ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη ψηφιακού πεταλιού πολυεφέ για ηλεκτρική κιθάρα

Της φοιτήτριας
Λιόγκα Κωνσταντίνα
Αρ. Μητρώου: 2020080

Επιβλέπων
Χατζόπουλος Αργύριος
Αναπ. Καθηγητής

Ημερομηνία 29/05/26

Τίτλος Δ.Ε.: Ανάπτυξη ψηφιακού πολυεφέ για ηλεκτρική κιθάρα
Κωδικός Δ.Ε.: 25228
Ονοματεπώνυμο φοιτητή/τών: Λιόγκα Κωνσταντίνα
Ονοματεπώνυμο εισηγητή Αργύριος Τ. Χατζόπουλος
Ημερομηνία ανάληψης Δ.Ε. 30/03/2025
Ημερομηνία περάτωσης Δ.Ε. 29/05/2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Στη μάθηση, που ίσταται ένας ατέρμονος βρόγχος.»

Πρόλογος

Η ιδέα της διπλωματικής μου εργασίας προήλθε από το ενδιαφέρον μου για την μουσική, συγκεκριμένα την rock και metal, είδη μουσικής που χρησιμοποιούν εφέ για την αλλοίωση του ήχου. Ως παίκτρια ηλεκτρικής κιθάρας, ήθελα να δημιουργήσω ένα οικείο εργαλείο. Επιπλέον, ήθελα να αναπτύξω ένα ολοκληρωμένο προϊόν αξιοποιώντας τις γνώσεις μου πάνω στην πληροφορική αλλά και στην ηλεκτρονική, συνδυάζοντας έτσι δύο μεγάλα πεδία της επιστήμης.

Περίληψη

Η διπλωματική εργασία περιγράφει ένα ολοκληρωμένο προϊόν μιας πεταλιέρας για ηλεκτρική κιθάρα. Ξεκινώντας από την έξοδο της κιθάρας, το ακουστικό σήμα περνάει μέσω του κυκλώματος προ-ενισχυτή, ο μικροελεγκτής εφαρμόζει τα ψηφιακά εφέ που είναι επιλεγμένα από τον χρήστη και έπειτα στέλνεται το σήμα στο ηχείο. Ο ήχος φιλτράρεται και ενισχύεται από ηλεκτρονικό κύκλωμα πριν σταλθεί στο ADC του μικροελεγκτή. Ο μικροελεγκτής STM32H7 είναι αρμόδιος για την παραγωγή των εφέ, μέσω αλγορίθμων που αξιοποιούν γνωστές μαθηματικές συναρτήσεις οι οποίες περιγράφουν ένα εφέ. Τα εφέ μπορούν να τοποθετηθούν σε σειρά, μέχρι και 4 την φορά, ώστε να προσομοιωθεί μια πεταλιέρα πολλαπλών εφέ, μέσω μιας και μόνο διεπαφής. Η διεπαφή χρησιμοποιεί οθόνη, κουμπιά και ποτενσιόμετρα ώστε ο χρήστης να μπορεί να επιλέξει εύκολα και γρήγορα τα εφέ που θέλει και να ρυθμίσει κατάλληλα τις παραμέτρους τους όπως αυτός επιθυμεί. Σχεδιάστηκε και υλοποιήθηκε PCB πλακέτα για το κύκλωμα και την διεπαφή όπως και 3D εκτυπωμένο κουτί. Για την ανάπτυξη της εργασίας χρειάστηκε εκτεταμένη έρευνα πάνω στους αλγορίθμους που απαρτίζουν ένα εφέ και συνδυασμό των γνώσεων μου πάνω στα ενσωματωμένα συστήματα και στην ηλεκτρονική. Ως αποτέλεσμα δημιουργήθηκε μια πεταλιέρα με συνολικά 5 διαφορετικά εφέ, distortion, echo, reverb, noise gate και compressor.

«Digital multi-effect pedal for the electric guitar»

«Liogka Konstantina»

Abstract

The thesis describes a complete pedal board for the electric guitar. Starting from the guitar's output, the audio signal passes through the pre-amplifier circuit, whereafter the microcontroller applies the user-selected digital effects and sends the signal to the speaker. The sound is filtered and amplified by an electronic circuit before being sent to the microcontroller's ADC. The STM32H7 microcontroller is responsible for producing the effects, through algorithms that utilize well-known mathematical functions which describe an effect. The effects can be placed in series, up to 4 at a time, in order to simulate a multi-effect pedalboard through a single interface. The interface uses a screen, buttons, and potentiometers so that the user can easily and quickly select the desired effects and adjust their parameters as they wish. A PCB board was designed and implemented for the circuit and the interface, as well as a 3D-printed enclosure. The development of the thesis required extensive research into the algorithms that make up an effect, combined with my knowledge of embedded systems and electronics. As a result, a pedalboard was created with a total of 5 different effects: distortion, echo, reverb, noise gate, and compressor.

Ευχαριστίες

Καταρχάς θα ήθελα να ευχαριστήσω τον σύντροφο μου για την υποστήριξη και ενθάρρυνση που μου παρείχε καθ’ όλη την διάρκεια της εκπόνησης της διπλωματικής μου, όπως φυσικά και του γονείς μου για την βοήθεια και τις θυσίες που έκαναν τόσα χρόνια για να φτάσω στο σημείο που είμαι.

Επίσης οφείλω τις θερμές ευχαριστίες μου στον επιβλέποντα καθηγητή μου κ. Χατζόπουλο Αργύριο για την καθοδήγηση του, όπως επίσης και στους συμφοιτητές και καθηγητές μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
BJT	Bipolar Junction Transistor
IC	Integrated Circuit
DSP	Digital Signal Processor/ing
DAW	Digital Audio Workstation
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
SNR	Signal-to-Noise Ratio
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
GPIO	General Purpose Input/Output
SPI	Serial Peripheral Interface
PCB	Printed Circuit Board
IDE	Integrated Development Environment
DR	Digital Register
DMA	Direct Memory Access
PLL	Phase-Locked Loop
MPU	Memory Protection Unit
PLA	Polylactide

Κεφάλαιο 1ο: Ιστορική Αναδρομή

1.1 Εισαγωγή

Η μουσική έχει υποστεί τεράστιες αλλαγές τα τελευταία 100 χρόνια, με την εμφάνιση πολλών μουσικών σχημάτων και καλλιτεχνών οι οποίοι αξιοποιούν ακουστικά εφέ στα όργανά τους ώστε να πετύχουν τον δικό τους μοναδικό “ήχο” στις δημιουργίες τους. Η ηλεκτρική κιθάρα, από την αρχή της ιστορίας της μέχρι και σήμερα, πρόκειται για ένα όργανο άμεσα συνδεδεμένο με την χρήση ακουστικών εφέ και πολυεφέ πεταλιέρων. Σε αυτό το κεφάλαιο θα αναλυθεί η δημιουργία της πρώτης ηλεκτρικής κιθάρας, πως προήλθαν τα ακουστικά εφέ οργάνων, όπως επίσης και η μεταβολή από αναλογικό σε ψηφιακό ήχο.

1.2 Η Εφεύρεση της Ηλεκτρικής Κιθάρας

Η κιθάρα ως όργανο κατέχει μια ιστορία εκατοντάδων χρόνων και υιοθετήθηκε από πληθώρα πολιτισμών, με αποτέλεσμα να αναπτυχθούν πολλές παραλλαγές της. Η μοντέρνα κιθάρα αποτελείται από (συνήθως) 6 χορδές, ξύλινο κοίλες σώμα και ξύλινο λαιμό με τάστα. Ο ήχος ενισχύεται χρησιμοποιώντας τον κενό χώρο του κοίλου σώματος [1]. Πλέον οι πιο συνήθεις παραλλαγές της κιθάρας είναι η ακουστική κιθάρα που χρησιμοποιεί χορδές από ατσάλι με κοίλο σώμα, η κλασσική κιθάρα η οποία διαφέρει από την ακουστική, με την κύρια διαφορά ότι χρησιμοποιεί χορδές από νάιλον ή έντερο και η ηλεκτρική κιθάρα η οποία στηρίζεται μόνο στην ενίσχυση μέσω ηχείου για να παράξει ήχο.

Στις αρχές του 20ου αιώνα, αναπτύχθηκε η ανάγκη για επιπλέον ενίσχυση του ήχου καθώς οι ορχήστρες μεγάλωναν και οι συναυλίες συγκέντρωναν όλο και περισσότερο κοινό. Οι μουσικοί χρειαζόντουσαν όλο και δυνατότερο ήχο για να καλύψουν αυτή την ανάγκη. Η πρώτη εφεύρεση της ηλεκτρικής κιθάρας δεν είναι βέβαιη, καθώς πολλοί πειραματιστές στις αρχές του 1920 χρησιμοποιούσαν διάφορους τρόπους να ενισχύσουν τον ήχο, όπως με την χρήση μικροφώνων τοποθετημένα πάνω στην γέφυρα του σώματος της κιθάρας [2]. Η πρώτη ηλεκτρική κιθάρα με κωδικό όνομα “Frying Pan” που έγινε διαθέσιμη στην αγορά δημιουργήθηκε το 1931 από τον George Beauchamp, τον γενικό γραμματέα του National String Instrument Corporation, υπό την αιγίδα του αντιπροέδρου Paul Barth και του Adolph Rickenbacker για την δημιουργία των ηλεκτρομαγνητών. Το τελικό σχέδιο της κιθάρας διέφερε αρκετά σε σύγκριση με αυτό της μοντέρνας ηλεκτρικής κιθάρας, καθώς ακολούθησε το πρότυπο της lap steel κιθάρας, ένα είδος κιθάρας το οποίο χρησιμοποιείται καθιστά. Τα επόμενα χρόνια διάφοροι κατασκευαστές προσπάθησαν να δημιουργήσουν τις δικές τους παραλλαγές ηλεκτροακουστικής κιθάρας, δηλαδή μιας ακουστικής κιθάρας που χρησιμοποιεί μαγνήτες, ωστόσο δεν είχαν ιδιαίτερη επιτυχία στην αγορά καθώς υπήρχαν προβλήματα στην ενίσχυση του ήχου λόγω σχεδίασης [3].

Εικόνα 1.1: Frying Pan: Η πρώτη ηλεκτρική κιθάρα διαθέσιμη στην αγορά [4].

Την δεκαετία του 1940 ένας κιθαρίστας και εφευρέτης θέλοντας να τελειοποιήσει το όργανο από άποψη ήχου και σχεδίασης, δημιούργησε μια κιθάρα σε σχήμα κλασσικής, η οποία αντί για κοίλο σώμα χρησιμοποιούσε ένα συμπαγές κομμάτι ξύλου (solid-body), ονομάζοντας την “The Log”. Ο εφευρέτης αυτός είναι γνωστός ως Les Paul και η εφεύρεση του επηρέασε ολόκληρη την μουσική βιομηχανία, δημιουργώντας το πρότυπο της ηλεκτρικής κιθάρας.

Εικόνα 1.2: Η πρώτη ηλεκτρική κιθάρα του Les Paul [5].

Τις επόμενες δεκαετίες έγιναν διάφορες αλλαγές πάνω στο πρωτότυπο του Les Paul ώστε να γίνει διαθέσιμο στο κοινό. Το 1950, οι πρώτες solid-body ηλεκτρικές κιθάρες που διατέθηκαν στην αγορά ήταν η Fender Esquire και η Fender Broadcaster (ή αργότερα Telecaster) από την Fender Musical Instruments Corporation. Μια ακόμη γνωστή σειρά μοντέλων της Fender είναι η Fender Stratocaster. Λίγα χρόνια μετά, η Gibson Guitar Corporation δημιούργησε την Gibson Les Paul ώστε να ανταγωνιστεί τα μοντέλα της Fender.

Εικόνα 1.3: Από αριστερά στα δεξιά: Fender Esquire, Fender Broadcaster, Gibson Les Paul, Fender Stratocaster.

Από το 1950 έως σήμερα η θεωρία της ηλεκτρικής κιθάρας έχει παραμείνει ίδια, με μερικές αλλαγές στους τύπους των μαγνητών, στον σχεδιασμό του σώματος και την προσθήκη προ-ενισχυτή πάνω στην κιθάρα τροφοδοτούμενο από μπαταρία

Η χρήση της ηλεκτρικής κιθάρας από μουσικούς καλλιτέχνες και μπάντες ήταν ραγδαία, ειδικότερα στην εποχή όπου ο ήχος μπορούσε πλέον να διαδοθεί μέσω τηλεόρασης και ραδιοφώνου, όπως στην περίπτωση της Rosetta Tharpe χρησιμοποιώντας την Gibson Les Paul SG ηλεκτρική κιθάρα.

1.3 Αναλογικά Εφέ Ήχου

1.3.1 Το Αναλογικό Πετάλι Εφέ Ήχου

Το αναλογικό πετάλι εφέ πρόκειται για μια συσκευή που χρησιμοποιεί αναλογικά ηλεκτρονικά εξαρτήματα για την παραμόρφωση, ενίσχυση ή αλλαγή του τόνου σε πραγματικό χρόνο. Η βασική αρχή λειτουργίας τους στηρίζεται στην εφαρμογή θεμελιωδών αρχών ηλεκτρονικής μηχανικής, που κάθε κύκλωμα επιτελεί έναν συγκεκριμένο ρόλο στη διαμόρφωση του ήχου. Ο πυρήνας ενός αναλογικού πεταλιού αποτελείται από ηλεκτρονικά εξαρτήματα όπως αντιστάσεις, πυκνωτές, διόδους, τρανζίστορ

και ολοκληρωμένα κυκλώματα τελεστικών ενισχυτών που συνδυάζονται για να δημιουργήσουν τα επιθυμητά ηχητικά αποτελέσματα.

1.3.2 Τα Πρώτα Αναλογικά Εφέ Ήχου

Στην αρχή τα εφέ ήχου καταγράφονταν σε στούντιο παραγωγής και μετέπειτα αναπαράγονταν σε ζωντανές παραστάσεις και συναυλίες. Προφανώς η τεχνική αυτή στηρίζονταν στην χρήση ήδη υπαρχόντων ηχογραφήσεων, με περιορισμένη γκάμα ηχητικών εφέ και ελλειπή δυνατότητα αλλαγής του ήχου σε πραγματικό χρόνο. Έτσι, στις αρχές του 1940, εμφανίστηκε για πρώτη φορά η αναλογική καθυστέρηση (analog delay) ήχου με την χρήση ταινιών (reel to reel) από τον Les Paul [6]. Στην συνέχεια ο Harold DeArmond ξεκίνησε την παραγωγή και πώληση της πρώτης μονάδας εφέ, Model 601 Tremolo Control υπό την εταιρεία Rowe Industries [7]. Τα περισσότερα εφέ ωστόσο ήταν ενσωματωμένα στους ενισχυτές κιθάρας και όχι ως αυτόνομες συσκευές. Πολλές φορές η δημιουργία νέων εφέ γίνονταν τυχαία, όπως στην περίπτωση της παραμόρφωσης (distortion) που ο ήχος ενισχύονταν πέρα από τα όρια των ενισχυτών λυχνίας, με αποτέλεσμα το ημιτονικό σήμα της κιθάρας να αποκόβεται και να τετραγωνίζεται, δημιουργώντας τον χαρακτηριστικό “σκληρό” ήχο.

Η εφεύρεση του τρανζίστορ επέφερε ραγδαία εξέλιξη στην επιστήμη της ηλεκτρονικής και ως κατ’ επέκταση στην επιστήμη της επεξεργασίας ήχου με την ενσωμάτωσή τους στα κυκλώματα εφέ στις αρχές του 1960. Παραδείγματος χάρη, το Maestro Fuzz Tone πετάλι χρησιμοποιήθηκε για να δημιουργήσει τον χαρακτηριστικό ήχο του τραγουδιού (I Can't Get No) Satisfaction των Rolling Stones [8].

Εικόνα 1.4: Maestro Fuzz Tone πετάλι κιθάρας [9].

Αργότερα κατά τις δεκαετίες του 1960 και 1970 κατασκευαστές όπως Vox, Boss και Dunlop άλλαξαν την αγορά δημιουργώντας πετάλια εφέ με μοναδικό ήχο και ποιότητα. Το κάθε πετάλι, ακόμα και αν βασίζονταν σε ένα συγκεκριμένο εφέ, ήταν φτιαγμένο έτσι ώστε να ξεχωρίζει από το άλλο. Οι μουσικοί μπορούσαν πλέον να δημιουργήσουν τον δικό τους χαρακτηριστικό ήχο συνδυάζοντας τα πετάλια μεταξύ τους, με το καθένα να παρέχει τις δικές του ρυθμίσεις μέσω ποτενσιόμετρων και διακοπτών [10]
.

1.4 Το Τρανζίστορ

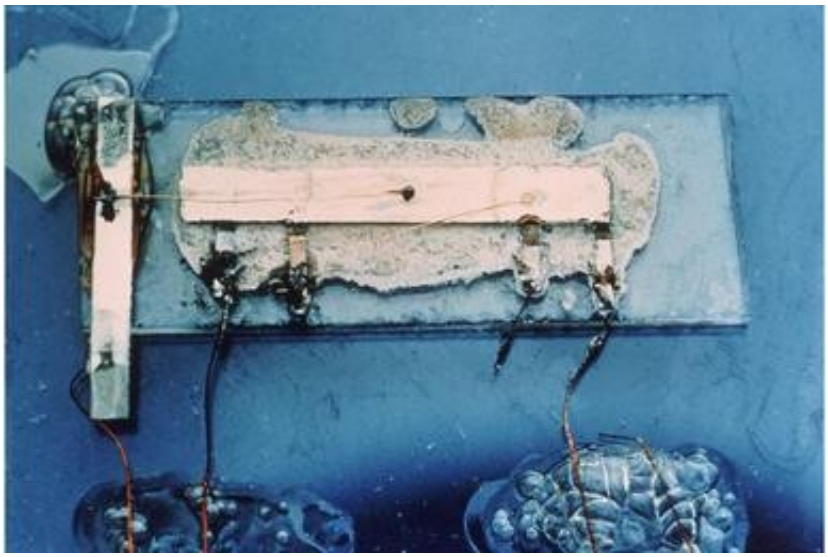
Το τρανζίστορ, ή αλλιώς όπως συχνά περιγράφεται και ως η σημαντικότερη ανακάλυψη του 20ου αιώνα, αποτέλεσε το κύριο έναυσμα για την εξέλιξη των πεταλιών ήχου. Η ανάπτυξή του δεν ήταν αποτέλεσμα μιας ενιαίας έμπνευσης, αλλά μιας σειράς πειραμάτων και τυχαίων ανακαλύψεων. Το 1947, η ομάδα του William Shockley στα Bell Laboratories ανέπτυξε τον πρώτο τρανζίστορ σημείου επαφής (point-contact transistor), αντικαθιστώντας τη λυχνία τριόδου που μέχρι τότε κυριαρχούσε στα ηλεκτρονικά κυκλώματα. Σε αντίθεση με τις λυχνίες, οι οποίες απαιτούσαν θέρμανση ηλεκτροδίου σε υψηλές θερμοκρασίες και είχαν σημαντικούς περιορισμούς στη λειτουργία υψηλών συχνοτήτων, το τρανζίστορ ήταν συμπαγές, μικρό σε μέγεθος, αξιόπιστο και ενεργειακά αποδοτικό. Λίγο αργότερα, το 1948, ο Shockley ανέπτυξε το διπολικό τρανζίστορ, bipolar junction transistor (BJT), το οποίο έγινε γρήγορα το κυρίαρχο στοιχείο στα ηλεκτρονικά κυκλώματα της δεκαετίας του 1950 και του 1960, ανοίγοντας τον δρόμο για τη συρρίκνωση των ηλεκτρονικών και κατ' επέκταση για την ανάπτυξη των πρώτων φορητών πεταλιών ήχου. Το τρανζίστορ αποτελείται από μια διάταξη τριών ημιαγώγιμων ακροδεκτών, εκπομπός, βάση και συλλέκτης και η κύρια λειτουργία του είναι να φέρεται ως ελεγχόμενος διακόπτης [11].

Εικόνα 1.5: Το πρώτο transistor του William Shockley [12].

1.5 Το Ολοκληρωμένο Κύκλωμα

Η εφεύρεση του τρανζίστορ άνοιξε τον δρόμο για το επόμενο μεγάλο τεχνολογικό άλμα, το ολοκληρωμένο κύκλωμα, integrated circuit (IC). Το 1958, ο μηχανικός Jack Kilby της Texas Instruments συνέλαβε μια ιδέα που θα άλλαζε για πάντα την ηλεκτρονική, δηλαδή την κατασκευή αντιστάσεων, πυκνωτών και τρανζίστορ πάνω σε μια ενιαία πλάκα πυριτίου. Αντί δηλαδή να συνδέονται διακριτά εξαρτήματα με καλώδια και συγκολλήσεις, που ήταν μια ακριβή διαδικασία με συχνές βλάδες και λάθη , όλα τα στοιχεία του κυκλώματος θα κατασκευάζονταν μαζί σε ένα ενιαίο

πακέτο. Έτσι, το πρώτο λειτουργικό IC παρουσιάστηκε το 1958 [13]. Η δυνατότητα ενσωμάτωσης πολύπλοκων κυκλωμάτων σε εξαιρετικά μικρό χώρο και χαμηλό κόστος έκανε εφικτή την κατασκευή συμπαγών, φορητών και οικονομικά προσιτών συσκευών και αποτέλεσε τη βάση για την ανάπτυξη μικροελεγκτών και Digital Signal Processor (DSP) που αργότερα βρήκαν εφαρμογή στα ψηφιακά πετάλια ήχου.



Εικόνα 1.6: Το πρώτο ολοκληρωμένο κύκλωμα του Jack Kilby [14].

1.6 Η Μετάβαση στον Ψηφιακό Ήχο

Οι συνεχείς εξελίξεις στην τεχνολογία την δεκαετία του 1980 επηρέασε σημαντικά την επιστήμη της επεξεργασίας σήματος καθώς για πρώτη φορά εμφανίστηκε η έννοια του “ψηφιακού” ήχου. Η μετάβαση από την αναλογική στην ψηφιακή επεξεργασία δεν ήταν απλώς μια τεχνολογική αναβάθμιση, αφού αποτέλεσε θεμελιώδη αλλαγή στον τρόπο με τον οποίο ο ήχος αναπαριστάται, αποθηκεύεται και επεξεργάζεται. Πολύ γρήγορα εμφανίστηκαν πετάλια τα οποία περιείχαν ψηφιακά ηλεκτρονικά που μπορούσαν να επεξεργαστούν τον ήχο μέσω ολοκληρωμένων κυκλωμάτων όπως μικροελεγκτών και DSP .

Ένα χαρακτηριστικό παράδειγμα αυτής της νέας γενιάς συσκευών είναι το HM80: The Baby Harmonizer, ένας αρμονιστής που χρησιμοποιούσε ψηφιακή επεξεργασία για να μετατοπίζει το ύψος (pitch) του εισερχόμενου σήματος σε πραγματικό χρόνο. Η λειτουργία του βασιζόταν σε γνωστούς αλγορίθμους επεξεργασίας ήχου, καθώς η συσκευή ανέλυε το εισερχόμενο ηχητικό σήμα, υπολόγιζε τη θεμελιώδη συχνότητά του και παρήγαγε ένα νέο σήμα σε διαφορετικό τονικό ύψος, το οποίο μπορούσε να αναμιχθεί με το αρχικό.

Εικόνα 1.7: HM80: The Baby Harmonizer [15].

Τα επόμενα χρόνια, μεγάλοι κατασκευαστές όπως Boss, Line 6 και Zoom δημιούργησαν τις δικές τους συσκευές ικανές να αναπαράγουν όλα τα είδη εφέ, σε συνδυασμό, σε πραγματικό χρόνο. Αυτές οι συσκευές ονομάζονται πολυεφέ πεταλιέρες και δίνουν την δυνατότητα στους μουσικούς να πειραματιστούν με τον ήχο με την χρήση μιας και μόνο συσκευής.

1.7 Επίλογος

Το κεφάλαιο αυτό περιέγραψε αναδρομικά την ιστορική εξέλιξη της ηλεκτρικής κιθάρας, των αναλογικών εφέ με την χρήση πεταλιών και την σημαντική τεχνολογική πρόοδο που επέφερε το τρανζίστορ για την ανάπτυξη ολοκληρωμένων κυκλωμάτων. Ως αποτέλεσμα μπορεί να αναλυθεί στα παρακάτω κεφάλαια η διαφορά ανάμεσα στον αναλογικό και ψηφιακό ήχο, η ενσωμάτωση των μικροελεγκτών και DSP στην επεξεργασία ήχου και το θεωρητικό υπόβαθρο των διακριτών σημάτων.

Κεφάλαιο 2ο: Σύγχρονα Ψηφιακά Συστήματα Ήχου

2.1 Εισαγωγή

Ενώ τα ακριβά επαγγελματικά συστήματα ήχου, όπως οι ενισχυτές, παραμένουν πιστά στον αναλογικό ήχο με την χρήση λυχνιών έναντι ολοκληρωμένων, ο ψηφιακός ήχος κυριαρχεί στην υπόλοιπη αγορά ηλεκτρονικών. Αυτό συμβαίνει διότι η τεχνολογία των τρανζίστορ είναι απλούστερη και φθηνότερη στην παραγωγή, ενώ παράλληλα είναι σημαντικά μικρότερη σε μέγεθος. Συσκευές όπως ακουστικά, μικρόφωνα και ενισχυτές διατίθενται πλέον στον μέσο καταναλωτή σε πολύ χαμηλές τιμές, με ικανοποιητική απόδοση και ποιότητα. Σε αυτό το κεφάλαιο θα επεξηγηθούν αναλυτικά οι βασικές αρχές της επεξεργασίας του ψηφιακού ήχου, η χρήση αυτού στα ηχοσυστήματα, όπως και τα πρότυπα για την μετάδοση του.

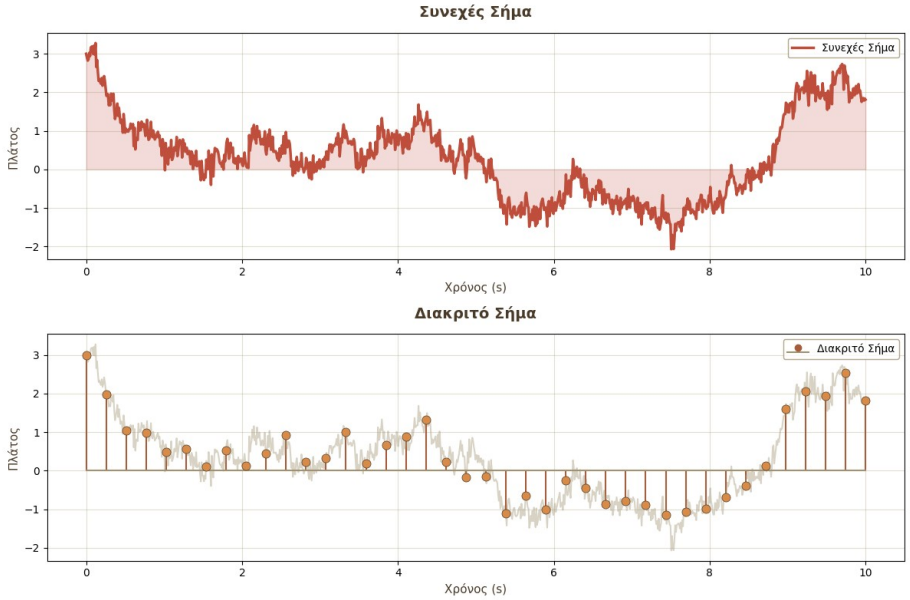
2.2 Θεμελιώδεις Αρχές Ψηφιακής Επεξεργασίας Ήχου

Στην επεξεργασία σήματος, ως σήμα ορίζεται μια συνάρτηση που περιγράφει πώς μια ποσότητα εξαρτάται από μια άλλη. Το ακουστικό σήμα χαρακτηρίζεται ως η μεταβολή της ηλεκτρικής τάσης συναρτήσει του χρόνου, η οποία, όταν μετατραπεί σε ηχητικά κύματα, καθίσταται αντιληπτή από την ανθρώπινη ακοή. Η δημιουργία του ακουστικού σήματος μπορεί να προέλθει είτε από τον φυσικό κόσμο, όπως μέσω κραδασμών, δονήσεων ή ανθρώπινης ομιλίας, είτε να κατασκευαστεί εξ ολοκλήρου από ένα ψηφιακό μέσο, όπως ένα συνθεσάιζερ [16].

2.2.1 Σήματα και Συστήματα Διακριτού Χρόνου

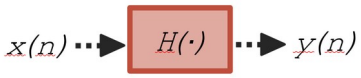
Όταν τόσο η εξαρτημένη μεταβλητή του πλάτους όσο και η ανεξάρτητη μεταβλητή του χρόνου μπορούν να λάβουν οποιαδήποτε τιμή σε ένα συνεχές διάστημα, αναφερόμαστε σε ένα συνεχές σήμα (continuous -time signal), το οποίο αποτελεί την κυρίαρχη μορφή σημάτων στη φύση.

Ένα σήμα διακριτού χρόνου (discrete-time signal) προκύπτει όταν η ανεξάρτητη μεταβλητή λαμβάνει διακριτές, ισαπέχουσες συνήθως, τιμές. Στην πράξη, αυτό σημαίνει ότι το σήμα ορίζεται μόνο σε συγκεκριμένες χρονικές στιγμές, οι οποίες ονομάζονται δείγματα (samples) [17].



Εικόνα 2.1: Παράδειγνα σύγκρισης ενός σήματος συνεχούς χρόνου με ενός σήματος διακριτού χρόνου.

Στην επεξεργασία σήματος, ένα σύστημα χαρακτηρίζεται ως ένα σύνολο ενεργειών που δέχονται ένα σήμα εισόδου και παράγουν ένα σήμα εξόδου. Το σύστημα μπορεί να παρουσιάζει φυσική μορφή, όπως ένα ηλεκτρονικό κύκλωμα σαν το μικρόφωνο ή τον ενισχυτή, αλλά μπορεί και να παρίσταται ως ένα σύνολο ψηφιακών αλγορίθμων όπως ένα πρόγραμμα Digital Audio Workstation (DAW).



Σχήμα 2.1: Παράδειγμα συστήματος.

Προφανώς, τα συστήματα, όπως και τα σήματα, χωρίζονται σε συνεχή και διακριτά. Στα συστήματα διακριτού χρόνου, όπως και στα σήματα διακριτού χρόνου, οι ιδιότητες τους αλλάζουν σε συγκεκριμένα διαστήματα χρόνου, που συνήθως πρόκειται για τις ίδιες ισαπέχουσες τιμές με τις οποίες δειγματοληπτείται το αρχικό σήμα. Επιπροσθέτως, υπάρχουν δύο ακόμη υποκατηγορίες συστημάτων, ανάλογα με την γραμμικότητα τους. Αρχικά, ένα γραμμικό σύστημα περιγράφεται ως ένα σύστημα που ακολουθεί τις αρχές της υπέρθεσης και ομοιογένειας [18].

Για παράδειγμα, έστω τα συστήματα:

$$y_1 = H(x_1)$$
$$y_2 = H(x_2) \quad (2.1)$$

Το τελικό σήμα εξόδου ακολουθάει την αρχή της υπέρθεσης ή επαλληλίας, όταν το αποτέλεσμα του είναι το άθροισμα των ξεχωριστών αποτελεσμάτων. Δηλαδή όταν ισχύει:

$$y_1 + y_2 = H(x_1) + H(x_2) = H(x_1 + x_2) \tag{2.2}$$

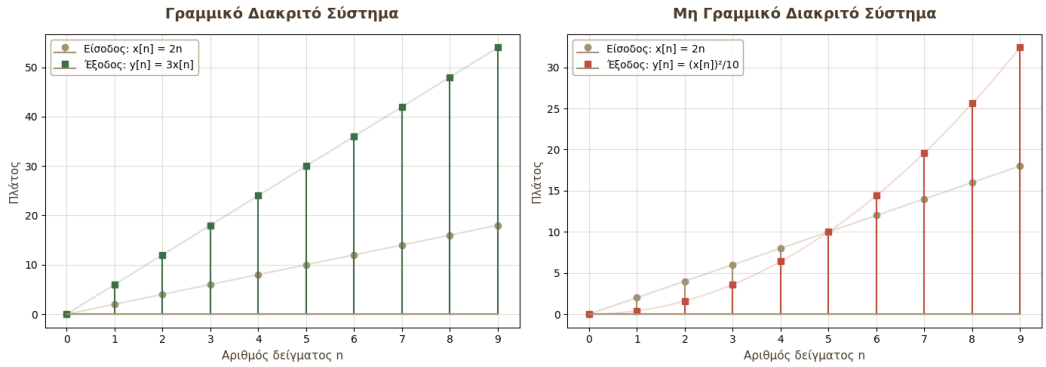
Από την άλλη, η αρχή της ομοιογένειας ορίζει πως η αναλογία εισόδου και εξόδου παραμένει σταθερή. Για παράδειγμα, έστω το σύστημα:

$$y = H(x) \tag{2.3}$$

Το σύστημα αυτό υπακούει στην αρχή της ομοιογένειας στην περίπτωση που κλιμακωθεί η είσοδος κατά *a* τότε και η έξοδος θα κλιμακωθεί κατά τόσο. Δηλαδή όταν ισχύει:

$$a \cdot y = a \cdot H(x) = H(a \cdot x) \tag{2.4}$$

Έτσι, όταν και οι δύο αρχές ισχύουν για ένα σύστημα, τότε το σύστημα είναι γραμμικό. Στη διαφορετική περίπτωση που δεν ισχύουν οι συγκεκριμένοι κανόνες, το σύστημα ονομάζεται μη γραμμικό.



Εικόνα 2.2: Παράδειγμα σύγκρισης ενός γραμμικού με ενός μη γραμμικού διακριτού συστήματος.

2.2.2 Κρουστική Απόκριση Συστήματος σε Ψηφιακό Σήμα

Η απόκριση συστήματος σε ένα διακριτό και χρονικά ανεξάρτητο σύστημα περιγράφει τον τρόπο με τον οποίο το σύστημα αντιδρά σε ένα σήμα εισόδου και χαρακτηρίζεται πλήρως από την απόκριση στο μοναδιαίο παλμό h[n]. Στα φίλτρα πεπερασμένης κρουστικής απόκρισης συστήματος finite impulse response (FIR), η απόκριση στο παλμό έχει πεπερασμένη διάρκεια, επειδή η έξοδος εξαρτάται μόνο από ένα πεπερασμένο πλήθος δειγμάτων της εισόδου, γεγονός που εξασφαλίζει εγγενή σταθερότητα και απουσία ανατροφοδότησης. Αντίθετα, στα φίλτρα άπειρης κρουστικής απόκρισης infinite impulse response (IIR), η απόκριση στο παλμό είναι άπειρης διάρκειας, καθώς η έξοδος εξαρτάται τόσο από τρέχουσες και προηγούμενες εισόδους όσο και από προηγούμενες εξόδους με την χρήση της αναδρομής (feedback), κάτι που επιτρέπει πιο αποδοτική υλοποίηση αλλά καθιστά το σύστημα λιγότερο σταθερό.

2.2.3 Δειγματοληψία και Συχνότητα Nyquist

Η δειγματοληψία είναι η διαδικασία μετατροπής ενός συνεχούς σήματος σε μια ακολουθία διακριτών τιμών που λαμβάνονται σε, συνήθως ισαπέχοντα, χρονικά διαστήματα. Στην πράξη, ένα κύκλωμα δειγματοληψίας καταγράφει την τιμή του αναλογικού σήματος μόνο σε συγκεκριμένες χρονικές στιγμές, αγνοώντας οποιαδήποτε μεταβολή συμβαίνει ενδιάμεσα. Η συχνότητα με την οποία λαμβάνονται αυτά τα δείγματα ονομάζεται συχνότητα δειγματοληψίας και είναι θεμελιώδης

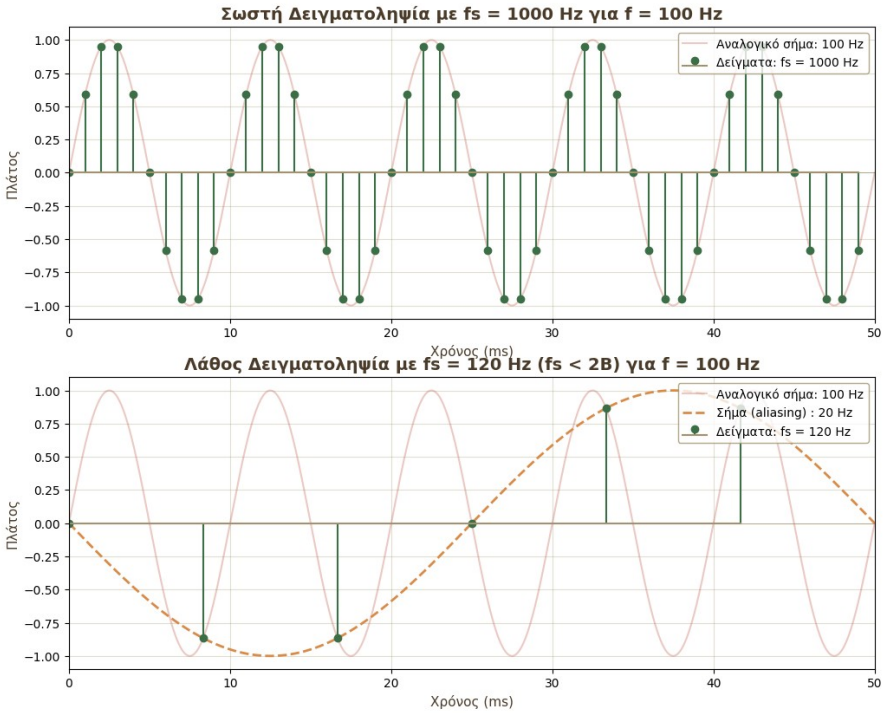
παράμετρος για την πιστή αναπαράσταση του αρχικού σήματος. Το θεώρημα Nyquist-Shannon ορίζει ότι για να μπορεί ένα συνεχές σήμα να ανακατασκευαστεί πλήρως από τα δείγματά του, η συχνότητα δειγματοληψίας (fs) πρέπει να είναι τουλάχιστον διπλάσια από την υψηλότερη συχνότητα που περιέχεται στο σήμα (B). Η συχνότητα που ισούται με το διπλάσιο της συχνότητας του σήματος ονομάζεται συχνότητα Nyquist.

$$f_s > 2B \tag{2.5}$$

Όταν το σήμα περιέχει συχνότητες πάνω από το όριο αυτό, εμφανίζεται το φαινόμενο της αναδίπλωσης (aliasing): οι υψηλές συχνότητες "μεταμφιέζονται" σε χαμηλότερες, δημιουργώντας ανεπιθύμητες συνιστώσες που αναμιγνύονται με το κανονικό σήμα και καθιστούν αδύνατη την ανακατασκευή του. Για τον λόγο αυτό, πριν από τη δειγματοληψία χρησιμοποιείται πάντα ένα χαμηλοπερατό φίλτρο που περιορίζει το φάσμα του σήματος κάτω από τη συχνότητα Nyquist [19].

2.2.4 Το Πρόβλημα του Κβαντισμού

Η κβαντοποίηση είναι η διαδικασία μετατροπής ενός συνεχούς μεγέθους (όπως η τάση) σε μια διακριτή τιμή, συνήθως έναν ακέραιο αριθμό. Το σφάλμα κβαντοποίησης συμπεριφέρεται στην πράξη σαν θόρυβος που προστίθεται στο σήμα. Στις περισσότερες περιπτώσεις, αυτός ο θόρυβος είναι ομοιόμορφα κατανεμημένος και δεν σχετίζεται με το ίδιο το σήμα, γεγονός που μας επιτρέπει να τον αντιμετωπίζουμε ως τυχαίο θόρυβο. Κάθε στάθμη αντιστοιχεί σε ένα συγκεκριμένο εύρος τάσης, με αποτέλεσμα όλες οι τιμές που βρίσκονται μέσα σε αυτό το εύρος να αντιπροσωπεύονται από τον ίδιο αριθμό. Αυτό εισάγει ένα σφάλμα, γνωστό ως σφάλμα κβαντοποίησης, το οποίο μπορεί να φτάσει έως ±½ του βήματος κβαντοποίησης [20]. Για παράδειγμα, για ένα σήμα 100Hz, το όριο Nyquist είναι > 200 Hz. Αν η δειγματοληψία γίνεται στα 1000Hz, δηλαδή 1000 δείγματα το δευτερόλεπτο, τα δείγματα αποτυπώνουν πιστά την αρχική κυματομορφή και το σήμα ανακατασκευάζεται χωρίς κανένα πρόβλημα. Αντίθετα, αν η δειγματοληψία γίνει στα 150 Hz τότε είναι κάτω από το όριο Nyquist, με αποτέλεσμα να εμφανιστεί το φαινόμενο του aliasing. Λόγω της ανεπαρκούς δειγματοληψίας, το σήμα των 100 Hz "μεταμφιέζεται" και εμφανίζεται ψευδώς ως ένα σήμα 20 Hz (120 Hz - 100 Hz = 20 Hz). Έτσι, ενώ το σήμα εισόδου έχει συχνότητα 100Hz, μετά από δειγματοληψία με συχνότητα κάτω από το όριο Nyquist, θα προκληθεί απώλεια πληροφορίας.



Εικόνα 2.3: Παράδειγμα σωστής και λανθάνουσας δειγματοληψίας για αναλογικό σήμα συχνότητας 100Hz.

2.2.5 Λόγος σήματος προς θόρυβο

Ο λόγος σήματος προς θόρυβο, ή αλλιώς Signal to Noise Ratio (SNR), πρόκειται για μια μέτρηση που περιγράφει την σύγκριση ανάμεσα στην ένταση ενός σήματος και στον θόρυβο. Ο θόρυβος είναι ανεξάρτητος από το σήμα και μπορεί να προκαλείται από πληθώρα περιπτώσεων, όπως παρεμβολές, ποιότητα εξοπλισμού και καλωδίων ή μέσω της διαδικασίας της ψηφιακής μετατροπής [21]. Ο λόγος ορίζεται ως η ένταση του σήματος προς την ένταση του θορύβου.

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{2.6}$$

Όσο μεγαλύτερο SNR υπολογίζεται, τόσο λιγότερος θόρυβος υπάρχει στο περιβάλλον, οπότε καλύτερη ποιότητα ήχου.Πρακτικά όμως το σήμα και ο θόρυβος μετριούνται σε decibels οπότε ο λόγος μεταφράζεται στην λογαριθμική κλίμακα.

$$SNR = 10 \log_{10} \left(\frac{P_{signal}}{P_{noise}} \right) \tag{2.7}$$

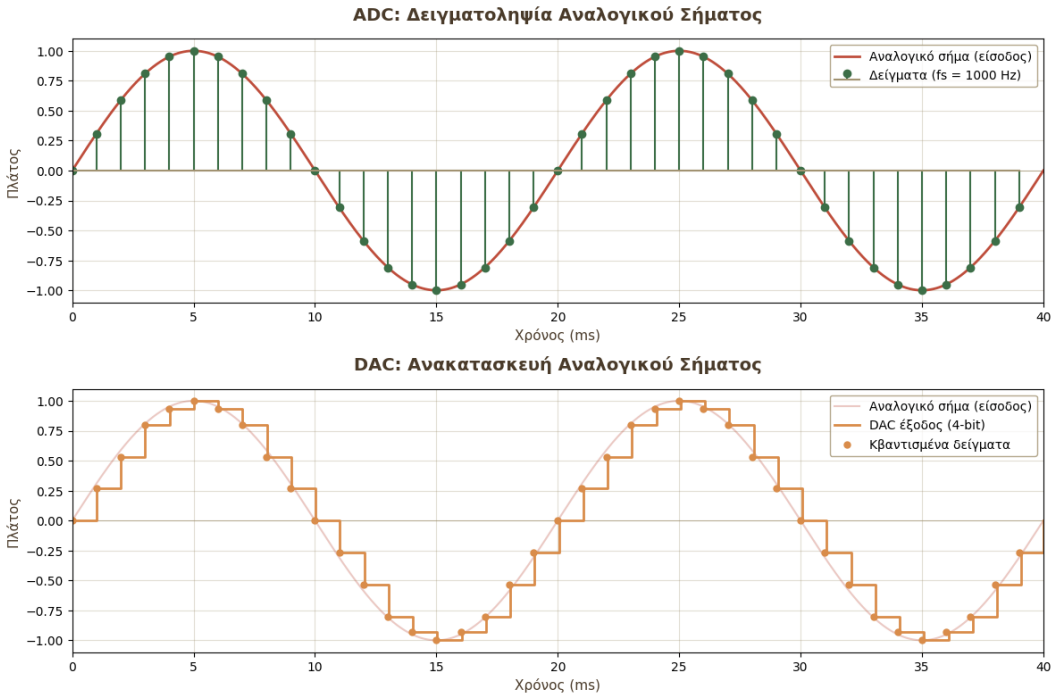
Σε αυτήν την περίπτωση εάν το SNR υπολογισθεί αρνητικό, τότε υπάρχει περισσότερος θόρυβος από ότι καθαρό σήμα ενώ εάν υπολογισθεί θετικό, τότε το καθαρό σήμα υπερτερεί.

2.2.6 Αναλογική και Ψηφιακή Μετατροπή Σήματος

Όταν ένα αναλογικό σήμα εισέρχεται σε έναν μετατροπέα αναλογικού σήματος σε ψηφιακό (ADC), η συνεχής τάση μετατρέπεται σε μια ακολουθία ακεραίων αριθμών. Η λειτουργία του περιλαμβάνει δύο βασικά στάδια. Αρχικά, ένα κύκλωμα δειγματοληψίας "παγώνει" την τιμή του αναλογικού σήματος σε τακτά χρονικά διαστήματα, κρατώντας τη σταθερή για όσο χρειάζεται. Στη συνέχεια, μετατρέπει αυτή την τάση σε έναν ακέραιο αριθμό. Η ακρίβεια αυτής της μετατροπής καθορίζεται από την ανάλυση (resolution) του μετατροπέα, που μετριέται σε bits (π.χ. 12-bit, 16-bit ή 24-bit). Όσο περισσότερα bits έχει ένας ADC, τόσο μικρότερο είναι το σφάλμα κβαντισμού που εισάγεται στη μετατροπή και τόσο πιο πιστά μπορεί να αναπαραστήσει το αρχικό σήμα.

Ένας μετατροπέας ψηφιακού σήματος σε αναλογικό (DAC) τελεί την αντίστροφη διαδικασία. Συλλέγει την ψηφιακή πληροφορία από τον επεξεργαστή και την μετατρέπει ξανά σε μια συνεχή τάση, ώστε να μπορέσει να σταλεί σε έναν ενισχυτή ή ένα ηχείο και να γίνει αντιληπτή ως ήχος. Κατά την ανακατασκευή του σήματος, ο DAC χρησιμοποιεί συνήθως μια τεχνική που ονομάζεται διάταξη μηδενικής τάξης (zero-order hold). Αυτό σημαίνει ότι κρατά σταθερή την τάση εξόδου στην τιμή κάθε ψηφιακού δείγματος καθ' όλη τη διάρκεια μέχρι το επόμενο δείγμα, δημιουργώντας ένα χαρακτηριστικό "σκαλοπάτι" στην κυματομορφή. Η ποιότητα ενός DAC επηρεάζεται από την ανάλυσή του, τη γραμμικότητά του, καθώς και από το φιλτράρισμα που ενδέχεται να ακολουθεί για να εξομαλύνει τα σκαλοπάτια του zero-order hold.

Η ποιότητα ενός ψηφιακού συστήματος ήχου εξαρτάται σε μεγάλο βαθμό από την ικανότητα των ADC και DAC να μετατρέπουν τα σήματα με τη μικρότερη δυνατή απώλεια πληροφορίας και την εισαγωγή όσο το δυνατόν λιγότερων σφαλμάτων. Η επιλογή της κατάλληλης ανάλυσης (bits) και συχνότητας δειγματοληψίας είναι κρίσιμη για την τελική ποιότητα ήχου [22].



Εικόνα 2.4: Παράδειγμα προσομοιωμένης λειτουργίας ADC και DAC.

2.3 Μοντελοποίηση Αναλογικών Κυκλωμάτων

Στις σύγχρονες ψηφιακές εφαρμογές ήχου, όπως ένα DAW, στόχος αποτελεί η πιστή μετάφραση ενός αναλογικού κυκλώματος σε ένα ψηφιακό. Η διαδικασία αυτή ονομάζεται φυσική μοντελοποίηση (physical modelling) και αποσκοπεί στην προσομοίωση των φυσικών εξαρτημάτων, όπως ολοκληρωμένα κυκλώματα, αντιστάσεις, πυκνωτές και τρανζίστορ, που σε συγκεκριμένες τοπολογίες παράγουν ήχο [23]. Αυτές οι βαθμίδες μπορούν να περιγραφούν με μαθηματικές εξισώσεις, οι οποίες μετέπειτα μετατρέπονται σε αλγορίθμους που τρέχουν σε ένα ψηφιακό σύστημα, όπως έναν DSP.

Στην πράξη, αυτή η προσέγγιση επιτρέπει την δημιουργία των εικονικών αναλογικών μοντέλων (virtual analog models), όπου αναλύεται το αναλογικό κύκλωμα και κατασκευάζεται ένα υπολογιστικό μοντέλο που προσομοιώνει τη ροή του σήματος και τα γραμμικά και μη γραμμικά συστήματα που εισάγουν τα εξαρτήματα[24]. Η εφαρμογή δεν στηρίζεται σε ηχογραφημένα εφέ και ήχους, αλλά παράγει εξ ολοκλήρου από την αρχή το ψηφιακό σήμα μέσω αλγορίθμων.

Σε μία αναλογική πεταλιέρα εφέ ή ένα αναλογικό συνθεσάιζερ ο χρήστης μπορεί να αλληλεπιδράσει με το κύκλωμα μέσω διακοπτών, ποτενσιόμετρων ή άλλων χειριστηρίων. Τα χειριστήρια αυτά ρυθμίζουν παραμέτρους ελέγχου που καθορίζουν τη συμπεριφορά του σήματος, επιτρέποντας στον χρήστη να διαμορφώσει τον ήχο σε πραγματικό χρόνο. Για παράδειγμα, μπορεί να αυξήσει την παραμόρφωση (distortion) για πιο επιθετικό ήχο, να μεταβάλλει την ταχύτητα μιας διαμόρφωσης (π.χ. tremolo ή vibrato), να ελέγξει την ένταση της ενίσχυσης (gain) ή να ρυθμίσει τον χρόνο επανάληψης σε ένα εφέ καθυστέρησης (delay). Η ενσωμάτωση αυτών των παραμέτρων σε έναν αλγόριθμο είναι κρίσιμη για τη δημιουργία ενός πλήρως παραμετροποιήσιμου εικονικού αναλογικού μοντέλου, καθώς επιτρέπει την πιστή αναπαραγωγή της δυναμικής συμπεριφοράς του αρχικού πρωτότυπου αναλογικού κυκλώματος.

Ένα σημαντικό εμπόδιο για την πιστή ψηφιακή μετατροπή ενός αναλογικού κυκλώματος είναι η έλλειψη πληροφορίας σχετικά με την εσωτερική του αρχιτεκτονική. Αυτό συμβαίνει, για παράδειγμα, όταν το σχηματικό του διάγραμμα δεν είναι προσβάσιμο ή όταν το κύκλωμα περιέχει ολοκληρωμένα των οποίων η εσωτερική δομή δεν τεκμηριώνεται. Σε αυτήν την περίπτωση, το σύστημα αντιμετωπίζεται ως ένα μαύρο κουτί (black box), όπου η μοντελοποίηση βασίζεται αποκλειστικά στη μελέτη της συμπεριφοράς εισόδου-εξόδου και ονομάζεται μοντέλο μαύρου κουτιού (black box model). Αντιθέτως, όταν η εσωτερική δομή ενός αναλογικού κυκλώματος μπορεί να περιγραφεί λεπτομερώς, γνωρίζοντας κατ’ επέκταση τη λειτουργία κάθε εξαρτήματος, τότε η αρχιτεκτονική ακολουθάει το μοντέλο άσπρου κουτιού (white box model). Με αυτό το μοντέλο μπορούν να υπολογιστούν με ακρίβεια οι μαθηματικοί αλγόριθμοι που απαρτίζουν το κύκλωμα, με αποτέλεσμα την αξιόπιστη αναπαράσταση του αναλογικού κυκλώματος σε ψηφιακή εφαρμογή [25].

2.4 Σύγχρονα Συστήματα Ψηφιακής Επεξεργασίας

Αφού αναλύθηκαν οι θεμελιώδεις αρχές μετατροπής ενός αναλογικού σήματος σε ψηφιακό, τίθεται το ερώτημα πού και πώς γίνεται η επεξεργασία αυτών των ψηφιακών δεδομένων. Τα συστήματα ψηφιακής επεξεργασίας σήματος, γνωστά και ως DSP, αποτελούνται από ένα εξειδικευμένο υπολογιστικό σύστημα, είτε ως φυσικός επεξεργαστής είτε ως λογισμικό, το οποίο δέχεται μια ακολουθία αριθμών, εκτελεί πάνω της μια σειρά μαθηματικών πράξεων και παράγει μια νέα επεξεργασμένη ακολουθία.

Σε αντίθεση με τους επεξεργαστές γενικής χρήσης, τα σύγχρονα συστήματα DSP είναι σχεδιασμένα για να εκτελούν συγκεκριμένες πράξεις με τεράστια ταχύτητα και αποδοτικότητα. Αυτό επιτυγχάνεται χάρη σε εξειδικευμένες αρχιτεκτονικές υλικού (hardware), όπως επεξεργαστές DSP, επιταχυντές υλικού (hardware accelerators) ή ενσωματωμένες μονάδες DSP σε σύγχρονους μικροελεγκτές. Η επιλογή της κατάλληλης πλατφόρμας εξαρτάται από την εκάστοτε εφαρμογή. Για παράδειγμα, μια επαγγελματική

πεταλιέρα κιθάρας απαιτεί έναν ισχυρό επεξεργαστή DSP για επεξεργασία σε πραγματικό χρόνο με εξαιρετικά χαμηλή καθυστέρηση, ενώ ένα έξυπνο ηχείο μπορεί να χρησιμοποιεί έναν πιο οικονομικό μικροελεκτή.

Η πρόκληση στον σχεδιασμό των σύγχρονων ενσωματωμένων συστημάτων ψηφιακής επεξεργασίας σήματος (DSP) έγκειται στην ικανοποίηση δύο φαινομενικά αντικρουόμενων απαιτήσεων, δηλαδή την εκτέλεση πολύπλοκων μαθηματικών πράξεων σε πραγματικό χρόνο με τη χαμηλότερη δυνατή κατανάλωση ενέργειας. Η λύση δεν βασίζεται σε έναν και μόνο ισχυρό επεξεργαστή, αλλά σε μια ετερογενή αρχιτεκτονική (heterogeneous architecture) που συνδυάζει διαφορετικούς τύπους υπολογιστικών μονάδων . Μια βασική τεχνική για τη μείωση της κατανάλωσης ενέργειας, είναι η χρήση παράλληλων αρχιτεκτονικών (parallel architectures) με πολλαπλές μονάδες που λειτουργούν ταυτόχρονα [26]. Αυτό επιτρέπει τη μείωση της τάσης τροφοδοσίας διατηρώντας παράλληλα την ίδια υπολογιστική απόδοση .

Η δύναμη των σύγχρονων ψηφιακών συστημάτων έγκειται στην ευελιξία τους. Μέσω αλγορίθμων, μπορούν να επιτελέσουν μια τεράστια γκάμα λειτουργιών που παλαιότερα απαιτούσαν πολύπλοκα αναλογικά κυκλώματα. Οι πιο χαρακτηριστικές εφαρμογές περιλαμβάνουν το φιλτράρισμα, τη δυναμική επεξεργασία, την προσομοίωση χώρο και την ανάλυση συχνοτήτων. Μέσω αυτών των λειτουργιών, τα σύγχρονα ψηφιακά συστήματα έχουν καταστήσει δυνατή τη δημιουργία εξελιγμένων εφέ και επεξεργασιών που παλαιότερα ήταν πρακτικά αδύνατα ή δυσπρόσιτες για τον μέσο χρήστη.

2.5 Επίλογος

Στο παρόν κεφάλαιο παρουσιάστηκαν οι θεμελιώδεις αρχές που διέπουν τα σύγχρονα ψηφιακά συστήματα ήχου, αναλύθηκαν τρόποι μοντελοποίησης αναλογικών κυκλωμάτων και εξετάστηκαν τα σύγχρονα συστήματα ψηφιακής επεξεργασίας και οι αρχιτεκτονικές τους. Η κατανόηση αυτών των αρχών αποτελεί απαραίτητο υπόβαθρο για τη μελέτη πιο εξειδικευμένων εφαρμογών, όπως τα ψηφιακά εφέ και η επεξεργασία ήχου σε πραγματικό χρόνο, που θα αναλυθούν σε επόμενα κεφάλαια.

Κεφάλαιο 3ο: Εικονικά Αναλογικά Μοντέλα Εφέ

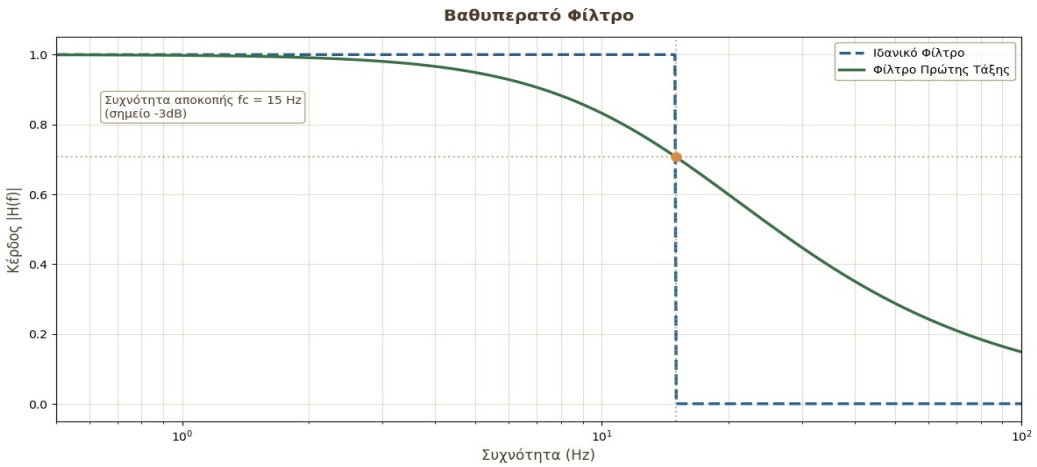
3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλυθούν εις βάθος οι αλγόριθμοι και τεχνικές επεξεργασίας σήματος που καθιστούν ένα μοντέλο εικονικού αναλογικού ακουστικού εφέ. Ένα αναλογικό κύκλωμα μπορεί να μεταφρασθεί ψηφιακά μέσω μαθηματικών συναρτήσεων που περιγράφουν τα ηλεκτρονικά του εξαρτήματα. Θα επεξηγηθούν αναλυτικά τα είδη των εφέ, όπως φίλτρα, παραμορφωτές, γραμμικά και μη γραμμικά μοντέλα.

3.2 Εικονικά Αναλογικά Φίλτρα

3.2.1 Βαθυπερατό Φίλτρο

Ένα ιδανικό βαθυπερατό ή χαμηλοπερατό φίλτρο χρησιμοποιείται για να εμποδίσει μεγάλες συχνότητες, ενώ δεν επηρεάζει χαμηλότερες συχνότητες.



Εικόνα 3.1: Παράδειγμα βαθυπερατού φίλτρου με συχνότητα αποκοπής 15 Hz.

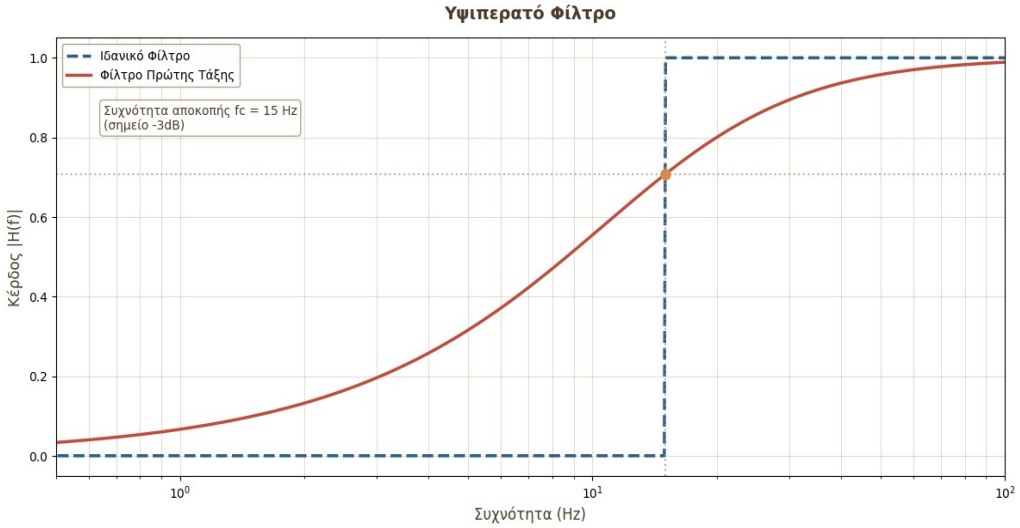
Στον διακριτό χρόνο, η εξίσωση που περιγράφει ένα απλό IIR χαμηλοπερατό φίλτρο, είναι ο εκθετικά σταθμισμένος κινητός μέσος όρος (exponentially weighted moving average) πρώτης τάξης [27]:

$$y(n) = \alpha \cdot x(n) + (1 - \alpha) \cdot y(n-1) \tag{3.1}$$

Όπου το *x(n)* συμβολίζει το πλάτος εισόδου στο φίλτρο σε χρόνο *n*, ενώ το *y(n)* είναι το πλάτος εξόδου σε χρόνο *n*. Στην συνάρτηση προστίθεται και το πλάτους εξόδου του προηγούμενου δείγματος *y(n – 1)*. Ο συντελεστής *α* πρόκειται για μια παράμετρο που ρυθμίζει την ένταση της εξομάλυνσης του φίλτρου. Όσο μικρότερη είναι η τιμή του *α*, τόσο εντονότερο είναι το φιλτράρισμα, καθώς η έξοδος εξαρτάται περισσότερο από προηγούμενες τιμές πως λειτουργούν ως feedback και συνεπώς αποκόπτονται οι υψηλές συχνότητες. Αντίθετα, όσο το *α* πλησιάζει τη μονάδα, τόσο το φίλτρο επηρεάζει λιγότερο το σήμα, επιτρέποντας τη διέλευση και των υψηλών συχνοτήτων.

3.2.2 Υψιπερατό φίλτρο

Ένα ιδανικό υψιπερατό φίλτρο τελεί την ακριβώς αντίθετη λειτουργία από ένα βαθυπερατό φίλτρο, δηλαδή επιτρέπει τις γρήγορες μεταβολές στο σήμα και εμποδίζει χαμηλές συχνότητες.



Εικόνα 3.2: Παράδειγμα υψιπερατού φίλτρου με συχνότητα αποκοπής 15 Hz

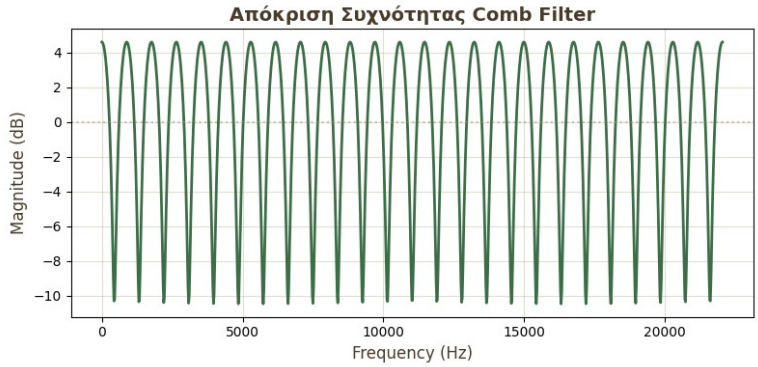
Ο απλούστερος αλγόριθμος που περιγράφει ένα IIR υψιπερατό φίλτρο δίνεται από μία διαφορική εξίσωση πρώτης τάξης [28]:

$$y(n) = x(n) - x(n-1) + \beta \cdot y(n-1) \tag{3.2}$$

Όπου *x(n)* είναι το πλάτος εισόδου τη στιγμή *n* και *y(n)* το πλάτος εξόδου. Το φίλτρο υπολογίζει αρχικά τη διαφορά μεταξύ της τρέχουσας και της προηγούμενης εισόδου *x(n) – x(n-1)*, η οποία ενισχύει τις απότομες μεταβολές του σήματος και εξαλείφει τα σημεία στα οποία το σήμα παραμένει σταθερό. Στη συνέχεια προστίθεται η προηγούμενη έξοδος *y(n-1)* που λειτουργεί ως feedback πολλαπλασιασμένη με τον συντελεστή *β* ($0 < \beta < 1$) και καθορίζει τον ρυθμό απόσβεσης της εξόδου. Όσο μεγαλύτερη είναι ητιμή του *β*, τόσο πιο αργά αποσβήνεται η έξοδος, με αποτέλεσμα το φίλτρο να αποκόπτει τις χαμηλές συχνότητες. Έτσι, το φίλτρο επιτρέπει τη διέλευση των υψηλών συχνοτήτων, ενώ ταυτόχρονα μπλοκάρει τις χαμηλές συχνότητες του σήματος.

3.2.3 Φίλτρο Χτένας (Comb Filter)

Ένα φίλτρο χτένας (comb filter) δημιουργείται όταν ένα σήμα προσθέτει στον εαυτό του καθυστερημένες εκδόσεις του αρχικού σήματος με αποτέλεσμα να δημιουργεί παρεμβολές που μπορούν να φανούν μέσω της απόκρισης συχνότητας του συστήματος καθώς δημιουργείται μια σειρά από κορυφές προσδίδοντας το χαρακτηριστικό σχήμα που μοιάζει με δόντια χτένας [29].



Εικόνα 3.3: Υλοποίηση all-pass φίλτρου με σύγκριση σήματος και φάσης εισόδου – εξόδου.

Η μαθηματική βάση του φίλτρου χτένας περιγράφεται από την ακόλουθη εξίσωση διαφορών:

$$y(n) = x(n) + g \cdot y(n - D) \tag{3.3}$$

για ένα φίλτρο χτένας ανάδρασης (feedback comb filter), όπου *D* είναι η καθυστέρηση σε δείγματα, *g* ο συντελεστής ανάδρασης *(0 < g < 1)*, *x(n)* η είσοδος και *y(n)* η έξοδος τη χρονική στιγμή *n* [30]. Μπορεί να εφαρμοστεί ένα πρόσθετο φίλτρο απόσβεσης (damping). Η βασική λειτουργία περιγράφεται από την εξίσωση:

$$y(n) = x(n) + g \cdot s(n-1) \tag{3.4}$$

όπου y*(n)* είναι η τιμή της εξόδου μετά την εφαρμογή απόσβεσης. Ο όρος $g \cdot s(n-1)$ αποτελεί τον μηχανισμό ανάδρασης, όπου η έξοδος μετά την απόσβεση πολλαπλασιάζεται με τον συντελεστή feedback *g* και προστίθεται στην είσοδο.

Το φίλτρο απόσβεσης εφαρμόζεται στο καθυστερημένο σήμα μέσω ενός χαμηλοπερατού φίλτρου. Όταν ο συντελεστής απόσβεσης είναι κοντά στο 1, οι υψηλές συχνότητες διατηρούνται για μεγαλύτερο χρονικό διάστημα, ενώ όταν είναι μικρός, αποσβένονται γρήγορα, δημιουργώντας ένα πιο "σκοτεινό" ηχητικό αποτέλεσμα.

Η απόσταση μεταξύ των κορυφών να καθορίζεται από τον χρόνο καθυστέρησης:

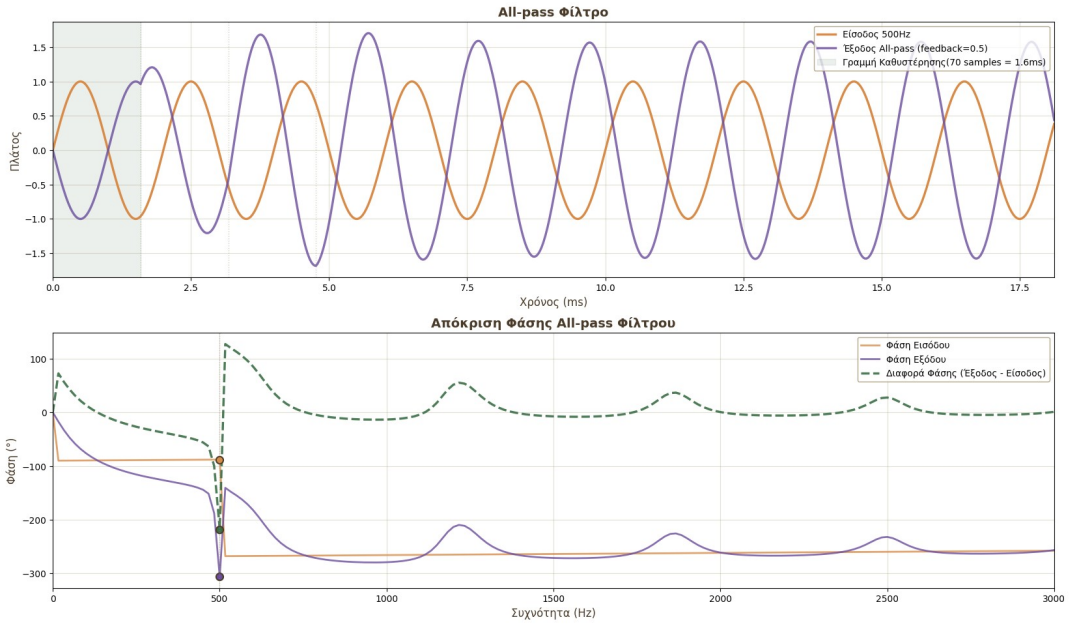
$$\Delta f = \frac{f_s}{D} \tag{3.5}$$

όπου *fs* είναι η συχνότητα δειγματοληψίας και *D* η καθυστέρηση σε δείγματα. Η εφαρμογή αυτή αποτελεί θεμελιώδες δομικό στοιχείο για την κατασκευή αντηχήσεων Schroeder, όπου πολλαπλά φίλτρα χτένας συνδέονται παράλληλα ή σε σειρά για τη δημιουργία πυκνών, φυσικών αντηχήσεων [31].

3.2.4 Φίλτρο All-Pass

Όταν ένα φίλτρο επιτρέπει την διέλευση όλων των συχνοτήτων με ίσο κέρδος για όλες τις συχνότητες, τότε αυτό ονομάζεται all-pass φίλτρο. Στην πράξη, το all-pass φίλτρο δεν χρησιμοποιείται για να

αποδώσει έμφαση ή το αντίθετο στο σήμα, όπως στα περισσότερα φίλτρα, αλλά για την μεταβολή της φάσης συναρτήσει της συχνότητας του [32].



Εικόνα 3.4: Υλοποίηση all-pass φίλτρου με σύγκριση σήματος και φάσης εισόδου – εξόδου.

Για την εφαρμογή ενός all-pass φίλτρου, μπορεί να χρησιμοποιηθεί η συνάρτηση μεταφοράς [33]:

$$H(z) = \frac{z^{-M} - g}{1 - g \cdot z^{-M}} \tag{3.6}$$

όπου M το μήκος της γραμμής καθυστέρησης και g ο συντελεστής ανάδρασης, με το z να αντιπροσωπεύει την καθυστέρηση M δειγμάτων. Η διαφορική εξίσωση εξίσωση του φίλτρου εκφράζεται από το σύστημα [34]:

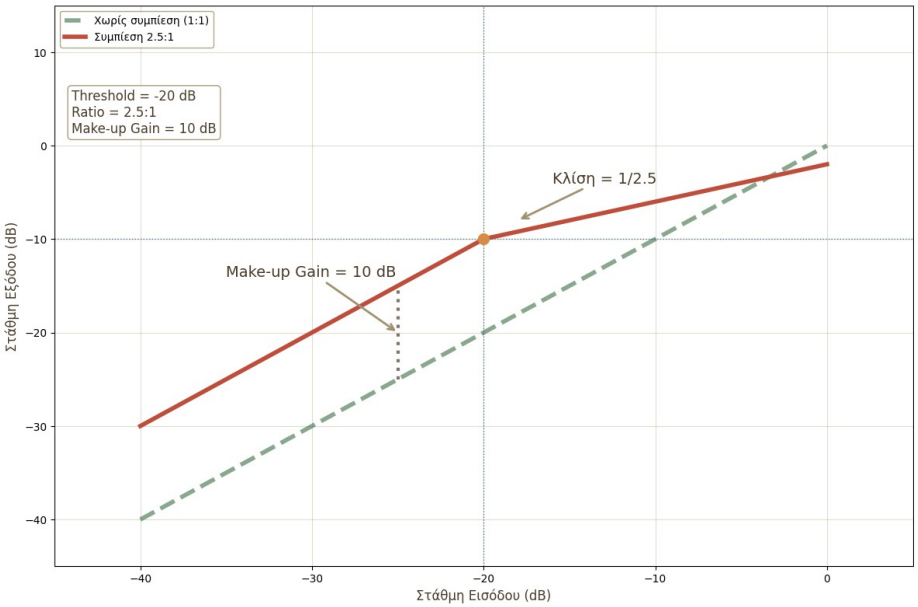
$$y(n) = -g \cdot x(n) + x(n-M) + g \cdot y(n-M) \tag{3.7}$$

όπου η καθυστερημένη έξοδος $y(n-M)$ αποθηκεύεται και προστίθεται με την τρέχουσα είσοδο για να υπολογισθεί η τρέχουσα έξοδος. Με αυτόν τον τρόπο, επιτρέπεται μια αποδοτική υλοποίηση ενός all-pass φίλτρου, με ελάχιστες μαθηματικές πράξεις .

3.3 Εικονικά Αναλογικά Μοντέλα Μη Γραμμικής Επεξεργασίας

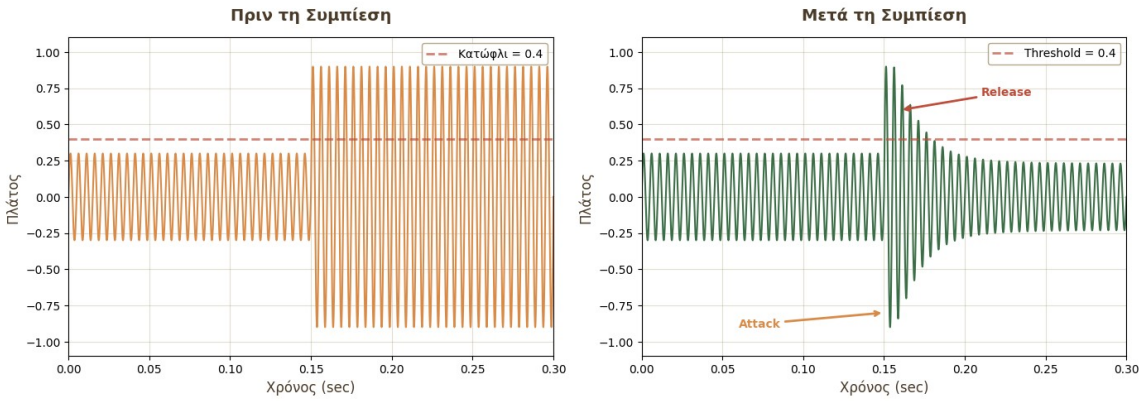
3.3.1 Συμπιεστής Δυναμικού Εύρους (Compression)

Ο συμπιεστής δυναμικού εύρους ή αλλιώς κομπρέσσορας (compressor), χρησιμοποιείται για τη μείωση της διαφοράς μεταξύ μικρότερης και μεγαλύτερης βαθμίδας ενός σήματος. Συμπιέζει το δυναμικό φάσμα του σήματος και ως εκ τούτου, το σήμα μπορεί να ενισχυθεί, χωρίς να υποστεί παραμόρφωση (overshoot), εφόσον δεν θα ξεπεράσει το επιτρεπτό όριο του εύρους πλάτους [35].



Εικόνα 3.5: Συμπίεση δυναμικού φάσματος με κατώφλι συμπίεσης στα -20dB, make-up gain στα 10dB και λόγος μεταβολής κλίσης συμπίεσης στα 2.5:1.

Στην πράξη, υπάρχουν συγκεκριμένες παράμετροι που καθορίζουν την ένταση και τον ρυθμό λειτουργίας ενός κομπρέσσορα. Αρχικά, το κατώφλι (threshold) συμπίεσης περιγράφει το όριο στο οποίο ξεκινάει να εφαρμόζεται η συμπίεση στο σήμα από ένα σημείο και μετά. Το κέρδος αντιστάθμισης (make-up gain) ή ενίσχυση εξόδου ορίζει το κέρδος με το οποίο θα ενισχυθεί το σήμα αφού πρώτα συμπιεστεί. Ο λόγος μεταβολής (ratio) καθορίζει το επίπεδο της συμπίεσης μετά το κατώφλι. Για παράδειγμα, η συμπίεση με λόγο μεταβολής 2.5:1 σημαίνει πως από τα 2.5dB εισόδου που θα ξεπεράσουν το κατώφλι που έχει οριστεί, μόνο το 1dB θα επιτραπεί να περάσει στην έξοδο από τον κομπρέσσορα. Ο χρόνος εκκίνησης (attack) και ο χρόνος απελευθέρωσης (release) αποτελούν σταθερές χρόνου που επηρεάζουν την ταχύτητα αντίδρασης του κομπρέσσορα πριν και μετά την συμπίεση. Κατά την αρχή της συμπίεσης όπου συμβάλει ο λόγος μεταβολής, υπάρχει μια επιπλέον επιλογή που προσδιορίζει πόσο ομαλή θα είναι η μετάβαση από το καθαρό σήμα εισόδου, στο συμπιεσμένο σήμα εξόδου. Ονομάζεται πλάτος καμπής ή πλάτος γονάτου (knee width) καθώς η αναλογία σήματος εισόδου και σήματος εξόδου, μετά το κατώφλι, σχηματίζει μια καμπή λόγω της κλίσης που εφαρμόζει ο λόγος μεταβολής. Η μετάβαση αυτή μπορεί να χαρακτηρισθεί είτε ως μια απότομη και αιχμηρή γωνία (hard knee) , είτε ως μια ομαλή καμπύλη (soft knee) [36].



Εικόνα 3.6: Συμπίεση σήματος με χρήση κατωφλιού.

Για να υλοποιηθεί ένα εικονικό αναλογικό μοντέλο ενός κομπρέσσορα, ο αλγόριθμος θα πρέπει αρχικά να ανιχνεύσει την στάθμη του σήματος εισόδου και αν υπερβαίνει το κατώφλι που έχει ορισθεί να εξομαλύνει το σήμα εξόδου, με αύξηση ή μείωση της έντασης του. Ένας αλγόριθμος που μπορεί να χρησιμοποιηθεί για την ανίχνευση έντασης του σήματος εισόδου, είναι η ανίχνευση κορυφής.

$$y_1(n) = \begin{cases} \alpha_A \cdot y[n-1] + (1-\alpha_A) \cdot x[n], & x[n] > y[n-1] \\ \alpha_R \cdot y[n-1] + (1-\alpha_R) \cdot x[n], & x[n] \leq y[n-1] \end{cases} \tag{3.8}$$

Η εξίσωση περιγράφει ένα IIR φίλτρο πρώτης τάξης που εναλλάσσει τους συντελεστές attack και release αναλόγως με την φάση λειτουργίας, δηλαδή εάν η ένταση βρίσκεται πάνω ή κάτω από το κατώφλι. Για την εφαρμογή λοιπόν του κομπρέσσορα στο σήμα, μπορεί να χρησιμοποιηθεί η παρακάτω εξίσωση για την υλοποίηση soft knee:

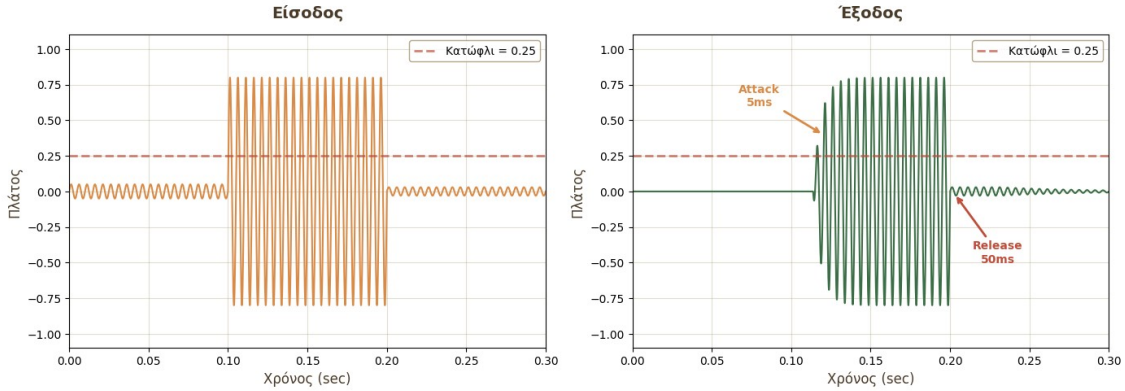
$$y(n) = x(n) + \frac{\left(\left(\frac{1}{R} - 1\right) \cdot x(n) - T + \frac{W}{2}\right)^2}{2 \cdot W} \tag{3.9}$$

Αντί να εφαρμοστεί απότομα η συμπίεση μόλις το σήμα ξεπεράσει το κατώφλι *T*, δημιουργείται μια ομαλή καμπύλη μετάβασης μέσα σε μια περιοχή πλάτους *W* μετά του κατωφλίου, και συγκεκριμένα όταν η απόλυτη διαφορά μεταξύ εισόδου *x(n)* και *T* είναι μικρότερη ή ίση του μισού *W*, τότε η έξοδος *y(n)* δεν είναι ούτε η γραμμική περιοχή χωρίς συμπίεση ούτε η πλήρης συμπίεση με λόγο *R*, αλλά μια παραβολική καμπύλη που ξεκινάει από την γραμμική περιοχή και καταλήγει ομαλά στην περιοχή πλήρους συμπίεσης, έτσι ώστε η κλίση της καμπύλης να μεταβάλλεται σταδιακά και να αποφεύγεται απότομη μεταβολή στο σημείο του κατωφλίου [37].

3.3.2 Καταστολέας Θορύβου (Noise Gate)

Το φίλτρο καταστολής θορύβου ή πύλη θορύβου (noise gate) αποτελεί ένα βασικό ακουστικό εφέ. Όπως περιγράφεται από το όνομα, χρησιμοποιείται για την αποκοπή θορύβου κάτω από ένα όριο, έτσι ώστε

η “πύλη” να ανοίγει όταν το σήμα έχει φτάσει ένα επίπεδο πάνω από αυτό του κατωφλιού και αποκόβει πλήρως το υπόλοιπο σήμα [38].



Εικόνα 3.7: Αφαίρεση θορύβου από σήμα με χρήση κατωφλιού, attack και release σταθερών.

Ομοίως με την υλοποίηση του κομπρέσσορα, η πύλη θορύβου μπορεί να εφαρμόσει τις σταθερές attack και release για να μεταβάλλει την ταχύτητα αντίδρασης και το threshold για την αποκοπή χαμηλότερων σταθμών σήματος. Ο χρόνος συγκράτησης (hold) ορίζει την σταθερά χρόνου για την οποία η πύλη θα μείνει ανοιχτή αφού η στάθμη του σήματος πέσει κάτω από το κατώφλι, εξομαλύνοντας το σήμα στην περίπτωση συνεχών ραγδαίων μεταβολών. Η επεξεργασία του ψηφιακού σήματος για την εφαρμογή του noise gate ξεκινάει με ένα IIR φίλτρο δεύτερης τάξης:

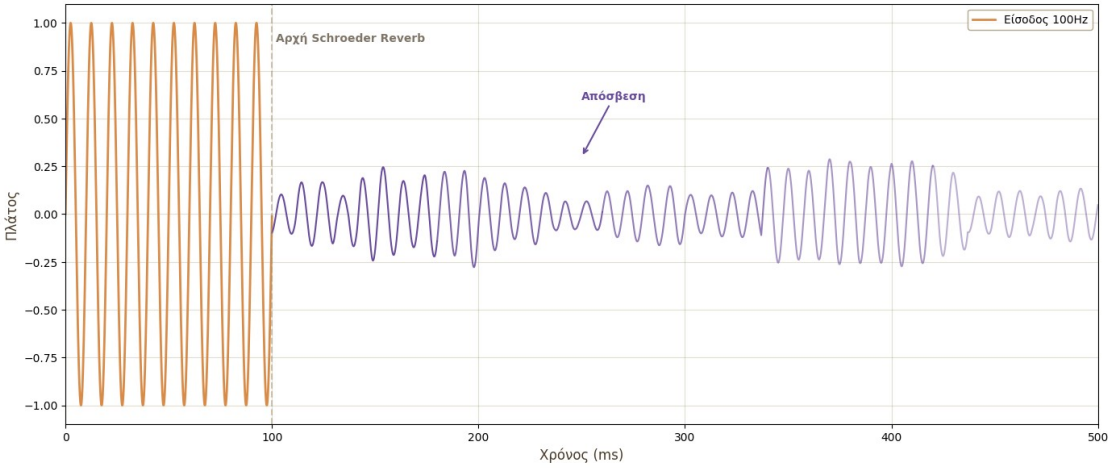
$$y_2(n) = (1-a)^2 \cdot |x(n)| + 2a \cdot y_1(n) - a^2 \cdot y(n-1) \quad (3.10)$$

όπου $|x(n)|$ είναι η απόλυτη τιμή του τρέχοντος δείγματος του εισερχόμενου σήματος για ανίχνευση του επιπέδου, $y_2(n)$ η τρέχουσα εξομαλυμένη έξοδος, $y_1(n)$ η ενδιάμεση εσωτερική κατάσταση και $y(n-1)$ η προηγούμενη έξοδος του φίλτρου, ενώ a είναι η σταθερά πόλου του φίλτρου που υπολογίζεται ως $a = exp(-1 / (1ms * Fs))$ με Fs τον ρυθμό δειγματοληψίας σε Hz. Το αποτέλεσμα συγκρίνεται με το threshold όπου αν το υπερβαίνει τότε η πύλη ανοίγει και το gain αυξάνεται γραμμικά με ρυθμό που καθορίζεται από την παράμετρο attack, διαφορετικά ελέγχεται αν πέρασε ο χρόνος hold με το gain να μειώνεται γραμμικά σύμφωνα με την παράμετρο release [38].

3.3.3 Προσομοιωτής Αντήχησης (Reverb)

Το εφέ αντήχησης, ή reverb, είναι το αποτέλεσμα των πολλαπλών ανακλάσεων του ήχου μέσα σε ένα κλειστό χώρο, το οποίο δημιουργεί την αίσθηση της ακουστικής παρουσίας και του βάθους. Κατά την ψηφιακή επεξεργασία με την χρήση reverb, το σήμα εισόδου χωρίζεται σε μια σειρά εσωτερικών καθυστερήσεων και φίλτρων που προσομοιώνουν πολλαπλές αντανακλάσεις σε έναν χώρο, όπου το μέγεθος του χώρου καθορίζει τις καθυστερήσεις και την πυκνότητα των αντανακλάσεων. Οι αντανακλάσεις αυτές μειώνονται σταδιακά, ή αλλιώς αποσβένονται, με αποτέλεσμα την δημιουργία της χαρακτηριστικής “ουράς” απόσβεσης. Με βάση τα πρότυπα ISO 3382-1, ISO 3382-2 και ASTM E2235, ο χρόνος αντήχησης (Reverberation Time - RT60) ορίζεται ως ο χρόνος που απαιτείται για να μειωθεί η στάθμη του ήχου κατά 60 dB, μετά τη διακοπή της πηγής ήχου. Η τιμή του RT60 δεν είναι σταθερή

σε όλον τον χώρο, αλλά μεταβάλλεται ανάλογα με τη θέση μέτρησης στον χώρο, επομένως για την ακριβή περιγραφή ενός δωματίου λαμβάνεται συνήθως μια μέση τιμή. Χώροι με RT60 μικρότερο από 0,3 δευτερόλεπτα χαρακτηρίζονται ακουστικά "νεκροί", ενώ μικρότεροι χώροι που παρουσιάζουν χρόνο αντήχησης μεγαλύτερο από 2 δευτερόλεπτα θεωρούνται έντονα αντηχητικοί [39].



Εικόνα 3.8: Παράδειγμα reverberation με ουρά απόσβεσης και αλλαγή φάσης.

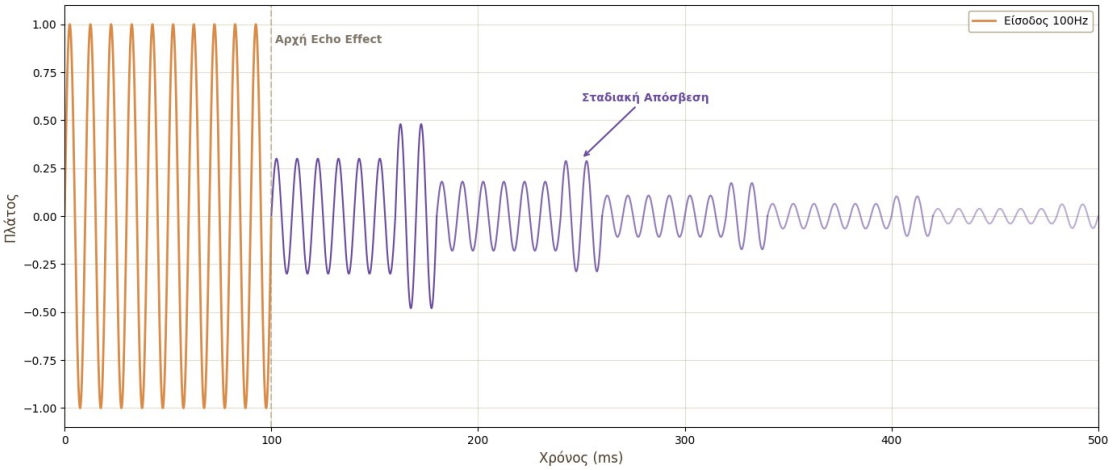
Ένας γνωστός αλγόριθμος για τεχνητό reverberation περιγράφηκε από τον Manfred R. Schroeder το 1962 και έως σήμερα αποτελεί θεμελιώδη αρχή για την σχεδίαση ενός ψηφιακού reverb. Η μέθοδος αυτή ονομάζεται τοπολογία Schroeder [40]. Χρησιμοποιεί πολλαπλά παράλληλα comb φίλτρα με διαφορετικούς χρόνους καθυστέρησης στο καθένα. Σε κάθε ένα από αυτά τα comb φίλτρα εφαρμόζεται ανάδραση, που ελέγχεται από την παράμετρο room size. Στη συνέχεια, το σήμα διέρχεται από πολλαπλά σειριακά all-pass φίλτρα, τα οποία διατηρούν ίση απόκριση προς όλες τις συχνότητες αλλά ενισχύουν την πυκνότητα της ηχώ στον τεχνητό χώρο[. Συχνά, οι χρόνοι καθυστέρησης για τα comb και all-pass φίλτρα προτιμούνται να είναι πρώτοι αριθμοί [41].

Υπάρχουν 3 βασικές παράμετροι που ελέγχουν ένα reverb εφέ. Αρχικά, το μέγεθος δωματίου (room size) καθορίζει τον χρόνο αντήχησης, όσο μικρότερο τόσο πιο γρήγορα εξασθενεί η ηχώ. Η παράμετρος της απόσβεσης (damping) λειτουργεί ως ένα υψηπερατό φίλτρο κατά την διάρκεια της αντήχησης, εάν πάρει μικρή τιμή τότε υπάρχει ελάχιστη απορρόφηση των υψηλών συχνοτήτων. Τέλος, η παράμετρος μίξης (mix) εναλλάσσει την ισορροπία μεταξύ του αρχικού σήματος εισόδου (dry) με το επεξεργασμένο σήμα εξόδου (wet), παίρνοντας όλες τις ενδιάμεσες τιμές ανάμεσα στο dry και στο wet σήμα [42].

3.3.4 Προσομοιωτής Ηχώς (Echo)

Στον φυσικό κόσμο, η έννοια της ηχώ (echo) χαρακτηρίζεται ως ένα φυσικό φαινόμενο που δημιουργείται από την ανάκλαση ηχητικών κυμάτων πάνω σε επιφάνειες [43]. Η διαφορά μεταξύ ηχώ και αντήχησης κρύβεται στο μέγεθος του χώρου στον οποίο δημιουργούνται τα δύο φαινόμενα. Συγκεκριμένα, η ηχώ εμφανίζεται όταν τα ηχητικά κύματα ανακλώνται σε απομακρυσμένες επιφάνειες, συνήθως σε απόσταση μεγαλύτερη από 17 μέτρα, επιστρέφοντας με αισθητή καθυστέρηση, δημιουργώντας διακριτές και επαναλαμβανόμενες αντανακλάσεις. Αντίθετα, η αντήχηση είναι ένα πιο σύνθετο φαινόμενο που συμβαίνει όταν τα κύματα ανακλώνται ταυτόχρονα από πολλαπλές επιφάνειες μέσα σε έναν κλειστό χώρο, με αποτέλεσμα οι επαναλήψεις να μην είναι τόσο αισθητές. Για

παράδειγμα, η ηχώ μπορεί να προκληθεί μέσα σε ένα μεγάλο γήπεδο, ενώ αντήχηση μέσα σε μια εκκλησία.



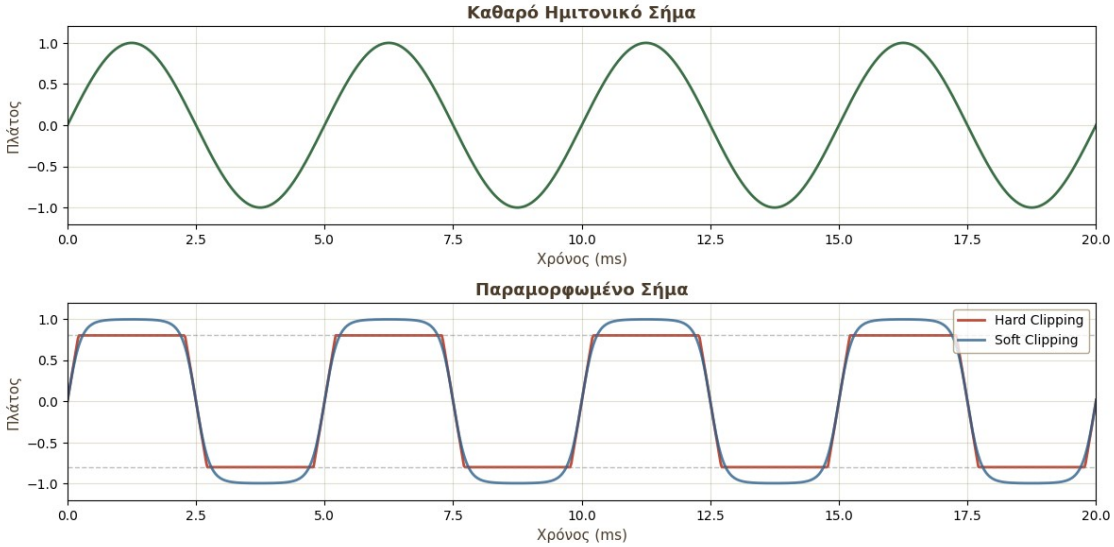
Εικόνα 3.9: Παράδειγμα ηχώ με σταδιακή ουρά απόσβεσης.

τον φυσικό κόσμο, η έννοια της ηχώ (echo) χαρακτηρίζεται ως ένα φυσικό φαινόμενο που δημιουργείται από την ανάκλαση ηχητικών κυμάτων πάνω σε επιφάνειες [44]. Η διαφορά μεταξύ ηχώ και αντήχησης κρύβεται στο μέγεθος του χώρου στον οποίο δημιουργούνται τα δύο φαινόμενα.

Για την ρύθμιση του εφέ echo αρχικά χρησιμοποιείται η παράμετρος του χρόνου καθυστέρησης (delay time) που ορίζει την χρονική απόσταση από την οποία θα ακουστεί το καθυστερημένο σήμα. Έπειτα, η ανάδραση (feedback) καθορίζει την εξασθένηση του σήματος στον χρόνο. Ομοίως με το εφέ αντήχησης, χρησιμοποιούνται οι παράμετροι damping και mix. Για την δημιουργία ενός απλού echo εφέ, χρησιμοποιείται ένα comb φίλτρο, εφαρμόζοντας ένα χαμηλοπερατό φίλτρο στο σήμα και αποσβένοντας σταδιακά την ηχώ.

3.3.5 Παραμορφωτής (Distortion)

Παραμόρφωση (distortion) σε ένα ηχητικό κύμα προκύπτει όταν το επίπεδο το κύματος υπερβαίνει το όριο του μέσου από το οποίο διέρχεται. Οι κορυφές του κύματος αποκόπτονται (clipping) με αποτέλεσμα να παίρνει μια τετραγωνική μορφή. Υπάρχουν δύο ευρέως χρησιμοποιημένες μέθοδοι για την εφαρμογή παραμόρφωσης σε ένα σήμα με την αποκοπή κορυφών. Αρχικά, κατά την ομαλή αποκοπή (soft clipping) υπάρχει σταδιακή μετάβαση στην αποκοπή των κορυφών, ώστε να μην δημιουργούνται αιχμηρές γωνίες στο κύμα με αποτέλεσμα ο ήχος να ακούγεται συμπιεσμένος και “μαλακός”. Από την άλλη, η μετάβαση στην απότομη αποκοπή (hard clipping) είναι πιο ραγδαία καθώς το σήμα φτάνει σε κορεσμό πιο γρήγορα, προσδίδοντας επιθετικότητα και σκληρότητα στον ήχο [45].



Εικόνα 3.10: Παραμόρφωση σήματος με την χρήση soft clipping και hard clipping.

Ένα εφέ παραμόρφωσης χρησιμοποιεί 3 βασικές παραμέτρους. Το κέρδος (gain) χρησιμοποιείται για να ρυθμίσει την ποσότητα παραμόρφωσης πάνω στο σήμα. Η χροιά (tone) δίνει έμφαση σε συγκεκριμένες συχνότητες, κάνοντας τον ήχο πιο φωτεινό ή πιο σκοτεινό. Το επίπεδο εξόδου (level) λειτουργεί ως ρύθμιση στάθμης μετά την παραμόρφωση, αντισταθμίζοντας την απώλεια έντασης που προκαλείται από το κλιπάρισμα του σήματος.

Ο αλγόριθμος που απαρτίζει ένα βασικό εφέ παραμόρφωσης ξεκινάει με την εφαρμογή ζωνοπερατών φίλτρων για την ρύθμιση της χροιάς, καθώς το σήμα διέρχεται αρχικά από ένα υψηπερατό φίλτρο και στην συνέχεια από ένα χαμηλοπερατό φίλτρο. Έπειτα, για την αποκοπή των κορυφών χρησιμοποιείται η μέθοδος soft clipping, που μπορεί να υλοποιηθεί με την αξιοποίηση της εξίσωσης υπερβολικής εφαπτομένης, καθώς παρουσιάζει γραμμική συμπεριφορά για σήματα χαμηλής στάθμης ενώ ακολουθεί σταδιακό κορεσμό για μεγάλα σήματα:

$$\tanh a = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} \tag{3.11}$$

Για την επέκταση αυτού του μοντέλου μπορεί επιπροσθέτως να ενσωματωθεί ασύμμετρη παραμόρφωση που δημιουργεί διαφορετική συμπεριφορά για τα διαφορετικά επίπεδα του σήματος:

$$y_1(n) = \tanh(\kappa \cdot (x(n) + x_0) - \tanh(\kappa \cdot x_0) \tag{3.12}$$

Ο αλγόριθμος αυτός περιγράφηκε από τους Schuck Jr. και Bodmann, με την παράμετρο κ να ορίζει την ένταση παραμόρφωσης (gain) ενώ η παράμετρος x_0 εισάγει την ασυμμετρία στο σύστημα [46].

3.4 Επίλογος

Συνοψίζοντας, αυτό το κεφάλαιο διερεύνησε τα βασικά εικονικά αναλογικά μοντέλα που απαρτίζουν ένα εφέ ήχου. Είναι προφανές πως η πλήρης κατανόηση αναλογικών κυκλωμάτων για τον σχεδιασμό πιστών εικονικών αναλογικών μοντέλων, αποτελεί μια σύνθετη και περίπλοκη επιστήμη. Η κατανόηση

της λειτουργίας των αλγορίθμων αυτών αποτελεί βασική γνώση για τον σχεδιασμό της αρχιτεκτονικής συστήματος. Τα παραπάνω εφέ απαρτίζονται από απλές μαθηματικές συναρτήσεις οι οποίες επιτρέπουν την ανάπτυξη εφαρμογής για μικροελεγκτές με την μεγαλύτερη δυνατή αποδοτικότητα.

Κεφάλαιο 4ο: Αρχιτεκτονική Ψηφιακού Πολυεφέ Πεταλιέρας

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα επεξηγηθεί η αρχιτεκτονική του συστήματος, χωρίς να αναλυθούν διεξοδικά οι τεχνολογίες, το λογισμικό και οι γνώσεις ηλεκτρονικής που χρησιμοποιήθηκαν για την ανάπτυξή του. Στόχος αυτού του κεφαλαίου είναι μια επιφανειακή προσέγγιση του συστήματος, δηλαδή ανάλυση της λογικής και των απαιτήσεων που καθιστούν την αρχιτεκτονική ανάπτυξης του. Πριν δηλαδή υλοποιηθεί το κύκλωμα του προενισχυτή και το λογισμικό, θα πρέπει να εξεταστούν διάφορες συνιστώσες που συμβάλουν στην επίτευξή συγκεκριμένων στόχων.

4.2 Απαιτήσεις Συστήματος

Το ψηφιακό πολυεφέ πετάλι, όπως εξετάστηκε σε προηγούμενο κεφάλαιο, πρόκειται για μια συσκευή η οποία επιτρέπει την επεξεργασία του σήματος εισόδου και την εφαρμογή πολλαπλών ακουστικών εφέ σε σειριακή σειρά. Ως αποτέλεσμα, το σήμα εξόδου προς τον ενισχυτή είναι το αποτέλεσμα της επεξεργασίας αυτής. Το σύστημα οφείλει να τελεί μια συγκεκριμένη ροή λειτουργιών που καθιστούν δυνατή την χρήση του.

- **Ποιότητα ήχου:** Το σήμα εισόδου θα πρέπει να φιλτράρεται από το κύκλωμα και να δειγματοληπτείται από το ADC και έπειτα το επεξεργασμένο σήμα εξόδου θα πρέπει να μετατρέπεται σε αναλογικό από το DAC και να φιλτράρεται στην βέλτιστη δυνατή ποιότητα. Το ακουστικό φάσμα της ανθρώπινης ακοής πρόκειται για το εύρος από 20 Hz μέχρι 20KHz. Ως αποτέλεσμα, η δειγματοληψία στα 48KHz θα ήταν η κατάλληλη λύση για την καταγραφή του ήχου, σύμφωνα με τον νόμο του Nyquist εφόσον η συχνότητα δειγματοληψίας είναι διπλάσια της συχνότητας του σήματος. Ωστόσο, σκοπός της εφαρμογής δεν είναι απλά η καταγραφή του σήματος, αλλά η επεξεργασία του. Για τα εφέ που το σήμα υφίσταται σημαντική παραμόρφωση όπως distortion ή τα εφέ που εφαρμόζεται πρόσθετο σήμα όπως reverb και echo, θα πρέπει να αυξηθεί σημαντικά η συχνότητα δειγματοληψίας ώστε να μπορέσει να επιτευχθεί ακόμα καλύτερη ανάλυση ήχου. Για αυτόν τον λόγο, έχει επιλεχθεί η συχνότητα δειγματοληψίας στα 96KHz. Ο τρόπος με τον οποίο επιτυγχάνεται αυτή η δειγματοληψία με όσο το δυνατόν καλύτερη αποδοτικότητα θα αναλυθεί παρακάτω.
- **Ευκολία χρήσης:** Η ανάπτυξη της διεπαφής προϋποθέτει έναν διαισθητικό σχεδιασμό, ο οποίος επιτρέπει στον χρήστη να πλοηγείται εύκολα στο σύστημα, χωρίς να περιπλέκονται οι λειτουργίες του. Αυτό σημαίνει πως τα εξαρτήματα στην φυσική συσκευή θα είναι τοποθετημένα με λογική σειρά, οι ρυθμίσεις είναι άμεσα προσβάσιμες χωρίς την χρήση πολλών μενού, ενώ η οθόνη παρέχει σαφή οπτική ανατροφοδότηση με κατάλληλες ενδείξεις και εικόνες.
- **Αποδοτικότητα Συστήματος:** Η επεξεργασία σήματος με την εφαρμογή πολλαπλών εφέ σε συχνότητα δειγματοληψίας 96KHz πρόκειται για μια διαδικασία που απαιτεί ένα δυνατό υπολογιστικό σύστημα, ικανό να τελέσει αυτήν την λειτουργία σε πραγματικό χρόνο, χωρίς κολλήματα και με επαρκούσα διαθέσιμη μνήμη. Το λογισμικό θα πρέπει να αξιοποιεί πλήρως τις δυνατότητες του μικροελεγκτή, όπως την χρήση επιταχυντών, ώστε το σύστημα να είναι αποδοτικό χωρίς να καταναλώνει πολλή ενέργεια.

- **Χαμηλό Κόστος Κατασκευής και Διαθεσιμότητα Υλικών:** Κατά τη διάρκεια του σχεδιασμού και της ανάπτυξης της παρούσας συσκευής, εφαρμόστηκε πιστά η ίδια φιλοσοφία που χρησιμοποιείται συνήθως και στον τομέα της ανάπτυξης εφαρμογών μέσα σε εργασιακά περιβάλλοντα. Συγκεκριμένα, δόθηκε έμφαση στην αξιοποίηση όσο το δυνατόν πιο απλών μεθόδων και τεχνικών, αποφεύγοντας την άσκοπη πολυπλοκότητα, ενώ παράλληλα επιδιώχθηκε με συνέπεια η επίτευξη του χαμηλότερου εφικτού κόστους κατασκευής. Βασική προτεραιότητα αποτέλεσε, επίσης, η χρήση υλικών που είναι εύκολα διαθέσιμα στην αγορά, ώστε να διασφαλίζεται η ευκολία στην παραγωγή.

4.3 Αρχιτεκτονική Συστήματος

Το σύστημα επιτελεί την λειτουργία μιας πολυεφέ πεταλιέρας για ηλεκτρική κιθάρα. Η εφαρμογή αυτή δεν περιορίζεται απλώς στην επεξεργασία του σήματος εισόδου, αλλά ενσωματώνει ένα πλήθος σύνθετων επιμέρους υποσυστημάτων, τα οποία βρίσκονται σε συνεχή αλληλεπίδραση μεταξύ τους. Η αλληλεπίδραση αυτή περιλαμβάνει τόσο ανταλλαγή δεδομένων όσο και τη σειρά και τον τρόπο εφαρμογής των διάφορων εφέ. Ως φυσικό επακόλουθο αυτής της πολυπλοκότητας, η συνολική αρχιτεκτονική του συστήματος μπορεί να διαχωριστεί σαφώς σε δύο θεμελιώδη και λειτουργικά διακριτά μέρη:

1. **Η διεπαφή του χρήστη (user interface):** Το τμήμα εκείνο που αναλαμβάνει την αλληλεπίδραση με τον χρήστη, επιτρέποντάς του να επιλέγει, να τροποποιεί και να οργανώνει τα ηχητικά εφέ σύμφωνα με τις προσωπικές του ανάγκες. Περιλαμβάνει εξαρτήματα όπως κουμπιά, ποτενσιόμετρα, φωτεινές ενδείξεις και οθόνη, εξασφαλίζοντας άμεση ανατροφοδότηση και ευχρηστία.
2. **Η σειριακή επεξεργασία των εφέ (signal processing chain):** Το τμήμα εκείνο που είναι υπεύθυνο για τη διαδοχική εφαρμογή των επιλεγμένων εφέ πάνω στο ηχητικό σήμα. Στο πλαίσιο αυτό, το σήμα της κιθάρας διέρχεται μέσα από μια αλυσίδα εφέ, τοποθετημένων το ένα μετά το άλλο, όπου κάθε στάδιο τροποποιεί την έξοδο του προηγούμενου, δημιουργώντας έτσι το τελικό ηχητικό αποτέλεσμα.

Αυτός ο διαχωρισμός καθιστά το σύστημα ευκολότερο στη σχεδίαση, στη συντήρηση και στη μελλοντική επέκταση, αφού η λογική της αλληλεπίδρασης με τον χρήστη είναι ανεξάρτητη από την ίδια την επεξεργασία του σήματος.

Σχήμα 4.2: Στοίβα αρχιτεκτονικής πολυεφέ πεταλιέρας για κιθάρα.

4.4 Διεπαφή

4.4.1 Ροή Λειτουργίας Διεπαφής

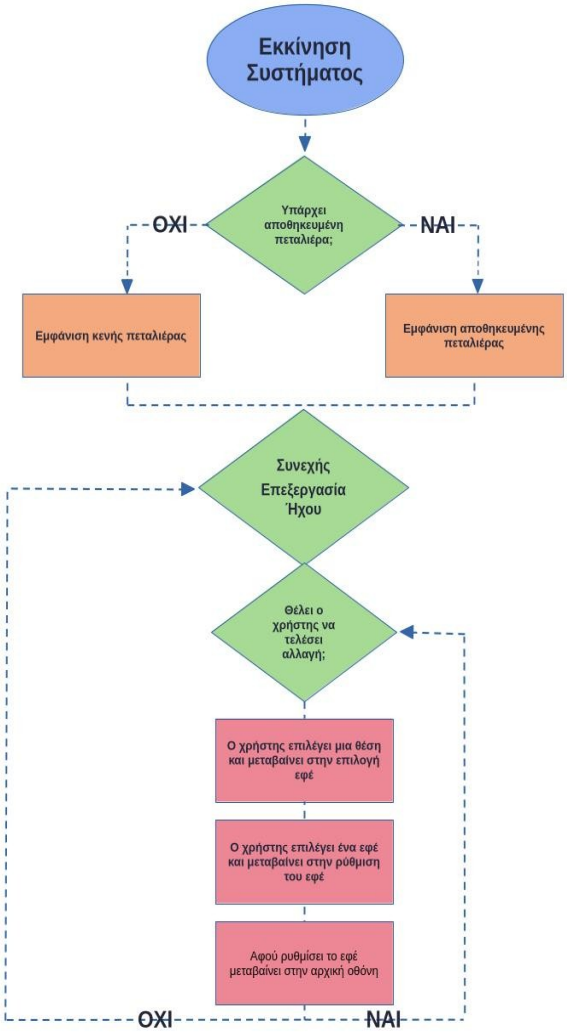
Για την πληρέστερη κατανόηση της λειτουργίας του συστήματος, κρίνεται σκόπιμο να περιγραφεί αναλυτικά η ροή αλληλεπίδρασης μεταξύ χρήστη και διεπαφής. Η ροή αυτή οργανώνεται σε επιμέρους στάδια, τα οποία εκτελούνται διαδοχικά, εξασφαλίζοντας μια συνεκτική και φιλική προς τον χρήστη εμπειρία:

- **Αρχικό Μενού:** Κατά την εκκίνηση του συστήματος, εάν δεν υπάρχει ήδη κάποια πεταλιέρα αποθηκευμένη στη μνήμη, η διεπαφή εμφανίζει αυτόματα μια κενή πεταλιέρα τεσσάρων θέσεων. Σε κάθε θέση μπορεί να τοποθετηθεί ένα ηχητικό εφέ. Ο χρήστης μπορεί να πλοηγηθεί και να επιλέξει μία από τις τέσσερις διαθέσιμες θέσεις, υποδεικνύοντας με τον τρόπο αυτό σε ποιο σημείο της αλυσίδας επιθυμεί να παρέμβει.
- **Επιλογή Εφέ:** Εφόσον μια θέση είναι επιλεγμένη, ο χρήστης πατά το κουμπί για το επόμενο μενού. Με το πάτημα αυτό, το σύστημα μεταβαίνει στη λειτουργία «Επιλογή εφέ», όπου εμφανίζεται ένα μενού διαθέσιμων εφέ. Από το μενού αυτό, ο χρήστης επιλέγει το επιθυμητό εφέ. Το σύστημα αποθηκεύει την επιλογή του και το εφέ τοποθετείται στην προηγουμένως επιλεγμένη θέση.
- **Προσαρμογή ρυθμίσεων του εφέ:** Πατώντας ξανά το ίδιο κουμπί, ο χρήστης εισέρχεται στη λειτουργία «Ρύθμισης παραμέτρων» του συγκεκριμένου εφέ που μόλις επέλεξε. Στο στάδιο αυτό, μπορεί να τροποποιήσει μέσω των ποτενσιόμετρων παραμέτρους όπως ένταση, ταχύτητα, ανάστροφη τροφοδότηση, χρόνο καθυστέρησης κ.ά., ανάλογα με το είδος του εφέ. Το εφέ καθ’ όλη αυτήν την διαδικασία είναι σε πλήρης λειτουργία, με την ανατροφοδότηση σε περίπτωση αλλαγής παραμέτρου σε πραγματικό χρόνο.
- **Επιστροφή στην κύρια οθόνη:** Πατώντας για τρίτη φορά το ίδιο κουμπί, το σύστημα εξέρχεται από τη λειτουργία ρύθμισης και επιστρέφει στην κύρια οθόνη. Στην οθόνη αυτή πλέον εμφανίζεται το επιλεγμένο εφέ στην αντίστοιχη θέση.

Αυτή η σχεδιαστική επιλογή απλοποιεί την αλληλεπίδραση, μειώνει τον αριθμό των απαιτούμενων χειριστηρίων και καθιστά τη λειτουργία της πεταλιέρας ιδιαίτερα διαισθητική, ακόμα και για χρήστες με περιορισμένη εμπειρία.

Τέλος, η διεπαφή περιλαμβάνει επιπλέον κουμπί για την εμφάνιση του μενού ρυθμίσεων. Σε αυτό το μενού, ο χρήστης μπορεί να τελέσει 3 λειτουργίες:

- **Αλλαγή Σειράς Εφέ:** Προφανώς, ο ήχος εξόδου αλλάζει με βάση ποιο εφέ ήχου εφαρμόζεται πρώτα στο σήμα εισόδου. Με το πάτημα της επιλογής αυτής, θα εμφανιστεί η αρχική οθόνη και ο χρήστης θα μπορεί να αλλάξει την σειρά ενός εφέ.
- **Αποθήκευση Πεταλιέρας:** Για να μην χαθεί η πεταλιέρα που έχει δημιουργηθεί από τον χρήστη, ο ίδιος έχει την επιλογή να αποθηκεύσει την πρόοδο του ώστε την επόμενη φορά που θα εκκινηθεί το σύστημα, να εμφανιστεί η πεταλιέρα που αποθήκευσε με όλες τις ρυθμίσεις για το κάθε πετάλι.
- **Φωτεινότητα Οθόνης:** Για την διασφάλιση της βέλτιστης φωτεινότητας οθόνης, με την επιλογή της ρύθμισης αυτής ο χρήστης θα μπορεί κυκλικά να αλλάξει την φωτεινότητα του συστήματος ανά διάφορα επίπεδα.

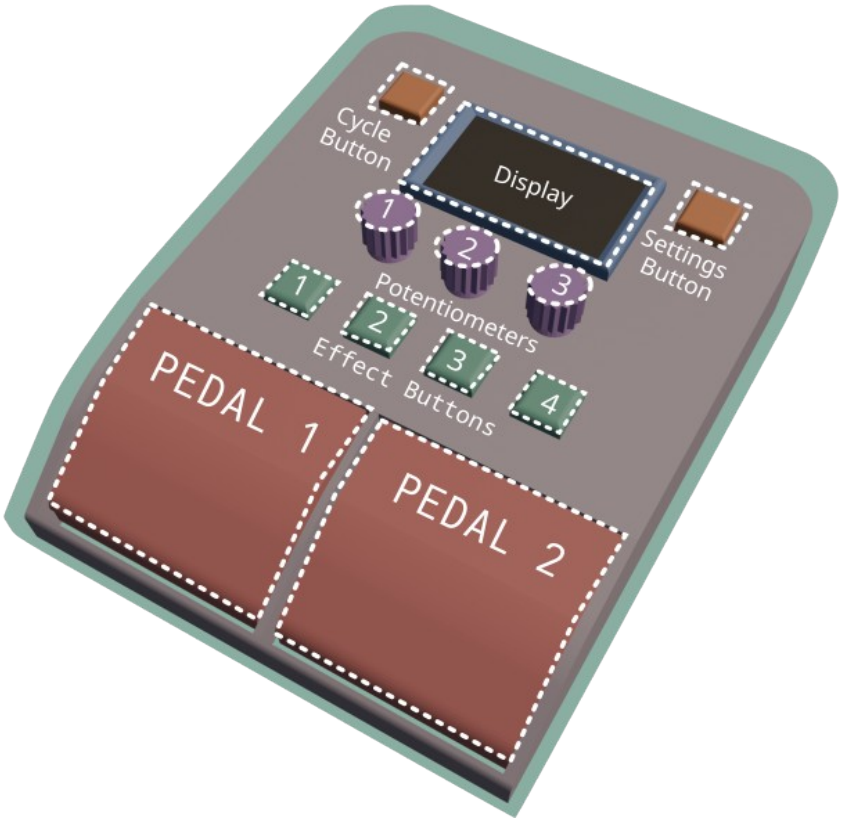


Σχήμα 4.3: Διάγραμμα ροής λειτουργίας

4.4.2 Αρχιτεκτονική Διεπαφής

Σύμφωνα με την ροή λειτουργίας της διεπαφής, η αρχιτεκτονική της θα είναι σχεδιασμένη έτσι ώστε να εξυπηρετεί όλες τις παραπάνω λειτουργίες και να παρέχει ευκολία αλληλεπίδρασης με τον χρήστη.

Η διεπαφή αρχικά περιέχει την οθόνη η οποία είναι τοποθετημένη στο κέντρο. Δεξιά της βρίσκεται το κουμπί για το μενού “Ρυθμίσεις” ενώ στα αριστερά της βρίσκεται το κουμπί που κυκλικά μεταβαίνει στα μενού των εφέ. Κάτω από την οθόνη είναι τοποθετημένα 3 ποτενσιόμετρα τα οποία αλλάζουν τις παραμέτρους των εφέ. Επειδή οι παράμετροι του κάθε εφέ μπορεί να είναι παραπάνω από 3, χρησιμοποιούνται το αριστερό και το δεξί πετάλι αναλόγως για να μεταβεί στην επόμενη σελίδα παραμέτρων των πεταλιών. Πάνω από τα 2 πετάλια, υπάρχουν 4 κουμπιά που είναι αρμόδια για την επιλογή και ενεργοποίηση ή απενεργοποίηση των εφέ και την επιλογή των ρυθμίσεων.



Εικόνα 4.1: Τρισδιάστατη αναπαράσταση της αρχιτεκτονικής της πεταλιέρας.

4.5 Επίλογος

Η αρχιτεκτονική του συστήματος διατηρεί μια απλότητα, χωρίς να καθιστά πολύπλοκη την χρήση την χρήση της διεπαφής. Καλύπτονται όλες οι βασικές λειτουργίες με την χρήση μιας λογικής σειράς κινήσεων για την επιλογή και παραμετροποίηση ενός εφέ.

Κεφάλαιο 5ο: Σχεδιασμός και Κατασκευή Ηλεκτρονικού Κυκλώματος

5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναλυθεί αρχικά εις βάθος ο σχεδιασμός, του κυκλώματος εισόδου – εξόδου. Θα συμπεριληφθούν οι προσομοιώσεις που διεξήχθησαν στο πρόγραμμα προσομοίωσης κυκλωμάτων LtSpice. Επιπλέον, θα περιγραφεί το κύκλωμα διεπαφής του χρήστη που απαρτίζεται από κουμπιά, ποτενσιόμετρα και οθόνη. Έπειτα θα γίνει επεξήγηση της ανάπτυξης και υλοποίησης του Printed Circuit Board (PCB) και θα δοθούν πραγματικές μετρήσεις του κυκλώματος. Τέλος θα δοθεί ο πίνακας με τα ονόματα, τις τιμές κατασκευής και υλικών που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος.

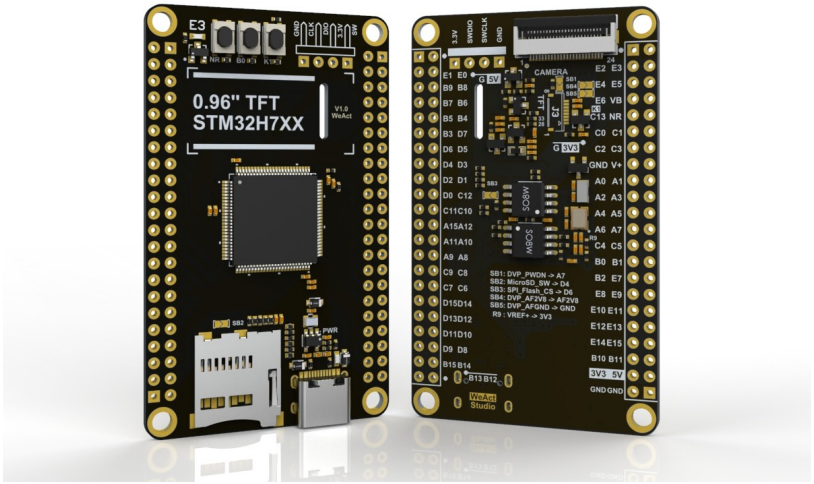
5.2 Τεχνικές Προδιαγραφές Κυκλώματος

Η ανάπτυξη του κυκλώματος προϋποθέτει προσεκτική ανάλυση και έρευνα των προδιαγραφών, ώστε να αποφευχθούν λάθη σχεδιασμού και να εξασφαλισθεί η ποιότητα κατασκευής.

- **Τάση Λειτουργίας Κυκλώματος:** Συνήθως για μουσικές εφαρμογές και συστήματα, χρησιμοποιείται διπλή τροφοδοσία. Στην συγκεκριμένη περίπτωση για λόγους απλότητας επιλέχθηκε η τροφοδοσία +5V από τον μικροελεγκτή, καθώς τα ρεύματα του κυκλώματος είναι χαμηλά. Για να μπορέσουν τα +5V να καλύψουν τις ανάγκες του κυκλώματος, θα χρειαστεί η χρήση ενός DC – DC μετατροπέα που να παράγει -5V με αποτέλεσμα η τελική τροφοδοσία να φτάσει τα +-5V (10V). Η τροφοδοσία θα γίνεται μέσω μπαταριών.
- **Ποιότητα Ήχου:** Για την διασφάλιση της ποιότητας του ήχου, χωρίς θόρυβο ή παρεμβολές, είναι αναγκαίο το κύκλωμα του PCB να είναι σχεδιασμένο με τέτοιον τρόπο έτσι ώστε να αποφεύγεται το παραπάνω πρόβλημα. Οι γραμμές ήχου θα πρέπει να διατρέχουν όσο το δυνατόν μικρότερη διαδρομή, χωρίς διελεύσεις σε άλλα επίπεδα και με το κατάλληλο φιλτράρισμα.
- **Βελτιστοποίηση Σχεδίασης και Κόστους:** Η σχεδίαση θα εξασφαλίζει την καλύτερη τοποθέτηση των εξαρτημάτων και γραμμών, βελτιστοποιώντας τόσο το μέγεθος της πλακέτας όσο και το κόστος κατασκευής.

5.3 Επιλογή Μικροελεγκτή

Ο μικροελεγκτής αποτελεί την βασική επεξεργαστική μονάδα του συστήματος που παρέχει, όχι μόνο υπολογιστική ισχύ, αλλά και όλα τα ηλεκτρονικά υλικά που συμβάλουν στην λειτουργία του κυκλώματος, όπως Analog to Digital Converter (ADC) και Digital to Analog Converter (DAC). Ο μικροελεγκτής θα πρέπει να είναι ικανός να διαχειρίζεται τα δεδομένα με την βέλτιστη ταχύτητα και εφαρμογή αλγορίθμων DSP σε πραγματικό χρόνο. Έτσι επιλέχθηκε η χρήση του STM32H743VIT6, με πλακέτα της WeActStudio η οποία παρέχει όλες τις δυνατότητες του μοντέλου STM32, με την χρήση ελαχίστου μεγέθους για την πλακέτα.



Εικόνα 5.1: Τρισδιάστατη απεικόνιση της STM32H7xx πλακέτας από την WeActStudio [47].

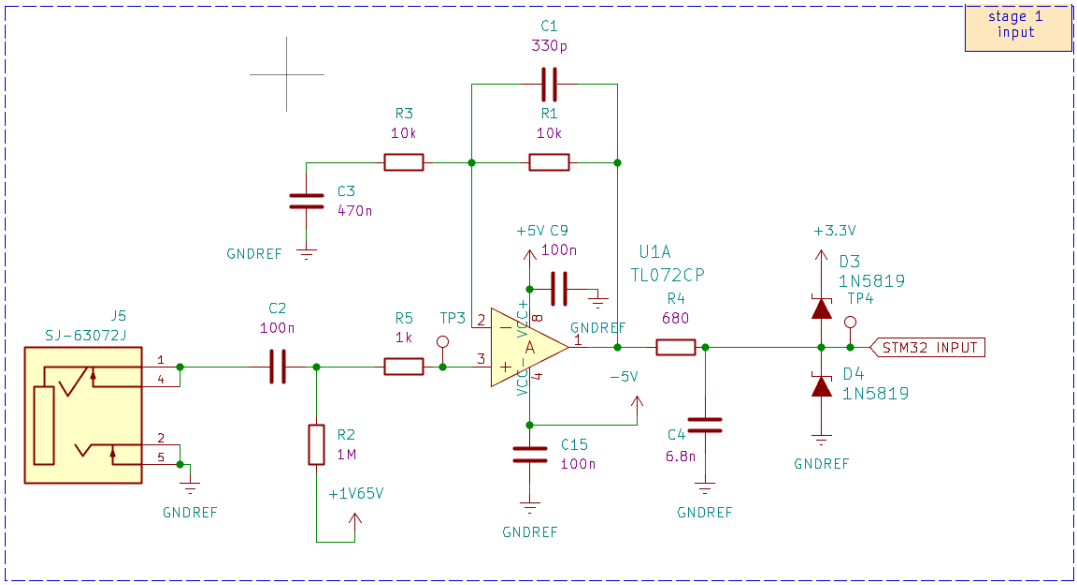
- **Επεξεργαστική Ισχύς:** Ο STM32H7 μικροελεγκτής χρησιμοποιεί τον μονοπύρηνο επεξεργαστή Cortex M7 αρχιτεκτονικής ARM. Ο επεξεργαστής αυτός είναι άρτια ικανός να τελέσει εφαρμογές DSP σε πραγματικό χρόνο, ειδικότερα μπορεί να ανταπεξέλθει στην σειριακή επεξεργασία πολλαπλών εφέ με γρήγορη ταχύτητα. Η μέγιστη συχνότητα του επεξεργαστή φτάνει στα 480MHz και είναι 32bit. Η υψηλή αυτή επίδοση οφείλεται στη “superscalar” διασωλήνωση 6 σταδίων και με την χρήση προσωρινών μνημών για τις εντολές και δεδομένα [48]. Το σημαντικό προτέρημα της ARM αρχιτεκτονικής είναι πως η υψηλή του απόδοση επιτυγχάνεται με χαμηλή κατανάλωση ενέργειας.
- **Μνήμη:** Το συγκεκριμένο μοντέλο περιλαμβάνει 2048KB ROM Flash, δηλαδή μνήμη που αξιοποιείται για την φόρτωση του λογισμικού άλλα και την αποθήκευση των δεδομένων της πεταλιέρας, καθώς και 1024MB RAM για την προσωρινή αποθήκευση των σημάτων, μεταβλητών και άλλων δεδομένων. Η συγκεκριμένη πλακέτα διαθέτει επίσης δύο εξωτερικές ROM μνήμες, μια 8MB SPI Flash και μια 8MB QSPI Flash.
- **Περιφερειακά:** Παρότι το μικρό του μέγεθος, ο STM32H7 υπερκαλύπτει τις ανάγκες για περιφερειακά καθώς κατέχει 3 16bit ADCs, 2 12bit DACs, 3 SPI διεπαφές, 4 DMA και συνολικά 32 διαθέσιμους General Input/Output (GPIO) ακροδέκτες.

5.4 Σχεδιασμός Κυκλώματος Εισόδου - Εξόδου

Κατά την σχεδίαση του κυκλώματος εισόδου – εξόδου, χρησιμοποιήθηκε το πρόγραμμα kiCAD για την σχεδίαση του σχηματικού διαγράμματος. Το κύκλωμα εισόδου είναι υπεύθυνο για την πόλωση, ενίσχυση και φιλτράρισμα του σήματος. Το κύκλωμα εξόδου λειτουργεί σαν ένα απλό φίλτρο και ακόλουθο τάσης χωρίς ενίσχυση στο σήμα, για την διέλευση του στον ενισχυτή. Επίσης χρησιμοποιούνται επιπλέον DC - DC μετατροπείς για την τροφοδοσία του τελεστικού ενισχυτή.

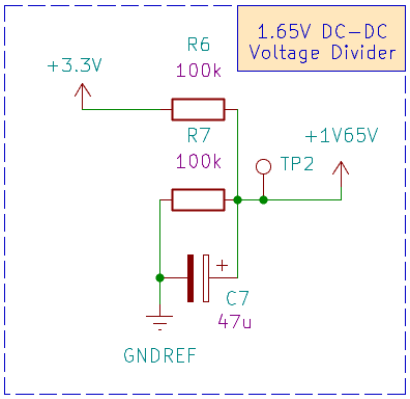
5.4.1 Προσαρμογή Σήματος Φάσης Εισόδου

Στόχος του κυκλώματος εισόδου είναι να πολώνει και να φιλτράρει το σήμα από μη επιθυμητές συχνότητες και να ενισχύσει το κέρδος των επιθυμητών, στέλνοντας το τελικά στην είσοδο του ADC του STM32 .



Εικόνα 5.2: Σχηματικό διάγραμμα κυκλώματος εισόδου.

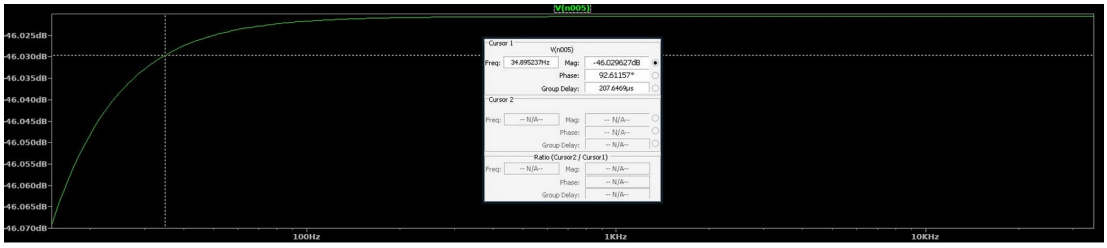
Αρχικά, για την είσοδο του σήματος στο ADC του STM32 χρησιμοποιείται ένα κανάλι που διαβάζει τιμές στο εύρος 0 – 3.3V, οπότε το σήμα θα πρέπει να είναι πολωμένο κατάλληλα στα 1.65V DC.



Εικόνα 5.3: Σχηματικό διάγραμμα διαιρέτη τάσης για την πόλωση του σήματος εισόδου στα 1.65V.

Επιπλέον στην έξοδο του τελεστικού εφαρμόζεται ένα υψηπερατό φίλτρο για την αφαίρεση χαμηλών συχνοτήτων, όπως τον θόρυβο που δημιουργείται από το ηλεκτρικό δίκτυο. Για την εφαρμογή αυτού του φίλτρου χρησιμοποιείται μια αντίσταση 10KΩ σε σειρά με έναν 470nF πυκνωτή στην αρνητική είσοδο με συχνότητα αποκοπής [49]:

$$f_s = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 10k \cdot 470n} = 33.86\ Hz \quad (5.1)$$



Εικόνα 5.4: Διάγραμμα Bode της προσομοίωσης του κυκλώματος υψηπερατού φίλτρου με το πρόγραμμα ltspice.

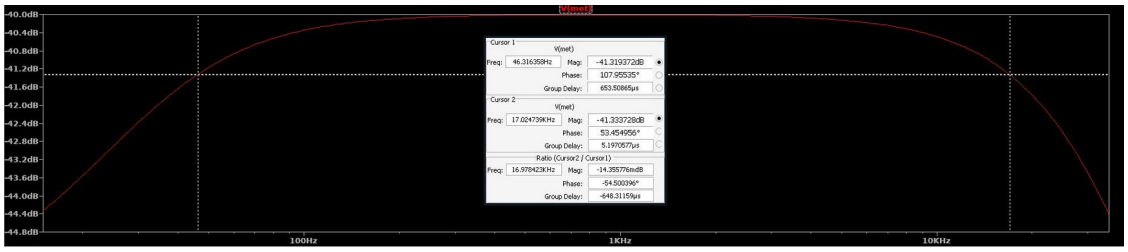
Από την άλλη εφαρμόζεται ενεργό χαμηλοπερατό φίλτρο με την χρήση 10KΩ αντίστασης παράλληλα με έναν 330pF πυκνωτή ανάμεσα σε αρνητική είσοδο και έξοδο με συχνότητα αποκοπής:

$$f_s = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 10k \cdot 330p} = 48.22\ kHz \quad (5.2)$$

Έπειτα το σήμα στην έξοδο του τελεστικού ενισχυτή περνάει από ένα παθητικό χαμηλοπερατό φίλτρο με αντίσταση 680Ω και πυκνωτλη 6.8nF με συχνότητα αποκοπής:

$$f_s = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 680 \cdot 6.8n} = 34.42\ kHz \quad (5.3)$$

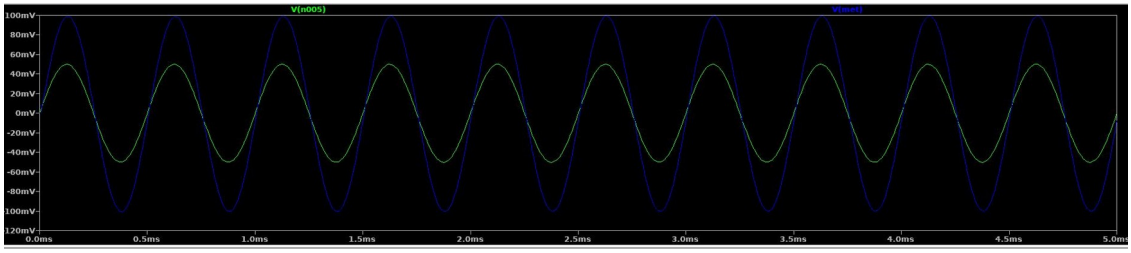
Η χρήση δύο χαμηλοπερατών φίλτρων, ενός ενεργού και ενός παθητικού βοηθάει στην αποτελεσματικότερη μείωση κέρδους των υψηλών συχνοτήτων με πιο απότομη αποκοπή. Στην πράξη, το υψηπερατό και τα χαμηλοπερατά φίλτρα δημιουργούν ένα ζωνοπερατό φίλτρο που στην πραγματικότητα καθώς δεν είναι ιδανικό ενισχύει συχνότητες στο εύρος 46Hz με 17kHz που είναι κατάλληλο για τις συχνότητες που παράγει μια ηλεκτρική κιθάρα.



Εικόνα 5.5: Διάγραμμα Bode της προσομοίωσης του κυκλώματος ζωνοπερατού φίλτρου με το πρόγραμμα ltspice.

Το μέγιστο κέρδος του τελεστικού σε μη αναστρέψιμη συνδεσμολογία όταν βρίσκεται σε κορεσμό διαμορφώνεται ως:

$$G = 1 + \frac{R_1}{R_2} = 1 + \frac{10k}{10k} = 2 \qquad (5.4)$$

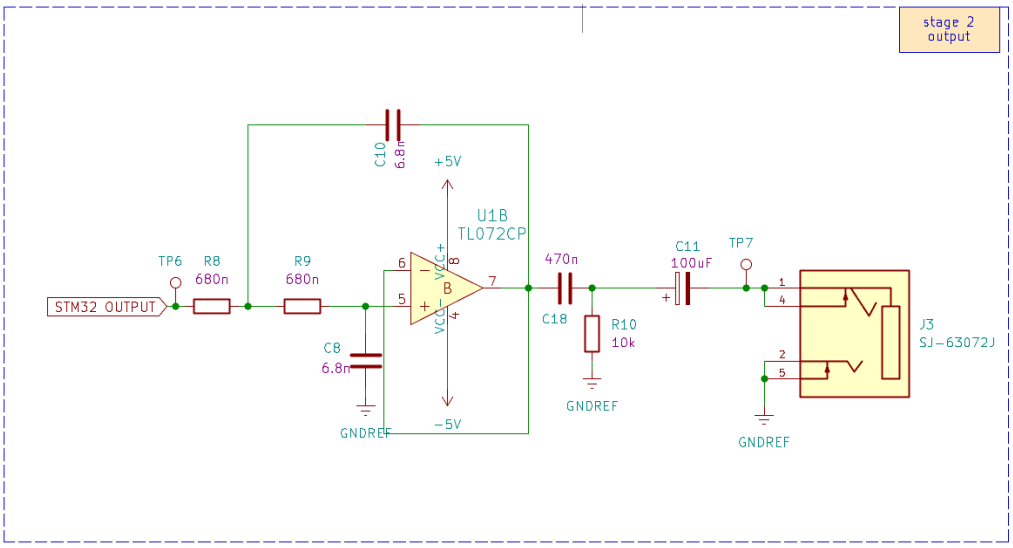


Εικόνα 5.6: Διάγραμμα Bode της προσομοίωσης του κυκλώματος εισόδου – εξόδου σε συχνότητα 2kHz. Το μπλε ημίτονο απαρτίζει την έξοδο με ενίσχυση 2 ενώ το πράσινο ημίτονο είναι το σήμα εισόδου.

Τέλος για να διασφαλισθεί πως το σήμα εισόδου δεν ξεπερνάει το εύρος τιμών 0 – 3.3V, χρησιμοποιήθηκε η τεχνική ψαλιδισμού μέσω διόδων, όπου εάν το σήμα περάσει την τιμή που άγουν οι δίοδοι, τότε η κορυφή αποκόβεται.

5.4.2 Προσαρμογή Σήματος Φάσης Εξόδου

Το κύκλωμα εξόδου χρησιμοποιεί έναν τελεστικό ενισχυτή σε μη αναστρέψιμη συνδεσμολογία. Το σήμα βγαίνει από το DAC του STM32 και προσαρμόζεται κατάλληλα για να σταλθεί στον ενισχυτή ηλεκτρικής κιθάρας.



Εικόνα 5.7: Σχηματικό διάγραμμα κυκλώματος εξόδου.

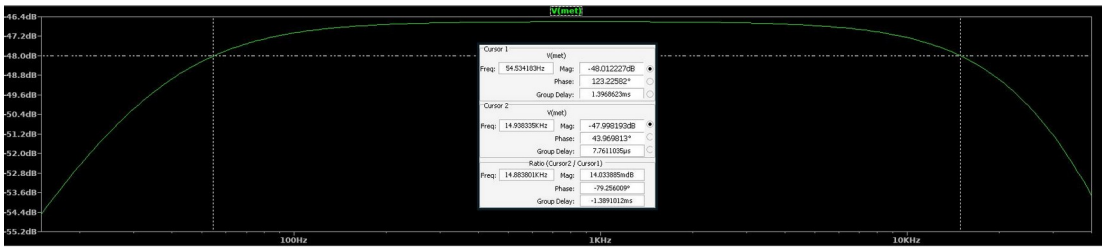
Αρχικά, το σήμα εισέρχεται σε ένα ενεργό χαμηλοπερατό φίλτρο σε Shallen – Key τοπολογία που αποτελείται από τον τελεστικό ενισχυτή, δύο αντιστάσεις και δύο πυκνωτές. Σε αυτήν την τοπολογία ο τελεστικός λειτουργεί ως ένας ακόλουθος τάσης, χωρίς να προσδίδει ενίσχυση στο σήμα, διατηρώντας κοινό κέρδος κατά όλο το εύρος συχνοτήτων [50]. Η συχνότητα αποκοπής, ομοίως όπως στο κύκλωμα εισόδου, διαμορφώνεται ως:

$$f_s = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 680 \cdot 6.8n} = 34.42\ kHz \quad (5.5)$$

Στη συνέχεια το σήμα διέρχεται μέσα από ένα παθητικό υψηπερατό φίλτρο με συχνότητα αποκοπής:

$$f_s = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 10K \cdot 470n} = 33.86\ Hz \quad (5.6)$$

και έπειτα το σήμα περνάει μέσα από έναν ηλεκτρολυτικό πυκνωτή με τιμή 100μF για την απόρριψη DC σταθερών. Στην πράξη, οι πραγματικές συχνότητες που ενισχύονται εν τέλει και στέλνονται στον ενισχυτή μπορούν να βρεθούν στο εύρος 54Hz με 15kHz.

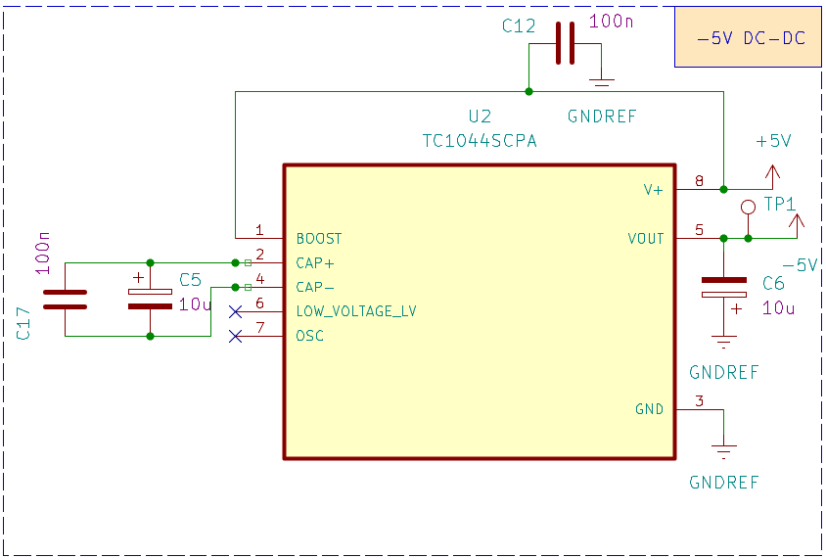


Εικόνα 5.8: Διάγραμμα Bode της προσομοίωσης του κυκλώματος εξόδου με συχνότητα 2kHz.

Στο παραπάνω διάγραμμα Bode, είναι εμφανές πως η χρήση ενός μόνο χαμηλοπερατού φίλτρου δεν αποδίδει εξίσου απότομη κλίση κατά την συχνότητα αποκοπής, χωρίς να επηρεάζει όμως αρνητικά το κύκλωμα.

5.4.3 Τροφοδοσία

Ο τελεστικός ενισχυτής TL072CP χρειάζεται το λιγότερο 4.5V τάση τροφοδοσίας για να λειτουργήσει. Ωστόσο για την αποδοτικότερη λειτουργία του συνίσταται τροφοδοσία τουλάχιστον 9V [51]. Για την βέλτιστη απόδοση του τελεστικού με την χρήση διπλής τροφοδοσίας +-5V, έγινε χρήση του DC – DC μετατροπέα TC1044SCPA ο οποίος είναι ικανός να αντιστρέψει την τάσεις εισόδου του εύρους +1.5V με +12V. Ένα σημαντικό προτέρημα του συγκεκριμένου μετατροπέα είναι ότι μπορεί να ενισχύσει την συχνότητα τροφοδοσίας στα 45kHZ ώστε οι συχνότητες που παράγονται από την τροφοδοσία να μην είναι ακουστές από την ανθρώπινη ακοή.

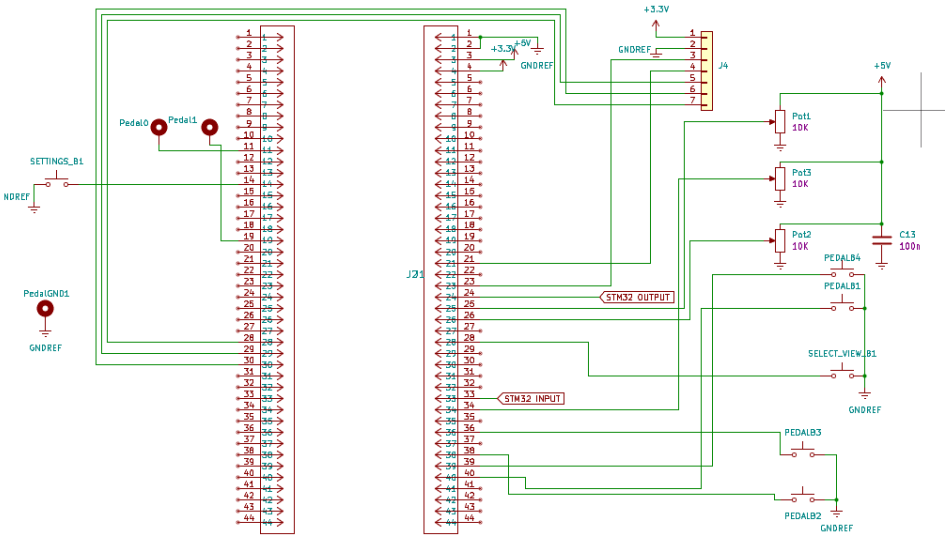


Εικόνα 5.9: Σχηματικό διάγραμμα κυκλώματος DC – DC μετατροπέα για παραγωγή -5V.

Η σχεδίαση του κυκλώματος ακολούθησε τις προδιαγραφές του κατασκευαστή με την προσθήκη ενός πυκνωτή κοντά στην είσοδο την τάσης τροφοδοσίας για σταθερότητα και αποκοπή θορύβου. Ομοίως τοποθετούνται πυκνωτές ίδιων τιμών κοντά στις τροφοδοσίες του τελεστικού ενισχυτή.

5.5 Σχεδιασμός Διεπαφής

Όπως αναλύθηκε στην αρχιτεκτονική της εργασίας, η διεπαφή του χρήστη θα αποτελείται από κουμπιά, ποτενσιόμετρα και την οθόνη. Το κύκλωμα της διεπαφής σχεδιάστηκε έτσι ώστε η τοποθέτηση των εξαρτημάτων να είναι όσο το δυνατόν πιο βέλτιστη για την σχεδίαση του PCB.



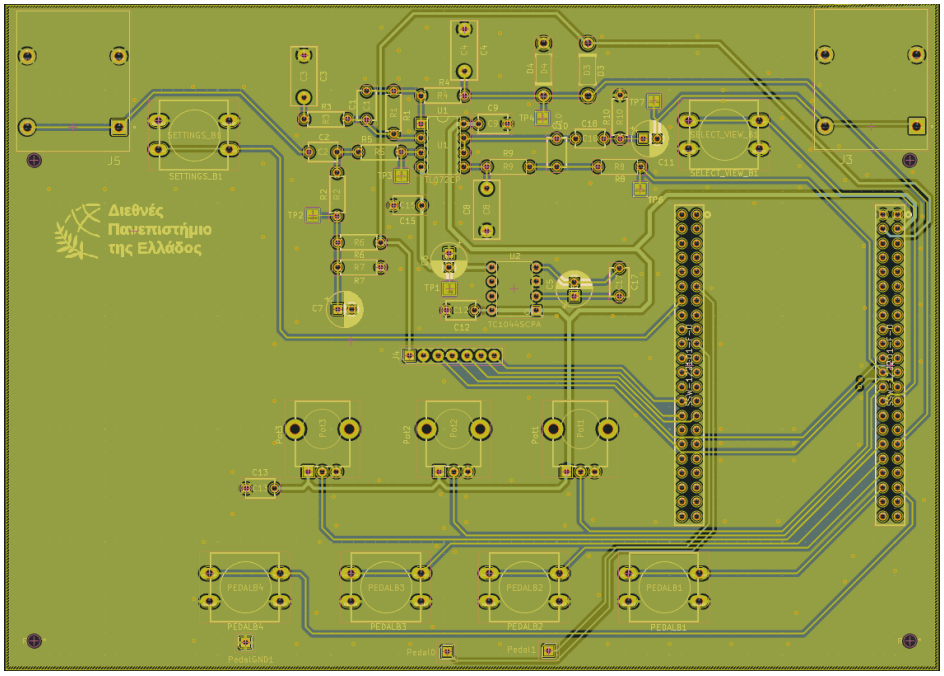
Εικόνα 5.10: Σχηματικό διάγραμμα κυκλώματος διεπαφής.

5.5.1 Οθόνη

Μετά από έρευνα και ανάλυση των απαιτήσεων του συστήματος, το μοντέλο οθόνης που επιλέχθηκε είναι το SSD1309, με πλακέτα driver ονόματι SFOMS242YZ7-12864WBYG-01 από τον κατασκευαστή Shenzhen Saef Technology [52]. Πρόκειται για μια μονόχρωμη οθόνη τεχνολογίας OLED, η οποία αποδίδει υψηλή ευκρίνεια και καταναλώνει χαμηλή ενέργεια. Επιπλέον, η ανάλυση της οθόνης είναι 128x64 pixels και χρώμα άσπρο. Τα προαναφερθέντα χαρακτηριστικά είναι πλήρως άρτια για την χρήση του συστήματος με την δημιουργία ενός κατάλληλου γραφικού περιβάλλοντος που θα αναλυθεί περαιτέρω παρακάτω. Η συνδεσμολογία της οθόνης είναι η Serial Peripheral Interface (SPI). Για την σύνδεση της οθόνης χρειάζονται 7 ακροδέκτες, οι 2 από τους οποίους είναι η τροφοδοσία και η γείωση. Η λειτουργία του Serial Peripheral Interface (SPI) πρωτοκόλλου θα εξηγηθεί σε μεταγενέστερο κεφάλαιο. Η συγκεκριμένη οθόνη λειτουργεί με τάση τροφοδοσίας +3.3V.

5.6 Υλοποίηση Τυπωμένου Κυκλώματος

Για τον σχεδιασμό της πλακέτας τυπωμένου κυκλώματος (PCB) χρησιμοποιήθηκε το πρόγραμμα kiCAD. Επιλέχθηκε η χρήση 2 επιπέδων χαλκού και το τελικό μέγεθος της πλακέτας είναι 165mmx118mm. Εφόσον σχεδιασθεί το σχηματικό διάγραμμα του κυκλώματος, το kiCAD επιτρέπει την άμεση αντιστοίχηση του κάθε συμβόλου με το αντίστοιχο footprint, που ορίζει το φυσικό αποτύπωμα του εξαρτήματος που θα τοποθετηθεί επάνω στο PCB. Επιπλέον συσχετίζει και τις αντίστοιχες συνδέσεις μεταξύ τους. Για την κατασκευή του κυκλώματος επιλέχθηκε η κινέζικη εταιρεία JLCPCB, καθώς προσφέρει γρήγορες και αποτελεσματικές λύσεις εκτυπώσεων κυκλωμάτων με χαμηλό κόστος.

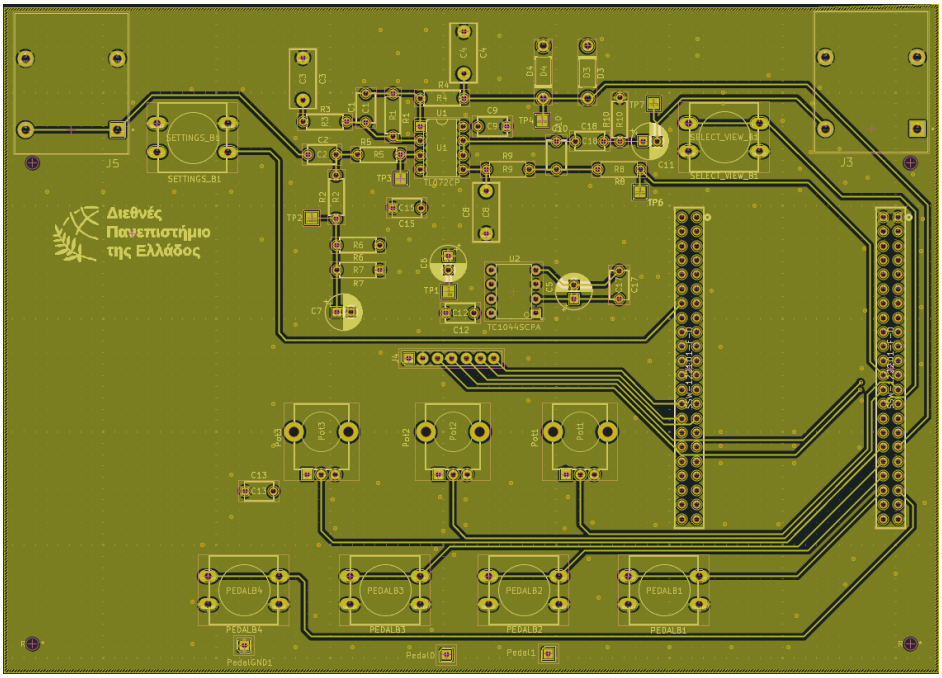


Εικόνα 5.12: Δισδιάστατη απεικόνιση του τυπωμένου κυκλώματος μέσω kiCAD.

Τα εξαρτήματα τοποθετήθηκαν στην πλακέτα με τέτοιον τρόπο, έτσι ώστε οι γραμμές να σχεδιαστούν με ευκολία, χωρίς να χρειάζονται πολλές διελεύσεις (vias), που επιτρέπουν την μεταλλαγή μιας γραμμής σε άλλο επίπεδο, καθώς αποδίδουν θόρυβο στο κύκλωμα. Στην πάνω μέση τοποθετήθηκε το κύκλωμα

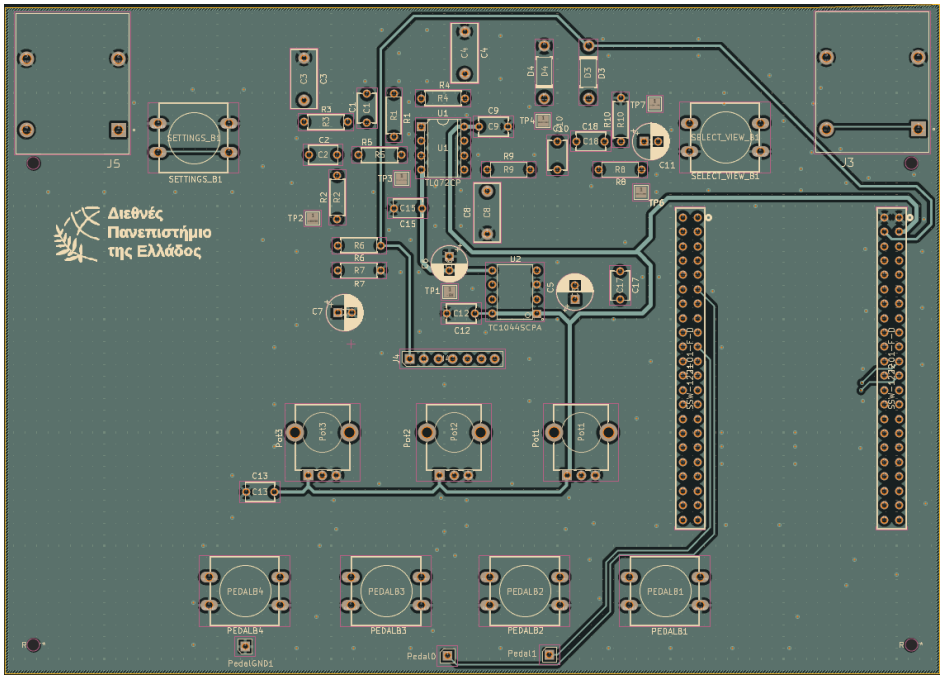
του προενισχυτή και φιλτραρίσματος των σημάτων εισόδου – εξόδου, ενώ στα αριστερά είναι η υποδοχή του ακροδέκτη για το σήμα εισόδου της κιθάρας και στα δεξιά η υποδοχή του ακροδέκτη για το σήμα εξόδου στον ενισχυτή. Στην μέση βρίσκονται οι ακροδέκτες για τις 7 συνδέσεις που χρειάζονται για την OLED οθόνη με πρωτόκολλο επικοινωνίας SPI και κάτω από αυτό βρίσκονται οι θέσεις που θα τοποθετηθούν τα κουμπιά και τα ποτενσιόμετρα. Στα δεξιά της πλακέτας έχουν τοποθετηθεί οι υποδοχές για τον STM32H7. Τέλος, τα κουμπιά για την επιλογή μενού βρίσκονται πάνω δεξιά και αριστερά καθώς η οθόνη θα τοποθετηθεί στην μέση.

Και στα δύο επίπεδα χαλκού χρησιμοποιήθηκε κοινή περιοχή για την γείωση με την αξιοποίηση της τεχνικής ground stitching που ορίζει την χρήση πολλαπλών vias για την σύνδεση των δύο περιοχών γείωσης. Η τεχνική αυτή βοηθάει στην μείωση θορύβου που μαζεύουν οι περιοχές γείωσης λόγω ανατροφοδότησης του ρεύματος [53].



Εικόνα 5.13: Δισδιάστατη απεικόνιση της μπροστινής όψης του τυπωμένου κυκλώματος μέσω kiCAD.

Στην μπροστινή όψη του κυκλώματος τρέχουν οι γραμμές οι οποίες μεταφέρουν το σήμα εισόδου και εξόδου και οι περισσότερες γραμμές για τα εξαρτήματα της διεπαφής.



Εικόνα 5.14: Δισδιάστατη απεικόνιση της πίσω όψης του τυπωμένου κυκλώματος μέσω kiCAD.

Στην πίσω όψη του κυκλώματος τρέχουν κυρίως οι γραμμές τροφοδοσίας +3.3V, +-5V.

5.7 Πίνακας Υλικών

Πίνακας 5.1: Bill of Materials

Description	Designator	Qty
Capacitor 100nF polyester	C9, C12, C13, C15, C17	5
Capacitor 470nF polyester	C3	1
Capacitor 6.8nF polyester	C4	1
Capacitor 10uF electrolytic	C5, C6	2
Capacitor 100uF electrolytic	C11	1
Capacitor 47uF electrolytic	C7	1
Capacitor 330pF ceramic	C1	1
Capacitor 6.8nF ceramic	C10	1
Capacitor 470nF MKT	C18	1
Capacitor 100nF MKT	C2	1
Resistor 10KOhm 1%	R1, R3, R10	3
Resistor 680Ohm 1%	R4, R8, R9	3
Resistor 100KOhm 1%	R6, R7	2
Resistor 1kOhm 1%	R5	1
Resistor 1MOhm 1%	R2	1
Diode 1N5819	D1, D2	2
Potentiometer 10KOhm Linear	Pot1, Pot2, Pot3	3
Push Button	PedalB1, PedalB2, PedalB3,	8

	PedalB4, SELECT_VIEW_B1, SETTINGS_B1, Pedal0, Pedal1	
Tl072CP	U1	1
TC1044S	U2	1

5.8 Δοκιμή Τυπωμένου Κυκλώματος

Στην τυπωμένη πλακέτα έχουν προστεθεί σημεία δοκιμής (test points) για την ευκολότερη παρακολούθηση της συμπεριφοράς του. Συγκεκριμένα έχουν τοποθετηθεί, στην επιφάνεια της πλακέτας, σημεία εκτεθειμένου χαλκού που αντιστοιχούν σε σημεία μέτρησης, όπως για παράδειγμα η μέτρηση τάσης του σήματος εξόδου.

5.9 Επίλογος

Συνοψίζοντας, το κύκλωμα εισόδου-εξόδου που σχεδιάστηκε και υλοποιήθηκε αποτελεί ένα πλήρως λειτουργικό σύστημα προενίσχυσης, φιλτραρίσματος και διεπαφής, κατάλληλο για εφαρμογές επεξεργασίας ήχου της ηλεκτρικής κιθάρας. Μέσα από την προσεκτική επιλογή των εξαρτημάτων, τη χρήση προσομοιώσεων με την χρήση του προγράμματος LtSpice, τον ορθολογικό σχεδιασμό του PCB και την ενσωμάτωση του μικροελεγκτή STM32H7, επιτεύχθηκε η διασφάλιση της ποιότητας ήχου με ελάχιστο θόρυβο και το κόστος παρέμεινε σε χαμηλά επίπεδα, θέτοντας τα θεμέλια για την περαιτέρω ανάπτυξη των αλγορίθμων DSP και την ολοκλήρωση της διπλωματικής εργασίας.

Κεφάλαιο 6ο: Λογισμικό

6.1 Εισαγωγή

Το λογισμικό που αναπτύχθηκε για τον μικροελεγκτή STM32H7 αποτελεί το κεντρικό και σημαντικότερο τμήμα ολόκληρης της διπλωματικής εργασίας. Συγκεκριμένα, το πρόγραμμα έχει σχεδιαστεί για να επιτελεί τρεις βασικούς και αλληλένδετους ρόλους. Πρώτον, είναι αρμόδιο για την ψηφιακή επεξεργασία του σήματος, μέσω της οποίας παράγονται τα εφέ. Δεύτερον, το πρόγραμμα αναλαμβάνει την πλήρη ρύθμιση και διαμόρφωση όλων των περιφερειακών του μικροελεγκτή, όπως οι χρονιστές, ADC, DAC και GPIOs. Τρίτον, υλοποιεί τον έλεγχο της ίδιας της συσκευής και της φυσικής διεπαφής. Συνοπτικά, το λογισμικό συντονίζει τη λειτουργία του συστήματος για την εκτέλεση των προγραμματισμένων λειτουργιών.

6.2 Τεχνικές Προδιαγραφές Λογισμικού

Ομοίως με τον σχεδιασμό του κυκλώματος, η δημιουργία του λογισμικού απαιτεί έρευνα για την επιλογή των τεχνολογιών, την οργάνωση των μερών του κώδικα και την μέθοδο ανάπτυξης του.

- **Τεχνολογίες Προγραμματισμού:** Για την ανάπτυξη του προγράμματος χρησιμοποιήθηκε εξ ολοκλήρου η αντικειμενοστραφής γλώσσα προγραμματισμού C++. Η αντικειμενοστρέφεια επιτρέπει την δημιουργία αντικειμένων που χωρίζει τα επί μέρους κομμάτια σε αυτοτελείς οντότητες. Με αυτόν τον τρόπο, συχνά προάγεται η καλύτερη οργάνωση και επαναχρησιμοποίηση του κώδικα, κάνοντας τον πιο ευανάγνωστο και συντηρήσιμο. Ακόμη, η μοντέρνα C++, όπως C++11 και C++17, εισάγει καινούργιες λειτουργίες όπως ασφαλέστερη διαχείριση μνήμης και εξτρά βιβλιοθήκες που αντικαθιστούν ανάπτυξη κώδικα με την χρήση C-style προγραμματισμού. Επιπλέον, χρησιμοποιήθηκαν οι HAL (Hardware Abstraction Layer) και LL (Low Level) βιβλιοθήκες, γραμμένες από την ST Microrelectronics, για την διαμόρφωση και ρύθμιση των περιφερειακών του STM32H7.
- **Αποδοτικότητα Αλγορίθμων:** Οι αλγόριθμοι που καθιστούν τα εφέ για την επεξεργασία ήχου θα πρέπει να είναι γραμμένοι έτσι ώστε να καταναλώνουν όσο το δυνατόν λιγότερη μνήμη και επεξεργαστικούς κύκλους, δηλαδή να λειτουργούν με την λιγότερο δυνατή πολυπλοκότητα. Αυτό σημαίνει πως θα πρέπει να αποφεύγονται βαριές πράξεις, όπως διαίρεση, και λειτουργίες που καταναλώνουν πολλή μνήμη, όπως πίνακες.
- **Οργάνωση Βιβλιοθηκών και Συγγραφή Κώδικα:** Τα αρχεία θα πρέπει να είναι έτσι οργανωμένα και ονοματισμένα ώστε ο προγραμματιστής να μπορεί με ευκολία να περιηγηθεί την βάση του κώδικα και να βρίσκει τα μέρη του κώδικα που τον ενδιαφέρουν. Ως εκ τούτου, τα μέρη του κώδικα θα πρέπει να είναι και κατάλληλα χωρισμένα σε διαφορετικά αρχεία. Ακόμη, καθ’ όλη την συγγραφή του προγράμματος, θα πρέπει να διατηρείται μια συγκεκριμένη φιλοσοφία, τόσο για την ονοματοδοσία των συμβόλων, όσο και για την δομή του κώδικα. Με άλλα λόγια, θα πρέπει να τηρηθεί μια συγκεκριμένη αρχιτεκτονική ώστε ο κώδικας να είναι ευανάγνωστος, ευέλικτος και συντηρήσιμος.

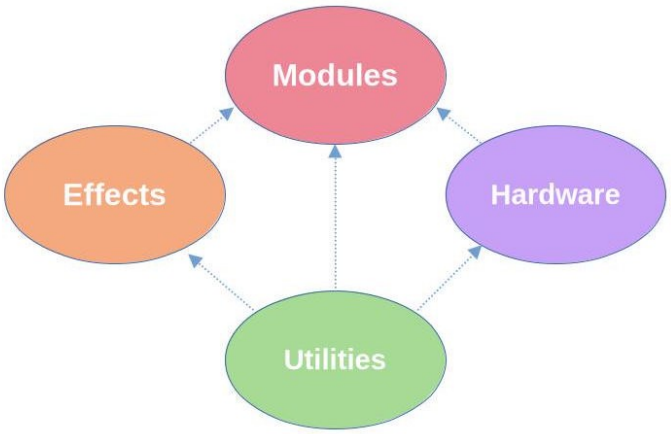
6.3 Εργαλεία Ανάπτυξης

Για την συγγραφή και οργάνωση του κώδικα επιλέχθηκε το Visual Studio Code ως το κύριο Ολοκληρωμένο Περιβάλλον Ανάπτυξης, integrated development environment (IDE) που πρόκειται για μια από τις πιο δημοφιλείς λύσεις IDE καθώς προσφέρει πολλές λειτουργίες που καθιστούν εύκολο τον

προγραμματισμό μιας εφαρμογής. Προφανώς, για την μεταγλώττιση της C++ σε γλώσσα μηχανής κατανοητή από τον STM32H7 χρησιμοποιήθηκε το STM32Cube IDE, δημιουργημένο από την ST Microelectronics αποκλειστικά για την χρήση με STM32 μικροελεγκτές. Ο μεταγλωττιστής που χρησιμοποιείται είναι ο GNU G++17 συγκεκριμένα για ARM επεξεργαστές και παράγει αρχείο .elf, το οποίο ανεβαίνει στον STM32H7 μέσω του προγράμματος STM32CubeProgrammer.

6.4 Αρχιτεκτονική Λογισμικού

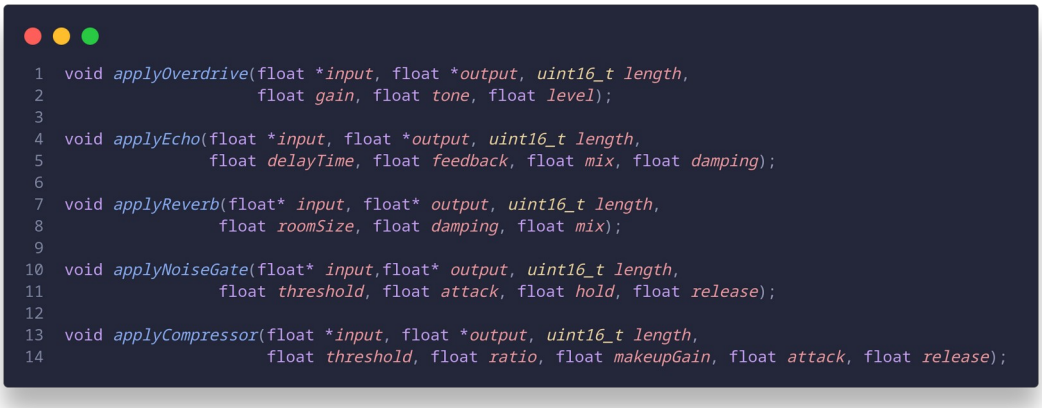
Το λογισμικό αποτελείται από 4 βασικές βιβλιοθήκες που απαρτίζουν τις διάφορες λειτουργίες του συστήματος. Ο διαχωρισμός αυτός έγινε με γνώμονα την καλύτερη οργάνωση του συστήματος, καθώς περιπλέκονται πολλά διαφορετικά κομμάτια μεταξύ τους. Έτσι λοιπόν χρησιμοποιήθηκε η τεχνική “διαίρει και βασίλευε” κατά την οποία το πρόβλημα διασπάται αναδρομικά σε επιμέρους προβλήματα, δηλαδή στην περίπτωση του λογισμικού, ο κώδικας διασπάται σε επιμέρους αρχεία κώδικα. Έτσι λοιπόν, δημιουργήθηκαν 4 βιβλιοθήκες, μια για την επεξεργασία και δημιουργία οντοτήτων που απαρτίζουν ένα εφέ, μια για την διαχείριση των περιφερειακών και υλικού, μια για την κύρια διεπαφή και μια κοινή βιβλιοθήκη με χρήσιμα εργαλεία που χρησιμοποιείται από τις υπόλοιπες. Η κύρια διεπαφή είναι αρμόδια για την διαχείριση των υπολοίπων και αλληλεπιδρά με όλες τις βιβλιοθήκες.



Σχήμα 6.4: Διάγραμμα αλληλεπίδρασης λογισμικού.

6.5 DSP

Η βιβλιοθήκη Digital Signal Processing (DSP) περιέχει όλους τους μαθηματικούς αλγορίθμους και μεθόδους που απαρτίζουν ένα εφέ. Η συνάρτηση *process* του κάθε εφέ, καλεί τη αντίστοιχη μέθοδο που αποτυπώνει το εικονικό αναλογικό μοντέλο του κάθε εφέ, σύμφωνα με τους αλγορίθμους που εξετάστηκαν στο Κεφάλαιο 3. Η κάθε συνάρτηση δέχεται ως παραμέτρους τους δείκτες των πινάκων εισόδου – εξόδου μαζί με το μέγεθος και τις μεταβλητές για το κάθε πετάλι εφέ.



```
void applyOverdrive(float *input, float *output, uint16_t length,
                    float gain, float tone, float level);

void applyEcho(float *input, float *output, uint16_t length,
               float delayTime, float feedback, float mix, float damping);

void applyReverb(float* input, float* output, uint16_t length,
                 float roomSize, float damping, float mix);

void applyNoiseGate(float* input,float* output, uint16_t length,
                 float threshold, float attack, float hold, float release);

void applyCompressor(float *input, float *output, uint16_t length,
                     float threshold, float ratio, float makeupGain, float attack, float release);
```

Εικόνα 6.10: Μέθοδοι του κάθε εφέ, προσβάσιμες από την κεντρική διεπαφή.

Τα φίλτρα εφαρμόζονται ως βοηθητικές συναρτήσεις και κλάσεις που χρησιμοποιούνται από τα εφέ και δεν επιλέγονται από την διεπαφή χρήστη.

```

namespace Config
{
    constexpr float    SAMPLE_RATE     = 96000.0f;
    constexpr float    SAMPLE_RATE_MS  = SAMPLE_RATE * 0.001f;

    constexpr uint32_t ECHO_BUFFER_SIZE = 32768;
    constexpr float    ECHO_FEEDBACK_GAMMA = 0.7f;
    constexpr float    ECHO_MIN_DELAY_MS   = 20.0f;
    constexpr float    ECHO_MAX_DELAY_MS   = 680.0f;

    constexpr uint8_t  NUM_COMBS    = 8;
    constexpr uint8_t  NUM_ALLPASS  = 4;
    constexpr float    ALLPASS_FEEDBACK = 0.5f;

    constexpr std::array<uint32_t, NUM_COMBS> COMB_DELAYS = {
        3163, 3617, 4013, 4409, 4801, 5209, 5597, 6007
    };
    constexpr std::array<uint32_t, NUM_ALLPASS> ALLPASS_DELAYS = {
        556, 441, 341, 225
    };

    constexpr float MIN_HOLD_MS    = 0.0f;
    constexpr float MAX_HOLD_MS    = 500.0f;
    constexpr float MIN_ATTACK_MS  = 1.0f;
    constexpr float MAX_ATTACK_MS  = 100.0f;
    constexpr float MIN_RELEASE_MS = 10.0f;
    constexpr float MAX_RELEASE_MS = 1000.0f;
}
```

Εικόνα 6.11: Κοινές ρυθμίσεις αλγορίθμων εφέ.

Στο αρχείο υπάρχουν βοηθητικές ρυθμίσεις υπό την μορφή σταθερών *constexpr* που παίρνουν τιμή κατά την μεταγλώττιση. Οι ρυθμίζεις αυτές συνιστούν στην κανονικοποίηση κάποιων παραμέτρων ώστε να λειτουργήσουν κατάλληλα πάνω στο εφέ.

6.5.1 Low Pass Filter

```
1  struct LpfState
2  {
3      float prevOut = 0.0f;
4
5      float process(float sample, float tone, LpfState& state)
6      {
7          const float out    = tone * sample + (1.0f - tone) * state.prevOut;
8          state.prevOut        = lpOut;
9          return lpOut ;
10     }
11 };
```

Εικόνα 6.12: Κώδικας IIR χαμηλοπερατού φίλτρου.

Ο αλγόριθμος του χαμηλοπερατού φίλτρου εμπεριέχεται στην κλάση που το απαρτίζει, *LpfState*. Εφόσον κληθεί η συνάρτηση *process*, εφαρμόζεται η αναδρομική εξίσωση πρώτου βαθμού για την εύρεση του κινητού μέσου όρου, όπως εξηγήθηκε στο 3ο Κεφάλαιο. Η έξοδος o*ut* υπολογίζεται με βάση το τρέχον δείγμα *sample* και της προηγούμενης εξόδου *state.prevOut*, με βάση την παράμετρο *tone* που “χρωματίζει” τον ήχο.

6.5.2 High Pass Filter

Ο αλγόριθμος του υψιπερατού φίλτρου εμπεριέχεται στην κλάση που το απαρτίζει, *HpfState*. Εφόσον κληθεί η συνάρτηση *process*, εφαρμόζεται μια αναδρομική εξίσωση πρώτου βαθμού, όπως εξηγήθηκε στο 3ο Κεφάλαιο. Η έξοδος o*ut* υπολογίζεται με βάση το τρέχον δείγμα *sample* και της προηγούμενης εισόδου *state.prevOut* και της προηγούμενης εξόδου, με βάση την παράμετρο *tone* που “χρωματίζει”τον ήχο.

```cpp
struct HpfState
{
    float prevIn  = 0.0f;
    float prevOut = 0.0f;

    float process(float sample, float tone, HpfState& state)
    {
        const float out  = sample - state.prevIn + tone * state.prevOut;
        state.prevIn  = sample;
        state.prevOut = out;
        return out;
    }
};
```

Εικόνα 6.13: Κώδικας IIR υψιπερατού φίλτρου.

6.5.3 Comb Filter

Ο αλγόριθμος του comb φίλτρου εμπεριέχεται στην κλάση που το απαρτίζει, *CombFilterState*. Η κλάση περιέχει έναν πίνακα μεγέθους 6000 (μέγιστη καθυστέρηση περίπου 60ms για 96kHz). Εφαρμόζει ένα χαμηλοπερατό φίλτρο στον πίνακα και χρησιμοποιεί μια κυκλική διαδικασία για την εφαρμογή της καθυστέρησης. Αρχικά, γράφει το καθυστερημένο δείγμα στον πίνακα με την προσθήκη του φίλτρου και της ανάδρασης *feedback* και έπειτα προσαυξάνει τον δείκτη ανάλογα με τον χρόνο καθυστέρησης *delayLength*.

```cpp
struct CombFilterState
{
    std::array<float, 6008> buffer{};
    uint32_t position   = 0;
    LpfState lpfState   = {};

    float process(float input, uint32_t delayLength, float feedback, float damping)
    {
        const float delayed = buffer[position];
        const float filtered = lpfState.process(delayed, damping, lpfState);
        buffer[position] = input + filtered * feedback;
        position = (position + 1 < delayLength) ? position + 1 : 0;
        return delayed;
    }
};
```

Εικόνα 6.14: Κώδικας comb φίλτρου.

6.5.4 All-Pass Filter

Ο αλγόριθμος του all-pass φίλτρου εμπεριέχεται στην κλάση που το απαρτίζει, Allpass*FilterState*. Η κλάση περιέχει έναν πίνακα μεγέθους 1024 δειγμάτων. Το φίλτρο αφήνει να περάσουν όλες οι συχνότητες και η απόκριση στο φάσμα συχνοτήτων παραμένει σταθερή. Αλλάζει μόνο την φάση του σήματος εισόδου

```
1  struct AllpassFilterState
2  {
3      std::array<float, 1024> buffer{};
4      uint32_t position = 0;
5
6      float process(float input, uint32_t delayLength, float feedback)
7      {
8          const float delayed = buffer[position];
9          buffer[position] = input + feedback * delayed;
10         const float output = -input + delayed;
11
12         position = (position + 1 < delayLength) ? position + 1 : 0;
13         return output;
14     }
15 };
```

Εικόνα 6.15: Κώδικας all-pass φίλτρου.

6.5.5 Overdrive Distortion

Η μέθοδος που εφαρμόζει τον παραμορφωτή αρχικά κανονικοποιεί την παράμετρο του κέρδους *gain* ώστε να είναι αισθητή στον αλγόριθμο. Έπειτα το κάθε δείγμα περνάει μέσα από ένα υψιπερατό φίλτρο και έπειτα εφαρμόζεται σε αυτό το κέρδος. Η παραμόρφωση γίνεται με την χρήση υπερβολικής εφαπτομένης που εισάγει ασυμμετρία και επιδρά απαλά στο σήμα με την βοήθεια του soft clipping, με την κανονικοποίηση των τιμών.

```
void applyOverdrive(float* input, float* output, uint16_t length,
                    float gain, float tone, float level)
 {
     static HpfState hpfState{};
     static LpfState lpfState{};

     float g = 1.0f + gain * 20.0f

     for (uint16_t i = 0; i < length; ++i)
     {
         float sample = input[i];

         sample = hpfState.process(sample, 0.95f, hpfState);

         sample *=  g;

         float x0 = 0.1f + gain * 0.3f;
         float k1 = 0.7f;
         float k2 = 1.3f;

         float y1 = tanhf(k1 * (sample + x0)) - tanhf(k1 * x0);
         float y2 = tanhf(k2 * (sample + x0)) - tanhf(k2 * x0);
         sample = (y1 + y2) * 0.5f;

         sample = lpfState.process(sample,  0.7f + tone * 0.8f, lpfState);

         output[i] = sample * level * 1.5f;
     }
 }
```

Εικόνα 6.16: Κώδικας overdrive distortion εφέ.

6.5.6 Reverb

Το εφέ της αντήχησης αποτελείται από 8 comb και 4 all-pass φίλτρα. Στην είσοδο εισέρχεται το dry σήμα το οποίο μετέπειτα περνάει από τα 8 παράλληλα comb φίλτρα και μετά σειριακά από τα all-pass. Η παράμετρος *mix* ορίζει την ένταση του επεξεργασμένου wet σήματος σε σχέση με το dry σήμα εισόδου.

```cpp
void applyReverb(float* input, float* output, uint16_t length,
                 float roomSize, float damping, float mix)
  {
      roomSize = clamp(roomSize, 0.0f, 1.0f);
      damping  = clamp(damping,  0.0f, 1.0f);
      mix      = clamp(mix,      0.0f, 1.0f);

      const float combFeedback = 0.7f + (roomSize * 0.25f);

      for (uint16_t i = 0; i < length; ++i)
      {
          const float dry = input[i];
          float wet       = 0.0f;

          for (size_t c = 0; c < Config::NUM_COMBS; ++c)
              wet += combFilters[c].process(dry, Config::COMB_DELAYS[c], combFeedback, damping);

          wet *= 0.125f;

          for (size_t a = 0; a < Config::NUM_ALLPASS; ++a)
          {
              wet = allpassFilters[a].process(wet, Config::ALLPASS_DELAYS[a], Config::ALLPASS_FEEDBACK);
          }

          output[i] = (1.0f - mix) * dry + mix * wet;
      }
  }
```

Εικόνα 6.17: Κώδικας reverb εφέ.

6.5.7 Echo

Η μέθοδος της ηχώ χρησιμοποιεί την γραμμή καθυστέρησης για την δημιουργία διαδοχικών ηχών που αναπαράγονται ανάλογα με τον χρόνο καθυστέρησης που έχει επιλεχθεί από τον χρήστη. Χρησιμοποιεί μονάχα ένα comb φίλτρο.

```cpp
void applyEcho(float *input, float *output, uint16_t length,
               float delayTime, float feedback, float mix)
{
    delayTime = clamp(delayTime, 0.0f, 1.0f);
    feedback  = clamp(feedback, 0.0f, 0.95f);
    mix       = clamp(mix, 0.0f, 1.0f);

    float delayMs = Config::ECHO_MIN_DELAY_MS + delayTime * (Config::ECHO_MAX_DELAY_MS - Config::ECHO_MIN_DELAY_MS);
    const uint32_t delaySamples = std::min(
        static_cast<uint32_t>((delayMs / 1000.0f) * Config::SAMPLE_RATE),
        Config::ECHO_BUFFER_SIZE - 1);

    for (uint16_t i = 0; i < length; ++i)
    {
        output[i] = echoState.process(input[i], delaySamples, feedback,
                                      Config::ECHO_FEEDBACK_GAMMA, mix);
    }
}
```

Εικόνα 6.18: Κώδικας echo εφέ.

6.5.8 Noise Gate

Για την εφαρμογή του noise gate στο σήμα αρχικά κανονικοποιούνται οι παράμετροι που ελέγχουν την συμπεριφορά του εφέ. Οι τιμές που προσδιορίζουν χρόνους κανονικοποιούνται σε χιλιοστά του δευτερολέπτου, ανάλογα με τον ρυθμό δειγματοληψίας. . Έπειτα εφαρμόζεται ο αλγόριθμος του καταστολέα θορύβου, ανάλογα με το κατώφλι αποκοπής που έχει ορισθεί.

```cpp
void applyNoiseGate(float *input, float *output, uint16_t length,
                    float threshold, float attack, float hold, float release)
{
    threshold = clamp(threshold, 0.0f, 1.0f);
    hold      = clamp(hold, 0.0f, 1.0f);
    attack    = clamp(attack, 0.0f, 1.0f);
    release   = clamp(release, 0.0f, 1.0f);

    static NoiseGateState state{};

    const float T = 0.01f + threshold * 0.5f;

    const float tHold    = Config::MIN_HOLD_MS + hold * (Config::MAX_HOLD_MS - Config::MIN_HOLD_MS);
    const float tAttack  = Config::MIN_ATTACK_MS + attack * (Config::MAX_ATTACK_MS - Config::MIN_ATTACK_MS);
    const float tRelease = Config::MIN_RELEASE_MS + release * (Config::MAX_RELEASE_MS - Config::MIN_RELEASE_MS);

    const uint32_t N_hold    = static_cast<uint32_t>(tHold * Config::SAMPLE_RATE_MS);
    const uint32_t N_attack  = static_cast<uint32_t>(tAttack * Config::SAMPLE_RATE_MS);
    const uint32_t N_release = static_cast<uint32_t>(tRelease * Config::SAMPLE_RATE_MS);

    const float a  = 0.9896f;                  // a = exp(-1 / (1ms * Fs))
    const float b0 = (1.0f - a) * (1.0f - a);  // (1-a)^2
    const float b1 = 2.0f * a;                 // 2a
    const float b2 = -(a * a);                 // -a^2

    const float Δg_attack  = (N_attack > 0) ? (1.0f / (float)N_attack) : 1.0f;
    const float Δg_release = (N_release > 0) ? (1.0f / (float)N_release) : 1.0f;

    float y2 = state.y2;
    float y1 = state.y1;
    float g  = state.g;
    uint32_t nBelow = state.nBelow;
    uint32_t nAbove = state.nAbove;

    for (uint16_t i = 0; i < length; ++i)
    {
        const float x_n = input[i];

        const float y2_n = b0 * fabsf(x_n) + b1 * y1 + b2 * y2;

        const float y1_next = y2;
        y2 = y2_n;
        y1 = y1_next;

        bool gateOpen = (y2 >= T);

        if (gateOpen)
        {
            nAbove++;
            nBelow = 0;

            g = g + Δg_attack;
            if (g > 1.0f)
                g = 1.0f;
        }
        else
        {
            nBelow++;
            nAbove = 0;

            if (nBelow > N_hold)
            {
                g = g - Δg_release;
                if (g < 0.0f)
                    g = 0.0f;
            }
        }

        output[i] = x_n * g;
    }

    state.y2 = y2;
    state.y1 = y1;
    state.g = g;
    state.nBelow = nBelow;
    state.nAbove = nAbove;
}
```

Εικόνα 6.19: Κώδικας noise gate εφέ.

6.5.9 Compressor

Το εφέ του κομπρέσσορα χρησιμοποιεί τις μαθηματικές συναρτήσεις που εξηγήθηκαν στο 3ο κεφάλαιο για την ανίχνευση της κορυφής του κάθε δείγματος, ώστε οι κορυφές του σήματος να έρθουν στο ίδιο επίπεδο.

```cpp
void applyCompressor(float* input, float* output, uint16_t length,
                     float threshold, float ratio, float makeupGain, float attack, float release)
{
    threshold  = clamp(threshold,  0.0f, 1.0f);
    ratio      = clamp(ratio,      0.0f, 1.0f);
    makeupGain = clamp(makeupGain, 0.0f, 1.0f);
    attack     = clamp(attack,     0.0f, 1.0f);
    release    = clamp(release,    0.0f, 1.0f);

    const float T = 0.01f + threshold * 0.5f;
    const float R = 1.0f / (1.0f + ratio * 2.5f);
    const float makeup = powf(10.0f, makeupGain * 2.0f);

    const float αA = expf(-1000.0f / ((Config::MIN_ATTACK_MS + attack * (Config::MAX_ATTACK_MS - Config::MIN_ATTACK_MS)) * Config::SAMPLE_RATE));
    const float αR = expf(-1000.0f / ((Config::MIN_RELEASE_MS + release * (Config::MAX_RELEASE_MS - Config::MIN_RELEASE_MS)) * Config::SAMPLE_RATE));

    static float y = 0.0f;
    static float gainSmooth = 1.0f;

    const float W = 0.2f;

    for (uint16_t i = 0; i < length; ++i)
    {
        const float x_n = fabsf(input[i]);

        if (x_n > y) {
            y = αA * y + (1.0f - αA) * x_n;
        } else {
            y = αR * y + (1.0f - αR) * x_n;
        }

        float targetGain = 1.0f;

        if (y > T - W/2.0f) {
            const float x_db = 20.0f * log10f(y + 1e-10f);
            const float T_db = 20.0f * log10f(T + 1e-10f);
            const float W_db = 20.0f * log10f(W + 1e-10f);

            const float x_minus_T = x_db - T_db + W_db/2.0f;
            const float slope = (1.0f / R) - 1.0f;
            float output_db = x_db + (slope * x_minus_T * x_minus_T) / (2.0f * W_db);

            targetGain = powf(10.0f, (output_db - x_db) / 20.0f);
        }

        if (targetGain < gainSmooth) {
            gainSmooth = αA * gainSmooth + (1.0f - αA) * targetGain;
        } else {
            gainSmooth = αR * gainSmooth + (1.0f - αR) * targetGain;
        }

        output[i] = input[i] * gainSmooth * makeup;
    }
}
```

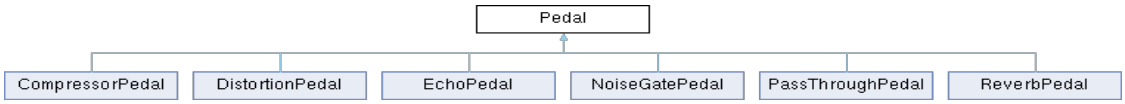
Εικόνα 6.20: Κώδικας compressor εφέ.

6.6 Βιβλιοθήκη Εφέ

Η συγκεκριμένη βιβλιοθήκη περιέχει όλες τις λειτουργίες και οντότητες που απαρτίζουν ένα αντικείμενο εφέ και μια σειρά από εφέ. Η βιβλιοθήκη καλείται και χρησιμοποιείται μόνο από την κύρια διεπαφή. Περιγράφει σε διαφορετικά αρχεία τις οντότητες των πεταλιών (Pedals) και της σειράς πεταλιών (Effects Chain) και οι αλγόριθμοι επεξεργασίας ήχου είναι ομαδοποιημένοι σε ένα αρχείο (DSP).

6.6.1 Pedal

Η κλάση Pedal περιγράφει ένα πετάλι και όλες τις παραμέτρους και συναρτήσεις που το περιγράφουν. Αυτή η κλάση κληρονομείται από υποκλάσεις που απαρτίζουν ένα συγκεκριμένο εφέ. Συνολικά υπάρχουν 6 διαφορετικά εφέ με το καθ’ ένα να περιέχει τις δικές του ξεχωριστές παραμέτρους.



Σχήμα 6.5: Διάγραμμα κληρονομικότητας Pedal κλάσης.

Αρχικά, η κλάση Pedal περιέχει κοινές συναρτήσεις για όλες τις υποκλάσεις. Το κάθε ένα πετάλι εφέ περιγράφεται από τις δικές του μοναδικές παραμέτρους και αριθμό παραμέτρων. Όλες οι παράμετροι λειτουργούν στο εύρος ανάμεσα στο 0 και 1. Η παράμετρος v*olume* είναι κοινή σε όλα τα πετάλια. Επιπλέον, το κάθε πετάλι περιέχει το δικό του μοναδικό όνομα και δύο bitmap εικόνες για την εμφάνιση ενός ενεργού ή ανενεργού πεταλιού.

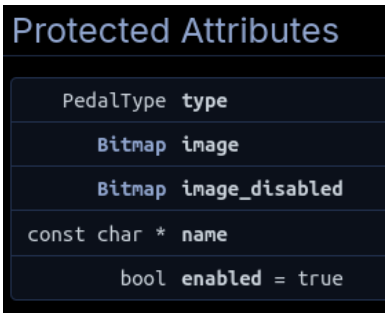
Public Member Functions

```
                          Pedal (PedalType t)
            const Bitmap & getImage () const
                PedalType  getType () const
              const char * getName () const
virtual const char *const * getMemberNames () const
           virtual uint8_t getMemberSize () const
             virtual void  set (float *values)=0
             virtual void  get (float *values) const =0
             virtual void  process (float *input, float *output, uint16_t length)=0
             virtual bool  isEnabled () const
             virtual void  setEnabled (bool state)
```

Εικόνα 6.21: Απεικόνιση των κοινών δημόσιων συναρτήσεων της κλάσης Pedal.

- *const Bitmap &* ***getImage*** *() const*: Επιστρέφει την εικόνα του πεταλιού η οποία χαρακτηρίζεται από την κλάση *Bitmap*. Το *const* στο τέλος εξασφαλίζει πως δεν θα αλλαχθούν τα δεδομένα της κλάσης *Bitmap*.

- *PedalType* ***getType*** *() const*: Επιστρέφει τον τύπο του πεταλιού που περιγράφεται από την enumeration κλάση *PedalType.* Εξασφαλίζει πως δεν θα αλλαχθούν τα δεδομένα της κλάσης *PedalType*.
- *const char * **getName** () const*: Επιστρέφει τον δείκτη από έναν πίνακα χαρακτήρων του ονόματος του πεταλιού. Εξασφαλίζει πως δεν θα αλλαχθούν τα δεδομένα του πίνακα.
- *virtual const char *const * **getMemberNames** () const*: Επιστρέφει τον δείκτη σε έναν πίνακα που περιέχει όλα τα ονόματα των μελών (παραμέτρων) ενός εφέ. Το *virtual* ορίζει πως η μέθοδος μπορεί να γίνει override από μια υποκλάση. Εξασφαλίζει πως δεν θα αλλαχθούν τα δεδομένα του πίνακα.
- *virtual uint8_t **getMemberSize** () const*: Επιστρέφει το μέγεθος των μελών ενός εφέ στην μορφή ακεραίου αριθμού με μέγιστη τιμή 255 (1 byte). Το *virtual* ορίζει πως η μέθοδος μπορεί να γίνει override από μια υποκλάση. Εξασφαλίζει πως δεν θα αλλαχθεί η τιμή.
- *virtual void **set** (float *values)=0*: Δέχεται έναν δείκτη από τον πίνακα με τιμές των παραμέτρων ενός εφέ και τις ορίζει στον πίνακα. Η προεπιλεγμένη τιμή είναι 0. Το virtual ορίζει πως η μέθοδος μπορεί να γίνει override από μια υποκλάση.
- *virtual void **get** (float *values) const =0*: Δέχεται έναν δείκτη από τον πίνακα με τιμές των παραμέτρων ενός εφέ και γεμίζει τον πίνακα με τις ονομασίες τους. Η προεπιλεγμένη τιμή είναι 0. Το virtual ορίζει πως η μέθοδος μπορεί να γίνει override από μια υποκλάση.
- *virtual void **process** (float *input, float *output, uint16_t length)*: Η μέθοδος που καλεί τις DSP μεθόδους για το αντίστοιχο εφέ. Σαν παραμέτρους δέχεται τους δύο δείκτες των πινάκων για την είσοδο και την έξοδο, μαζί με το μέγεθος της εισόδου. Το *virtual* ορίζει πως η μέθοδος μπορεί να γίνει override από μια υποκλάση.
- *virtual bool **isEnabled** () const*: Γυρνάει True ή False αναλόγως εάν το εφέ είναι ενεργό ή οχι.
- *virtual void **setEnabled** (bool state)*: Ενεργοποιεί ή απενεργοποιεί το εφέ.



Εικόνα 6.22: Απεικόνιση των προστατευμένων ιδιοτήτων της κλάσης *Pedal*.

Η κλάση Pedal περιέχει μερικές ιδιότητες που απαρτίζουν ένα πετάλι εφέ. Συγκεκριμένα:

- **type:** Υποδηλώνει τον τύπο του πεταλιού εφέ, χρησιμοποιώντας την *PedalType* enumeration κλάση η οποία περιέχει τους 6 τύπους πεταλιών εφέ:

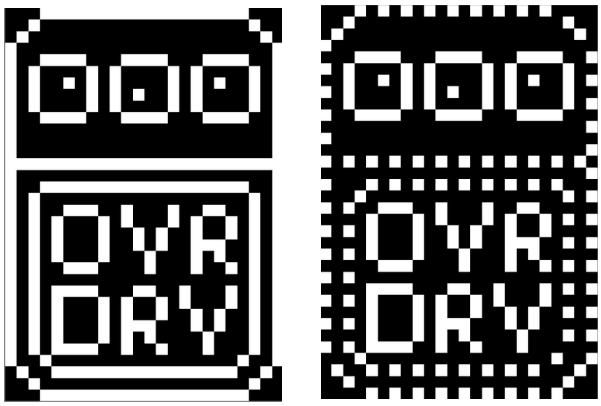
```
enum class PedalType {
    OVERDRIVE_DISTORTION,
    ECHO,
    REVERB,
    NOISE_GATE,
    COMPRESSOR,
    PASS_THROUGH
};
```

Εικόνα 6.23: Απεικόνιση της *PedalType* κλάσης.

- **image:** Ορίζει την εικόνα του ενεργοποιημένου πεταλιού που περιγράφεται από την κλάση *Bitmap*
- **image_disabled:** Ορίζει την εικόνα του απενεργοποιημένου πεταλιού που περιγράφεται από την κλάση *Bitmap*.
- **name:** Το όνομα του πεταλιού εφέ σε μορφή σειράς χαρακτήρων, χρησιμεύει για την εμφάνιση του ονόματος του στην διεπαφή.
- **enabled:** Boolean μεταβλητή που δηλώνει την ενεργή κατάσταση του πεταλιού.

Η κλάση *Pedal* κληρονομείται από όλα τα πετάλια εφέ. Το κάθε πετάλι χρησιμοποιεί τις δικές του μοναδικές παραμέτρους που το απαρτίζουν. Αρχικά , η κλάση *DistortionPedal* αποτελείται από 3 μοναδικές παραμέτρους.

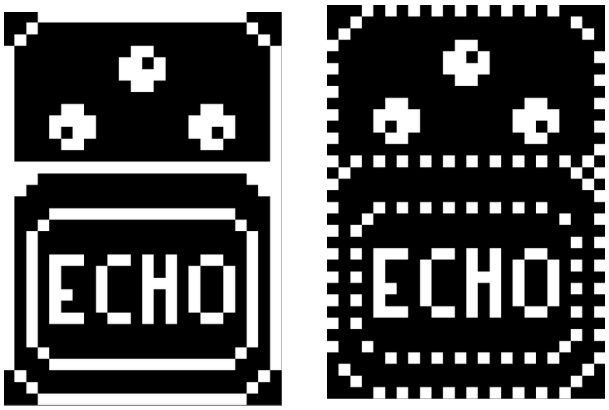
- **Gain:** Ορίζει την ένταση παραμόρφωσης.
- **Tone:** Ορίζει την χροιά του σήματος, όσο πιο κοντά στο 0 τόσο πιο πολύ απορρίπτονται υψηλές συχνότητες
- **Level:** Ορίζει την ένταση του συνολικού σήματος εξόδου, χωρίς να το παραμορφώσει, όπως το *volume*.



Εικόνα 6.24: Απεικόνιση των bitmap του παραμορφωτή στην ενεργή και ανενεργή κατάσταση.

Η κλάση EchoPedal αποτελείται από 4 μοναδικές παραμέτρους:

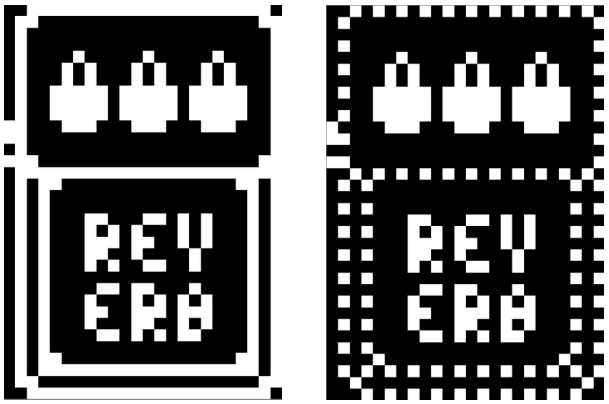
- **DelayTime:** Ορίζει τον χρόνο που θα μεσολαβήσει για την επόμενη ηχώ, κανονικοποιημένος στο εύρος 0-1
- **Feedback:** Ορίζει την ένταση ανατροφοδότησης.
- **Mix:** Ορίζει την ένταση του dry σήματος έναντι του wet. Όσο πιο κοντά στο 0 τόσο πιο πολύ ένταση προσδίδει στο dry.
- **Damping:** Ορίζει την απόσβεση.



Εικόνα 6.25: Απεικόνιση των bitmap της ηχώ στην ενεργή και ανενεργή κατάσταση.

Η κλάση ReverbPedal αποτελείται από 3 μοναδικές παραμέτρους:

- **RoomSize:** Ορίζει το μέγεθος του δωματίου, που αντιστοιχεί στο feedback
- **Mix:** Ορίζει την ένταση του dry σήματος έναντι του wet. Όσο πιο κοντά στο 0 τόσο πιο πολύ ένταση προσδίδει στο dry.
- **Damping:** Ορίζει την απόσβεση.

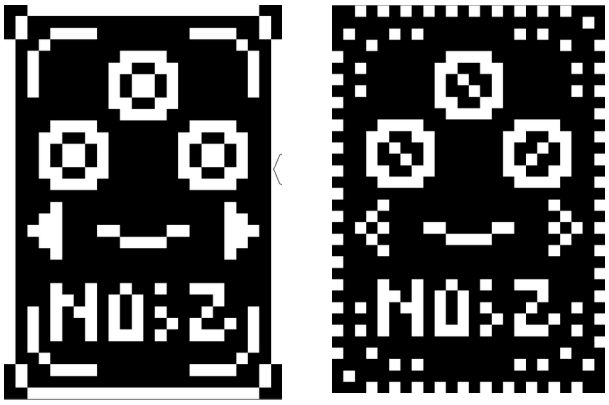


Εικόνα 6.26: Απεικόνιση των bitmap της αντήχησης στην ενεργή και ανενεργή κατάσταση.

Η κλάση NoiseGatePedal αποτελείται από 4 μοναδικές παραμέτρους:

- **Threshold:** Ορίζει το κατώφλι αποκοπής.
- **Attack:** Ορίζει τον χρόνο εκκίνησης του καταστολέα θορύβου, κανονικοποιημένος στο εύρος 0 – 1.

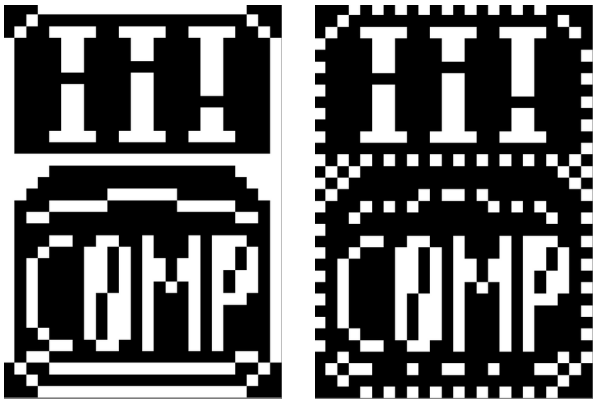
- **Hold:** Ορίζει τον χρόνο συγκράτησης του καταστολέα θορύβου, κανονικοποιημένος στο εύρος 0 – 1.
- **Release:** Ορίζει τον χρόνο απελευθέρωσης του καταστολέα θορύβου, κανονικοποιημένος στο εύρος 0 – 1.



Εικόνα 6.27: Απεικόνιση των bitmap του καταστολέα θορύβου στην ενεργή και ανενεργή κατάσταση.

Η κλάση CompressorPedal αποτελείται από 5 μοναδικές παραμέτρους:

- **Threshold:** Ορίζει το κατώφλι αποκοπής.
- **Ratio:** Ορίζει το επίπεδο συμπίεσης.
- **MakeUpGain:** Ορίζει την ένταση του σήματος εξόδου.
- **Attack:** Ορίζει τον χρόνο εκκίνησης του καταστολέα θορύβου, κανονικοποιημένος στο εύρος 0 – 1.
- **Release:** Ορίζει τον χρόνο απελευθέρωσης του καταστολέα θορύβου, κανονικοποιημένος στο εύρος 0 – 1.

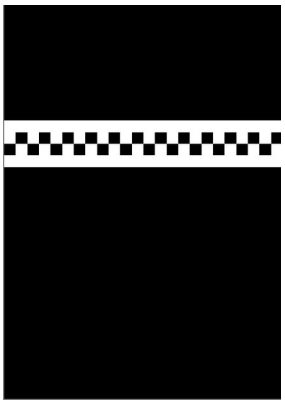


Εικόνα 6.28: Απεικόνιση των bitmap του κομπρέσσορα στην ενεργή και ανενεργή κατάσταση.

Η κλάση PassthroughPedal αποτελείται από 3 μοναδικές παραμέτρους:

- **Highs:** Ορίζει την ένταση των πρίμων συχνοτήτων.

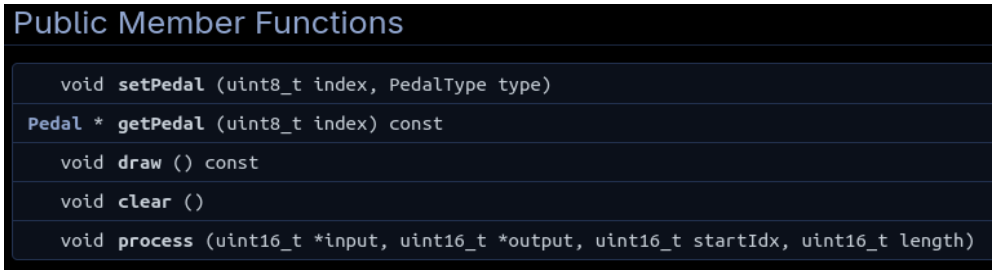
- **Mids:** Ορίζει την ένταση των μεσαίων συχνοτήτων.
- **Lows:** Ορίζει την ένταση των μπάσων συχνοτήτων.



Εικόνα 6.29: Απεικόνιση του bitmap της διέλευσης. Δεν υπάρχει ανενεργή κατάσταση!

6.6.2 Effects Chain

Η κλάση *EffectsChain* ορίζει την οντότητα που λειτουργεί ως ένας πίνακας 4 θέσεων για την τοποθέτηση μέχρι και τεσσάρων εφέ. Η σειρά εφαρμογής των εφέ επηρεάζει τον τελικό ήχο. Η οντότητα αυτή εμφανίζεται στην αρχική οθόνη της διεπαφής είτε κενή (4 passthrough πετάλια) εάν δεν υπάρχει αποθηκευμένη πεταλιέρα στην μνήμη είτε εμφανίζει την αποθηκευμένη πεταλιέρα.



Εικόνα 6.30: Απεικόνιση των κοινών δημόσιων συναρτήσεων της κλάσης *EffectsChain*.

- *void **setPedal** (uint8_t index, PedalType type)*: Ορίζει σε μια συγκεκριμένη θέση (*index)* ένα πετάλι συγκεκριμένου τύπου (*PedalType)*.
- *Pedal * **getPedal** (uint8_t index) const*: Επιστρέφει τον δείκτη από το πετάλι που περιγράφεται από την κλάση *Pedal* για την συγκεκριμένη θέση που δόθηκε. Εξασφαλίζει πως δεν θα αλλαχθούν τα δεδομένα της κλάσης *Pedal*.
- *void **draw** () const*: Αξιοποιεί την βιβλιοθήκη *Display* για τον γραφικό σχεδιασμό της πεταλιέρας. Εξασφαλίζει πως δεν θα αλλαχθούν τα δεδομένα της κλάσης *EffectsChain*.
- *void **clear** ()*: Τροποποιεί τα δεδομένα της κλάσης *EffectsChain*, διαγράφοντας τα προηγούμενα πετάλια και δημιουργώντας μια κενή (*passthrough)* πεταλιέρα.
- *void **process** (uint16_t *input, uint16_t *output, uint16_t startIdx, uint16_t length)*: Συνάρτηση επεξεργασίας σήματος, η οποία δέχεται σαν παραμέτρους τους πίνακες εισόδου και εξόδου, το σημείο του πίνακα από το οποίο θα αρχίσει να διαβάζει και το συνολικό μέγεθος. Το *startIdx*

είναι απαραίτητο καθώς το ADC είναι έτσι ρυθμισμένο ώστε να επεξεργάζεται τον μισό πίνακα ανά φορά. Η λειτουργία του θα επεξηγηθεί περαιτέρω παρακάτω.

Εικόνα 6.31: Απεικόνιση των δημοσίων ιδιοτήτων της κλάσης *EffectsChain*.

- **selectedPedal:** Δηλώνει το τρέχον επιλεγμένο πετάλι εφέ. Αρχική τιμή 0.

6.7 Βιβλιοθήκη Περιφερειακών

Η βιβλιοθήκη των περιφερειακών ενσωματώνει όλες τις ρουτίνες που είναι υπεύθυνες για τη ρύθμιση και τον έλεγχο του υλικού (hardware) κατά την εκκίνηση του συστήματος. Για το σκοπό αυτό, χρησιμοποιεί τη βιβλιοθήκη HAL (Hardware Abstraction Layer) της STMicroelectronics, η οποία παρέχει μια ομοιόμορφη διεπαφή για όλο το υλικό των STM32. Η βιβλιοθήκη καλείται από το κύριο πρόγραμμα και αρχικοποιεί το ADC, DAC, ρολόι χρονισμού, DMA, GPIOs, MPU, SPI και τους χρονιστές.

- **ADC1:** Αρμόδιο για την μετατροπή του αναλογικού σήματος της κιθάρας σε ψηφιακό για την επεξεργασία του. Αρχικοποιείται με ανάλυση 12-bit (0-4095) και αξιοποιεί τον Timer 6 για χρονισμό, ο οποίος είναι ρυθμισμένος στα 96kHz. Χρησιμοποιεί 1 κανάλι ως είσοδο και τα δεδομένα στέλνονται στην κύρια μνήμη μέσω Direct Memory Access (DMA). Η δειγματοληψία γίνεται με 8.5 κύκλους με διαίρεση ρολογιού κατά DIV2.
- **ADC2:** Χρησιμοποιείται για να διαβάσει τις τιμές των ποτενσιόμετρων. Αρχικοποιείται με ανάλυση 8-bit (0-255) και ξεκινάει με εντολή από το λογισμικό (software trigger), έναντι του χρονιστή. Χρησιμοποιεί 1 κανάλι ως είσοδο και τα δεδομένα διαβάζονται απευθείας από τον καταχωρητή δεδομένων (DR). Η δειγματοληψία γίνεται με 8.5 κύκλους με διαίρεση ρολογιού κατά DIV4.
- **DAC:** Μετατρέπει το επεξεργασμένο σήμα σε αναλογικό και το στέλνει στο κύκλωμα εξόδου. Ομοίως με το ADC1, αρχικοποιείται με ανάλυση 12-bit (0-4095) και αξιοποιεί τον Timer 6 για χρονισμό, ο οποίος είναι ρυθμισμένος στα 96kHz. Τα δεδομένα μεταφέρονται στο DAC από την κύρια μνήμη μέσω Direct Memory Access (DMA).
- **Clock:** Ορίζει τα ρολόγια του κύριου συστήματος και των διαφόρων περιφερειακών. Συγκεκριμένα το ρολόι του κύριου επεξεργαστή είναι ρυθμισμένο στα 480MHz. Η διαίρεση της συχνότητας επιτυγχάνεται μέσω του συστήματος ελέγχου Phase-Locked Loop (PLL).
- **DMA:** Αρμόδιο για την μεταφορά δεδομένων από περιφερειακά στην μνήμη (peripheral to memory) ή από την μνήμη σε περιφερειακά (memory to peripheral). Η χρήση αυτού του υλικού επιτυγχάνει την άμεση μεταφορά δεδομένων χωρίς την παρέμβαση του επεξεργαστή, ελευθερώνοντας πόρους και κύκλους επεξεργαστή.
- **GPIO:** Αρχικοποιεί όλους τους ακροδέκτες εισόδου και εξόδου του συστήματος.

- **MPU:** Ρυθμίζει το Memory Protection Unit (MPU) που είναι υπεύθυνο για την πρόσβαση συγκεκριμένων θέσεων μνήμης.
- **SPI:** Ορίζει τον τρόπο λειτουργίας του πρωτοκόλλου SPI για την επικοινωνία με την οθόνη. Ο STM32 λειτουργεί ως ο host ενώ η οθόνη ως client, καθώς δεν στέλνει δεδομένα πίσω στον μικροελεγκτή αλλά μόνο δέχεται.

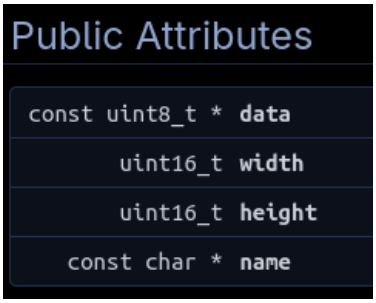
Επιπροσθέτως, η κύρια μέθοδος που είναι αρμόδια για την κλήση των παραπάνων μεθόδων αρχικοποίησης υλικού, τελεί την επιπλέον λειτουργία της βαθμονόμησης των υλικών ADC1, ADC2 και DAC μέσω εργαλείων του HAL.

6.8 Βιβλιοθήκη Βοηθητικών Αρχείων

Η βιβλιοθήκη των βοηθητικών αρχείων περιλαμβάνει όλα τα αρχεία και μέρη κώδικα που χρησιμοποιούνται από τις υπόλοιπες βιβλιοθήκες.

6.8.1 Bitmaps

Η βιβλιοθήκη των bitmaps περιέχει όλες τις εικόνες της διεπαφής υπό την κλάση Bitmap. Δεν περιέχει καμία μέθοδο παρά μόνο τα αντικείμενα Bitmap.



Εικόνα 6.32: Απεικόνιση των δημοσίων ιδιοτήτων της κλάσης *Bitmap*.

Η κλάση *Bitmap* περιέχει όλες τις ιδιότητες που περιγράφουν ένα αντικείμενο *Bitmap* το οποίο μπορεί να χρησιμοποιηθεί σε συνδυασμό με την βιβλιοθήκη της οθόνης για την απεικόνισή του.

- **data:** Περιέχει τον δείκτη για τον 8bit πίνακα που περιέχει τα στοιχεία της bitmap εικόνας σε δεκαεξαδική μορφή.
- **width:** Ορίζει το μήκος του bitmap.
- **height:** Ορίζει το ύψος του bitmap.
- **name:** Ορίζει το όνομα του bitmap καθώς πολλές φορές εμφανίζεται στην οθόνη μαζί με το bitmap.

```
const uint8_t mod_pedal_val_7[MOD_BAR_SIZE]
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x80, 0x40, 0x40, 0x40, 0x40,
0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40,
    0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40,
0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40,
    0x40, 0x80, 0x00, 0x07, 0x08, 0x12, 0x17, 0x10,
0x17, 0x17, 0x10, 0x17, 0x17, 0x10, 0x17, 0x57,
    0xd0, 0xd7, 0xd7, 0xd0, 0xd7, 0xd7, 0x50, 0x17,
0x17, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10,
    0x10, 0x10, 0x08, 0x07, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x01, 0x03, 0x03, 0x01, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00
};
```

Εικόνα 6.33: Παράδειγμα 33x36 bitmap και ο αντίστοιχος uint8_t πίνακας του.

6.8.2 Display

Η βιβλιοθήκη Display παρέχει όλες τις δυνατότητες και λειτουργίες για την οδήγηση της οθόνης SSD1309 μέσω SPI. Ορίζει τους ακροδέκτες του SPI και υλοποιεί όλες τις μεθόδους που χρειάζονται για την εμφάνιση των στοιχείων στην οθόνη.

- *void **init**()*: Αρχικοποιεί την οθόνη SSD1309. Ρυθμίζει τις παραμέτρους λειτουργίας γράφοντας απευθείας πάνω στους registers και ενεργοποιεί την οθόνη.

- *void **setBrightness**(uint8_t level)*: Ρυθμίζει τη φωτεινότητα της οθόνης. Δέχεται μια τιμή από 0 έως 255 και ελέγχει την αντίθεση του display μέσω του κατάλληλου command.

- *void **writeCommand**(uint8_t cmd)*: Στέλνει μία εντολή στην οθόνη μέσω SPI. Χρησιμοποιείται για εσωτερικές λειτουργίες (π.χ. ρύθμιση column address, page address, κ.λπ.) και καλείται από άλλες μεθόδους.

- *void **writeData**(uint8_t* data, uint16_t size)*: Στέλνει δεδομένα εικόνας bitmap στην οθόνη. Δέχεται τον δείκτη ενός πίνακα και το μέγεθος του, και τα μεταφέρει στην οθόνη μέσω SPI.

- *void **reset**()*: Πραγματοποιεί hardware reset της οθόνης. Ενεργοποιεί τον αντίστοιχο pin reset για συγκεκριμένο χρόνο, επαναφέροντας την οθόνη στην αρχική της κατάσταση.

- *void **setCursor**(uint8_t column, uint8_t page)*: Θέτει τη θέση του κέρσορα για την επόμενη εγγραφή κειμένου. Η column ορίζει την οριζόντια θέση (0-127) και η page ορίζει την κάθετη σελίδα (0-7, όπου κάθε σελίδα έχει 8 pixels ύψος).

- *void **drawChar**(char c, uint8_t x, uint8_t page)*: Σχεδιάζει έναν χαρακτήρα στη θέση (x, page). Χρησιμοποιεί ενσωματωμένη γραμματοσειρά 5x7 ή 6x8 pixels και ενημερώνει το frameBuffer.

- *void **drawString**(const char* str, uint8_t x, uint8_t page)*: Σχεδιάζει μια συμβολοσειρά χαρακτήρων ξεκινώντας από τη θέση (x, page). Καλεί επαναληπτικά την drawChar για κάθε χαρακτήρα και προχωράει τον κέρσορα.

- *void **drawDigit**(uint8_t digit, uint8_t x, uint8_t page)*: Σχεδιάζει ένα ψηφίο (0-9) στη θέση (x, page).

- *void **drawFloat**(float value, uint8_t x, uint8_t page)*: Σχεδιάζει έναν δεκαδικό αριθμό (float) στη θέση (x, page). Μετατρέπει τον αριθμό σε συμβολοσειρά με συγκεκριμένη μορφοποίηση (π.χ. δύο δεκαδικά ψηφία) και τον εμφανίζει.
- *void **drawBitmap**(const Bitmap& bmp, uint8_t x, uint8_t pageStart)*: Σχεδιάζει μια εικόνα bitmap στη θέση (x, pageStart). Τα bitmap χρησιμοποιούνται για τα γραφικά των πεταλιών και αντιγράφονται στο frameBuffer στη συγκεκριμένη θέση.
- *void **clear**()*: Καθαρίζει ολόκληρη την οθόνη. Μηδενίζει το frameBuffer και στέλνει εντολή στην οθόνη για να σβήσει όλα τα pixels.
- *void **printf**(const char* format, ...)*: Εμφανίζει μορφοποιημένο κείμενο στην τρέχουσα θέση του κέρσορα. Λειτουργεί όπως η συνάρτηση printf της C, υποστηρίζοντας τους γνωστούς specifiers (%d, %f, %x, %s, κ.λπ.) για εύκολη εκτύπωση τιμών.
- *void **printf**(uint8_t x, uint8_t page, const char* format, ...)*: Εμφανίζει μορφοποιημένο κείμενο σε συγκεκριμένη θέση (x, page). Συνδυάζει την setCursor με την printf, επιτρέποντας τ.ην άμεση εκτύπωση στη θέση που θέλουμε.
- *extern uint8_t **frameBuffer**[DISPLAY]*: Εξωτερικός πίνακας που αποθηκεύει την τρέχουσα εικόνα της οθόνης στην προσωρινή μνήμη. Το μέγεθός του είναι DISPLAY (128x64/8 = 1024 bytes). Κάθε bit αντιστοιχεί σε ένα pixel (1 = αναμμένο, 0 = σβηστό). Η οθόνη ενημερώνεται αντιγράφοντας το frameBuffer μέσω της writeData.

Η οθόνη χρησιμοποιεί το πρωτόκολλο SPI με την χρήση 7 συνδέσεων.

- VCC: Η τροφοδοσία της οθόνης ανέρχεται στα 3.3V DC.
- GND: Η γείωση.
- SCK: Το ρολόι συγχρονισμού κατά την μεταφορά δεδομένων.
- SDA: Η σύνδεση που μεταφέρει τα δεδομένα.
- CS: Chip Select ή αλλιώς ο επιλογέας περιφερειακού. Στην συγκεκριμένη περίπτωση η οθόνη είναι το μοναδικό περιφερειακό.
- RST: Reset δηλαδή επαναφέρει την αρχική λειτουργία της οθόνης.
- DC: Data/Command που αναφέρει εάν τα δεδομένα που έρχονται είναι κάποια εντολή ή φορτίο για εμφάνιση.

6.8.3 QSPI Flash

Η QSPI Flash πρόκειται για μια εξωτερική μνήμη flash μεγέθους 8MB από την Winbond με κωδικό W25Q64JV και χρησιμοποιείται για την αποθήκευση των πεταλιέρων. Με αυτόν τον τρόπο απελευθερώνεται η κύρια flash μνήμη που έχει μέγεθος 2MB.

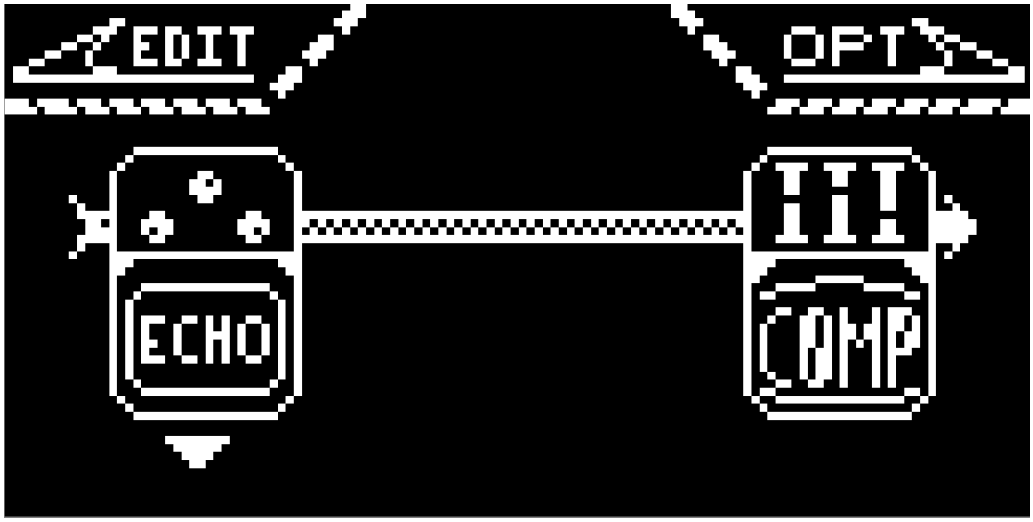
- *bool **is_ready()***: Ελέγχει αν η QSPI Flash είναι έτοιμη να δεχθεί νέα εντολή. Επιστρέφει true όταν η μνήμη δεν εκτελεί κάποια λειτουργία (εγγραφή, σβήσιμο) και false αν είναι απασχολημένη.
- *HAL_StatusTypeDef **init()***: Αρχικοποιεί την QSPI Flash εξωτερική μνήμη. Ρυθμίζει το πρωτόκολλο επικοινωνίας QSPI, ελέγχει την αναγνώριση της μνήμης (device ID) και την προετοιμάζει για λειτουργίες ανάγνωσης, εγγραφής και διαγραφής.

- *HAL_StatusTypeDef* ***read****(uint32_t address, uint8_t data, uint32_t size)*: Διαβάζει δεδομένα από την QSPI Flash. Δέχεται την διεύθυνση (address), δείκτη σε buffer αποθήκευσης (data) και το μέγεθος (size). Αντιγράφει τα δεδομένα από την εξωτερική μνήμη στον εσωτερικό buffer.
- *HAL_StatusTypeDef* ***write****(uint32_t address, const uint8_t* data, uint32_t size): Εγγράφει δεδομένα στην QSPI Flash. Δέχεται την διεύθυνση (address), δείκτη στα δεδομένα προς εγγραφή (data) και το μέγεθος (size). Πριν την εγγραφή η περιοχή πρέπει να είναι σβησμένη.
- *HAL_StatusTypeDef* ***erase_sector****(uint32_t address)*: Σβήνει έναν τομέα (sector) της QSPI Flash στη διεύθυνση που δόθηκε. Το σβήσιμο μετατρέπει όλα τα bits της περιοχής σε 1 (0xFF), απαραίτητο πριν από κάθε εγγραφή.
- *HAL_StatusTypeDef* ***erase_chip****()*: Σβήνει ολόκληρη την QSPI Flash (chip erase). Διαγράφει όλα τα δεδομένα από την εξωτερική μνήμη των 8MB.
- *HAL_StatusTypeDef* ***loadEffectsChain****(EffectsChain* chain): Φορτώνει μια αποθηκευμένη πεταλιέρα (EffectsChain) από την QSPI Flash στη μνήμη RAM. Διαβάζει τα αποθηκευμένα δεδομένα από την εξωτερική μνήμη και ανακατασκευάζει την σειρά εφέ με όλες τις παραμέτρους τους.
- *HAL_StatusTypeDef* ***saveEffectsChain****(const EffectsChain* chain): Αποθηκεύει την τρέχουσα πεταλιέρα (EffectsChain) στην QSPI Flash. Γράφει όλα τα δεδομένα των πεταλιών (τύπος, παράμετροι, σειρά) στην εξωτερική μνήμη ώστε να διατηρηθούν μετά από επανεκκίνηση του συστήματος.

6.9 Κύρια Ροή Προγράμματος

6.9.1 Γραφική Διεπαφή

Κατά την εκκίνηση του συστήματος, το κύριο πρόγραμμα εκτελεί πρώτα όλες τις απαραίτητες αρχικοποιήσεις των υλικών περιφερειακών. Αυτό περιλαμβάνει την αρχικοποίηση της QSPI Flash μνήμης, η οποία αποτελεί την εξωτερική μνήμη αποθήκευσης των πεταλιέρων, καθώς και την προετοιμασία της οθόνης, των κουμπιών, του ADC και του DAC. Μόλις ολοκληρωθούν οι αρχικοποιήσεις, το σύστημα επιχειρεί να φορτώσει μια αποθηκευμένη αλυσίδα εφέ από την QSPI Flash. Σε περίπτωση που η φόρτωση αποτύχει, είτε επειδή το σύστημα εκκινεί για πρώτη φορά είτε επειδή τα αποθηκευμένα δεδομένα έχουν καταστραφεί, το πρόγραμμα δημιουργεί αυτόματα μια προεπιλεγμένη πεταλιέρα. Αυτή η προεπιλεγμένη πεταλιέρα περιλαμβάνει τέσσερα πετάλια τύπου PASS_THROUGH, τα οποία δεν μεταβάλλουν καθόλου τον ήχο. Στη συνέχεια, το πρόγραμμα προχωρά στην αρχικοποίηση των γραφικών της οθόνης. Σχεδιάζει τη βασική διάταξη της πεταλιέρας και στη συνέχεια καλεί τη μέθοδο *chain.draw()* για να εμφανίσει στην οθόνη τα τέσσερα πετάλια με τις αντίστοιχες εικόνες τους.

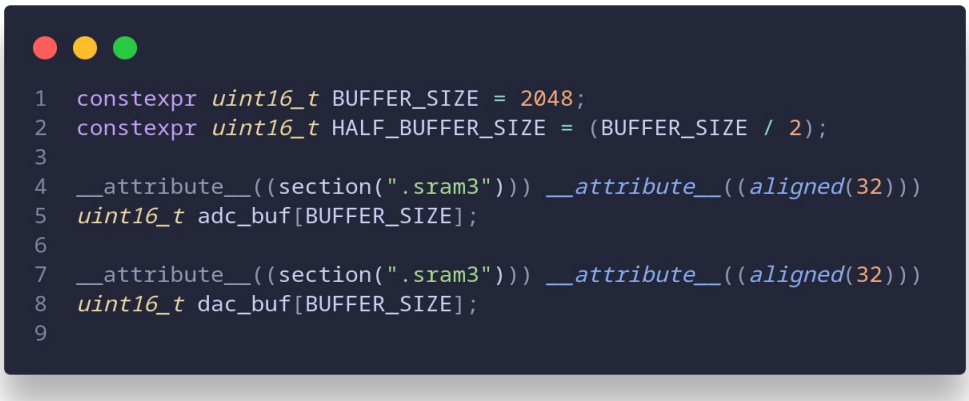


Εικόνα 6.33: Παράδειγμα αρχικής οθόνης με την αλυσίδα εφέ.

Η αλληλεπίδραση του χρήστη με το σύστημα πραγματοποιείται μέσω εξωτερικών διακοπών (GPIO interrupts) που ενεργοποιούνται κάθε φορά που πατιέται ένα φυσικό κουμπί. Όπως περιγράφθηκε από την αρχιτεκτονική της διεπαφής, η συσκευή διαθέτει κυκλικό κουμπί για την επιλογή μενού της πεταλιέρας, κουμπιά για την επιλογή μεμονωμένων πεταλιών (SELECT_PEDAL_0 έως SELECT_PEDAL_3) και κουμπιά για την αλλαγή τύπου πεταλιού ή σελίδας (PEDAL_0, PEDAL_1). Ανάλογα με το ποιο κουμπί πατήθηκε και την τρέχουσα κατάσταση της εφαρμογής, η οποία μπορεί να είναι μιά από τις ακόλουθες: PEDALCHAIN_VIEW, PEDALSELECT_VIEW , PEDALEDIT_VIEW, ή SETTINGS_VIEW, το πρόγραμμα ανταποκρίνεται αναλόγως. Παράλληλα με την επεξεργασία του ήχου, ο χρονιστής TIM8 ενεργοποιείται περιοδικά και διαβάζει τις τιμές από τρία αναλογικά ποτενσιόμετρα χρησιμοποιώντας το ADC2. Αυτά τα ποτενσιόμετρα λειτουργούν ως φυσικά χειριστήρια που επιτρέπουν στο χρήστη να μεταβάλλει παραμέτρους των εφέ, όπως η ένταση, η συχνότητα ή οποιαδήποτε άλλη ρυθμιζόμενη παράμετρος. Οι τιμές αυτές αξιοποιούνται αποκλειστικά όταν το σύστημα βρίσκεται στην οθόνη επεξεργασίας (PEDALEDIT_VIEW), όπου ο χρήστης μπορεί να τροποποιεί δυναμικά τα χαρακτηριστικά κάθε πεταλιού.

6.9.2 Λειτουργία των ADC και DAC

Η επεξεργασία του ηχητικού σήματος πραγματοποιείται ασύγχρονα μέσω των συναρτήσεων callback που καλούνται από το ADC. Το σύστημα χρησιμοποιεί έναν κυκλικό πίνακα συνολικού μεγέθους 2048 δειγμάτων, ο οποίος είναι χωρισμένος σε δύο μισά των 1024 δειγμάτων το καθένα.



```
constexpr uint16_t BUFFER_SIZE = 2048;
constexpr uint16_t HALF_BUFFER_SIZE = (BUFFER_SIZE / 2);

__attribute__((section(".sram3"))) __attribute__((aligned(32)))
uint16_t adc_buf[BUFFER_SIZE];

__attribute__((section(".sram3"))) __attribute__((aligned(32)))
uint16_t dac_buf[BUFFER_SIZE];

```

Εικόνα 6.34: Κώδικας αρχικοποίησης των πινάκων ADC και DAC.

Κάθε φορά που το ADC συμπληρώνει το πρώτο μισό του πίνακα, καλείται το callback *ConvHalfCpltCallback*, ενώ όταν συμπληρώνεται ολόκληρο το buffer, καλείται το callback *ConvCpltCallback*. Αυτή είναι η αυτοματοποιημένη διαδικασία που τελείται μέσω των βιβλιοθηκών STM32 HAL. Μέσα σε αυτά τα callbacks, το σύστημα καλεί τη μέθοδο *chain.process()*, η οποία επεξεργάζεται το αντίστοιχο τμήμα του σήματος, δηλαδή είτε τα πρώτα 1024 δείγματα είτε τα επόμενα 1024 δείγματα. Η *chain.process()* εφαρμόζει διαδοχικά όλα τα ενεργά εφέ που υπάρχουν στην αλυσίδα, περνώντας το σήμα από το ένα πετάλι στο επόμενο. Τα επεξεργασμένα δεδομένα αποθηκεύονται στον buffer εξόδου που προορίζεται για το DAC. Επειδή ο μικροελεγκτής διαθέτει προσωρινή μνήμη cache, η οποία θα μπορούσε να προκαλέσει ασυνέπειες μεταξύ των δεδομένων στην cache και στην κύρια μνήμη, το πρόγραμμα καλεί την *SCB_CleanDCache_by_Addr* για να καθαρίσει την cache και να εξασφαλίσει ότι το DAC θα διαβάσει τα πιο πρόσφατα, σωστά δεδομένα μέσω DMA.

```
extern "C" void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef *hadc)
{
    if (hadc->Instance == ADC1)
    {
        chain.process(adc_buf, dac_buf, 0, HALF_BUFFER_SIZE);
        SCB_CleanDCache_by_Addr((uint32_t *)dac_buf,
                                HALF_BUFFER_SIZE * sizeof(uint16_t));
    }
}

extern "C" void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
{
    if (hadc->Instance == ADC1)
    {
        chain.process(adc_buf, dac_buf, HALF_BUFFER_SIZE, HALF_BUFFER_SIZE);
        SCB_CleanDCache_by_Addr((uint32_t *)&dac_buf[HALF_BUFFER_SIZE],
                                HALF_BUFFER_SIZE * sizeof(uint16_t));
    }
}
```

Εικόνα 6.35: Κώδικας *ConvHalfCpltCallback* και *ConvCpltCallback*.

6.10 Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκε η αρχιτεκτονική και η λειτουργία του λογισμικού που φορτώνεται επάνω στον STM32H743 μικροελεγκτή. Συνολικά δημιουργήθηκαν 20 διαφορετικά αρχεία εκτελέσιμου κώδικα και 20 διαφορετικά αρχεία που περιέχουν τις δηλώσεις των συμβόλων, πέρα από τις εξωτερικές βιβλιοθήκες HAL και LL της STM.

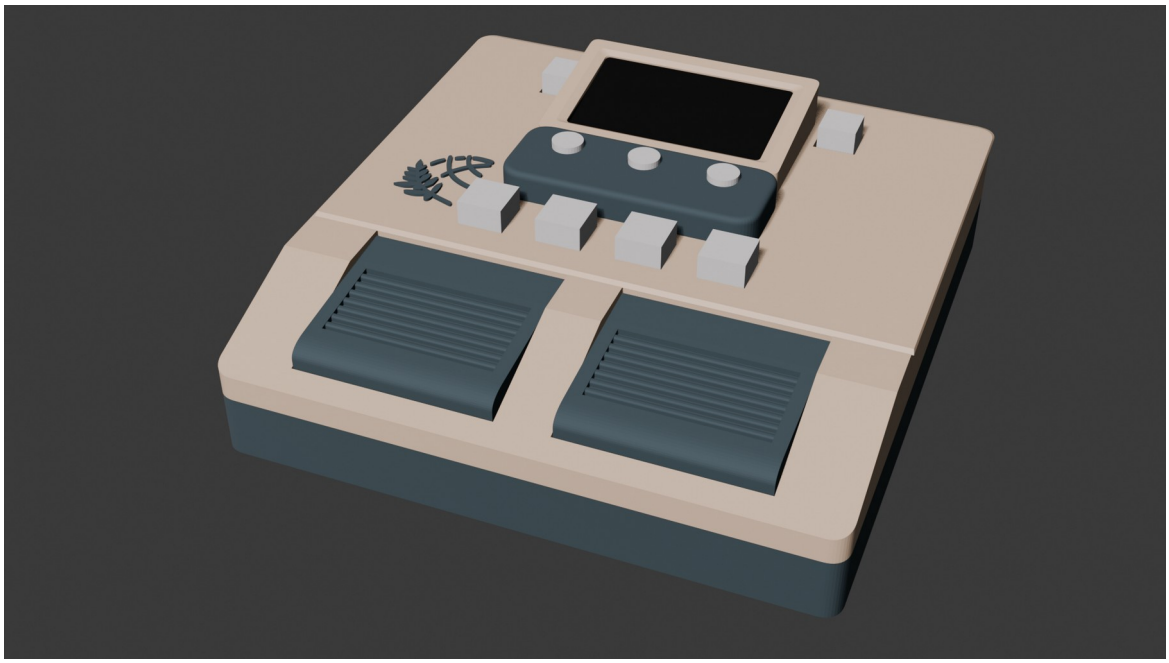
Κεφάλαιο 7ο: Περίβλημα Πολυεφέ

7.1 Εισαγωγή

Ως το τελικό στάδιο της διπλωματικής εργασίας, είναι απαραίτητη η σχεδίαση και κατασκευή ενός περιβλήματος για την στέγαση των εξαρτημάτων. Το περίβλημα προσφέρει επίσης εύκολη μεταφορά και τοποθέτηση του κυκλώματος, προστατεύοντας το από εξωτερικούς παράγοντες.

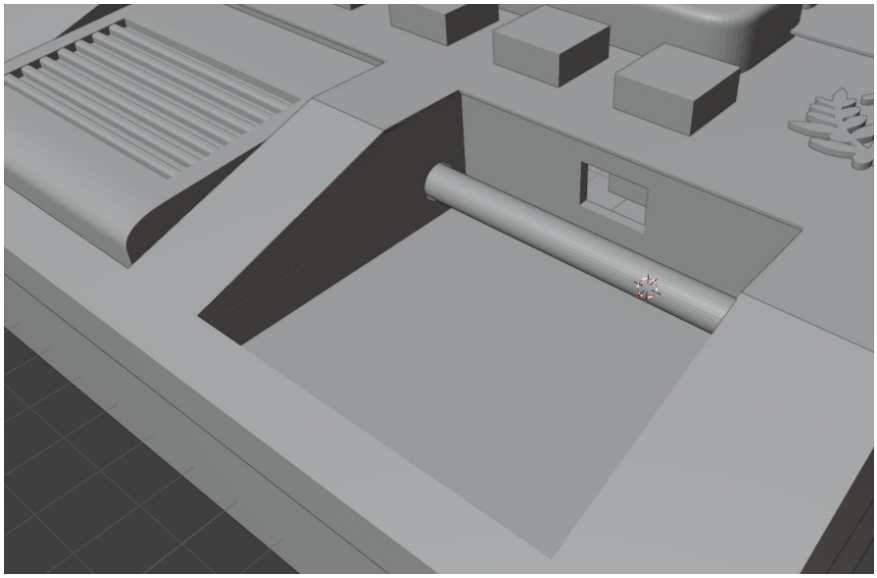
7.2 Σχεδίαση Περιβλήματος

Για την σχεδίαση και τρισδιάστατη αναπαράσταση του περιβλήματος, χρησιμοποιήθηκε το πρόγραμμα Blender, ένα ανοιχτό λογισμικό τρισδιάστατης σχεδίασης [55]. Σημαντικός παράγοντας της σχεδίασης είναι η άνεση του χρήστη καθώς θα χρησιμοποιεί την πεταλιέρα, για αυτό προστέθηκαν διάφορα κομμάτια που εξυπηρετούν αυτόν τον σκοπό. Αρχικά η οθόνη σχεδιάστηκε έτσι ώστε να έχει μια κλήση 10 μοιρών ώστε ο χρήστης να έρχεται σε άμεση οπτική επαφή με την οθόνη. Έπειτα, προστέθηκαν δύο μεγάλα πετάλια για την εύκολη μετάβαση σε άλλη πεταλιέρα. Υιοθετήθηκε μια “ρέτρο”αισθητική κατά την σχεδίαση που αναμιγνύει παστέλ απαλά χρώματα του μπεζ και του μπλε, με την προσθήκη του εικονιδίου του Διεθνούς Ελληνικού Πανεπιστημίου.



Εικόνα 7.1: Τρισδιάσταση αναπαράσταση του περιβλήματος μέσω Blender.

Τα πετάλια σχεδιάστηκαν για να κινούνται μηχανικά προς τα κάτω και να εφάπτονται με το κουμπί, με την χρήση ελατηρίων για να επανέρχονται στην αρχική τους θέση. Ο τρόπος σύνδεσης με το κύριο σώμα είναι ένας κύλινδρος που εισέρχεται κατά μήκος των πεταλιών και της πεταλιέρας.

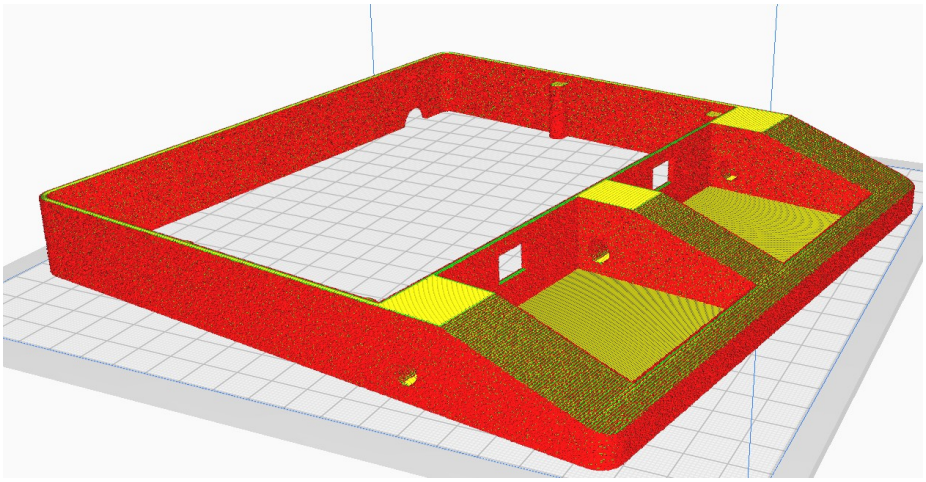


Εικόνα 7.2: Σχεδιασμός των πεταλιών στο Blender.

7.3 Υλοποίηση Περιβλήματος

Η υλοποίηση του περιβλήματος έγινε μέσω 3D εκτύπωσης, με τον 3D εκτυπωτή Creality Ender 3V3 KE και το υλικό polylactide **(**PLA). Για την δημιουργία των κομματιών διαφορετικού χρώματος και την ευκολότερη εκτύπωση, η κατασκευή χωρίστηκε σε πολλά κομμάτια τα οποία εξάχθηκαν ως .stl αρχεία. Η μετατροπή ενός τρισδιάστατου μοντέλου σε κώδικα κατανοητό από έναν 3d εκτυπωτή, γίνεται μέσω ενός προγράμματος slicer, το οποίο εξάγει την πληροφορία του τρισδιάστατου μοντέλου σε καρτεσιανές συντεταγμένες αλλά και τις ρυθμίσεις εκτύπωσης με την χρήση του προτύπου gcode. Το πρόγραμμα που χρησιμοποιήθηκε είναι το Cura slicer και οι ρυθμίσεις εκτύπωσης περιείχαν:

- Θερμοκρασία μύτης: 220 βαθμοί Κελσίου.
- Θερμοκρασία Βάσης: 60 βαθμοί Κελσίου.
- Ύψος Υποστρώματος: 0.2mm
- Ταχύτητα εκτύπωσης: 250mm/S.
- Ειδικές λειτουργίες: Fuzzy skin, χωρίς υποστήριξη.



Εικόνα 7.3: Παράδειγμα μετατροπής stl αρχείου σε gcode μέσω Cura.

7.4 Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκε η σχεδίαση του περιβλήματος μέσω του προγράμματος Blender για την τρισδιάστατη μοντελοποίηση του όπως και η υλοποίηση του μέσω τρισδιάστατης εκτύπωσης.

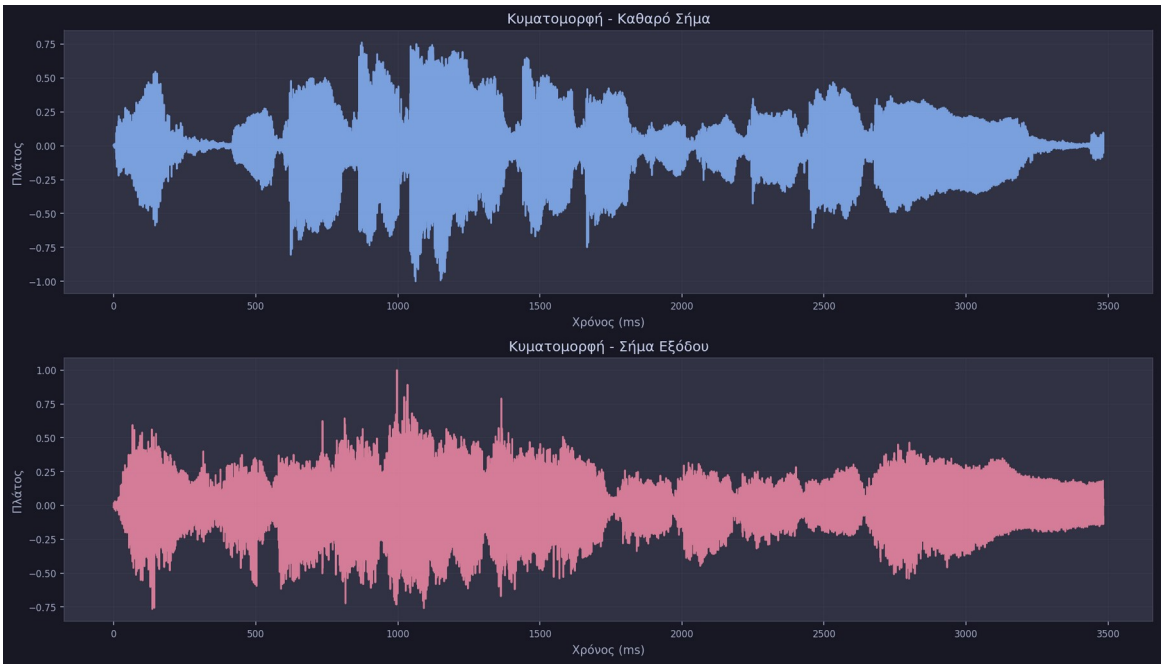
Κεφάλαιο 8ο: Πειραματικές Μετρήσεις

8.1 Εισαγωγή

Στα προηγούμενα κεφάλαια, παρουσιάστηκε το θεωρητικό υπόβαθρο των εικονικών αναλογικών μοντέλων εφέ, η ανάλυση του κυκλώματος και η ερμηνεία της λειτουργίας του λογισμικού. Στο παρόν κεφάλαιο, η διπλωματική εργασία εισέρχεται στο πειραματικό στάδιο, όπου τα αποτελέσματα διερευνώνται μέσω πραγματικών μετρήσεων. Οι μετρήσεις βασίζονται στον αυθεντικό ήχο που παράγει το πετάλι, με στόχο την πιστή ψηφιακή αποτύπωση των κυματομορφών και τη δημιουργία γραφικών παραστάσεων που αναδεικνύουν τη διαφορά μεταξύ σημάτων εισόδου και εξόδου, καθώς και τις κρουστικές αποκρίσεις και τα χρονικά διαγράμματά τους. Οι πραγματικές μετρήσεις του πεταλιού έγιναν με την χρήση του στούντιο ηχοληψίας του τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε. με την χρήση πυκνωτικών μικροφώνων που κατέγραφαν τον ήχο όπως βγαίνει από τον ενισχυτή της ηλεκτρικής κιθάρας.

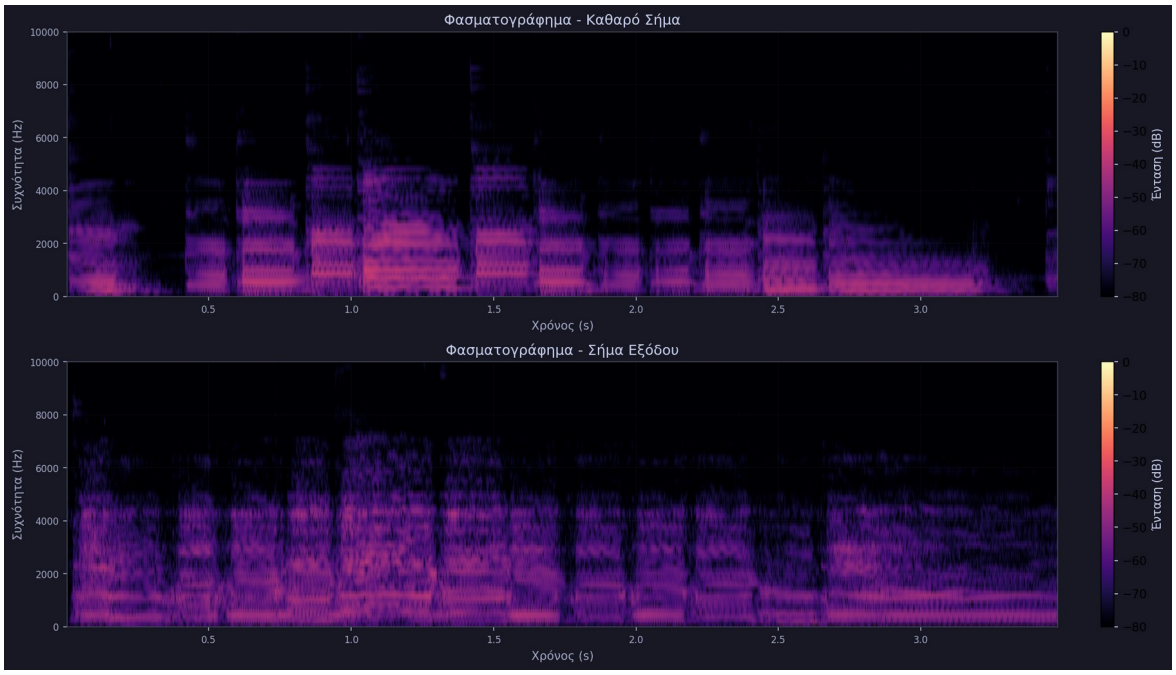
8.2 Overdrive Distortion

Για την δοκιμή του εφέ παραμόρφωσης χρησιμοποιήθηκε μια απλή πεντατονική κλίμακα σε διάρκεια 3.5 δευτερολέπτων. Οι παράμετροι που χρησιμοποιήθηκαν για την παραγωγή του εφέ ανέρχονται στην χρήση υψηλού gain και μερική απόσβεση χαμηλών συχνοτήτων.



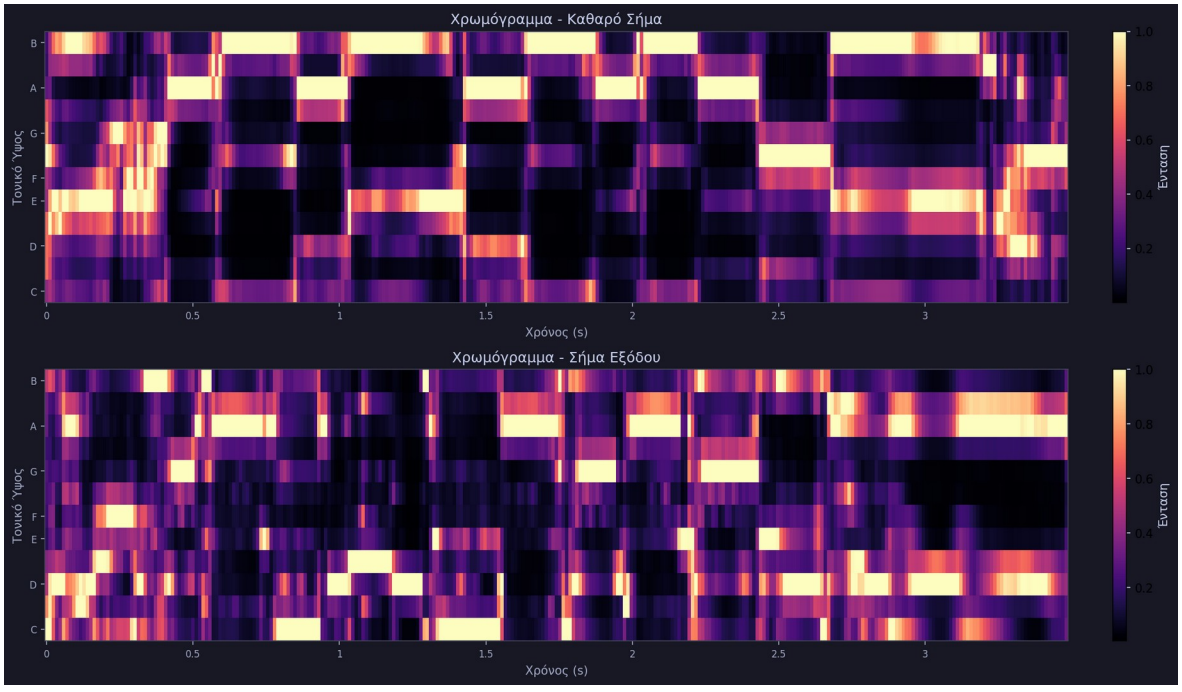
Εικόνα 8.1: Σύγκριση κυματομορφών καθαρού σήματος με σήματος παραμόρφωσης.

Όπως είναι εμφανές αρχικά στο παραμορφωμένο σήμα, δηλαδή στο σήμα εξόδου, υπάρχει παραπάνω θόρυβος.



Εικόνα 8.2: Σύγκριση φασματογραφημάτων καθαρού σήματος με σήματος παραμόρφωσης.

Φασματογράφημα είναι μια οπτική απεικόνιση του φάσματος συχνοτήτων σε συναρτήσει του χρόνου. Όσο πιο φωτεινό το χρώμα τόσο πιο μεγάλη η ένταση. Στο σήμα εξόδου φαίνεται πως η παραμόρφωση προσθέτει παραπάνω συχνότητες λόγω του θορύβου, ενώ παράλληλα εξομαλύνει χαμηλές συχνότητες λόγω του υψιπερατού φίλτρου.

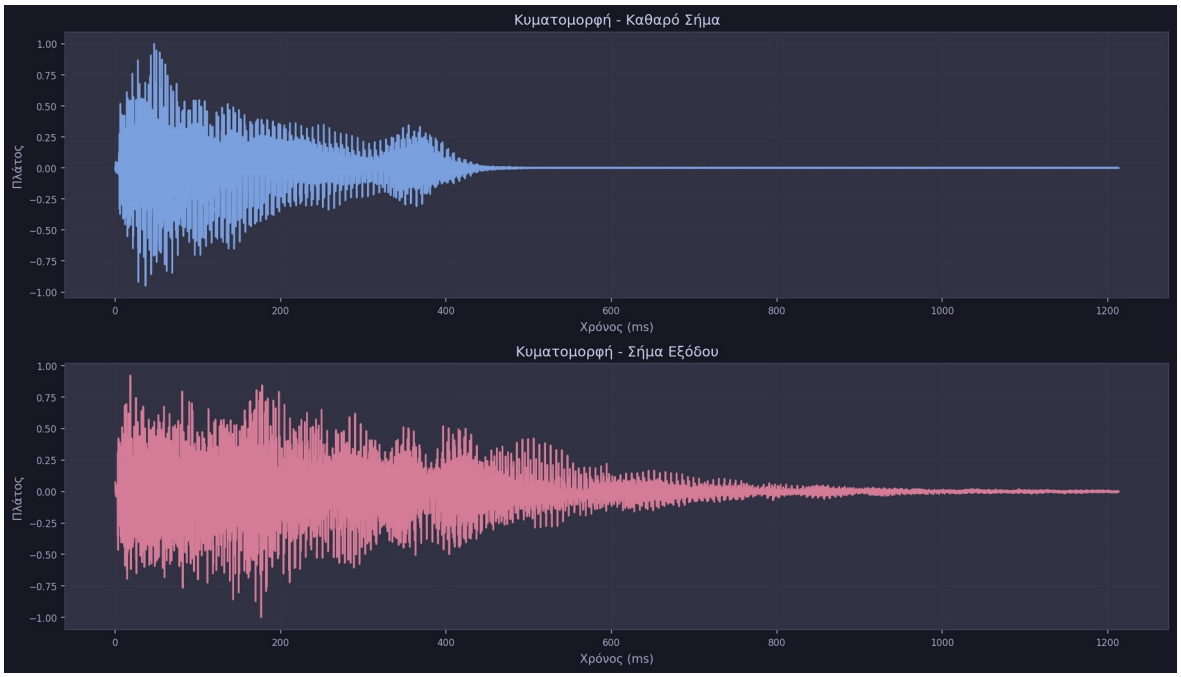


Εικόνα 8.3: Σύγκριση χρωμογραμμάτων καθαρού σήματος με σήματος παραμόρφωσης.

Το χρωμόγραμμα είναι η αντιστοίχηση των διαφόρων συχνοτήτων σε τονικό ύψος, δηλαδή μουσικές νότες. Όσο πιο φωτεινό το χρώμα τόσο πιο μεγάλη η ένταση της νότας. Όπως φαίνεται, ενισχύονται οι αρμονικές σε σχέση με την κύρια συχνότητα (νότα).

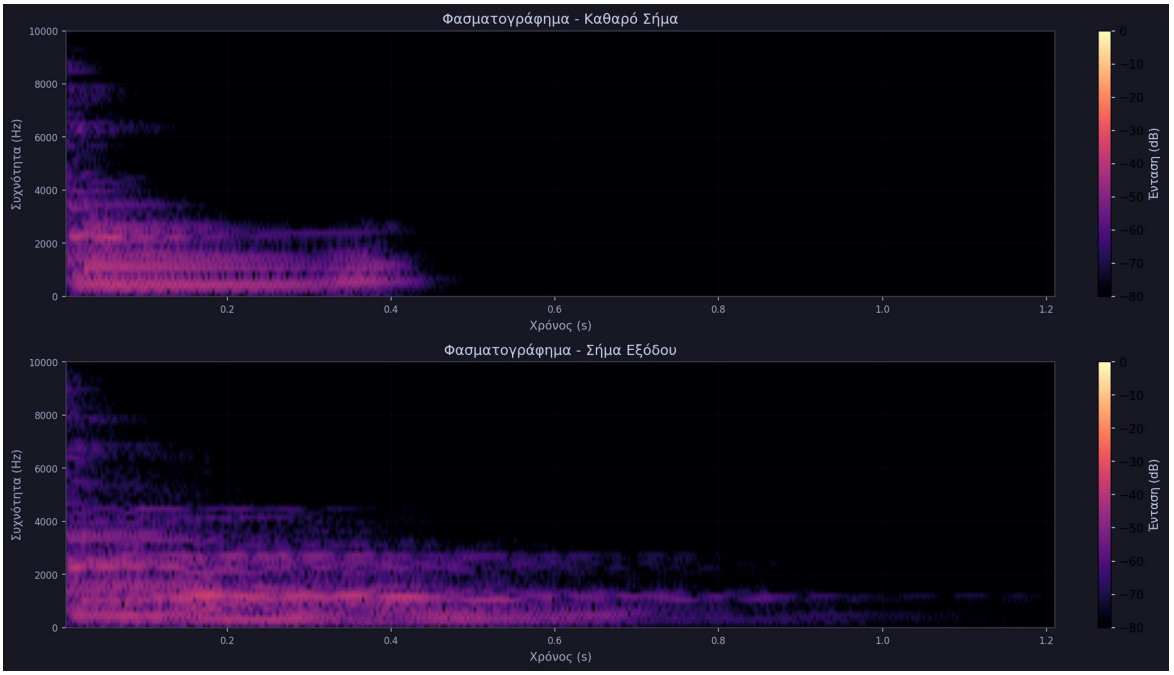
8.3 Reverb

Για την δοκιμή του εφέ αντήχησης χρησιμοποιήθηκε η νότα D3 (146.83 Hz) που αναπαράγεται σε standard E κούρδισμα με το παίξιμο της 2η χορδής (A2 110Hz) του 5ου τάστου. με διάρκεια 1 δευτερολέπτου μέχρι να εξασθενήσει. Οι παράμετροι που χρησιμοποιήθηκαν για την παραγωγή του εφέ ανέρχονται σε δύο εκδόσεις. Αρχικά την χρήση ενός ασθενούς reverb εφέ και έπειτα ενός ισχυρού με μεγάλη τιμή για το feedback και room size.



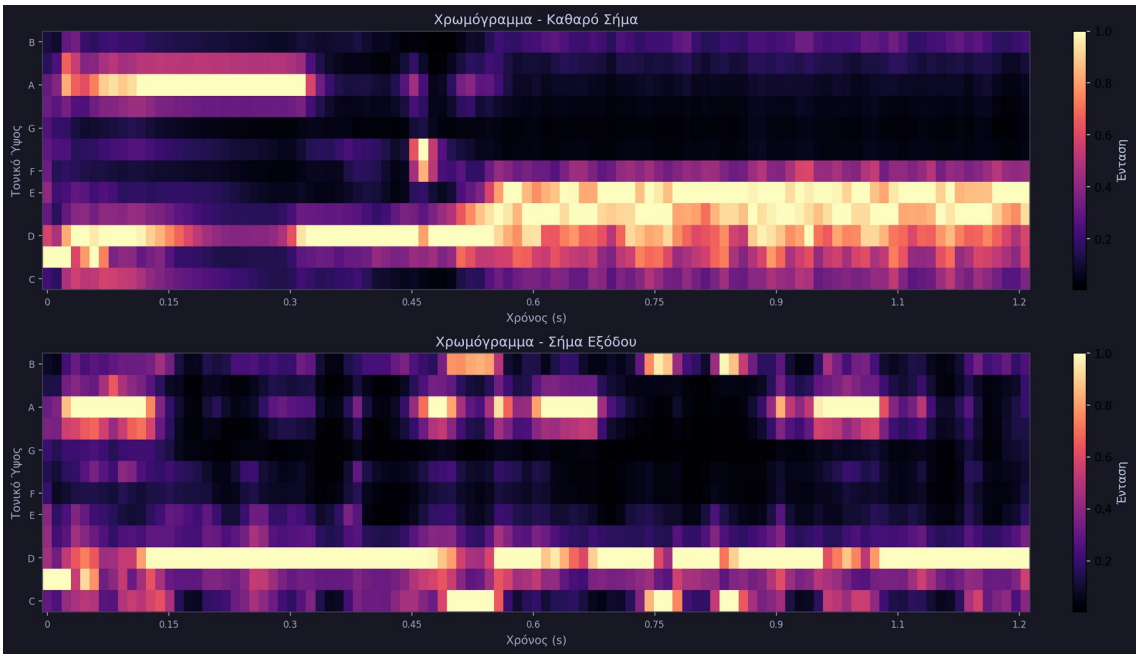
Εικόνα 8.4: Σύγκριση κυματομορφών καθαρού σήματος με σήματος ασθενούς αντήχησης.

Αρχικά είναι εμφανές πως το σήμα διατηρεί μια μεγαλύτερη ουρά απόσβεσης εφαρμόζοντας το εφέ ασθενούς αντήχησης, με μια ασθενής περιοδική διακύμανση του πλάτους.



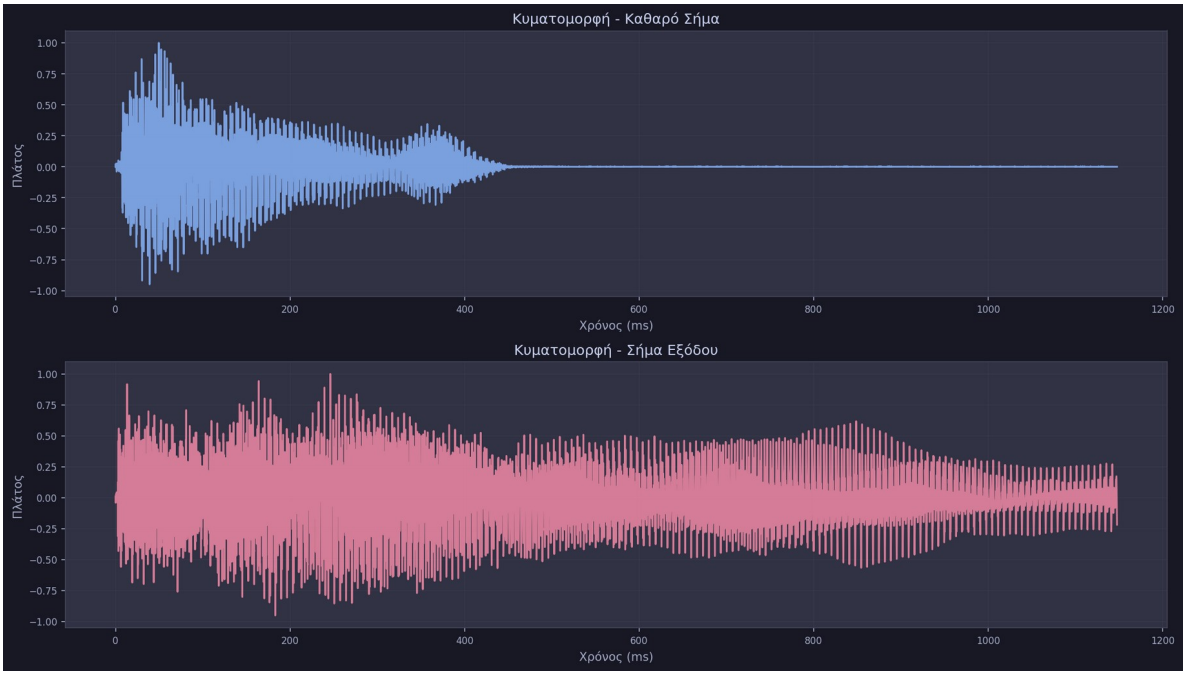
Εικόνα 8.5: Σύγκριση φασματογραφημάτων καθαρού σήματος με σήματος ασθενούς αντήχησης.

Το φασματογράφημα παρουσιάζει ένα όμοιο αποτέλεσμα, καθώς το σήμα αποσβένεται πιο αργά.



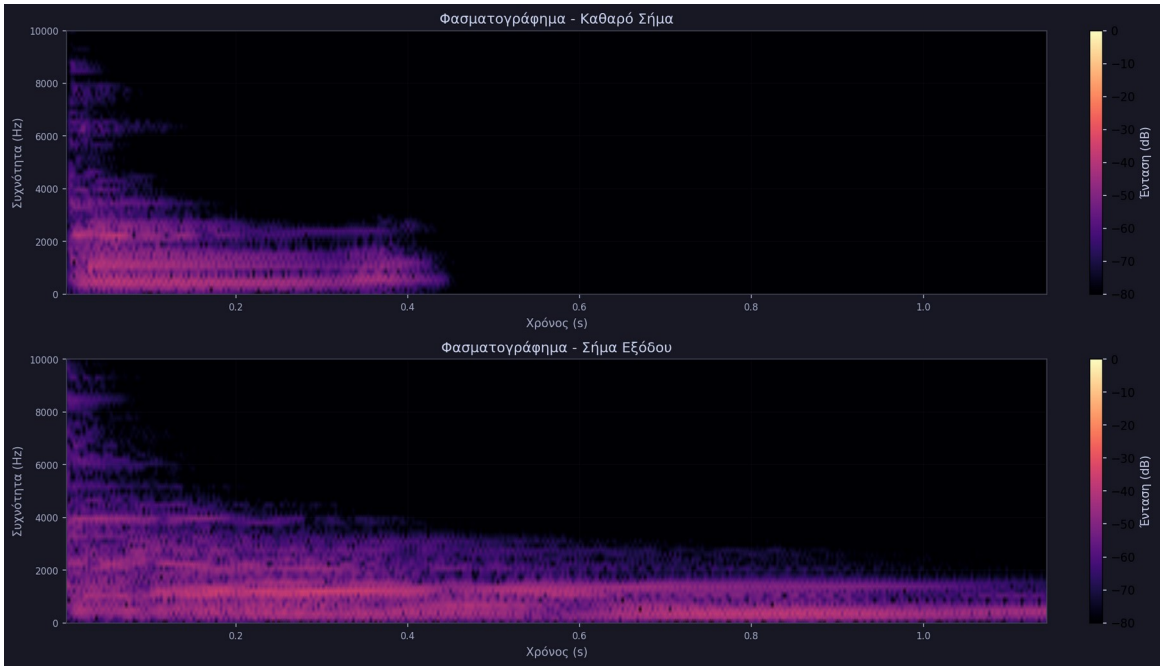
Εικόνα 8.6: Σύγκριση χρωμογραμμάτων καθαρού σήματος με σήματος ασθενούς αντήχησης.

Ιδιαίτερο ενδιαφέρον παρουσιάζουν τα χρωμογράμματα που δείχνουν την επιρροή των comb και allpass φίλτρων. Οι αρμονικές συχνότητες είναι οι D3 και η A2, με το εφέ αντήχησης να φέρνει μια περιοδική εμφάνιση τους.



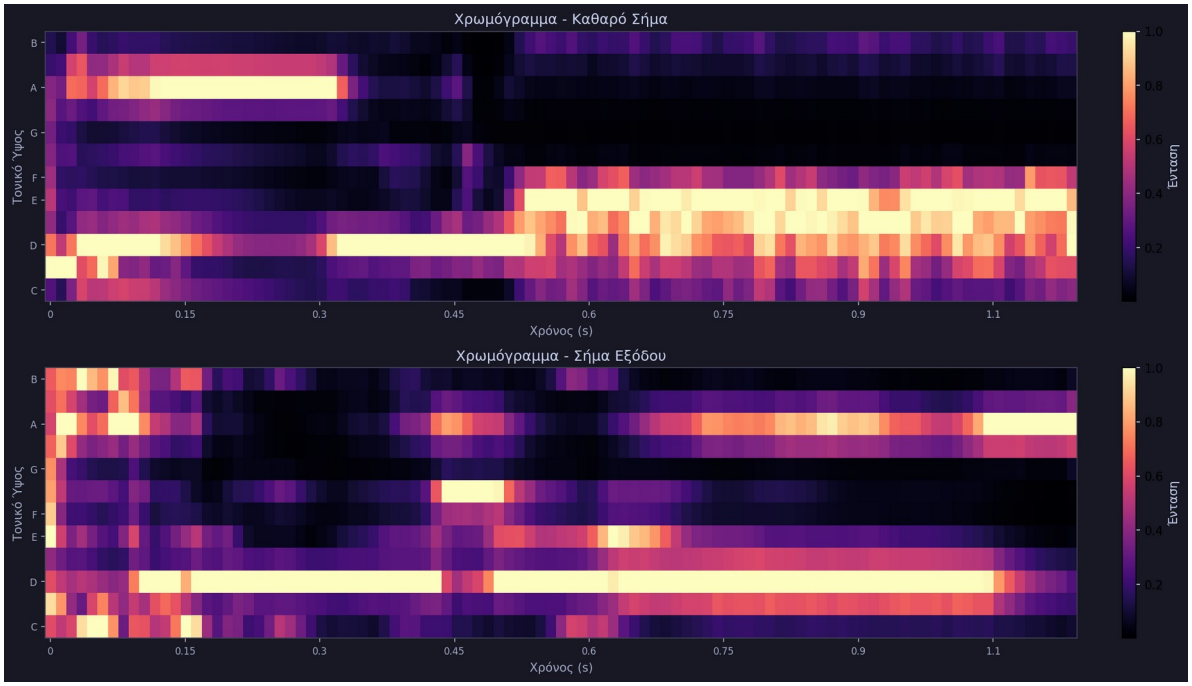
Εικόνα 8.7: Σύγκριση κυματομορφών καθαρού σήματος με σήματος ισχυρής αντήχησης.

Η αλλαγή των παραμέτρων για την δημιουργία ισχυρής αντήχησης προσφέρει εμφανώς μεγαλύτερη διάρκεια ουράς απόσβεσης, με μια ισχυρή περιοδική διακύμανση του πλάτους.



Εικόνα 8.8: Σύγκριση φασματογραφημάτων καθαρού σήματος με σήματος ισχυρής αντήχησης.

Οι συχνότητες συνεχίζονται να αναπαράγονται περιοδικά με όμοια συμπεριφορά όπως οι κυματομορφές.

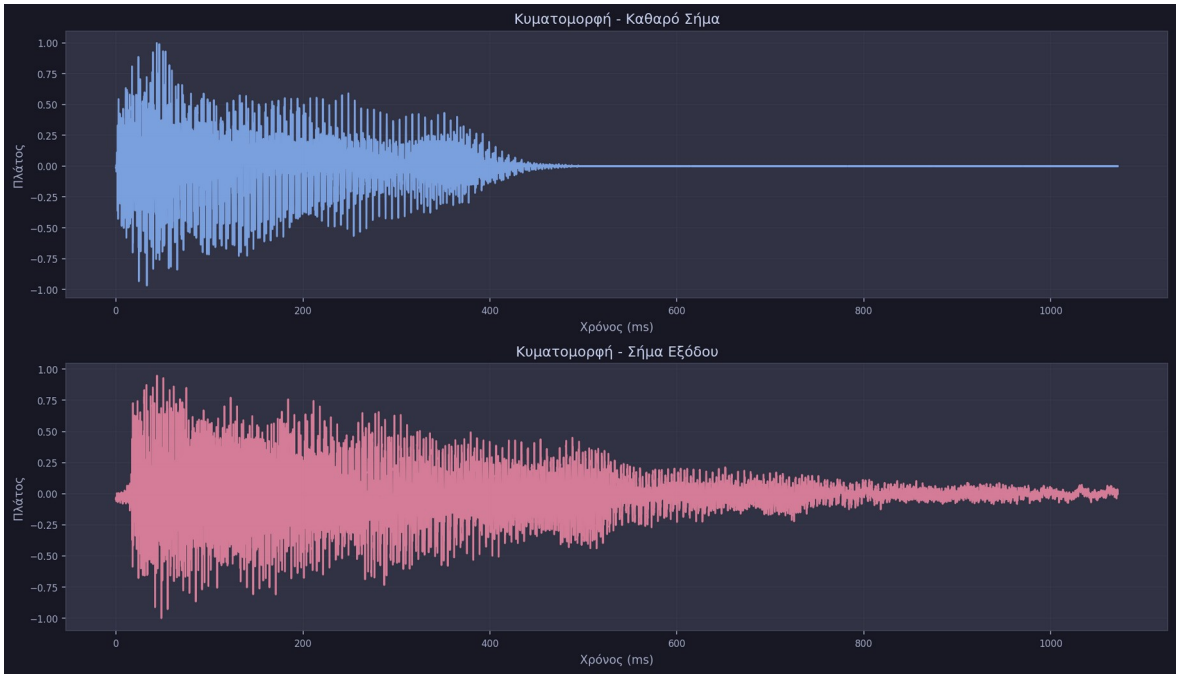


Εικόνα 8.9: Σύγκριση χρωμογραμμάτων καθαρού σήματος με σήματος ισχυρής αντήχησης.

Η ισχυρή αντήχηση κάνει πιο εμφανή την περιοδικότητα των νότων όταν εφαρμόζονται τα comb και all pass φίλτρα.

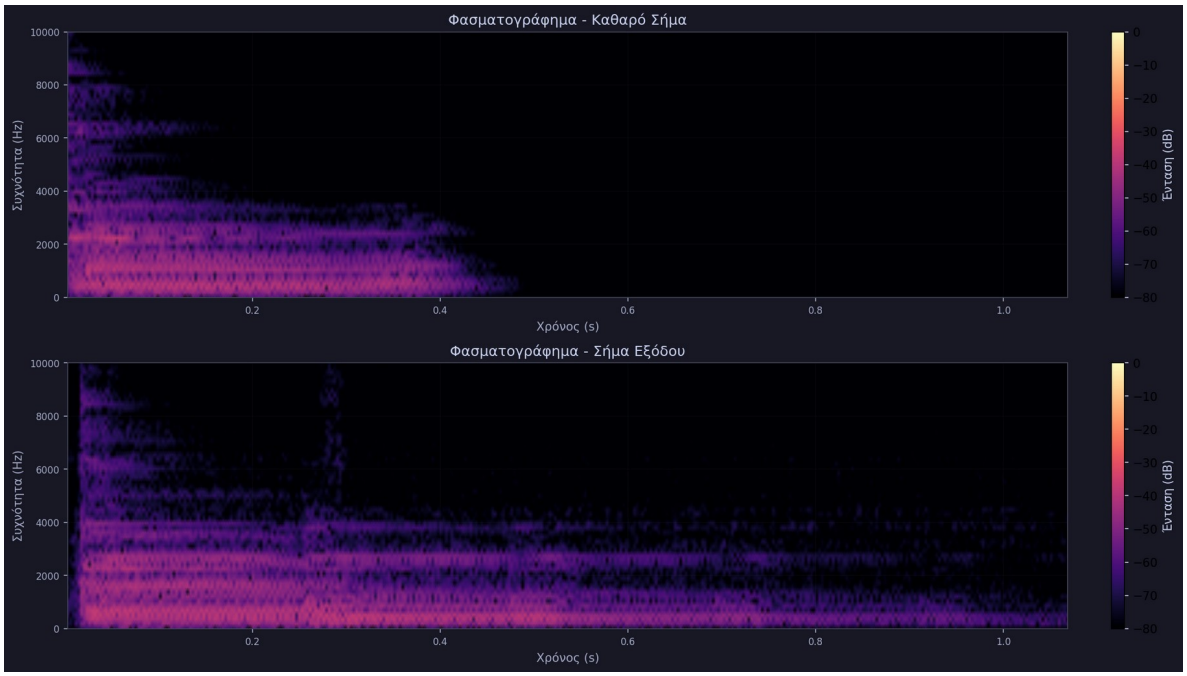
8.4 Echo

Για την δοκιμή του εφέ ηχώ χρησιμοποιήθηκε η νότα D3 (146.83 Hz) που αναπαράγεται σε standard E κούρδισμα με το παίξιμο της 2η χορδής (A2 110Hz) του 5ου τάστου. με διάρκεια 1 δευτερολέπτου μέχρι να εξασθενήσει. Οι παράμετροι που χρησιμοποιήθηκαν για την παραγωγή του εφέ ανέρχονται σε δύο εκδόσεις. Αρχικά την χρήση ενός ασθενούς echo εφέ με μικρό χρόνο delay και dry mix και έπειτα ενός ισχυρού echo με μεγάλο χρόνο delay και wet mix.



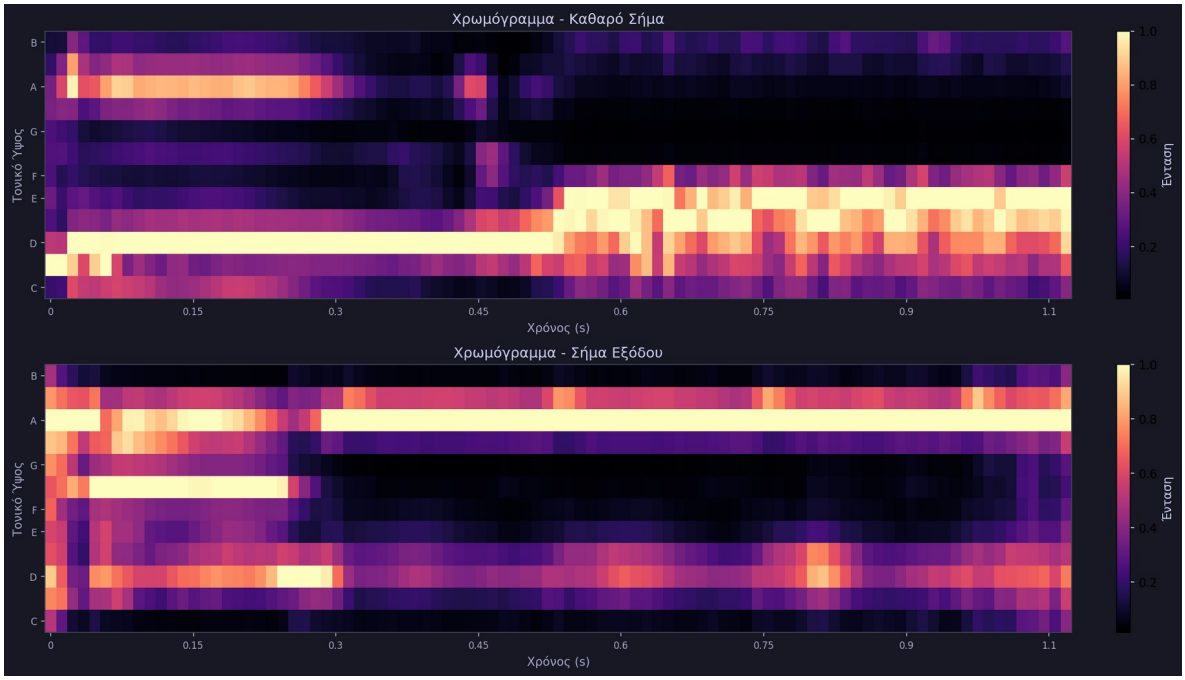
Εικόνα 8.10: Σύγκριση κυματομορφών καθαρού σήματος με σήματος ασθενούς ηχώς.

Στο διάγραμμα φαίνεται, ομοίως όπως και με την αντήχηση, η ουρά απόσβεσης με την διαφορά ότι είναι σταδιακή, χωρίς διακύμανση του πλάτους του σήματος. Το dry mix προδίδει πιο ομαλή απόσβεση αναμειγνύοντας τον αρχικό ήχο με τον καθυστερημένο.



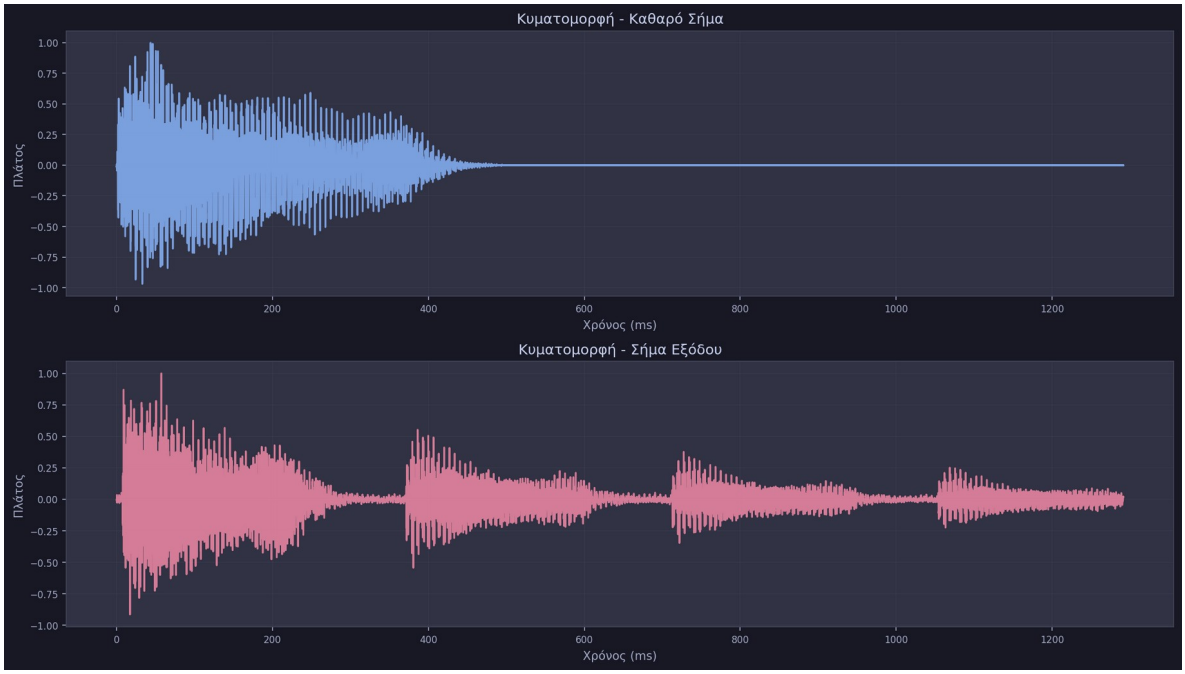
Εικόνα 8.11: Σύγκριση φασματογραφημάτων καθαρού σήματος με σήματος ασθενούς ηχώς.

Από το φασματογράφημα φαίνονται αχνά η επαναλαμβανόμενη αναπαραγωγή τμήματος του σήματος κάθε περίπου 200ms.



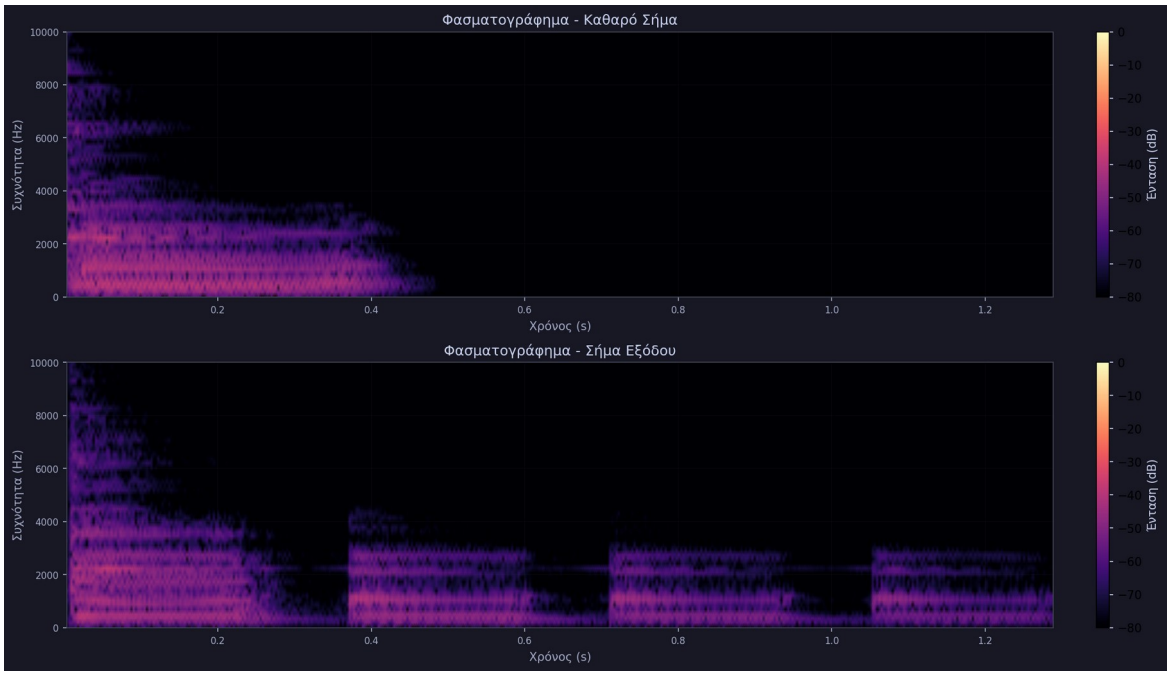
Εικόνα 8.12: Σύγκριση χρωμογράμματος καθαρού σήματος με σήματος ασθενούς ηχώς.

Όμοιως και στο χρωμόγραμμα είναι εμφανής η περιοδικότητα συγκεκριμένων συχνοτήτων των νότων D3 και A2.



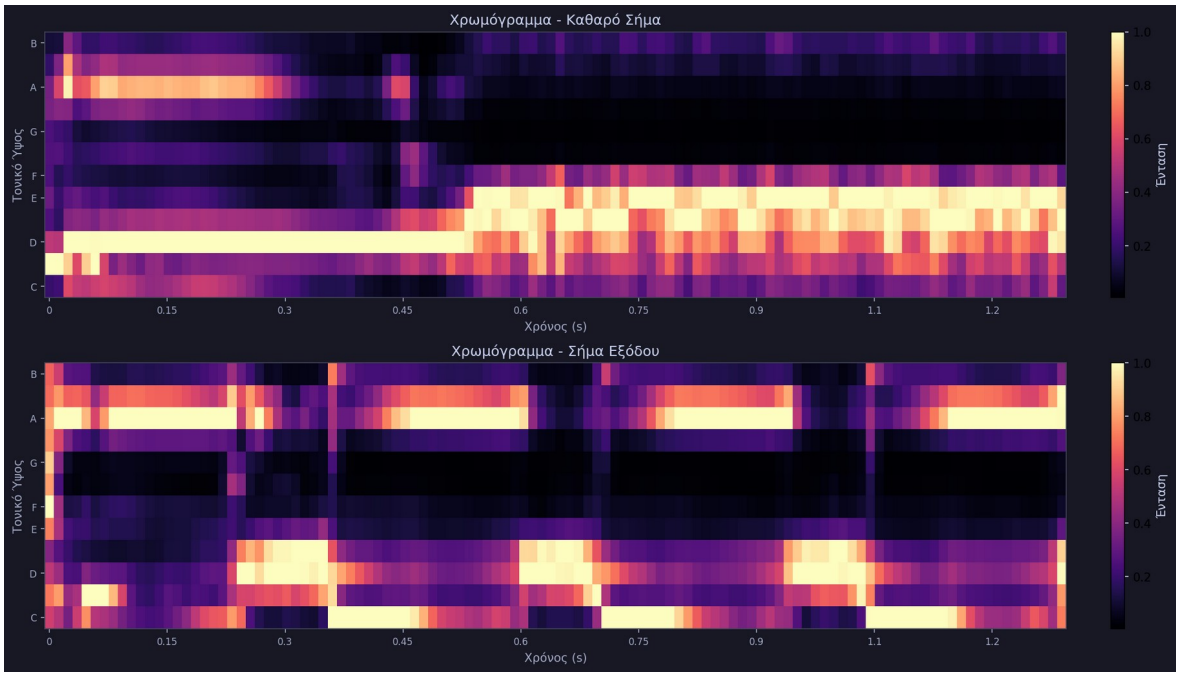
Εικόνα 8.13: Σύγκριση κυματομορφών καθαρού σήματος με σήματος ισχυρής ηχώς.

Σε αυτό το δείγμα ηχού φαίνεται ξεκάθαρα η επιρροή του wet σήματος που προσδίδει περισσότερη έμφαση στο εφέ ηχού. Η περιοδικότητα είναι πλέον ολοφάνερη, περίπου ανά 400ms.



Εικόνα 8.14: Σύγκριση φασματογραφημάτων καθαρού σήματος με σήματος ισχυρής ηχώς.

Το φασματογράφημα εμφανίζει όμοια συμπεριφορά, με την σταδιακή εξασθένηση του επαναλαμβανόμενου σήματος.

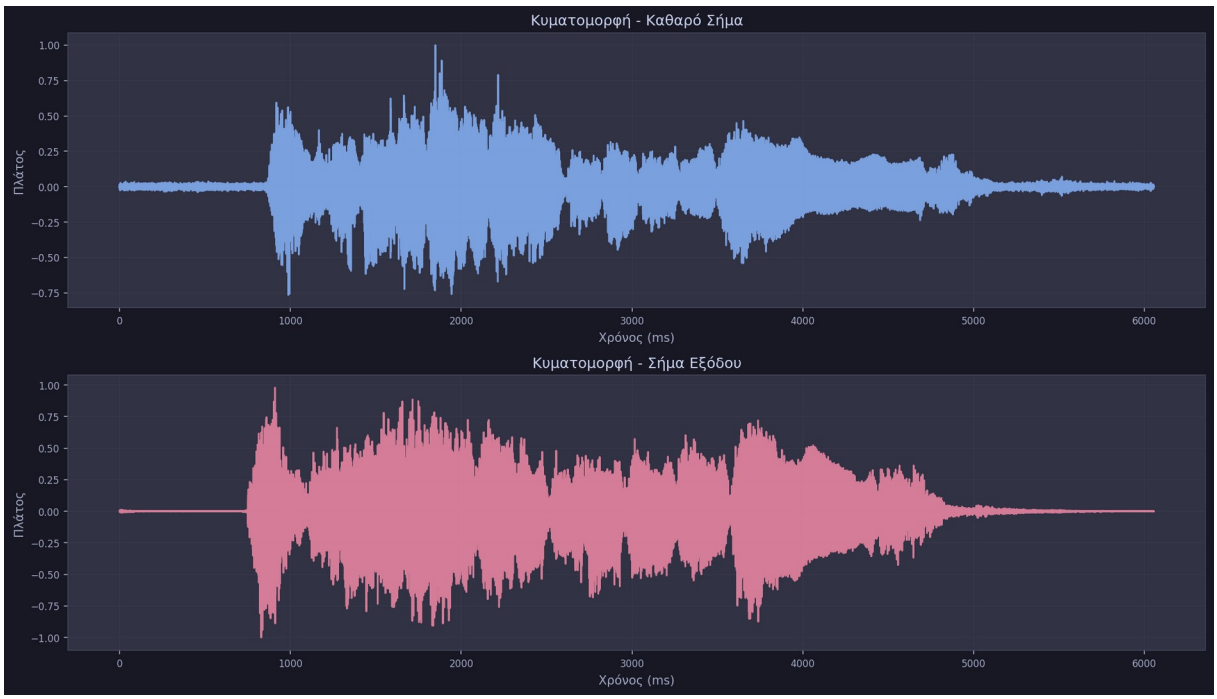


Εικόνα 8.15: Σύγκριση χρωμογράμματος καθαρού σήματος με σήματος ισχυρής ηχώς.

Το χρωμόγραμμα, εμφανίζει την ίδια περιοδικότητα.

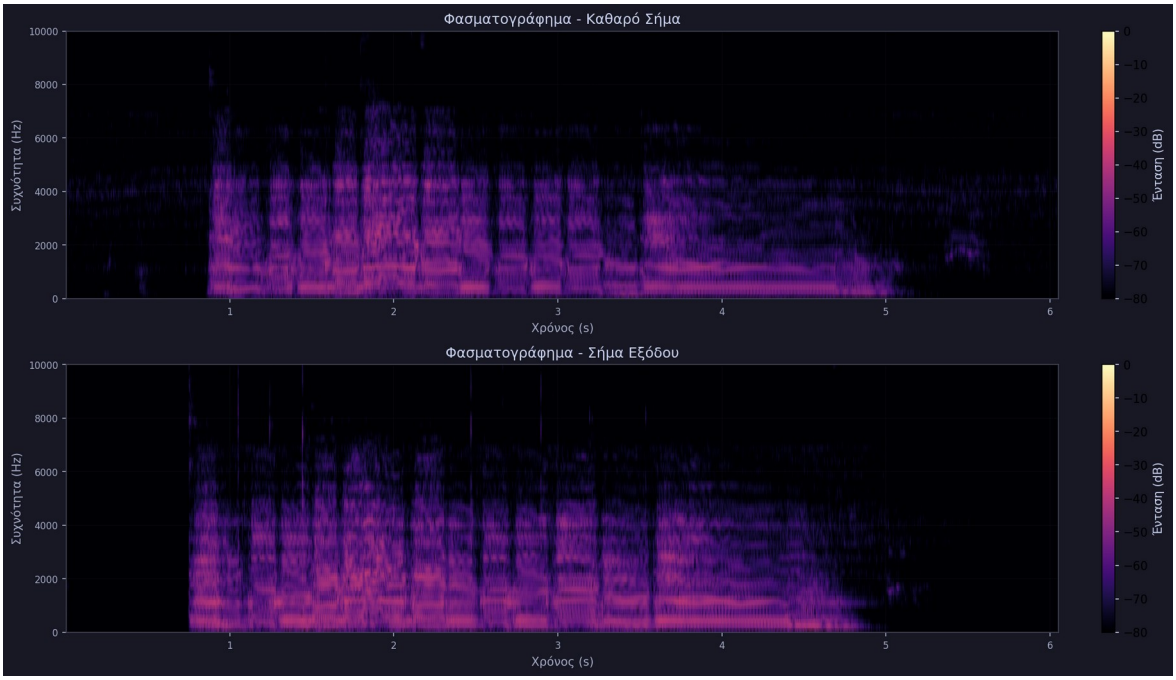
8.5 Noise Gate

Για την δοκιμή του εφέ καταστολέα θορύβου χρησιμοποιήθηκε μια απλή πεντατονική κλίμακα σε διάρκεια περίπου 6 δευτερολέπτων σε συνδυασμό με το εφέ παραμόρφωσης. Το εφέ είναι ρυθμισμένο έτσι ώστε να αποκόβει θόρυβο χαμηλού πλάτους.



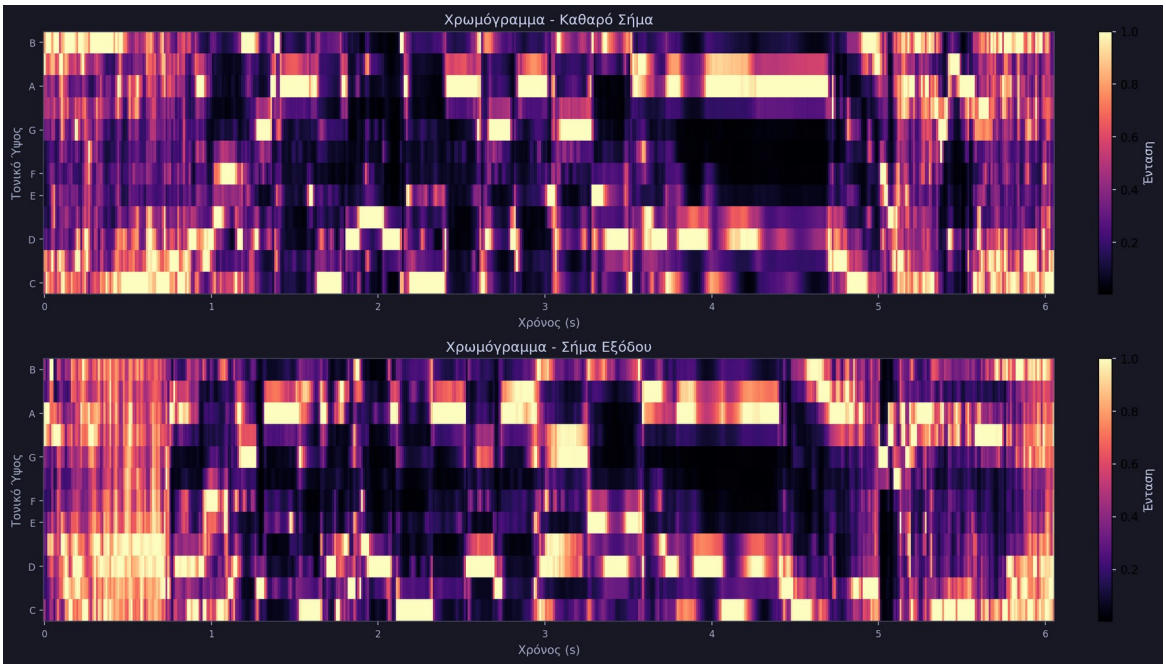
Εικόνα 8.16: Σύγκριση κυματομορφών καθαρού σήματος με σήματος καταστολέα θορύβου.

Το αρχικό σήμα έχει θόρυβο στην αρχή και στο τέλος ενώ μετά την εφαρμογή του noise gate ο θόρυβος μειώνεται σημαντικά.



Εικόνα 8.17: Σύγκριση φασματογραφημάτων καθαρού σήματος με σήματος καταστολέα θορύβου.

Στο διάγραμμα του φασματογραφήματος φαίνονται αχνά οι συχνότητες που προσδίδουν τον θόρυβο στο καθαρό σήμα, ενώ στο σήμα εξόδου έχουν εξαλειφθεί πλήρως.

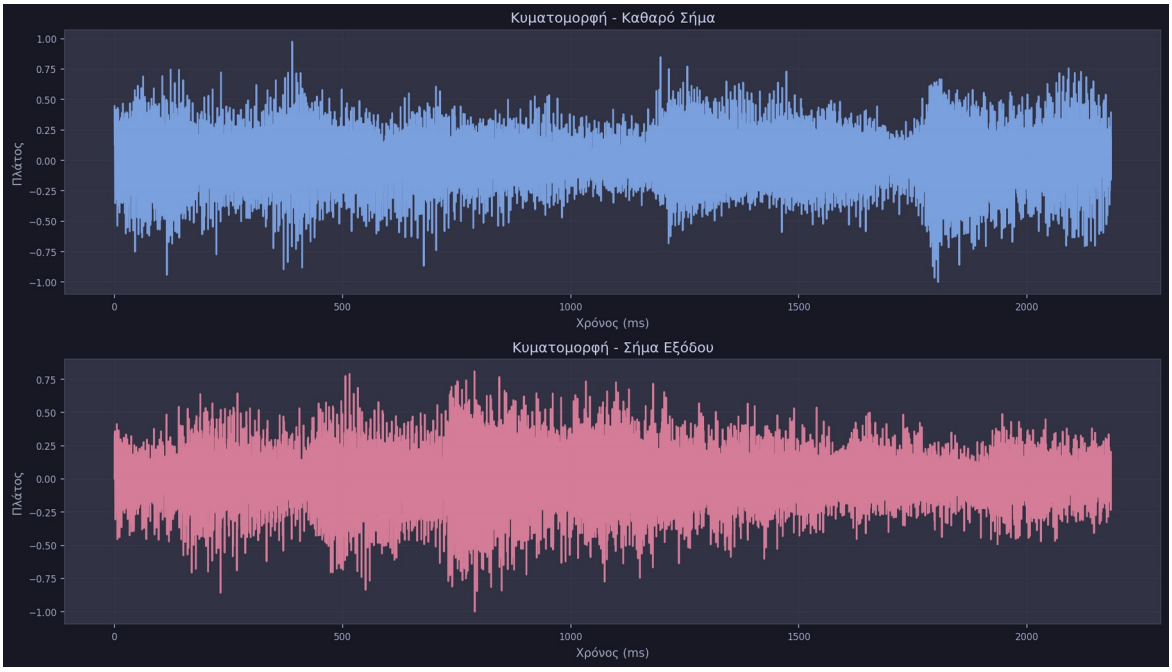


Εικόνα 8.18: Σύγκριση χρωμογράμματος καθαρού σήματος με σήματος καταστολέα θορύβου.

Το χρωμόγραμμα δεν εμφανίζει ιδιαίτερες αλλαγές και παραμένει όμοιο και στις δύο πτυχές.

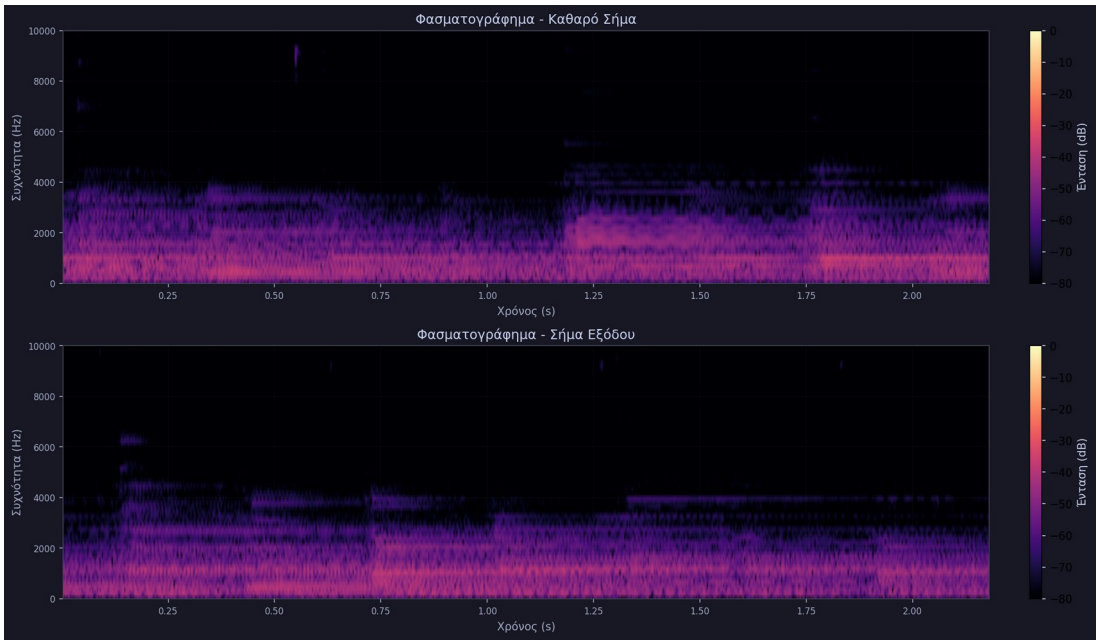
8.6 Compressor

Για την δοκιμή του εφέ κομπρέσσορα χρησιμοποιήθηκε η αρχή του τραγουδιού Beneath the remains των Sepultura με διάρκεια περίπου 2 δευτερολέπτων.



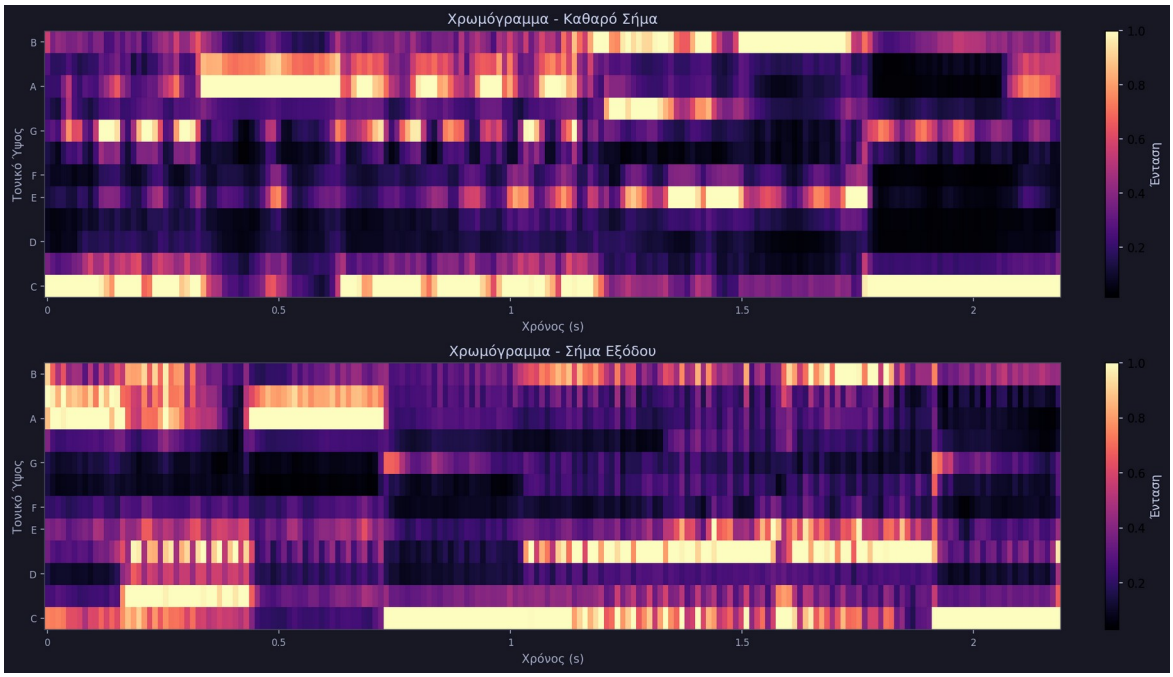
Εικόνα 8.19: Σύγκριση κυματομορφών καθαρού σήματος με σήματος κομπρέσσορα.

Από την σύγκριση των κυματομορφών και μόνο φαίνεται η επίδραση του κομπρέσσορα καθώς φέρνει τις κορυφές του σήματος στο περίπου ίδιο επίπεδο μεταξύ τους.



Εικόνα 8.20: Σύγκριση φασματογραφημάτων καθαρού σήματος με σήματος κομπρέσσορα.

Το φασματογράφημα δείχνει πως όλες οι συχνότητες βρίσκονται επί το πλείστον στην ίδια ένταση ανά επίπεδο.



Εικόνα 8.21: Σύγκριση χρωμογράμματος καθαρού σήματος με σήματος κομπρέσσορα.

Το χρωμόγραμμα παραμένει όμοιο, χωρίς αλλαγές στο τονικό ύψος.

8.7 Επίλογος

Τα αποτελέσματα των πειραματικών μετρήσεων αποδεικνύουν την ορθή λειτουργία των εφέ. Τα διαγράμματα και τα γραφήματα βοηθούν να αναλύσουν την συμπεριφορά που προσδίδει το κάθε εφέ στο καθαρό σήμα.

Κεφάλαιο 9ο: Συμπεράσματα και Μελλοντικές Βελτιώσεις

9.1 Εισαγωγή

Ολοκληρώνοντας την διπλωματική εργασία, διαπιστώθηκε το μεγάλο εύρος γνώσεων που απαιτήθηκε για την δημιουργία της. Κατασκευάζοντας μια πολυεφέ πεταλιέρα για ηλεκτιρκή κιθάρα από την αρχή, χωρίς έτοιμα κυκλώματα και βιβλιοθήκες επεξεργασίας ήχου, αποτέλεσε μια ενδιαφέρουσα εμπειρία καθώς αξιοποιήθηκαν γνώσεις πάνω στην σχεδίαση κυκλωμάτων, δημιουργία λογισμικού για ενσωματωμένα και προφανώς δημιουργία βιβλιοθηκών για ψηφιακή επεξεργασία ήχου. Το τελικό αποτέλεσμα της εργασίας είναι αρκούντος λειτουργικό, με αρκετές δυνατότητες επέκτασης και βελτίωσης σε μελλοντικό χρόνο.

9.2 Μελλοντικές Βελτιώσεις

Η σχεδίαση και ανάπτυξη του συστήματος αποτέλεσε έναν συνεχή βρόγχο έρευνας, σχεδίασης και συλλογής δεδομένων εκ νέου χωρίς προηγούμενη εμπειρία στο αντικείμενο. Για αυτόν τον λόγο, σημειώθηκαν διάφορες μελλοντικές βελτιώσεις στο κύκλωμα αλλά και στο λογισμικό του συστήματος, που θα μπορούσαν να ενσωματωθούν σε μελλοντικό χρόνο.

9.2.1 Βελτιώσεις Κυκλώματος

1. Το κύκλωμα αποτελείται από δύο μέρη, το κύκλωμα από το οποίο περνάει το αναλογικό και ψηφιακό σήμα και το κύκλωμα της διεπαφής. Στο πρωτότυπο σύστημα το οποίο ήταν κατασκευασμένο πάνω σε διάτρητη πλακέτα χρησιμοποιούσε διαφορετικές πλακέτες με την δική της γείωση το καθένα, με αποτέλεσμα τα κυκλώματα να είναι χωρισμένα. Στο καινούργιο κύκλωμα του PCB, τα εξαρτήματα ανήκουν όλα στην ίδια πλακέτα και ως αποτέλεσμα το κύκλωμα της διεπαφής προσδίδει ελάχιστο θόρυβο στο σήμα εξόδου. Αυτό οφείλεται στο switching supply της οθόνης OLED το οποίο μοιράζεται την ίδια γείωση με το κύκλωμα εξόδου. Για την αντιμετώπιση αυτού του προβλήματος υπάρχουν δύο τρόποι επίλυσης. Ο πιο αποτελεσματικός είναι η χρήση δύο διαφορετικών γειώσεων για το ψηφιακό και αναλογικό κομμάτι. Μια ακόμη λύση είναι η χρήση ground stitching και πυκνωτών γύρω από την διεπαφή SPI της οθόνης.
2. Χρήση ακολούθου τάσης κατά την είσοδο και έξοδο του σήματος. Ο ακόλουθος τάσης συμβάλει στον διαχωρισμό του ρεύματος εισόδου του σήματος με την χρήση του ρεύματος του τελεστικού, προσδίδοντας μια σταθερότητα στο σήμα, χαμηλή αντίσταση και μειώνοντας τον θόρυβο.
3. Προσθήκη 9V τροφοδοτικού αντί της χρήσης του τροφοδοτικού του STM32 ώστε να αποφευχθεί η χρήση του DC-DC αντιστροφέα τάσης για να παραχθούν τα 9V (+-5V) που χρειάζεται ο τελεστικός TL072. Ενώ ο TC1044S δίνει την δυνατότητα χρήσης του boost pin για να ενισχύσει το switching mode στα 30kHz, υπάρχει ακόμα ελάχιστος θόρυβος λόγω αυτού.
4. Προσθήκη ποτενσιόμετρου για ρύθμιση συνολικής έντασης του συστήματος, έναντι της χρήσης λογισμικού. Θα μπορούσε να προστεθεί ένα sliding ποτενσιόμετρο το οποίο απλώς συνδέεται με το σήμα εξόδου και διαμορφώνει την τελική ένταση.

9.2.2 Βελτιώσεις Λογισμικού

1. Αρχικά οι βιβλιοθήκες της HAL θα μπορούσαν να είναι γραμμένες με τη χρήση των βιβλιοθηκών LL και CMSIS, οι οποίες είναι αντίστοιχες βιβλιοθήκες υλικού βελτιστοποιημένες για την καλύτερη δυνατή αποδοτικότητα. Η βασική διαφορά τους από την HAL έγκειται στο ότι λειτουργεί ως ένα στρώμα αφαίρεσης που προσφέρει εύχρηστες συναρτήσεις για γρήγορη ανάπτυξη, αλλά συχνά εισάγει καθυστερήσεις και περιττό κώδικα. Αντίθετα, οι βιβλιοθήκες LL (Low Layer) και CMSIS (Cortex Microcontroller Software Interface Standard) επιτρέπουν την απευθείας αλληλεπίδραση με τους καταχωρητές (registers) κάθε περιφερειακού υλικού, χωρίς τα σύμβολα και τις αφαιρέσεις υψηλού επιπέδου που χρησιμοποιεί η HAL. Σε εφαρμογές όπως τα πετάλια κιθάρας, όπου ο χρόνος απόκρισης μετριέται σε μικροδευτερόλεπτα, η χρήση LL και CMSIS μπορεί να μειώσει σημαντικά την καθυστέρηση, να ελευθερώσει πόρους του επεξεργαστή και να επιτρέψει την ταυτόχρονη εκτέλεση περισσότερων εφέ σε πραγματικό χρόνο.
2. Αποθήκευση πολλών σειρών εφέ και κυκλική επιλογή τους μέσω των μεγάλων πεταλιών. Αυτή η βελτίωση αφορά τη διαχείριση προεπιλογών και τη δυνατότητα άμεσης εναλλαγής τους κατά τη διάρκεια της εκτέλεσης.
3. Δημιουργία καινούργιας λειτουργίας για κούρδισμα της κιθάρας. Η λειτουργία αυτή μπορεί να γίνει μέσω αλγορίθμων αναγνώρισης της κύριας αρμονικής συχνότητας μέσα από το καθαρό σήμα. Η ακρίβεια μπορεί να είναι πολύ πιο αξιόπιστη από τα συμβατικά κουρδιστ ήρια που χρησιμοποιούν μικρόφωνο για την καταγραφή, καθώς ο ήχος ψηφιοποιείται άμεσα μέσω καλωδίωσης.
4. Προσθήκη παραπάνω εφέ. Μέχρι στιγμής αναπτύχθηκαν 5 συνολικά εφέ, τα οποία καλύπτουν ένα εύρος από τα πρωταρχικά εφέ που βγήκαν στην αγορά. Χωρίς όμως να αντικατοπτρίζουν τα δεκάδες σύγχρονα πολύπλοκα εφέ που μπορούν να βρεθούν σήμερα, όπως αυτά που βασίζονται σε μοντελοποίηση πραγματικών ενισχυτών (amp modeling), συνθετικά εφέ αρμονικών (pitch shifting, octave effects), φίλτρα με μεταβαλλόμενη συχνότητα (phasers, wah-wah με αυτόματη κίνηση), πολύπλοκες χρονικές καθυστερήσεις με ρυθμικές υποδιαιρέσεις (multi-tap delay) ή ακόμα εφέ που χρησιμοποιούν τεχνικές επεξεργασίας φάσματος, όπως το shimmer reverb ή το dynamic delay. Η προσθήκη νέων εφέ δεν είναι μόνο θέμα αριθμού αλλά και ποιότητας και ευελιξίας, καθώς κάθε νέο εφέ απαιτεί προσεκτική ρύθμιση παραμέτρων, χαμηλή κατανάλωση επεξεργαστικής ισχύος και δυνατότητα παραλληλοποίησης ή σειριακής σύνδεσης με άλλα εφέ.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Wikipedia, *“Guitar”*, https://en.wikipedia.org/wiki/Guitar

[2] Vintage Guitar, *“Ro-Pat-In’s First Electric Spanish”*,

https://www.vintageguitar.com/3588/ro-pat-ins-first-electric-spanish/

[3] Smithsonian, *“The Invention of the Electric Guitar”*, https://invention.si.edu/invention-stories/invention-electric-guitar

[4] Wikipedia, Museum of Making Music, https://commons.wikimedia.org/wiki/File:Elektrofryingpan.jpg

[5] Nigel Osbourne / Redferns, *“The Log”*, https://www.gettyimages.com/detail/news-photo/studio-still-life-of-les-pauls-1940-log-guitar-a-prototype-news-photo/102167386

[6] PBS, Les Paul: Chasing Sound,

https://www.pbs.org/wnet/americanmasters/les-paul-chasing-sound/100/

[7] PRESTO MUSIC TIMES, *“Music – The Universal Language”*,https://elibrary.arcade-museum.com/magazines/presto/PRESTO-1941-2301/PRESTO-1941-2301-04.pdf

[8] The Rolling Stones, *“(I Can’t Get No) Satisfaction”*,

https://www.youtube.com/watch?v=nrIPxlFzDi0

[9] Red Rooster, *“Gibson Maestro FZ-1 Fuzz-tone”*, https://en.wikipedia.org/wiki/Maestro_FZ-1_Fuzz-Tone#/media/File:Gibson_maestro_fuzz_tone_1_752.jpg

[10] Pedal Players, *“History of Guitar Effects Pedals: From Fuzz to Modern Tech”*, Gareph Joseph,

https://pedalplayers.com/history-of-guitar-effects-pedals/

[11] H. Iwai, *"History of transistor invention: 75th anniversary," 2022 IEEE 16th International Conference on Solid-State & Integrated Circuit Technology (ICSICT)*, Nangjing, China, 2022, pp. 1-10.

[12] Nokia, *“1956 Nobel Prize in Physics”*,

https://www.nokia.com/bell-labs/about/awards/1956-nobel-prize-physics/

[13] C. Phipps, *"The Early History of ICs at Texas Instruments: A Personal View"*, in *IEEE Annals of the History of Computing*, vol. 34, no. 1, pp. 37-47, Jan. 2012

[14] Texas Instruments, *“Kilby's original integrated circuit, 1958”*, http://www.ti.com/corp/graphics/press/image/on_line/co1034.jpg

[15] Eventide, *“Flashback #6: HM80 — The Baby Harmonizer®”*, https://www.eventideaudio.com/blog/50th-flashback-6-hm80-the-baby-harmonizer/

[16] Digital Audio Theory: A Practical Guide, Christopher L. Bennett,2020, p. 1.1

[17] Steven W. Smith, Ph.D. The Scientist and Engineer's Guide to Digital Signal Processing, chapter 2, Signal and Graph Terminology

[18] Christopher L. Bennett, Digital Audio Theory: A Practical Guide, p. 1.3

[19] Federico Avanzini and Giovanni De Poli, Fundamentals of digital audio processing, 1.4.2.2 The sampling theorem and the Nyquist frequency

[20] Steven W. Smith,, The Scientist and Engineer's Guide to Digital Signal Processing, Ph.D. Quantization

[21] Christopher L. Bennett, Digital Audio Theory: A Practical Guide, p. 1.1.4 Signal to noise ratio (SNR)

[22] Principles of Data Acquisition and Conversion, Application Report, Texas Instruments – https://www.ti.com/lit/an/sbaa051a/sbaa051a.pdf

[23] Polotti, P., & Rocchesso, (2008). Sound to Sense, Sense to Sound: A State of the Art in Sound and Music Computing. p. 1.4.5

[24] Vesa Välimäki, Antti Huovilainen, Oscillator and Filter Algorithms for Virtual Analog Synthesis, p. 1, June 2006

[25] Smith, J. O. Two Approaches to Physical Modeling. Stanford University CCRMA

[26] Ingrid Verbauwhede, Patrick Schaumont, Christian Piguet, Bart Kienhuis, Architectures and Design techniques for energy efficient embedded DSP and multimedia processing, February 2004

[27] Wikipedia, *"Low Pass Filter"*, https://en.wikipedia.org/wiki/Low-pass_filter

[28] Wikipedia, *"High Pass Filter"*, https://en.wikipedia.org/wiki/High-pass_filter

[29] Wikipedia, *"Comb Filter"*, https://en.wikipedia.org/wiki/Comb_filter

[30] Zölzer, U. (2011), DAFX: Digital Audio Effects, John Wiley & Sons, 2.5.2 IIR comb filter

[31] Schroeder, M. R. (1962), Natural Sounding Artificial Reverberation, Journal of the Audio Engineering Society.

[32] Wikipedia, *"All-Pass Filters"*, https://www.uaudio.com/blogs/ua/allpass-filters

[33] PHILLIP A. REGALIA, The Digital All-Pass Filter: A Versatile Signal Processing Building Block,

[34] Standford University, PHYSICAL AUDIO SIGNAL PROCESSING, JULIUS O. SMITH III, All-pass two combs, https://ccrma.stanford.edu/~jos/pasp/Allpass_Two_Combs.html

[35] Zölzer, U. (2011), DAFX: Digital Audio Effects, John Wiley & Sons, 4.2.2 Compressor and expander

[36] Arlyn Reese Madsen III, Dynamic Range Compression and Its Effect on Music Genre Classification

[37] Giannoulis, Dimitrios, et al. "Digital Dynamic Range Compressor Design—A Tutorial and Analysis.", AES: Journal of the Audio Engineering Society, vol. 60, July 2012

[38] Zölzer, U. (2011), DAFX: Digital Audio Effects, John Wiley & Sons, 4.2.3 Noise Gate

[39] AVID, *"What is Reverb? Types, Parameters, & Uses Explained"*,

https://www.avid.com/resource-center/what-is-reverb

[40] University of Rochester, M. R. Schroeder, “Natural Sounding Artificial Reverberation”, July 1962

[41] Standford University, PHYSICAL AUDIO SIGNAL PROCESSING, JULIUS O. SMITH III, Schroeder Reverberators, https://ccrma.stanford.edu/~jos/pasp/Schroeder_Reverberators.html

[42] SoundBridge, *“Reflect on reverb”*, https://www.soundbridge.io/reverb-reflection

[43] Wikipedia, *“Echo”*, https://el.wikipedia.org/wiki/%CE%97%CF%87%CF%8E

[44] MusicStreet, *“What is echo? Complete guide for guitarists.*”, https://www.musicstreet.co.uk/blogs/blog-post/what-is-echo-guitar-guide

[45] Michael Banfield Guitar, *“Soft vs. hard clipping in drive pedals*.” https://www.michaelbanfieldguitar.com/blog/soft-vs-hard-clipping-in-dr

[46] Schuck, A., Jr., & Bodmann, B. E. J. (2016). Audio nonlinear modeling through hyperbolic tangent functionals. In Proceedings of the 19th International Conference on Digital Audio Effects (DAFx-16) (pp. 1–6). Brno, Czech Republic.

[47] WeActStudio. MiniSTM32H7xx, https://github.com/WeActStudio/MiniSTM32H7xx

[48] STMicroelectronics, ARM Cortex-M7,

https://www.st.com/content/st_com/en/arm-32-bit-microcontrollers/arm-cortex-m7.html

[49] Wikipedia, *“Cutoff frequency”*, https://en.wikipedia.org/wiki/Cutoff_frequency

[50] Jaegger, R. C., & Blalock, T. N., *Μικροηλεκτρονικη Σχεδίαση Κυκλωμάτων*.

[51] Texas Instruments, TL072 datasheet, https://www.ti.com/lit/ds/symlink/tl072.pdf

[52] SAEF Display, *“128x64 dots OLED display module*”, https://greek.saefdisplay.com/sale-25304728-128x64-dots-oled-display-module-2-4-2-42-iso9001-iso14000-certificate.html

[53] Peterson, Z, *“Everything you need to know about stitching vias”*, Altium, https://resources.altium.com/p/everything-you-need-know-about-stitching-vias

[54] Blender Foundation, *“About Blender”*, https://www.blender.org/abo.ut/