UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

TREBALL DE FI D'ESTUDIS

Desenvolupament d'una plataforma per a la simulació d'efectes d'audio

Document:

Memòria

Autor/Autora:

Joan Moya Muñoz

Director/Directora - Codirector/Codirectora:

Nestor Berberl Artal

Titulació:

Màster/Grau en Enginyeria de Sistemes Audiovisuals

Convocatòria:

Primavera, 2022.

Capítol 1

Agraïments

M'agradaria començar aquest treball agraint a tothom que ha fet que aquest treball sigui possible:

En primer lloc, m'agradaria agrair a l'Alba, la meva parella, per tot el suport que m'ha donat durant tota la carrera, per estar allà quan ho necessitava i per descobrir-me el que seria la meva nova passió. Sense ella possiblement aquest treball no seria possible

Als meus pares, el Joan i la Lourdes, que durant tots aquests anys han estat al meu costat, a les bones i a les dolentes. Sense ells possiblement aquest treball no seria possible.

Al meu tutor, en Néstor per aguantar-me durant gran part de la carrera al seu despatx donant la tabarra. Sense ell aquest treball no seria possible.

I finalment a tothom que ha estat amb mi durant aquests anys. Sense vosaltres aquest treball no sería possible.

Moltes gràcies per tot.

Resum

El principal objectiu d'aquest projecte de fi de grau és el desenvolupament d'una plataforma per a la simulació d'efectes electrònics analògics basats en efectes de guitarra.

La principal finalitat del projecte serà el desenvolupament d'un aplicatiu web a través del qual l'usuari pugui seleccionar un efecte de guitarra base, modificar els seus paràmetres i o components, i finalment carregar un arxiu d'àudio al qual se li aplicarà l'efecte. Un cop pujat l'arxiu es farà una simulació que l'usuari podrà descarregar per veure si el seu disseny és el desitjat.

Es buscarà desenvolupar al projecte sempre que sigui amb eines de programari lliure, tant per a la simulació com per al desenvolupament de l'aplicatiu web. Obtenir una bona qualitat de simulació serà una de les prioritats del projecte, ja que en cas de no aconseguir-ho el projecte perdria totalment el sentit.

També es procurarà que qualsevol usuari sigui capaç d'utilitzar l'eina desenvolupada, no sent necessaris coneixements avançats de disseny de circuits electrònics.

Amb el desenvolupament d'aquest projecte es busca optimitzar el procés de fabricació d'aquests efectes millorant la fase de disseny i estalviant temps de proves.

Abstract

The main objective of this thesis is the development of a platform for the simulation of analog electronic effects based on guitar effects.

The main purpose of the project will be the development of a web application through which the user can select a base guitar effect, modify its parameters and components, and finally upload an audio file to which the effect will be applied. Once the file is loaded, a simulation will be made and the user will be able to download it to see if the design is the desired one.

The project will be developed with free software tools, both for the simulation and for the development of the web application. Obtaining a good quality of simulation will be one of the priorities of the project, since in case of not achieving it, the project would lose its sense.

It is also intended that any user will be able to use the developed device, no advanced knowledge of electronic circuit design is required.

The development of this project aims to optimize the manufacturing process of these effects by improving the design phase and saving testing time.

Índex

Llista de figures

Llista de taules

Llista d'abreviatures

SPICE: Simulation Program with Integrated Circuit Emphasis

DC: Direct Current

PNP: Positive-Negative-Positive transistor

IR: Impulse Response

ASIO: Audio Stream Input/Output

VST: Virtual Studio Technology

DAW: Digital Audio Workstation

PIP: Package Installer for Python

HTML: Hyper Text Markup Language

MIME: Multi-Purpose Internet Mail Extensions

TINA-TI: Toolkit for Interactive Network Analysis - Texas Instruments

Capítol 2

Introducció

2.1 Objecte

El principal objectiu d'aquest projecte de fi de grau és el desenvolupament d'una plataforma per a la simulació d'efectes de guitarra.

La principal idea és que un usuari sense haver de construir físicament l'efecte de guitarra o de fer ell mateix les simulacions pugui modificar els seus components o ajustar els paràmetres segons les seves preferències.

El principal objectiu del projecte serà el desenvolupament d'un aplicatiu web a través del qual l'usuari pugui seleccionar un efecte de guitarra base, modificar els seus paràmetres i o components, i finalment, carregar un arxiu d'àudio al qual se li aplicarà l'efecte. Un cop pujat l'arxiu es farà una simulació que l'usuari podrà descarregar per veure si el seu disseny és el desitjat.

2.2 Abast

Durant el projecte s'executaran les següents tasques:
- Estudi d'efectes de guitarra.
- Estudi dels simuladors d'efectes de guitarra.
- Estudi d'eines per al desenvolupament web.
- Simulació d'un efecte de guitarra.
- Implementació de la simulació d'un efecte de guitarra en Python.
- Disseny d'una pàgina web.
- Implementació d'una pàgina web en Python.

2.3 Requeriments

Ús de programari lliure:

El projecte es durà a terme mitjançant l'ús de programari lliure. Això ens permetrà poder modificar el programari i poder adaptar-lo a totes les necessitats que ho requereixin. A més això permetrà a l'usuari utilitzar la plataforma sense cap cost.

Bona qualitat de simulació:

Una bona qualitat de simulació és vital per al projecte. Si no l'aconseguim, l'usuari no podrà dissenyar el seu propi efecte de guitarra d'una manera fiable. Per això obtenir una bona qualitat de simulació és un requeriment imprescindible.

Baix cost d'execució del projecte:

Com no es disposa d'una font de finançament externa, sinó que el projecte és autofinançat no serà possible dur a terme el projecte si aquest té un cost d'execució elevat.

Senzillesa d'ús per a l'usuari final:

Al cap i a la fi, l'aplicatiu web està plantejat perquè pugui ser utilitzat per qualsevol usuari pugui modificar l'efecte de manera senzilla, és per això que es buscarà que tot l'aplicatiu presenti una interfície que sigui fàcil d'utilitzar per a l'usuari.

2.4 Justificació

La intenció del projecte és permetre a l'usuari amant de la guitarra i dels seus efectes poder donar-li la possibilitat de veure quin resultat tindrà les seves modificacions, i així optimitzar el procés de construcció de l'efecte de guitarra desitjat.

Els principals avantatges que presenta aquest projecte és que pot permetre a l'usuari veure quin efecte tindrien algunes modificacions en el disseny de l'efecte de guitarra. També que és un sistema més àgil i senzill que fer la simulació i modificació de l'efecte per ell mateix.

El principal desavantatge és que l'usuari haurà de partir d'un efecte d'àudio base i a partir d'aquest fer modificacions sobre aquest efecte, no podrà crear el seu propi efecte des de zero.

Capítol 3

Estat de la questió

3.1 Els efectes de guitarra

Els efectes de guitarra, coneguts també com a pedals de guitarra, són uns circuits electrònics utilitzats pels guitarristes per tal de modificar el so de la seva guitarra mentre toquen. Aquests circuits reben el senyal provinent de la guitarra i el modifiquen segons el tipus de circuit que tinguin a dins. Podríem trobar des d'efectes que donin un guany molt elevat al senyal fent que aquest es distorsioni molt com els Overdrives, fins a efectes que canviïn la fase del nostre senyal, com per exemple un pedal Phaser.

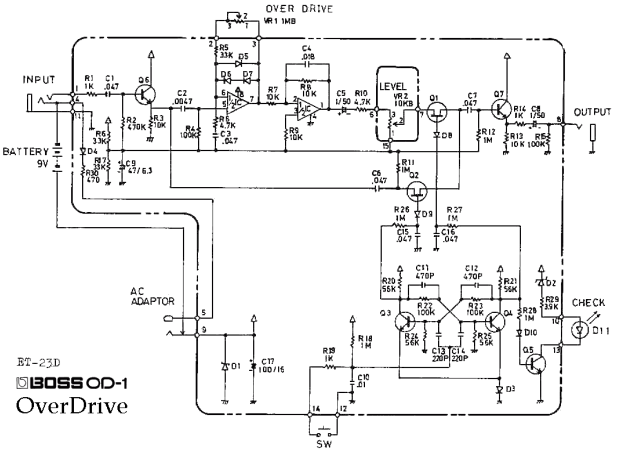

Figura 3.1: Esquema electrònic d'un efecte BOSS OD1 [1].

Com podem veure a la Figura 3.1 veiem com rebem el senyal d'entrada a través de l'Input i aquesta travessa tot el circuit electrònic fins a arribar a la sortida.

Una de les característiques dels efectes de guitarra és que es poden connectar en sèrie, permetent-nos així anar afegint efectes fins a obtenir el so desitjat. La cadena de senyal, per tant, estaria formada per 3 elements principals, la guitarra elèctrica, els efectes i l'amplificador, tal com es mostra a la Figura 3.2.

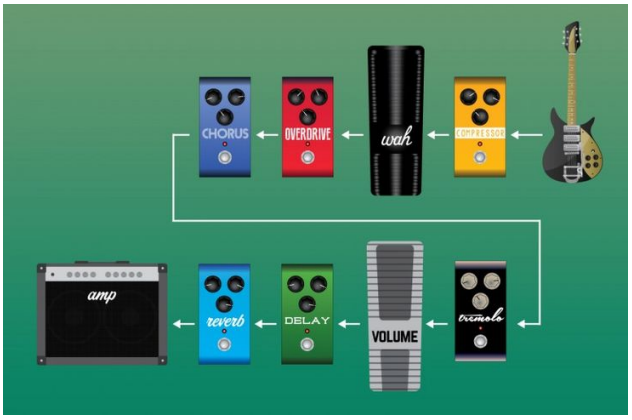

Figura 3.2: Exemple de cadena de senyal [2].

3.1.1 Efectes Booster i Dallas Rangemaster

Els efectes booster són un dels primers efectes analògics de guitarra a ser desenvolupats. Acostumen a ser circuits molt senzills i amb molt pocs components que serveixen per amplificar el senyal de la nostra guitarra i a més potenciar unes freqüències determinades. Gràcies a la seva senzillesa i simplicitat s'ha escollit el pedal Dallas Rangemaster com a pedal exemple per a dur a terme aquest treball.

Figura 3.3: Efecte Dallas Rangemaster [3].

El Dallas RangeMaster Treble Booster (Figura 3.3), més conegut com a Dallas RangeMaster va ser un efecte de guitarra dissenyat el 1959 i posat a la venda posteriorment al 1960 que tenia 2 objectius: el primer era amplificar el senyal que anava des de la guitarra fins a l'amplificador i el segon era donar més brillantor a aquelles freqüències més elevades de l'espectre.

El Dallas RangeMaster és un pedal molt senzill, ja que utilitza al voltant d'uns 12 components electrònics per obtenir el seu característic so. És a causa de la seva senzillesa que en aquest treball s'utilitzarà com a pedal de referència per a les simulacions.

Un cop feta aquesta petita introducció es farà una breu anàlisi de funcionament d'aquest efecte de guitarra.

3.1.2 Funcionament del Dallas RangeMaster

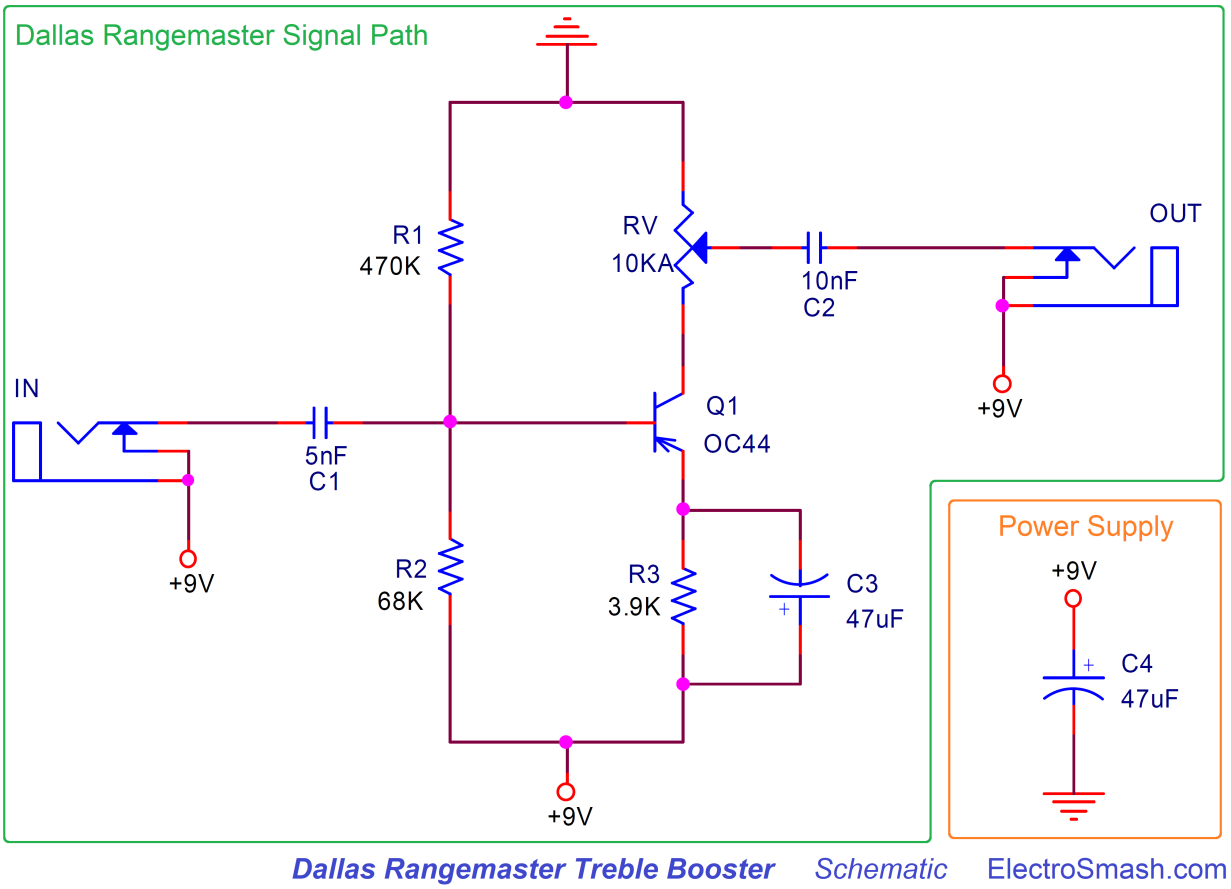

Figura 3.4: Esquema del Dallas Rangemaster[3].

Com podem comprovar a la Figura 3.4, el Dallas RangeMaster és un efecte de guitarra molt senzill. En ser un circuit senzill es pot fer una anàlisi per component per veure quina és la funció de cadascun[3]:

C1: És l'encarregat de bloquejar qualsevol resta de corrent DC provinent de l'entrada. A més també forma un filtre passa-alts amb R_1 i R_2. Aquest component és molt important de cara a l'equalització de l'efecte, si el modifiquem podem provocar grans canvis en l'efecte. Podem observar la Figura 3.5 que mostra el guany en freqüència del circuit que, a mesura que el valor de C_1 disminueix, cada cop s'atenuen més les freqüències greus.

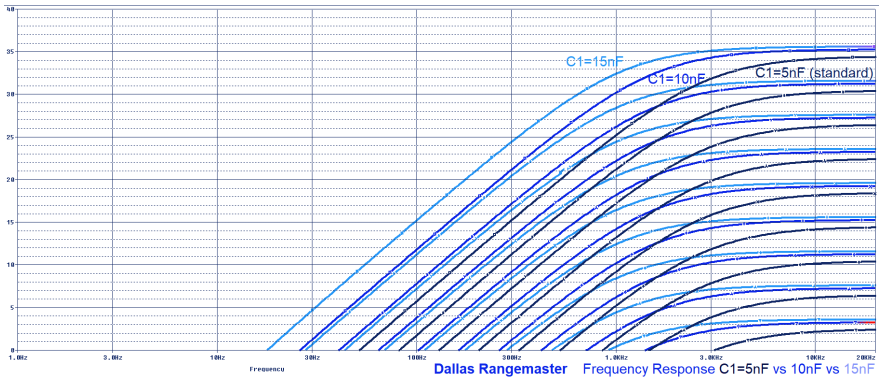

Figura 3.5: Resposta en freqüència segons el valor del condensador C1 [3].

R1 i R2: Aquestes resistències són les encarregades de crear un divisor de tensió per tal de polaritzar al transistor Q1.

RV: És el potenciòmetre encarregat de controlar el guany i volum que tindrà l'efecte.

Q1: El transistor Q_1 és el principal component que ens permet obtenir el característic so del Dallas RangeMaster. Es tracta d'un transistor PNP model Mullard OC44 col·locat en estructura d'emissor comú.

R3: És l'encarregada de proporcionar la tensió negativa de realimentació per tal de mantenir el punt de bias del transistor estable.

C3: Permet al senyal continuar tenint un guany elevat malgrat la realimentació negativa de R_3.

C2: És l'encarregat de filtrar la tensió DC que pugui haver-hi a la sortida del circuit.

En conclusió, el Dallas Rangemaster és un efecte de guitarra que permet modificar l'espectre del so provinent de la guitarra. A partir de modificar la posició del potenciòmetre RV, controlarem la quantitat de guany aplicada al nostre senyal provinent de la guitarra.

3.1.3 Anàlisi matemàtica del funcionament del Dallas RangeMaster

Polarització del transistor:

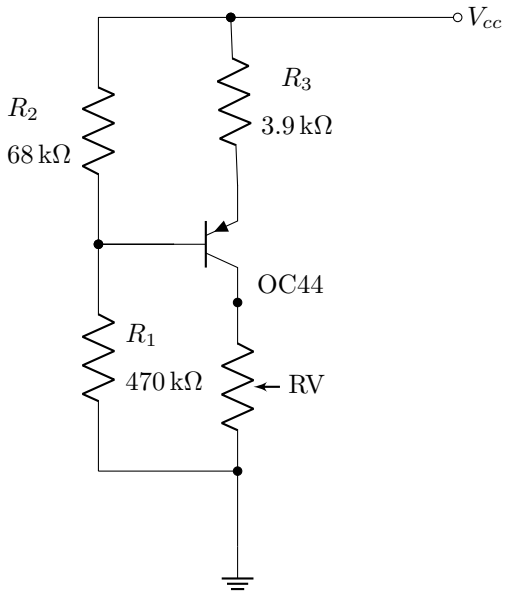

Figura 3.6: Components del circuit que polaritzen Q1.

Amb les resistències R_1, R_2, R_3 i el potenciòmetre R_V, es polaritza el transistor perquè circuli un corrent pel terminal de base i de col·lector. Analitzant el circuit de la Figura 3.6, s'obtenen les expressions del corrent de base (I_{BQ} (Equació 3.1) i del corrent de col·lector (I_{CQ}) (Equació 3.2), tot suposant que el transistor es troba a la seva regió activa.

$$I_{BQ} = \frac{V_{CC} - V_{CC} \cdot \frac{R_1}{R_1+R_2} - V_{EBQ}}{\frac{R_1 \cdot R_2}{R_1+R_2} + R_3 \cdot (\beta + 1)} \tag{3.1}$$

$$I_{CQ} = \beta \cdot \frac{V_{CC} - V_{CC} \cdot \frac{R_1}{R_1+R_2} - V_{EBQ}}{\frac{R_1 \cdot R_2}{R_1+R_2} + R_3 \cdot (\beta + 1)} \tag{3.2}$$

Anàlisi en petit senyal del circuit.:

Per a trobar el model matemàtic de l'amplificador Dallas RangeMaster, s'ha d'analitzar el circuit en petit senyal. Per això, se substitueix el transistor BJT pel model en petit senyal (Figura 3.7.)

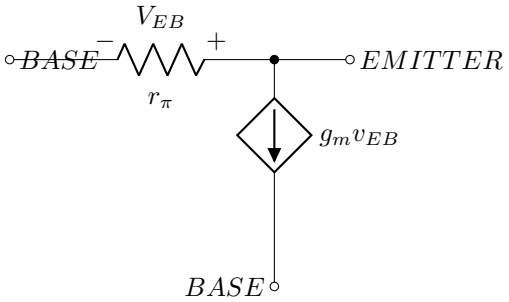

Figura 3.7: Substitució del BJT pel model en petit senyal.

El model consta de dos paràmetres: la resistència r_π i el guany en transconductància g_m, que es calculen segons les equacions 3.3 i 3.4.

$$r_\pi = \frac{V_T}{I_{BQ}} \tag{3.3}$$

$$g_m = \frac{I_{CQ}}{V_T} \tag{3.4}$$

El paràmetre V_T és la tensió tèrmica, un paràmetre físic dels semiconductors que depèn de la temperatura a la qual estigui sotmesa el semiconductor. La tensió tèrmica V_T es calcula segons l'equació 3.5.

$$V_T = \frac{k_B \cdot T}{q} \tag{3.5}$$

On k_B és la constant de Boltzmann, T la temperatura, en Kelvin, a la qual està sotmesa el dispositiu i q la càrrega de l'electró.

Per a trobar una expressió matemàtica que permeti relacionar la tensió de sortida v_{OUT} i la tensió d'entrada v_{IN} del circuit, se substitueix el transistor pel model en petit senyal i s'anul·len les fonts de contínua que hi hagui en el circuit. Realitzant aquestes substitucions, el circuit que queda és el que es mostra a la Figura: 3.8.

Aplicant l'anàlisi de nodes, i imposant les tres tensions de node V_A, V_B i V_C, s'obtenen les equacions 3.6, 3.7 i 3.8.

$$\frac{v_{IN} - V_A}{\frac{1}{s \cdot C_1}} = \frac{V_A}{R_2} + \frac{V_A}{R_1} + \frac{V_A - V_B}{r_\pi} \tag{3.6}$$

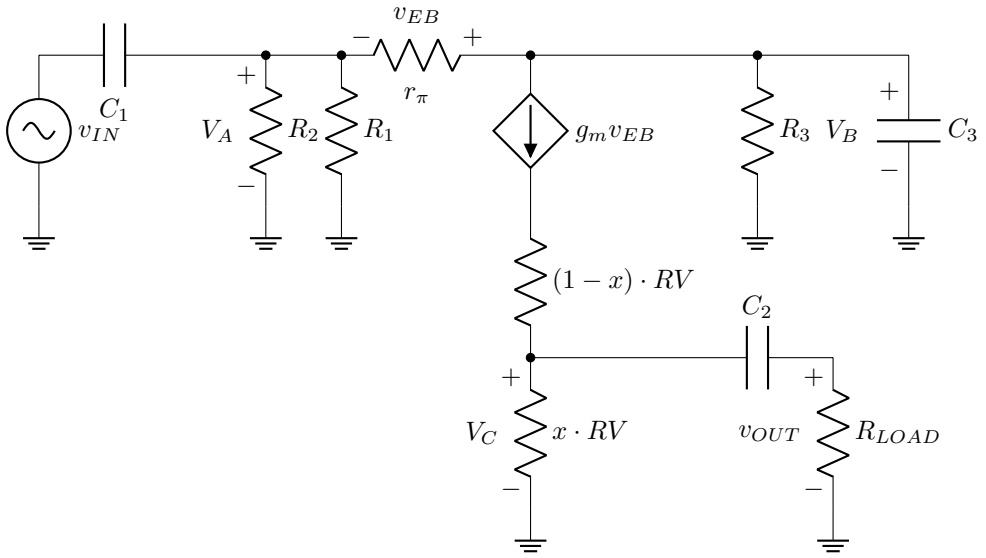

Figura 3.8: Circuit del Dallas RangeMaster sense fonts de corrent continua.

$$\frac{V_A - V_B}{r_\pi} = g_m \cdot v_{EB} + \frac{V_B}{R_3} + \frac{V_B}{\frac{1}{s \cdot C_3}} \tag{3.7}$$

$$g_m \cdot v_{EB} = \frac{V_C}{x \cdot RV} + \frac{V_C}{\frac{1}{s \cdot C_2} + R_{LOAD}} \tag{3.8}$$

On la tensió V_{EB} es relaciona amb les tensions de node V_A i V_B a partir de l'equació 3.9.

$$V_{EB} = V_B - V_A \tag{3.9}$$

La tensió de sortida V_{OUT} es relaciona amb la tensió de node V_C a partir de l'equació 3.10.

$$v_{OUT} = V_C \cdot \frac{R_{LOAD}}{\frac{1}{s \cdot C_2} + R_{LOAD}} \tag{3.10}$$

Donant solució a totes les equacions anteriors, es troba la relació entre la tensió de sortida v_{OUT} i la tensió d'entrada v_{IN} (Equació 3.11).

$$G(s) = \frac{v_{OUT}}{V_{IN}} = \frac{NUM}{DEN} \tag{3.11}$$

On el numerador es detalla a l'equació 3.12.

$$NUM = -(C_3 \cdot R_3 \cdot s + 1) \cdot s^2 \cdot R_2 \cdot R_1 \cdot C_1 \cdot R_V \cdot g_m \cdot r_\pi \cdot x \cdot R_{LOAD} \cdot C_2 \tag{3.12}$$

I el denominador es detalla a l'equació 3.13

$$DEN = A \cdot s^3 + B \cdot s^2 + C \cdot s + D \tag{3.13}$$

On els paràmetres A, B, C i D es detallen a

$$A = C_1 \cdot C_3 \cdot R_1 \cdot R_2 \cdot R_3 \cdot r_\pi \cdot C_2 \cdot (RV \cdot x + R_{LOAD}) \tag{3.14}$$

$$\begin{aligned} B = \Bigg(\Big(\Big((C_1 \cdot g_m \cdot r_\pi + C_1 + C_3) \cdot R_3 + C_1 \cdot r_\pi \Big) \cdot R_2 + r_\pi \cdot C_3 \cdot R_3 \Big) \cdot R_1 \\ + C_3 \cdot R_2 \cdot R_3 \cdot r_\pi \Bigg) \cdot C_2 \cdot \Big(RV \cdot x + R_{LOAD} \Big) + C_1 \cdot C_3 \cdot r_\pi \cdot R_3 \cdot R_2 \cdot R_1 \end{aligned} \tag{3.15}$$

$$\begin{aligned} C = \Big(\Big(R_2 + (g_m \cdot r_\pi + 1) \cdot R_3 + r_\pi \Big) \cdot R_1 + R_2 \cdot \Big((g_m \cdot r_\pi + 1) \cdot R_3 + r_\pi \Big) \Big) \cdot C_2 \cdot \Big(RV \cdot x + R_{LOAD} \Big) \\ + \Big(\Big((C_1 \cdot g_m \cdot r_\pi + C_1 + C_3) \cdot R_3 + C_1 \cdot r_\pi \Big) \cdot R_2 + r_\pi \cdot C_3 \cdot R_3 \Big) \cdot R_1 + C_3 \cdot R_2 \cdot R_3 \cdot r_\pi \end{aligned} \tag{3.16}$$

$$D = (R_3 \cdot g_m + 1) \cdot (R_1 + R_2) \cdot r_\pi + (R_2 + R_3) \cdot R_1 + R_2 \cdot R_3 \tag{3.17}$$

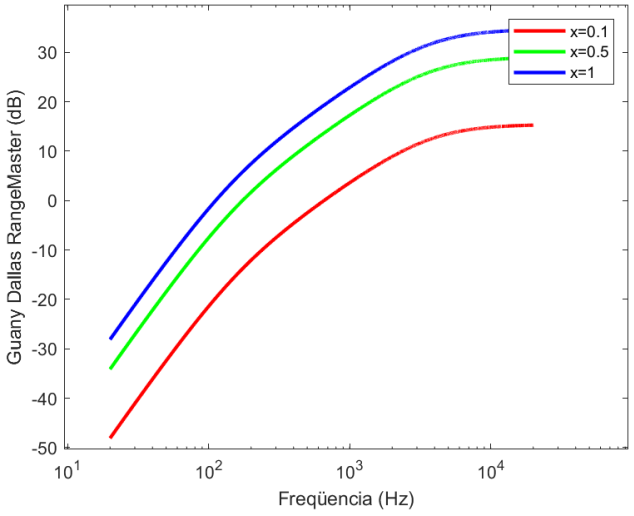

Figura 3.9: Variació de la resposta freqüencial del Dallas RangeMaster en funció de la posició del potenciòmetre. Elaboració pròpia.

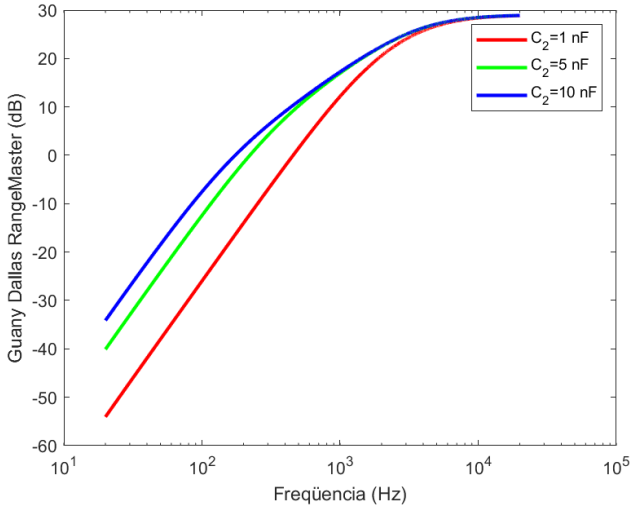

Figura 3.10: Variació de la resposta freqüencial del Dallas RangeMaster en funció del condensador C_2.Elaboració pròpia.

3.2 Simulació dels efectes de guitarra

Actualment, al mercat existeixen una gran quantitat de plataformes que ens permeten dur a terme la simulació d'efectes de guitarra. En aquest apartat es veuran les 2 plataformes disponibles que s'han agafat com a referència per dur a terme aquest projecte i es veuran tant els avantatges com els inconvenients de cadascuna.

3.2.1 Eines actuals de simulació

- **StompenbergFX**

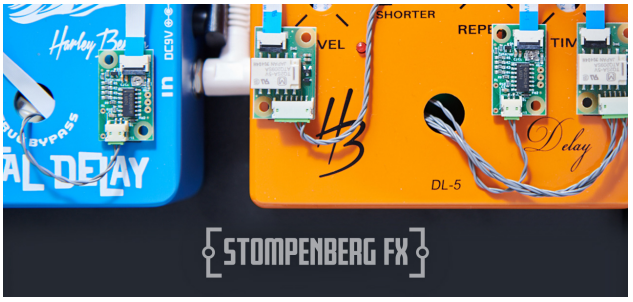

Figura 3.11: Logo de Stompenberg FX[4].

StompbergFX[4] és la plataforma de simulació d'efectes de guitarra de la coneguda botiga d'instruments Thomann. Per tal de poder dur a terme la simulació els pedals físics han estat modificats per tal que el senyal d'entrada sigui el que seleccioni l'usuari.

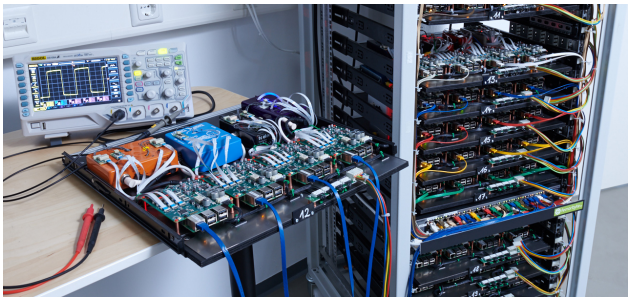

Figura 3.12: Efectes modificats de la plataforma StompbergFx[4].

Avantatges d'StompbergFx:

Com a avantatges d'aquesta eina de simulació tenim que la simulació és molt fiable, ja que l'àudio passa físicament a través de l'efecte desitjat, això permet que el so que obtenim sigui el mateix que obtindríem en cas d'utilitzar nosaltres l'efecte. També StompbergFx permet a l'usuari modificar els paràmetres de l'efecte molt fàcilment, presenta una interfície gràfica molt amable per l'usuari.

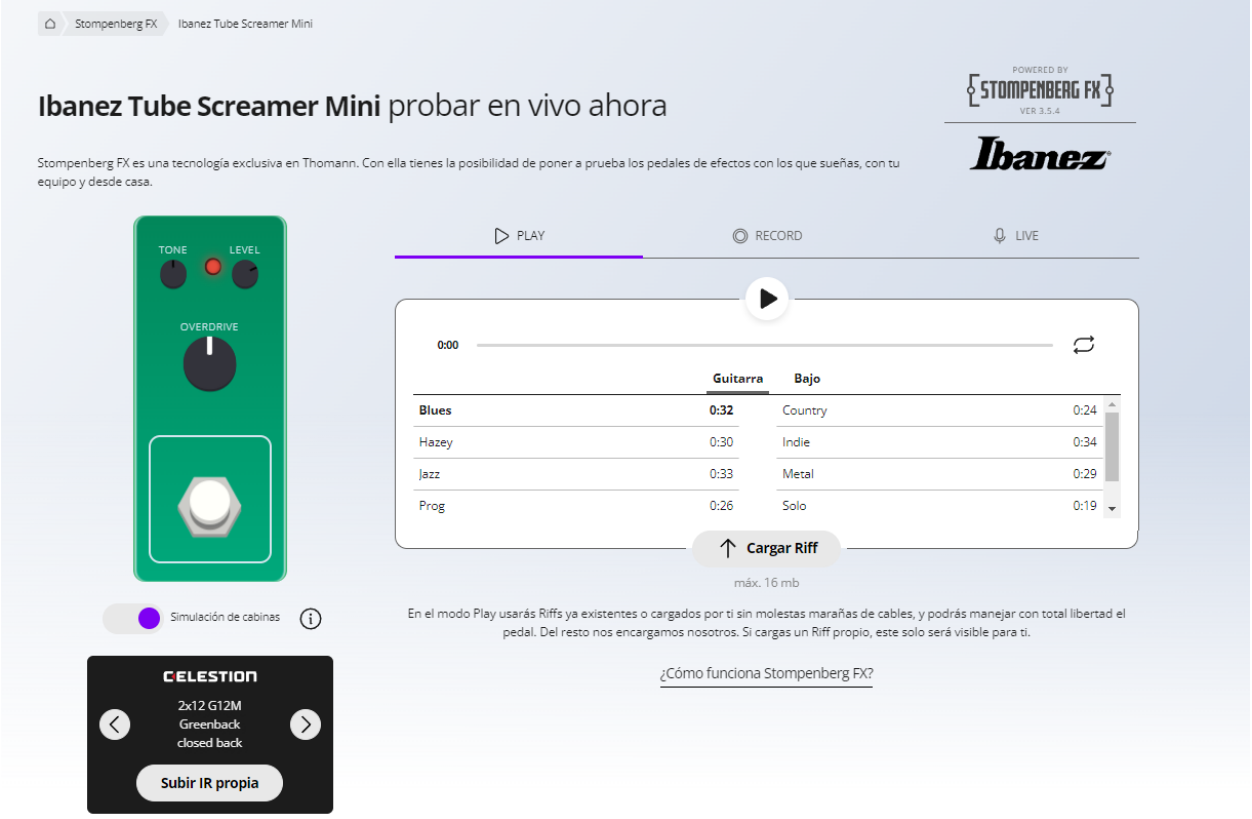

Figura 3.13: Interfície d'StompbergFx[4].

Com podem veure a la Figura 3.13 disposem d'una representació de l'efecte que hem seleccionat amb el que podem interactuar en temps real modificant els valors dels potenciòmetres. També disposa d'una opció que permet la simulació d'IR's, així com diferents formes per introduir l'àudio perquè es faci la simulació: Amb arxius predefinits de diferents estils musicals, pujant el nostre arxiu gravat, enregistrar la nostra guitarra directament sobre la pàgina web i tocant en temps real la nostra guitarra.

Inconvenients d'StompbergFx:

StompbergFx com s'ha comentat abans està construït físicament, això comporta algunes limitacions, per exemple que un mateix efecte no pot ser utilitzat per més d'un mateix usuari, en cas que vulguem utilitzar un efecte que estigui sent utilitzat per un altre usuari ens posarà en cua d'espera i se li donarà un temps de finalització a l'usuari de 5 minuts.

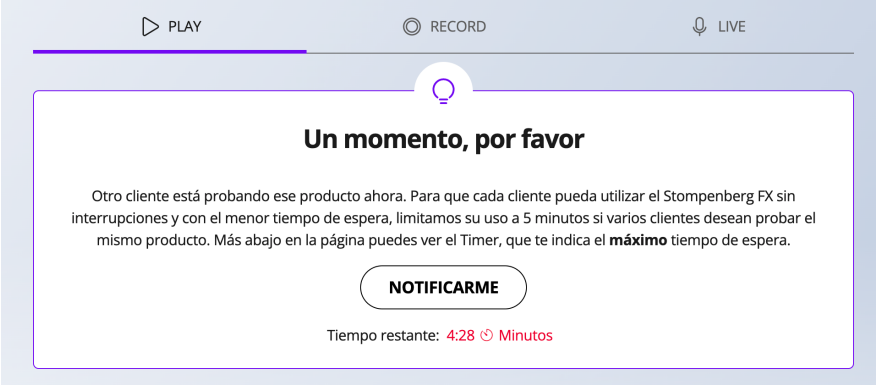

Figura 3.14: Avís de time out d'StompenbergFx[4].

També un altre inconvenient d'StompenbergFx és que no permet modificar els efectes que proporciona, això té sentit, ja que Thomann en tractar-se d'una botiga especialitzada en efectes de guitarra els articles que té a la venda són els efectes originals.

- **LiveSPICE**

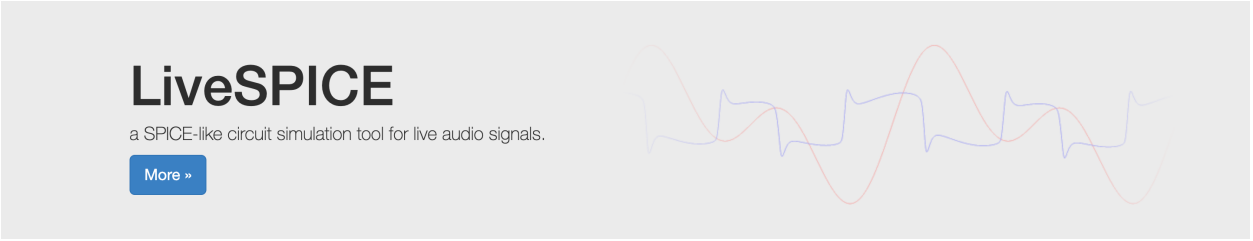

Figura 3.15: LiveSPICE[5].

LiveSPICE[5] és una aplicació de codi obert desenvolupada per DIllion Sherpet l'any 2020. Aquesta aplicació permet la creació de circuits electrònics orientats a àudio, sobretot efectes de guitarra. Permet que l'usuari dissenyi el seu propi efecte i després vegi com quedaria el resultat sobre un enregistrament de guitarra. Aquesta aplicació presenta avantatges i inconvenients.

Avantatges de LiveSPICE:
El principal avantatge de LiveSPICE és la completa personalització del circuit, ja que el mateix usuari és el que decideix cada element segons la seva preferència. Presenta una àmplia llibreria de components electrònics per a poder fer el disseny desitjat.

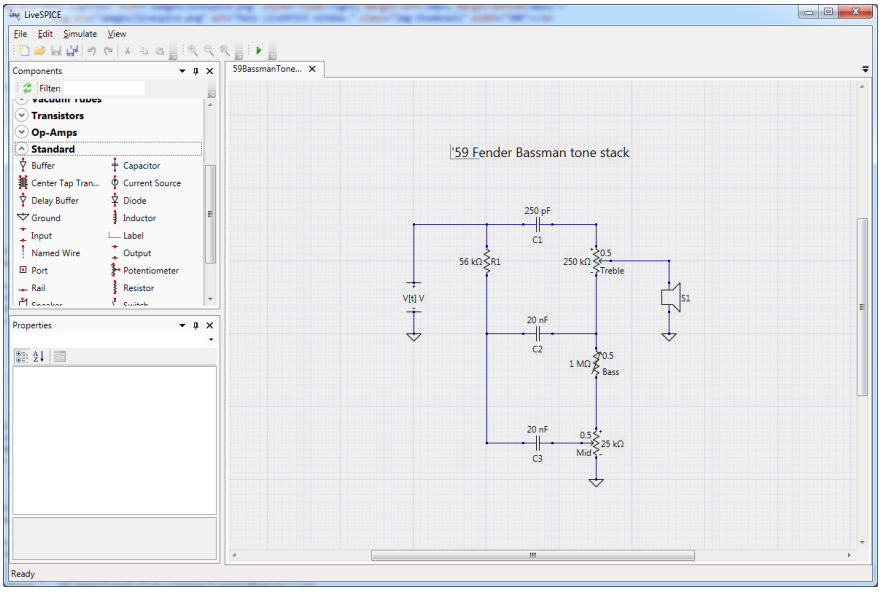

Figura 3.16: Exemple de circuit amb LiveSPICE[5].

També igual que l'StompenbergFx permet la modificació dels paràmetres en temps real, i a més permet la visualització del senyal en temps real.

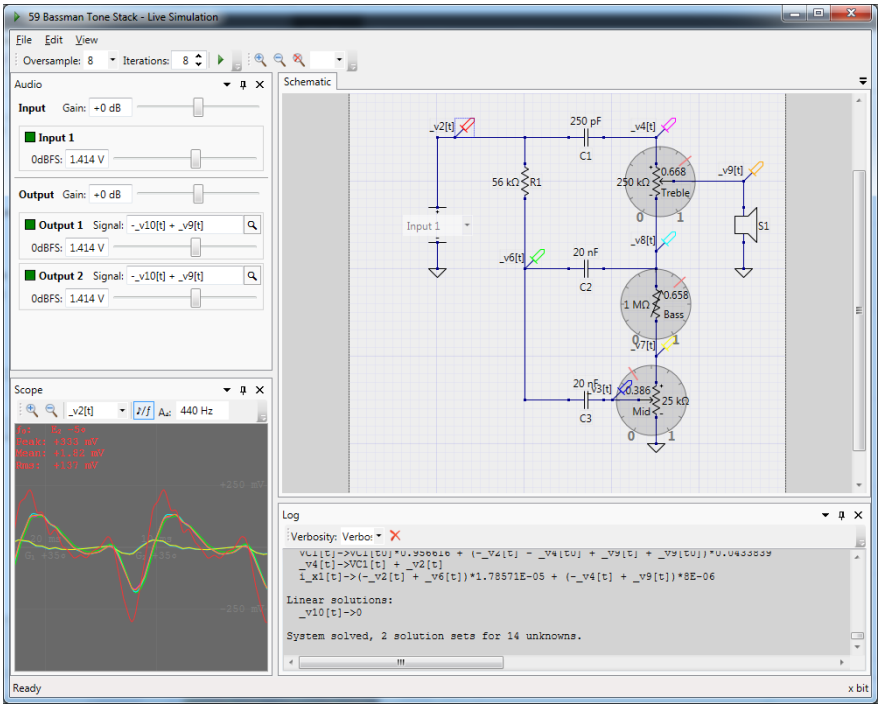

Figura 3.17: Exemple de controls de LiveSPICE[5].

També permet compatibilitats amb programari ASIO i a més disposa d'un plugin VST per interactuar amb DAW's. Aquestes característiques fan de LiveSPICE molt versàtil per a l'usuari.

Desavantatges de LiveSPICE:

Un dels principals desavantatges és que LiveSPICE només està disponible per a la plataforma Windows, així

que es limita l'ús d'aquesta aplicació als usuaris que disposin d'aquest sistema operatiu.
També com a inconvenient trobem que LiveSPICE no utilitza els components reals per dur a terme les simulacions, sinó que n'utilitza de simplificats per tal d'alleugerir els càlculs que s'han de fer per tal de simular l'efecte i que el temps es vegi reduït a costa de perdre una mica de fiabilitat de simulació.

3.3 Conclusions dels estudis previs

Un cop acabats aquests estudis previs ja s'ha obtingut una idea més especifica que és exactament el simulador que es vol implementar. La idea del simulador que es dissenyi és que permeti combinar la fiabilitat i realisme de l'StompenbergFX juntament amb la personalització del LiveSPICE. A més a més s'ha aprofundit una mica sobre el funcionament de l'efecte Dallas RangeMaster per entendre una mica millor el seu funcionament.

Capítol 4

Metodologia

En aquest capítol es veurà quines són les eines amb les quals es durà a terme el projecte. Més concretament es descriurà l'ordinador escollit, així com el programari que s'utilitzarà

4.1 Raspberry Pi 4

Per a dur a terme aquest projecte s'ha adquirit un ordinador de la família Raspberry Pi 4. La Raspberry Pi és l'última generació de l'ordinador monoplaca Raspberry PI de la Fundació Raspberry. S'ha escollit el model Raspberry Pi 4 4 GB RAM per a elaborar aquest projecte a causa del seu reduït preu i a les seves característiques. Disposar d'un processador Quad Core Cortex-A72 (ARM v8) 64-bit a 1,5 GHz i d'una quantitat de memòria de 4 GB fan que sigui un ordinador de baix cost ideal per a dur a terme aquest projecte.

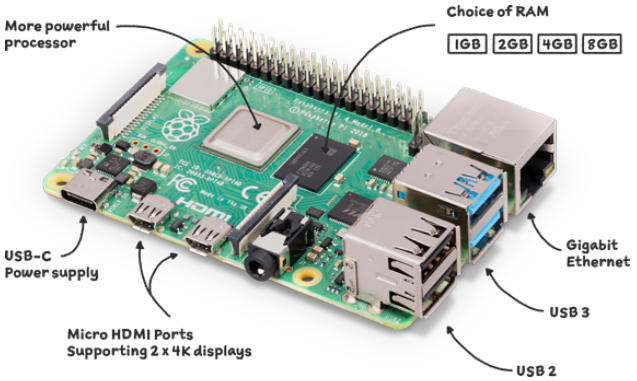

Figura 4.1: Raspberry Pi 4[6]

Un cop adquirida la Raspberry Pi 4 es va procedir a instal·lar el sistema operatiu. Es va decidir instal·lar el sistema operatiu Raspberry PI OS Bullseyes de 64 bits. Aquest serà el sistema base amb el qual implementarem el simulador.

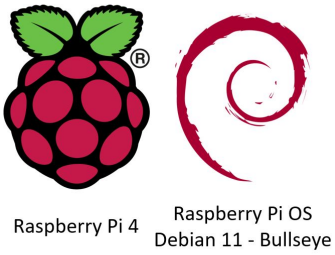

Figura 4.2: Logo del sistema operatiu Bullseye

Un cop es disposa de la Raspberry Pi amb una instal·lació neta es procedeix a la instal·lació del següent programari:

- Visual Studio Code
- VNC Server
- NgSpice
- Python 3
- Gestor de paquets de Python
- Thonny Python IDE
- VIM

4.2 Llenguatge SPICE

Per tal de dur a terme la simulació dels efectes de guitarra s'ha escollit el llenguatge SPICE. Aquest és un llenguatge de programació desenvolupat a la Universitat de Berkeley l'any 1973 per Donald O. Pederson i Laurence W. Nagel. Està enfocat sobretot a la simulació de circuits analògics i el que el fa tan interessant per a aquest projecte de final de grau és que és programari lliure, per tant, no requereix una llicència activa per poder-se utilitzar.

Amb el llenguatge SPICE podem simular una gran quantitat de circuits electrònics, ja que disposa d'una gran quantitat de components integrats, a més, en cas que no tingui el component que necessitem, SPICE permet crear un model propi o importar un desenvolupat per la comunitat.

SPICE, al cap i a la fi, és només un llenguatge i per ell mateix no fa res. Per tal de poder realitzar les simulacions necessàries caldria l'ús d'un intèrpret. Per tal de poder fer les simulacions s'han utilitzat dos intèrprets, NgSpice [7] per aprendre el llenguatge SPICE i per executar-lo a nivell de sistema i després el mòdul de Python PySpice per tal d'executar-lo a nivell d'aplicació.

Per aprendre el funcionament del llenguatge SPICE s'ha utilitzat el llibre SPICE "A Guide to Circuit Simulation & Analysis Using PSpice"[8] i s'han seguit alguns dels exemples per tal d'aprendre el funcionament.

4.2.1 Els fitxers .CIR

El llenguatge SPICE utilitza fitxers .CIR per tal de dur a terme les simulacions. En aquest fitxer és on es descriu el circuit a simular, així com el tipus de simulació que es farà a terme, condicions especials, etc.

L'estructura general d'un fitxer .CIR és la següent:

Nom de fitxer
//Definició de components
Component1 NodeX NodeX Valor
Component2 NodeX NodeX Valor
...
ComponentX NodeX NodeX Valor
.control
// A l'apartat control escriurem les comandes que vulguem que s'executin automàticament en fer l'anàlisi
// poden ser per exemple imprimir per pantalla la tensió en un determinat node del circuit,
// fer un anàlisi transitori i guardar els resultats, etc.
.endc
.end

Taula 4.1: Components bàsics en SPICE

Component	**Nomenclatura**	**Unitat**	**Exemple**
Resistència	RXXX	Ohm	R1 1 2 3k
Condensador	CXXX	Farad	C3 12 3 1.8u
Inductor	LXXX	Henry	L3 7 6 10m
Font de tensió	VXXX	Volts	V1 3 0 1.2k
Font de corrent	IXXX	Amper	I4 12 2 15m

4.3 Python

Per a dur a terme aquest projecte s'ha escollit com a llenguatge de programació Python[9], més concretament la versió 3. Aquest llenguatge és el que ens permetrà implementar tota la plataforma de simulació, així com la pàgina web desenvolupada.

Figura 4.3: Logo de Python[9]

Avantatges de Python:

- **Codi Obert i multiplataforma.**
  Python és un llenguatge de programació de codi obert, així que és idoni per al projecte. També disposa d'una gran comunitat molt implicada en el desenvolupament del projecte. Finalment, Python és multiplataforma, així que és compatible tant amb Sistemes Operatius Windows, Linux o iOS.
- **Gran quantitat de mòduls desenvolupats.**
  Un altre dels avantatges de l'ús de Python en el projecte és la gran quantitat de mòduls desenvolupats per part de la comunitat. En ser un llenguatge d'alt nivell i molt fàcil d'implementar al llarg dels anys els usuaris han anat desenvolupant mòduls per tal de dotar de més funcionalitats al llenguatge. Aquests mòduls s'instal·len de manera molt senzilla mitjançant el gestor de paquets PIP.

Figura 4.4: Logo del módul de Python Numpy[10]

Desavantatges de Python:

- **Consum de recursos**
  Python és un llenguatge que per culpa del seu funcionament tindrà tendència a utilitzar una gran quantitat de recursos dels quals tingui a l'abast. Això implica si el codi no està optimitzat l'ordinador, pugui arribar a quedar-se sense memòria disponible.
- **Dificultat del llenguatge**
  En disposar d'una gran quantitat de mòduls desenvolupats per part de la comunitat. Cada usuari al final sempre té la seva pròpia manera d'entendre la programació. Això comporta que a vegades utilitzar el llenguatge sigui complicat i s'hagin de visitar la documentació del mòdul

A continuació es farà una explicació dels mòduls més importants a l'hora de desenvolupar el projecte.

4.3.1 Mòdul PySpice

El mòdul PySpice[11] és el que utilitzarem per tal d'implementar el simulador basat en NgSpice en Python.

```
/* #Per tal d'instal·lar el mòdul PySpice introduirem la següent comanda
al terminal de la raspberry
*/
pip install PySpice
```

Figura 4.5: Logo del mòdul de Python PySpice[11]

Un cop tinguem el mòdul PySpice instal·lat realitzar una simulació és bastant senzill, només hem de definir el circuit amb format string i inicialitzar una instància de simulació. Ara es mostrarà quin és el procediment a seguir.

```
/*
Importem els moduls necessaris
*/
from PySpice.Spice.NgSpice.Shared import NgSpiceShared
from PySpice.Probe.Plot import plot
from PySpice.Unit import *
/*
Iniciem una instancia de simulació
*/
ngspice = NgSpiceShared.new_instance()
/*
Definim el circuit que volem simular
*/
circuit= "DEFINICIO DEL CIRCUIT DESITJAT"
/*
Carreguem el nostre circuit al simulador i correm la simulació
*/
ngspice.load_circuit(circuit)
ngspice.run()
```

4.3.2 Mòdul PyLTSpice

PyLTSpice[12] és un mòdul de Python bastant senzill i petit orientat al tractament de dades contingudes en fitxers .RAW obtinguts durant les simulacions amb SPICE.

```
/* #Per tal d'instal·lar el mòdul PyLTSpice introduirem la següent comanda
al terminal de la raspberry
*/
pip install PyLTSpice
```

Un cop ja tinguem el mòdul instal·lat, ja podrem realitzar el tractament de dades segons calgui. Ara es mostra un exemple de funcionament.

```
/*
Importem els moduls necessaris
*/
from PyLTSpice.LTSpice_RawRead import LTSpiceRawRead
/*
Llegim el fitxer amb les dades de la simulació feta
*/
LTR = LTSpiceRawRead("fitxer.raw")
/*
Obtenim les dades que necessitem i les salvem
*/
print(LTR.get_trace_names())
print(LTR.get_raw_property())
variable1 = LTR.get_trace("v(nodeSpice1)")
variable2= LTR.get_trace("v(nodeSpice2)")
/*
Imprimim per pantalla una figura per tal de visualitzar les dades que hem obtingut
*/
x = LTR.get_trace("time")
steps = LTR.get_steps()
for step in range(len(steps)):
    plt.plot(x.get_time_axis(step), vo.get_wave(step), label=steps[step])
plt.legend()
plt.show()
```

4.3.3 Mòdul PyWebIo

Figura 4.6: Logo del mòdul de Python PyWebIO[13]

PyWebIO[13] és el mòdul que s'ha escollit per realitzar tota la part de disseny web del projecte. És un mòdul que permet implementar una pàgina web utilitzant Python en comptes d'HTML (encara que continua sent possible implementar blocs de codi HTML).

El principal avantatge d'utilitzar Python amb el mòdul PyWebIO com a llenguatge principal de disseny web en comptes d'HTML és que és molt més intuïtiu i senzill. A més ens permet realitzar un disseny per a l'usuari molt senzill i eficaç

Dins de PyWebIO podem trobar dos grans apartats

- **PyWebIO.Input**

  Pywebio.Input fa referència a totes aquelles funcions que permeten a l'usuari enviar dades cap al nostre servidor. Disposa d'una gran quantitat de recursos per tal que l'usuari pugui introduir les dades que se li sol·liciten. A continuació es veurà alguns exemples.

```
1  /*
2  Importem el mòdul pywebio.input i la funció per iniciar el servidor
3  */
4  from pywebio.input import *
5  from pywebio import start_server
6  /*
7  Definirem una funció que permetrà a l'usuari introduir una edat
8  i comprovarà si aquesta està entre 10 i 30.
9  En cas que l'edat introduïda es trobi en aquesta franja el programa continuarà sense problema.
10 */
11 def main():
12     def check_age(age):
13         if age>30:
14             return "Too old"
15         elif age<10:
16             return "Too young"
17     input("Input your age", type=NUMBER, validate=check_age)
```

```
18
19  start_server(main, port=8080, debug=True)
```

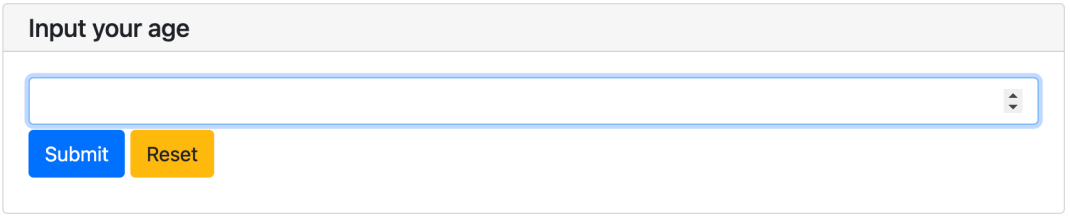

Figura 4.7: Exemple d'introducció de dades en format text

A part d'entrades de dades en format text també podem oferir a l'usuari unes opcions predeterminades, per fer-ho tenim diferents tipus de format, podem fer-ho amb desplegables, o oferint les opcions.

```
1  /*
2  Importem el mòdul pywebio.input i la funcio per iniciar el servidor
3  */
4  from pywebio.input import *
5  from pywebio import start_server
6  /*
7  Crearem un seleccionable que permeti a l'usuari escollir entre 2 opcions
8  */
9  def main():
10     selecció = input.select("Selecciona la opció que vulguis", ["Opcio1", "Opcio2"])
11 start_server(main, port=8080, debug=True)
```

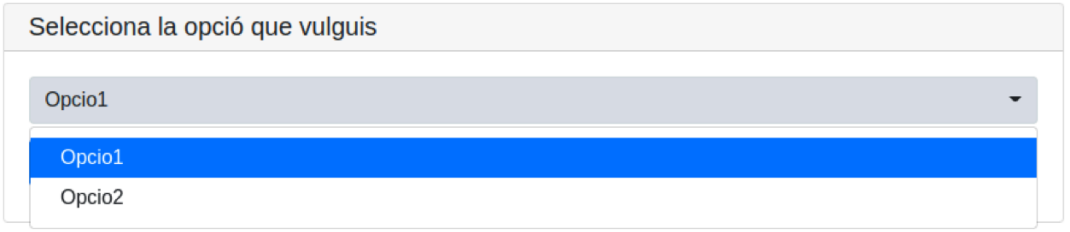

Figura 4.8: Exemple d'introducció de dades en format selecció

Finalment, una de les funcions d'entrada que es veurà és la pujada de fitxers. Aquesta funció permetrà a l'usuari carregar el fitxer que vulgui i enviar-lo al nostre servidor. Com a particularitat aquesta funció retorna un diccionari de Python amb els següents camps: Nom d'arxiu, contingut del fitxer en bytes,

"MIME type" del fitxer i última data de modificació. Com el contingut del fitxer està en format bytes potser hem de realitzar algun tipus de tractament de dades segons el que vulguem fer. A continuació es mostra un exemple en el qual es demana a l'usuari que pugi un arxiu .WAV. Aquest fitxer el llegirem i el salvarem al nostre servidor.

```
/*
Importem el mòdul pywebio.input i la funcio per iniciar el servidor
*/
from pywebio.input import *
from pywebio import start_server
/*
Crearem un seleccionable que permeti a l'usuari escollir entre 2 opcions
*/
def main():
    contingut_fitxer = file_upload("Puja un fitxer",multiple=False, required=True)
    f=open("/home/pi/TFG/WEB/USER_WAV/fitxer_usuari.wav", "wb")
    f.write(contingut_fitxer["content"])
    f.close()
start_server(main, port=8080, debug=True)
```

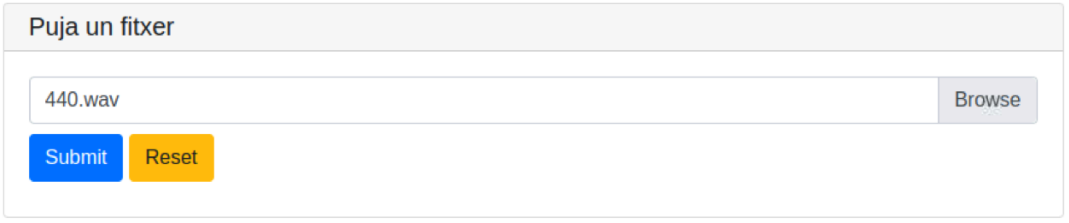

Figura 4.9: Exemple de pujada d'un fitxer amb PyWebIO

- **PyWebIO.Output**

  PyWebIO.Output fa referència a totes aquelles funcions que permeten enviar contingut de sortida a la pàgina web. Aquest contingut pot ser en format text, d'imatge, fitxer, etc. A més el podem mostrar de diverses formes Disposem de 4 tipus de funcions dins de PyWebIO.Output per mostrar el contingut Web.

  – Funcions de gestió de finestra

    PyWebIO mostra el contingut en "finestres", així que per tal que es pugui organitzar la informació de la manera que es vulgui, es proporcionen funcions que ens faciliten la feina. A continuació veurem algunes de les funcions més importants

    La principal funció d'aquest apartat seria put'_'scope. Aquesta funció ens permet crear una nova

finestra de continguda la nostra pàgina web. Un exemple de com utilitzar-la seria el següent

```
/*
Importem el mòdul pywebio.output i la funció per iniciar el servidor
*/
from pywebio.output import *
from pywebio import start_server
/*
Crearem una finestra amb el text Prova de finestra
*/
def main():
    put_scope("Finestra1",put_text("Prova de finestra"))
start_server(main, port=8080, debug=True)
```

Un cop hem creat aquesta finestra podríem utilitzar les altres funcions de gestió de finestra segons les nostres necessitats

```
/*
clear permet eliminar el contingut de la finestra que desitgem
*/
clear("Finestra1")
 /*
remove ens permet eliminar la finestra que vulguem
*/
remove("Finestra1")
```

– Funcions de contingut

Dins de les funcions de contingut trobaríem algunes de les funcions que ens permeten enviar contingut a la nostra pàgina web, podem enviar qualsevol mena de contingut, com per exemple, text, imatges, fitxers, animacions, etc. A continuació es veuran alguns exemples d'enviament de contingut.

Suposem que tenim una plana web amb 3 finestres, i en cadascuna volem enviar una cosa diferent, a la finestra 1 volem enviar un text, a la segona una imatge i a la tercera volem deixar un enllaç per descarregar un fitxer. El codi sería el següent

```
/*
Importem el mòdul pywebio.output i la funció per iniciar el servidor
*/
```

```
4  from pywebio.output import *
5  from pywebio import start_server
6  /*
7  Crearem una finestra amb el text Prova de finestra
8  */
9  def main():
10 /*
11 Creem les nostres finestres amb la funció put_scope vista anteriorment
12 */
13     put_scope("Finestra1")
14     put_scope("Finestra2")
15     put_scope("Finestra3")
16     put_text("Text a la primera finestra",scope="Finestra1")
17     put_image("Directori de la imatge",scope="Finestra2")
18     put_file("NomArxiu.txt",contingut_fitxer,"Fes click per descarregar",
19     scope="Finestra3")
20 start_server(main, port=8080, debug=True)
```

– Altres interaccions

Dins de les funcions d'altres interaccions trobem totes aquelles destinades principalment a notificar a l'usuari d'una manera més eficaç. Aquí trobaríem per exemple les funcions per tal d'implementar notificacions i popups. Ara es mostrarà un exemple d'un popup que mostraria un codi HTML, permetria descarregar un fitxer i disposaria d'un botó per tancar el popup.

```
1  /*
2  Importem el mòdul pywebio.output i la funció per iniciar el servidor
3  */
4  from pywebio.output import *
5  from pywebio import start_server
6  /*
7  Crearem una finestra amb el text Prova de finestra
8  */
9  def main():
10 /*
11 Creem el popup amb el contingut desitjat
12 */
13     popup("Simulació Realitzada amb èxit!", [
14         put_html("<h3>Gracies per utilitzar el simulador</h3>"),
15         put_html("<h4>En el següent enllaç podràs descarregar
```

```
16          la teva simulació</h4>"),
17          put_file("Simulacio.wav", content, "Fes click per descarregar!"),
18          put_button("Tanca", onclick=close_popup)
19      ])
20  start_server(main, port=8080, debug=True)
```

Figura 4.10: Exemple de popup amb PyWebIO

– Estil

Finalment, ens trobem amb les funcions d'estil. Aquestes ens permeten modificar l'estil de la nostra pàgina aplicant un arxiu CSS si volguéssim. Així com mostrar el contingut en forma de barra o columna específicament. Suposem que a la nostra plana tenim dues Finestres, en una volem mostrar contingut en barra i en l'altre en columna. Es faria de la següent manera:

```
1   /*
2   Importem el mòdul pywebio.output i la funció per iniciar el servidor
3   */
4   from pywebio.output import *
5   from pywebio import start_server
6   /*
7   Crearem una finestra amb el text Prova de finestra
8   */
9   def main():
10  /*
11  Creem les nostres finestres amb la funció put_scope vista anteriorment
12  */
13      put_scope("Finestra1")
14      put_scope("Finestra2")
15  /*
16  Enviem el contingut a les dues finestres
17  */
```

```
18        with use_scope("Finestra1"):
19            put_row([put_code('A'), None, put_code('B')])
20        with use_scope("Finestra2"):
21            put_column([put_code('A'), None, put_code('B')]")
22
23    start_server(main, port=8080, debug=True)
```

Capítol 5

Desenvolupament del simulador

Un cop vistes les eines amb les quals es durà a terme el projecte, en aquest apartat s'explicarà quina ha sigut l'estructura de pàgina web proposada i com s'han implementat cadascuna de les parts que la formen.

5.1 Estructura de la plataforma de simulació

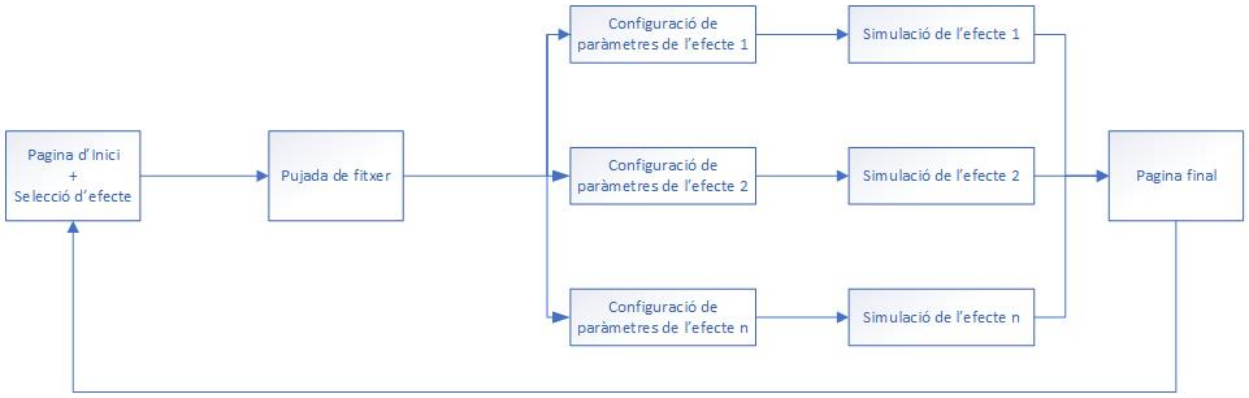

Figura 5.1: Estructura de la pàgina web a desenvolupar. Elaboració pròpia.

Com podem veure en el diagrama de flux de la figura 5.4 l'estructura de la pàgina web a desenvolupar és molt senzilla, l'usuari en entrar es troba amb una pàgina de benvinguda on es presenta el projecte. Des d'aquesta pàgina accedirem al simulador com tal, on l'usuari podrà escollir quin dels efectes disponibles voldrà simular. Un cop escollit quin és l'efecte a simular haurà d'escollir quins són els paràmetres amb els quals vol realitzar la simulació, i un cop hagi fet la tria de paràmetres es durà a terme la simulació. En acabar la simulació un popup amb l'enllaç de descàrrega i es redirigirà a l'usuari a una pàgina final on podrà consultar informació sobre la simulació realitzada i podrà tornar a la pàgina d'inici en cas que vulgui realitzar una altra simulació.

5.2 Simulació

Al moment de desenvolupar el projecte es va decidir començar duent a terme les simulacions d'un efecte i analitzar els resultats. Es va decidir per aquesta part per tal de garantir que el projecte era viable. Es va començant simulant l'efecte amb el simulador NgSpice, i un cop ja es va veure que les simulacions eren fiables, es va implementar amb el llenguatge Python on també es va comprovar que era possible realitzar les modificacions del circuit i que també es podia introduir un fitxer .WAV proporcionat per l'usuari com a font.

5.2.1 Simulació del Dallas Rangemaster en NgSpice

L'objectiu d'aquesta simulació és la de verificar que la simulació amb el llenguatge SPICE és correcta. Per aquest motiu es va decidir fer que V1 fos un senyal sinusoidal de 0.5mV amb una freqüència de 440Hz.
A continuació es pot observar com seria el circuit de l'efecte Dallas RangeMaster en format SPICE.

```
1  Dallas RangeMaster
2  .MODEL OC44 PNP(IS=1.423u BF=307.0 BR=20.27 NF=1.022 NR=1.025 VT=25.5m VAF=8.167 VAR=14.84
3  IKF=43.82m IKR=611.7m ISE=30.54n ISC=213.5n NE=1.316 NC=1.258 RB=32.83 RE=968.7m RC=989.9u
4  CEB=410p CCB=10p)
5  .PARAM RPOT=10K SET=1
6  VCC 100 0 DC 9V
7  V1 100 101 DC 0 AC 1 SIN(0 500m 440Hz)
8  C1 intb 101 5n
9  R1 0 intb 470k
10 R2 100 intb 68k
11 XPOT1 0 intc out POT RPOT={RPOT} SET={SET}
12 Q1 intc intb SortTran OC44
13 R3 100 SortTran 3.9k
14 C3 100 SortTran 47u
15 C2 out Sortida 10n
16 RC Sortida 100 100k
17
18 .SUBCKT POT 1 T 2 RPOT=10k SET=0.5
19 RT 1 T {RPOT*SET+0.001}
20 RB T 2 {RPOT*(1-SET)+0.001}
21 .ENDS
22 .CONTROL
23 RUN
24 .ENDC
25 .END
```

Un cop ja tenim el circuit en un fitxer .CIR es pot procedir a realitzar la simulació. Per tal de verificar que els resultats de simulació són correctes. Es prendrà com a referència el model matemàtic desenvolupat a l'apartat 3.1.3. A continuació es mostra una comparativa dels resultats obtinguts després de realitzar les simulacions amb ambdós programes.

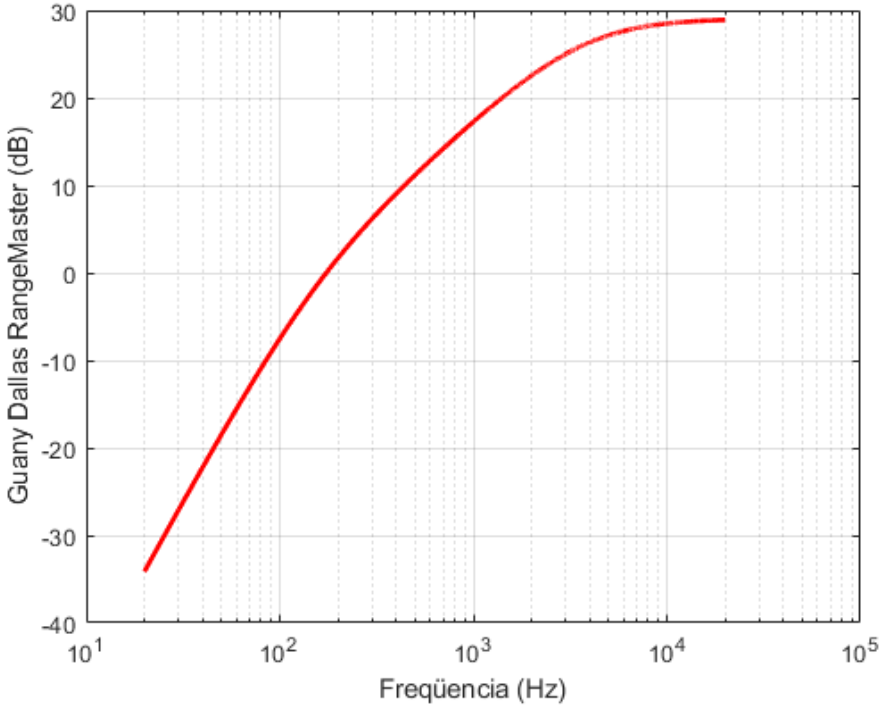

Figura 5.2: Simulació amb el model matemàtic. Elaboració pròpia.

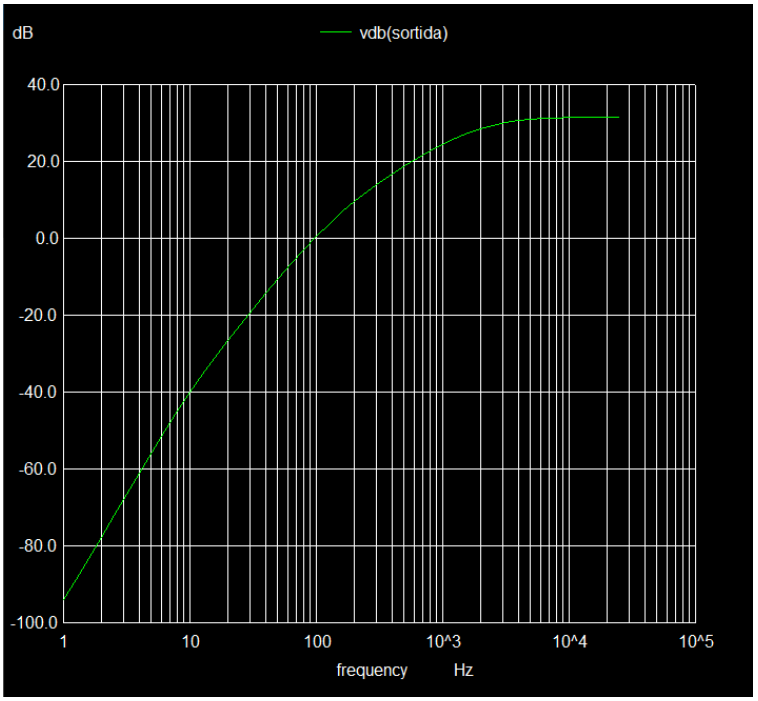

Figura 5.3: simulació obtinguda amb NgSpice. Elaboració pròpia.

Després d'aquesta comparació podem veure que els resultats de simulació són correctes, i que, per tant, el simulador PySpice amb Python són correctes. El següent pas és fer que el senyal d'entrada deixi de ser el senyal sinusoidal definit al circuit i passi a ser un senyal .WAV proporcionat per l'usuari.
Com a primera opció es va contactar amb un dels desenvolupadors del simulador NgSpice per veure si hi havia alguna manera d'introduir un fitxer .WAV com a font en el circuit. El desenvolupador va proporcionar l'enllaç a una versió de NgSpice amb capacitat per tractar fitxers d'àudio, així que es va procedir a instal·lar aquesta versió.
Un cop instal·lada aquesta versió es van realitzar proves de la nova versió de NgSpice, però no es va aconseguir que funciones de manera correcta. A causa d'això es va descartar utilitzar aquesta solució per a introduir un fitxer d'àudio com a font de senyal del circuit.

5.2.2 Implementació en Python del simulador

Vist que la simulació amb SPICE és viable, ara s'ha d'implementar amb Python. Com s'ha comentat anteriorment s'utilitzaran el mòdul PySpice per dur a terme les simulacions i LTSpice per analitzar les dades dels fitxers .RAW.

Implementació del circuit amb Python amb PySpice:

El primer pas per dur a terme la implementació en Python és implementar el circuit del Dallas RangeMaster en Python amb el mòdul PySpice, com es va veure a l'apartat 4.3.1 s'implementaria de la següent forma:

```
import PySpice.Logging.Logging as Logging
logger = Logging.setup_logging()
from PySpice.Spice.NgSpice.Shared import NgSpiceShared
from PySpice.Probe.Plot import plot
from PySpice.Unit import *

circuit = '''
*==============  Begin SPICE netlist of main design ============

Circuit del pedal Dallas RangeMaster
.MODEL OC44 PNP(IS=1.423u BF=307.0 BR=20.27 NF=1.022 NR=1.025 VT=25.5m VAF=8.167 VAR=14.84
IKF=43.82m IKR=611.7m ISE=30.54n ISC=213.5n NE=1.316 NC=1.258 RB=32.83 RE=968.7m RC=989.9u
CEB=410p CCB=10p)
.PARAM RPOT=10K SET=1

VCC 100 0 DC 9V
V1 100 101 DC 0 AC 1 SIN(0 500m 440Hz)

C1 intb 101 5n
```

```
R1 0 intb 470k
R2 100 intb 68k
XPOT1 0 intc out POT RPOT={RPOT} SET={SET}
Q1 intc intb SortTran OC44
R3 100 SortTran 3.9k
C3 100 SortTran 47u
C2 out Sortida 10n
RC Sortida 100 100k

.SUBCKT POT 1 T 2 RPOT=10k SET=0.5
RT 1 T {RPOT*SET+0.001}
RB T 2 {RPOT*(1-SET)+0.001}
.ENDS

.CONTROL
RUN
.ENDC
.END

'''
ngspice = NgSpiceShared.new_instance()
ngspice.load_circuit(circuit)
ngspice.run()
```

Introducció d'un fitxer .WAV com a senyal d'entrada:

Com s'ha comentat a l'apartat anterior, no es va considerar viable introduir el senyal en format .wav a través del simulador NgSpice, així que un dels objectius la implementació amb Python és la de substituir el senyal d'entrada sinusoidal per un senyal en format .WAV proporcionat per l'usuari.

Navegant pel fòrum del projecte NgSpice es va trobar una entrada on el desenvolupador que va proporcionar la versió de NgSpice pel tractament de fitxers d'àudio proporcionava un enllaç al GitHub d'un de l'usuari Henrik Forstén[14] que havia desenvolupat en Python dos scripts que permetien la conversió d'arxius .WAV al format de font de tensió d'SPICE i a la inversa.

Es va decidir provar si el funcionament d'aquests scripts integrant-los en la simulació del circuit. Per fer-ho es va haver de buscar una forma d'introduir un fitxer extern com a una font dins del mateix circuit.

Com s'ha comentat a l'apartat 4.2 a SPICE podem importar un component del qual no es disposa, així que importarem un component que serà una font de tensió externa, i a continuació, s'introduirà al circuit.

En el següent codi podem veure com s'importaria una font de tensió provinent d'un arxiu .txt i com s'hauria d'introduir en el circuit.

```
.model filesrc filesource (file="fitxer.txt" amploffset=[0] amplscale=[1.0] timeoffset=0
timescale=1 timerelative=false amplstep=false)

 A1 %v[node] filesrc
```

Un cop ja l'hem introduït al nostre circuit, ara tocaria que després de fer la simulació els valors de la sortida se salvessin i finalment s'exportessin a un fitxer que després es passarà al format .WAV.
A continuació es mostrarà un exemple de com es realitzaria el tractament de dades i com es convertiria al format .WAV.

```
        wrdata trdata v(node)
        write trdata.raw

        os.system("/PATH/spicetowav.py trdata.raw /PATH/Audio_Simulat.wav")
```

Vist com s'introduiriria un fitxer extern com a font de tensió del circuit, el diagrama de blocs a implementar seria el següent.

Figura 5.4: Procés de simulació amb un fitxer WAV extern. Elaboració pròpia.

I el codi per tal d'introduir un fitxer d'àudio com a font de tensió en el circuit del Dallas RangeMaster és el següent.

```
os.system("/home/pi/TFG/Python/wavtospice.py /home/pi/TFG/WEB/USER_WAV/fitxer_usuari.wav
nes.txt")
circuit = '''
*==============  Begin SPICE netlist of main design ============
.model filesrc filesource (file="nes.txt" amploffset=[0] amplscale=[1.0]
timeoffset=0 timescale=1 timerelative=false amplstep=false)
.MODEL OC44 PNP(IS=1.423u BF=307.0 BR=20.27 NF=1.022 NR=1.025 VT=25.5m
VAF=8.167 VAR=14.84 IKF=43.82m IKR=611.7m ISE=30.54n ISC=213.5n NE=1.316
NC=1.258 RB=32.83 RE=968.7m RC=989.9u CEB=410p CCB=10p)

.PARAM RPOT=10K SET=1

```

```
VCC 100 0 DC 9V
A1 %v[101] filesrc
C1 intb 101 5n
R1 0 intb 470k
R2 100 intb 68k
XPOT1 0 intc out POT RPOT={RPOT} SET={SET}
Q1 intc intb SortTran OC44
R3 100 SortTran 3.9k
C3 100 SortTran 47u
C2 out Sortida 10n
RC Sortida 0 100k
.SUBCKT POT 1 T 2 RPOT=10k SET=0.5
RT 1 T {RPOT*SET+0.001}
RB T 2 {RPOT*(1-SET)+0.001}
.ENDS
.control
save v(sortida)
tran 20.833u 9s
run
wrdata trdata v(sortida)
write trdata.raw
.endc
.END
'''
ngspice = NgSpiceShared.new_instance()
ngspice.load_circuit(circuit)
ngspice.run()

os.system("/PATH/spicetowav.py trdata /PATH/Audio_Simulat.wav")
os.system("rm /home/pi/TFG/WEB/nes.txt")
os.system("rm /home/pi/TFG/WEB/trdata.raw")
os.system("rm /home/pi/TFG/WEB/trdata")
os.system("rm /home/pi/TFG/AudioSimulat/Audio_Simulat.wav")
```

Modificació dels components del circuit, així com dels seus valors:

Un dels altres objectius d'aquest projecte era permetre que l'usuari modifiqués el circuit segons les seves necessitats. En treballar amb Python això ho podem dur a terme d'una manera molt senzilla.

Com vam veure a l'apartat 4.3.1 definim el circuit en un format string, això ens permet utilitzar els mètodes propis de la classe, en aquest cas, utilitzarem el mètode "replace()" que ens permetrà substituir una string

per una de nova. Aleshores per tal de modificar el component que vulguem, s'hauria de crear una nova string amb el mateix format, però amb el valor canviat i substituir-la per l'original. A continuació es mostrarà un exemple de com modificar el valor del condensador C_1 del Dallas RangeMaster.

```
valorC1 = radio("Quin valor en nanoFarads vols que tingui el condesadorC1 ?",
options=["1", "5", "10"])
nou_condensadorC1 = "C1 intb 101 "+valorC1+"n"
os.system("/home/pi/TFG/Python/wavtospice.py /home/pi/TFG/WEB/USER_WAV/fitxer_usuari.wav
nes.txt")
circuit = '''
*==============  Begin SPICE netlist of main design ============
.model filesrc filesource (file="nes.txt" amploffset=[0] amplscale=[1.0]
timeoffset=0 timescale=1 timerelative=false amplstep=false)
.MODEL OC44 PNP(IS=1.423u BF=307.0 BR=20.27 NF=1.022 NR=1.025 VT=25.5m
VAF=8.167 VAR=14.84 IKF=43.82m IKR=611.7m ISE=30.54n ISC=213.5n NE=1.316
NC=1.258 RB=32.83 RE=968.7m RC=989.9u CEB=410p CCB=10p)

.PARAM RPOT=10K SET=1

VCC 100 0 DC 9V
A1 %v[101] filesrc
C1 intb 101 5n
R1 0 intb 470k
R2 100 intb 68k
XPOT1 0 intc out POT RPOT={RPOT} SET={SET}
Q1 intc intb SortTran OC44
R3 100 SortTran 3.9k
C3 100 SortTran 47u
C2 out Sortida 10n
RC Sortida 0 100k
.SUBCKT POT 1 T 2 RPOT=10k SET=0.5
RT 1 T {RPOT*SET+0.001}
RB T 2 {RPOT*(1-SET)+0.001}
.ENDS
.control
save v(sortida)
tran 20.833u 9s
run
wrdata trdata v(sortida)
```

```
36  write trdata.raw
37  .endc
38  .END
39  '''
40  circuit = circuit.replace("C1 intb 101 5n", nou_condensadorC1)
41  ngspice = NgSpiceShared.new_instance()
42  ngspice.load_circuit(circuit)
43  ngspice.run()
```

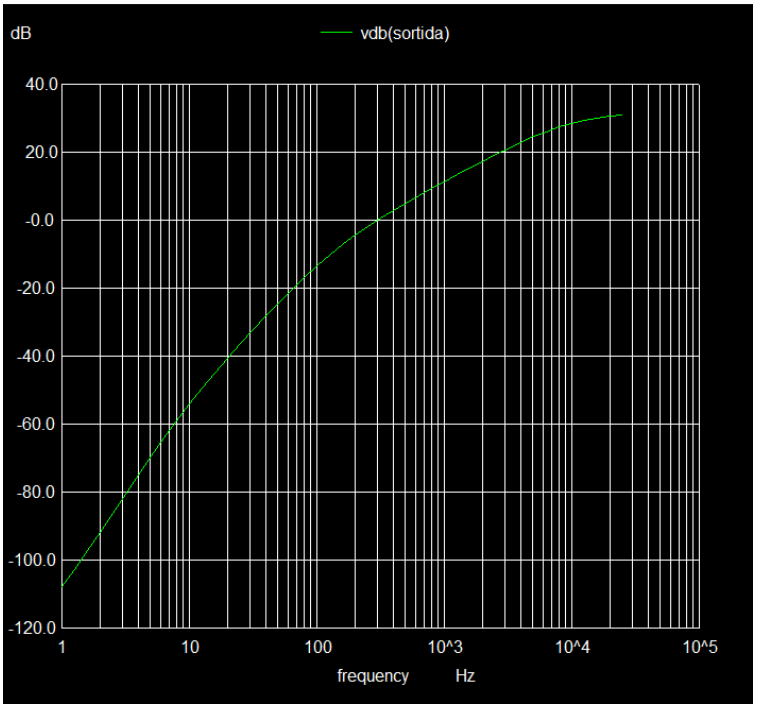

Figura 5.5: Simulació obtinguda amb el valor C1 = 1nF. Elaboració pròpia.

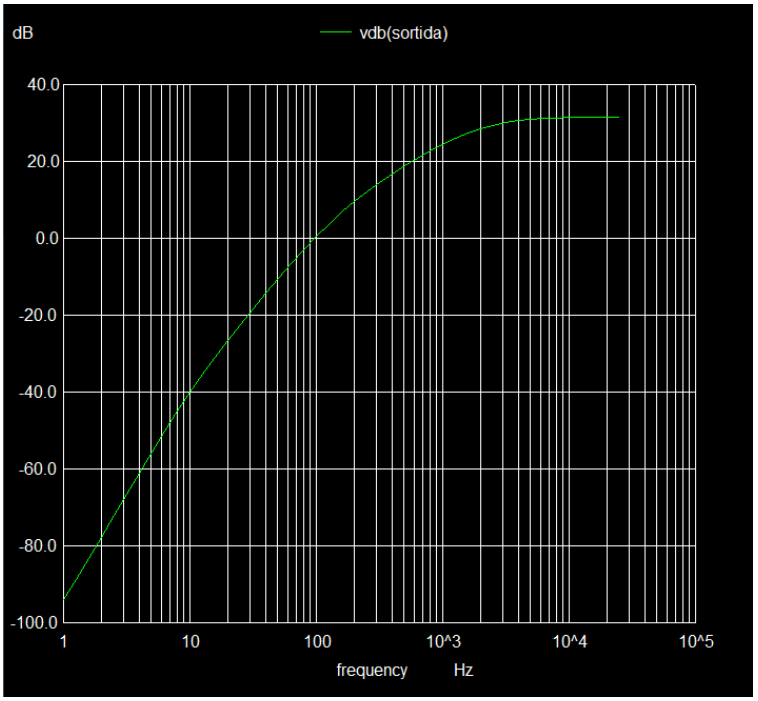

Figura 5.6: Simulació obtinguda amb el valor C1 = 5nF. Elaboració pròpia.

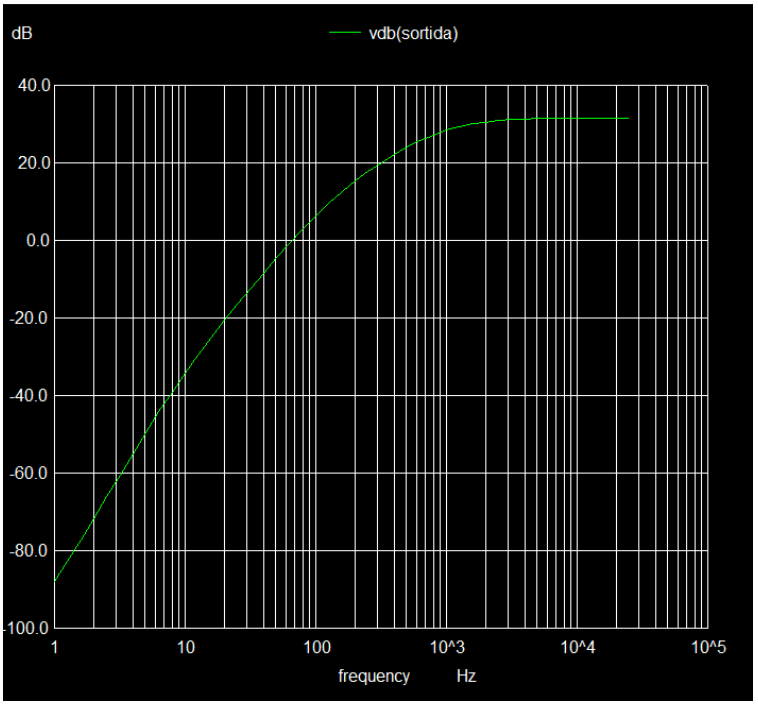

Figura 5.7: Simulació obtinguda amb el valor C1 = 10nF. Elaboració pròpia.

Un cop ja s'ha fet la implementació en Python del simulador, s'ha permès que un fitxer .WAV sigui la font de tensió d'entrada i s'ha aconseguit modificar de manera correcta els components del circuit, la implementació del simulador en Python es pot donar per tancada i es pot continuar amb la implementació del simulador a la pàgina Web.

5.2.3 Implementació en Python de la pàgina Web

- **Pàgina d'Inici i selecció d'efecte:**

  La pàgina web comença d'una manera molt simple, mostra el títol de la plana web i demana a l'usuari que seleccioni l'efecte que vol simular. El codi Python per implementar això és el següent.

```
put_row(put_html('<h1>Plataforma per a la simulació d\'efectes de guitarra</h1>'))
put_scope("seleccio")
with use_scope("seleccio"):
    pedal = input.select('Selecciona el pedal que vulguis',['Dallas Rangemaster',
    'Fuzz Face'])
```

El resultat seria el següent:

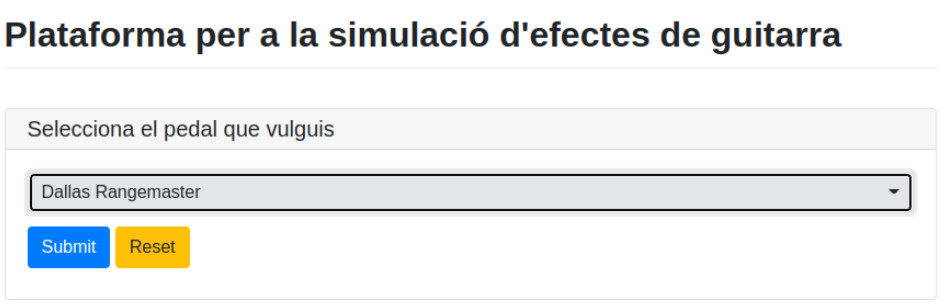

Figura 5.8: Pàgina web d'inici. Elaboració pròpia.

- **Mostra d'informació sobre l'efecte, introducció de paràmetres i simulació:**

  Un cop l'usuari ja hagi seleccionat l'efecte que vol simular, se li mostrarà una imatge de l'esquema de l'efecte i també una llista dels elements modificables del circuit, així com una descripció de com afecten les modificacions. A més just a la part de sota de la informació, l'usuari anirà introduint els paràmetres necessaris per dur a terme la simulació. S'implementaria amb el següent codi.

```
if (pedal == "Dallas Rangemaster"):

    with use_scope("informacio"):
    img_dallas = open("/home/pi/TFG/Media/dallas_esq.png", "rb").read()
    put_row(put_html("<h2>Pedal Dallas RangeMaster</h2>"))
    put_row([
        put_column([
            put_text("Els elements modificables del Dallas RangeMaster són els següents"),
            put_text("C1: A més baix sigui el valor de C1, més greus tindrà el pedal"), None,
```

```
10              put_text("RV: Si augmenta el valor de RV podrem tenir un nivell més elevat
11              de guany"), None,
12              put_text("Posició de RV: Indica a quin nivell es col·loca el potenciòmetre
13              sent 1 el mínim i 0 el màxim"),
14              ]), None,
15          put_image(img_dallas)
16
17  ])
18          caracteristiques=puja_file()
19          simula_dallas(caracteristiques)
```

Funció puja_file()

Aquesta funció és la que ens permetrà rebre un fitxer .WAV per part de l'usuari.

```
1   def puja_file():
2
3       contingut_fitxer = file_upload("Puja un fitxer",multiple=False, required=True)
4       f=open("/home/pi/TFG/WEB/USER_WAV/fitxer_usuari.wav", "wb")
5       f.write(contingut_fitxer["content"])
6       sampling_rate, data=read_wav("/home/pi/TFG/WEB/USER_WAV/fitxer_usuari.wav")
7
8       tamany=len(data)
9       duracio=tamany/sampling_rate
10
11      parametres_fitxer = {
12        "fs": sampling_rate,
13        "duracio": duracio,
14          }
15      return parametres_fitxer
```

Funció simulla_dallas()

Aquesta funció és la que demanarà a l'usuari introduir els paràmetres necessaris i durà a terme la simulació.

```
1   def simula_dallas(diccionari_fitxer):
2       fs = diccionari_fitxer["fs"]
3       duracio = diccionari_fitxer["duracio"]
4       inversa_fs = 1/fs
5       nou_transitori = "tran "+str(inversa_fs)+"s "+str(duracio)+"s"
```

```
6
7       valor_pote = radio("De quants KiloOhms vols el potenciometre Rv?",
8       options=["1", "10", "100"])
9       nivell = radio("Quin nivell de potenciometre vols?",
10      options=["0.1", "0.33", "0.5", "1"])
11      valorC1 = radio("Quin valor en nanoFarads vols que tingui el condesadorC1 ?",
12      options=["5", "10", "15"])
13      nou_condensadorC1 = "C1 intb 101 "+valorC1+"n"
14
15      modificacio_nivell_pote = ".PARAM RPOT=10k SET="+nivell
16      modificacio_valor_pote = "RPOT="+valor_pote+"k"
17      nou_pote = modificacio_nivell_pote.replace("RPOT=10k", modificacio_valor_pote)
18      put_scope("SimulacioDallas")
19      with use_scope("SimulacioDallas"):
20          put_text("Realitzant la simulació")
21          with put_loading():
22              os.system("/home/pi/TFG/Python/wavtospice.py"
23              "/home/pi/TFG/WEB/USER_WAV/fitxer_usuari.wav nes.txt")
24              circuit = '''
25          *==============  Begin SPICE netlist of main design ============
26
27          .model filesrc filesource (file="nes.txt" amploffset=[0] amplscale=[1.0]
28          timeoffset=0 timescale=1 timerelative=false amplstep=false)
29          .MODEL OC44 PNP(IS=1.423u BF=307.0 BR=20.27 NF=1.022 NR=1.025 VT=25.5m
30          VAF=8.167 VAR=14.84 IKF=43.82m IKR=611.7m ISE=30.54n ISC=213.5n NE=1.316
31          NC=1.258 RB=32.83 RE=968.7m RC=989.9u CEB=410p CCB=10p)
32          .PARAM RPOT=10K SET=1
33
34          VCC 100 0 DC 9V
35
36          A1 %v[101] filesrc
37
38          * V1 100 101 DC 0 AC 1 SIN(0 500m 440Hz)
39
40          C1 intb 101 5n
41          R1 0 intb 470k
42          R2 100 intb 68k
43          XPOT1 0 intc out POT RPOT={RPOT} SET={SET}
44          Q1 intc intb SortTran OC44
```

```
45          R3 100 SortTran 3.9k
46          C3 100 SortTran 47u
47          C2 out Sortida 10n
48          RC Sortida 0 100k
49
50          .SUBCKT POT 1 T 2 RPOT=10k SET=0.5
51          RT 1 T {RPOT*SET+0.001}
52          RB T 2 {RPOT*(1-SET)+0.001}
53          .ENDS
54
55          .control
56          save v(sortida)
57          tran 20.833u 9s
58          run
59          * save v(Ni), V(No)E
60          wrdata trdata v(sortida)
61          write trdata.raw
62          .endc
63          .END
64          '''
65              circuit = circuit.replace(".PARAM RPOT=10K SET=1", nou_pote)
66              circuit = circuit.replace("tran 20.833u 9s", nou_transitori)
67              circuit = circuit.replace("C1 intb 101 5n", nou_condensadorC1)
68
69              ngspice = NgSpiceShared.new_instance()
70              ngspice.load_circuit(circuit)
71              ngspice.run()
72
73              os.system("/home/pi/TFG/Python//spicetowav.py trdata"
74              "/home/pi/TFG/AudioSimulat/Audio_Simulat.wav")
75              os.system("rm /home/pi/TFG/WEB/nes.txt")
76              os.system("rm /home/pi/TFG/WEB/trdata.raw")
77              os.system("rm /home/pi/TFG/WEB/trdata")
```

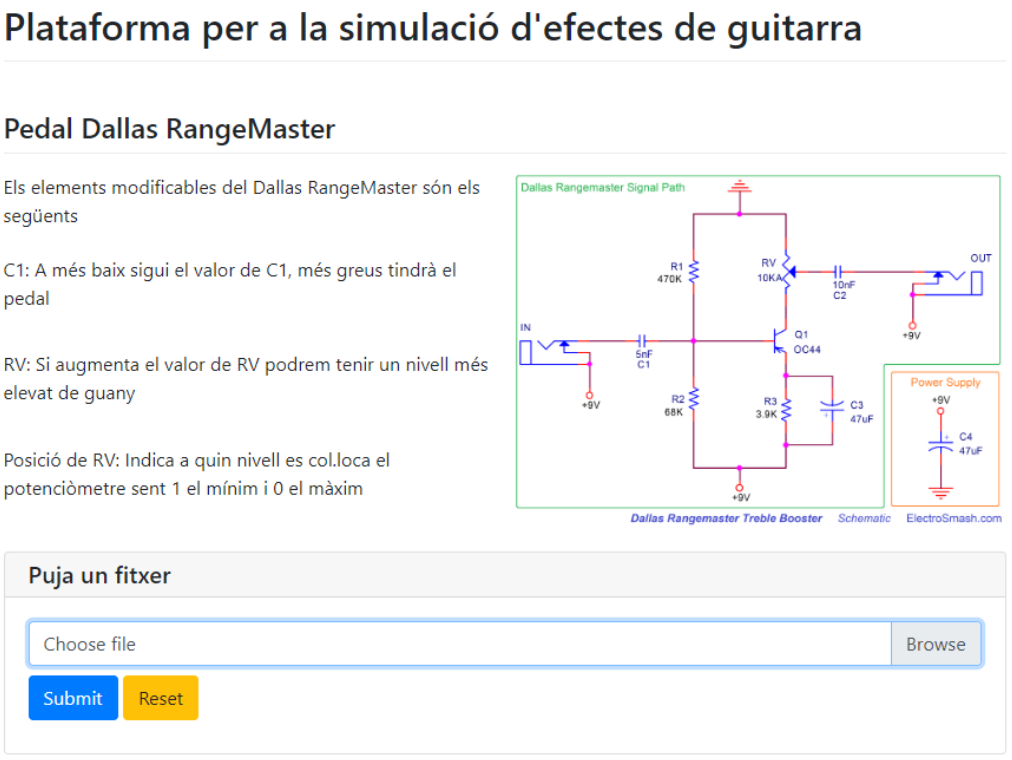

Figura 5.9: Pàgina web amb informació i quadre de pujada de fitxer. Elaboració pròpia.

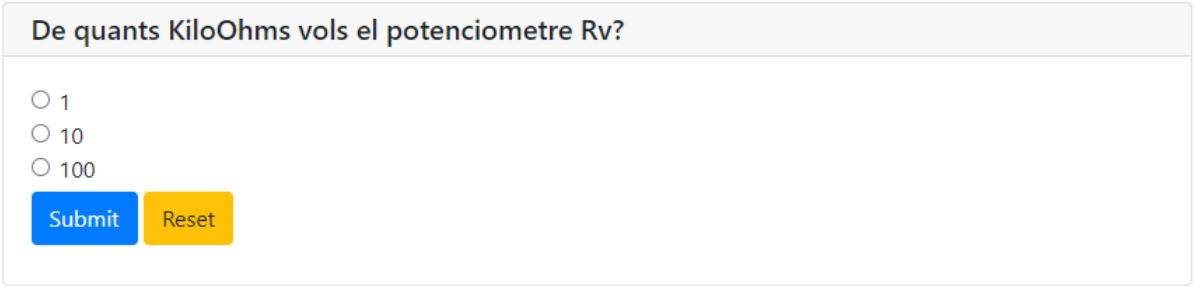

Figura 5.10: Selecció del valor del Potenciòmetre RV. Elaboració pròpia.

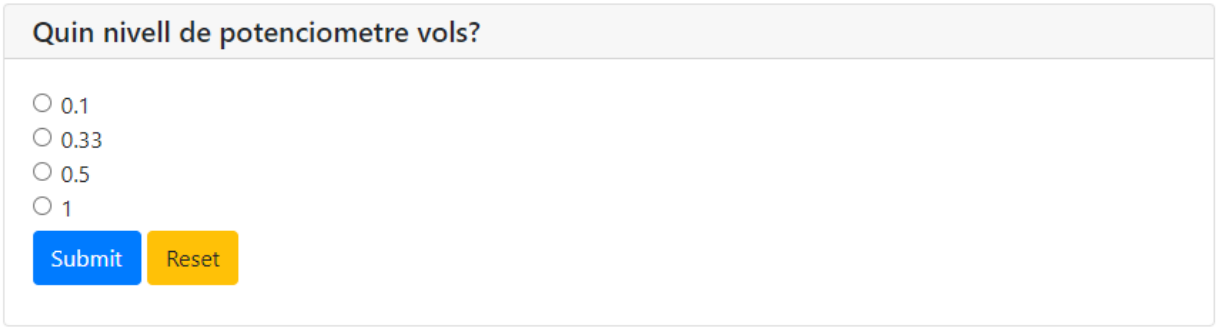

Figura 5.11: Selecció del nivell del Potenciometre RV. Elaboració pròpia.

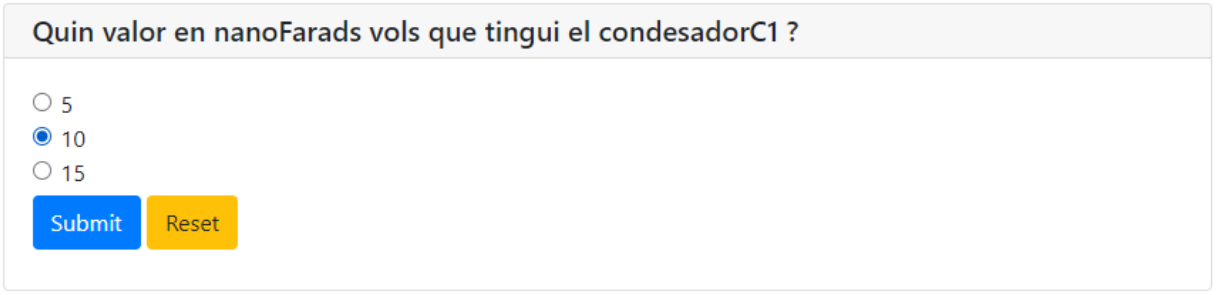

Figura 5.12: Selecció del valor del condensador C1. Elaboració pròpia.

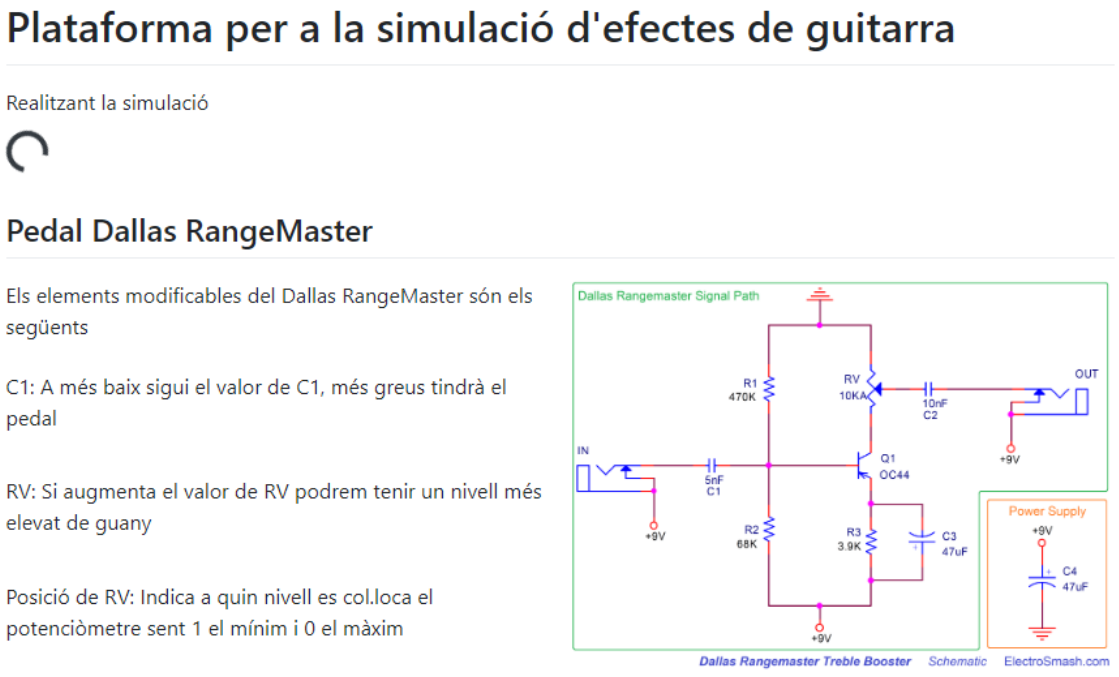

Figura 5.13: Pàgina Web durant la simulació. Elaboració pròpia.

- **Popup final i retorn a la pàgina principal**

  Un cop ja s'ha realitzat la simulació es mostrà un popup a l'usuari indicant que la simulació és correcte i proporcionant-li un enllaç per descarregar el fitxer.

  El codi per implementar aquesta funcionalitat es trobaria just al final de la funció simula_dallas el següent.

```
content = open("/home/pi/TFG/AudioSimulat/Audio_Simulat.wav', 'rb").read()


popup("Simulació Realitzada amb èxit!", [
    put_html("<h3>Gracies per utilitzar el simulador</h3>"),
    put_html("<h4>En el seguent enllaç podràs descarregar la teva simulació</h4>"),
    put_file("Simulacio.wav", content, "Fes click per descarregar!"),
    put_button("Tanca", onclick=close_popup)
    ])
        clear(scope="SimulacioDallas")
        put_text("Simulació realitzada amb èxit", scope="SimulaDallas")
```

  Un cop l'usuari tanqui el popup trobarà un botó que li permetrà tornar a la pàgina d'inici. El codi per implementar aquest botó es trobaria just a sota de la crida a la funció simula'_'dallas al main, el codi per implementar-lo seria el següent.

```
put_button("Torna a la pàgina d\'inici", onclick=lambda:run_js("window.location.reload()"))
```

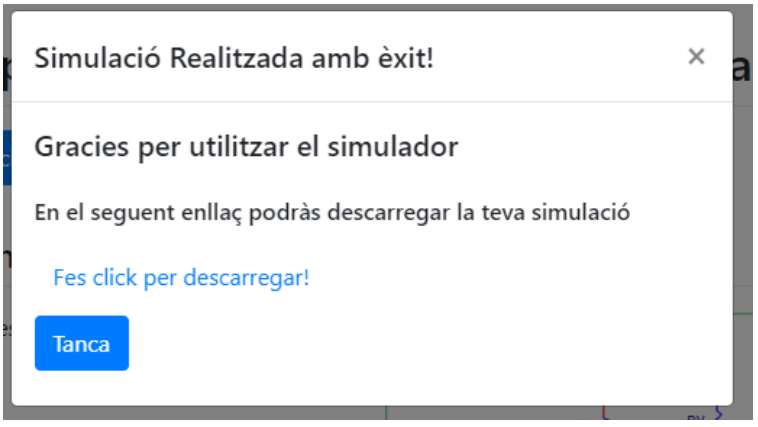

Figura 5.14: Popup en finalitzar la simulació. Elaboració pròpia.

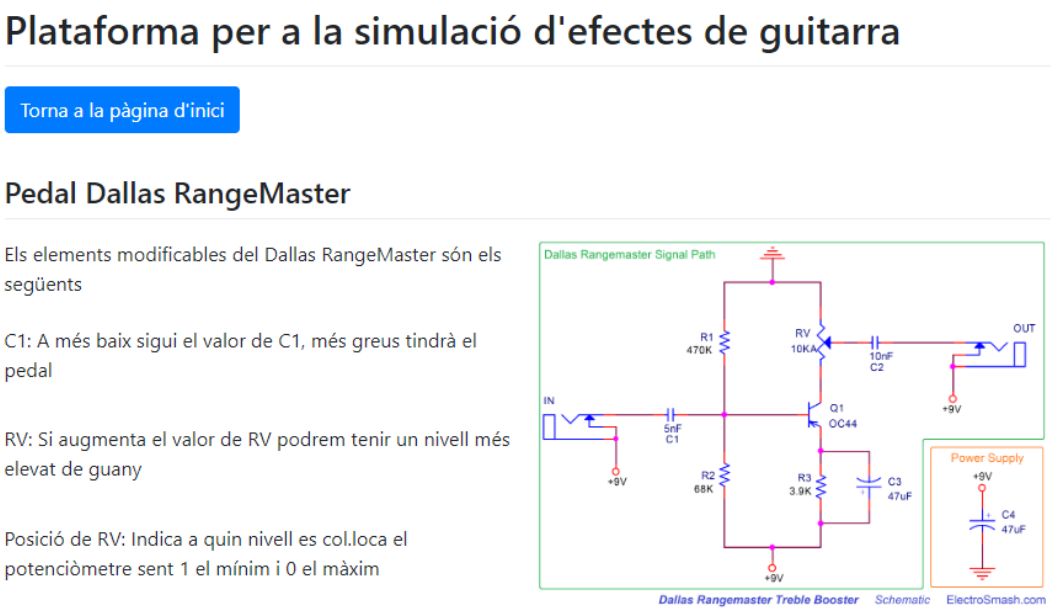

Figura 5.15: Pàgina Web en tancar el popup. Elaboració pròpia.

Capítol 6

Resum del pressupost i/o estudi de viabilitat econòmica

El cost material del desenvolupament del projecte ha estat d'uns 200 Euros, sent aquesta despesa necessària per a la compra de la màquina amb la qual es desenvoluparà el projecte.
D'altra banda, també s'han comptabilitzat les hores de feina com a despesa, per tal que quedi constància de les hores invertides en aquest projecte i per tal de donar valor. Han sigut al voltant de 620 hores de feina amb un valor de 40 Euros per hora. Fent un total de 24.800 Euros
Es pot observar que s'ha complert l'objectiu de mantenir el projecte amb un cost baix, ja que els elements amb els quals s'ha dut a terme el projecte han tingut un valor de 200 Euros.

Capítol 7

Conclusions i Proposta de futur

7.1 Conclusions

Després de la realització del projecte de desenvolupament d'una plataforma d'àudio per a la simulació d'efectes analògics es poden arribar a les següents conclusions:

- En primer lloc, es pot dir que s'han complert tots els requeriments proposats per al projecte. S'ha sigut capaç d'implementar tot el projecte amb programari lliure, obtenint a més una molt bona qualitat de simulació. També s'ha mantingut un baix cost d'execució del projecte.
- En segon lloc, comentar també que gràcies a les eines de desenvolupament web de Python s'ha pogut implementar una pàgina web senzilla per als usuaris.
- Finalment, cal comentar que tot i que els resultats de simulació són correctes, els temps d'execució d'aquestes poden arribar a ser llargs, en cas de tenir una màquina amb més capacitat de procediment aquestes simulacions podrien executar-se més ràpid.

7.2 Proposta de futur

A continuació es proposaran diferents propostes de millora del projecte

- **Expandir el nombre de circuits.**
  Augmentar el nombre de circuits, així com els paràmetres modificables dels quals hi ha pot ser una bona millora per al projecte. Un efecte molt popular a causa de les seves múltiples modificacions podria ser l'Ibanez TubeScreamer TS9.
- **Simulació de circuits proposats per l'usuari expert.**
  En cas que un usuari amb coneixements d'electrònica volgués utilitzar la plataforma de simulació, permetre-li utilitzar el seu propi circuit seria una millora, així no hauria de partir d'un dels efectes base, sinó que podria utilitzar el seu propi disseny

- **Adaptació de formats**.

  Actualment, el simulador només accepta fitxers .WAV amb una freqüència de mostratge de 44100 Hz (si la freqüència és diferent es força un "upsampling" o "downsampling"). Una proposta de millora, seria permetre utilitzar fitxers d'altres extensions tals com .MP3 .MP4, així com fitxers amb una freqüència de mostratge de 48000Hz

- **Reducció temps de simulació.**

  Per a fitxers molt pesats els temps d'execució de les simulacions poden allargar-se en excés, així que una proposta de millora seria reduir aquests temps de computació. Una manera de fer-ho seria utilitzant una màquina amb més capacitat de càlcul, un altre seria optimitzant el càlcul. La darrera opció seria la d'utilitzar la paral·lelització d'ordinadors per a realitzar càlculs en paral·lel

- **Posada en marxa de la pàgina web.**

  En aquest projecte l'objectiu és el desenvolupament de la pàgina web, així que una proposta de futur, seria fer la posada en marxa real de la pàgina web, això implicaria la creació d'un servidor web públic, l'obtenció d'un domini web i realitzar un estudi per assegurar la pàgina de possibles atacs informàtics.

Referències

1. *Esquema Boss OD1*. Available also from: `https://www.hobby-hour.com/electronics/s/od1-overdrive.php/`.
2. *Cadena de senyal de la guitarra elèctrica*. Available also from: `https://www.txirula.com/blog/orden-de-los-pedales-en-una-pedalera.html,`.
3. *Anàlisi de l'efecte Dallas RangeMaster*. Available also from: `https://www.electrosmash.com/dallas-rangemaster`.
4. *Pagina principal de la plataforma StompenbergFx*. Available also from: `https://www.thomann.de/es/stompenberg_devices.html,`.
5. *Aplicació LiveSpice*. Available also from: `https://www.livespice.org/,`.
6. *Pàgina de Raspberri Pi 4*. Available also from: `https://www.raspberrypi.com/products/raspberry-pi-4-model-b/,`.
7. *Pàgina Web del Projecte NgSpice*. Available also from: `https://sourceforge.net/projects/ngspice/`.
8. W.TUINENGA, Paul. *SPICE A Guide to Circuit Simulation Analysis Using Pspice*. Prentice Hall, 1992. ISBN 0-13-747270-6.
9. *Pàgina de Python*. Available also from: `https://www.python.org/,`.
10. *Mòdul Numpy*. Available also from: `https://numpy.org/,`.
11. SALVAIRE, Fabrice. *Mòdul PySpice*. Available also from: `https://pyspice.fabrice-salvaire.fr`.
12. NUNOBRUM. *Mòdul PyLTSpice*. Available also from: `https://github.com/nunobrum/PyLTSpice`.
13. *Mòdul PyWebIo*. Available also from: `https://www.pyweb.io/`.
14. FORSTÉN, Henrik. *Repositori de Github amb els scripts per la conversio de fitxers en format Spice a Wav*. Available also from: `https://github.com/Ttl/spice-audio-tools`.

Capítol 8

Annexes

Codi desenvolupat

```
from pywebio.input import *
from pywebio.output import *
from pywebio import *
from pywebio.session import run_js
import subprocess as sp
import os

import PySpice.Logging.Logging as Logging
logger = Logging.setup_logging()
from PySpice.Spice.NgSpice.Shared import NgSpiceShared
from PySpice.Unit import *
from PyLTSpice.LTSpice_RawRead import LTSpiceRawRead
import numpy as np
import os
from scipy.io.wavfile import read as read_wav

def puja_file():

        contingut_fitxer = file_upload("Puja un fitxer",multiple=False, required=True)
        f=open("/home/pi/TFG/WEB/USER_WAV/fitxer_usuari.wav", 'wb')
        f.write(contingut_fitxer['content'])
        sampling_rate, data=read_wav("/home/pi/TFG/WEB/USER_WAV/fitxer_usuari.wav")

        tamany=len(data)
        duracio=tamany/sampling_rate
```

```
        parametres_fitxer = {
          "fs": sampling_rate,
          "duracio": duracio,
            }

        return parametres_fitxer

def simula_dallas(diccionari_fitxer):
        fs = diccionari_fitxer["fs"]
        duracio = diccionari_fitxer["duracio"]
        inversa_fs = 1/fs
        nou_transitori = "tran "+str(inversa_fs)+"s "+str(duracio)+"s"

        valor_pote = radio("De quants KiloOhms vols el potenciometre Rv?",
        options=["1", "10", "100"])
        nivell = radio("Quin nivell de potenciometre vols?",
        options=['0.1', '0.33', '0.5', '1'])
        valorC1 = radio("Quin valor en nanoFarads vols que tingui el condesadorC1 ?",
        options=["5", "10", "15"])
        nou_condensadorC1 = "C1 intb 101 "+valorC1+"n"

        modificacio_nivell_pote = ".PARAM RPOT=10k SET="+nivell
        modificacio_valor_pote = "RPOT="+valor_pote+"k"
        nou_pote = modificacio_nivell_pote.replace("RPOT=10k", modificacio_valor_pote)
        put_scope("SimulacioDallas")
        with use_scope("SimulacioDallas"):
            put_text("Realitzant la simulaci ")
            with put_loading():
                os.system('/home/pi/TFG/Python/wavtospice.py
                /home/pi/TFG/WEB/USER_WAV/fitxer_usuari.wav nes.txt')

                circuit = '''
            *============== Begin SPICE netlist of main design ============

            .model filesrc filesource (file="nes.txt" amploffset=[0] amplscale=[1.0]
            timeoffset=0 timescale=1 timerelative=false amplstep=false)
            .MODEL OC44 PNP(IS=1.423u BF=307.0 BR=20.27 NF=1.022 NR=1.025 VT=25.5m
            VAF=8.167 VAR=14.84 IKF=43.82m IKR=611.7m ISE=30.54n ISC=213.5n NE=1.316
```

```
NC=1.258 RB=32.83 RE=968.7m RC=989.9u CEB=410p CCB=10p)
.PARAM RPOT=10K SET=1

VCC 100 0 DC 9V

A1 %v[101] filesrc

* V1 100 101 DC 0 AC 1 SIN(0 500m 440Hz)

C1 intb 101 5n
R1 0 intb 470k
R2 100 intb 68k
XPOT1 0 intc out POT RPOT={RPOT} SET={SET}
Q1 intc intb SortTran OC44
R3 100 SortTran 3.9k
C3 100 SortTran 47u
C2 out Sortida 10n
RC Sortida 0 100k

.SUBCKT POT 1 T 2 RPOT=10k SET=0.5
RT 1 T {RPOT*SET+0.001}
RB T 2 {RPOT*(1-SET)+0.001}
.ENDS

.control
save v(sortida)
tran 20.833u 9s
run
* save v(Ni), V(No)E
wrdata trdata v(sortida)
write trdata.raw
.endc
.END
'''
    circuit = circuit.replace(".PARAM RPOT=10K SET=1", nou_pote)
    circuit = circuit.replace("tran 20.833u 9s", nou_transitori)
    circuit = circuit.replace("C1 intb 101 5n", nou_condensadorC1)

    ngspice = NgSpiceShared.new_instance()
```

```
            ngspice.load_circuit(circuit)
            ngspice.run()

            os.system('/home/pi/TFG/Python//spicetowav.py trdata
            /home/pi/TFG/AudioSimulat/Audio_Simulat.wav')
            os.system('rm /home/pi/TFG/WEB/nes.txt')
            os.system('rm /home/pi/TFG/WEB/trdata.raw')
            os.system('rm /home/pi/TFG/WEB/trdata')

        content = open('/home/pi/TFG/AudioSimulat/Audio_Simulat.wav', 'rb').read()


        popup('Simulaci Realitzada amb xit !', [
            put_html('<h3>Gracies per utilitzar el simulador</h3>'),
            put_html('<h4>En el seguent enlla podr s descarregar la
            teva simulaci </h4>'),
            put_file('Simulacio.wav', content, 'Fes click per descarregar!'),
            put_button('Tanca', onclick=close_popup)
        ])
        clear(scope="SimulacioDallas")
        put_text("Simulaci realitzada amb xit ", scope="SimulaDallas")




    os.system('rm /home/pi/TFG/AudioSimulat/Audio_Simulat.wav')


def simula_fuzz(diccionari_fitxer):

    fs = diccionari_fitxer["fs"]
    duracio = diccionari_fitxer["duracio"]
    inversa_fs = 1/fs
    nou_transitori = "tran "+str(inversa_fs)+"s "+str(duracio)+"s"

    nivell_guany = radio("Quin nivell de potenciometre 1 vols?",
    options=['0.1', '0.33', '0.5', '1'])
    nou_pote_guany = ".PARAM RPOT=1k SET="+nivell_guany
```

```
    nivell_volum = radio("Quin nivell de potenciometre 2 vols?",
    options=['0.1', '0.33', '0.5', '1'])
    nou_pote_volum = ".PARAM RPOT2=500k SET2="+nivell_volum

    put_scope("SimulacioFuzz")
    with use_scope("SimulacioFuzz"):
        put_text("Realitzant la simulaci ")
        with put_loading():
            os.system('/home/pi/TFG/Python/wavtospice.py
            /home/pi/TFG/WEB/USER_WAV/fitxer_usuari.wav nes.txt')

            circuit = '''
*============== Begin SPICE netlist of main design ===========

.model filesrc filesource (file="nes.txt" amploffset=[0] amplscale=[1.0]
timeoffset=0 timescale=1 timerelative=false amplstep=false)

*AC128 PNP Germanium Transistor Spice Model
*
.MODEL AC128_X PNP IS=1.41f ISC=0 ISE=0 IKF=80m IKR=0 ITF=0.4
+NC=2 NE=1.5 BF=70 BR=4.977 RB=10 RC=2.5 CJC=9.728p CJE=8.063p
+TR=33.42n TF=179.3p FC=0.5 EG=1.11 VJC=0.2 VJE=0.2 VTF=4
+MJC=0.5776 MJE=0.3677 XTB=1.5 XTF=6 XTI=3

*AC127 NPN Germanium Transistor Spice Model
*
.MODEL AC127_X NPN IS=1.41f ISC=0 ISE=0 IKF=80m IKR=0 ITF=0.4
+NC=2 NE=1.5 BF=70 BR=4.977 RB=10 RC=2.5 CJC=9.728p CJE=8.063p
+TR=33.42n TF=179.3p FC=0.5 EG=1.11 VJC=0.2 VJE=0.2 VTF=4
+MJC=0.5776 MJE=0.3677 XTB=1.5 XTF=6 XTI=3

.PARAM RPOT=1K SET=0.5
.PARAM RPOT2=500K SET2=0.5

.SUBCKT POT 1 T 2 RPOT=10k SET=0.5
RT 1 T {RPOT*SET+0.001}
RB T 2 {RPOT*(1-SET)+0.001}
.ENDS
```

```
VCC 0 100 DC 9V

 A1 %v[101] filesrc

* V1 101 0 DC 0 AC 1 SIN(0 500m 440Hz)

C1 101 BAQ1 2.2u IC=0
Q1 COQ1 BAQ1 0 AC128_X
R1 COQ1 100 33k
R2 102 100 470
R3 COQ2 102 8.2k
R4 BAQ1 EQ2 100k
Q2 COQ2 COQ1 EQ2 AC128_X
XPOT1 0 104 EQ2 POT RPOT={RPOT} SET={SET}
C2 0 104 20u IC=0
C3 102 103 10n IC=0
XPOT2 0 SAL 103 POT RPOT={RPOT2} SET={SET2}
RLOAD SAL 0 100k
.control
save v(sal)
tran 20.833u 9s
run
* save v(Ni), V(No)E
wrdata trdata v(sal)
write trdata.raw
.endc
.END
'''
            circuit = circuit.replace(".PARAM RPOT=1K SET=0.5", nou_pote_guany)
            circuit = circuit.replace(".PARAM RPOT2=500K SET2=0.5", nou_pote_volum)
            circuit = circuit.replace("tran 20.833u 9s", nou_transitori)
            print(circuit)
            print(nou_transitori)
            ngspice = NgSpiceShared.new_instance()
            ngspice.load_circuit(circuit)
            ngspice.run()

            os.system('/home/pi/TFG/Python//spicetowav.py trdata /home/pi/TFG/AudioSimula
            os.system('rm /home/pi/TFG/WEB/nes.txt')
```

```
            os.system('rm /home/pi/TFG/WEB/trdata.raw')
            os.system('rm /home/pi/TFG/WEB/trdata')

        content = open('/home/pi/TFG/AudioSimulat/Audio_Simulat_fuzz.wav', 'rb').read()
        popup('Simulaci  Realitzada amb  xit !', [
            put_html('<h3>Gracies per utilitzar el simulador</h3>'),
            put_html('<h4>En el seguent enlla  podr s descarregar la teva simulaci </h4
            put_file('Simulacio.wav', content, 'Fes click per descarregar!'),
            put_button('Tanca', onclick=close_popup)
        ])
        put_button('Torna al main', onclick=lambda:run_js('window.location.reload()'))
        os.system('rm /home/pi/TFG/AudioSimulat/Audio_Simulat_fuzz.wav')

def main():
    put_row(put_html('<h1>Plataforma per a la simulaci  d\'efectes de guitarra</h1>'))
    put_scope("seleccio")
    put_scope("informacio")
    with use_scope("seleccio"):
        pedal = input.select('Selecciona el pedal que vulguis',
        ['Dallas Rangemaster', 'Fuzz Face'])
        if (pedal == "Dallas Rangemaster"):
            with use_scope("informacio"):
                img_dallas = open('/home/pi/TFG/Media/dallas_esq.png',
                'rb').read()
                put_row(put_html('<h2>Pedal Dallas RangeMaster</h2>'))
                put_row([
                    put_column([
                        put_text("Els elements modificables del Dallas RangeMaster
                        s n els seg ents"),
                        put_text('C1: A m s baix sigui el valor de C1, m s greus
                        tindr  el pedal'), None,
                        put_text('RV: Si augmenta el valor de RV podrem tenir un
                        nivell m s elevat de guany'), None,
                        put_text('Posici  de RV: Indica a quin nivell es col.loca
                        el potenci metre sent 1 el m nim i 0 el m xim'),
                        ]), None,
                put_image(img_dallas)
])
            caracteristiques=puja_file()
```

```
            simula_dallas(caracteristiques)
            put_button('Torna a la p gina d\'inici',
            onclick=lambda:run_js('window.location.reload()'))
    if (pedal == "Fuzz Face"):
        with use_scope("informacio"):
            put_text("HAS SELCCIONAT EL EFECTE 1")
        with use_scope("fitxer"):
            caracteristiques=puja_file()
        simula_fuzz(caracteristiques)
    # put_text('gift = %r' % gift)

start_server(main, port=8080, debug=True)
```