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

BACHELOR FINAL THESIS

Real time acoustic analysis and correction

Document:

Report

Author:

Alexandr Ramos Sundukov

Director:

Albino Nogueiras Rodríguez

Degree:

Bachelor's degree in Audiovisual Systems Engineering

Examintaion session:

Spring, 2025

Summary

This project addresses a common challenge faced by audio engineers and technicians: adjusting sound systems based on the acoustic properties of a given environment. While there are existing commercial solutions that offer advanced analysis tools, this work proposes the development of a custom, software-based alternative using open-source technologies.

The proposed solution is implemented in Python and designed to perform real-time acoustic analysis and correction. It integrates several scientific libraries, including NumPy and SciPy for signal processing, and uses Sounddevice for real-time audio input and output. The program interacts with the environment through a basic hardware setup consisting of a microphone, a speaker, and a sound card.

A graphical user interface is developed using Tkinter and Matplotlib to ensure usability and ease of interaction. This graphical user interface allows users to configure audio parameters, select input channels, visualize data, and monitor system behavior in real time. Features include Fourier transformation, filter-based processing, and measurement tools such as delay, phase, and frequency response analysis.

Throughout the development, special attention is given to flexibility and accessibility. The software is intended to run on standard hardware under Linux (specifically Ubuntu 22.04.5 LTS), making it usable in both professional and home studio environments.

Finally, the project reflects on the difficulties encountered during development, acknowledging that not all objectives have been fully achieved. It discusses the main technical and practical challenges, and offers constructive criticism to guide future iterations. Despite these limitations, the result is a solid foundation on which to build. While the tool is not yet fully functional, it provides a working base from which the missing features can be implemented and the existing ones improved.

Índex

List of figures

List of tables

List of abbreviations

UPC Universitat Politècnica de Catalunya

ESEIAAT Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa [UPC25].

RTA Real Time Analysis

SMAART System Measurement Acoustic Analysis Real-time Tool

FT Fourier Transform

FFT Fast Fourier Transform

DFT Discrete Fourier Transform

RFFT Real-valued Fast Fourier Transform

IEC International Electrotechnical Commision

RMS Root Mean Square

IIR Infinite Impulse Response

FIR Finite Impulse Response

GUI Graphical User Interface

OS Operating System

RT60 Reverberation time to decay 60 dB.

HI-FI High fidelity

DSP Digital Signal Processor

Chapter 1

Introduction

1.1 Objecte

Resultat final que es vol aconseguir. En aquest cas, l'objecte d'aquesta plantilla és donar les pautes d'estructura i contingut de la Memòria del TFE.

El cos tant d'aquest document "Memòria" com dels altres documents integrants del TFE (Pressupost, Annexos i Plec de condicions) serà amb lletra Times New Roman o Arial d'una mida d'11 punts, marge lateral esquerra de 3 cm, dret de 2,5, superior i inferior de 2,5 i espaiat senzill.

L'alumne/a ha de revisar l'ortografia i gramàtica de tots els documents del TFE; ha d'utilitzar les unitats del Sistema Internacional; ha d'utilitzar un nombre coherent de decimals; i ha d'identificar els eixos dels gràfics inclosos al llarg del text.

Es recomana que la memòria no superi una extensió màxima de 60-70 pàgines. Tant les taules com figures han d'estar enumerades i tenir un títol. Si s'han obtingut d'algun altre document consultat, s'haurà de dir la font d'on s'ha tret [UPC25].

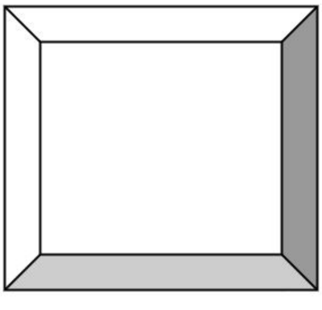


Figure 1.1: Imatge d'exemple

Table 1.1: Risks assessment

1	X2	X	X
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

1.2 Abast

Paquets de treball i lliurables necessaris per arribar a la solució.

1.3 Requeriments

O especificacions bàsiques. Restriccions sobre la solució final.

1.4 Justificació

Plantejament de la necessitat del treball des d'una visió global i aproximant-lo a una visió més específica. Serveix per centrar i contextualitzar el treball.

1.4.1 Subsection

1.4.1.1 superseccion

divisiones.py

```
"""
Biblioteca con definiciones importantes para la división de números.
Se incluyen las funciones divideSiDivisible() y cocienteModulo().
"""

def divideSiDivisible(nume, deno):
"""
Si nume es divisible por deno, devuelve la división
entera. Si no lo es, devuelve None.
"""

if not nume % deno:
return nume // deno
```

```python


def cocienteModulo(nume, deno):
"""
Devuelve el conciente entero y el resto de
la divisi
ón entera (mod) de dos números.
"""

return nume // deno, nume % deno

```

src/divisions.py

```python
def divideSiDivisible(nume, deno):
        """
        Si nume es divisible por deno, devuelve la división
        entera. Si no lo es, devuelve None.
        """

        if not nume % deno:
        return nume // deno
```

Chapter 2

Background and/or status of the matter

Nowadays, there are many solutions that can fit to solve our problem. Some are very expensive, and others have shortcomings. In this chapter, we will have a look at some of the most popular solutions.

2.1 Smaart [DONE]

Smaart, an acronym for *System Measurement Acoustic Analysis Real-time Tool* [25c], is a software-based solution commercialized by Rational Acoustics. It is probably the most used and well-known solution for professional acoustic analysis, used in big venues, concert halls, stadiums, touring productions, as well as in professional audio studios and speaker development laboratories. Common uses are:

- **Speaker Alignment:** When we have multiple sound sources, this software helps us find the phase and delay between them. For example, it can be used to find the time and phase alignment between a subwoofer and a full-range speaker.
- **RTA, Frequency and Phase Response** used to view live spectrograms, phase deviation, or energy in frequency bands. One example of use is identifying resonances at specific frequencies.
- **Coherence Analysis** to evaluate the quality of the measured data. A common use is to detect reflections and background noise.
- **Delay Time** between different sources or signals. Widely used to synchronize different elements of the system.
- **Room and architectural acoustics** to identify the frequency and phase response of a room, as well as reverberation and echoes.

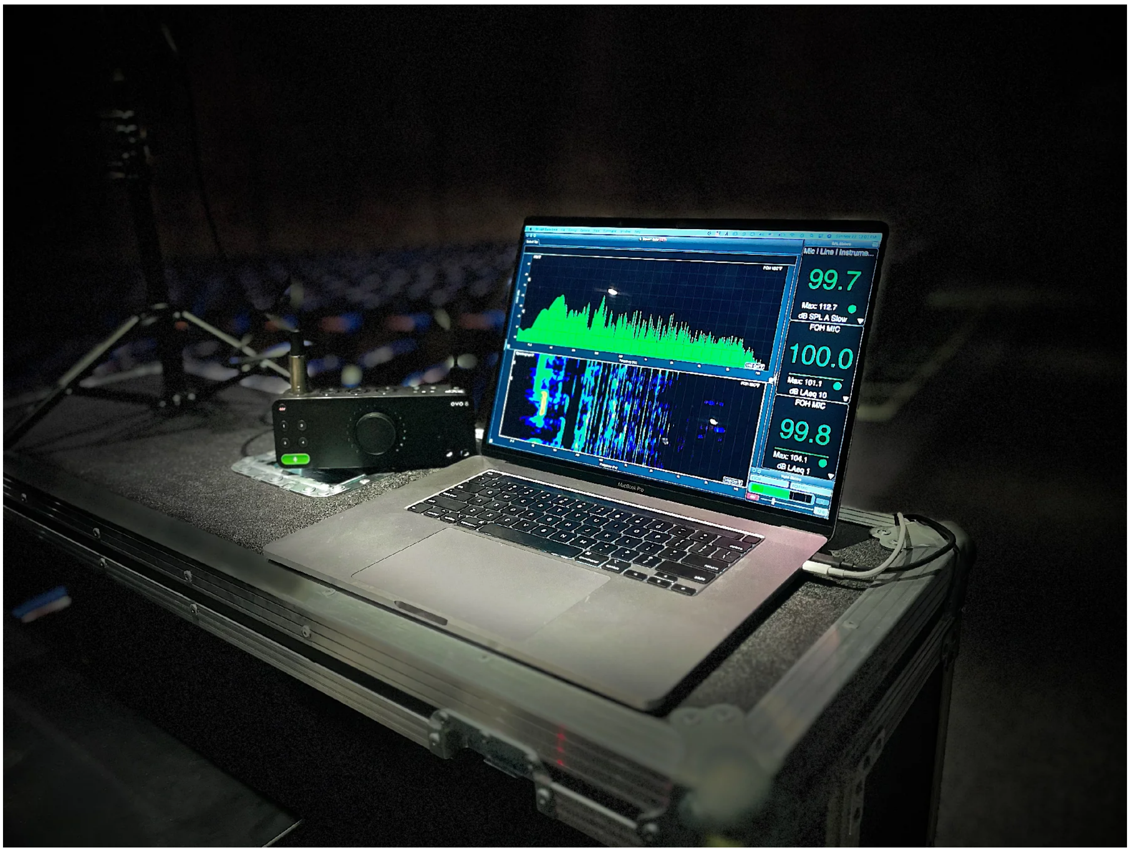


Figure 2.1: Notebook using Smaart, where we can see some of the tools it includes. The notebook is connected to the EVO 8 (a USB interface that acts as an external sound card).

The strongest points of this program are:

- **Flexibility:** As a software-based solution, it can run on any Windows or Mac computer (meeting the minimum required specs), and can be used with most external audio sound cards, allowing the connection of unlimited types of microphones or direct signals.

- **More than one channel:** This software can analyze and display information from more than one input channel at the same time, allowing comparisons between different channels. This is used to compare an original signal with the signal captured by a microphone inside a room with a sound system, helping to detect room acoustics or sound system issues. Another common use is to measure the sound in different places of the same room simultaneously.

- **Widely used:** It is very common to see professionals in the sector using this software, or at least being familiar with it. It has become a kind of standard, which leads other companies to ensure maximum compatibility with it. For example, Audix makes the Audix TM-1 Plus microphone [And23], which includes a file that can be imported into SMAART to apply microphone correction during analysis.

On the other hand, it requires a license, external hardware such as sound cards and microphones, and it does not have any correction capabilities—only analysis.

2.2 Dirac

Home correction solution ———————————————————————

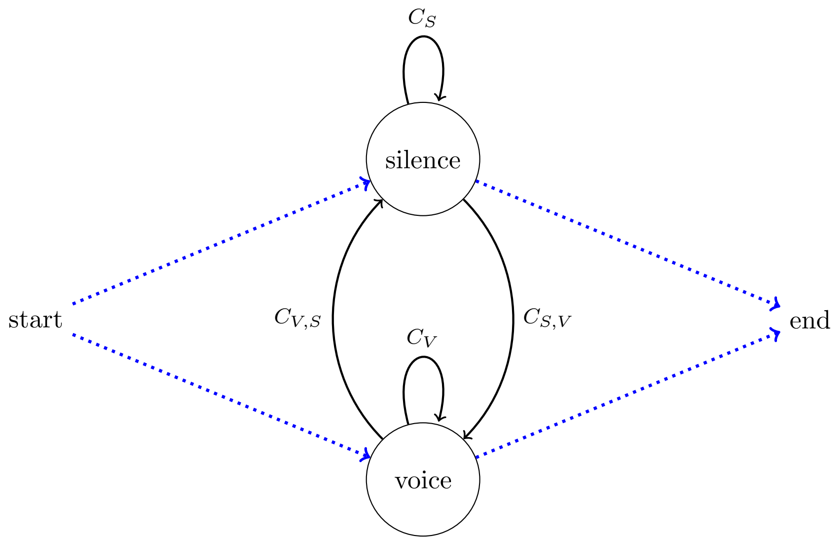
2.3 REW

Free software for room measurement ———————————————————————

2.4 Trinnov

Hardware base solution with correction ———————————————————-

******************************** Exemple d'esquema *************************************



Chapter 3

Methodology, consideration and decision on alternative solutions [DONE]

Looking at the "Background and/or status of the matter" chapter, I want to develop a solution that incorporates as many of the strengths of existing solutions as possible.

Considering the available resources, the most effective solution is a **software-based solution** that interacts with the sound card, display, mouse, and keyboard available in an OS (*Operating System*).

This decision allows me to implement the following strength points:

- **Analysis capabilities:** This will include: spectrogram, 31-band graphic display, delay measurement, phase measurement, RT60 measurement, and waterfall diagram.
- **Processing capabilities:** It must be able to process a live signal using an equalizer implemented with digital filters.
- **Signal management:** It must handle an external clean signal, a signal that has passed through the system, and output a processed signal — all in real-time.
- **Low resource usage:** Considering the time available for this project and the limited budget. In my case, I already have everything I need for development, but anyone with a PC, a sound card, a microphone, and a speaker can use this solution.

Of course, the type and quality of the sound card and microphone will be critical to obtaining accurate and realistic results. Under optimal conditions, a high-resolution sound card with good SNR, along with a high-quality measurement microphone, is required. These microphones are typically small-diaphragm, condenser, omnidirectional, with a flat frequency response and good signal-to-noise ratio.

The importance of the speaker depends on whether we consider it part of the system (a fixed speaker that is part of the room) or simply a tool used to excite the room. In the first case, the program should correct the speaker's imperfections, as it is part of the system being analyzed and corrected. The only requirement for this speaker is that it must be able to reproduce the full frequency spectrum that will be analyzed and corrected.

However, in the second case, it is very important for the speaker to have the flattest response possible, since any imperfections in the speaker will introduce false data about the room acoustics.

I will be using a Behringer U-Phoria UMC204HD sound card, an old Electro-Voice desktop microphone, and an old AIWA SX-NAVH1000 home hi-fi speaker. These are not the best hardware for analysis tools, but they are sufficient to develop the software under home conditions.

Figure 3.1: Home setup showing the sound card, the speaker, and the microphone.

As a computer, I will be using a laptop running Ubuntu 22.04.5 LTS and Python 3.

In order to achieve the overall goal, the program to be developed needs a set of tools that must communicate with each other. To achieve this, a **GUI** (*Graphical User Interface*) will be very helpful, and it should be easy and intuitive to use.

Chapter 4

Development of the chosen solutions

Once we have defined what we want to do in the previous chapter, **Methodology, Considerations, and Decisions on Alternative Solutions**, it is time to get to work.

4.1 Graphic interface

About user friendly graphic interface.................................

This tools can be differenciated in two very important groups, taht also will be two diferent windows of the program: REVIOSARRRRRRRRRRRRRRRRRRRRRRRR

- **Analysis Window** that will recieve sound from external input, and sound that comes from System (Input form System).
- **DSP Window** that will recieve sound form external input, data about how it have to process, and send processed signal to Output (Output to System)

....

4.2 Signal Path [DONE]

To perform any kind of analysis or signal processing, the first thing we need is data—more specifically, signal data captured from the sound card. This data is sent to different modules so that each one can carry out its specific task. Additionally, in the case of the correction module, it is necessary to send output data back to the sound card (the corrected signal). Additionally, there will be non-signal data containing the results of the analysis, which will be sent to the DSP (*Digital Signal Processor*) or the correction window.

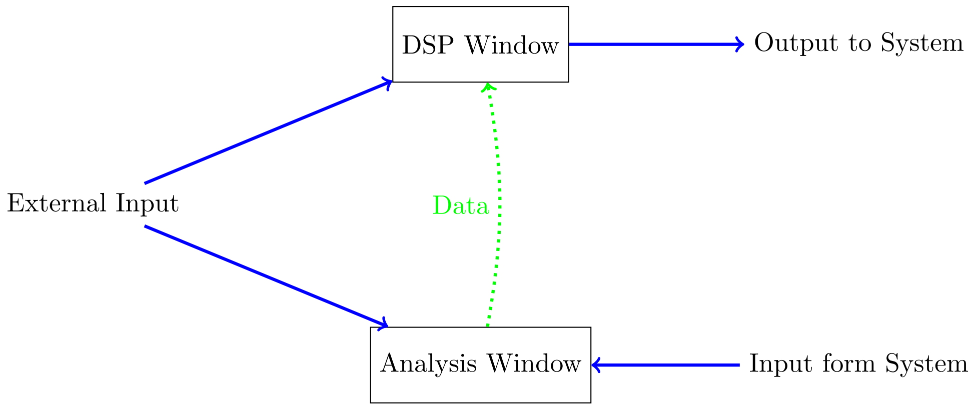


Figure 4.1: Basic representation of the required signal path

To achieve real-time processing, we must manage a constant flow of data. For this purpose, I will use a stream-style data management approach, which consists of capturing small blocks of data, processing them, and continuously sending the results. To capture and send this stream to and from the sound card, the Sounddevice library [24] will be used.

To simplify things, the two required inputs—**External Input** and **Input from System**—will be handled using a single input stream, while the output—**Output to System**—will be managed through a separate, independent output stream. All streams will share the same parameters, such as bit depth, sample rate, and block size. However, the audio device and channel parameters will be independently set for each stream. In the case of the input stream, each input channel must also be configured separately.

Once the streams are running and managed from the root window, we need to handle the data coming from the input streams and the data to be sent to the output stream. To achieve this, I will use one input buffer for both input channels and a separate output buffer. These buffers must be accessible to all tool modules in the program and will be locked whenever a module accesses them, in order to prevent simultaneous reads and writes.

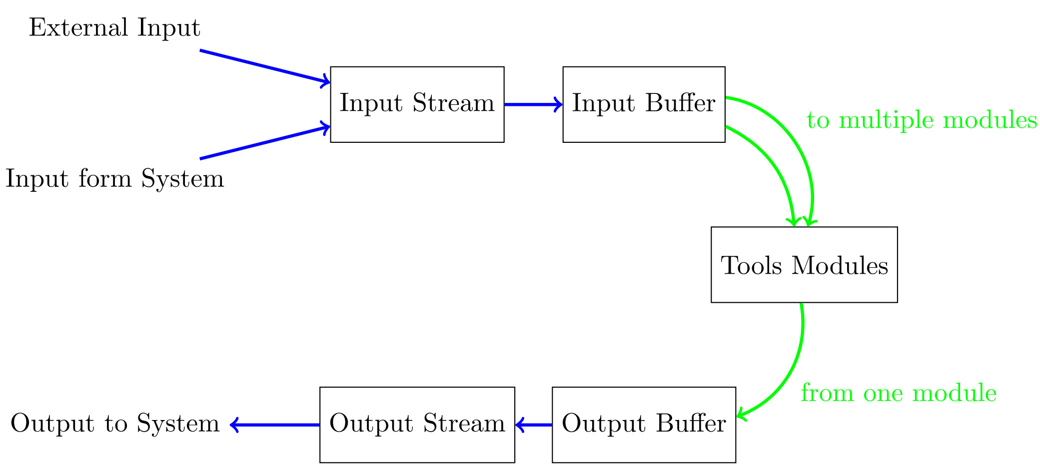


Figure 4.2: Stream management

4.3 Settings page [DONE]

In order to be flexible with hardware, we need a settings window where we can tell the program which sound card, which channel, and which parameters we want to use. These parameters will also affect the resolution and fidelity of the analysis and processing, according to the sound card's capabilities.

I will splitt setting window in tow pages:

- **Device Settings:** Where we can define which sound card and which channels we want to use.
- **Audio Settings:** Where we can define which audio parameters we want to use.

4.3.1 Device Settings

I will use sounddevice methods to identify which sound cards and channels are available, and filter them into inputs and outputs. Then, using a questionnaire-style interface, I will let the user select which sound card and which channel they want for each required signal using a dropdown menu.

At the bottom of the page, there will be a "Confirm" button that performs some basic checks, such as verifying that the "external input" and the "input from system" are not the same input channel. Then, it will display a green message if the settings are applied successfully, or a red one if something goes wrong. Finally, it will update the values and labels on the root window to indicate which channels are currently selected.

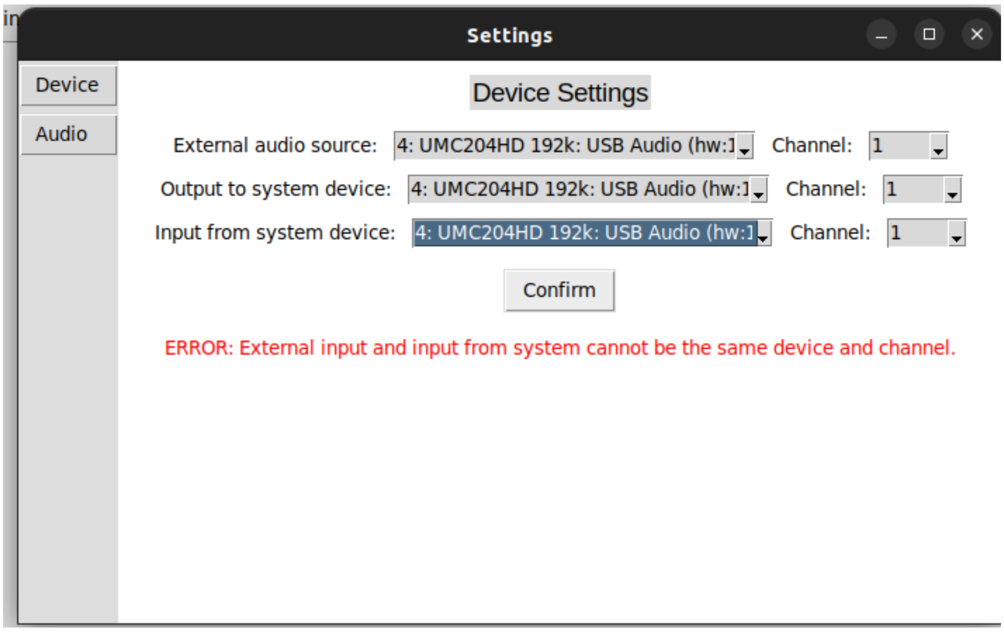


Figure 4.3: Device Settings page showing a red message because the user selected an invalid configuration.

4.3.2 Audio Settings

On this page, just like in the previous one, I will use a questionnaire-style interface with dropdown menus to define the Sample Rate, Bit Depth, and Block Size to be used.

For the Sample Rate and Bit Depth, I will suggest the most common values — such as 44,100 Hz, 48,000 Hz, 96,000 Hz, and 192,000 Hz for the sample rate, and 16, 24, and 32 bits for bit depth. However, the user is free to enter any value. The same applies to the Block Size.

At the bottom of the page, there is also a "Confirm" button, similar to the one in the "Device Settings" page. It performs basic checks, updates the selected values, and displays a message: green if everything is OK, orange in case of a warning (it applies the changes but alerts the user when a non-recommended or uncommon value is entered), and red if the Block Size is not a power of two — something required for the algorithms to work properly.

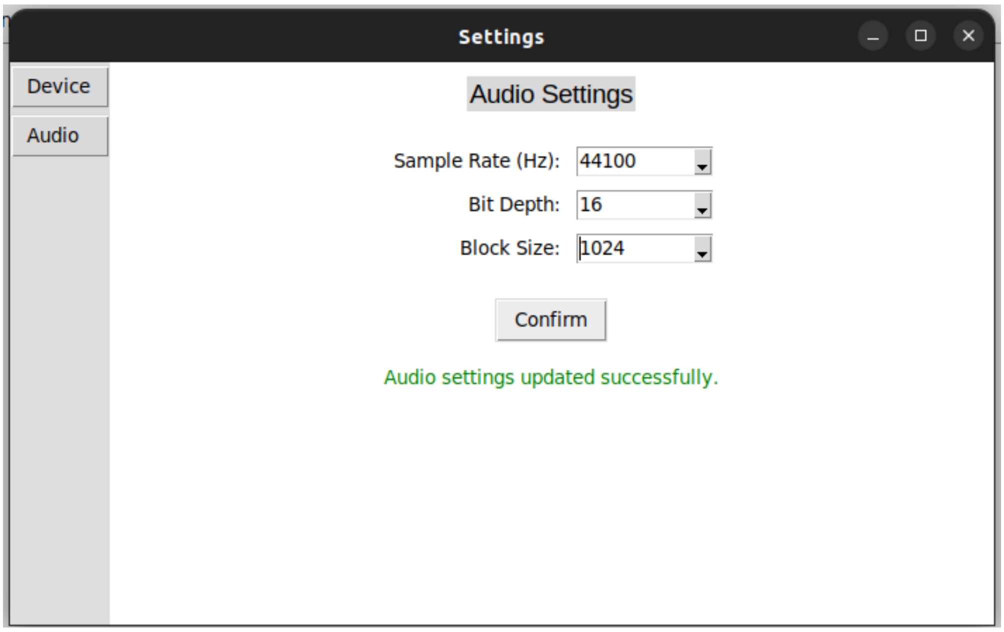


Figure 4.4: Audio Settings page showing a green message.

4.4 Acoustics analysis

The acoustic analysis is devoloped with python librari LIBROSA[McF+25]

4.4.1 Spectrogram (FT)

How implemented the spectogram

4.4.2 RTA [DONE]

Usually, any form of analysis that is performed in real time can be considered **RTA** (*Real-Time Analysis*). This includes a wide range of operations such as spectrum monitoring, transfer function measurements, phase and coherence analysis, and more—all happening as the signal flows. However, in my experience, in common usage, when someone refers to "RTA", they are often specifically referring to the classic 31-band graphical spectrum display. Also, the data collected on this page is especially important because it will be used to set the correction parameters. For all of that, this page has been named **RTA**.

Also, there is another conflict. When someone defines the 31 bands and their bandwidth, it is common to use the definition provided in **IEC 61260**. However, this standard does not mathematically respect the logarithmic spacing between bands. The 31 bands are defined as 1/3 of an octave per band.

Table 4.1: Center frequencies for true 1/3 octave bands and IEC 61260 bands

1/3 octave	19.69	24.8	31.25	39.37	49.61	62.5	78.75	99.21	125	157.49	
IEC 61260	20	25	31.5	40	50	63	80	100	125	160	
1/3 octave	198,43	250	314.98	396.85	500	629.96	793.7	1000	1259.92	1587.4	
IEC 61260	200	250	315	400	500	630	800	1000	1250	1600	
1/3 octave	2000	2519.84	3174.8	4000	5039.68	6349.6	8000	10079.37	12699.21	16000	20158.74
IEC 61260	2000	2500	3150	4000	5000	6300	8000	10000	12500	16000	20000

ipython
```python
"""
In order to obtain the true 1/3 octave values, the following line of code was used with IPython3.
The results were rounded to the second decimal place.
"""

[round(1000*2**(band/3), 2) for band in range (-17,14)]

```

On the other hand, this standard is widely used in many professional devices and software. One important example is the DBX 231s graphic equalizer, which is commonly used in analog processing chains.

Figure 4.5: Image of the front panel of the DBX 231s [25a], where we can observe that the center frequency bands are the same as those defined in IEC 61260.

In order to achieve the greatest possible compatibility and coherence with industry standards, I prefer to use the IEC 61260 standard.

The signal path on this page is very similar to the one used on the "FT" page. We have a buffer with two blocks of input data ("Input from external device" and "Input from system").

First, we copy the buffer data to the "delay buffer", where, if needed, the data will be adjusted to make it coincide with the applied delay. If necessary, the adjustment will use data from previous blocks stored in the same "delay buffer".

I created this buffer with the capacity to store 1 second of data, which can be used to apply a maximum delay of (1 - "Block Size in seconds") seconds.

Once the data in the "delay buffer" is adjusted, we can start applying algorithms to perform the analysis.

The algorithm is based on the calculation of RMS (*root mean square*) to obtain the energy for each frequency band, which is divided using filters for each band.

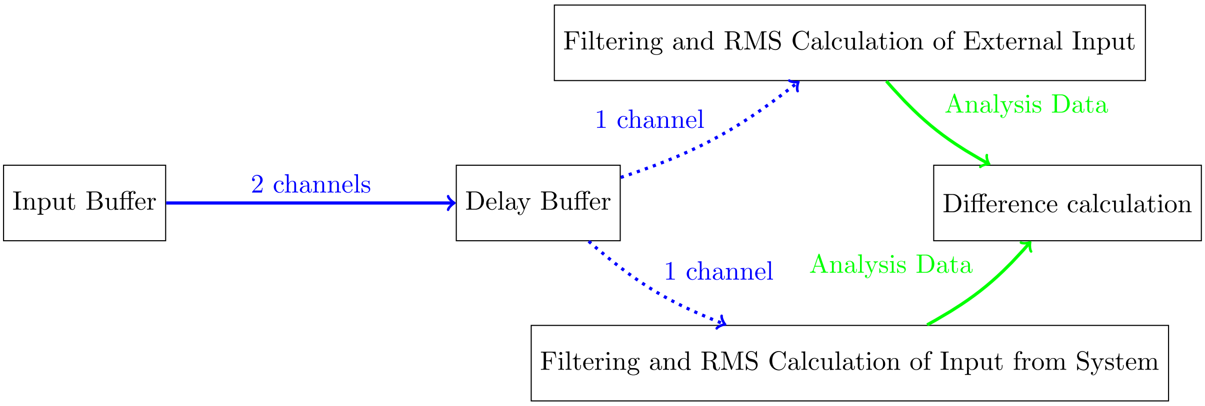


Figure 4.6: Diagram of the architecture of the RTA page

In this case, I'm not using any kind of windowing. As a starting point, I'm using 4th-order IIR (*infinite impulse response*) Butterworth band-pass filters for each band. All these filters are created using the `scipy.signal` library [25b], which returns SOS (*Second-Order Section*) parameters.

Figure 4.7: Second-Order Sections for IIR filters with their parameters

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

When the program applies each filter to the signal block, it also calculates the RMS and converts it to a logarithmic scale, which will be plotted on the graph and used to calculate the difference graph.

Figure 4.8: Root Mean Square to calculate energy from filtered signal

$$RMS = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} x^2[n]}$$

At the end, there is: a pause button, which blocks the update function and can be used to pause the graphics; a save button, to store the current values of the difference graph for later use in the correction window; and a time averaging section that works exactly the same as the averaging section from the FT page, except that it does not include frequency averaging (since it doesn't make sense to apply frequency averaging between bands).

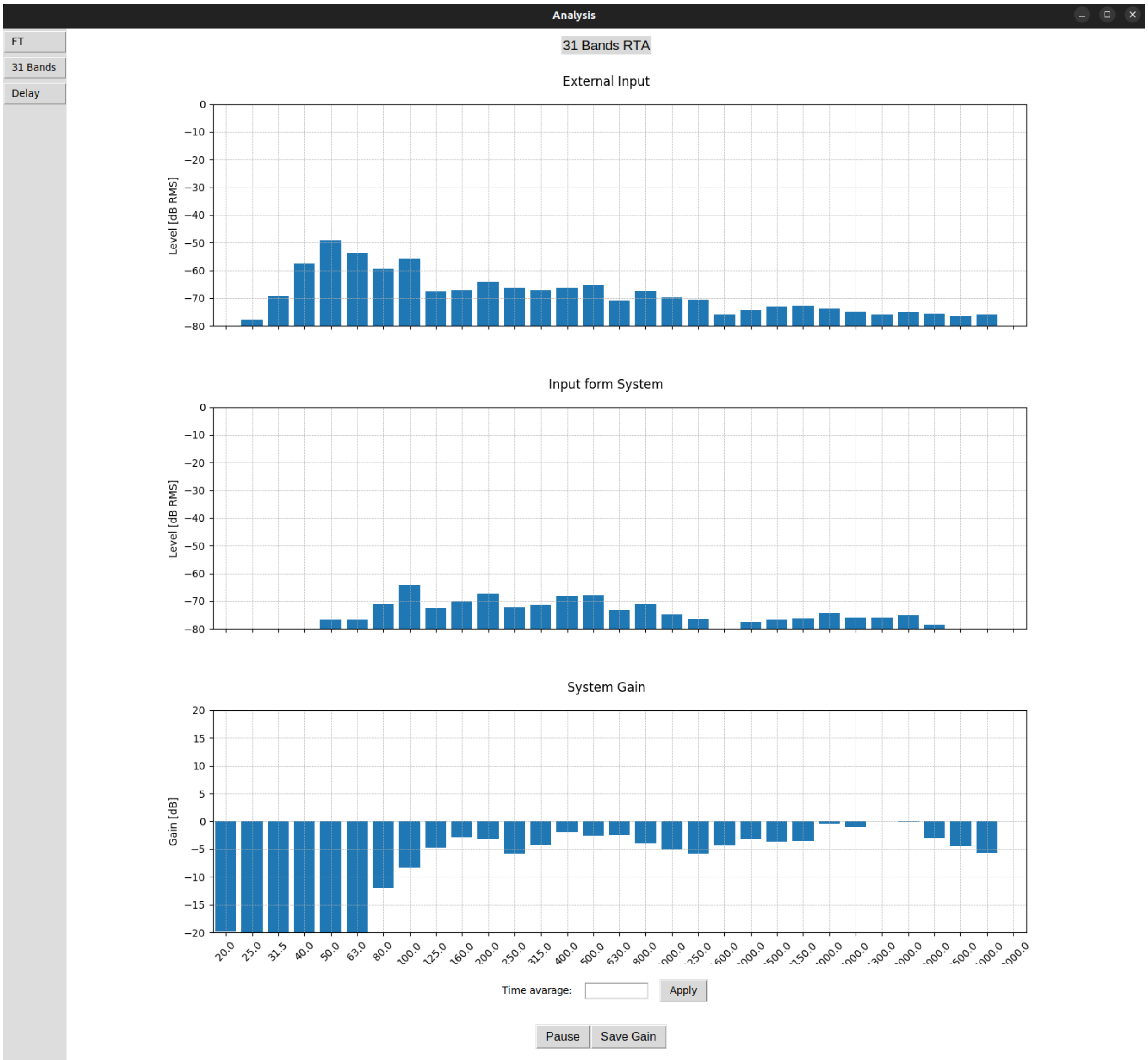


Figure 4.9: Analysis window - RTA page.

4.4.3 Delay

Explain delay page

4.5 Acoustic correction

Acoustic correction = DSP, allways have to be somethuing on the output buffer, by default, zeros.

4.5.1 Bypass

How Bypass works, and why it works bad.

4.5.2 31 Bars

Implementation of correction

4.6 Integration of monitoring mechanism

Home page and information that it appears / start, stop streams buttons...

4.7 Others

More problems that I didn't expect, losing time solving them or at least trying to. No more time to implement additional functionalities...

Chapter 5

Results

About the final program results

There ar not just things to finish or implement, also, it is nedded to solve some actual problems. Most important of them are:

- **Filter Algorithms:**
- **User Limitations:** For example, since we are using the Sounddevice library and managing both inputs as a single input stream, we have the limitation that both channels must come from the same sound card. However, the program currently allows the selection of different sound cards for each input channel.

5.1 Final tests

About testing program on real situations

5.2 User experiance

About user experiance.

Chapter 6

Conclusions

Fa la funció de síntesi final i s'elabora a partir de la interpretació dels resultats assolits. La conclusió sol ser breu i s'ha de relacionar directament amb els objectius del treball.

També s'hi han d'incloure les recomanacions de continuació del treball i la planificació i programació del treball futur proposat.

Bibliography

[24] *API Documentation — python-sounddevice, version 0.5.1.* [Online; accessed 31. May 2025]. Oct. 2024. URL: https://python-sounddevice.readthedocs.io/en/0.5.1/api/index.html.

[25a] *DBX 231s.* [Online; accessed 26. May 2025]. May 2025. URL: https://www.thomann.es/dbx_231s.htm.

[25b] *Signal processing (scipy.signal) SciPy v1.15.3 Manual.* [Online; accessed 26. May 2025]. May 2025. URL: https://docs.scipy.org/doc/scipy/reference/signal.html.

[25c] *Smaart Home.* [Online; accessed 25. May 2025]. May 2025. URL: https://www.rationalacoustics.com/pages/smaart-home.

[And23] Karen Anderson. "Audix TM1 Plus Now Available". In: *Rational Acoustics* (Sept. 2023). URL: https://www.rationalacoustics.com/blogs/news/audix-tm1-plus-now-available.

[McF+25] Brian McFee et al. *librosa/librosa: 0.11.0rc1.* Version 0.11.0rc1. Feb. 2025. DOI: 10.5281/zenodo.14908061. URL: https://doi.org/10.5281/zenodo.14908061.

[UPC25] UPC. *The School of Industrial, Aerospace and Audiovisual Engineering of Terrassa.* Spring of 2025. URL: https://eseiaat.upc.edu/en?set_language=en.