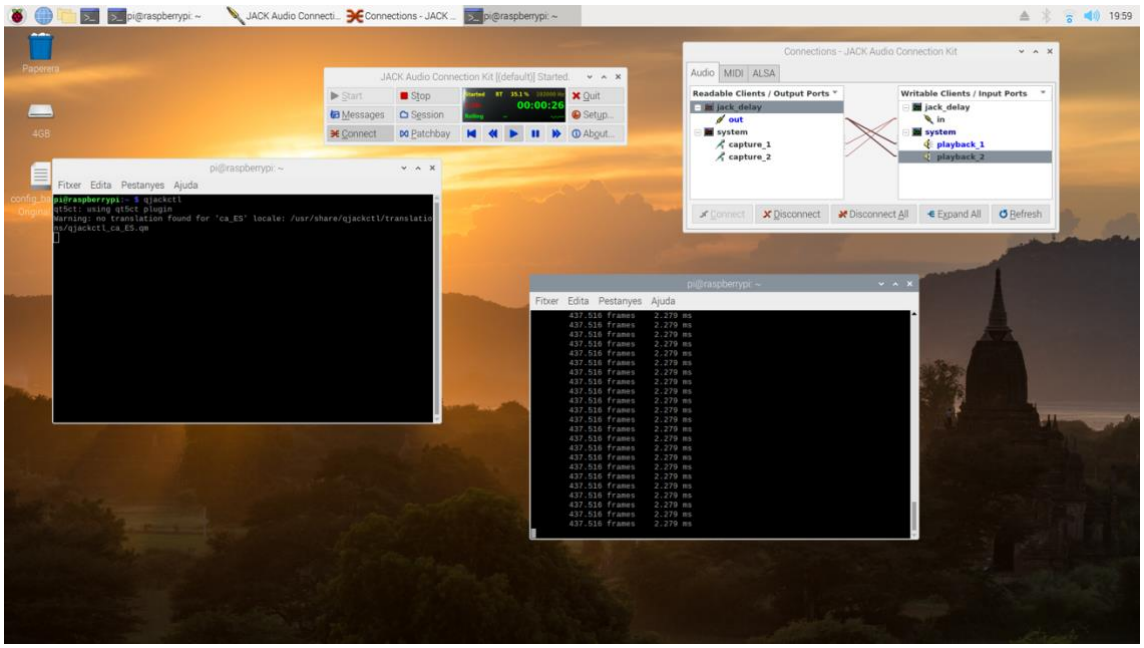
Il·lustració 73 - Captura de pantalla d'una sessió ssh sobre Elk Audio OS

Tot i saber que seria molt complicat (per no dir impossible), durant el procés d'investigació es va intentar instal·lar els complements de MathWorks per treballar amb aquesta distribució i el Simulink. Òbviament, va resultar impossible per la diferència d'arquitectura de kernel de la distribució. El procés, una vegada autoritzats tots els usuaris per modificar el sistema (Elk Audio OS no té usuari "sudo" explícit i s'han de assignar usuaris a grups concrets per tal de garantir-los permisos en el tractament de l'àudio), l'error obtingut un cop acabada la instal·lació és el que segueix:

Il·lustració 72 - Captura de pantalla de l'error en la gestió d'usuaris sobre Elk Audio OS

Per tant, per poder treballar amb el Simulink es va descartar aquesta possibilitat i es deixa plantejat com a una possible línia futura.

Per la seva banda el projecte *Pi-FX: A Raspberry Pi-Based Pedal Board[7]* es basa en Raspbian però modifica part dels divers amb la instal·lació del software de gestió d'àudio Jack [53] que permet reduir la latència.



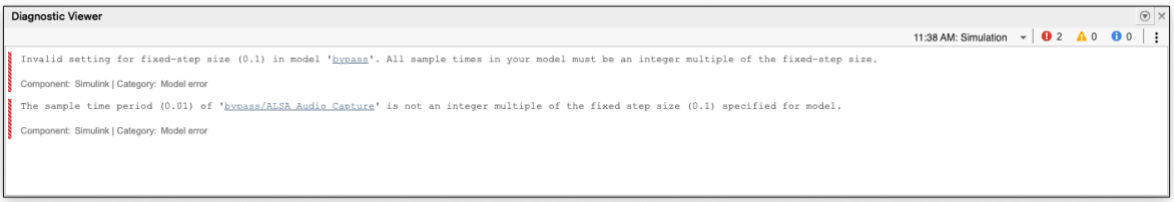
Il·lustració 74 - Mesura de latència dins al Raspbian amb el driver JACK

Una vegada instal·lat el paquet a la nostra distribució, el mateix programari permet fer el càlcul de la latència del sistema. En aquest cas el nostre sistema ens ofereix uns valors molt més acceptables que els que ens ofereix el sistema amb el nostre model.
Paral·lelament a això, val a dir que el sistema també es va optimitzar amb el que es consideren les instruccions canòniques pel desenvolupament de dispositius d'àudio sobre distribucions Linux[38].

Aquests passos són utilitzats en molts dels projectes pel tractament d'àudio sobre Linux, i, tot i necessitar d'ordres amb la línia de comandes, són prou fàcils d'aplicar.

Tot i això, la latència percebuda segueix sent la mateixa, sense canvis perceptibles amb les noves mesures.

Paral·lelament, podem fer el mateix que fan els programaris de gravació (*Digital Audio Workstation, DAW* en les seves sigles en anglès) i reduir el número de *samples per frame* (SpF, a partir d'ara). En aquest cas, el nostre problema és que necessitem que sigui un múltiple del *fixed-step* amb què estiguem treballant al *Solver*.



Il·lustració 75 - Captura de l'error generat amb els canvis de configuració

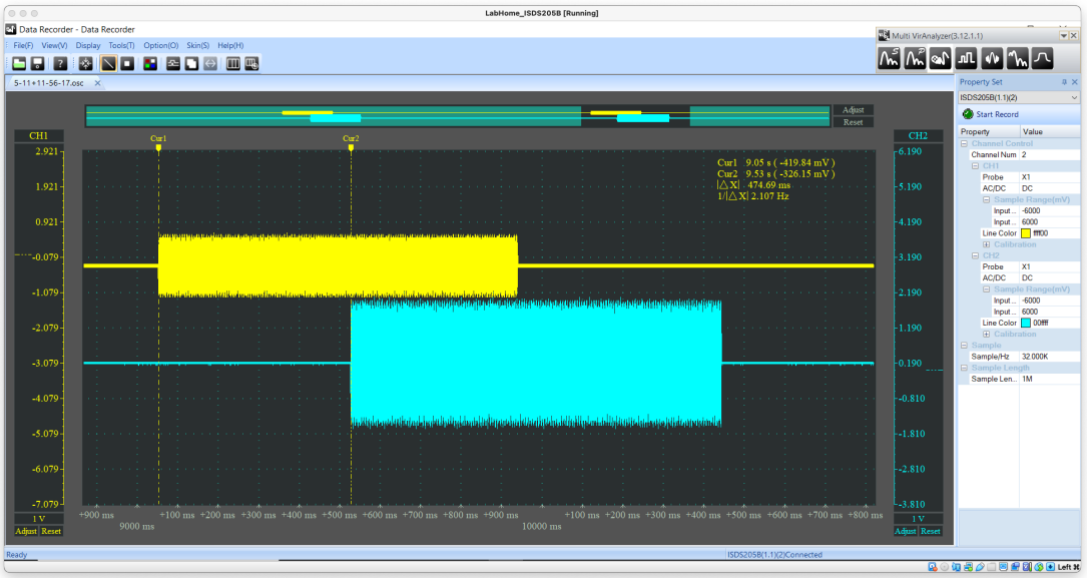
Recordar que estem mostrejant l'entrada a 44100Hz i que hem de complir, sempre amb valors enters, que:

$$\boldsymbol{Frequencia\ de\ mostreig} = \frac{\boldsymbol{N \cdot numero\ de\ samples}}{\boldsymbol{fixed - step}}$$

Equació 12 - Expressió per el càlcul de la freqüència de mostreig

Per tant, podem fer proves si amb altres configuracions tenim menys latència.

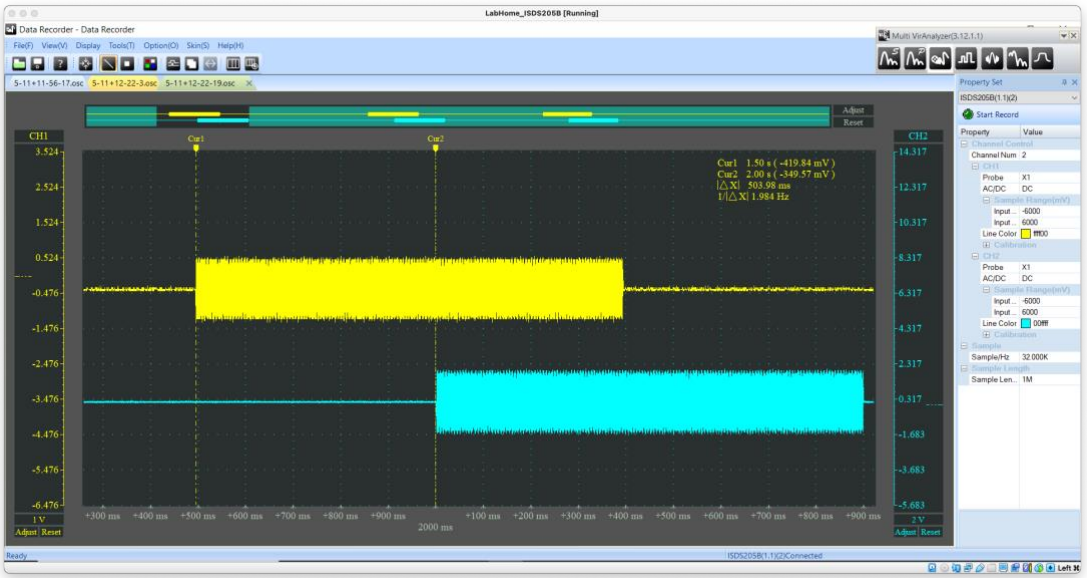
Amb *fixed-step* de 0.05 i SpF a 2205:



Il·lustració 76 - Mesura de latència amb fixed-step de 0.05 i SpF a 2205

Podem veure que reduïm en 30 mil·lisegons la latència del sistema. Per tant, podem provar si encara podem disminuir més tant el numero de *samples,* com el tamany del *fixed-step*:

Amb *fixed-step* de 0.01 i SpF a 441:

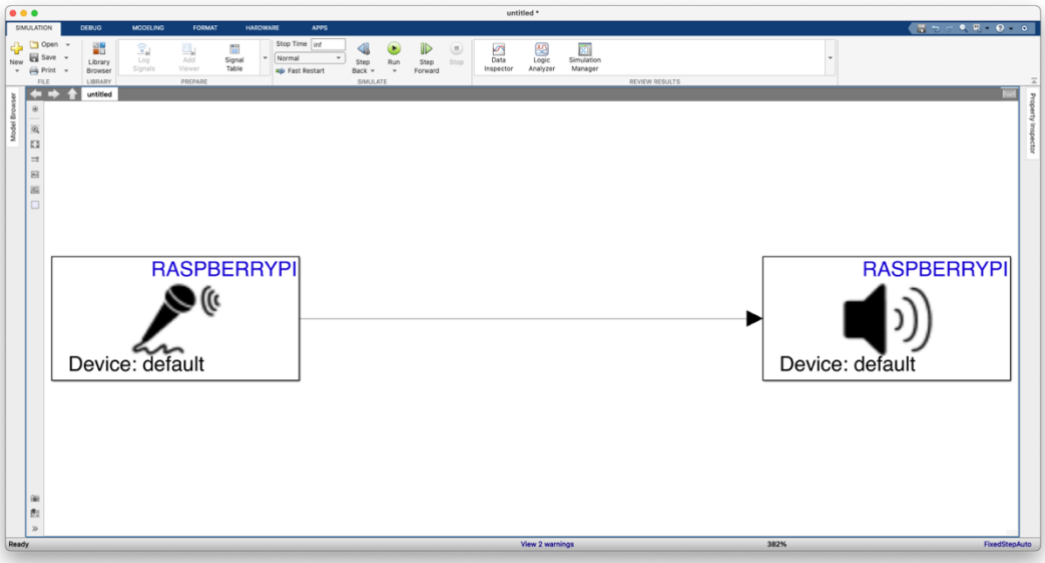


Il·lustració 77 - Mesura de latencia amb fixed-step de 0.01 i SpF a 441

Veiem que tornem a recuperar els 500 mil·lisegons i que, per tant, el sistema no és capaç de treballar amb menys latència seguint aquests ajustos.

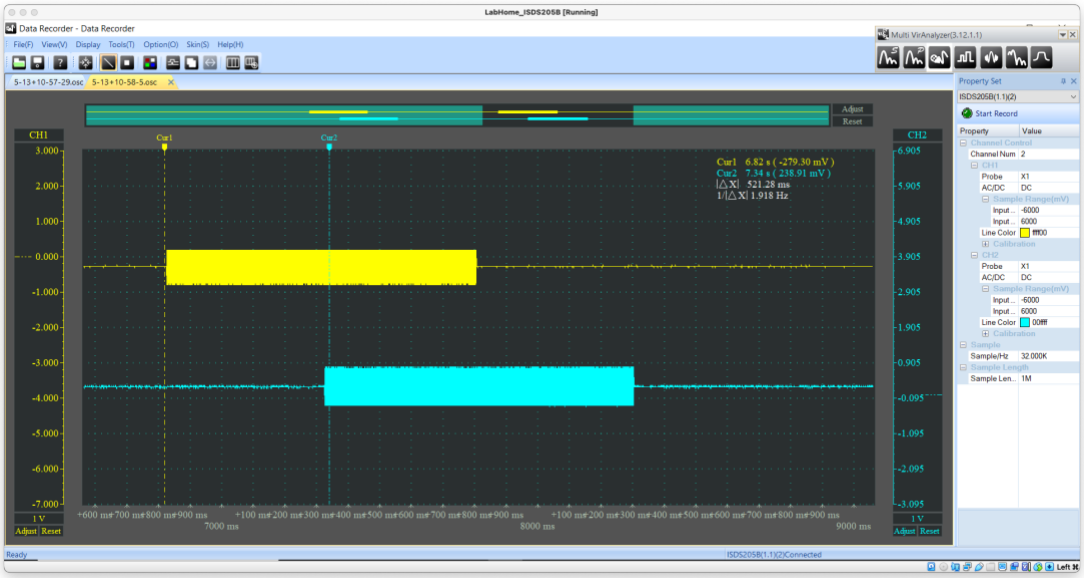
Posteriorment es realitza una prova reduint molt el tamany del *fixed-step,* fins a 1/44100 segons i treballant amb només 100 SpF. Amb aquesta configuració es produeixen talls en l'àudio que reprodueix el RPi 4. Aquests problemes es repeteixen amb un temps de *fixed-step* de 1/4410 segons.

Per acabar, es genera un model de Bypass amb el mínim de blocs possibles; es deixa configurat amb un *fixed-step* "automàtic" i amb 205 *samples per frame*. El model és el que segueix:



Il·lustració 78 - Captura de pantalla corresponent a la cadena més simple possible

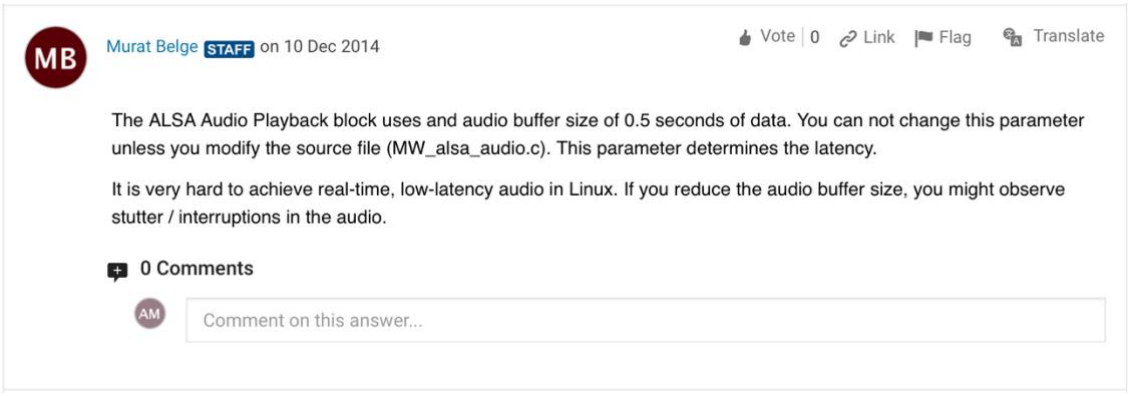
I la seva mesura de latència és la que apareix a la il·lustració 79:



Il·lustració 79 - Mesura de latencia per a la cadena més simple possible

Finalment i arribats en aquest punt, el problema sembla inherent al mateix programari de MathWorks i en com desenvolupa el codi un cop generat el model.

Seguint aquest fil s'arriba a una consulta[39] on es parla d'un arxiu anomenat MW_alsa_audio.c que afegeix 0.5s de buffer al nostre àudio.



Il·lustració 80 - Captura del fòrum on es parla del retard incorporat en codi

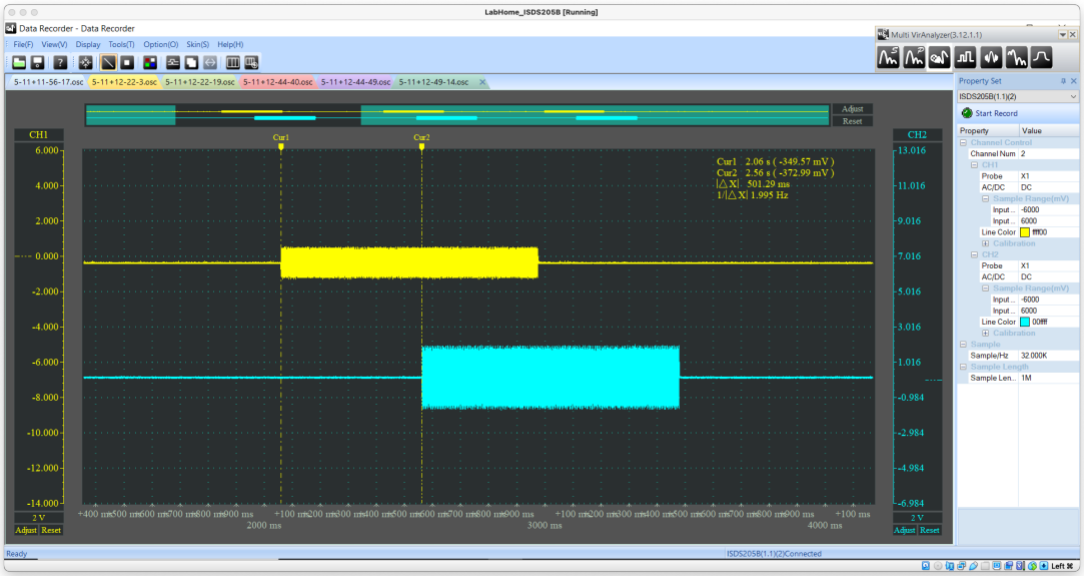
Concretament, aquesta resposta parla d'aquesta funció de l'arxiu amb aquest codi:

```
static void adjust_queueDuration(audioDeviceParams_t *devPtr)
{
   /* Make queueDuration an integer multiple of frameDuration */
   devPtr->queueDuration = ((int)(devPtr->queueDuration / devPtr->frameDuration + 0.5))
* devPtr->frameDuration;

   /* queuDuration should be at least four times frameDuration */
   devPtr->queueDuration = (devPtr->queueDuration < (QF_FACTOR*devPtr-
>frameDuration)) ?
      (QF_FACTOR*devPtr->frameDuration):devPtr->queueDuration;
}
```

Provem de baixar aquests temps de latència a veure si aconseguim baixar el temps mesurat.

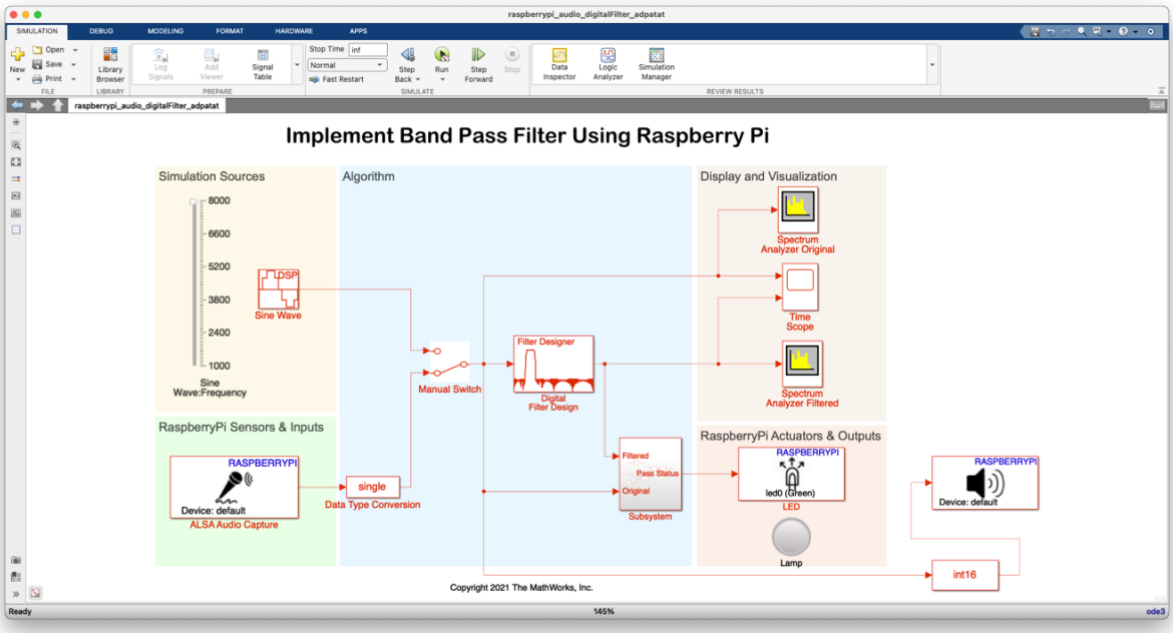
Una vegada modificada la funció amb un valor de 0.001 el temps de latència segueix sent l'habitual per aquest *fixed-step* i pel número de *samples*.



Il·lustració 81 - Mesura de latència amb la configuració de l'arxiu MW_alsa_audio.c canviada

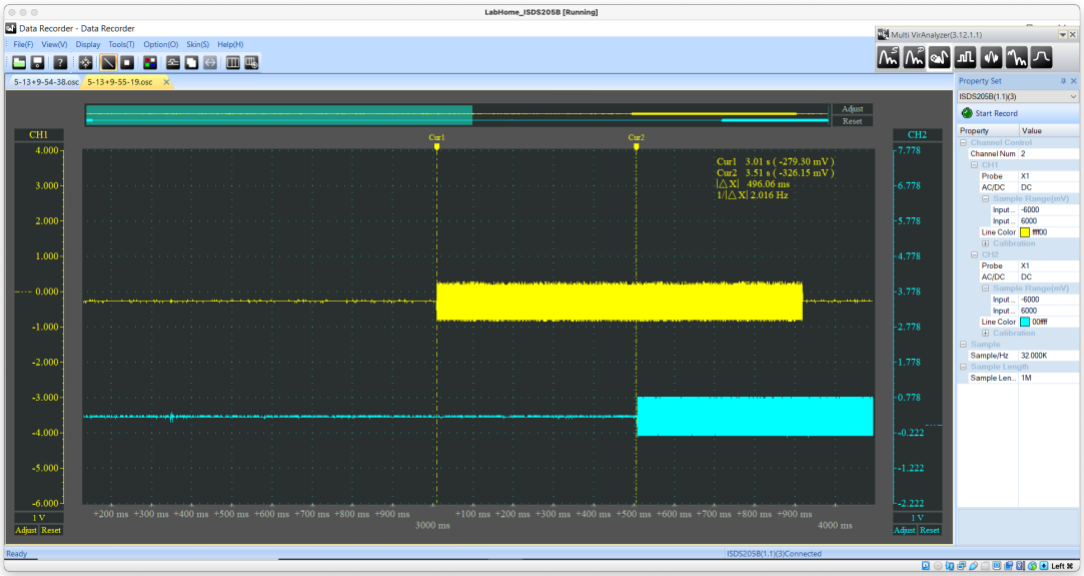
Per acabar les proves, modifiquem un dels exemples que ofereix la mateixa Mathworks per al tracament d'àudio amb RPi. En aquest cas, es tracta d'un exemple que implementa un filtre passa banda sobre un senyal capturat. L'exemple és consultable a la mateixa pàgina web de suport de MathWorks[40].

I les úniques modificacions que es realitzen són sobre la nomenclatura de la interfície d'entrada i que afegim el mòdul de reproducció. El model resultant és el que es mostra a la il·lustració 82:



Il·lustració 82 - Captura del model [40] d'exemple modificat

La latència mesurada amb aquest model en execució és la que es mostra a la il·lustració 83:



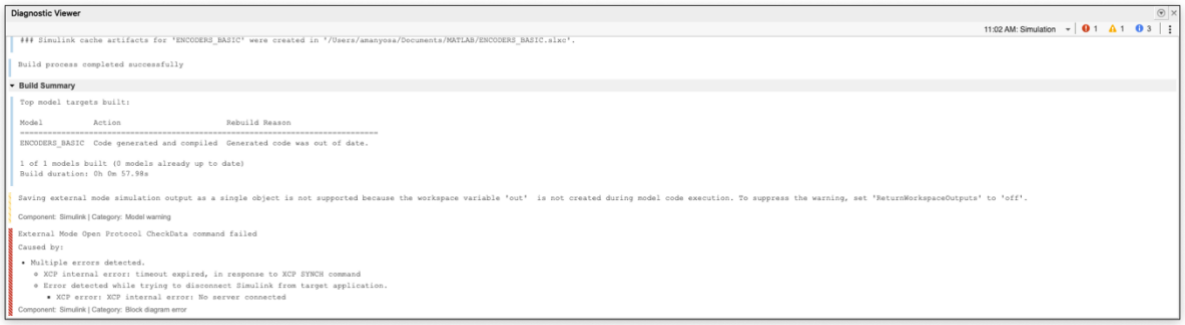
Il·lustració 83 - Mesura de latència variant el tamany de samples

Així doncs, després d'haver provat múltiples opcions per reduir la latència, assumim que és impossible treballar amb un model apte per a actuacions en viu. El modelat servirà això sí, per treballar a temps real (els efectes es processen en viu) però no amb prou rapidesa per poder-se integrar com un multiefectes per actuacions en viu.

3.4.4. XCP i GPIO

Una vegada muntats i comprovats els diferents potenciòmetres, al moment de voler-los fer funcionar simultàniament amb l'àudio, el Simulink és capaç de compilar i carregar el codi al RPi 4, però a banda de tenir uns talls i *bursts* en l'àudio, el sistema queda inoperatiu i el Simulink no aconsegueix comunicar-s'hi.

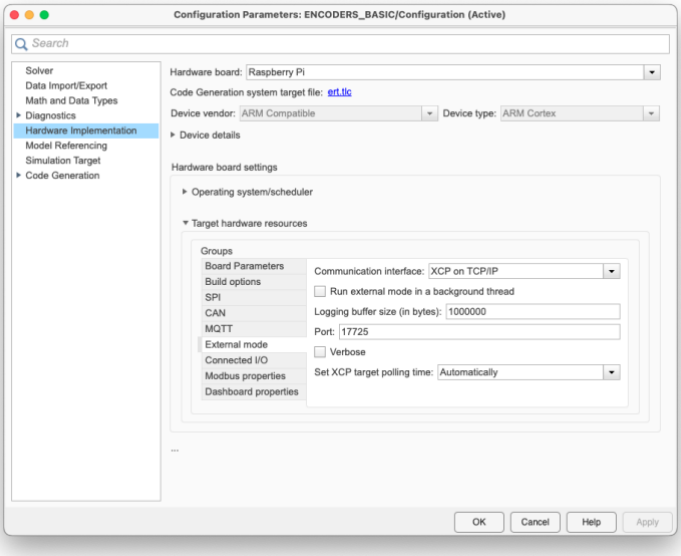
Aquest problema és derivat del protocol XCP, tal i com es mostra a la il·lustració següent:



Il·lustració 84 - Captura de l'error en la sincronia del protocol XCP

Investigant sobre el protocol [41][42] podem veure que es tracta d'un protocol destinat a la indústria automotriu per a la mesura i calibrat d'unitats de control electrònic. Tot i que d'entrada no sembli gaire adequat pel sistema, podem veure que el Simulink és un programa altament utilitzat en la indústria i que, a més a més, no es tracta d'una solució pròpia per part de la Rapsberry Pi Foundation sinó que és la mateixa MathWorks la que ho implementa com a mode de comunicació al bloc d'*Encoder* de la seva llibreria per treballar amb el Rpi 4.

El problema persisteix tot i intentar canviar el mode de funcionament de l'XCP (ja sigui via TCP/IP o sense) i en les diferents opcions que ofereix el Simulink. En la figura següent es poden veure totes les opcions de configuració que permet el programari i que s'han anat variant seguint diferents articles consultats[42] [44] [45] al llarg del disseny.

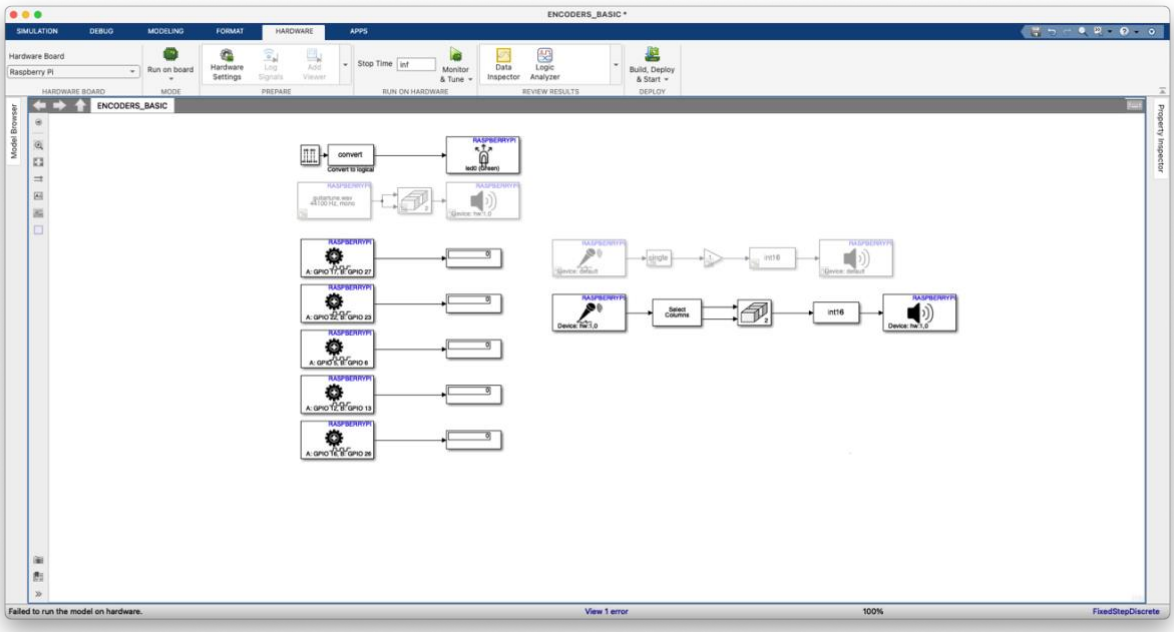


Il·lustració 85 - Captura de la pantalla de configuració del protocol XCP dins el Simulink

En aquest aspecte podem deixar constància que també s'ha comentat a una consulta ja oberta a la comunitat d'usuaris de MathWorks on un altre usuari exposa un problema similar[46]. A dia d'escriure aquesta memòria (17 de maig) seguim esperant resposta - tot i que un usuari indica que el fet de no treballar amb un desenvolupament considerat a temps real pot ser l'arrel del problema.

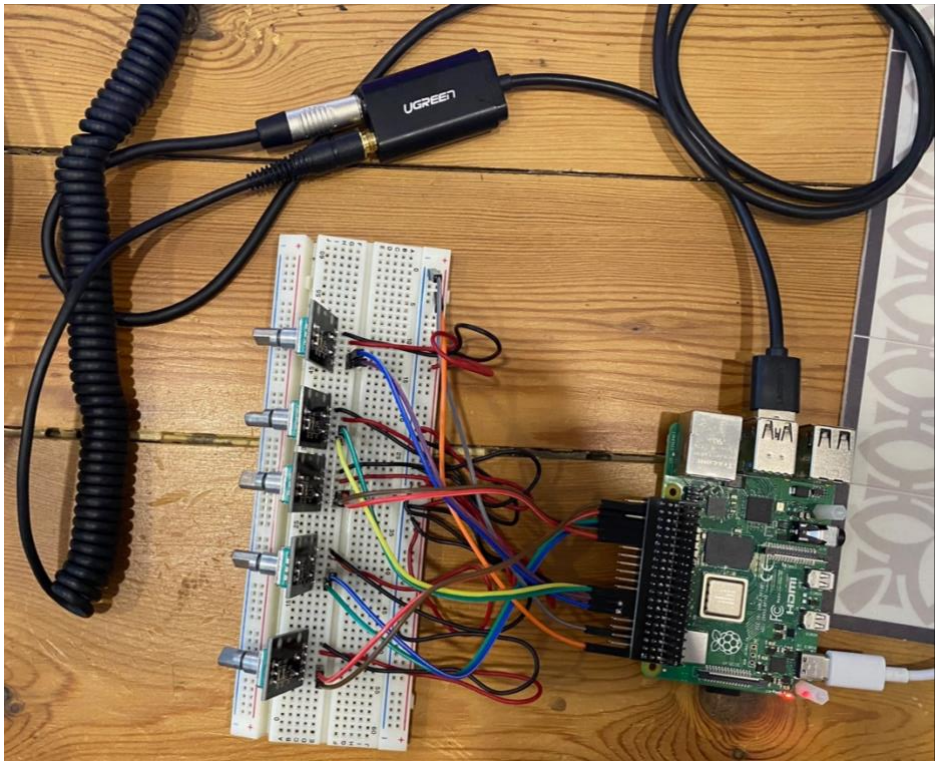
També als fòrums de suport de la targeta d'àudio s'ha obert una incidència[47]. Recordem que també funciona via GPIO i per tant, modifica com es transmeten els senyals via els pins i pot fer modificar com el RPI 4 recull o envia el senyals de tots els pins, no només els destinats a la targeta.

Assumint que els problemes són provocats per compartir el GPIO entre àudio i *encoders* plantegem un model per treballar amb GPIO i la targeta USB. El model és el que segueix:



Il·lustració 86 - Captura del model de cadena amb GPIO i targeta USB

En aquesta il·lustració 86, s'hi veuen diferents cadenes d'àudio (algunes comentades) i els 5 potenciòmetres amb el mateix esquema que teníem anteriorment. El muntatge físic és el que segueix a la il·lustració 87:



Il·lustració 87 - Muntatge del prototip amb potenciòmetres GPIO i targeta USB

Però tot i aquest muntatge i les diverses opcions pel tractament de l'àudio, no s'aconsegueix trobar una cadena d'àudio que funcioni correctament i de forma simultània amb els potenciòmetres.

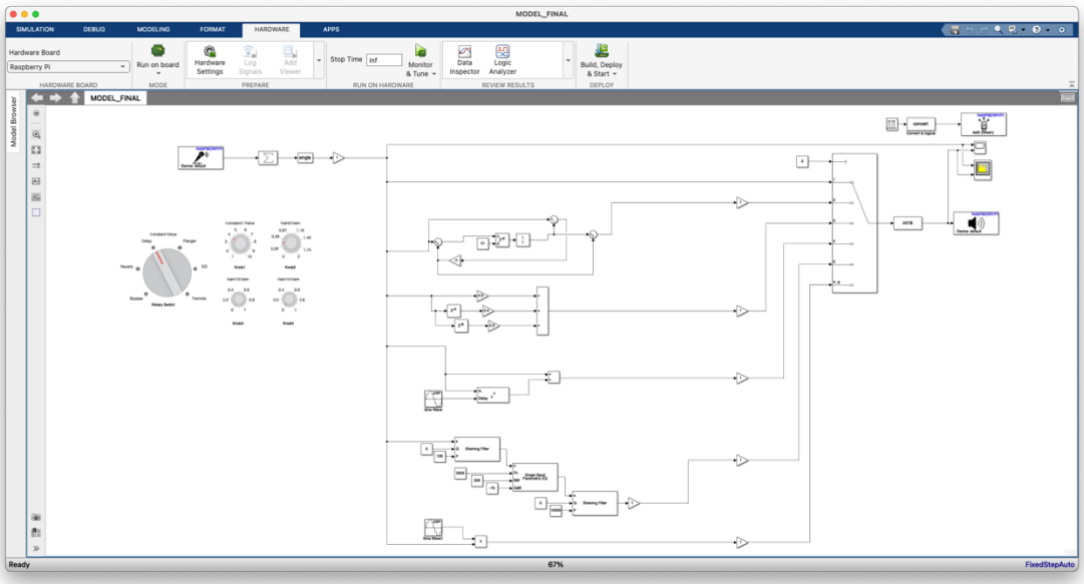
Finalment però, i encara a l'espera de possibles respostes, es desenvolupa un model sense cap mena de hardware extern de control i utilitzant la targeta d'àudio via GPIO i on tots els paràmetres es controlen via el *Dashboard* del mateix ordinador de desenvolupament.

3.4.5. El model final

Una vegada modelats els diversos efectes que integra el multiefectes i analitzats tots els problemes i entrebancs que han aparegut, es planteja com ha de ser la construcció del model final pel seu *deploy* al RPi 4.

A l'inici del projecte es pensava que el model podia seguir una estructura "en paral·lel" de forma que, mitjançant un selector vinculat a un potenciòmetre hardware es pogués escollir l'efecte a aplicar. Aquest model es concreta en aquest model del Simulink generat amb els models d'efectes que s'han anat comentant.

En aquest model final podem veure els models presentats (amb alguna variació que s'analitza tot seguit) i dues parts completament noves que es poden veure a la il·lustració 88. La primera part, a l'esquerra de la il·lustració, el sistema de control que substitueix els potenciòmetres hardware. La segona part, al final de la cadena de reproducció s'ha afegit un analitzador d'espectre i un *Time Scope* per veure la forma d'ona que genera el model. Aquests blocs nous es poden observar a la il·lustració 88.



Il·lustració 88 - Model final amb 4 potenciòmetres

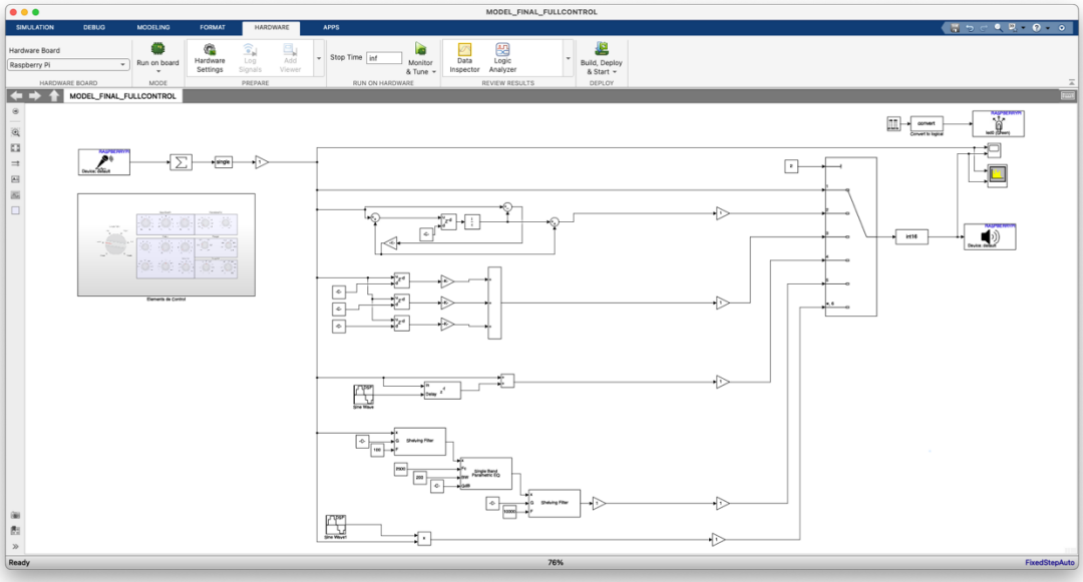
L'enèsim problema apareix quan els 4 potenciòmetres modificadors s'han de vincular amb diferents paràmetres de diferents efectes. Tant el bloc *Encoder* com els blocs que s'utilitzen per la interfície, els *Rotary Knob*, només poden modificar un tipus de paràmetre a la vegada. Per posar un exemple, al modificar una constant, aquesta pot ser compartida entre diversos blocs però no pot modificar dos valors de blocs de guany diferents simultàniament.

Aquesta limitació té la seva lògica ja que, al cap i a la fi, el model genera un codi que no sap discernir quin efecte està actuant a la sortida; tots estan funcionant paral·lelament i és el bloc selector el que fa d'interruptor per connectar una sortida o l'altra.

Arribats a aquest punt es tornen a plantejar dues solucions possibles; o fer la implementació final amb un model *if-case*, és a dir, diferents blocs que s'executin en funció d'una condició (el valor del selector principal) i intentar que els potenciòmetres siguin compartits o directament; incloure en el model tants potenciòmetres com paràmetres es vulguin modificar.

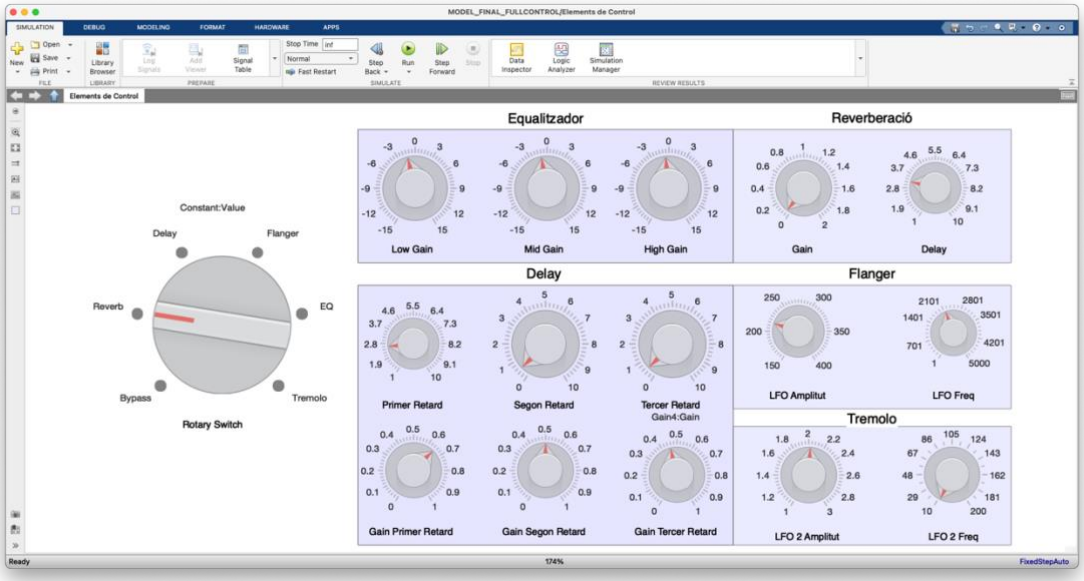
Finalment (i per manca de temps) es realitza la segona opció, és a dir, la implementació amb tots els paràmetres a la vista. Assumint que ja no es pot treballar amb hardware extern per les limitacions ja comentades, generem tants potenciòmetres o elements de control gràfics com paràmetres es vulguin controlar.

Així, el model final de processament en paral·lel i tots els paràmetres de control a vista queda com apareix a la il·lustració 89.



Il·lustració 89 - Model final amb tots els paràmetres controlables

A la il·lustració 89, podem veure els paràmetres de control que es construeixen amb un subsistema anomenat *dashboard* que es mostra a la il·lustració 90. Val a dir que tots els paràmetres són modificables en execució però que per la realització d'aquesta memòria s'han configurat amb uns paràmetres concrets per a poder dur a terme les captures de mostra.

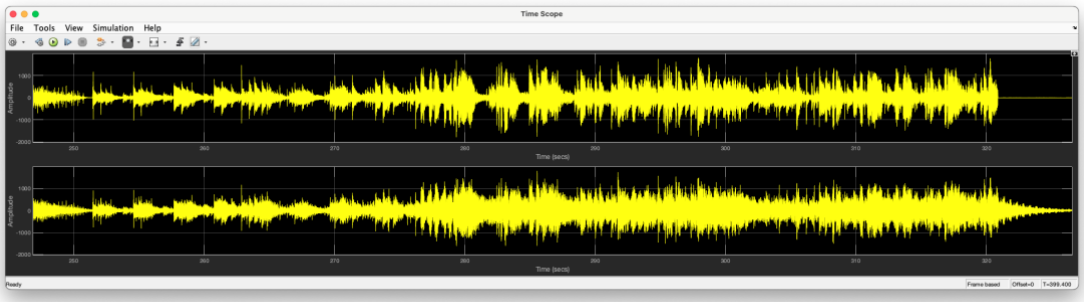


Il·lustració 90 - Control del multiefectes dins el Simulink

Analitzant els efectes finals en detall, podem veure que a la reverberació s'ha variat el bloc de retard per controlar-lo mitjançant una constant. El valor d'aquesta constant es controla mitjançant un potenciòmetre configurat amb valors entre 0 i 10. A més a més, hem afegit un bloc *Unit Delay* a la sortida del bloc retardador per evitar que es produeixi un llaç algebraic tancat infinit en funció del valor que prengui aquesta primera variable. Configurem els paràmetres de control com apareix a la il·lustració 91.

Il·lustració 91 - Paràmetres de la reverberació aplicada

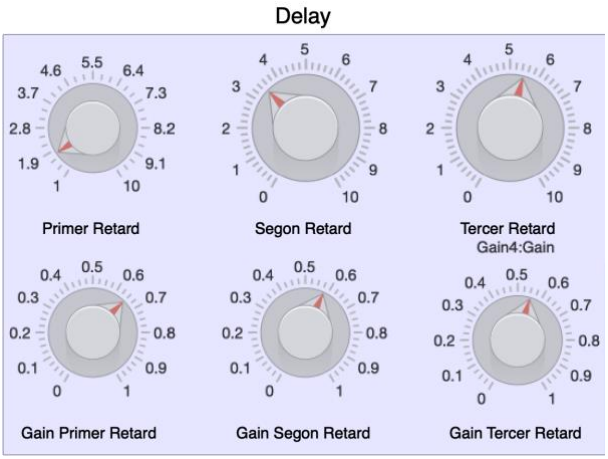
Si apliquem l'efecte sobre un arxiu d'àudio reproduït des de l'ordinador de desenvolupador obtenim el resultat mostrat a la il·lustració 92.



Il·lustració 92 - Comparativa entre senyals aplicant la reverberació

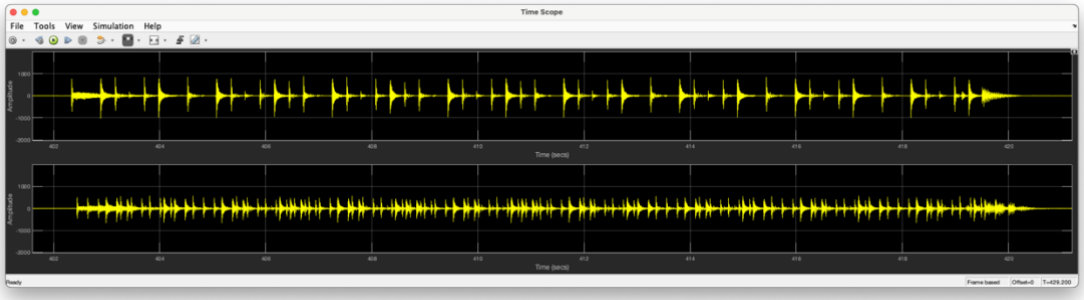
Podem veure, en aquesta il·lustració 92, el senyal original a la part superior i el senyal processat a la inferior i observem que els transistors queden reduïts i una cua causada per la reverberació al final de la reproducció.

A l'efecte del *Delay* s'han afegit tres constants que controlen quan s'executen els retards. Aquestes constants també es configuren amb valors entre 0 i 10. A més a més, els tres blocs de guany de la sortida permeten variar el seu valor entre 0 i 1 i deixen la mescla de l'efecte en mans de l'usuari. En el nostre cas, configurem els paràmetres del delay com apareix a la il·lustració 93:



Il·lustració 93 - Paràmetres del delay aplicat

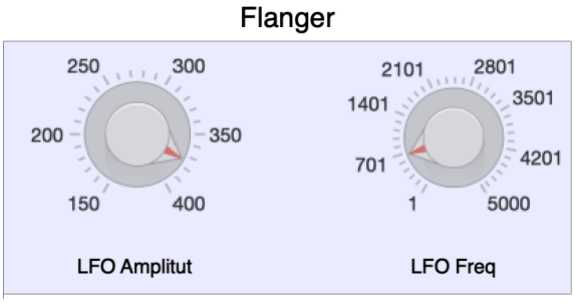
I la sortida comparada davant d'un àudio de sons percutius és la que es mostra a la il·lustració 94:



Il·lustració 94 - Comparativa entre senyals aplicant el delay

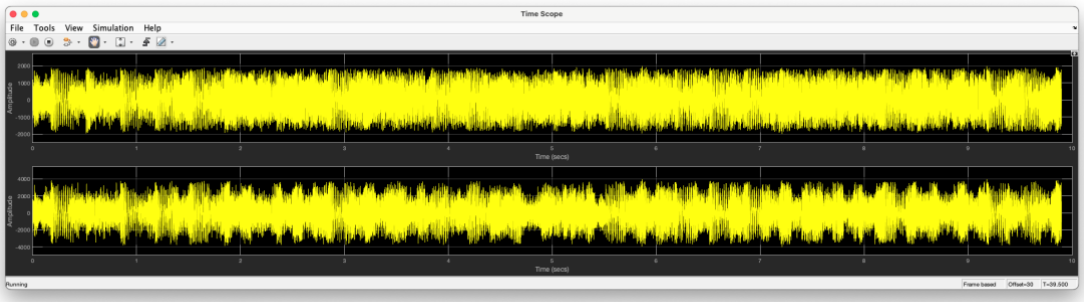
En els efectes de *Flanger* i de *Tremolo* es controlen els paràmetres d'amplitud i de freqüència del LFO. Els valors d'amplitud van des de 150 a 400 en el cas del *Flanger* i de 1 a 5 en el cas del Tremolo. L'espectre freqüencial es situa entre 10 i 5kHz, tot i saber que els efectes es noten més quan estem treballant en freqüències típiques de LFO.

La configuració utilitzada pel *Flanger* es mostra a la il·lustració 95.



Il·lustració 95 - Paràmetres del flanger aplicat

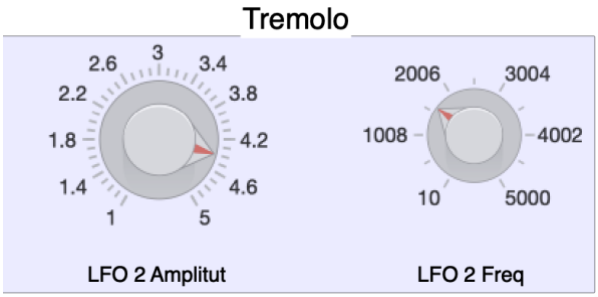
I la seva sortida comparada es mostra a la il·lustració 96.



Il·lustració 96 - Comparativa entre senyals aplicant el flanger

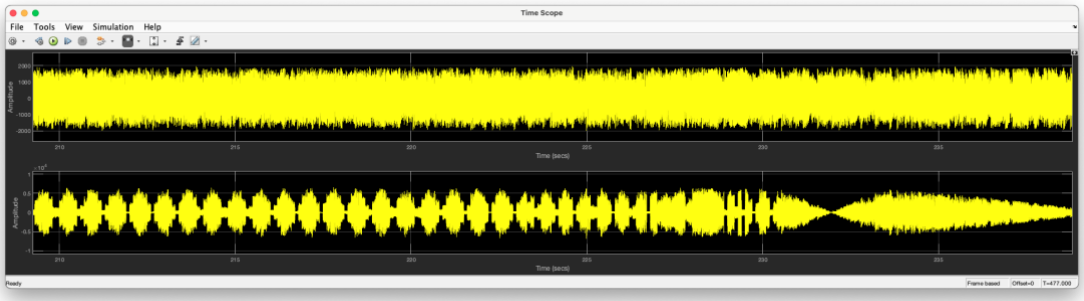
En aquesta il·lustració 96, ja és difícil percebre'n l'efecte de forma visual. Veiem que el senyal inferior adquireix una forma semblant a una sinusoide que, en l'espectre freqüencial ens originarà el to metàl·lic típic d'aquest tipus d'efecte.

La configuració utilitzada pel *tremolo* es detalla a la il·lustració 97.



Il·lustració 97 - Paràmetres del tremolo aplicat

I la seva sortida comparada es pot veure a la il·lustració 98.

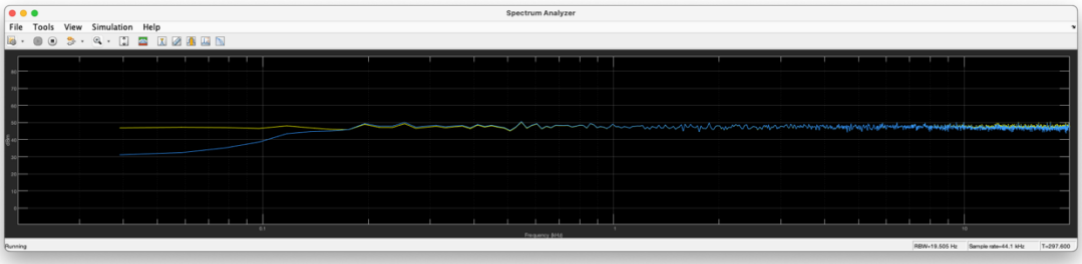


Il·lustració 98 - Comparativa entre senyals aplicant el tremolo

En aquesta il·lustració 98, veiem com l'efecte, amb valors més extrems i variant temporalment, modula l'amplitud del senyal original.

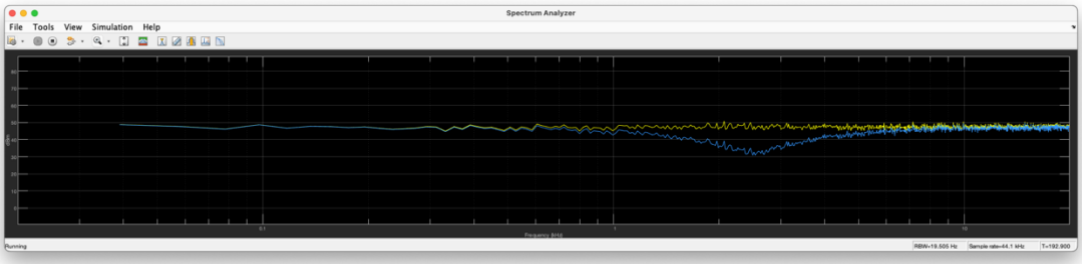
Finalment, al bloc equalitzador es controla amb només tres potenciòmetres (com el del model real amb que s'inspira [30]) que controlen el guany de cada filtre plantejat. Les il·lustracions 99, 100 i 101 corresponen a les tres bandes d'actuació. En aquest cas, el senyal d'excitació és un soroll rosa generat des de l'ordinador de desenvolupament:

Amb una atenuació de -15 dBs a la banda *Low*, en podem veure l'efecte a la il·lustració 99.



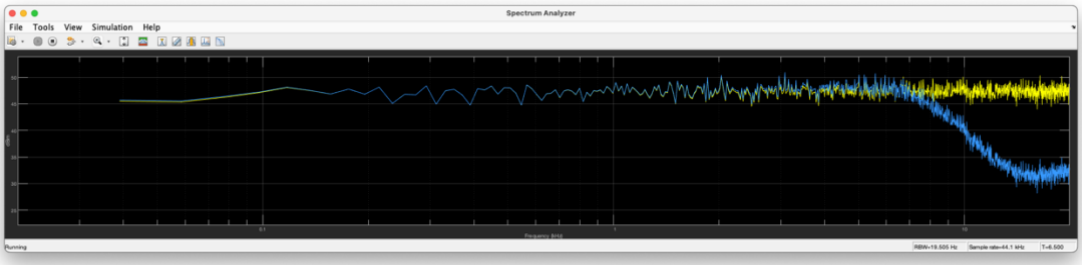
Il·lustració 99 - Comparativa freqüencial entre senyals atenuant la banda Low

Amb una atenuació de -15 dBs a la banda *Mid*, en podem veure l'efecte a la il·lustració 100:



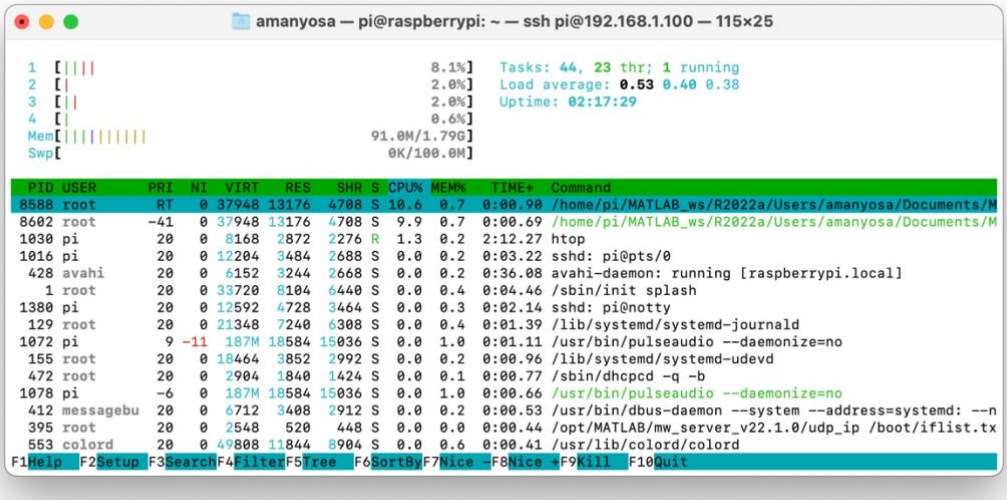
Il·lustració 100- Comparativa freqüencial entre senyals atenuant la banda Mid

Amb una atenuació de -15 dBs a la banda *High*, , en podem veure l'efecte a la il·lustració 101:



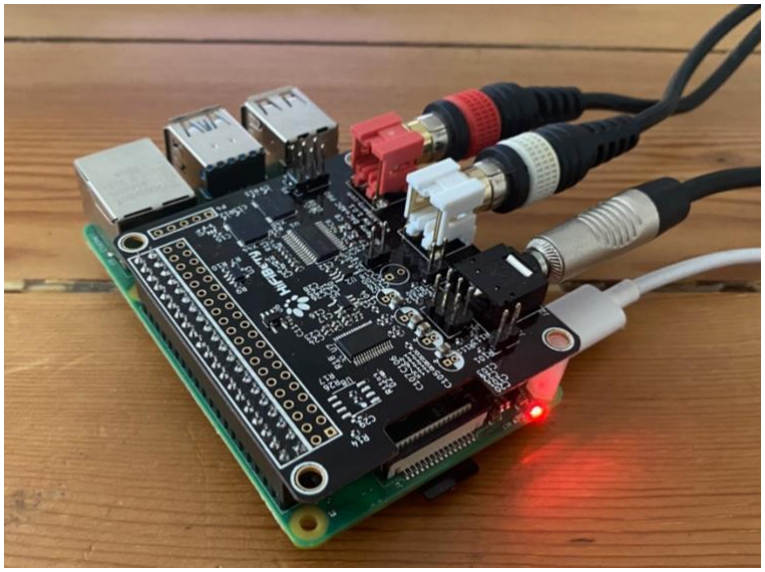
Il·lustració 101 - Comparativa freqüencial entre senyals atenuant la banda High

A nivell de hardware, podem veure l'ús que fa el model generat sobre el hardware del RPI 4. Mitjançant la comanda *htop* sobre una sessió de SSH, podem veure que encara tenim marge per codificar alguns efectes més, ja que tenim capacitat de processat de sobres. El nostre model (amb 5 efectes) només ocupa un 10% de la capacitat de processadors del total del RPI 4, tal i com es pot veure a la il·lustració 102.



Il·lustració 102 - Captura de pantalla d'una sessió ssh amb la comanda htop

Per acabar aquest apartat del model final dissenyat, remarcar que l'últim model de hardware no disposa dels potenciòmetres ni de *l'splitter* de GPIO ja que han estat substituïts per el control via *dashboard*. Per tant, el hardware final és el que es mostra a la il·lustració 103.



Il·lustració 103 - Hardware per el model final

El model final (i tots els modelats del efectes comentats) s'adjunten a l'entrega d'aquesta memòria. La llista completa apareix a l'Annex.

4. Conclusions i línies futures

4.1. Conclusions

La primera conclusió, i la més evident, és que tot i la gran potència de la plataforma Simulink com a generadora de models també ha estat el gran impediment per aconseguir el producte imaginat.

Els problemes amb la latència han sigut crucials i fonamentals durant tot el procés. De fet, hi ha hagut més hores de dedicació en els càlculs de la latència que en la construcció d'algun dels efectes plantejats.

La dependència que mostra el Simulink de les llibreries de Raspbian no ens permeten utilitzar altres distribucions de Linux que permetrien assolir una latència acceptable per a l'ús de la plataforma com a generadora d'efectes per a actuacions en viu.

La configuració creuada entre el *Solver*, el temps de sampleig i el nombre de frames per sample ha sigut complicada en tot moment i una vegada vist que la latència es situava sempre en valors similars, la decisió va ser assumir que no es podien reduir aquests 500 mil·lisegons amb les eines que teníem disponibles.

Tot i això, els efectes construïts i modelats són efectes perfectament vàlids en altres implementacions, són un bon exemple de la capacitat del programari per poder manipular àudio de forma fàcil i sense haver d'escriure codi. La facilitat de l'entorn gràfic és ideal per poder esbossar esquemes ràpids i manipular camins de senyal i conversions sense gaire preocupacions. A més a més, la utilització del Simulink fa que els efectes i els seus modelats siguin independents al hardware escollit. En aquest sentit, els models són traslladables per exemple, a una placa FPGA.

Per tant, queda clar que l'objectiu principal plantejat no s'ha aconseguit. El producte "semi-professional" no s'ha assolit amb les característiques que s'havien plantejat inicialment. Primer pels problemes amb la latència ja comentats i en bona part, també per la impossibilitat d'aconseguir un hardware compatible amb el sistema plantejat.

Aquest hardware, d'implementació electrònica molt senzilla, no s'ha pogut fer funcionar correctament també pel desconeixement de com realment funciona el programari escollit. El Simulink ofereix molts modes de funcionament (en local, amb connectivitat externa, etc) i el que asseguràvem per algun costat, ens ho retallava el canvi d'especificacions. Sabem que el programari pot gestionar el protocol XCP i ens ofereix blocs per fer-ho, però una altra vegada altres decisions de disseny (en aquest cas treballar amb àudio) generen conflictes a un nivell que no hem pogut assolir amb el temps de què s'ha disposat.

També és necessari parlar de la planificació pressupostària inicial ja que s'ha variat substancialment. Inicialment s'havia pressupostat un total de 82€ per tot el projecte que però es van disparar amb la necessitat de comprar una targeta d'àudio externa de més

qualitat i amb la necessitat de l'*splitter* de GPIO. En total, el pressupost es va incrementar un 50% i va arribar als 123€.

4.2. Línies futures

Després del temps de desenvolupament i de la quantitat de dificultats que hi ha hagut durant el modelatge del producte, han quedat algunes línies per poder seguir la investigació. La més evident és la de la gestió de la latència.

Des de l'inici del plantejament del projecte es va creure que tant el RPi 4 com el programari serien capaços d'aconseguir treballar amb un nivell de latència acceptable. Com ja s'ha comentat, l'opció de treballar a temps real no significa necessàriament treballar amb latència mínima, i en tractaments d'àudio en viu és crucial. És obvi que el hardware no té ni un DSP dedicat i potser només incorporant una targeta d'àudio que sí que en tingui ja s'haurien reduït aquests temps. Seria interessant, doncs, fer una comparativa amb diferents targetes de diferents fabricants per a veure si és possible millorar en aquest aspecte.

Seguint amb la latència, la part d'optimització de programari també és interessant. Al final, la distribució de Raspbian és una en concret però es pot adaptar i modificar fins i tot a nivell de *kernel*. Això queda fora de l'abast d'aquest TFG, però una línia possible seria adaptar aquestes llibreries que són imprescindibles per fer funcionar el Simulink a altres distribucions de Linux (per exemple Elk Audio OS).

Una altra línia d'investigació és la del protocol XCP i què necessita i com es pot configurar per no interferir en el tractament de l'àudio del RPi 4. D'aquesta manera es podria aconseguir fer funcionar el hardware inicial i, per exemple, dissenyar una carcassa per allotjar tot el sistema.

Una altra línia futura seria per dotar el producte d'un acabat més professional; es podria treballar en l'electrònica de l'entrada d'àudio per treballar a nivells de micròfon i no només a nivells de línia.

Per acabar, una vegada consolidades totes les línies anteriors, es podria dur a terme el que era un dels objectius inicials del TFG; la construcció d'una carcassa per encabir tot el sistema mitjançant una impressora 3D i el disseny del contenidor que ens encabís el RPI 4, l'splitter del GPIO, la targeta finalment escollida i les entrades i sortides d'àudio pertinents.

5. Glossari

***Add-on*:** complement del programari. No són imprescindibles pel correcte funcionament del programari bàsic i serveixen per expandir-ne les capacitats. En el nostre cas, els complements per comunicar-nos amb el RPI n'és un exemple evident. No tots els usuaris de Simulink el necessiten i per tant, es pot afegir a posteriori.

Analitzador d'espectre: Equip de mesura electrònic que permet veure en una pantalla l'espectre de freqüències d'un senyal. En el nostre cas, acotat entre 20 i 20kHz, s'implementa sobre un bloc del Simulink. A l'eix d'ordenades presenta el nivell de magnitud (normalment en decibels) i a l'eix d'abscisses mostra la freqüència.

Arduino: plataforma de desenvolupament basada en microcontroladors [4].

Bursts: en argot col·loquial i traduït com a ràfega, es tracta d'una ràfega molt curta però amb molta energia. En el nostre cas, acostuma a produir també àudio molt distorsionat degut a l'increment d'energia dins el marge dinàmic de l'aparell en qüestió.

Bypass: mode de funcionament on el senyal passa pel nostre sistema però no se li aplica cap efecte. Podem dir que si està en bypass, evita qualsevol manipulació (més enllà de la conversió analògica-digital).

Codificació: procés pel qual un senyal quantificat adquireix uns valors de zeros i uns per a treballar-la en domini digital.

Dashboard: podem traduir-ho com a tauler de control. En el nostre cas, és amb el què manipularem els paràmetres dels efectes per substituir els potenciòmetres.

DAW: *Digital Audio Workstation*. Categoria de programari. Els programes de tractament d'àudio professional que ens serveixen per a gravar, editar, mesclar i manipular senyals d'àudio.

Decibel (dB): unitat de mesura per a mesurar la intensitat sonora. Es tracta d'una relació logarítmica entre la intensitat mesurada i un valor de referència.

Delay: retard. Un efecte de *delay* és aquell que ens endarrereix el senyal que estem manipulant un temps o número de mostres determinat.

***Deploy*:** traduït per a desplegar. En argot, *fer un deploy* es entregar el programari a un maquinari per tal que l'executi.

Digital Filter Design: Eina del Simulink per a la creació de filtres digitals. [28]

DIY: *Do It Yourself*. Acrònim per a definir el moviment "fes-ho tu mateix".

DSP: *Digital Signal Processor.* Microprocessador destinat i optimitzat principalment al processament de senyals.

Encoder: podem parlar *d'encoder* referint-nos al bloc Encoder del Simulink que és el model que ens permet comunicar els potenciòmetres físics i els models creats o per altra banda, podem parlar d'un *Encoder* quan ens referim a la peça hardware també anomenada potenciòmetre.

Equalitzador: processador del senyal d'àudio que ens permet variar-ne el contingut freqüencial.

Estereofònic: en anglès, *stereo.* Mode de captació i gravació del so que utilitza dos canals en tot el procés.

Fixed-step: mode de funcionament del *Solver* del Simulink on especifiquem un temps fix per a fer els càlculs del model generat. Aquest temps, serà clau pel desenvolupament del model. Temps baixos impliquen càlculs més acurats, però també incrementen el temps de computació. És el contrari del *Variable Step* i és imprescindible per a fer funcionar els potenciòmetres i el RPi.

Flanger: tipus d'efecte que es caracteritza per atorgar al senyal original un to metàl·lic a freqüències mitges i agudes.

FPGA: *"Field Programable Gate Array".* És una arquitectura on el dispositiu també és programable [9].

frame: la traducció literal és "marc", però en el nostre àmbit parlem de bloc de dades. Un *frame* és un conjunt de dades seqüencials originades en un canal en concret.

Freqüència de mostreig: és el nombre de mostres per unitat de temps que prenem d'un senyal analògic per digitalitzar-lo. En el nostre cas, la freqüència de mostreig ha de complir amb el Teorema de Nyquist.

GETiST: acrònim del Grau d'Enginyeria de Tecnologies i Serveis de Telecomunicació de la UOC.

GND: marca per indicar els punts de connexió a terra dels diferents aparells.

GPIO: *General Purpose Input Output*. Interfície per la connexió entre aparells. En el cas del RPi són dues línies de 20 pins cadascuna. Poden transmetre informació, voltatge o servir de pressa de terra en funció de la configuració.

HDMI: protocol de transmissió de vídeo. El micro-HDMi és el mateix amb un connector més reduït.

Header: interfície de connexió basada en pins. En el nostre cas, el *header* és de GPIO.

IDE: *Integrated Development Enviroment.* És un paquet de programari que inclou un entorn de programació complert. Acostuma a disposar d'un editor, un compilador i un programari per solucionar errors.

if-case: expressió condicional que evalua una condició (*if*) i en cas que es compleixi, executarà un *case* (una acció) concret. Si tenim diverses condicions inicials, podem tenir diverses execucions en funció del resultat del primer condicional.

Jumper: peça de plàstic que, gràcies a la seva posició, determina algun paràmetre del hardware. En el nostre cas, en funció de la posició que tingui, la targeta HIFiBerry té un guany d'entrada diferent.

Kernel: És el nucli del sistema operatiu, per tant, com es comunica el programari amb el maquinari. Les versions de Linux poden tenir *kernels* modificats i així aconseguir un comportament més adequat per les nostres necessitats.

Latència: és la diferència temporal entre la captació i la reproducció de l'àudio en un mateix sistema. Sempre tenim latència, en major o menor mesura.

LFO: *Low Frequency Oscilator:* Oscil·lador de baixa freqüència. Es tracta d'un generador d'ones, normalment sinusoidal que funciona a baixes freqüències (entre 20 o 100Hz). En el nostre cas els utilitzem per a modificar algun paràmetre dels efectes.

Linux: o GNU/Linux. És un sistema operatiu de tipus UNIX compost de software lliure i de codi obert.

Llibreries: en el nostre cas, són els arxius que ens instal·laran els *add-on* del MATLAB o el Simulink per a comunicar-nos amb el RPi. Al cap i a la fi, són paquets d'arxius que apliquen configuracions i opcions.

Mac OSX: sistemes operatius desenvolupats per Apple pels seus ordinadors. Basat en UNIX, és un sistema propietari.

MathWorks: companyia desenvolupadora de MATLAB i Simulink [17].

MATLAB: paquet de programari desenvolupat per MathWorks [17].

Monofònic: so captat i reproduït utilitzant solament un canal.

Mostra: en el nostre àmbit, punts d'un senyal continu que formaran el senyal discret resultat de mostrejar-lo. Com més mostres tinguem, més precisió obtenim respecte els dos senyals.

Mostratge: procés de mostrejar el senyal analògic de forma periòdica per a poder-lo discretitzar.

Mostres per *frame:* número de mostres que conté cadascun dels nostres *frames* a processar. Com més mostres tinguem, més temps necessitem per l'anàlisi i processat. També hem d'assumir que si treballem amb senyals estereofòniques, tindrem el doble de mostres per *frames,* ja que treballarem amb dos canals.

Multi efectes: aparell capaç d'aplicar un o més efectes d'àudio sobre un senyal capturat i entregar el senyal resultant a la sortida.

Phaser: Tipus d'efecte caracteritzat per combinar un senyal original amb una còpia del mateix al qual s'ha modificat lleugerament la fase. La modificació de la fase es realitza mitjançant un LFO.

Pinout: comanda típica dels sistemes Raspbian per a saber quina configuració ofereix el nostre RPi en execució.

Pins: cadascun dels punts de connexió d'una interfície. En el cas concret del GPIO, disposem de 40 pins distribuïts en dues línies de 20 unitats cadascun.

Profunditat de bits: *Bit Depth,* és el número de bits que utilitzarem per emmagatzemar cada valor discret del nostre senyal. En aquest sentit, a més resolució, més marge dinàmic disponible. Es treballa en múltiples de 4. Una resolució estàndard és la del CD que és de 16 bits.

Quantificació: segon pas de la conversió d'analògic a digital. En aquest pas, les mostres preses anteriorment, adquireixen els valors definits en els intervals de quantització definits prèviament.

Raspberry Pi: ordinador del tipus SBC utilitzat. [6]

Reverberació: efecte produït per retards en el senyal d'àudio. Es produeix quan treballem amb retards de menys de 50 mil·lisegons entre els retards i el senyal original.

Sample time: en el cas del MATLAB, indica quan, en mode Simulació, s'actualitzen els valors tant de sortida com d'entrada.

Sample: mostra, en el nostre cas, d'àudio adquirida segons una freqüència de mostreig determinada.

SBC: *Single Board Computer* o Ordinadors d'una sola placa com el RPi, on trobem gairebé tots els components soldats a la mateixa placa.

Shield: placa impresa que es connecta a un maquinari, un RPi per exemple, per dotar-lo de noves capacitats. Poden ser targetes d'àudio, com en el nostre cas, o més interfícies de connexió, sensors o actuadors.

Simulink: paquet de programari desenvolupat per MathWorks [17].

Solver: Eina que ens permet calcular com es comporta el model generat al Simulink. Pot ser d'estat continu o d'estat discret. Determinarem les seves característiques en funció de les nostres necessitats concretes. En funció del tipus de *Solver* podem obtenir resultats diferents.

Splitter: divisor. Ho utilitzem per bifurcar el senyal entre dues o més continuacions.

SSH: *Secure Shell.* És un protocol que ens permet accedir a terminals remots via xarxa. Nosaltres l'utilitzem per accedir al RPi de forma remota a través de l'ordinador de desenvolupament.

Switch: interruptor commutador

Tarjeta SD: mètode d'emmagatzematge flash que utilitzen els sistemes RPi per funcionar.

TFG: acrònim de Treball de Final de Grau.

Time Scope: analitzador temporal del senyal. És un bloc del Simulink que ens permet visualitzar temporalment la forma d'ona del senyal amb què estem treballant.

USB:*Universal Serial Bus,* interfície de connexió per a perifèrics present a la majoria d'ordinadors actuals i també a alguns dels RPI.

VHDL: *Very High Speed Integrated Circuits Hardware Description Language*. Llenguatge de descripció de circuits electrònics que permet accelerar el procés de disseny.

Windows: sistema operatiu desenvolupat i distribuït per Microsoft. És programari propietari.

XCP: o *"Universal Measurement and Calibration Protocol".* Protocol de xarxa per la conectivitat i calibració de sistemes electrònics [41].

6. Bibliografia

[1] Duxans Barrobés, H [Helena]; Ruiz Costa-jussà, M [Marta] (2021). Efectes digitals del senyal d'àudio [recurs d'aprenentatge]. Recuperat de Universitat Oberta de Catalunya. https://materials.campus.uoc.edu/daisy/Materials/PID_00188060/pdf/PID_00188052.pdf

[2] Definició de Latència [en línia] [consulta: 9 de març del 2022] Disponible a: https://www.termcat.cat/ca/cercaterm/latencia?type=basic

[3] Line 6 announces tonecore™ dsp developer kit [en línia] [consulta: 9 de març del 2022] Disponible a: https://line6.com/news/pressReleases/644

[4]What is Arduino? [en línia] [consulta: 9 de març del 2022] Disponible a: https://www.arduino.cc/en/about#what-is-arduino

[5] echoTrek - Digital Delay / Echo - Audio Effects with Arduino [en línia] [consulta: 9 de març del 2022] Disponible a: https://create.arduino.cc/projecthub/CesarSound/echotrek-digital-delay-echo-audio-effects-with-arduino-b017a2

[6] RaspberryPi [en línia] [consulta: 9 de març del 2022] Disponible a: https://www.raspberrypi.com

[7] Pi-FX : A Raspberry Pi-Based Pedal Board [en línia] [consulta: 9 de març del 2022] Disponible a: https://tibbbz.medium.com/guitarix-the-pi-dle-board-8d6298ca8e42

[8] Raspberry Pi PC: DIY Guitar Pedal using NeuralPi [en línia] [consulta: 9 de març del 2022] Disponible a: https://opencloudware.com/raspberry-pi-pc-diy-guitar-pedal-using-neuralpi/

[9] Field-programmable gate array [en línia] [consulta: 9 de març del 2022] Disponible a: https://es.wikipedia.org/wiki/Field-programmable_gate_array

[10] Repositori del codi del projecte fpga-multieffect [en línia] [consulta: 9 de març del 2022] Disponible a: https://github.com/Vladilit/fpga-multi-effect

[11] FPGA Guitar Multi-Effects Processor [en línia] [consulta: 9 de març del 2022] Disponible a: https://web.mit.edu/6.111/www/f2017/projects/harisb_Project_Final_Report.pdf

[12] ZedBoard. Complete Development Kit for designers. [en línia] [consulta: 9 de març del 2022] Disponible a: https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/zedboard/

[13] Pàgina de la marca ElectroSmash [en línia] [consulta: 9 de març del 2022] Disponible a: https://www.electrosmash.com

[14] Time Manipulator - Arduino Delay/Echo/Reverb [en línia] [consulta: 9 de març del 2022] Disponible a: https://www.electrosmash.com/time-manipulator

[15]Raspberry Pi 4. Especificacions oficials[en línia] [consulta: 21 de març del 2022] Disponible a: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[16]Article a la Wikipedia sobre el Raspberry Pi OS [en línia] [consulta: 21 de març del 2022] Disponible a: https://en.wikipedia.org/wiki/Raspberry_Pi_OS

[17] Pàgina oficial del programari MATLAB de Mathworks[en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/products/matlab.html

[18] Pàgina oficial del programari Simulink de Mathworks [en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/products/simulink.html

[19] Pàgina oficial del complement per a treballar amb el RPi i Simulink de Mathworks [en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/help/supportpkg/raspberrypi/

[20] Fotografia del model de potenciòmetre utilitzat durant el projecte [en línia] [consulta: 21 de març del 2022] Disponible a: https://cdn.shopify.com/s/files/1/0176/3274/articles/DSC_0700_1100x.jpg?v=1561726902

[21]Tutorial de Mathworks sobre com treballar amb RPi 4 [en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/help/supportpkg/raspberrypi/ref/getting-started-with-raspberry-pi-hardware.html

[22] Article a la Wikipedia sobre l'emmascarament sonor[en línia] [consulta: 21 de març del 2022] Disponible a: https://ca.wikipedia.org/wiki/Emmascarament_sonor

[23]Pàgina a la wikipedia sobre la latència en l'àudio [en línia] [consulta: 21 de març del 2022] Disponible a: https://en.wikipedia.org/wiki/Latency_(audio)

[24] Definició de Reverberació [en línia] [consulta: 21 de març del 2022] Disponible a: https://dlc.iec.cat/Results?DecEntradaText=reverberació&AllInfoMorf=False&OperEntrada=0&OperDef=0&OperEx=0&OperSubEntrada=0&OperAreaTematica=0&InfoMorfType=0&OperCatGram=False&AccentSen=False&CurrentPage=0&refineSearch=0&Actualitzacions=False

[25] Pàgina oficial de l'equalitzador gràfic FCS-966 de la marca BSS by Harman [en línia] [consulta: 21 de març del 2022] Disponible a: https://bssaudio.com/en/products/fcs-966#product-thumbnails

[26] 7 different types of eqs (and when to use each) [en línia] [consulta: 21 de març del 2022] Disponible a: https://producerhive.com/ask-the-hive/different-types-of-eq/

[27] Pàgina oficial de l'equalitzador britànic GSXL4070 de la marca G-Sonique [en línia] [consulta: 21 de març del 2022] Disponible a: https://www.g-sonique.com/gsxl4070.html

[28] Pàgina oficial de suport de l'eina de disseny de filtres Difgital Filter Design [en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/help/dsp/ref/digitalfilterdesign.html

[29] Pàgina oficial de suport del complement d'àudio del Simulink de MathWorks [en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/products/audio.html

[30]Pàgina oficial de la família de taules de mescles MG12 de Yamaha Pro Audio [en línia] [consulta: 21 de març del 2022] Disponible a: https://es.yamaha.com/files/download/other_assets/2/1139202/mg12xuk_om_a0_es.pdf

[31] Pàgina de compra i especificacions de l'splitter de GPIO a la plataforma de començ electrònic Amazon [en línia] [consulta: 21 de març del 2022] Disponible a: https://www.amazon.com/-/es/Conectores-Raspberry-40-pin-expansión-RAS-GP02/dp/B07MCW4KCM?th=1

[32] Comparativa entre la capacitat de processament entre diverses versions de RPi [en línia] [consulta: 21 de març del 2022] Disponible a: https://browser.geekbench.com/v5/cpu/compare/14667135?baseline=9142258

[33] Pàgian de producte de la targeta de so externa HiFiBerry DAC+ADC Pro[en línia] [consulta: 21 de març del 2022] Disponible a: https://www.hifiberry.com/shop/boards/hifiberry-dac-adc-pro/

[34] Espceificació del protocol Inter-IC Sound Bus (I2S) [en línia] [consulta: 21 de març del 2022] Disponible a: https://www.infineon.com/dgdl/Infineon-Component_I2S_V2.0-Software+Module+Datasheets-v02_07-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ea2ed3c2596&utm_source=cypress&utm_medium=referral&utm_campaign=202110_globe_en_all_integration-files

[35] Play High-Quality Audio from Raspberry Pi Using I2S-Based DAC [en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/help/supportpkg/raspberrypi/ug/play-high-quality-audio-from-raspberry-pi-using-i2s-based-dac.html

[36] Pàgina oficial del sistema operatiu enfocat al tractament d'àudio Elk Audio OS[en línia] [consulta: 21 de març del 2022] Disponible a: https://elk.audio

[37]Pàgina de característiques del sistema operatiu Elk Audio OS [en línia] [consulta: 21 de març del 2022] Disponible a: https://elk.audio/how-elk-works/

[38] How do I get an out-of-the-box working Linux audio workstation? [en línia] [consulta: 21 de març del 2022] Disponible a: https://wiki.linuxaudio.org/wiki/system_configuration

[39] Consulta als fòrums de suport del MathWorks sobre "High latency for audio use on a Raspberry Pi using Simulink" [en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/matlabcentral/answers/164113-high-latency-for-audio-use-on-a-raspberry-pi-using-simulink

[40] Model d'exemple sobre el tractament d'àuido i la implementació d'un filtre de banda de pas sobre RPI proporcionat per MathWorks[en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/help/supportpkg/raspberrypi/ref/implement-bandpass-filter-on-raspberry-pi.html

[41] Article a la wikpedia sobre el protocol XCP [en línia] [consulta: 21 de març del 2022] Disponible a: https://en.wikipedia.org/wiki/XCP_(protocol)

[42] Article als fórums de MathWorks sobre "Communicate with ECUs from MATLAB and Simulink using XCP Protocol" [en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/discovery/xcp-protocol.html

[43] "Introduction to XCP Protocol" [en línia] [consulta: 21 de març del 2022] Disponible a: https://piembsystech.com/xcp-protocol/

[44] Article a AdocLib Simulink Raspberry Hardware External Mode Error Issue [en línia] [consulta: 21 de març del 2022] Disponible a: https://www.adoclib.com/blog/simulink-raspberry-hardware-external-mode-error-issue.html

[45]Recurs de MathWorks sobre l'External Mode del Simulink [en línia] [consulta: 21 de març del 2022] Disponible a: https://www.youtube.com/watch?v=rTgLpmFluGg

[46] Pregunta als fórums de MathWorks sobre "Raspberry PI Simulink Package Two Encoders XCP Error" [en línia] [consulta: 21 de març del 2022] Disponible a: https://es.mathworks.com/matlabcentral/answers/1669794-raspberry-pi-simulink-package-two-encoders-xcp-error

[47]Pregunta pròpia als fòrums de suport de HiFiBerry sobre el tractament de les dades al GPIO [en línia] [consulta: 21 de març del 2022] Disponible a: https://support.hifiberry.com/hc/en-us/requests/21162

[48] D.Reis Joshua y P.McPherson, Andrew, 2015. *Audio Effects. Theory, Implementation and Application*. Boca Raton, USA:CRC Press. ISBN :978-1-4665-6029-1.

[49] Imatge oficial del pedal digital DD8 de Boss [en línia] [consulta: 25 de maig del 2022] Disponible a: https://static.roland.com/assets/images/products/main/dd-8_main.jpg

[50] Imatge oficial del multiefectes digital G-Natural de TC Electronics [en línia] [consulta: 25 de maig del 2022] Disponible a: https://mediadl.musictribe.com/media/PLM/data/images/products/HE068/2000Wx2000H/G-NATURAL_HE068_Top_XL.png

[51] Manual d'usuari del tonecore™ dsp developer kit [en línia] [consulta: 25 de maig del 2022] Disponible a: https://l6c-acdn2.line6.net/data/6/0a06434c139e6520bcdfc67791/application/pdf/TCDDK%20Users%20Guide%20(%20Rev%20A%20).pdf?_ga=2.127025485.1761089947.1646669672-1386819651.1646669672

[52] Article de suport de MathWorks sobre el *Sample based* i el *frame-based* [en línia] [consulta: 25 de maig del 2022] Disponible a: https://es.mathworks.com/help/dsp/ug/sample-and-frame-based-concepts.html

[53]Pàgina oficial de producte del driver Jack Audio [en línia] [consulta: 25 de maig del 2022] Disponible a: https://jackaudio.org

7. Annex

Conjuntament amb aquest document s'entrega un arxiu comprimit amb un total de 10 models executables dins el propi Simulink. Per limitacions del programari els noms no poden ser els mateixos amb que ens hi referim al llarg de la memòria. L'equivalència és la que segueix:

Nom de l'arxiu entregat	Referència en aquesta memòria
Bypass	Apartat 3.3.1
Delay	Apartat 3.3.2
Disseny del hardware	Apartat 3.3.7
Equalitzador_amb_AudioTools	Apartat 3.3.6 - Opció número 2
Equalitzador_amb_FDD	Apartat 3.3.6 - Opció número 1
Flanger	Apartat 3.3.4
Model_final_4potenciometres	Apartat 3.4.5
Model_final	Apartat 3.4.5
Reverb	Apartat 3.3.3
Tremolo	Apartat 3.3.5

Taula 4 - Equivalències en la nomenclatura dels models entregats i presentats