ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
Τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών
www.eee.uniwa.gr
Θηβών 250, Αθήνα-Αιγάλεω 12244
Τηλ. +30 210 538-1225, Fax. +30 210 538-1226

UNIVERSITY of WEST ATTICA
FACULTY OF ENGINEERING
Department of Electrical & Electronics Engineering
www.eee.uniwa.gr
250, Thivon Str., Athens, GR-12244, Greece
Tel:+30 210 538-1225, Fax:+30 210 538-1226

Πρόγραμμα Μεταπτυχιακών Σπουδών
Τεχνολογίες Ήχου, Βίντεο και Μετάδοσης

Master of Science in
Audio, Video and Broadcasting Engineering

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ AUDIO STREAMER ΒΑΣΙΣΜΕΝΟΥ ΣΕ ΠΕΡΙΒΑΛΛΟΝ RASPBERRY

Μεταπτυχιακός Φοιτητής : Νικόλαος Ανδρεόπουλος, ΑΜ MSCAVB-001
Επιβλέπων : Οδυσσέας Τσακιρίδης, Επίκουρος Καθηγητής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, 25 / 06 / 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
Τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών
www.eee.uniwa.gr
Θηβών 250, Αθήνα-Αιγάλεω 12244
Τηλ. +30 210 538-1225, Fax. +30 210 538-1226

UNIVERSITY of WEST ATTICA
FACULTY OF ENGINEERING
Department of Electrical & Electronics Engineering
www.eee.uniwa.gr
250, Thivon Str., Athens, GR-12244, Greece
Tel:+30 210 538-1225, Fax:+30 210 538-1226

Πρόγραμμα Μεταπτυχιακών Σπουδών
Τεχνολογίες Ήχου, Βίντεο και Μετάδοσης

Master of Science in
Audio, Video and Broadcasting Engineering

MSc Thesis

DESIGN AND IMPLEMENTATION OF AUDIO STREAMER BASED ON RASPBERRY ECOSYSTEM.

Student: Andreopoulos Nikolaos, Registration Number MSCAVB-001
MSc Thesis Supervisor: Tsakiridis Odysseas, Assistant Professor

ATHENS-EGALEO, 25 / 06 / 2022

Η Μεταπτυχιακή Διπλωματική Εργασία έγινε αποδεκτή, εξετάστηκε και βαθμολογήθηκε από την εξής τριμελή εξεταστική επιτροπή:

Επιβλέπων	Μέλος	Μέλος
Odyssefs Tsakiridis Digitally signed by Odyssefs Tsakiridis Date: 2022.07.18 20:47:41 +03'00'	 Digitally signed by Stylianos Potirakis Date: 2022.07.18 21:09:17 +03'00'	Ilias Stavrakas Digitally signed by Ilias Stavrakas Date: 2022.07.19 09:30:30 +03'00'
Οδυσσέας Τσακιρίδης	Στυλιανός Ποτηράκης	Ηλίας Σταύρακας
Επίκουρος Καθηγητής	Καθηγητής	Καθηγητής

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Ανδρεόπουλος Νικόλαος του Αθανασίου, με αριθμό μητρώου MSCAVB-001 φοιτητής του Προγράμματος Μεταπτυχιακών Σπουδών «Τεχνολογίες Ήχου, Βίντεο και Μετάδοσης» του Τμήματος Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

Ανδρεόπουλος Νικόλαος

ΠΕΡΙΛΗΨΗ

Στόχος της παρούσας διπλωματικής εργασίας είναι η υλοποίηση και αξιολόγηση ενός λήπτη ψηφιακών μουσικών σημάτων (music streamer) μικρού μεγέθους και χαμηλού κόστους από απομακρυσμένους εξυπηρετητές (Youtube, Bandcamp, TuneIn, Internet Archive, web radio κ.α.) και από τοπικά μέσα αποθήκευσης καθώς και την αναπαραγωγή του αναλογικού ακουστικού σήματος. Η υλοποίηση του συστήματος θα περιλαμβάνει την φυσική διασύνδεση της συσκευής με το διαδίκτυο ενσύρματα ή ασύρματα, την επικοινωνία με τους απομακρυσμένους εξυπηρετητές, τον έλεγχο της συσκευής μέσω κατάλληλου λογισμικού (human interface) καθώς και τον στερεοφωνικό ψηφιοαναλογικό μετατροπέα. Για την επίτευξη του στόχου αυτού θα χρησιμοποιήσουμε τον υπολογιστή Raspberry Pi, το λογισμικό ανοιχτού κώδικα Mopidy και ένα στερεοφωνικό ψηφιοαναλογικό μετατροπέα (DAC) συμβατό με το Raspberry Pi.

Στην παρούσα Διπλωματική εργασία γίνεται μια ιστορική αναδρομή στις τεχνικές αποθήκευσης και αναπαραγωγής μουσικού περιεχομένου, μέχρι τη σημερινή εποχή που έχουν κυριαρχήσει οι υπηρεσίες απομακρυσμένων μουσικών εξυπηρετητών. Επίσης γίνεται μία αναφορά στα διάφορα μοντέλα Raspberry Pi που κυκλοφορούν στην αγορά, τα κύρια τεχνικά τους χαρακτηριστικά, καθώς και συγκριτικοί πίνακες τιμών τους στην Ελληνική αγορά. Έπειτα αναλύεται βήμα - βήμα η διαδικασία της εγκατάστασης του λειτουργικού συστήματος Raspberry Pi OS, η φυσική διασύνδεση του στερεοφωνικού ψηφιοαναλογικού μετατροπέα, η εγκατάσταση του απαραίτητου λογισμικού ανοιχτού κώδικα όσον αφορά το κομμάτι του server και του client και τέλος γίνεται μία ανάλυση πάνω σε σημαντικά κομμάτια του κώδικα που χρησιμοποιήθηκε, ώστε να κατανοήσουμε τον τρόπο λειτουργίας του.

ΛΕΞΕΙΣ – ΚΛΕΙΔΙΑ: απομακρυσμένοι εξυπηρετητές, λειτουργικό σύστημα Raspberry Pi OS, λήπτης ψηφιακών μουσικών σημάτων, μουσικός σέρβερ, Mopidy, ψηφιοαναλογικός μετατροπέας, Raspberry Pi, Bandcamp, TuneIn, Internet Archive , web radio, Youtube.

ABSTRACT

The aim of this thesis is the implementation and the evaluation of a small-sized and low-cost audio streamer, that receives music streams from remote music servers (Youtube, Bandcamp, TuneIn, Internet Archive, web radio etc) as well as the reproduction of the analog audio signal. That implementation will include the physical interconnection of the device with internet, the communication with the remote servers, the control of the device through appropriate software (human interface) and a digital to analog converter (DAC). To achieve that goal, we will use a Raspberry Pi, the open source software Mopidy and a digital to analog converter compatible with Raspberry Pi.

In this thesis, a historical review concerning the recording, storage and reproduction techniques is taking place, from the early attempts up to the present era that has been dominated by the remote music services. Moreover, there is a reference to the various Raspberry Pi models, their main technical characteristics, as well as comparative tables that include the prices of every model in the Greek market. Furthermore, it is described step-by-step the process of installing the Raspberry Pi OS, the physical interconnection of the stereo digital to analog converter (DAC), the installation of the appropriate open-source software for the client and the server side and finally an analysis on important parts of the source-code is performed, in order to understand how it works.

KEYWORDS: Bandcamp, digital to analog converter (DAC), Mopidy, music server, music streamer, Raspberry Pi, Raspberry Pi OS, remote music server, TuneIn, web radio, Youtube.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή Δρ. Τσακιρίδη Οδυσσέα για την καθοδήγηση, τις συμβουλές του αλλά και την έμπνευση που μου πρόσφερε κατά την διάρκεια αυτής της διπλωματικής εργασίας.

ΠΙΝΑΚΑΣ ΣΥΜΒΟΛΩΝ-ΑΚΡΩΝΥΜΙΩΝ-ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

Ακρωνύμιο	Περιγραφή
CD	Compact Disc
DAC	Digital To Analog Converter
dB	Decibel
HAT	Hardware Attached On Top
HDMI	High - Definition Multimedia Interface
I2C	Inter - Integrated Circuit
LP	Long Play
mp3	MPEG - 2 Audio Layer III
MPD	Music Player Daemon
PCM	Pulse Code Modulation
PWM	Pulse Width Modulation
SBC	Single - Board Computer
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver – Transmitter
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
wav	Waveform Audio File Format

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΕΙΣΑΓΩΓΗ:
Αντικείμενο, ερευνητικά ερωτήματα και διάρθρωση της εργασίας

Με τον όρο music streamer εννοούμε μια υλοποίηση η οποία έχει τη δυνατότητα να αναπαράγει διαδικτυακά μουσική από απομακρυσμένες υπηρεσίες παροχής μουσικού περιεχομένου ή τοπικά από αποθηκευτικά μέσα όπως εξωτερικούς σκληρούς δίσκους ή usb sticks [1]. Γιατί όμως να διαθέτουμε ένα music streamer; Ένας music server διαθέτει αρκετά πλεονεκτήματα έναντι των παραδοσιακών μορφών ακρόασης μουσικής. Μερικά από αυτά είναι τα παρακάτω :

- Παρέχει ένα εύκολο τρόπο να αναζητούμε και να αναπαράγουμε μουσικό περιεχόμενο. Μπορούμε μέσα σε λίγα δευτερόλεπτα να έχουμε άμεση πρόσβαση σε όλη τη μουσική μας συλλογή, να δημιουργούμε λίστες αναπαραγωγής ανάλογα με τις προτιμήσεις μας και να την ελέγχουμε απομακρυσμένα με ένα τηλεχειριστήριο, κινητό τηλέφωνο ή προσωπικό υπολογιστή. Επίσης δεν θα χρειάζεται να αλλάζουμε διαρκώς CD ή δίσκους βινυλίου στο στερεοφωνικό μας, με αποτέλεσμα να αποφεύγεται η φθορά τους, διότι όπως είναι γνωστό, ειδικά οι δίσκοι βινυλίου είναι πολύ ευπαθείς και φθείρονται εύκολα κατά τη χρήση τους.

- Είναι σαφώς πιο οικονομικοί σαν αγορά, από το να αγοράσει κάποιος ξεχωριστές συσκευές αναπαραγωγής ήχου όπως CD player ή συσκευή pickup για αναπαραγωγή δίσκων βινυλίου. Και το σημαντικό εδώ είναι ότι δεν υστερούν σε ποιότητα ήχου, σε σχέση με τις προαναφερθείσες συσκευές.

- Δίνει τη δυνατότητα να ανακαλύπτουμε συνεχώς νέα μουσική χρησιμοποιώντας τις υπηρεσίες streaming όπως πχ το Youtube [2] ή το Spotify [3] μέσω της δυνατότητας αναζήτησης μουσικής ανά καλλιτέχνη, είδος μουσικής, χώρας προέλευσης ή μέσω των προτεινόμενων λιστών αναπαραγωγής από τις υπηρεσίες αυτές.

- Από τη στιγμή που δεν χρειάζεται να χρησιμοποιούμε CD και δίσκους βινυλίου, μπορούμε να εξοικονομήσουμε χώρο στα ράφια, όπου θα τοποθετούσαμε τη συλλογή μας.

Μειονεκτήματα music streamers

Οι συσκευές music streamers αν και έχουν όμορφο σχεδιασμό, το κύριο μειονέκτημά τους είναι ότι κοστίζουν αρκετά χρήματα [4]. Επίσης το λογισμικό δεν ενημερώνεται τόσο τακτικά και δεν είναι επεκτάσιμο, δηλαδή ο ακροατής δεν μπορεί να προσθέσει περεταίρω μουσικές πηγές στο μέλλον. Σε περίπτωση βλάβης εκτός εγγύησης το κόστος επαναπόκτησης είναι μεγάλο και πολλές φορές η υποστήριξη είναι προβληματική είτε όσον αφορά το κόστος επισκευής είτε λόγω αδυναμίας επισκευής. Συνήθως διαθέτουν εφαρμογή για έλεγχο μέσω κινητού τηλεφώνου η οποία δεν μπορεί να αντικατασταθεί με άλλη ή να παραμετροποιηθεί ώστε να μπορεί κάποιος να φτιάξει ένα interface προσαρμοσμένο σε συγκεκριμένες απαιτήσεις όπως για παράδειγμα για άτομα με προβλήματα όρασης, το οποίο να έχει συγκεκριμένα κουμπιά με κατάλληλο μέγεθος και λειτουργίες, όπως φωνητικές εντολές. Το κόστος αυξάνει εάν θέλουμε να προμηθευτούμε streamers και για άλλα δωμάτια του σπιτιού. Οι εμπορικές εφαρμογές πολλές φορές απαιτούν δημιουργία προφίλ χρήστη, απαιτώντας προσωπικά στοιχεία του ακροατή. Ένα άλλο μειονέκτημα που δεν πρέπει να υποεκτιμηθεί είναι ότι για να ακούσεις μουσική διαδικτυακά θα πρέπει να διαθέτεις αξιοπρεπή σύνδεση στο διαδίκτυο. Στην αντίθετη περίπτωση το περιεχόμενο θα αναπαράγεται με διακοπές ή με καθυστέρηση (lag).

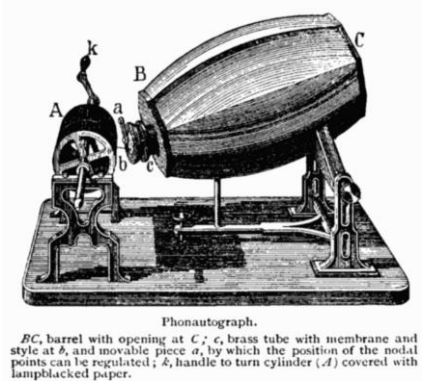
Στην αγορά υπάρχουν πολλές έτοιμες λύσεις οι οποίες απλά συνδέονται σε κάποιο στερεοφωνικό σύστημα αλλά λόγω των παραπάνω μειονεκτημάτων και κυρίως για το ότι κοστίζουν αρκετά χρήματα τα οποία αρκετοί άνθρωποι δεν μπορούν να διαθέσουν, θα εξετάσουμε την η εναλλακτική της κατασκευής ενός music streamer από την αρχή, με υλικά σε προσιτή τιμή και αρκετές ευκαιρίες για πειραματισμό [1] [5]

Στην παρούσα διπλωματική εργασία θα δείξουμε τον τρόπο με τον οποίο είναι δυνατό να κατασκευαστεί ένας μουσικός streamer με υλικά χαμηλού κόστους, και με πολλές δυνατότητες τροποποίησης, παραμετροποίησης και περεταίρω ανάπτυξης. Θα χρησιμοποιηθεί ένα Raspberry Pi 400 [6], τον ψηφιοαναλογικό μετατροπέα (DAC) και συγκεκριμένα τον IQAudio Pro DAC+ [7], ένα 40- Pin GPIO Ribbon Cable Extension [8] για τη διασύνδεση του DAC με το Raspberry Pi και το δωρεάν λογισμικό ανοιχτού κώδικα Mopidy [9], το οποίο είναι κατασκευασμένο από τον Stein Magnus Jodal και τους συνεισφέροντες (contributors) και τα πνευματικά δικαιώματα επίσης ανήκουν στον Stein Magnus Jodal και τους συνεισφέροντες (contributors). Επιπλέον είναι γραμμένο σε γλώσσα προγραμματισμού Python [10], στηρίζεται στο μοντέλο client - server [11] και χρησιμοποιεί το GStreamer [12] για την αναπαραγωγή του ήχου. Συγκεκριμένα θα αναπαράγει μουσική τόσο από τοπικούς εσωτερικούς και εξωτερικούς αποθηκευτικούς χώρους όπως usb sticks, όσο και από απομακρυσμένους εξυπηρετητές όπως το Youtube,

το Bandcamp [13], το TuneIn [14], το Internet Archive [15], διαδικτυακούς ραδιοφωνικούς σταθμούς [16] κ.α.

ΚΕΦΑΛΑΙΟ 1:
Θεωρητικό πλαίσιο του θέματος – Ανασκόπηση του πεδίου

Από τα αρχαία χρόνια η μουσική ήταν κάτι πολύ σημαντικό για τους ανθρώπους, οι οποίοι τη χρησιμοποιούσαν τη μουσική για διάφορους λόγους όπως θρησκευτικούς και τελετουργικούς λόγους, καθώς και για διασκέδαση και αναψυχή. Πριν την ανακάλυψη των τεχνικών ηχογράφησης, η αναπαραγωγή μουσικής ήταν εφικτή μόνο σε περιπτώσεις ζωντανής εκτέλεσης από μουσικούς, με συνέπεια μόνο πολύ λίγοι άνθρωποι να έχουν πρόσβαση σε μουσική. Τα πράγματα άλλαξαν με την έλευση των τεχνικών και των μηχανημάτων ηχογράφησης [17]. Συγκεκριμένα το 1857, ο Édouard-Léon Scott de Martinville εφηύρε μια συσκευή η οποία είχε τη δυνατότητα να ηχογραφήσει ήχο πάνω σε χαρτί καλυμμένο με αιθάλη και το οποίο ήταν τυλιγμένο πάνω σε ένα περιστρεφόμενο κύλινδρο, και την ονόμασε phonautograph. Το μειονέκτημα αυτής της εφεύρεσης ήταν ότι εκείνη την εποχή δεν υπήρχε δυνατότητα αναπαραγωγής [18].



Phonautograph.

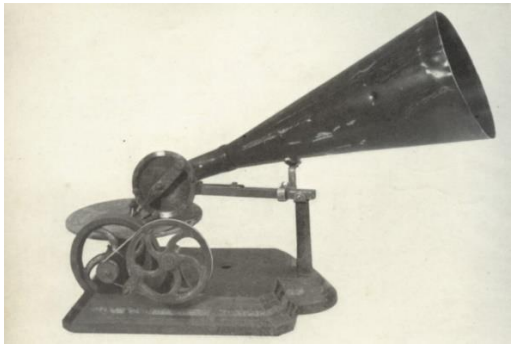
BC, barrel with opening at *C*; *c*, brass tube with membrane and style at *b*, and movable piece *a*, by which the position of the nodal points can be regulated; *k*, handle to turn cylinder (*A*) covered with lampblacked paper.

Εικόνα 1: Phonautograph **[19]**

Μερικά χρόνια αργότερα, και δη το 1877, ο Thomas Edison εφηύρε τον φωνογράφο (phonograph), ο οποίος είχε τη δυνατότητα εγγραφής ήχου αρχικά σε αλουμινόχαρτο τυλιγμένο σε ένα μεταλλικό κύλινδρο, και έπειτα ηχογράφηση σε κυλίνδρους από κερί. Η καινοτομία του φωνογράφου ήταν ότι εκτός της δυνατότητας ηχογράφησης, είχε και τη δυνατότητα αναπαραγωγής του ηχογραφημένου υλικού [20] [21] [22]

Εικόνα 2: Thomas Edison [23]

Ο Emil Berliner εφηύρε το γραμμόφωνο μεταξύ του 1887 and 1893. Αντί για εγγραφή σε κυλίνδρους, έκανε εγγραφή σε δίσκους οι οποίοι αρχικά ήταν καλυμμένοι με αιθάλη, και αργότερα με κερί. Οι δίσκοι αυτοί είχαν το πλεονέκτημα ότι μπορούσαν να αντιγραφούν με αποτέλεσμα τη μαζική παραγωγή δίσκων μουσικής και την εμπορική τους εκμετάλλευση. Οι δίσκοι του γραμμοφώνου είχαν επίσης το πλεονέκτημα ότι ήταν πιο πρακτικοί, φτηνότεροι να κατασκευαστούν και ο ήχος κατά την αναπαραγωγή ήταν δυνατότερος σε σύγκριση με τους κυλίνδρους του Edison. [22]



Εικόνα 3: Berliner Gramophone 1888 [24]

Εικόνα 4: Δίσκος γραμμοφώνου [25]

Με την έλευση των μικροφώνων, των ενισχυτών, των ηλεκτρομηχανικών καταγραφέων και των ηχείων, τα πράγματα έγιναν ακόμα ευκολότερα και τώρα ο ήχος μπορούσε να ενισχυθεί και να φιλτραριστεί, με αποτέλεσμα ακόμα μεγαλύτερη πιστότητα ήχου χρησιμοποιώντας τεχνικές ηχογράφησης με περισσότερα μικρόφωνα και πολυκάναλες ηχογραφήσεις σε δίσκους βινυλίου πλέον. [26]

Μετά το Β' Παγκόσμιο πόλεμο υιοθετήθηκε η τεχνολογία της μαγνητικής ταινίας, η οποία απέδιδε τον ήχο με ακόμα καλύτερα πιστότητα, και επίσης παρείχε τη δυνατότητα επεξεργασίας του περιεχομένου της ταινίας (tape editing), συνδυασμό των ηχογραφήσεων και πολυκάναλων ηχογραφήσεων. Η συνεχής εξέλιξη στην τεχνολογία του ήχου είχαν ως αποτέλεσμα να προκύψουν νέου τύπου συσκευές αναπαραγωγής και αποθήκευσης ήχου, όπως ο δίσκος διαμέτρου 12 ιντσών με ταχύτητα περιστροφής 33 στροφές το δευτερόλεπτο (LP), δίσκοι διαμέτρου 7 ιντσών και ταχύτητα περιστροφής 45 στροφές το δευτερόλεπτο, καθώς και κασέτες με μαγνητική ταινία. Τα αποθηκευτικά αυτά μέσα, εκτός από την πιστότητα ήχου, ήταν και πρακτικά και εύκολο να μεταφερθούν και να παραχθούν με αποτέλεσμα την άνθιση μίας νέας βιομηχανίας, της μουσικής βιομηχανίας. [27]

Εικόνα 5: Κασέτες [28]

Εικόνα 6 : Δίσκος 45 στροφών **[29]**

Εικόνα 7: Δίσκος LP **[30]**

Η επόμενη φάση της μουσικής εξέλιξης είναι η λεγόμενη Ψηφιακή εποχή, η οποία έχει κυριαρχήσει, προσφέροντας τα καλύτερα δυνατά αποτελέσματα από πλευράς πιστότητας ήχου. Πλέον η μουσική δεν ηχογραφείται με αναλογικό τρόπο, αλλά ψηφιακά όπου ο ήχος δειγματοληπτείται με μεθόδους όπως πχ παλμοκωδική διαμόρφωση PCM σε μεγάλο αριθμό δειγμάτων. Τα δείγματα αυτά όταν επανασυνδυαστούν, παράγουν μια συνεχή ροή ήχου. Πλέον η μουσική αποθηκεύεται σε ένα νέο φορμάτ, το CD (compact disk), το οποίο σταδιακά αντικατέστησε το βινύλιο και τις κασέτες και κυριάρχησε στη μουσική βιομηχανία. Τα CD είναι μικρά σε μέγεθος και μεταφέρονται εύκολα και όχι τόσο ευπαθή σε κακουχίες όσο οι δίσκοι βινυλίου, και η ποιότητά τους δεν αλλοιώνεται με τον χρόνο όπως στην περίπτωση των κασετών, διαθέτουν μεγάλη δυναμική περιοχή (˜96dB) και εξασφαλίζουν άριστη ποιότητα και καθαρό ήχο, χωρίς παραμορφώσεις. Ένα άλλο πλεονέκτημα των CD έναντι των δίσκων βινυλίου είναι ότι ενώ είναι σαφώς μικρότερα σε διαστάσεις, μπορούν να εμπεριέχουν μουσική μεγαλύτερης συνολικής διάρκειας, με μέγιστη διάρκεια τα 80 λεπτά της ώρας.

Το CD είχε κυριαρχήσει ως βασικό μέσο αποθήκευσης μουσικής μέχρι τα τέλη της προηγούμενης χιλιετίας, ώσπου μια νέα μορφή ήχου, το ψηφιακό αρχείο ήχου όπως πχ το .mp3, το .wav κλπ., έκανε την εμφάνιση του. Τα αρχεία αυτά μπορούσαν να έχουν μειωμένο μέγεθος επειδή εκμεταλλεύονταν τις νεοεμφανιζόμενες τεχνικές και αλγορίθμους συμπίεσης, χωρίς να χάνουν σημαντικά σε ποιότητα σε σχέση με το CD. Αυτό όμως το γεγονός σε συνδυασμό με την ευρεία διάδοση του διαδικτύου, οδήγησε σε παράνομη διακίνηση μουσικών κομματιών μέσω διαδικτύου με τη χρήση υπηρεσιών ανταλλαγής αρχείων όπως το Napster, το eMule κλπ., με αποτέλεσμα οι καλλιτέχνες και οι δισκογραφικές εταιρίες να χάνουν κέρδη από πνευματικά δικαιώματα. Από την άλλη πλευρά όμως, τα ψηφιακά αρχεία ήχου είναι πάρα πολύ πρακτικά, γιατί μπορούν και να αγοραστούν διαδικτυακά από ενδιαφερόμενους ακροατές, ή να ψηφιοποιήσουν οι ίδιοι τις συλλογές τους από βινύλια, κασέτες και CD, και να τις αποθηκεύσουν σε αποθηκευτικά μέσα όπως σκληροί δίσκοι ή usb sticks, και να έχουν άμεση πρόσβαση στο σύνολο της συλλογής τους. [31]

Εικόνα 8: Δίσκος CD **[32]**

Παράλληλα κάνει την εμφάνισή του και το ραδιόφωνο μέσω διαδικτύου (Internet Radio). Το Internet radio είναι μία υπηρεσία ήχου που χρησιμοποιεί το διαδίκτυο ως μέσο μεταφοράς της μετάδοσης αντί των παραδοσιακών ραδιοκυμάτων. Είναι και αυτό μια υπηρεσία streaming, όπου πολλές φορές το περιεχόμενο του μεταδίδεται ζωντανά, και δεν είναι προηχογραφημένο. Οι παραδοσιακοί ραδιοφωνικοί σταθμοί συνηθίζουν να μεταδίδουν διαδικτυακά το ζωντανό τους πρόγραμμα παράλληλα με τη μετάδοση τους μέσω ραδιοκυμάτων. Η τεχνική αυτή παράλληλης μετάδοσης ονομάζεται simulcast. Υπάρχουν επίσης και ραδιοφωνικοί σταθμοί οι οποίοι είναι καθαρά διαδικτυακοί, όπου ο καθένας με ένα προσωπικό ηλεκτρονικό υπολογιστή, μικρόφωνο και σύνδεση στο διαδίκτυο, μπορεί να παράξει το δικό του ραδιοφωνικό πρόγραμμα και να το μεταδώσει μέσω διαδικτύου. [33]

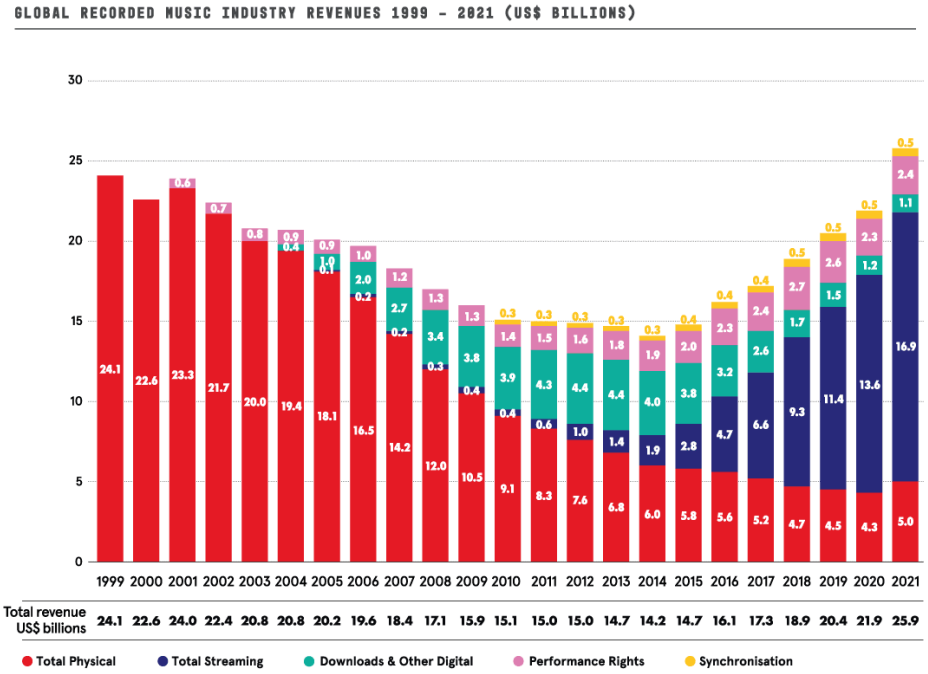
Και ερχόμαστε στη σημερινή εποχή όπου πλέον έχουν επικρατήσει νέοι τρόποι κατανάλωσης μουσικού περιεχομένου, και δεν είναι άλλες από τις υπηρεσίες streaming. Απομακρυσμένοι εξυπηρετητές όπως το Spotify [3], το YouTube, το SoundCloud [34] κ.α. παρέχουν μουσικό περιεχόμενο είτε δωρεάν, αλλά με κάποιους περιορισμούς ως προς το παρεχόμενο περιεχόμενο (free) ή παρεχόμενες ευκολίες όπως αποθήκευση λιστών αναπαραγωγής ή αναπαραγωγή ακόμα και όταν βρισκόμαστε εκτός σύνδεσης, είτε με μια μηνιαία συνδρομή, χωρίς περιορισμούς (premium account). Το μοντέλο αυτό που συνδυάζει δωρεάν παροχές με επί πληρωμή παροχές ονομάζεται freemium business model [35]. Το μεγάλο πλεονέκτημα των streaming υπηρεσιών είναι ότι ο χρήστης δεν χρειάζεται πλέον να αγοράζει το φυσικό μέσο όπως το CD ή το δίσκο βινυλίου και να διαθέτει τα μουσικά κομμάτια, αλλά δωρεάν ή με ένα μικρό αντίτιμο κάθε μήνα να ακούει τη μουσική της επιλογής του, από το κινητό του τηλέφωνο ή τον υπολογιστή του.

Με μια μηνιαία συνδρομή μπορούν να μετατραπούν σε premium και να μπορεί κάποιος να ακούει μουσική χωρίς ενοχλητικές διαφημίσεις. Να κατεβάζει μουσική στο κινητό του και να την ακούει ακόμα και όταν βρίσκεται εκτός σύνδεσης, σε ποιότητα που μπορεί να φτάσει και στα 320 Kbps. Διατίθενται ατομικά πακέτα, οικογενειακά και φοιτητικά. [36] [37].

Η μουσική είναι διαθέσιμη παντού και οποιαδήποτε στιγμή, και οι ακροατές έχουν πρόσβαση σε απεριόριστη ακρόαση και έχουν τη δυνατότητα να ανακαλύπτουν νέα μουσική σε όλα τα είδη της μουσικής. Επίσης μπορούν να ενημερώνονται για τις τελευταίες εξελίξεις στη μουσική, όσον αφορά νέους καλλιτέχνες και τρέχουσες επιτυχίες.

Στο παρακάτω γράφημα της Διεθνούς Ομοσπονδίας Φωνογραφικής Βιομηχανίας (IFPI) [38] βλέπουμε πώς άλλαξε η πηγή των κερδών της μουσικής βιομηχανίας τις τελευταίες

δύο δεκαετίες. Από το διάγραμμα είναι ξεκάθαρο ότι από το 2017 και έπειτα τα έσοδα από τις υπηρεσίες streaming ξεπεράσανε αυτά από τις πωλήσεις μουσικής σε φυσική μορφή (CD, LP κλπ.). Αυτό δείχνει ότι οι υπηρεσίες streaming πλέον κυριαρχούν όσον αφορά την ακρόαση μουσικής.



Εικόνα 9: Κέρδη της παγκόσμιας μουσικής βιομηχανίας 1999 - 2021 **[38]**

Από την άλλη, οι ακροατές δεν κατέχουν τη μουσική όπως αν αγόραζαν ένα άλμπουμ σε φυσική μορφή, εάν δεν πληρώσουν για κάποια premium συνδρομή στην ακρόαση παρεμβάλλονται ενοχλητικές διαφημίσεις και τίθεται και ζήτημα όσον αφορά εάν τελικά οι καλλιτέχνες πληρώνονται επαρκώς για τη μουσική που δημιουργούν. Αρκετοί είναι οι καλλιτέχνες που διαμαρτύρονται για τον τρόπο με τον οποίο πληρώνονται μέσω της διάθεσης της μουσικής τους από τις πλατφόρμες διότι θεωρούν ότι δεν είναι δίκαιος. [39] [40]

Επίσης μία μελέτη των Joel Waldfogel and Luis Aguiar το 2015 καταλήγει ότι το Spotify μπορεί να περιορίσει τη μουσική πειρατεία, αλλά τα έσοδα που αποκτώνται από το streaming της μουσικής αντισταθμίζονται από τις απώλειες εσόδων που θα προέκυπταν από την αγορά ψηφιακών μουσικών αρχείων. [41]

Η περίπτωση του Bluetooth

Εύλογα θα μπορούσε να αναρωτηθεί κάποιος γιατί από τη στιγμή που όλα τα έξυπνα κινητά πλέον διαθέτουν την δυνατότητα σύνδεσης με Bluetooth, καθώς επίσης και οι περισσότερες συσκευές αναπαραγωγής όπως τα ηχοσυστήματα hi-fi, ραδιόφωνα, ακουστικά και ηχεία Bluetooth, οπότε θα μπορούν να συνδεθούν μεταξύ τους ασύρματα μέσω Bluetooth υλοποιώντας μια point–to-point σύνδεση και κάνοντας λήψη της μουσικής από απομακρυσμένους εξυπηρετητές με τις κατάλληλες εφαρμογές ή αναπαραγωγή μουσικής αποθηκευμένης στην μνήμη του κινητού, να απολαμβάνουν τη μουσική χωρίς την ανάγκη ενδιάμεσων λύσεων όπως ένα music streamer. Η απάντηση στο ερώτημα αυτό είναι ότι η υλοποίηση αυτή, αν και ανέξοδη και με κάποια πλεονεκτήματα, παρουσιάζει και κάποια σημαντικά μειονεκτήματα. Μερικά πλεονεκτήματα της χρήσης του Bluetooth είναι τα εξής [42]:

- Απαλλασσόμαστε από την χρήση καλωδίων. Με το Bluetooth μπορούμε να στέλνουμε και να λαμβάνουμε αρχεία και να μεταφέρουμε μουσική από το κινητό τηλέφωνο σε εμβέλεια γύρω στα 10 μέτρα [43], χωρίς την παρεμβολή καλωδίων.

- Η δυνατότητα σύνδεσης Bluetooth συναντάται πλέον σε μεγάλο αριθμό συσκευών, καθώς και στο σύνολο των κινητών τηλεφώνων και τάμπλετ και είναι πολύ εύκολο να χρησιμοποιηθεί ακόμα και από άτομα με ελάχιστη εξοικείωση στην τεχνολογία, γιατί απλά χρειάζεται ενεργοποίηση της λειτουργίας και στις δύο συσκευές που πρόκειται να συνδεθούν, και πιθανόν εισαγωγή κάποιου κωδικού ασφαλείας.

- Ένα ακόμα πλεονέκτημα της τεχνολογίας Bluetooth είναι ότι καταναλώνει χαμηλή ισχύ ειδικά με τη χρήση του στάνταρντ Bluetooth Low Energy (LE), οπότε επιτυγχάνεται εξοικονόμηση της μπαταρίας του κινητού τηλεφώνου ή του τάμπλετ [44].

Όμως η χρήση της τεχνολογίας Bluetooth παρουσιάζει και μειονεκτήματα. Συγκεκριμένα:

- Οι συνδέσεις Bluetooth δέχονται παρεμβολές από συσκευές όπως wi-fi routers και άλλες συσκευές που λειτουργούν στο ίδιο φάσμα συχνοτήτων, ασύρματα τηλέφωνα, φούρνους μικροκυμάτων καθώς και εμπόδια που παρεμβάλλονται μεταξύ των συνδεδεμένων συσκευών. [45]

- Σε περίπτωση που μεταφέρουμε (streaming) μουσική από το κινητό μας μέσω Bluetooth σε μία συσκευή αναπαραγωγής μουσικής και το κινητό απομακρυνθεί

τόσο από τη συσκευή αυτή ώστε να βγει εκτός εμβέλειας, η σύνδεση θα διακοπεί με αποτέλεσμα να διακοπεί και η αναπαραγωγή της μουσικής. [46]

- Το Bluetooth είναι ευάλωτο όσον αφορά κακόβουλες επιθέσεις οπότε δεν θεωρείται ασφαλής τρόπος σύνδεσης, γιατί μπορεί κάποιος να αποκτήσει πρόσβαση σε προσωπικά μας δεδομένα. [47]

- Κατά τη διάρκεια της μεταφοράς μουσικής από το κινητό σε μία συσκευή μέσω Bluetooth, δεν είναι δυνατή εισερχόμενη ή εξερχόμενη κλήση χωρίς τη διακοπή της αναπαραγωγής της μουσικής.

Raspberry Pi υλοποίηση

Επιλέγοντας την υλοποίηση μουσικού streamer με Raspberry Pi, ο ενδιαφερόμενος μπορεί να δοκιμάσει από μια πληθώρα επιλογών όσον αφορά τους μικροεπεξεργαστές, εξαρτήματα, DACs και εφαρμογές ανοιχτού κώδικα, να πειραματιστεί και να παραμετροποιήσεις, και αν διαθέτει γνώσεις προγραμματισμού, να κατασκευάσει τη δική του εφαρμογή ή να τροποποιήσει τις ήδη υπάρχουσες. Επιτυγχάνουμε χαμηλότερη κατανάλωση ρεύματος όταν χρησιμοποιούμε Raspberry [48], το Raspberry μπορεί να χρησιμοποιηθεί και για άλλους σκοπούς εκτός από το να γίνεται αναπαραγωγή μουσικής όπως για πλοήγηση στο διαδίκτυο, εφαρμογές γραφείου ή παιχνίδια και μπορεί ο καθένας ανάλογα με τη φαντασία του να το εσωκλείσει σε ένα κουτί δικής του εμπνεύσεως και κατασκευής, να του εφαρμόσει touch screen οθόνη ή LCD display ή να προσθέσει αναλογικά κουμπιά χειρισμού [49].

Το αρνητικό με την υλοποίηση με Raspberry είναι ότι επειδή πρόκειται για υλοποιήσεις που πραγματοποιούνται από προγραμματιστές οι οποίοι είναι λάτρεις της μουσικής και που έχουν τάση για πειραματισμό, οι οποίοι όμως δεν έχουν την υποστήριξη κάποιας εταιρίας που να χρηματοδοτεί το όλο εγχείρημα και επειδή συνήθως αυτές οι υλοποιήσεις δεν έχουν ως σκοπό το κέρδος αφού είναι ανοιχτού κώδικα, εθελοντικά σπαταλώντας προσωπικό χρόνο, είναι πολύ πιθανό ο προγραμματιστής να εγκαταλείψει την προσπάθεια κάποια στιγμή οπότε να μην παρέχει πλέον αναβαθμίσεις στο λογισμικό και κάποια στιγμή να υπολειτουργεί ή να μην λειτουργεί καθόλου. Το πιο πιθανό να μην υπάρχει λεπτομερές documentation, να είναι αναξιόπιστο, κίνδυνοι από κακόβουλες επιθέσεις σε περίπτωση που χρειάζεται να εισάγουμε ευαίσθητα στοιχεία όπως όνομα χρήστη, κωδικό ασφαλείας και στοιχεία τραπεζικής κάρτας γιατί ακόμα και αν ο κώδικας είναι ανοιχτός και ορατός για όλους, οι χρήστες οι οποίοι δεν γνωρίζουν από προγραμματισμό δεν μπορούν να καταλάβουν τί ακριβώς κάνει το λογισμικό αυτό. [50] [51].

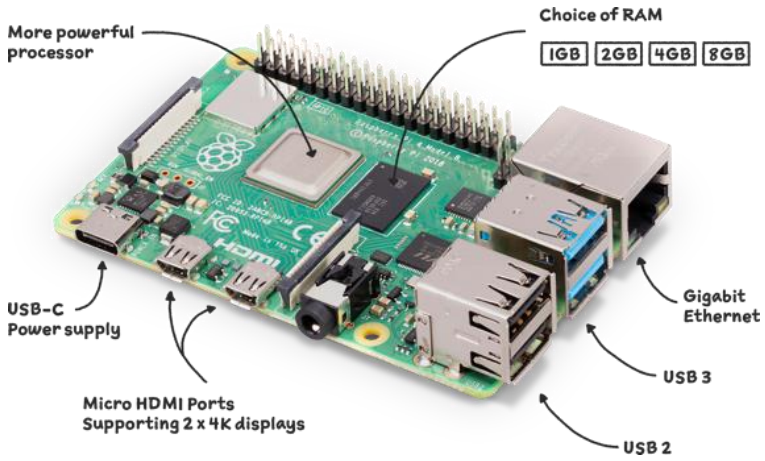
Επειδή οι υλοποιήσεις σε Raspberry Pi τρέχουν σε περιβάλλον Raspberry Pi OS, δηλαδή σε μια έκδοση Linux, ο ακροατής είναι πιθανόν να μην είναι εξοικειωμένος με Linux και ειδικότερα με εκτέλεση εντολών κονσόλας τερματικού. Επειδή το λειτουργικό Raspberry Pi OS εγκαθίσταται σε κάρτα micro SD, υπάρχει κίνδυνος να καταστραφεί η κάρτα αυτή λόγω του ότι έχει πεπερασμένο αριθμό εγγραφών και να σταματήσει να λειτουργεί η υλοποίηση. [52]

ΚΕΦΑΛΑΙΟ 2:
Η προτεινόμενη μέθοδος – Θεμελίωση, Σχεδίαση, Ανάπτυξη

2.1 Hardware

2.1.1 Raspberry Pi

Το Raspberry Pi [53] είναι ένας πλήρης λειτουργικός ηλεκτρονικός υπολογιστής χαμηλού κόστους, εξολοκλήρου χτισμένος σε μία πλακέτα (SBC) μεγέθους όσο περίπου μια πιστωτική κάρτα, το οποίο συνδέεται σε μια οθόνη υπολογιστή ή τηλεόραση μέσω HDMI. Επιτρέπει σε άτομα όλων των ηλικιών να ανακαλύψουν τους ηλεκτρονικούς υπολογιστές, και να πειραματιστούν με γλώσσες προγραμματισμού όπως η Scratch και η Python. Είναι ικανό να εκτελέσει ό,τι λειτουργία εκτελούμε και με ένα υπολογιστή, όπως περιήγηση στο διαδίκτυο, αναπαραγωγή μουσικής και βίντεο, κειμενογράφο καθώς και ηλεκτρονικά παιχνίδια [54].



Εικόνα 10 : Raspberry Pi 4 model B **[55]**

Ο αρχικός του σκοπός ήταν να προάγει την επιστήμη των υπολογιστών σε σχολεία και αναπτυσσόμενες χώρες [56], αλλά λόγω της χαμηλής του τιμής, της υιοθέτησης HDMI και USB, καθώς και των αμέτρητων επιλογών χρήσης του, τελικά έγινε δημοφιλές μεταξύ

των χομπίστων ηλεκτρονικών και βρήκε εφαρμογή και σε άλλους κλάδους όπως πχ στη ρομποτική.

2.1.1.1 Υπάρχοντα μοντέλα

Στην αγορά διατίθεται η σειρά Raspberry Pi, το Raspberry Pi Zero και το Raspberry Pi Pico. Παρακάτω παρατίθενται τα μοντέλα που αντιπροσωπεύουν την κάθε σειρά, καθώς και τεχνικά τους χαρακτηριστικά [57] [58] [59]. Τα μοντέλα αυτά θα συνεχίζουν να παράγονται έως τουλάχιστον τον Ιανουάριο του 2026.

Raspberry Pi 1 Model A+ και B+

Το Model A+ είναι το φθηνότερο μοντέλο της σειράς, και αντικατέστησε το αρχικό Model A το 2014. Η διαφορά του με το αρχικό μοντέλο είναι ότι έχει 40 ακίδες που χρησιμεύουν ως είσοδοι/έξοδοι γενικού σκοπού (GPIO) αντί για 26. Διαθέτει θύρα υποδοχής micro SD, χαμηλή κατανάλωση της τάξεως των 0.5W έως 1W, αποδίδει καλύτερης ποιότητας ήχο γιατί το κύκλωμα ήχου ενσωματώνει αποκλειστική παροχή ισχύος χαμηλού θορύβου. Έχει 512 MB μνήμη RAM, Broadcom BCM2711 SoC με επεξεργαστή ARM1176JZF-S στα 700 MHz, έξοδο HDMI και Composite video, έξοδο ήχου 3.5mm audio jack και δεν έχει δυνατότητα σύνδεσης σε δίκτυο. Επίσης διαθέτει διεπαφές για σύνδεση κάμερας (CSI), Display (DSI) και διπύρηνο επεξεργαστή γραφικών VideoCore IV 3D στα 250 MHz.

Το Model B+ έχει τα ίδια χαρακτηριστικά με το A+, αλλά διαθέτει και 4 θύρες USB 2.0 και θύρα RJ45 10/100 MBits/s Ethernet, για ενσύρματη σύνδεση σε δίκτυο.

Raspberry Pi 2 Model B

Το Raspberry Pi 2 Model B αντικατέστησε το Raspberry Pi 1 Model B+ τον Φεβρουάριο του 2015. Σε σύγκριση με το Raspberry Pi 1 διαθέτει 1 GB RAM, Broadcom BCM2836 SoC με τετραπύρηνο επεξεργαστή ARM Cortex-A7 CPU στα 900 MHz.

Raspberry Pi 3 Model B

Το μοντέλο αυτό αντικατέστησε το Raspberry Pi 2 Model B τον Φεβρουάριο του 2016. Πλέον διαθέτει Broadcom BCM2837 SoC με τετραπύρηνο επεξεργαστή ARM Cortex-A53

στα 1200 MHz, διπύρηνο επεξεργαστή γραφικών VideoCore IV στα 400 MHz όπως και δυνατότητα ασύρματης σύνδεσης 2.4GHz 802.11n και Bluetooth 4.1.

Raspberry Pi 3 Model A+

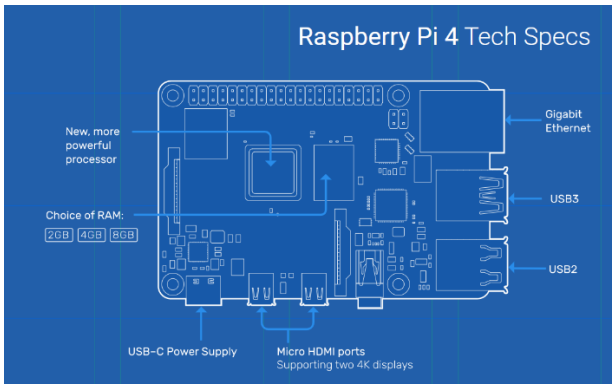
Το μοντέλο αυτό είναι η compact έκδοση του Raspberry Pi 3 Model B και ανακοινώθηκε το 2018. Διαθέτει 512 MB RAM, μόνο μία θύρα USB 2.0 και έχει δυνατότητα μόνο για ασύρματη σύνδεση στα 2.4GHz και 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE (Bluetooth Low Energy) και επίσης διαθέτει Broadcom BCM2837B0 SoC με τετραπύρηνο επεξεργαστή 64-bit ARM Cortex-A53 CPU στα 1.4GHz.

Raspberry Pi 3 Model B+

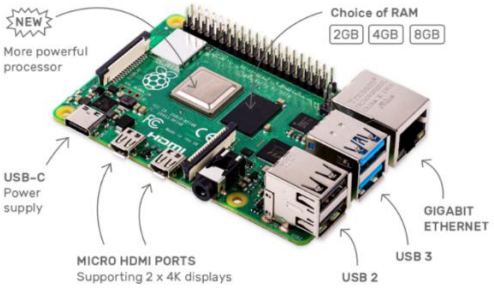
Το μοντέλο αυτό είναι η εξέλιξη του Raspberry Pi 3 Model B και ανακοινώθηκε το 2018. Σε σχέση με το προηγούμενο μοντέλο, οι διαφορές τους είναι ότι διαθέτει Broadcom BCM2837B0 SoC με τετραπύρηνο επεξεργαστή 64-bit ARM Cortex-A53 CPU στα 1.4GHz, 2.4GHz και 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE (Bluetooth Low Energy), Gigabit Ethernet over USB 2.0 , υποστήριξη Power-over-Ethernet (PoE) (με τη χρήση ξεχωριστού PoE HAT).

Raspberry Pi 4 Model B

Ανακοινώθηκε τον Ιούνιο του 2019 και είναι ότι πιο κοντινό σε ηλεκτρονικό υπολογιστή όσον αφορά την απόδοση. Οι κύριες διαφορές του σε σχέση με τα προηγούμενα μοντέλα είναι ότι βγαίνει σε εκδόσεις με 1, 2, 4 και 8 GB RAM, διαθέτει Broadcom BCM2711 SoC με τετραπύρηνο επεξεργαστή 64-bit στα 1.5GHz ARM Cortex-A72 CPU, Full-throughput Gigabit Ethernet, Bluetooth 5.0, δύο θύρες USB 3.0 και δύο θύρες USB 2.0, έξοδο εικόνας σε δύο οθόνες με σύνδεση micro HDMI και υποστήριξη ανάλυσης βίντεο έως και 4K, υποστήριξη OpenGL ES 3.x, τροφοδοσία μέσω USB type C και πλήρη συμβατότητα με τα παλαιότερα μοντέλα.



Εικόνα 11: Raspberry Pi 4 Tech Specs [**60**]



Εικόνα 12: Raspberry Pi 4 model B [**55**]

Raspberry Pi 400

Επίσης υπάρχει και η έκδοση Raspberry Pi 400 [6] η οποία είναι ένας ολοκληρωμένος υπολογιστής, μέσα σε ένα πληκτρολόγιο. Οι διαφορές του σε σύγκριση με το Raspberry Pi 4 είναι ότι περιλαμβάνει 64-bit Broadcom BCM2711 SoC με τετραπύρηνο επεξεργαστή Cortex-A72 (ARM v8) στα 1.8GHz και ότι έχει δύο θύρες USB 3.0 αλλά μία θύρα USB 2.0 . Διατίθεται στην αγορά είτε ως αυτόνομη μονάδα, είτε σε μορφή κιτ, όπου περιλαμβάνει και όλα τα παρελκόμενα που χρειάζονται όπως USB ποντίκι, καλώδιο micro HDMI σε HDMI, κάρτα micro SD με προεγκατεστημένο το λειτουργικό σύστημα Raspberry Pi OS, τροφοδοσία 5V USB type C και ένα βιβλίο - οδηγό για αρχάριους, σχετικά με Raspberry Pi 400.

Εικόνα 13: Raspberry Pi 400 [**6**]

Raspberry Pi Zero [61]

Το Raspberry Pi Zero είναι ένα μικροσκοπικό Raspberry Pi, σε πολύ προσιτή τιμή. Τα κύρια χαρακτηριστικά του είναι ότι περιλαμβάνει ένα μονοπύρηνο επεξεργαστή στα 1 GHz, 512 MB RAM, mini HDMI θύρα, micro USB OTG θύρα, τροφοδοσία micro -USB, επέκταση (HAT) 40 ακίδων εισόδου/εξόδου γενικού σκοπού (GPIO), Composite video και CSI camera connector.

Εικόνα 14: Raspberry Pi Zero [**61**]

Raspberry Pi Zero W

Το Raspberry Pi Zero W [62] είναι η βελτιωμένη έκδοση του Raspberry Pi Zero, όπου έχει προστεθεί δυνατότητα ασύρματης σύνδεσης και σύνδεσης Bluetooth. Συγκεκριμένα τα πρόσθετα χαρακτηριστικά είναι 802.11 b/g/n wireless LAN, Bluetooth 4.1 και Bluetooth Low Energy (BLE).

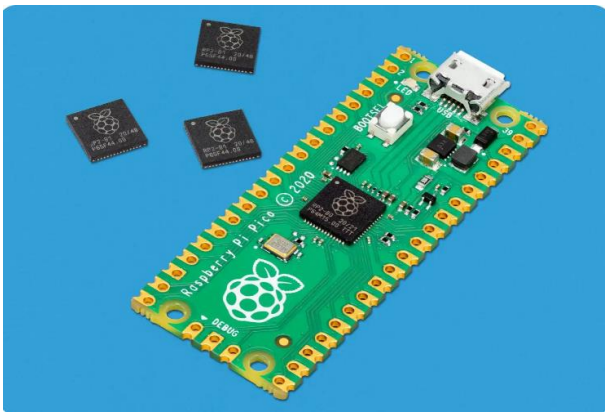
Εικόνα 15: Raspberry Pi Zero W **[62]**

Raspberry Pi Zero 2 W

Εικόνα 16: Raspberry Pi Zero 2 W **[63]**

Το Raspberry Pi Zero 2 W [63] είναι το πιο ισχυρό μοντέλο της σειράς Zero, με κυριότερες διαφορές από τα προηγούμενα μοντέλα τον τετραπύρηνο επεξεργαστή 64-bit Arm Cortex-A53 CPU στο 1 GHz, Bluetooth 4.2, CSI-2 camera connector, H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30) και OpenGL ES 1.1, 2.0 graphics.

Raspberry Pi Pico



Εικόνα 17: Raspberry Pi Pico Tech Specs **[64]**

Το Raspberry Pi Pico [64] είναι μικρό σε μέγεθος, οικονομικό, γρήγορο και με πολλές εφαρμογές και βασίζεται στον μικροελεγκτή RP2040. Περιλαμβάνει διπύρηνο επεξεργαστή Arm Cortex-M0+ το ρολόι του οποίου τρέχει στα 133 MHz. Έχει 264KB on-chip SRAM και 2MB on-board QSPI Flash, 26 ακίδες GPIO συμπεριλαμβανομένων τριών αναλογικών εισόδων, 8 προγραμματιζόμενες I/O PIO καταστάσεις μηχανής (state machines) , 2 × UART, 2 × SPI controllers, 2 × I2C controllers, 16 × PWM channels , 1 × USB 1.1 controller και PHY.

2.1.1.2 Ενδεικτικές τιμές

Στους παρακάτω πίνακες παρατίθενται ενδεικτικές τιμές στην ελληνική αγορά, στο κατάστημα του επίσημου μεταπωλητή του Raspberry Pi στην Ελλάδα Nettop [65]

Πίνακας 1: τιμές Raspberry Pi 1 και 2

Μοντέλο	Raspberry Pi Model A+ 512MB RAM	Raspberry Pi Model B+	Raspberry Pi 2 Quad Core CPU, 1GB RAM
Τιμή	28 €	50 €	55 €

Πίνακας 2: τιμές Raspberry Pi 3

Μοντέλο	Raspberry Pi 3 Model A+	Raspberry Pi 3 Model B Quad Core CPU 64 Bit, 1GB RAM	Raspberry Pi 3 Model B+ Quad Core CPU 64 Bit 1.4GHz, 1GB RAM
Τιμή	28 €	43 €	39,90 €

Πίνακας 3: τιμές Raspberry Pi 4 και 400

Μοντέλο	Raspberry Pi 4 Model B/1GB	Raspberry Pi 4 Model B/2GB	Raspberry Pi 4 Model B/4GB	Raspberry Pi 4 Model B/8GB	Raspberry Pi 400 Personal Computer
Τιμή	41,90 €	50,90 €	63,90 €	84,50 €	76,50 €

Πίνακας 4: τιμές Raspberry Pi Zero και W

Μοντέλο	Raspberry Pi Zero	Raspberry Pi Zero W	Raspberry Pi Zero W with Headers	Raspberry Pi Zero 2 W
Τιμή	6.50 €	11.90 €	16.30 €	17 €

Πίνακας 5: τιμές Raspberry Pi Pico

Μοντέλο	Raspberry Pi Pico without Headers	Raspberry Pi Pico with Headers
Τιμή	4.50 €	5.90 €

2.1.1.3 Raspberry Pi GPIO pinout

Στον παρακάτω πίνακα απεικονίζονται ο αριθμός των ακίδων του GPIO των τελευταίων μοντέλων Raspberry, τα οποία πλέον διαθέτουν 40 ακίδες αντί για 26 που είχαν τα πρώτα μοντέλα, καθώς και γίνεται περιγραφή της λειτουργίας τους.

Πίνακας 6: Ακίδες του GPIO του Raspberry Pi

Περιγραφή	Ακίδα				Ακίδα	Περιγραφή
3.3V	1				2	5V
GPIO 2	3				4	5V
GPIO 3	5				6	Γείωση
GPIO 4	7				8	GPIO 14
Γείωση	9				10	GPIO 15
GPIO 17	11				12	GPIO 18
GPIO 27	13				14	Γείωση
GPIO 22	15				16	GPIO 23
3.3V	17				18	GPIO 24
GPIO 10	19				20	Γείωση
GPIO 9	21				22	GPIO 25
GPIO 11	23				24	GPIO 8
Γείωση	25				26	GPIO 7
GPIO 0 (ID EEPROM)	27				28	GPIO 1 (ID EEPROM)
GPIO 5	29				30	Γείωση
GPIO 6	31				32	GPIO 12
GPIO 13	33				34	Γείωση
GPIO 19	35				36	GPIO 16
GPIO 26	37				38	GPIO 20
Γείωση	39				40	GPIO 21

Δύο από τις ακίδες αυτές παρέχουν τάση 5V, και δύο 3,3V. Επίσης υπάρχουν οκτώ ακίδες που η χρησιμότητά τους είναι ως γείωση (ground) στα 0V. Οι υπόλοιπες ακίδες είναι είσοδοι/έξοδοι γενικού σκοπού

Όπως παρατηρούμε από τον παραπάνω πίνακα, το Raspberry Pi διαθέτει δύο ακίδες για τροφοδοσία τάσης 3.3V, δύο ακίδες για τροφοδοσία τάσης 5V, οκτώ ακίδες που χρησιμεύουν για γείωση και είκοσι οκτώ ακίδες εισόδου/εξόδου γενικού σκοπού (GPIO).

Κάποιες από τις ακίδες GPIO μπορούν να χρησιμοποιηθούν και για άλλους σκοπούς, εκτός από είσοδο και έξοδο. Συγκεκριμένα :

Ακίδα 3 και ακίδα 5 : είναι ακίδες I2C οι οποίες επιτρέπουν την επικοινωνία του Raspberry Pi μέσω επικοινωνίας two-wire με μια πληθώρα εξωτερικών συσκευών και αισθητήρων.

Ακίδα 7 : Ρολόι γενικού σκοπού (General Purpose Clock). Μπορεί να δώσει στην έξοδο μια σταθερή συχνότητα χωρίς να ελέγχεται μέσω λογισμικού.

Ακίδα 8 και 10 : Με τις ακίδες αυτές μπορούμε να επικοινωνήσουμε μέσω σειριακής επικοινωνίας UART (Universal asynchronous receiver / transmitter) με συσκευές που το υποστηρίζουν. Η ακίδα 8 χρησιμοποιείται για την εκπομπή TX ενώ η ακίδα 10 για λήψη δεδομένων RX.

Ακίδες 12, 35, 38 και 40 : Οι ακίδες αυτές είναι ακίδες PCM και χρησιμοποιούνται για την ψηφιακή έξοδο ήχου διαμορφωμένου με παλμοκωδική διαμόρφωση (PCM), που οδηγείται σε κάποιο μετατροπέα ψηφιακού ήχου σε αναλογικό (DAC). Η ακίδα 12 παρέχει σήμα χρονισμό σε εξωτερικές συσκευές όπως πχ DAC. Η ακίδα 35 παρέχει σήμα συγχρονισμού frame-sync στις εξωτερικές συσκευές. Η ακίδα 38 λειτουργεί ως είσοδος δεδομένων ήχου από εξωτερικές συσκευές ήχου, όπως ένα I2C μικρόφωνο. Τέλος η ακίδα 40 χρησιμεύει ως έξοδος δεδομένων ήχου σε κάποια εξωτερική συσκευή.

Ακίδες 11, 12, 19, 21, 23, 24, 26, 35, 36, 38 και 40 : Αυτές οι ακίδες βοηθούν στο να συνδέονται εξωτερικές συσκευές συμβατές με το Raspberry Pi μέσω του Serial Peripheral Interface, αλλιώς SPI.

Ακίδες 12, 32, 33 και 35 : Ακίδες που χρησιμοποιούνται για να εκμεταλλευτούμε την τεχνική ελέγχου ισχύος PWM (Pulse Width Modulation) με απώτερο σκοπό τον έλεγχο ισχύος που οδηγείται σε ένα κινητήρα, ώστε να επηρεάζουμε την ταχύτητα περιστροφής του.

Ακίδες 27 και 28 : Οι δύο αυτές ακίδες είναι συνήθως δεσμευμένες για I2C διασύνδεση με πρόσθετες συσκευές HAT (Hardware Attached On Top). [66] [67] [68] [69] [70] [71]

2.1.2 40 Pin GPIO Male to Female Ribbon Cable

Το καλώδιο αυτό χρησιμοποιείται ώστε να επεκτείνουμε το GPIO πάνω στο Raspberry Pi. Το θηλυκό βύσμα συνδέεται με το GPIO, ενώ στο αρσενικό βύσμα μπορούμε να συνδέσουμε HATs της επιλογής μας. Θέλει πολλή μεγάλη προσοχή ώστε το αρσενικό βύσμα να συνδεθεί σωστά, σύμφωνα με τη διάταξη των ακίδων, και όχι ανάποδα γιατί στην περίπτωση αυτή θα προκληθεί ζημιά στο Raspberry Pi αλλά και στο HAT.

Εικόνα 18: 40 Pin GPIO Male to Female Ribbon Cable

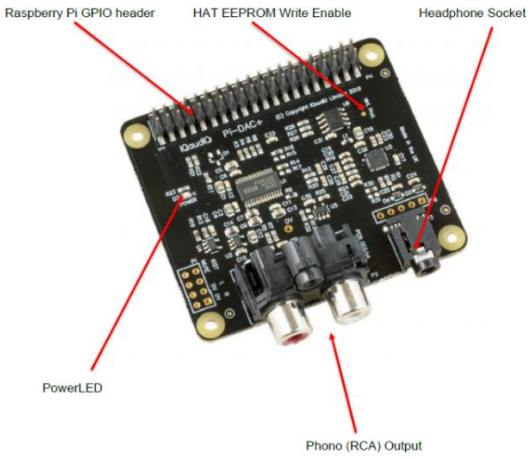
2.1.3 IQaudio Pro DAC+

Όλα τα μοντέλα Raspberry Pi πλην του μοντέλου 400, διαθέτουν μία αναλογική έξοδο ήχου για ακουστικά (3.5mm audio jack) από όπου μπορούμε να ακούσουμε ήχο μέσω

ακουστικών ή με το κατάλληλο καλώδιο να το συνδέσουμε πάνω σε ένα ηχοσύστημα, ηχεία υπολογιστή κλπ.

Το αρνητικό σε αυτή την περίπτωση είναι ότι η ποιότητα ήχου είναι αρκετά φτωχή, με αποτέλεσμα να μην απολαμβάνουμε στο έπακρο την ποιότητα της ψηφιακής μουσικής που μας παρέχουν πλέον οι απομακρυσμένες μουσικές υπηρεσίες ακόμα και αν διαθέτουμε καλής ποιότητας ηχεία και ακουστικά. Για να υπερπηδήσουμε αυτό το πρόβλημα, μπορούμε να χρησιμοποιήσουμε έναν από τους πολλούς ψηφιοαναλογικούς μετατροπείς ή αλλιώς DACs (Digital To Analog Converters) που διατίθενται για το Raspberry Pi.

Με τον όρο DAC εννοούμε έναν μετατροπέα ψηφιακού ήχου σε αναλογικό, ώστε να μπορέσουμε να τον ακούσουμε σε ηχεία ή ακουστικά. Για την υλοποίηση μας θα χρησιμοποιήσουμε το IQaudio DAC+ [7], το οποίο είναι ένα πρόσθετο HAT που συνδέεται στο 40-pin GPIO χωρίς να χρειάζεται να προβούμε σε κόλληση (soldering) και υποστηρίζει υψηλής ανάλυσης ψηφιακό ήχο 24-bit 192kHz. Μας παρέχει έξοδο Line Out που συνδέεται πάνω της βύσμα RCA για σύνδεση με προενισχυτή ή ενισχυτή ή για σύνδεση με αυτοενισχυόμενα (ενεργά) ηχεία. Επίσης παρέχει στερεοφωνική έξοδο ακουστικών μέσω 3.5 mm jack socket, η οποία μπορεί να χρησιμοποιηθεί και σαν aux out για σύνδεση με ηχοσύστημα.



Εικόνα 19: Raspberry Pi IQaudio DAC+ [7]

Το συγκεκριμένο DAC περιλαμβάνει μια EEPROM η οποία είναι προγραμματισμένη εκ των προτέρων, οπότε το Raspberry Pi OS αναγνωρίζει αυτόματα το DAC κατά τη

σύνδεση του με το Raspberry Pi (plug and play). Εάν θέλουμε να ρυθμίσουμε χειροκίνητα το Raspberry Pi OS, μπορούμε να επέμβουμε στις ρυθμίσεις του αρχείου εκκίνησης /boot/config.txt με την εντολή :

sudo nano /boot/config.txt

και να προσθέσουμε την παρακάτω γραμμή :

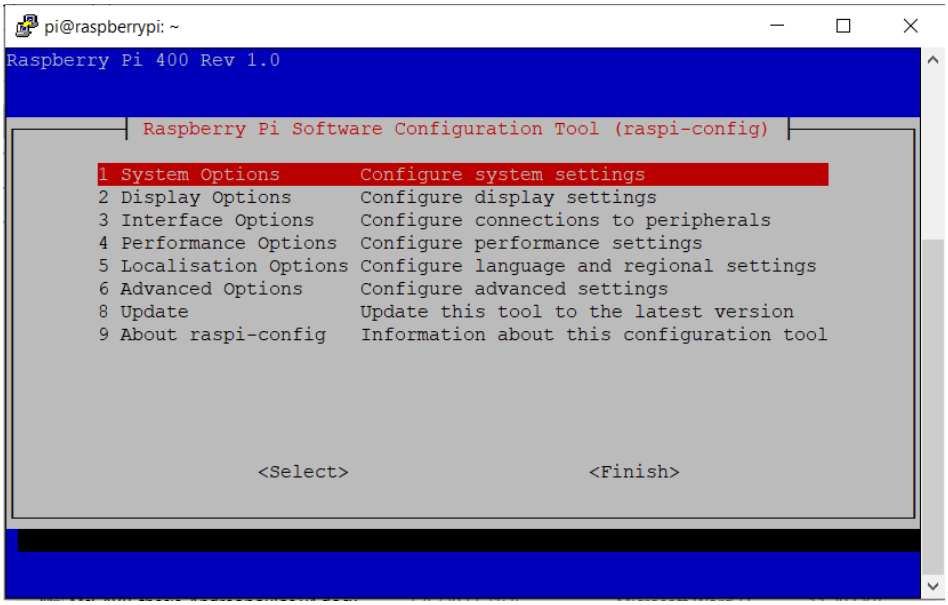
dtoverlay=iqaudio-dacplus

Για να ορίσουμε την κάρτα ήχου IQaudio ως κύρια κάρτα ήχου του συστήματος, θα πρέπει να απενεργοποιήσουμε την on-board κάρτα ήχου του Raspberry Pi. Για να επιτευχθεί αυτό, θα πρέπει να επέμβουμε στις ρυθμίσεις του αρχείου εκκίνησης /boot/config.txt και να μετατρέψουμε σε σχόλιο την γραμμή dtparam=audio=on βάζοντας μπροστά στη γραμμή το σύμβολο της δίεσης (#).

#dtparam=audio=on

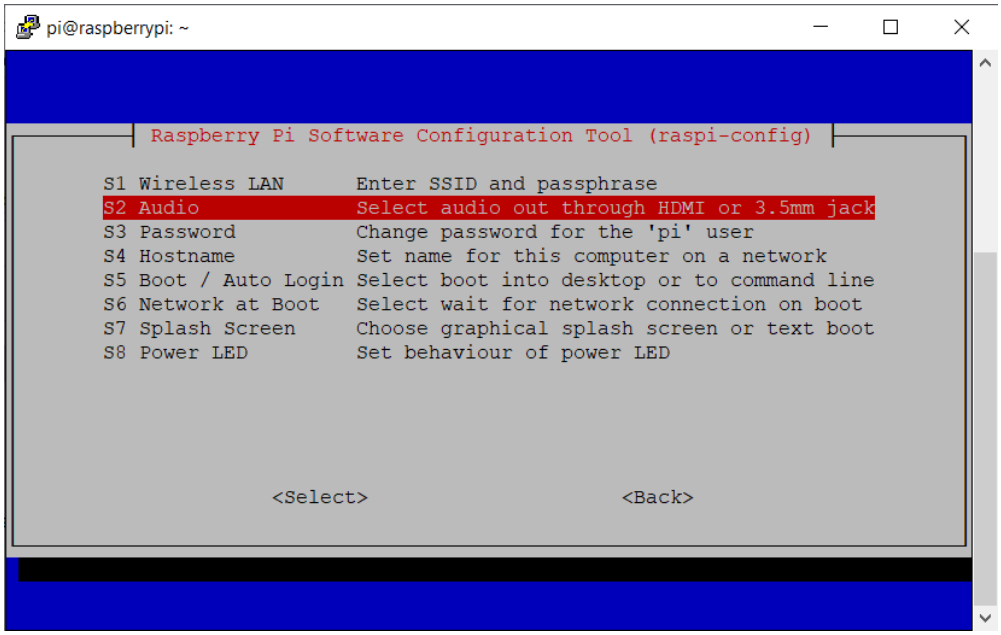
Τέλος αποθηκεύουμε την αλλαγή και δίνουμε την εντολή :

sudo raspi-config



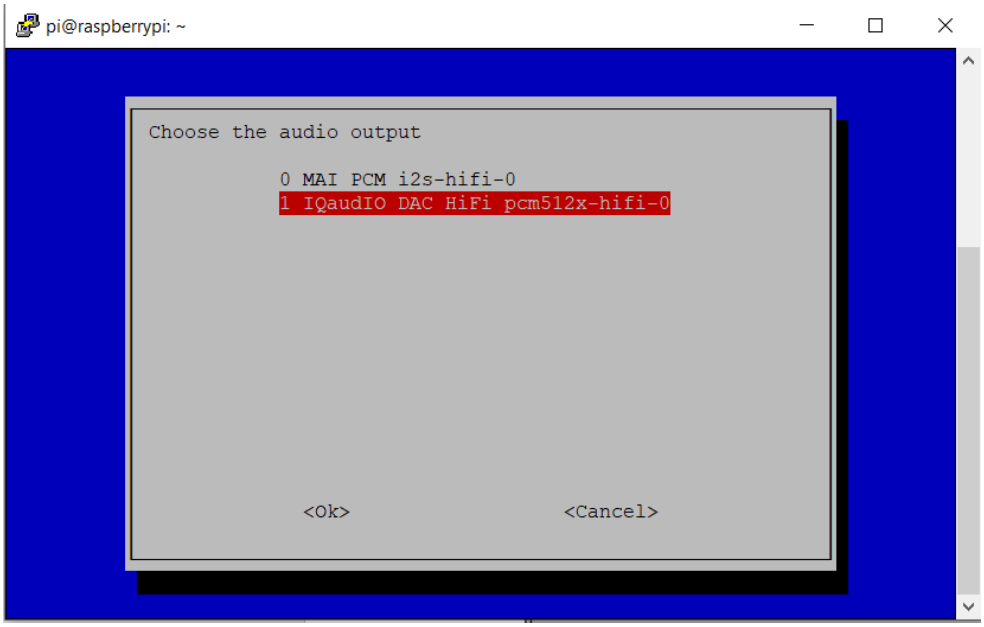
Εικόνα 20: Raspi-config

Επιλέγουμε την επιλογή “System Options” και μετά την επιλογή “Audio” για να επιλέξουμε ποια έξοδο ήχου επιθυμούμε.



Εικόνα 21: Raspi-config

Από τις επιλογές που μας δίνονται επιλέγουμε την επιλογή “1 IQaudIO DAC HiFi pcm512x-hifi-0 και μετά “Finish” ώστε να αποθηκευτεί η αλλαγή.

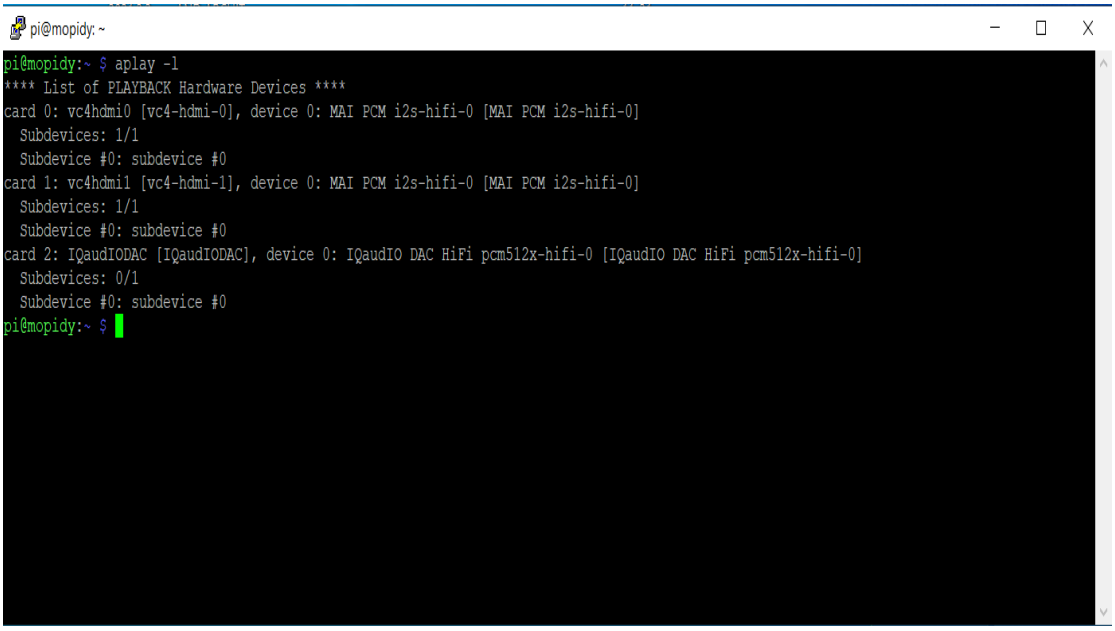


Εικόνα 22: Raspi-config

Τέλος κάνουμε επανεκκίνηση του συστήματος.

Ένας άλλος τρόπος να ορίσουμε την κάρτα IQaudio ως κύρια κάρτα του συστήματος είναι να τρέξουμε πρώτα την εντολή aplay ώστε να δούμε τη λίστα με τις διαθέσιμες συσκευές ήχου.

aplay -l



Εικόνα 23: Αποτέλεσμα εντολής aplay -l

Βλέπουμε ότι η IQaudio κάρτα ήχου είναι η card 2.

Έπειτα επεμβαίνουμε στο αρχείο ρυθμίσεων του ALSA driver με την ακόλουθη εντολή:

sudo nano /usr/share/alsa/alsa.conf

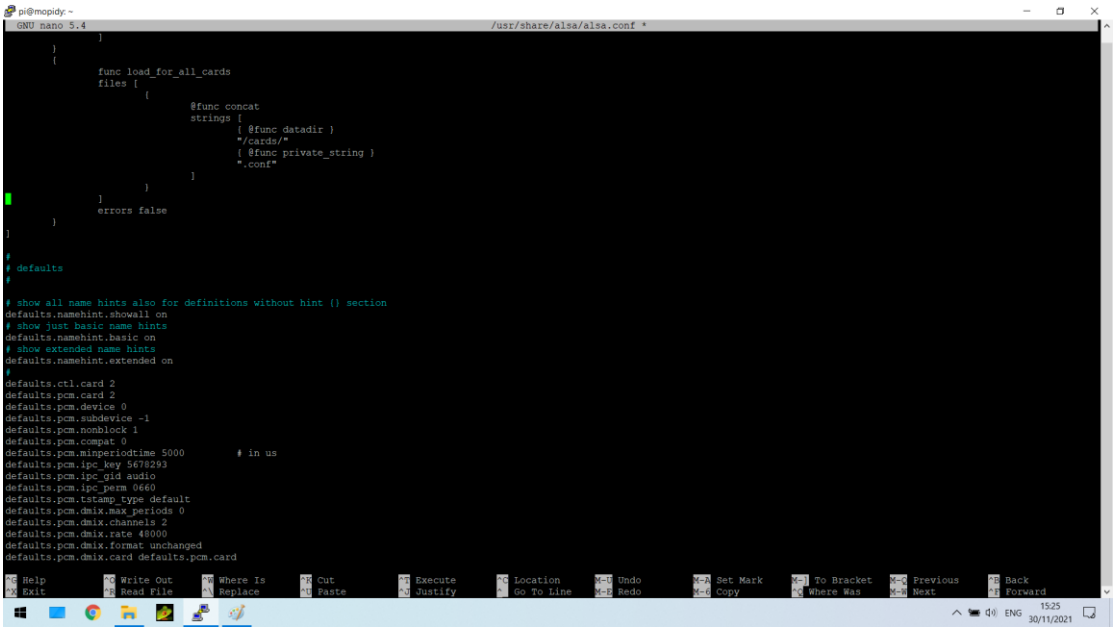
Παρακάτω βλέπουμε το περιεχόμενο του αρχείου ρυθμίσεων.

Εικόνα 24: Αρχείο ρυθμίσεων ALSA

Αναζητούμε το σημείο όπου περιλαμβάνει τις γραμμές

defaults.ctl.card 0
defaults.pcm.card 0

και αντικαθιστούμε την τιμή 0 με την τιμή 2, που αντιστοιχεί στην κάρτα IQaudio.



Εικόνα 25: Αρχείο ρυθμίσεων ALSA

Αποθηκεύουμε τις αλλαγές και επανεκκινούμε το σύστημα.

Εικόνα 26: Raspberry Pi 400 και IQaudio Pro DAC+, συνδεδεμένα με 40-Pin ribbon

2.2 Software

2.2.1 Raspberry Pi OS

Το Raspberry Pi OS [72] είναι το προτεινόμενο λειτουργικό σύστημα για το Raspberry Pi από το 2015 [73] και είναι μία έκδοση του Linux, και συγκεκριμένα βασίζεται στη διανομή Debian. Η αρχική του έκδοση ήταν το Raspbian και είχε δημιουργηθεί το 2012. [74] Το Raspberry Pi OS είναι παραμετροποιημένο ώστε να συνεργάζεται άψογα με όλα τα μοντέλα του Raspberry Pi και τον ARM επεξεργαστή τους, εκτός από το μοντέλο Pico.

Διαθέτει γραφικό περιβάλλον το οποίο ονομάζεται PIXEL, και βασίζεται στο γραφικό περιβάλλον LXDE. [75] Σε περίπτωση που κάποιος επιθυμεί ένα πολύ ελαφρύ περιβάλλον το οποίο θα καταναλώνει ελάχιστη μνήμη RAM και υπολογιστικούς πόρους, μπορεί να εγκαταστήσει την έκδοση Raspberry Pi OS Lite [76]. Στην περίπτωση αυτή θα εγκατασταθούν μόνο τα βασικά πακέτα, χωρίς όμως να έχει γραφικό περιβάλλον, αλλά όλες οι εντολές θα δίνονται μέσω της γραμμής εντολών (terminal).

Αν και το Raspberry Pi OS είναι το προτιμότερο λειτουργικό, το Raspberry Pi μπορεί να τρέξει και άλλα λειτουργικά. Ενδεικτικά αναφέρονται κάποια από αυτά :

- **LibreElec** [77]

  Μια διανομή για να υλοποιήσουμε ένα media center, βασισμένο στο Kodi.

- **OSMC** (open source media center) [78]

  Άλλη μια υλοποίηση media center, βασισμένη στο Kodi.

- **Ubuntu Desktop** [79]

  Η διάσημη διανομή Ubuntu, που και αυτή βασίζεται στο Debian.

- **Ubuntu Server** [80]

  Διανομή Ubuntu για υπηρεσίες cloud , data center και servers.

- **Ubuntu Core** [81]

Διανομή Ubuntu για προγραμματιστές

- **Arch Linux for ARM** [82]

  Έκδοση της διανομής Arch Linux για επεξεργαστές ARM

- **SARPI project** [83]

  Πρόκειται για την έκδοση μίας εκ των παλαιότερων διανομών Linux, τη διανομή Slackware, διαμορφωμένη να τρέχει σε ARM.

- **RetroPie** [84]

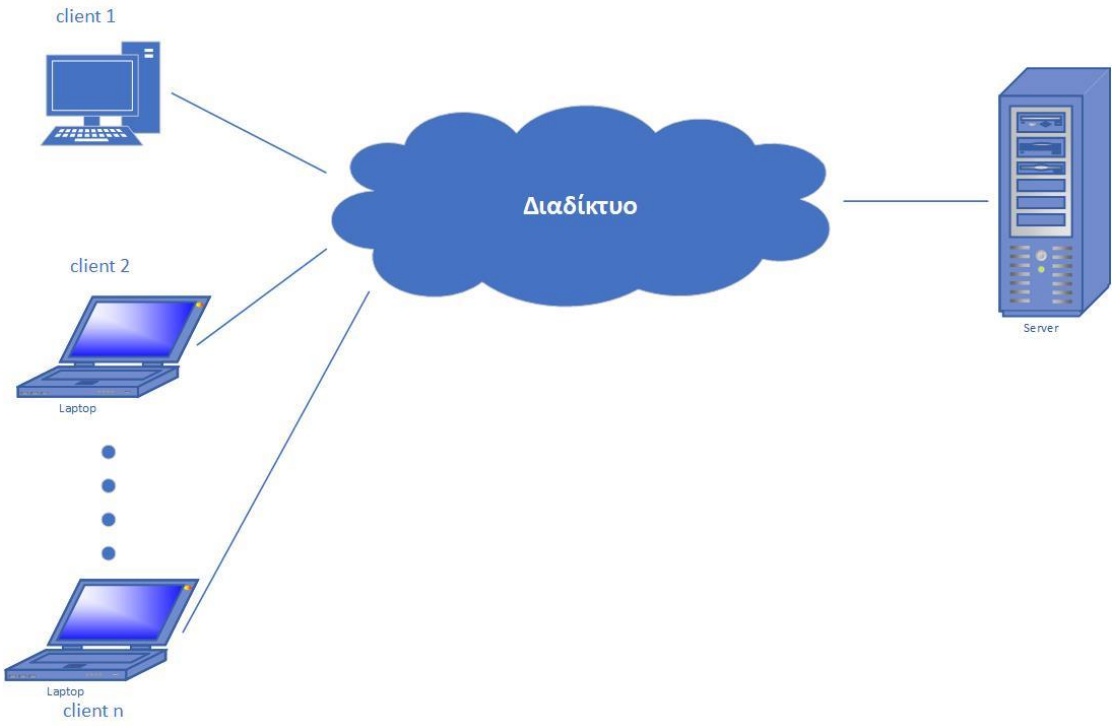
  Με τη διανομή αυτή, μπορούμε να μετατρέψουμε το Raspberry Pi σε παιχνιδομηχανή παλαιών (retro) ηλεκτρονικών παιχνιδιών.

- **Lakka** [85]

  Μια ακόμα διανομή παρόμοια με το RetroPie, για να παίζουμε παλιά παιχνίδια.

2.2.2 Client – Server Architecture

Η αρχιτεκτονική client – server είναι ένα υπολογιστικό μοντέλο κατά το οποίοι ο server φιλοξενεί, διανέμει και ελέγχει την πλειονότητα των πόρων και των υπηρεσιών που πρόκειται να χρησιμοποιηθούν από τον client. Στην ουσία είναι δύο υπολογιστές που συνδέονται μεταξύ τους μέσω ενός δικτύου ή του διαδικτύου, αν και είναι πιθανόν να βρίσκονται και στο ίδιο σύστημα. Ένας ή περισσότεροι clients (ή αλλιώς front-end) συνδέονται στο server (ή αλλιώς back end) μέσω δικτύου και μοιράζονται τους υπολογιστικούς πόρους. Οι clients στέλνουν αιτήματα προς τον server μέσω δικτύου, και ο server δέχεται τα αιτήματα, τα επεξεργάζεται και επιστρέφει στους clients τα δεδομένα που του ζητήθηκαν. Επίσης ένας client μπορεί να συνδεθεί σε περισσότερους του ενός servers, και να δεχτεί διαφορετικές υπηρεσίες από τον καθένα. [11] [86] [87] [88] [89] [90] [91] [92]



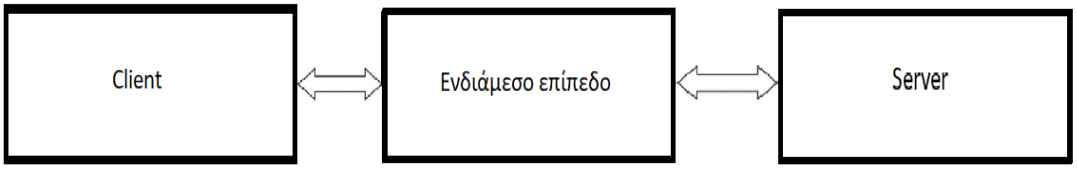
Εικόνα 27: Client – Server μοντέλο

Υπάρχουν διάφοροι τύποι της αρχιτεκτονικής client – server ανάλογα με τα επίπεδα (tiers) που περιλαμβάνουν και γενιά η αρχιτεκτονική αυτή ονομάζεται N – Tier όπου N ο αριθμός των επιπέδων αυτών, οι οποίοι περιγράφονται παρακάτω :

- Αρχιτεκτονική 1 – Tier όπου ο client και ο server βρίσκονται στο ίδιο σύστημα.

- Αρχιτεκτονική 2 – Tier όπου το interface του client βρίσκεται στο client σύστημα, ενώ πχ μια βάση δεδομένων στην οποία ο client στέλνει ερωτήματα βρίσκεται αποθηκευμένη στο server. Ο client και ο server είναι απευθείας συνδεδεμένοι με κάποιο πρωτόκολλο επικοινωνίας, χωρίς να παρεμβάλλεται ενδιάμεσο επίπεδο. Η αρχιτεκτονική αυτή είναι εύκολη στο σχεδιασμό και είναι αποτελεσματική ως προς την ταχύτητα μεταφοράς της πληροφορίας.

- Αρχιτεκτονική 3 – Tier στην οποία απαιτείται ένα ενδιάμεσο επίπεδο.



Εικόνα 28: Αρχιτεκτονική 3 – Tier

Στην περίπτωση αυτή, όταν ο client στέλνει ένα αίτημα, το αίτημα περνάει από ένα ενδιάμεσο επίπεδο, αντί να φτάσει απευθείας στο server. Το ενδιάμεσο αυτό επίπεδο λειτουργεί ως διεπαφή μεταξύ του client και του server, επεξεργάζεται τα αιτήματα που προέρχονται από τον client και ελέγχει τα δεδομένα για την εγκυρότητά τους και την ποιότητά τους (validation), παίρνει τις κατάλληλες λογικές αποφάσεις, εκτελεί υπολογισμούς και μεταφέρει δεδομένα μεταξύ του client και του server.

- Η αρχιτεκτονική client – server μπορεί να περιλαμβάνει και περισσότερα των τριών επιπέδων (tier) οπότε τότε μιλάμε για 4 – Tier, 5 -Tier κλπ., γενικότερα N - Tier.

Στην περίπτωση αυτή το ενδιάμεσο επίπεδο αποτελείται από περισσότερα επίπεδα τα οποία εκτελούν διαφορετικές διεργασίες και έχουν διαφορετικές ευθύνες, και τα οποία μπορούν να επικοινωνούν μεταξύ τους. [92] [93]

Πλεονεκτήματα και μειονεκτήματα του της αρχιτεκτονικής client – server N-Tier

Η αρχιτεκτονική N - Tier παρουσιάζει αρκετά πλεονεκτήματα όπως είναι η κεντρική διαχείριση των εφαρμογών και των δεδομένων, ότι δεν είναι απαραίτητο οι clients και ο server να βρίσκονται στον ίδιο χώρο ή σε κοντινή απόσταση, ότι σε ένα server μπορούν να φιλοξενηθούν όλα τα δεδομένα οπότε είναι πιο εύκολο να προσπελαστούν και να ανακτηθούν, εξασφαλίζοντας παράλληλα προστασία των δεδομένων αυτών με μεθόδους αυθεντικοποίησης των χρηστών (authentication) και παρέχοντας επιλεκτική πρόσβαση στα δεδομένα ανάλογα με το χρήστη (authorization). Ένα άλλο πλεονέκτημα είναι η επεκτασιμότητα (scalability) όπου στο σύστημα μπορούν να προστεθούν περεταίρω clients και servers, χωρίς να επηρεαστεί η απρόσκοπτη λειτουργία του συστήματος. Τυχόν αναβαθμίσεις ή ακόμα και η αντικατάσταση πχ ενός client πάλι δεν θα επηρεάσει τη συνολική συμπεριφορά του συστήματος. Ακόμα μπορεί οι clients και οι servers να τρέχουν σε διαφορετικές πλατφόρμες και να συνεργάζονται χωρίς πρόβλημα. [92] [93] [94]

Όσον αφορά τα μειονεκτήματα, με τη χρήση τέτοιων μοντέλων αυξάνεται ο υπολογιστικός φόρτος εργασίας για παράδειγμα όταν πολλοί clients στέλνουν ταυτόχρονα αιτήματα στο server και γενικότερα είναι πιο πολύπλοκα από πλευράς υλοποίησης. [93]

2.2.3 GStreamer

Το GStreamer [12] είναι ένα framework για την υλοποίηση εφαρμογών streaming ήχου, βίντεο ή και των δύο. Δεν περιορίζεται μόνο σε βίντεο ή ήχο, αλλά μπορεί να διαχειριστεί κάθε τύπο ροής δεδομένων (data flow). Πρόκειται για ένα δωρεάν και ανοιχτού κώδικα λογισμικό γραμμένο σε γλώσσα προγραμματισμού C [95] και που προστατεύεται από την άδεια GNU LGPL 2.1. [96] [97]

Λειτουργεί σε όλα τα γνωστά λειτουργικά συστήματα όπως Linux, Windows, Android, iOS, Max OS X, στα περισσότερα BSD, εμπορικά UNIX, Symbian και Solaris. Τρέχει σε όλες τις γνωστές αρχιτεκτονικές hardware όπως x86, ARM, MIPS, SPARC και PowerPC, σε 32-bit καθώς και 64-bit.

Μια από τις πιο συνηθισμένες χρήσεις του είναι για την κατασκευή media player. Το framework αυτό βασίζεται σε plugins τα οποία παρέχουν τα διάφορα codec και κάθε άλλη λειτουργικότητα. Τα plugins αυτά συνδέονται και σχηματίζουν ένα pipeline, το οποίο ορίζει τη ροή των δεδομένων. Η βασική λειτουργία του GStreamer είναι να παρέχει ένα framework για plugins, ροή δεδομένων και διαχείριση των τύπων του περιεχομένου (media type handling/negotiation). Επίσης παρέχει ένα API για την δημιουργία εφαρμογών χρησιμοποιώντας τα διάφορα plugins.

Συγκεκριμένα το GStreamer παρέχει :

- Ένα API για εφαρμογές πολυμέσων
- Μια αρχιτεκτονική plugin
- Μια αρχιτεκτονική pipeline
- Ένα μηχανισμό για διαχείριση τύπων περιεχομένου (media type handling/negotiation)
- Ένα μηχανισμό για συγχρονισμό
- Πάνω από 250 plugins που παρέχουν πάνω από 100 στοιχεία (elements)
- Ένα σετ εργαλείων

Τα plugins μπορούν να ταξινομηθούν σε :

- Πρωτόκολλα διαχείρισης
- Πηγές εξόδου (sources) για ήχο και βίντεο
- Μορφές (formats) όπως αναλυτές (parsers), πολυπλέκτες (muxers), αποπολυπλέκτες (demuxers), μεταδεδομένα (metadata) και υπότιτλοι (subtitles).
- Φίλτρα (filters) όπως μετατροπείς (converters), μίκτες (mixers), εφέ (effects) κλπ.
- Εισόδους (sinks) για ήχο και βίντεο

Επειδή το GStreamer είναι ένα μεγάλο πρότζεκτ, είναι χωρισμένο σε διάφορα modules [98]. Για τις περισσότερες υλοποιήσεις χρειάζονται τουλάχιστον τα παρακάτω modules :

- gstreamer όπου περιλαμβάνει τη βασική βιβλιοθήκη και στοιχεία (elements) [99]
- gst-plugins-base που περιλαμβάνει μια ενημερωμένη συλλογή από plugins και elements τα οποία είναι δοκιμασμένα ότι λειτουργούν σωστά, βιβλιοθήκες και κλάσεις για τη δημιουργία elements και μια ευρεία γκάμα από encoders, decoders και φίλτρα ήχου και βίντεο [100]
- gst-plugins-good που περιλαμβάνει κώδικα καλής ποιότητας, σωστή λειτουργικότητα και την επιθυμητή άδεια LGPL [101]
- gst-plugins-ugly που περιλαμβάνει plugins τα οποία είναι καλής ποιότητας, αλλά ενδέχεται να παρουσιάζουν προβλήματα σχετικά με την άδεια χρήσης τους [102]
- gst-plugins-bad που περιλαμβάνει plugins τα οποία δεν είναι τόσο καλής ποιότητας [103]

Βασικές έννοιες GStreamer [104] [105]

Στοιχεία (Elements)

Ένα element είναι το πιο σημαντικό αντικείμενο του GStreamer. Συνήθως κατασκευάζεται μία αλυσίδα από επιμέρους elements τα οποία συνδέονται μεταξύ τους και επιτρέπουν στα δεδομένα να "ρέουν" μέσω αυτής της αλυσίδας από elements, τα οποία το καθένα επιτελούν και κάποιο σκοπό. Τα συνδεδεμένα elements δημιουργούν ένα pipeline το οποίο επιτελεί μία συγκεκριμένη εργασία όπως αναπαραγωγή πολυμέσων. Το GStreamer περιλαμβάνει μια μεγάλη συλλογή από elements, καθώς επίσης μπορεί κάποιος να κατασκευάσει τα δικά του elements.

Ένα element μπορεί να λάβει τέσσερις διαφορετικές καταστάσεις όπως η NULL και READY κατά τις οποίες το element δεν διαχειρίζεται καθόλου δεδομένα. Στην κατάσταση PLAYING γίνεται μεταφορά δεδομένων. Η ενδιάμεση κατάσταση PAUSED γεμίζει όλα τα συνδεδεμένα elements με δεδομένα και ενεργοποιεί τα pads, πρώτα τις πηγές εξόδου (source pads) και έπειτα τις εισόδους (sink pads). Αν ένα element είναι σε κατάσταση READY και μετά μπει σε κατάσταση PLAYING, πρώτα θα περάσει από την κατάσταση PAUSED. Αυτό γίνεται ώστε η μετάβαση σε κατάσταση PLAYING να γίνει πολύ γρήγορα.

Υπάρχουν διάφορες κατηγορίες elements όπως :

- source elements τα οποία παρέχουν δεδομένα στο pipeline
- sink elements τα οποία εξάγουν δεδομένα σε μια συσκευή εξόδου
- transform elements τα οποία μετατρέπουν ένα stream εισόδου από ένα format σε ένα άλλο format.
- demuxer elements που αναλύουν ένα stream και παράγουν πολλαπλά streams
- mixer/muxer elements που συνδυάζουν διάφορα streams εισόδου σε ένα μοναδικό stream εξόδου.

Pads

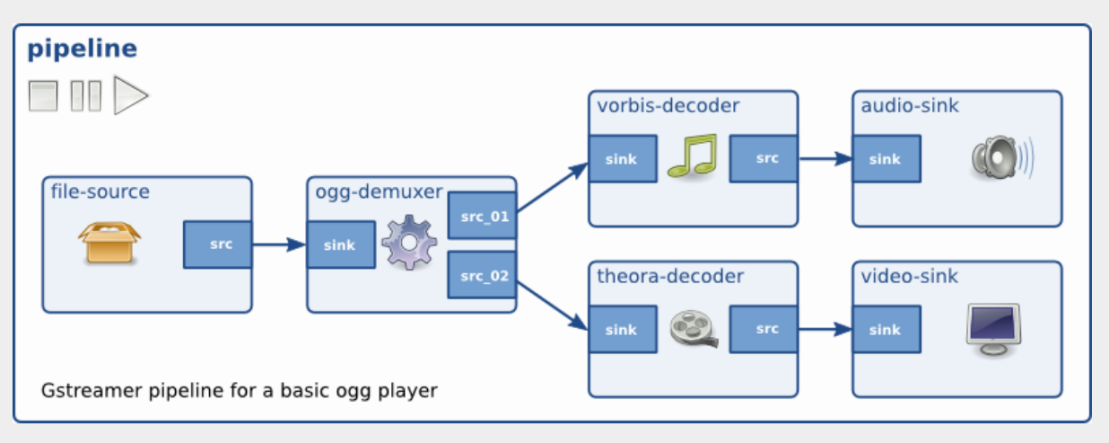
Τα pads είναι οι είσοδοι και οι έξοδοι ενός element μέσω των οποίων γίνεται η ροή των δεδομένων και πάνω στις οποίες συνδέονται άλλα elements. Επίσης μπορούν να κάνουν διαχείριση των δεδομένων, όπως να επιτρέπουν ή να αποτρέπουν τη διέλευση συγκεκριμένων τύπων δεδομένων. Συνδέσεις επιτρέπονται μόνο μεταξύ δύο pads όταν οι επιτρεπόμενοι τύποι αρχείων (capabilities) των δύο pads είναι συμβατοί. Η επιλογή των τύπων δεδομένων γίνεται με τη διαδικασία που ονομάζεται caps negotiation. Η ροή των δεδομένων γίνεται συνήθως προς τη μία κατεύθυνση μεταξύ της σύνδεσης μεταξύ των elements. Τα δεδομένα εξέρχονται από ένα element μέσω ενός ή περισσότερων source pads και εισέρχονται στο επόμενο μέσω ενός ή περισσότερων sink pads.

Bins

Ένα bin είναι ένα κοντέινερ που περιέχει διάφορα elements οπότε ελέγχοντας ένα bin μπορεί κάποιος να ελέγξει τα επιμέρους elements, που εμπεριέχονται στο bin αυτό. Για παράδειγμα αν γίνει αλλαγή στην κατάσταση ενός bin, θα αλλάξει αυτομάτως και η κατάσταση των elements που περιέχει. Ένα bin μπορεί να έχει τις δικές του εξόδους και εισόδους, χρησιμοποιώντας τις εξόδους και εισόδους ενός από τα elements που περιέχει (ghostpadding).

Pipeline

Ένα pipeline είναι ένα bin ανώτατου επιπέδου που παρέχει ένα σύστημα επικοινωνίας (bus) για την εφαρμογή και διαχειρίζεται το συγχρονισμό όλων των επιμέρους στοιχείων του και παρέχει τρόπους επικοινωνίας μεταξύ της εφαρμογής και των elements. Αν ένα pipeline τεθεί σε κατάσταση PAUSED ή PLAYING, θα αρχίσει η ροή των δεδομένων μέχρι να ζητηθεί η διακοπή της ροής δεδομένων ή φτάσει στο τέλος της.



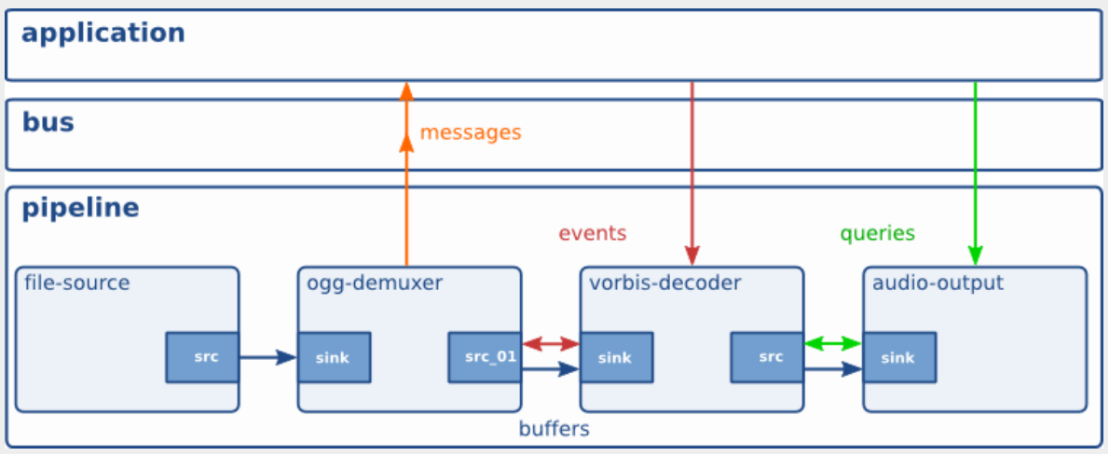
Εικόνα 29: GStreamer pipeline [**104**]

Επικοινωνία

Το GStreamer παρέχει διάφορους τρόπους επικοινωνίας και ανταλλαγής δεδομένων μεταξύ της εφαρμογής και του pipeline, που περιγράφονται παρακάτω :

- buffers όπου είναι αντικείμενα τα οποία περνάνε δεδομένα που σχετίζονται με το stream, μεταξύ των στοιχείων του pipeline. Τα αντικείμενα αυτά πάντοτε μεταφέρονται με φορά από την έξοδο του ενός προς την είσοδο του επόμενου στοιχείου (downstream).
- συμβάντα (events) τα οποία είναι αντικείμενα τα οποία στέλνονται μεταξύ των στοιχείων ή από την εφαρμογή προς τα στοιχεία. Τα events μπορούν να ταξιδεύουν κατά τη φορά της ροής δεδομένων, αλλά μπορεί και προς την αντίθετη φορά.

- μηνύματα (messages) τα οποία είναι αντικείμενα που στέλνονται από τα elements και τα συλλέγει η εφαρμογή. Τα μηνύματα χρησιμεύουν στο να αποστέλλονται πληροφορίες όπως σφάλματα, λέξεις-κλειδιά (tags), αλλαγές καταστάσεων κλπ. από τα elements στην εφαρμογή.
- ερωτήματα (queries) τα οποία επιτρέπουν στις εφαρμογές να ζητούν πληροφορίες σχετικά με τη διάρκεια ή την τρέχουσα θέση αναπαραγωγής από το pipeline. Επίσης και τα elements μπορούν να ζητούν πληροφορίες από άλλα elements, όπως πληροφορίες για το μέγεθος του αρχείου ή διάρκεια. Τα ερωτήματα μπορούν να μεταφερθούν μέσα στο pipeline και προς τις δύο κατευθύνσεις, αν και η πιο συνηθισμένη φορά είναι η αντίθετη (upstream) της φοράς που γίνεται η ροή δεδομένων (downstream).



Εικόνα 30: Επικοινωνία και ανταλλαγή δεδομένων μεταξύ εφαρμογής και pipeline **[104]**

2.2.4 Γλώσσα Προγραμματισμού Python

Η Python [10] είναι μία γλώσσα προγραμματισμού υψηλού επιπέδου, δωρεάν και ανοιχτού κώδικα, αντικειμενοστραφής και γενικού σκοπού, με ευρεία εφαρμογή σε υλοποιήσεις όπως εφαρμογές για υπολογιστές οι οποίες είναι δυνατό να τρέξουν στα περισσότερα γνωστά λειτουργικά συστήματα όπως Windows, Linux, MacOS, Unix, Raspberry Pi OS κλπ., καθώς και ηλεκτρονικά παιχνίδια, εφαρμογές διαδικτύου κλπ. Δημιουργός της γλώσσας αυτής είναι ο Guido van Rossum και εμπνεύστηκε το όνομα Python από την κωμική σειρά του BBC " Monty Python's Flying Circus ", που προβαλλόταν τη δεκαετία του '70. [106] Στην περίπτωση της Python ο κώδικας μεταγλωττίζεται αυτόματα σε μια δυαδική ενδιάμεση γλώσσα, που ονομάζεται byte code

την οποία αναγνωρίζει ο υπολογιστής οπότε την κάνει κατάλληλη και για συγγραφή scripts (σεναρίων). Λόγω του ότι προγραμματίζοντας με την Python γίνεται χρήση κλάσεων (classes), αντικειμένων (objects), συναρτήσεων (functions), modules και εμφωλευμένα (nested) κομμάτια κώδικα και το ότι είναι αντικειμενοστραφής γλώσσα προγραμματισμού, μπορεί κάποιος να γράψει εύκολα αναγνώσιμο και έξυπνα δομημένο κώδικα για εφαρμογές μικρής ή μεγάλης κλίμακας [107]. Μια άλλη σημαντική δυνατότητα της Python είναι ότι ο κώδικας μπορεί να επεκταθεί με νέα κομμάτια γραμμένα σε C ή C++ με αποτέλεσμα να μπορούν να εκτελεστούν γρήγορα απαιτητικές διαδικασίες. [108]

Ο κώδικας μιας εφαρμογής που γράφεται σε Python είναι επίσης μικρότερος σε μέγεθος σε σύγκριση με άλλες γλώσσες πχ από τη Java. Επίσης η Python θεωρείται πιο εύκολη στην εκμάθηση για έναν αρχάριο, διότι έχει ευκολότερη και πιο ξεκάθαρη σύνταξη. Είναι γλώσσα δυναμικού τύπου (dynamically typed) που σημαίνει ότι δεν χρειάζεται να δηλώσουμε τον τύπο των δεδομένων και ότι ο έλεγχος των τύπων (type checking) γίνεται κατά την εκτέλεση του προγράμματος, σε αντίθεση πχ με την Java η οποία είναι μια statically typed γλώσσα προγραμματισμού και η δήλωση των τύπων είναι υποχρεωτική και μετά δεν μπορεί να αλλάξει και ο έλεγχος τους γίνεται κατά τη μεταγλώττιση (compile). Το ότι η Python είναι dynamically typed γλώσσα όμως, έχει και κάποια μειονεκτήματα όπως είναι ότι τα λάθη των τύπων εντοπίζονται αργότερα σε σχέση με τις statically typed γλώσσες, και συγκεκριμένα κατά την εκτέλεση του προγράμματος με συνέπεια να καθυστερεί η εκτέλεση, είναι πιο ασταθής καθώς και ότι ο κώδικας είναι λιγότερο βελτιστοποιημένος (optimized) και είναι πιο επιρρεπής στα σφάλματα εκτέλεσης (runtime errors). Η γλώσσα Python προτιμάται επίσης για χρήση σε εφαρμογές Μηχανικής μάθησης (machine learning) και για την κατασκευή εφαρμογών επεξεργασίας εικόνας. [109] [110]

2.2.5 Λίγα λόγια για το Pykka

Το Pykka [111] είναι μια υλοποίηση του actor μοντέλου [112], αλλά με τη χρήση της γλώσσας προγραμματισμού Python. Τα πνευματικά δικαιώματα ανήκουν στον Stein Magnus Jodal και τους υπόλοιπους συνεισφέροντες (contributors) και προστατεύεται από την άδεια Apache License, Version 2.0 [111] [113]. Το Mopidy χρησιμοποιεί Pykka actors σε μεγάλο βαθμό ώστε τα ανεξάρτητα μέρη του όπως τα frontends και οι επεκτάσεις του να τρέχουν ανεξάρτητα, με συνέπεια αν εμφανιστεί κάποιο πρόβλημα σε κάποιο από αυτά, να μην επηρεαστεί η συνολική λειτουργία του συστήματος. [114]

Η χρήση του μοντέλου actors εισάγει απλούς κανόνες ώστε να μπορούν να συνεργάζονται διάφορες μονάδες μεταξύ τους, με αποτέλεσμα να είναι πιο εύκολη η κατασκευή εφαρμογών που τρέχουν παράλληλα (concurrent applications). Δηλαδή ένας actor μπορεί να εκτελείται παράλληλα με άλλους actors. Οι actors μπορούν να έχουν τη δική τους κατάσταση χωρίς να τη μοιράζονται με κανέναν άλλον και μπορεί να επικοινωνεί με άλλους actors στέλνοντας ή λαμβάνοντας μηνύματα. Λαμβάνοντας ένα μήνυμα ένας actor, μπορεί να αλλάξει την κατάστασή του ή να στείλει μηνύματα σε άλλους actors, καθώς και να ξεκινήσει νέους actors. Τέλος ένας actor μπορεί να διαχειριστεί μόνο ένα μήνυμα κάθε φορά. [115]

Τα χαρακτηριστικά (attributes) και οι μέθοδοι (methods) ενός pykka actor μπορούν να προσπελαστούν με υλοποιήσεις του τύπου ThreadingActor το οποίο εκτελείται με thread, ThreadingFuture που διαχειρίζεται μελλοντικά (future) αποτελέσματα και proxies. Ένας actor ξεκινά με τη μέθοδο start() στην οποία μπορούμε να εισάγουμε και ορίσματα (arguments) αν το επιθυμούμε. Για να σταματήσει ένας actor πρέπει να χρησιμοποιηθεί η μέθοδος stop().

Ένας actor για να στέλνει μηνύματα χρησιμοποιεί τις μεθόδους tell() και ask(), των οποίων η διαφορά είναι ότι με την tell() ο actor δεν περιμένει απάντηση και συνεχίζει κανονικά τη λειτουργία του, ενώ με την ask() ο actor μπλοκάρει μέχρι να λάβει απάντηση. Το πόσα δευτερόλεπτα θα περιμένει την απάντηση μπορεί να οριστεί με ένα όρισμα, το timeout. Αν δοθεί όρισμα block=false, θα επιστραφεί άμεσα ένα “future” αντικείμενο. [116]

Το pykka μπορεί να χρησιμοποιηθεί και χωρίς τη χρήση των proxies, αλλά στην περίπτωση του Mopidy οι proxies βρίσκουν μεγάλη χρησιμότητα. Δημιουργώντας proxy αντικείμενα, τα αντικείμενα αυτά ελέγχουν εσωτερικά τον actor ώστε να διαπιστώσουν ποια χαρακτηριστικά και ποιες μέθοδοι είναι δημόσιες (public), ενώ τα χαρακτηριστικά και οι μέθοδοι που ξεκινούν με κάτω παύλα, θα αγνοηθούν ώστε να κρατηθούν ιδιωτικά (private). Όταν γίνεται προσπέλαση των χαρακτηριστικών ή γίνεται κλήση των μεθόδων στον proxy, θα ζητήσει από τον actor πρόσβαση στα χαρακτηριστικά και τις μεθόδους αυτές, και επιστρέφει τα αποτελέσματα. Επειδή τα αποτελέσματα αυτά περιλαμβάνονται σε “future” αντικείμενα, θα πρέπει να χρησιμοποιείται η μέθοδος get() ώστε να αποκτηθούν τα δεδομένα αυτά. Η διαφορά με το να χρησιμοποιούσαμε κανονικούς actors είναι ότι με τη χρήση των proxy αγνοούνται τα private χαρακτηριστικά και μέθοδοι και δημιουργούνται αυτόματα τρεις διαφορετικοί τύποι μηνυμάτων, ένας για τις κλήσεις των μεθόδων, ένας για την ανάγνωση των χαρακτηριστικών και ένας για την εγγραφή των χαρακτηριστικών. [117]

Σε περίπτωση που κάποιος επιθυμεί να έχει πρόσβαση στις μεθόδους ή τα χαρακτηριστικά ενός actor μέσω proxy, μπορεί να δηλώσει τα χαρακτηριστικά αυτά ως traversable με τη χρήση της συνάρτησης traversable(), οπότε το Pykka δεν θα επιστρέφει τα χαρακτηριστικά αυτά, άλλα θα τα περιλαμβάνει σε ένα νέο proxy, και θα επιστρέφει τον proxy αυτόν. Με αυτόν τον τρόπο είναι δυνατή η προσπέλαση μεθόδων και χαρακτηριστικών όσο βαθιά και να είναι στον κώδικα, αρκεί όλα τα χαρακτηριστικά και οι μέθοδοι στη διαδρομή μεταξύ του actor και αυτών να δηλωθούν ως traversable. [118]

2.2.6 Mopidy Music Server

Ο Mopidy server είναι ένας μουσικός σέρβερ με δυνατότητες επέκτασης, γραμμένος σε γλώσσα προγραμματισμού Python και μπορεί να τρέξει από την κονσόλα ή στο παρασκήνιο σε υπολογιστές που έχουν λειτουργικό Linux ή Mac οι οποίοι έχουν δυνατότητα σύνδεσης σε δίκτυο και έξοδο ήχου και είναι ιδιαίτερα δημοφιλής σε συστήματα Raspberry Pi. Έχει τη δυνατότητα να αναπαράγει μουσική από τοπικά αποθηκευτικά μέσα, δηλαδή αρχεία ήχου αποθηκευμένα σε σκληρούς δίσκους και USB sticks, καθώς και να αναπαράγει μουσική μέσω διαδικτύου από online μουσικές υπηρεσίες όπως το YouTube, Sound Cloud, Bandcamp κλπ. Επίσης έχει τη δυνατότητα αναπαραγωγής ραδιοφωνικών σταθμών μέσω διαδικτύου. Αυτό επιτυγχάνεται με την επιλογή και προσθήκη καταλλήλων επεκτάσεων, τα λεγόμενα extensions. Μέχρι τις 16/05/2022 υποστήριζε και αναπαραγωγή μουσικής από το Spotify με την επέκταση Mopidy-Spotify, αλλά έκτοτε διακόπηκε η λειτουργία της βιβλιοθήκης libspotify, στην οποία στηριζόταν η λειτουργία της επέκτασης Mopidy-Spotify [119]. Μέχρι να βρεθεί κάποια εναλλακτική, δεν είναι δυνατή η αναπαραγωγή μουσικής από το Spotify.

Επιλέγοντας από μία πληθώρα MPD και Web Clients, μπορεί ο καθένας να διαμορφώσει και να ελέγξει την playlist του, με τη βοήθεια κινητού τηλεφώνου smart phone, ταμπλέτας ή ηλεκτρονικού υπολογιστή [9]. Το πρότζεκτ του Mopidy είναι ανοιχτού κώδικα και φιλοξενείται στο GitHub και είναι διαθέσιμο για όποιον ενδιαφέρεται να το κατεβάσει και να το χρησιμοποιήσει ή να κάνει τις δικές του αλλαγές, συνεισφέροντας στο όλο εγχείρημα [120]. Οι βασικοί προγραμματιστές του Mopidy είναι ο Stein Magnus Jodal και η υπόλοιπη ομάδα του Mopidy [121], τα πνευματικά δικαιώματα ανήκουν στον Stein Magnus Jodal και τους συνεισφέροντες (contributors) και προστατεύεται από την άδεια Apache License, Version 2.0 [113]. Επίσης διαθέτει αναλυτικό εγχειρίδιο που ονομάζεται Mopidy documentation [122], γραμμένο από τον Stein Magnus Jodal και τους συνεισφέροντες (contributors) του οποίου η άδεια είναι η CC BY-SA 3.0 Unported License [123]

Η αρχική ιδέα για την υλοποίηση του Mopidy, έγινε στις 23 Δεκεμβρίου του 2009, όταν ο Stein Magnus Jodal συζητούσε με τον φίλο του και τότε συνάδελφο Johannes Knutsen και σκεφτήκανε να κατασκευάσουν έναν MPD σέρβερ ο οποίος θα είχε τη δυνατότητα να αναπαράγει μουσική από το Spotify, αντί για τοπικά αρχεία. Εξού και το όνομα Mopidy, το οποίο προκύπτει από το συνδυασμό των λέξεων MPD και Spotify. Σύντομα στο πρότζεκτ προστέθηκε και ο Thomas Adamcik και έπειτα από μερικές μέρες κατασκευάσανε ένα πρωτόγονο MPD σέρβερ ο οποίος δούλευε με τον Sonata MPD client. Με την υλοποίηση ενός ενδιάμεσου επιπέδου, του λεγόμενου core layer, το οποίο παρεμβάλλεται μεταξύ του frontend και του backend επιπέδου ήταν δυνατό πλέον να προστεθούν και να ελεγχθούν επιπλέον επεκτάσεις (extensions) για αναπαραγωγή μουσικής και από άλλες πηγές, με το να επιτρέπει στα backends να λειτουργούν ως μια οντότητα καθώς και υιοθετήθηκε ο GStreamer ώστε το Mopidy να αναπαράγει ήχο, δίνοντας παράλληλα τη δυνατότητα στον τελικό χρήστη να μπορεί να ρυθμίζει το GStreamer pipeline απευθείας, από τις ρυθμίσεις του audio/output που βρίσκεται στο αρχείο ρυθμίσεων του Mopidy. Με το πέρασμα των χρόνων και με τη συνεισφορά περισσότερων συνεισφερόντων (contributors) ανά τον κόσμο, έγιναν αρκετές άλλες προσθήκες και αλλαγές, όπως για παράδειγμα την προσθήκη του HTTP frontend με τη χρήση JSON-RPC μέσω WebSocket, τη δημιουργία της βιβλιοθήκης Mopidy.js που δίνει τη δυνατότητα χρήσης web clients, του Stream backend για μεταφορά (streaming) μουσικής μέσω διαδικτύου, την υιοθέτηση του Zeroconf που έδινε πλέον τη δυνατότητα ανίχνευσης από εφαρμογές κινητών τηλεφώνων όπως το Mopidy-Mobile και άλλες αλλαγές ώσπου να φτάσει στη σημερινή του μορφή. [124]

Ο Mopidy server μπορεί να τρέχει σε ένα τερματικό ή σε ένα υπολογιστή με λειτουργικό Linux ή σε ένα υπολογιστή Mac σαν εφαρμογή παρασκηνίου αρκεί να υπάρχει δυνατότητα σύνδεσης με το διαδίκτυο και φυσικά κάποια έξοδος ήχου. Πολλοί χρήστες επιλέγουν να τον εγκαταστήσουν σε Raspberry Pi, ώστε να δημιουργήσουν ένα οικονομικό και πλήρως λειτουργικό μουσικό σταθμό και επιλέγουν να το τρέχουν ως service, δηλαδή να τρέχει στο παρασκήνιο οπότε να ξεκινάει αυτόματα όταν εκκινεί το σύστημα. Την επιλογή να τρέχει από τερματικό την χρησιμοποιούν κυρίως χρήστες που αναπτύσσουν επεκτάσεις στο Mopidy οπότε θέλουν να βλέπουν διαρκώς την έξοδο log, και να έχουν τη δυνατότητα να σταματούν και να εκκινούν το Mopidy με τη χρήση εντολών.

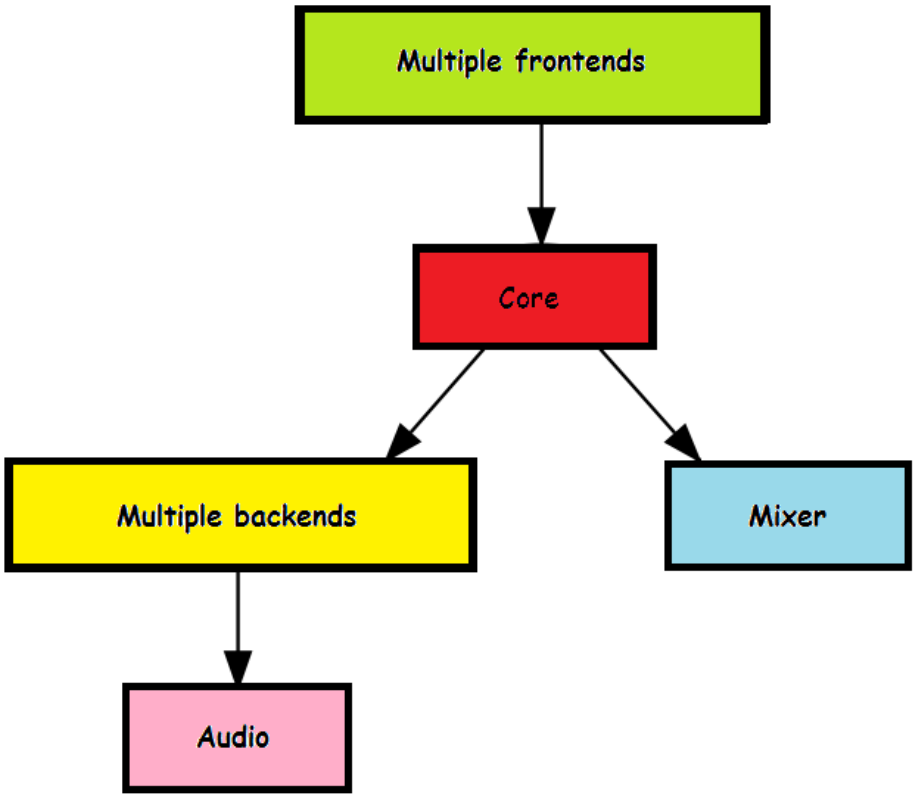
Για να προσθέτουμε frontends και backends ώστε να έχουμε υποστήριξη αναπαραγωγής μουσικής από πολλαπλές πηγές, εγκαθιστούμε επεκτάσεις (extensions). Το Mopidy διαθέτει ένα αρχείο ρυθμίσεων που ονομάζεται mopidy.conf το οποίο κάθε φορά που προσθέτουμε μια επέκταση στο Mopidy server, στο αρχείο ρυθμίσεων προστίθεται το αντίστοιχο κομμάτι με τις ρυθμίσεις της επέκτασης, το οποίο μετά μπορούμε να το τροποποιήσουμε όπως επιθυμούμε. Οι επεκτάσεις αυτές είναι αρχικά ενεργοποιημένες,

ακόμα και αν αυτό δεν αναφέρεται μέσα στο αρχείο ρυθμίσεων. Τις επεκτάσεις αυτές μπορούμε να τις ενεργοποιούμε και να τις απενεργοποιούμε χωρίς να χρειάζεται να τις απεγκαταστήσουμε, τροποποιώντας στο αρχείο ρυθμίσεων την ρύθμιση enabled που αντιστοιχεί στην επέκταση που θέλουμε να ενεργοποιήσουμε ή να απενεργοποιήσουμε και θέτοντάς τη σε τιμή true ή false αντίστοιχα. Επίσης διαθέτει ένα επικυρωτή ρυθμίσεων (config validator) που ελέγχει όλες τις τιμές των ρυθμίσεων στο αρχείο ρυθμίσεων και ειδοποιεί με μήνυμα λάθους αν έχουμε εισάγει λάθος ρυθμίσεις.

Κυρίως είναι ένας HTTP σέρβερ ο οποίος ελέγχεται με Web clients, εκτός εάν ο χρήστης επιθυμεί να εγκαταστήσει την επέκταση Mopidy – MPD ώστε να λειτουργεί και ως MPD σέρβερ και να ελέγχεται με MPD clients. Το μεγάλο του πλεονέκτημα είναι ότι σε συνδυασμό με τις υπάρχουσες διαθέσιμες επεκτάσεις του και με τη βοήθεια της γλώσσας προγραμματισμού Python, των JavaScript APIs και Jason-PRC (JavaScript Object Notation) μπορεί κάποιος να πειραματιστεί, να τροποποιήσει, ακόμα και να δημιουργήσει νέα δικά του πρότζεκτ και νέες επεκτάσεις. [122] [125]

2.2.7 Αρχιτεκτονική του Mopidy Music Server

Η αρχιτεκτονική του Mopidy [126] [122] οργανώνεται σε πολλαπλά frontends και backends. Τα frontends υλοποιούν τη διεπαφή προγραμματισμού εφαρμογών (API) core API. Το core actor μπορεί να συνδυάσει πολλά backends τα οποία συνδέονται σε διάφορες μουσικές πηγές, ώστε να λειτουργούν ως μια οντότητα. Το core actor χρησιμοποιεί το mixer actor για τον έλεγχο του ήχου, ενώ τα backends χρησιμοποιούν το audio actor για να επιτευχθεί η αναπαραγωγή ήχου.



Εικόνα 31: Αρχιτεκτονική Mopidy

Η βασική ιδέα είναι ότι τα επιμέρους επίπεδα δηλαδή τα frontend, backend και audio επίπεδα είναι πλήρως διαχωρισμένα και ότι το frontend επίπεδο καλεί το core επίπεδο και το core καλεί το backend επίπεδο. Το backend επίπεδο δεν έχει πρόσβαση στο core επίπεδο. Και το core επίπεδο και τα backends επιτρέπεται να καλούν το audio επίπεδο. Η ροή των πληροφοριών προς την αντίθετη κατεύθυνση γίνεται με αποστολή συμβάντων (events) στους listeners όπως για παράδειγμα μέσω του CoreListener και του AudioListener [122].

2.2.7.1 Frontend

Τα frontends επιτρέπουν την επικοινωνία του Mopidy με τον έξω κόσμο μέσω διαφόρων clients. Μπορούν να υλοποιούν servers για πρωτόκολλα όπως τα πρωτόκολλα HTTP, MPD και MPRIS, να δημιουργούν threads και να ανοίγουν TCP πόρτες. Τα frontends για να είναι λειτουργικά θα πρέπει να εισάγουμε τις κατάλληλες ρυθμίσεις που αντιστοιχούν

στα frontends αυτά στο αρχείο ρυθμίσεων, καθώς και να τα ενεργοποιήσουμε θέτοντας στην ρύθμιση enabled την τιμή true . Ένα frontend θα πρέπει να υλοποιεί τουλάχιστον ένα Pykka actor, που θα ονομάζεται main actor και το οποίο θα μπορεί να ξεκινάει και να σταματάει το frontend όταν το actor αυτό ξεκινάει ή σταματάει. Ο main actor δέχεται δύο constructor arguments, το config το οποίο είναι ένα dict structure με όλες τις ρυθμίσεις του Mopidy και το core το οποίο είναι ένα ActorProxy για το core actor. Αυτό δίνει πρόσβαση στο Core API (mopidy.core). Σε περίπτωση που οι τιμές των ρυθμίσεων που ορίσαμε στο αρχείο ρυθμίσεων κάποιου frontend δεν είναι οι κατάλληλες, το main actor θα πρέπει να μας ειδοποιήσει με μήνυμα λάθους με τη βοήθεια του mopidy.exceptions.FrontendError. Επίσης κάθε actor που ανήκει στα frontends, θα πρέπει να χρησιμοποιεί το mopidy.core.CoreListener API ώστε να λαμβάνει ειδοποιήσεις σχετικά με τα συμβάντα (events) και αλλαγές που λαμβάνουν χώρα στο core [122]

2.2.7.2 Core

Το core [122] οργανώνεται ως ένα σύνολο από ελεγκτές (controllers) οι οποίοι είναι υπεύθυνοι για τον έλεγχο διαφόρων λειτουργιών (functionalities) και ουσιαστικά παρεμβάλλεται μεταξύ των frontends και των backends. Είναι ο actor στον οποίο στέλνουν τα αιτήματά τους οι frontends. Για κάθε τέτοιο αίτημα το core καλεί ένα ή περισσότερα backends τα οποία εκτελούν τις απαιτούμενες εργασίες, και όταν τα backends ανταποκριθούν, το core actor είναι υπεύθυνο να συνδυάσει τις επιμέρους αποκρίσεις σε μία μόνο απόκριση στο frontend που στέλνει το αίτημα. Το core actor καταγράφει επίσης και τη λίστα αναπαραγωγής, αφού δεν ανήκει σε κάποιο συγκεκριμένο backend. Το core υλοποιείται με το Core API.

Το core αποτελείται από τους εξής ελεγκτές :

- **Tracklist Controller**

Διαχειρίζεται οτιδήποτε σχετίζεται με τη λίστα των κομματιών που θέλουμε να αναπαράγουμε. Για παράδειγμα μπορεί να προσθέτει, να αφαιρεί, να μετακινεί, να ανακατεύει ή να διαγράφει τα κομμάτια από το tracklist, να υπολογίζει τον αριθμό κομματιών που ανήκουν στο tracklist, να φιλτράρει το tracklist, να παρέχει πληροφορίες σχετικά με το προηγούμενο ή το επόμενο κομμάτι, να επιστρέφει τη λίστα του tracklist ή μέρος αυτής της λίστας και να θέτει την επιλογή του τρόπου αναπαραγωγής σε διάφορες επιλογές όπως consume, random, repeat και single ή να ενημερώνει για την τρέχουσα επιλογή του τρόπου αναπαραγωγής.

- **Playback Controller**

Διαχειρίζεται την κατάσταση αναπαραγωγής και το τρέχον κομμάτι που αναπαράγεται. Αναπαράγει κάποιο συγκεκριμένο κομμάτι, μεταβαίνει σε προηγούμενο ή επόμενο κομμάτι, διακόπτει, παύει και επανεκκινεί την αναπαραγωγή και μπορεί να μεταβεί σε μια συγκεκριμένη χρονική στιγμή ενός κομματιού. Ακόμα μπορεί να ενημερώσει για το ποιο είναι το τρέχον κομμάτι ή stream που αναπαράγεται, σε ποια χρονική στιγμή βρίσκεται η αναπαραγωγή του κομματιού και να ορίσει ή να ενημερώσει για την κατάσταση αναπαραγωγής, δηλαδή αν είναι σε κατάσταση αναπαραγωγής (PLAYING), παύσης (PAUSED) ή διακοπής (STOPPED).

- **Library Controller**

Διαχειρίζεται τη μουσική βιβλιοθήκη όπως πχ αναζήτηση κομματιών με συγκεκριμένα κριτήρια όπως όνομα κομματιού, το URI του, όνομα δίσκου, καλλιτέχνη, συνθέτη, εκτελεστή, μουσικό είδος, ημερομηνία κυκλοφορίας κλπ., καθώς και περιήγηση μεταξύ των διαθέσιμων κομματιών. Παρέχει επίσης τη δυνατότητα ανανέωσης της μουσικής βιβλιοθήκης και αναζήτησης εικόνων που σχετίζονται με συγκεκριμένα κομμάτια, όπως πχ εξώφυλλα δίσκων.

- **Playlists Controller**

Είναι ένα controller που διαχειρίζεται τις αποθηκευμένες λίστες αναπαραγωγής (playlists). Δηλαδή μπορεί να δημιουργήσει ή να διαγράψει μια λίστα αναπαραγωγής, να αποθηκεύσει αλλαγές σε μια playlist, να επιστρέψει λίστα με URI schemes που υποστηρίζουν playlists, λίστα με όλες τις διαθέσιμες playlists, να αναζητήσει playlists με βάση το URI τους και να ανανεώσει τις playlists που σχετίζονται με κάποιο backend ή και με όλα τα backend.

- **Mixer Controller**

Διαχειρίζεται τα επίπεδα της στάθμης ήχου και την κατάσταση σίγασης. Μπορεί να θέτει το σύστημα σε κατάσταση σίγασης ή το αντίστροφο και να ρυθμίσει το επίπεδο της στάθμης ήχου αναπαραγωγής. Επίσης μπορεί να δώσει πληροφορίες σχετικά με την τρέχουσα κατάσταση σίγασης, καθώς και για το τρέχον επίπεδο της στάθμης του ήχου.

- History Controller

Τηρεί αρχείο με το ποια κομμάτια έχουν αναπαραχθεί καθώς και δίνει πληροφορίες για τον αριθμό των κομματιών που περιέχονται στο ιστορικό.

Core Listener

Το Core Listener [122] είναι υπεύθυνο να ειδοποιεί για τις συμβάντα που συμβαίνουν στο επίπεδο Core. Για παράδειγμα ειδοποιεί αν αλλάξει η κατάσταση σίγασης, αν αλλάξει η κατάσταση consume, single, repeat και random, αν αλλάξει η κατάσταση αναπαραγωγής, αν γίνει έναρξη, παύση, επανέναρξη ή διακοπή αναπαραγωγής, αν κάποια playlist αλλάξει, φορτωθεί ή διαγραφεί, αν αλλάξει ο τίτλος του stream που αναπαράγεται καθώς και για τυχόν αλλαγές στο tracklist και στη στάθμη αναπαραγωγής του ήχου.

2.2.7.3 Backend

Το επίπεδο Backend [122] παρεμβάλλεται μεταξύ του επιπέδου Core και του επιπέδου Audio και περιλαμβάνει τα backends. Τα backends οργανώνονται ως ένα σύνολο παρόχων υπηρεσιών υπεύθυνων για διαφορετικά σύνολα λειτουργικότητας (functionalities), παρόμοια με το core actor. Για να συμπεριλάβουμε νέες μουσικές πηγές στο Mopidy server, προσθέτουμε ένα νέο backend. Όταν το core δέχεται κάποιο αίτημα από έναν client, δρομολογεί το αίτημα αυτό στο αντίστοιχο backend που απευθύνεται το αίτημα, βάσει των URI (Uniform Resource Identifier) που χρησιμοποιούνται από το αίτημα. Τα URI αυτά συγκρίνονται με την λίστα των URI schemes των backends, ώστε τελικά το αίτημα να δρομολογηθεί στο κατάλληλο backend. Σε περίπτωση υλοποίησης ενός καινούριου backend, θα πρέπει να δημιουργήσουμε το δικό μας URI scheme, το οποίο θα χρησιμοποιείται για όλα τα κομμάτια, τις λίστες αναπαραγωγής κλπ. που αφορούν το backend αυτό και το όνομά του θα σχετίζεται με το συγκεκριμένο backend. Αυτά τα URI θα μετατραπούν σε URI τα οποία κατανοεί το GStreamer, ώστε να είναι εφικτή η αναπαραγωγή τους από αυτό. Για να δημιουργηθεί ένα νέο backend, πρέπει να υλοποιηθεί το backend API.

Το backend layer αποτελείται από τους εξής παρόχους (providers) :

- **Library Provider**

Δέχεται αιτήματα από το core επίπεδο και εκτελεί διάφορες λειτουργίες που σχετίζονται με τη διαχείριση της μουσικής βιβλιοθήκης του backend που αντιστοιχεί στο αίτημα από το core επίπεδο και συγκεκριμένα από το Library Controller δηλαδή λειτουργίες όπως αναζήτησης μουσικής και περιήγησης στις μουσικές βιβλιοθήκες, ανανέωση βιβλιοθηκών και ανάκτηση εξωφύλλων δίσκων των ζητούμενων κομματιών.

- **Playback Provider**

Το Playback Provider διαχειρίζεται τα αιτήματα που δέχεται από το Playback Controller του επιπέδου Core για κάποιο συγκεκριμένο backend και προωθεί τα αιτήματα στο επίπεδο Audio. Κάποιες από τις λειτουργίες που εκτελεί είναι να ενημερώνει σε ποια χρονική στιγμή βρίσκεται το κομμάτι, να προωθεί στο Audio επίπεδο τις εντολές παύσης, έναρξης, διακοπής και επανέναρξης της αναπαραγωγής, μετατροπής ενός URI που είναι φτιαγμένο ειδικά για ένα backend σε URI που να μπορεί να το κατανοήσει ο GStreamer και μετάβαση κατά την αναπαραγωγή ενός κομματιού σε μια συγκεκριμένη χρονική στιγμή.

- **Playlist Provider**

Το Playlist Provider δέχεται εντολές από το Playlist Controller του Core επιπέδου και εκτελεί λειτουργίες σχετικά με τις υπάρχουσες playlists όπως ανανέωσης, αποθήκευσης, αναζήτησης και διαγραφής playlists καθώς και δίνει τη δυνατότητα δημιουργίας νέων playlists,.

Backend Listener

Όπως και στην περίπτωση των frontends, και εδώ υπάρχει το Backend Listener (mopidy.backend.BackendListener) το οποίο αλληλεπιδρά κυρίως με το core actor και το οποίο χρησιμεύει στο να ψάχνει ποιους actors να ειδοποιεί για τα συμβάντα που λαμβάνουν χώρα στο backend actor όπως για παράδειγμα ότι φορτωθήκανε ή ανανεωθήκανε οι playlists και να παρέχει default υλοποιήσεις στους listeners που δεν ενδιαφέρονται για τα συμβάντα αυτά [122].

2.2.7.4 Audio

Το audio actor [122] είναι ένα thin wrapper που περιλαμβάνει τα κομμάτια της βιβλιοθήκης GStreamer, τα οποία χρησιμοποιούνται από τα backends για την αναπαραγωγή του ήχου. Εάν υλοποιούμε ένα εξεζητημένο backend, μπορεί να χρειαστεί να υλοποιήσουμε ένα δικό μας πάροχο αναπαραγωγής (playback provider) χρησιμοποιώντας το Audio API, αλλά τα περισσότερα backends μπορούν να χρησιμοποιήσουν το βασικό πάροχο αναπαραγωγής χωρίς να χρειαστεί να κάνουμε αλλαγές. Εκτός των άλλων δέχεται αιτήματα από το τους frontends διαδοχικά μέσω του core επιπέδου και του backend επιπέδου και αυτό με τη σειρά του ελέγχει τον GStreamer, όπως για παράδειγμα λαμβάνει και δίνει εντολή στο GStreamer για έναρξη, λήξη και παύση της αναπαραγωγής, θέτει το URI του κομματιού που θα αναπαραχθεί, θέτει το χρονικό σημείο που θα γίνει η αναπαραγωγή του κομματιού κλπ., καθώς και από το Mixer actor για να ανακτήσει την κατάσταση σίγασης ή τη στάθμη του ήχου αναπαραγωγής, ή για να θέσει το σύστημα σε κατάσταση σίγασης και να αλλάξει τη στάθμη του ήχου.

Το Audio Listener (mopidy.audio.AudioListener) χρησιμεύει στο να ψάχνει ποιους Pykka actors να ειδοποιεί για τα συμβάντα που συμβαίνουν στον audio actor και να παρέχει default υλοποιήσεις στους listeners που δεν ενδιαφέρονται για τα συμβάντα αυτά. Στέλνει ειδοποιήσεις όταν η χρονική στιγμή της αναπαραγωγής αλλάξει, όταν η αναπαραγωγή ενός stream φτάσει στο τέλος της ή όταν το stream αλλάξει, όταν η κατάσταση αναπαραγωγής αλλάξει και αν τυχόν αλλάξουν τα μεταδεδομένα ενός stream [122].

2.2.7.5 Mixer

Ο mixer actor [122] είναι υπεύθυνος για τον έλεγχο του ήχου όπως την στάθμη του ήχου ή τη σίγαση. Ο default μίκτης είναι ο μίκτης software ο οποίος χρησιμοποιεί το audio actor για να ελέγχει τη στάθμη ήχου σε επίπεδο λογισμικού και μόνο εσωτερικά του Mopidy, στέλνοντας τον ήχο σε μια έξοδο του GStreamer. Εναλλακτικά χρησιμοποιούνται βιβλιοθήκες κατασκευασμένες από τρίτους σε Python, οι οποίες είναι ανεξάρτητες από τον audio actor. Με την εγκατάσταση κατάλληλων επεκτάσεων, μπορούν να χρησιμοποιηθούν και hardware μίκτες. Σε περίπτωση που παρουσιάζονται προβλήματα κατά την εκκίνηση του mixer, θα λάβουμε μήνυμα λάθους μέσω του mopidy.exceptions.MixerError ώστε να διορθώσουμε το πρόβλημα. Ο Mixer στέλνει αιτήματα στο audio actor σχετικά με τον έλεγχο της κατάστασης σίγασης, την τιμή της

στάθμης του παραγόμενου ήχου καθώς και ζητάει από το audio actor να θέσει τον GStreamer σε κατάσταση σίγασης, ή να του ορίσει συγκεκριμένη στάθμη ήχου αναπαραγωγής.

Ο Mixer Listener (mopidy.mixer.MixerListener) [122] χρησιμεύει στο να ψάχνει ποιους Pykka actors να ειδοποιεί για τα συμβάντα που λαμβάνουν χώρα στον mixer actor και να παρέχει default υλοποιήσεις στους listeners που δεν ενδιαφέρονται για τα συμβάντα αυτά. Βασικά στέλνει events αν αλλάξει η κατάσταση σίγασης και η στάθμη του ήχου του συστήματος.

2.2.8 Clients

Από τη στιγμή που ο Mopidy server ξεκινήσει να λειτουργεί, θα χρειαστεί ένα client για να τον ελέγχει. Η ήχος δεν μεταφέρεται (streaming) στον client, αλλά αναπαράγεται στο σύστημα που τρέχει το Mopidy. Στην υλοποίησή μας θα χρησιμοποιήσουμε τους web clients MusicBox [127] και Iris [128], καθώς και μια εφαρμογή έξυπνου κινητού τηλεφώνου σε περιβάλλον Android, που ονομάζεται Mopidy Mobile. [129]

2.2.8.1 MusicBox

Η επέκταση Mopidy MusicBox Webclient (MMW) είναι ένας web client βασισμένος στη JavaScript και κατασκευασμένος ειδικά για το Mopidy. Αρχικός δημιουργός του είναι ο Wouter van Wijk και συντηρείται από τον Nick Steel και δέχεται συνεισφορές από συνεισφέροντες (contributors). Προστατεύεται από την άδεια Apache 2.0 [113]. Ο σχεδιασμός του επιτρέπει να λειτουργεί σωστά τόσο σε browsers υπολογιστών, όσο και κινητών τηλεφώνων παρέχοντας και λειτουργία πλήρους οθόνης για όποιον το επιθυμεί. Συνεργάζεται με όλα τα backend και υποστηρίζει όλες τις επιλογές αναπαραγωγής του Mopidy, απεικονίζει τα εξώφυλλα των δίσκων στους οποίους ανήκουν τα αναπαραγόμενα κομμάτια, εφόσον διατίθενται και μετά την εγκατάστασή του λειτουργεί κανονικά με τις default ρυθμίσεις του, οπότε δεν χρειάζεται περεταίρω ρύθμιση των παραμέτρων του στο αρχείο ρυθμίσεων.

2.2.8.2 Iris

Ο web client Iris είναι μια επέκταση για το Mopidy ο οποίος κατασκευάστηκε και συντηρείται από τον James Barnsley και προστατεύεται και αυτός από την άδεια λογισμικού Apache 2.0 [113], διαθέτει αρκετά φιλικό προς τον χρήστη περιβάλλον και λειτουργεί άψογα με όλα τα backend του Mopidy.Δίνει τη δυνατότητα περιήγησης σε όλες τις βιβλιοθήκες και playlists και streaming μουσικής ταυτόχρονα σε διάφορους clients με τη βοήθεια του Snapcast και υποστηρίζει πολλαπλές γλώσσες.

2.2.8.3 Mopidy-Mobile

Η εφαρμογή Mopidy-Mobile είναι μια εφαρμογή για έξυπνα κινητά με λειτουργικό Android την οποία δημιούργησε ο Thomas Kemmer και μας επιτρέπει να ελέγχουμε με εύκολο τρόπο απομακρυσμένα ένα μουσικό σέρβερ Mopidy, με το κινητό μας τηλέφωνο.

Με την εφαρμογή αυτή μπορούμε να περιηγηθούμε και να αναζητήσουμε μουσική σε ολόκληρη τη βιβλιοθήκη του Mopidy, μόνο σε επιλεγμένους φακέλους, να επεξεργαστούμε τα κομμάτια στο τρέχον tracklist, να δημιουργούμε και να επεξεργαστούμε playlists, να ανακτήσουμε εξώφυλλα από επιλεγμένες διαδικτυακές πηγές, να μπορούμε να ελέγχουμε την αναπαραγωγή ενώ η οθόνη του κινητού μας είναι κλειδωμένη, να αυξομειώνουμε την ένταση του ήχου αναπαραγωγής χρησιμοποιώντας τα κουμπιά που χρησιμοποιούνται για την αυξομείωση του ήχου του κινητού μας, να εναλλασσόμαστε μεταξύ πολλαπλών Mopidy servers που βρίσκονται στο δίκτυό μας και επίσης υποστηρίζει πολλές γλώσσες.

2.2.9 Mopidy Επεκτάσεις

Για την υλοποίησή μας θα χρειαστεί να εγκαταστήσουμε και κάποιες επεκτάσεις ώστε να έχουμε δυνατότητα αναπαραγωγής από συγκεκριμένες μουσικές πηγές. Αυτές είναι οι παρακάτω :

- Mopidy-Youtube [130], κατασκευασμένη από τον Janez Troha, και τώρα συντηρείται από τον Nikolas Tumbri με τη συνεισφορά διαφόρων συνεισφερόντων (contributors). Μας επιτρέπει αναπαραγωγή μουσικής από το Youtube.

- Mopidy-TuneIn [131], κατασκευασμένη από τον Nick Steel και τους συνεισφέροντες (contributors) και καλύπτεται από την άδεια Apache -2.0. Μας επιτρέπει αναπαραγωγή μουσικής από το TuneIn.

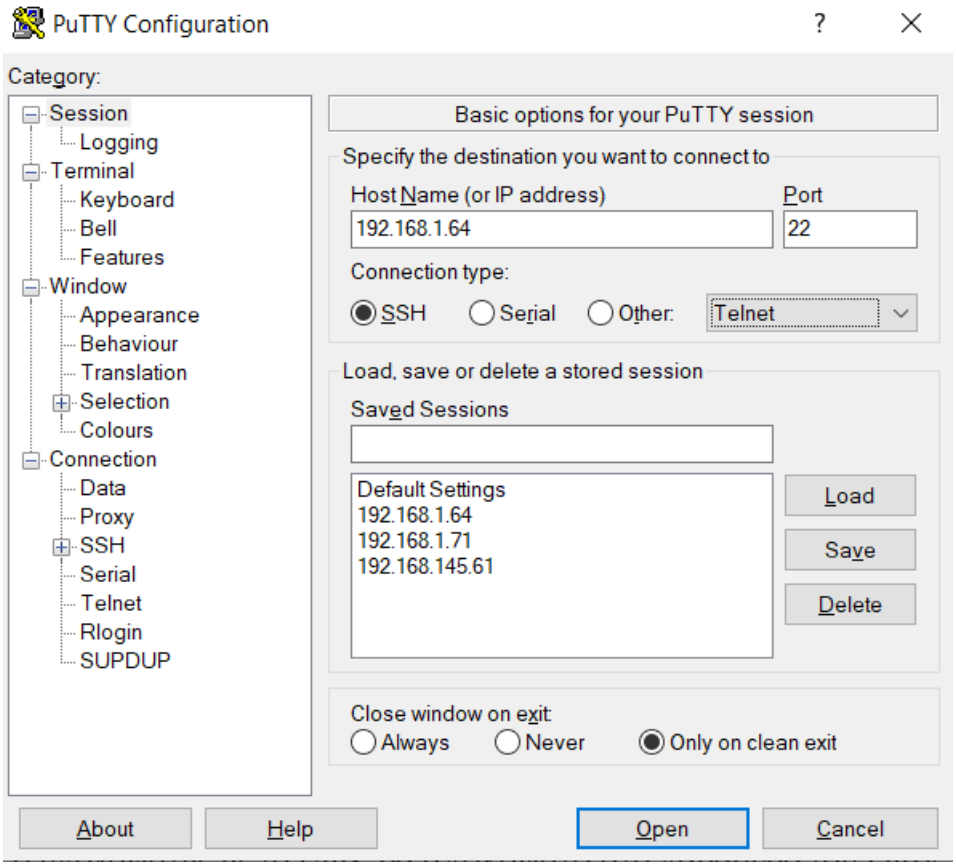
- Mopidy-Bandcamp [132], κατασκευασμένη από τον Dave Maez και τους συνεισφέροντες (contributors) και καλύπτεται από την άδεια MIT. Μας επιτρέπει αναπαραγωγή μουσικής από το Bandcamp.

- Mopidy-InternetArchive [133] κατασκευασμένη από τον Thomas Kemmer και τους και τους συνεισφέροντες (contributors) και καλύπτεται από την άδεια Apache-2.0. Μας επιτρέπει αναπαραγωγή μουσικής από το InternetArchive.

- Mopidy-Local [134], κατασκευασμένη από τον Stein Magnus Jodal και τον Thomas Adamcik. Το κομμάτι του SQLite storage και η υποστήριξη για ενσωματωμένο album art. Στην επέκταση αυτή συνεισφέρουν και άλλοι συνεισφέροντες (contributors). Μας επιτρέπει αναπαραγωγή μουσικής από τοπικά μέσα αποθήκευσης.

2.2.10 Putty

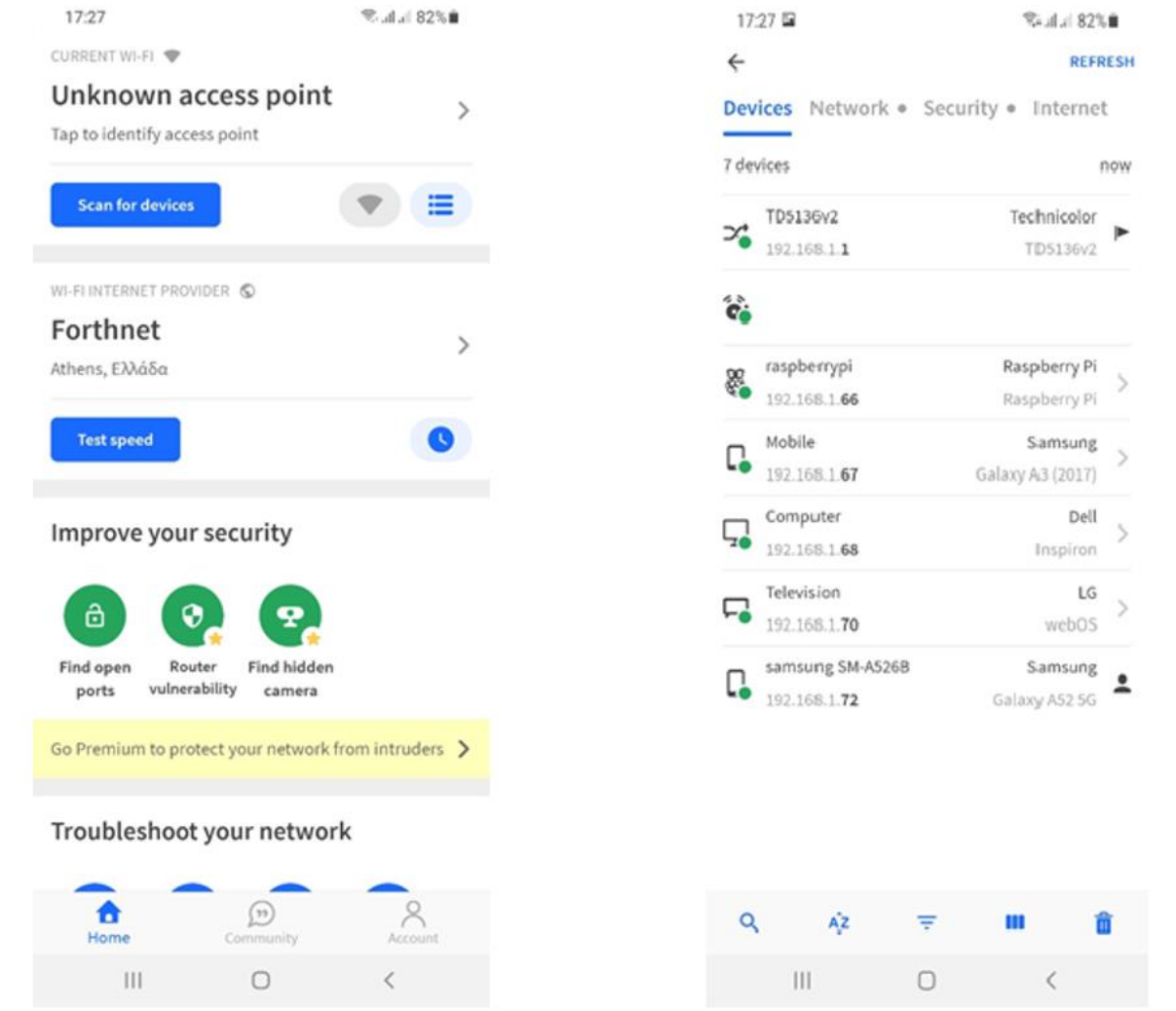
Το Putty [135] είναι ένας δωρεάν εξομοιωτής ανοιχτού κώδικα, και χρησιμοποιείται για την υλοποίηση πρωτοκόλλων δικτύου όπως τα SSH, Telnet, Rlogin και SUPDUP σε πλατφόρμες Windows και Unix. Τα πρωτόκολλα αυτά χρησιμοποιούνται για να τρέχουμε απομακρυσμένες συνεδρίες σε ένα υπολογιστή, μέσω δικτύου. Δημιουργός του είναι ο προγραμματιστής Simon Tatham, ο οποίος είναι και υπεύθυνος για τη συντήρησή του. Στην ουσία το Putty υλοποιεί το client άκρο σε μία απομακρυσμένη συνεδρία. Για παράδειγμα το Putty τρέχει σε ένα υπολογιστή με Windows και μέσω αυτού συνδέεται σε ένα άλλο υπολογιστή που τρέχει για παράδειγμα Unix και ότι εντολή δώσουμε στο Putty, θα σταλεί στον υπολογιστή με το Unix και ότι επιστρέψει ο υπολογιστής με το Unix, θα απεικονιστεί στο παράθυρο του Putty. [136]



Εικόνα 32: Εφαρμογή Putty

2.2.11 Fing

Η εφαρμογή Fing [137] η οποία είναι διαθέσιμη σε Android OS ή iOS είναι ένα εργαλείο το οποίο μας επιτρέπει να σαρώσουμε το τοπικό μας δίκτυο και να ανακαλύψει όλες τις συσκευές που είναι συνδεδεμένες στο δίκτυο αυτό, να εντοπίσει τυχόν εισβολείς και να εκτελέσει μετρήσεις της ταχύτητας του Ίντερνετ. Με το εργαλείο αυτό θα μπορούμε εύκολα με τη χρήση του κινητού μας τηλεφώνου να βρούμε την IP του Raspberry Pi που είναι συνδεδεμένο στο δίκτυο αυτό είτε ασύρματα είτε ενσύρματα, χωρίς να χρειάζεται να έχουμε συνδεδεμένη οθόνη πάνω του. Αφού βρούμε την διεύθυνση IP, μπορούμε να συνδεθούμε απομακρυσμένα με τη βοήθεια εφαρμογών όπως το Putty.



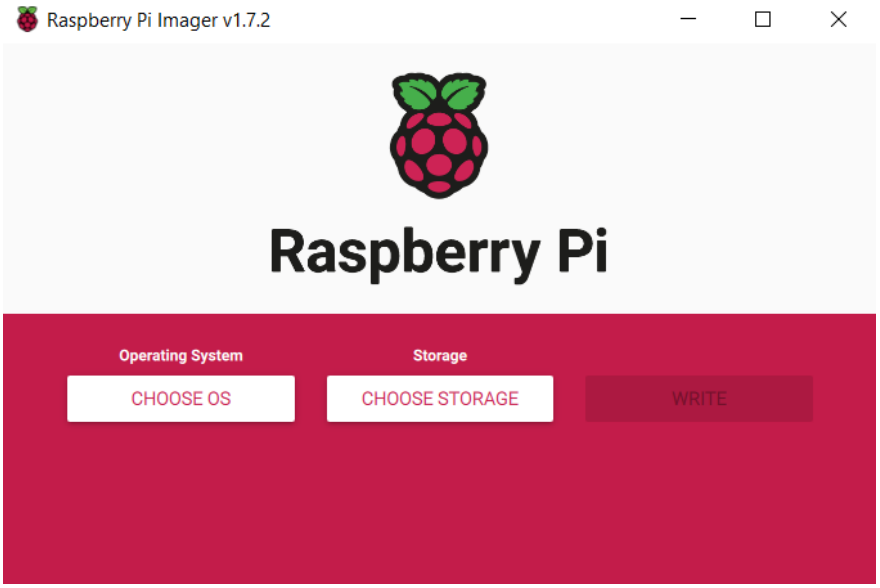
Εικόνα 33: Εφαρμογή Fing

ΚΕΦΑΛΑΙΟ 3:
Εφαρμογή και Αποτελέσματα

3.1 Υλοποίηση Music Streamer

3.1.1 Εγκατάσταση Raspberry Pi OS

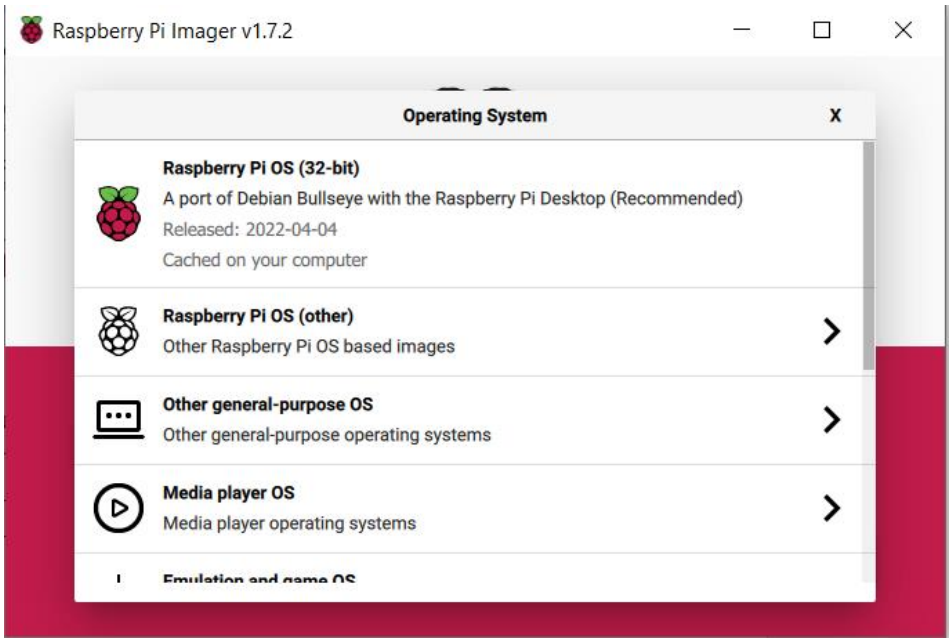
Για να εγκαταστήσουμε το Raspberry Pi OS στο Raspberry Pi, θα χρειαστούμε ένα ηλεκτρονικό υπολογιστή που να διαθέτει λειτουργικό Windows, Linux ή MacOS, card reader που να διαβάζει micro SD κάρτες και σύνδεση στο διαδίκτυο. Η κάρτα SD θα πρέπει να είναι χωρητικότητας τουλάχιστον 8 GB, εκτός εάν εγκαταστήσουμε την ελαφριά έκδοση του Raspberry Pi OS (lite), όπου αρκεί και μία κάρτα χωρητικότητας 4 GB. Έπειτα κατεβάζουμε το εργαλείο Raspberry Pi Imager [138] και το εγκαθιστούμε στον υπολογιστή μας.



Εικόνα 34: Εγκατάσταση Raspberry Pi OS

Τοποθετούμε την κάρτα SD στο card reader και αγνοούμε τυχόν μηνύματα για διαμόρφωση της κάρτας SD.

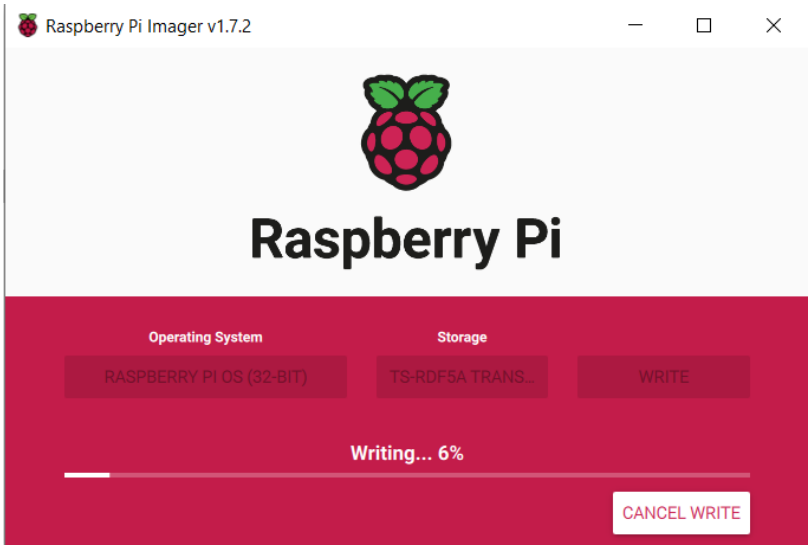
Τρέχουμε το εργαλείο Raspberry Pi Imager και από την επιλογή 'CHOOSE OS' επιλέγουμε την επιθυμητή επιλογή. Στην περίπτωσή μας θα επιλέξουμε την πρώτη επιλογή η οποία είναι και η προτεινόμενη και από το εργαλείο Raspberry Pi Imager, την επιλογή 'Raspberry Pi OS(32-bit)'. Επιλέγοντας αυτή την επιλογή το εργαλείο θα κατεβάσει και έπειτα θα εγκαταστήσει την τελευταία έκδοση του λειτουργικού με γραφικό περιβάλλον. Εάν δεν θέλουμε να χρησιμοποιούμε το Raspberry Pi για άλλες χρήσεις αλλά μόνο για το μουσικό server, μπορούμε να εγκαταστήσουμε και την έκδοση Raspberry Pi Lite. Εάν επιθυμούμε να εγκαταστήσουμε την έκδοση Raspberry Pi OS Lite, θα πρέπει να επιλέξουμε την επιλογή 'Raspberry Pi OS (other). Υπάρχει η δυνατότητα να έχουμε κατεβάσει πρώτα το Raspberry Pi OS ως αρχείο image (.img) οπότε να επιλέξουμε την τελευταία επιλογή 'Use custom και να επιλέξουμε το αρχείο image από τον φάκελο που το έχουμε αποθηκεύσει.



Εικόνα 35: Εγκατάσταση Raspberry Pi OS

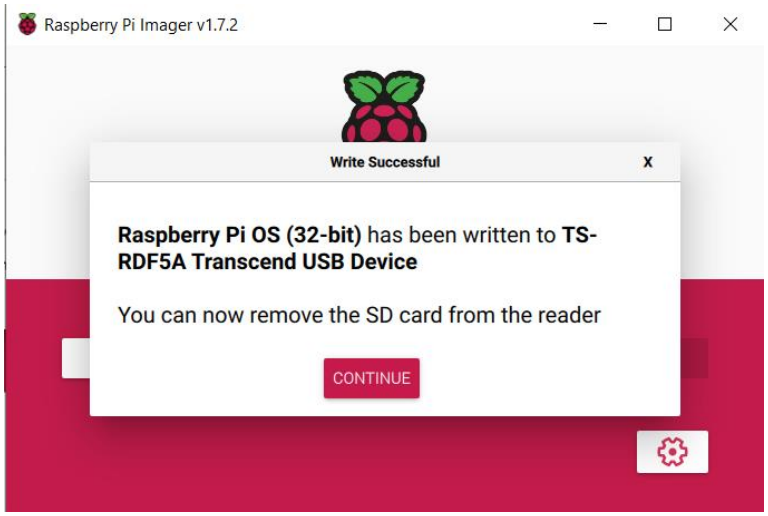
Επόμενο βήμα είναι να επιλέξουμε την επιλογή 'CHOOSE STORAGE'. Από το αναδυόμενο παράθυρο επιλέγουμε την κάρτα SD στην οποία θα εγκαταστήσουμε το λειτουργικό σύστημα το Raspberry Pi. Θέλει ιδιαίτερη προσοχή ώστε να μην επιλέξουμε λάθος κάρτα, και διαγράψουμε οριστικά τα δεδομένα που περιέχει. Αφού επιλέξουμε και την κάρτα SD, πατάμε 'WRITE'.

Αφού ενημερωνόμαστε ότι αν συνεχίσουμε τη διαδικασία θα διαγραφούν όλα τα δεδομένα που περιέχονται στην κάρτα SD, επιλέγουμε ‘YES’.



Εικόνα 36: Εγκατάσταση Raspberry Pi OS

Η διαδικασία διαρκεί αρκετά λεπτά, και κατά τη διάρκεια εγκατάστασης δεν πρέπει να αφαιρέσουμε την κάρτα χωρίς να επιλέξουμε ‘CANCEL WRITE’, γιατί μπορεί να προκαλέσουμε ανεπανόρθωτη βλάβη στην κάρτα.



Εικόνα 37 : Εγκατάσταση Raspberry Pi OS

Αφού ολοκληρωθεί η διαδικασία εγκατάστασης πατάμε ‘CONTINUE’, αφαιρούμε την κάρτα SD και την τοποθετούμε στην κατάλληλη υποδοχή του Raspberry Pi.

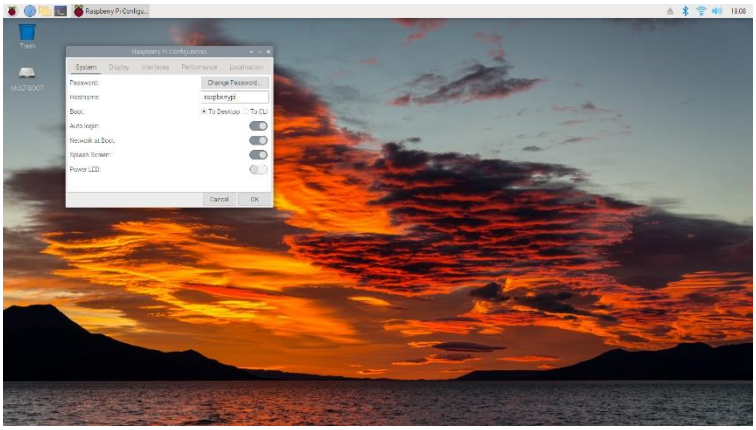
Εκκίνηση του Raspberry Pi

Εκτός της πλακέτας του Raspberry Pi, θα χρειαστούμε ένα USB πληκτρολόγιο και ένα ποντίκι, καλώδιο HDMI, το προτεινόμενο από τον κατασκευαστή τροφοδοτικό και μία οθόνη υπολογιστή με είσοδο HDMI. Μπορούμε να χρησιμοποιήσουμε και τηλεόραση, αν θέλουμε. Εάν η οθόνη έχει ενσωματωμένα ηχεία, θα μπορούμε να ακούμε τον ήχο μέσω του HDMI. Εάν η οθόνη έχει είσοδο DVI ή VGA, θα χρειαστούμε τον κατάλληλο προσαρμογέα από HDMI σε DVI ή VGA αντίστοιχα αλλά σε αυτές τις περιπτώσεις μέσω του καλωδίου θα μεταφέρεται μόνο εικόνα αλλά όχι ήχος. Σε αυτή την περίπτωση πρέπει να χρησιμοποιήσουμε κάποια πρόσθετη κάρτα ήχου, ή να ακούσουμε τον ήχο μέσω του 3.5 inch audio jack. Να σημειωθεί ότι το μοντέλο Raspberry Pi 400 δεν διαθέτει έξοδο ήχου 3.5 inch audio jack καθώς και ότι αυτά τα μοντέλα Raspberry Pi 4 και Raspberry Pi 400 για τη σύνδεση με την οθόνη χρειάζονται καλώδιο micro HDMI σε HDMI και τροφοδοσία με USB type C.

Αφού έχουμε τοποθετήσει την micro SD κάρτα στην θύρα υποδοχής στο Raspberry, τροφοδοτούμε την πλακέτα και εμφανίζεται η splash screen του λειτουργικού Raspberry Pi OS. Κατά την αρχική ρύθμιση του λειτουργικού θα μας ζητηθεί κάποιες φορές να κάνουμε επανεκκίνηση του συστήματος, ώστε να ενεργοποιηθούν οι επιλεγμένες ρυθμίσεις. Αρχικά θα ερωτηθούμε για την τοποθεσία που βρισκόμαστε, τη γλώσσα που χρησιμοποιούμε, τη ζώνη ώρας και αν θέλουμε να χρησιμοποιούμε την αγγλική γλώσσα και αμερικάνικο πληκτρολόγιο. Μετά θα μας ζητηθεί να ορίσουμε όνομα συστήματος και κωδικό πρόσβασης. Επόμενο βήμα είναι να επιλέξουμε αν θέλουμε το μέγεθος της επιφάνειας εργασίας να ταιριάζει με το μέγεθος της οθόνης. Κατόπιν θα μας ζητηθεί να επιλέξουμε από τα διαθέσιμα δίκτυα Wi-Fi. Αν δεν θέλουμε να παρακάμψουμε αυτό το βήμα, επιλέγουμε σε ποιο ασύρματο δίκτυο θέλουμε να συνδεθούμε και εισάγουμε τον κωδικό πρόσβασης. Έπειτα θα ερωτηθούμε αν θέλουμε να κάνουμε ενημέρωση του λογισμικού. Επιλέγουμε Next και αφού ενημερωθεί το λογισμικό, θα γίνει επανεκκίνηση ώστε να λάβουν χώρα οι αλλαγές. Επανεκκινούμε και το λειτουργικό είναι έτοιμο για χρήση.

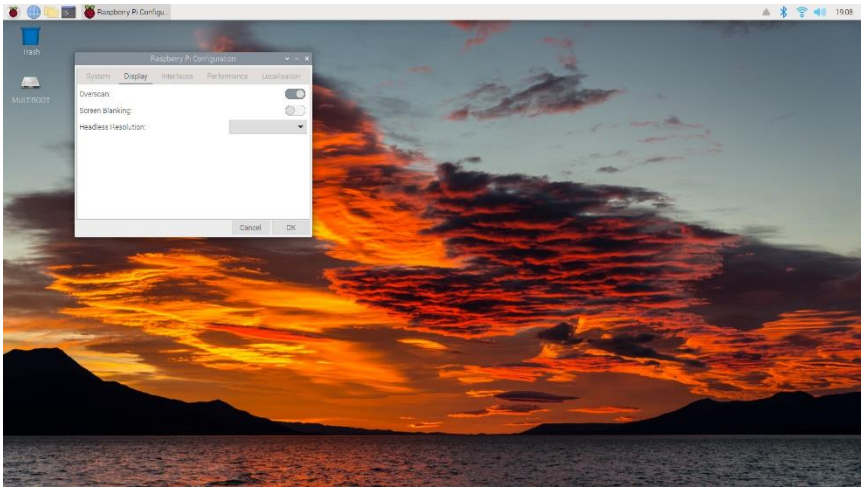
Θα χρειαστεί να κάνουμε μερικές ακόμα ρυθμίσεις μέσα από τις επιλογές του Raspberry Pi OS, και συγκεκριμένα πηγαίνουμε στο μενού 'Preferences / Raspberry Pi Configuration'.

Από την καρτέλα 'System' αλλάζουμε την επιλογή 'Network at Boot' σε 'Wait for network'.



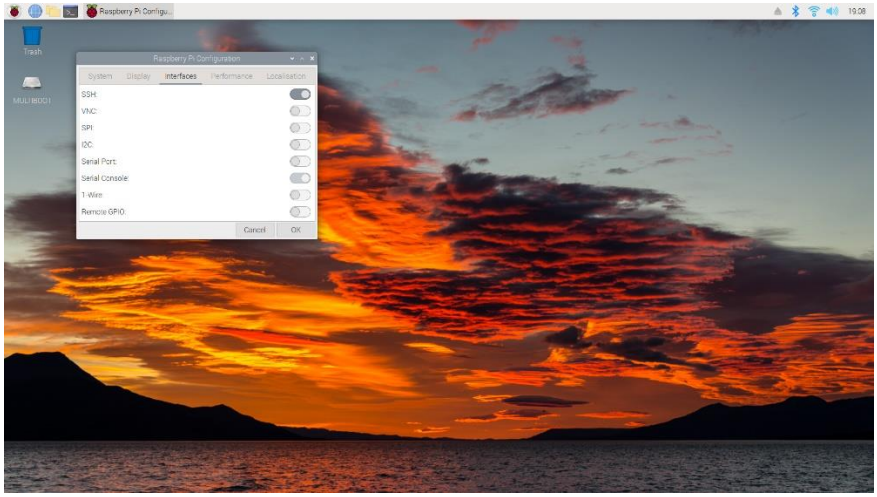
Εικόνα 38: Ρυθμίσεις Raspberry Pi OS

Από την καρτέλα 'Display' απενεργοποιούμε την λειτουργία 'Screen Blanking', σε περίπτωση που δεν θέλουμε να σβήνει η οθόνη άμα δεν χρησιμοποιούμε το ποντίκι ή το πληκτρολόγιο μετά από ένα χρονικό διάστημα.



Εικόνα 39: Ρυθμίσεις Raspberry Pi OS

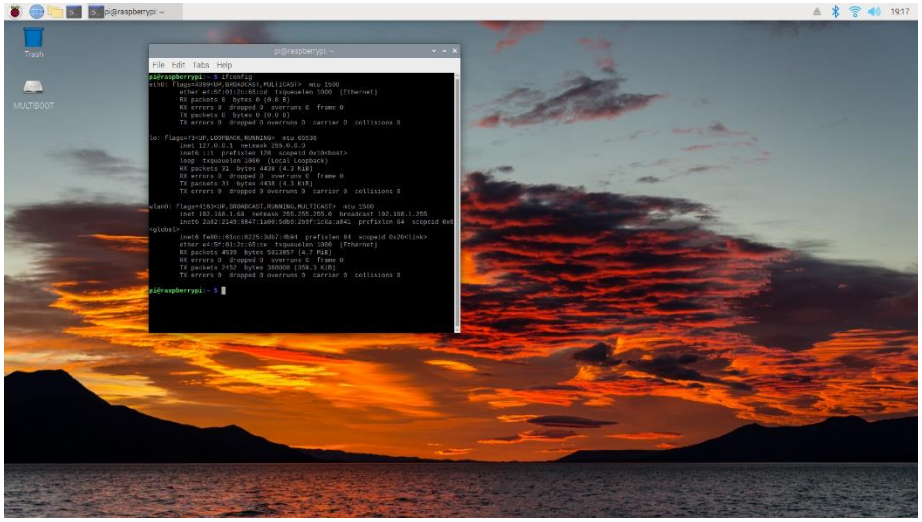
Από την καρτέλα ‘Interfaces’ ενεργοποιούμε την δυνατότητα απομακρυσμένης σύνδεσης με τη χρήση του πρωτοκόλλου secure shell ‘SSH’. Οπότε χρησιμοποιώντας την διεύθυνση IP του Raspberry και ένα πρόγραμμα client όπως το Putty, μπορούμε να συνδεόμαστε από άλλο υπολογιστή με το Raspberry, μέσω δικτύου.



Εικόνα 40: Ρυθμίσεις Raspberry Pi OS

Για να βρούμε την διεύθυνση IP του Raspberry, τρέχουμε την εφαρμογή Fing από ένα έξυπνο κινητό τηλέφωνο ή αλλιώς ανοίγουμε την κονσόλα εντολών (terminal) και πληκτρολογούμε την εντολή

ifconfig



Εικόνα 41: Εύρεση διεύθυνσης IP

Η διεύθυνση IP του συστήματος στην παραπάνω περίπτωση, είναι αυτή στο πεδίο eth0. Σε περίπτωση ασύρματης σύνδεσης, η διεύθυνση IP θα εντοπίζεται στο πεδίο wlan0. Η διεύθυνση 127.0.0.1 στο πεδίο lo είναι η διεύθυνση localhost, που δηλώνει τον ίδιο τον υπολογιστή, και χρησιμοποιείται για την εκτέλεση υπηρεσιών δικτύου χωρίς να χρειάζεται φυσική σύνδεση σε κάποιο δίκτυο (loopback).

Αφού ορίσουμε ως κύρια κάρτα του συστήματος το IQAudio Pro DAC+ σύμφωνα με τις οδηγίες που περιγράφονται στο κεφάλαιο 2.1.3 , προχωράμε στην εγκατάσταση του Mopidy.

3.1.2 Εγκατάσταση Mopidy, Clients και επεκτάσεις

(τελευταία εγκατάσταση 23/05/2022, έκδοση Mopidy-Youtube 3.5)

Για να εγκαταστήσουμε το Mopidy στο Raspberry Pi, ακολουθούμε τις οδηγίες που δίνονται στο εγχειρίδιο του Mopidy [122]. Ο ευκολότερος τρόπος είναι να το εγκαταστήσουμε από το Mopidy apt archive. Όταν εγκαθιστούμε από το APT archive, θα λαμβάνουμε αυτόματα ενημερώσεις για το Mopidy, με τον ίδιο τρόπο που λαμβάνουμε ενημερώσεις για το υπόλοιπο σύστημά μας. Τα πακέτα που παρέχονται στο apt.mopidy.com είναι διαθέσιμα για διάφορες αρχιτεκτονικές CPU, όπως amd64, i386 και armhf ,και είναι συμβατά με όλα τα μοντέλα του Raspberry Pi.

Παρακάτω παρατίθενται τα βήματα που πρέπει να ακολουθηθούν ώστε να εγκατασταθεί το Mopidy στο Raspberry Pi :

Βήμα 1. Καταρχάς κάνουμε ενημέρωση του συστήματος μας με την εντολή

sudo apt update

Έπειτα κάνουμε αναβάθμιση του συστήματος με την εντολή

sudo apt full-upgrade

Αφαιρούμε τα πακέτα που ανακτήσαμε με την εντολή

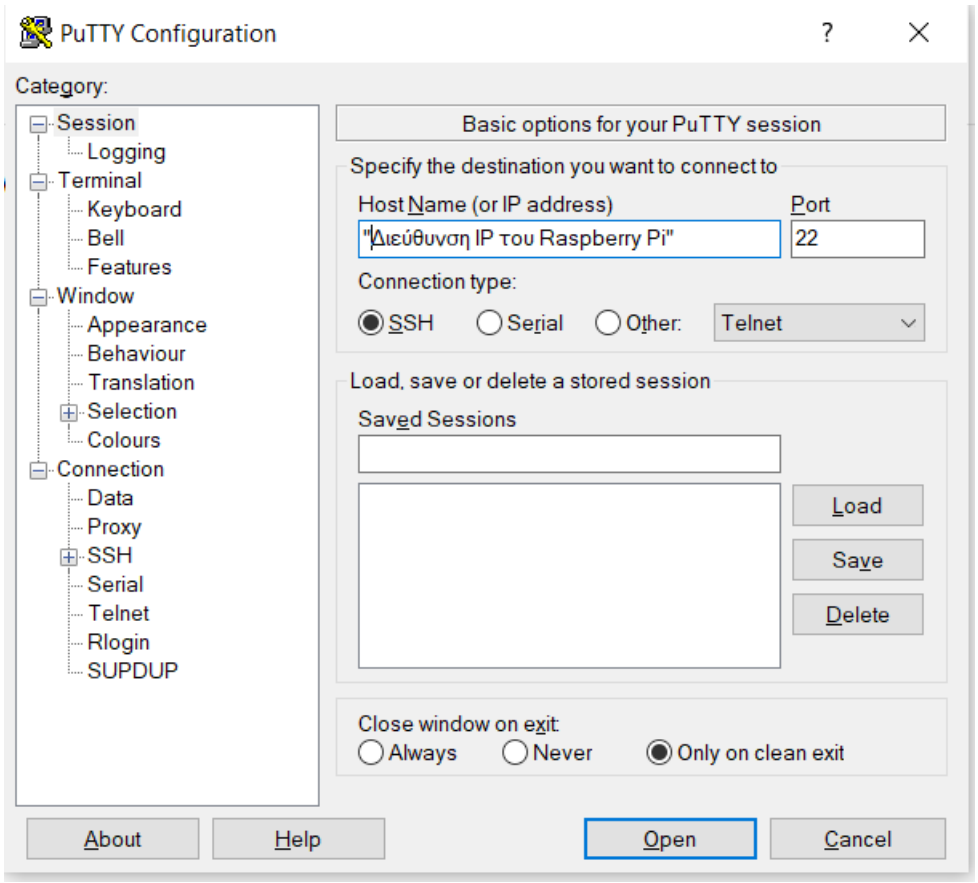
sudo apt clean

και τέλος επανεκκινούμε το σύστημά μας δίνοντας την εντολή

sudo reboot

Βήμα 2. Σε περίπτωση που θέλουμε να χρησιμοποιούμε το Raspberry Pi συνδεόμενοι απομακρυσμένα και όχι τοπικά, θα πρέπει να γνωρίζουμε την διεύθυνση IP του, και να χρησιμοποιήσουμε κάποιον SSH client, όπως το Putty. Απαραίτητη προϋπόθεση είναι να έχει ενεργοποιηθεί πρώτα στο Raspberry ο SSH server. Για να μάθουμε την διεύθυνση IP του συστήματός μας θα πρέπει να τρέξουμε την εντολή :

ifconfig



Εικόνα 42: Client Putty

Βήμα 3. Προσθέτουμε το κλειδί GPG και το APT repo των πακέτων που θέλουμε να κατεβάσουμε με τις εξής εντολές :

```
sudo mkdir -p /usr/local/share/keyrings
```

```
sudo wget -q -O /usr/local/share/keyrings/mopidy-archive-keyring.gpg \
https://apt.mopidy.com/mopidy.gpg
```

```
sudo wget -q -O /etc/apt/sources.list.d/mopidy.list https://apt.mopidy.com/buster.list
```

Βήμα 4. Ενημερώνουμε τα local repositories

```
sudo apt update
```

Βήμα 5. Εγκαθιστούμε το Mopidy και το pip που είναι ο εγκαταστάτης πακέτων της Python, για να εγκαθιστούμε πακέτα από το PyPI.

sudo apt install mopidy python3-pip

Βήμα 6. Χρησιμοποιούμε την εντολή pip για να εγκαταστήσουμε επεκτάσεις για το Mopidy. Θα εγκαταστήσουμε την επέκταση του Bandcamp με όνομα Mopidy-Bandcamp, του TuneIn με όνομα Mopidy-TuneIn, του Internet Archive με όνομα Mopidy-InternetArchive, την απαραίτητη επέκταση Mopidy-Local ώστε να μπορούμε να αναπαράγουμε μουσική από εξωτερικούς δίσκους και USB sticks και τους Web Clients που ονομάζονται MusicBox και Iris. Εννοείται πως το παράδειγμα είναι ενδεικτικό και δεν είναι απαραίτητο να εγκατασταθούν αυστηρά οι συγκεκριμένες προεκτάσεις και clients, αυτό επαφίεται στις προτιμήσεις του εκάστοτε χρήστη.

sudo python3 -m pip install Mopidy-Bandcamp Mopidy-TuneIn Mopidy-InternetArchive Mopidy-Local Mopidy-Iris Mopidy-MusicBox-Webclient

Εγκαθιστούμε την επέκταση του Mopidy-Youtube με τις εντολές:

sudo python3 -m pip install https://github.com/natumbri/mopidy-youtube/archive/develop.zip

και

sudo pip3 install youtube-dlc

Αφού εγκαταστήσαμε το Mopidy από το APT, θα εγκατασταθεί αυτόματα και το Pykka.

Για να βλέπουμε ποιες επεκτάσεις είναι διαθέσιμες στο apt.mopidy.com, θα τρέχουμε την εντολή :

apt search mopidy

3.1.3 Ρύθμιση Mopidy και επεκτάσεων

Αφού εγκατασταθεί το Mopidy και οι επιθυμητές επεκτάσεις, δηλαδή η επέκταση του YouTube, του Bandcamp, του TuneIn, του Internet Archive, του local και οι clients,

προχωρούμε στην ρύθμισή τους. Να σημειωθεί ότι υπάρχουν δύο τρόποι λειτουργίας του Mopidy, μέσω του τερματικού ή σαν service, το οποίο θα εκκινεί αυτόματα κατά την εκκίνηση του συστήματος. Για την πλειοψηφία των χρηστών είναι προτιμότερη η λειτουργία ως service, οπότε θα επικεντρωθούμε σε αυτή την κατεύθυνση. Η λειτουργία μέσω τερματικού απευθύνεται κυρίως σε προγραμματιστές οι οποίοι ενδιαφέρονται να βλέπουν συνέχεια τα log files και να μπορούν να εκκινούν, να σταματούν και να επανεκκινούν τον Mopidy όποτε το επιθυμούν.

Βήμα 1. Εκκινούμε το Mopidy service με την εντολή

```
sudo systemctl start mopidy
```

Αφού θα τρέχουμε το Mopidy ως service, θα εισάγουμε τις απαραίτητες ρυθμίσεις στο αρχείο mopidy.conf στην τοποθεσία /etc/mopidy/ με την εντολή :

```
sudo nano /etc/mopidy/mopidy.conf
```

Βήμα 2. Εισάγουμε τις παρακάτω ρυθμίσεις στο αρχείο mopidy.conf

```
[core]
cache_dir = /var/cache/mopidy
config_dir = /etc/mopidy
data_dir = /var/lib/mopidy
max_tracklist_length = 10000
restore_state = false

[logging]
verbosity = 0
format = %(levelname)-8s [%(threadName)s] %(name)s %(message)s
color = false
config_file =

[audio]
mixer = software
mixer_volume =
```

output = alsasink device= hw:IQaudIODAC
buffer_time =

[proxy]
scheme =
hostname =
port =
username =
password =

[youtube]
enabled = true
allow_cache =
youtube_api_key =
search_results = 15
playlist_max_videos = 20
api_enabled = false
channel_id =
musicapi_enabled = false
musicapi_cookie =
autoplay_enabled = false
strict_autoplay = false
max_autoplay_length = 600
max_degrees_of_separation = 3
youtube_dl_package = youtube_dlc

[tunein]
enabled = true
timeout = 5000
filter =

[musicbox_webclient]
enabled = true
musicbox = false

```
websocket_host =
websocket_port =
on_track_click = PLAY_ALL

[local]
enabled = true
max_search_results = 100
media_dir =  /media
scan_timeout = 1000
scan_flush_threshold = 100
scan_follow_symlinks = false
included_file_extensions =
excluded_file_extensions =
  .cue
  .directory
  .html
  .jpeg
  .jpg
  .log
  .nfo
  .pdf
  .png
  .txt
  .zip
directories =
  Albums                local:directory?type=album
  Artists               local:directory?type=artist
  Composers             local:directory?type=artist&role=composer
  Genres                local:directory?type=genre
  Performers            local:directory?type=artist&role=performer
  Release Years         local:directory?type=date&format=%25Y
  Tracks                local:directory?type=track
  Last Week's Updates   local:directory?max-age=604800
  Last Month's Updates  local:directory?max-age=2592000
```

```
timeout = 10
use_artist_sortname = false
album_art_files =
  *.jpg
  *.jpeg
  *.png

[iris]
enabled = true
country = NZ
locale = en_NZ
verify_certificates = true
snapcast_enabled = true
snapcast_host = localhost
snapcast_port = 1780
snapcast_ssl = false
snapcast_stream = Default
spotify_authorization_url = https://jamesbarnsley.co.nz/iris/auth_spotify.php
lastfm_authorization_url = https://jamesbarnsley.co.nz/iris/auth_lastfm.php
genius_authorization_url = https://jamesbarnsley.co.nz/iris/auth_genius.php
data_dir = $XDG_DATA_DIR/iris

[internetarchive]
enabled = true
base_url = http://archive.org
collections =
  audio
  etree
  librivoxaudio
  audio_bookspoetry
  audio_tech
  audio_music
  audio_news
  audio_foreign
```

```
  audio_podcast
  audio_religion
audio_formats =
  VBR MP3
  64Kbps MP3
image_formats =
  JPEG
  JPEG Thumb
browse_limit = 100
browse_views =
  downloads desc|Views
  titleSorter asc|Title
  publicdate desc|Date Archived
  date desc|Date Published
  creatorSorter asc|Creator
search_limit = 20
search_order =
cache_size = 128
cache_ttl = 86400
retries = 3
timeout = 10

[bandcamp]
enabled = true
discover_pages = 1
collection_items = 50
discover_genres =
  All
  Electronic
  Rock
  Metal
  Alternative
  Hip-Hop/Rap
  Experimental
```

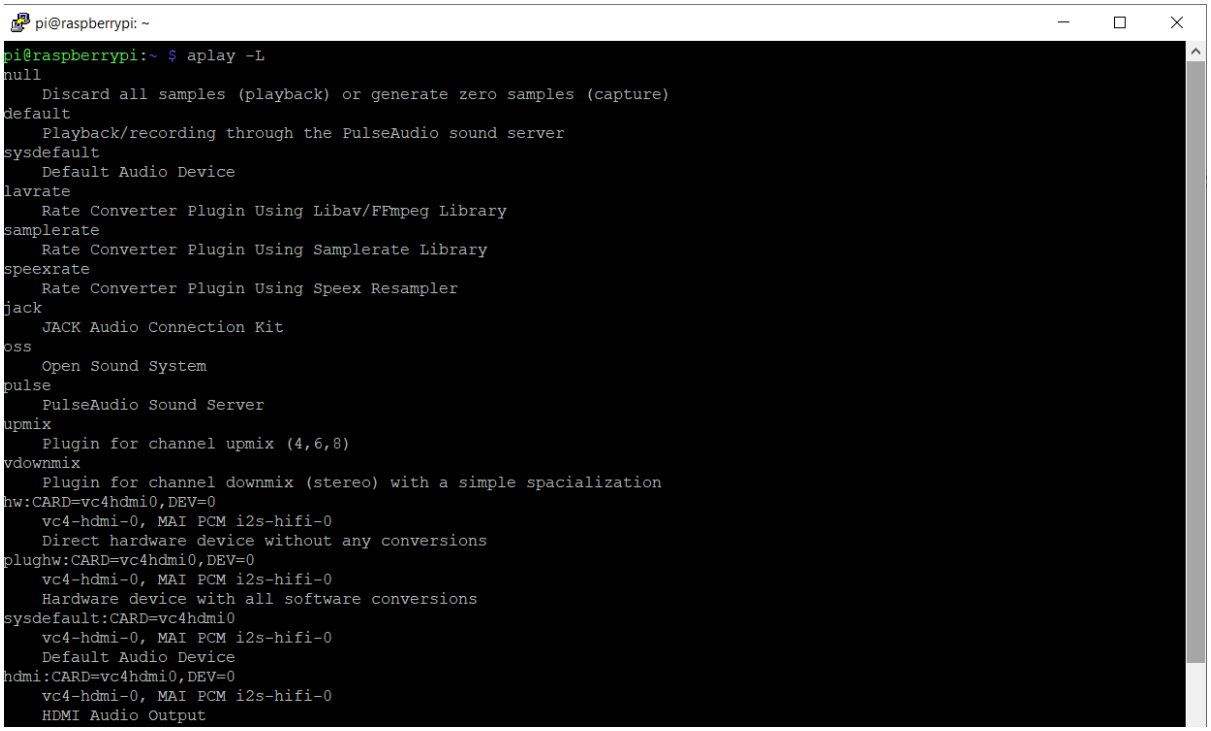
```
  Punk
  Folk
  Pop
  Ambient
  Soundtrack
  World
  Jazz
  Acoustic
  Funk
  R&B/Soul
  Devotional
  Classical
  Reggae
  Podcasts
  Country
  Spoken Word
  Comedy
  Blues
  Kids
  Audiobooks
  Latin
discover_tags =
  Outrun
  Future Funk
  Alternative Hip-Hop
  Tokyo, Japan
image_sizes =
  10
  5
  2
identity =

[file]
enabled = true
```

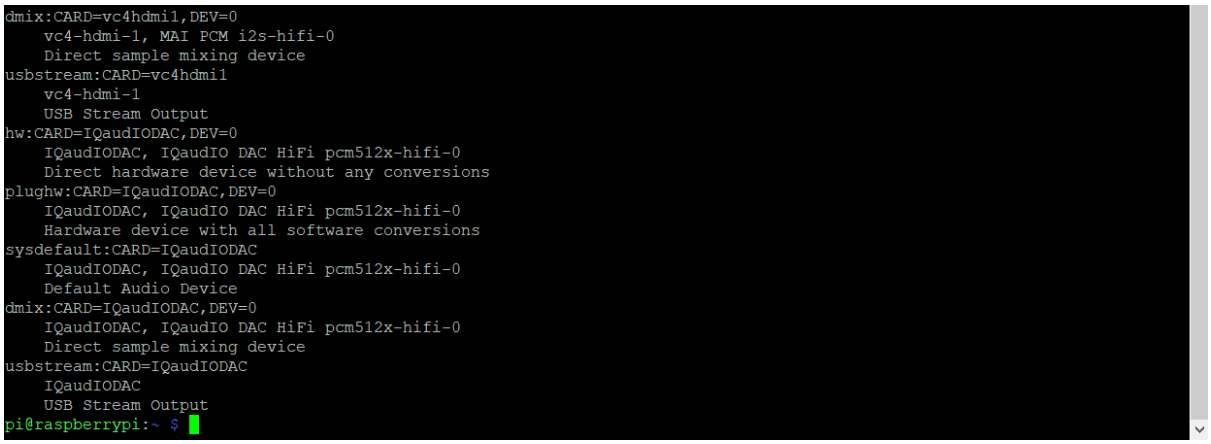
```
media_dirs =
  $XDG_MUSIC_DIR|Music
  ~/|Home
  /home/pi/Music
excluded_file_extensions =
  .directory
  .html
  .jpeg
  .jpg
  .log
  .nfo
  .pdf
  .png
  .txt
  .zip
show_dotfiles = false
follow_symlinks = false
metadata_timeout = 1000

[http]
enabled = true
hostname = ::
port = 6680
zeroconf = Mopidy HTTP server on $hostname
allowed_origins =
csrf_protection = true
default_app = mopidy

[m3u]
enabled = true
base_dir =
default_encoding = latin-1
default_extension = .m3u8
playlists_dir =
```

[softwaremixer]
enabled = true

[stream]
enabled = true
protocols =
  http
  https
  mms
  rtmp
  rtmps
  rtsp
metadata_blacklist =
timeout = 5000

Βήμα 3. Κανονικά η default επιλογή για την έξοδο ήχου το Mopidy είναι η επιλογή autoaudiosink. Όμως για να χρησιμοποιήσει το Mopidy ως έξοδο ήχου το IQAudioPro Dac+, τρέχουμε την εντολή aplay -L και βλέπουμε την παρακάτω λίστα :





Εικόνα 43: Εκτέλεση εντολής aplay -L

Εντοπίζουμε την κάρτα μας στο σημείο που αναφέρει

```
hw:CARD=IQaudIODAC,DEV=0
    IQaudIODAC, IQaudIO DAC HiFi pcm512x-hifi-0
    Direct hardware device without any conversions
```

οπότε τροποποιούμε το κομμάτι του αρχείου ρυθμίσεων που σχετίζεται με τον ήχο ως εξής :

[audio]
mixer = software
mixer_volume =
output = alsasink device= hw:IQaudIODAC
buffer_time =

Βήμα 3. Αφού εισάγουμε τις ρυθμίσεις στο αρχείο ρυθμίσεων mopidy.conf, το σώζουμε πιέζοντας το συνδυασμό πλήκτρων Ctrl + X, μετά πατάμε y και μετά Enter.

Βήμα 4. Για να ελέγξουμε αν το αρχείο ρυθμίσεων λειτουργεί σωστά και δεν έχει λάθη, τρέχουμε την εντολή :

sudo mopidyctl config

3.1.4 Εκτέλεση του Mopidy

Βήμα 1. Ενεργοποιούμε το Mopidy service. Εκτελώντας την εντολή αυτή, το Mopidy θα ξεκινάει αυτόματα κατά την εκκίνηση του συστήματος.

```
sudo systemctl enable mopidy
```

Βήμα 2. Εκκινούμε το Mopidy service.

```
sudo systemctl start mopidy
```

Εάν κάποια στιγμή θέλουμε να σταματήσουμε το Mopidy service πληκτρολογούμε :

```
sudo systemctl stop mopidy
```

Για να επανεκκινήσουμε το Mopidy service πληκτρολογούμε :

```
sudo systemctl restart mopidy
```

Βήμα 3. Ελέγχουμε ότι το Mopidy τρέχει και ότι δεν υπάρχουν σφάλματα :

```
sudo systemctl status mopidy
```

Βήμα 4. Ελέγχουμε αν η πόρτα του Mopidy ακούει (εξ ορισμού είναι η πόρτα 6680)

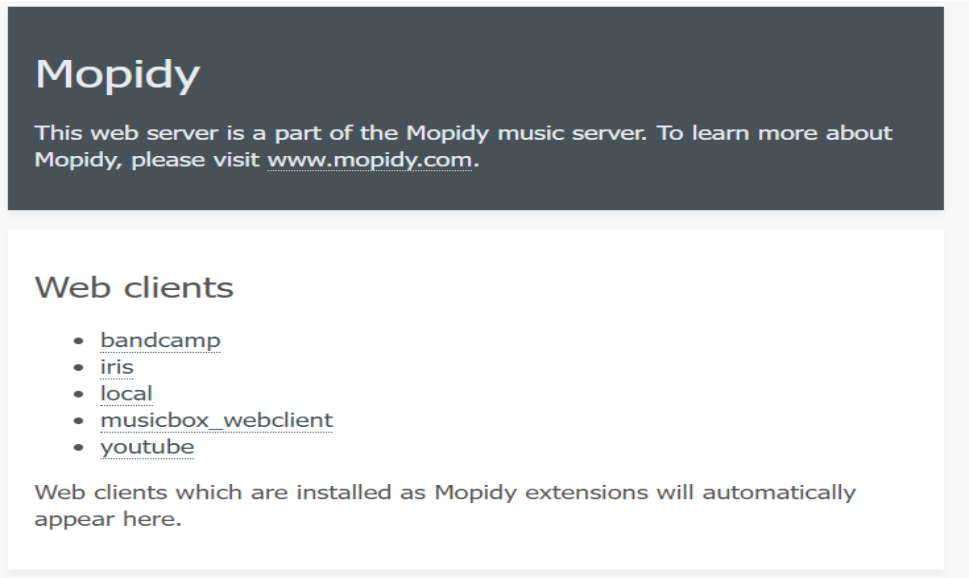
```
sudo netstat -tulpn | grep LISTEN
```

3.2 Αναπαραγωγή μουσικής με τη χρήση των client

Αφού όλα πάνε καλά, από ένα έξυπνο κινητό τηλέφωνο (smartphone), ταμπλέτα ή ηλεκτρονικό υπολογιστή, συνδεόμαστε στο Mopidy server. Ανοίγουμε ένα πρόγραμμα περιήγησης, και στη γραμμή διευθύνσεων δίνουμε την παρακάτω διεύθυνση :

http://"διεύθυνση IP του Raspberry Pi":6680

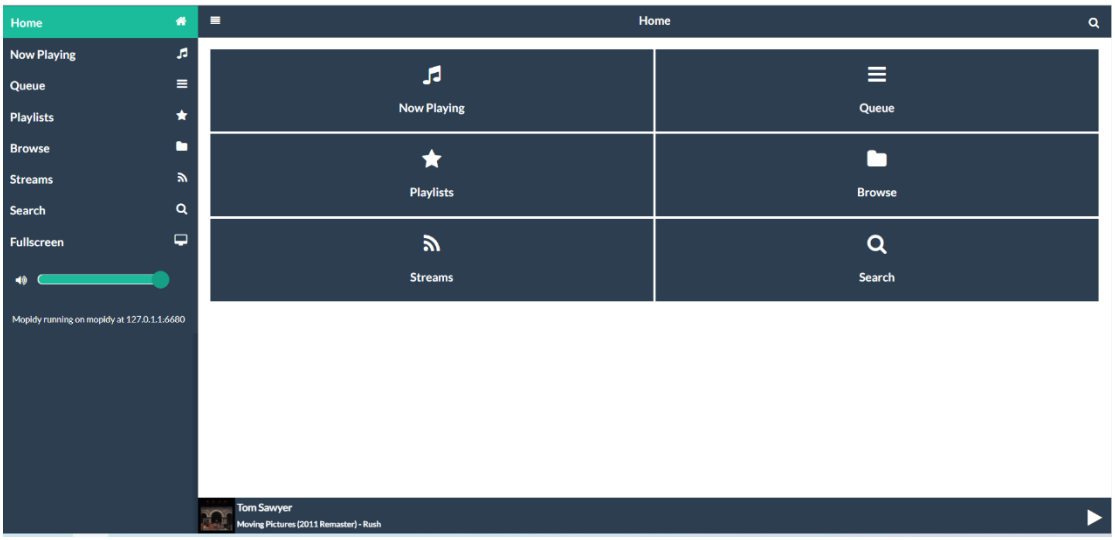
Εμφανίζεται η αρχική σελίδα του Mopidy server και μετά επιλέγουμε τον επιθυμητό webclient.



Εικόνα 44: Αρχική σελίδα Mopidy

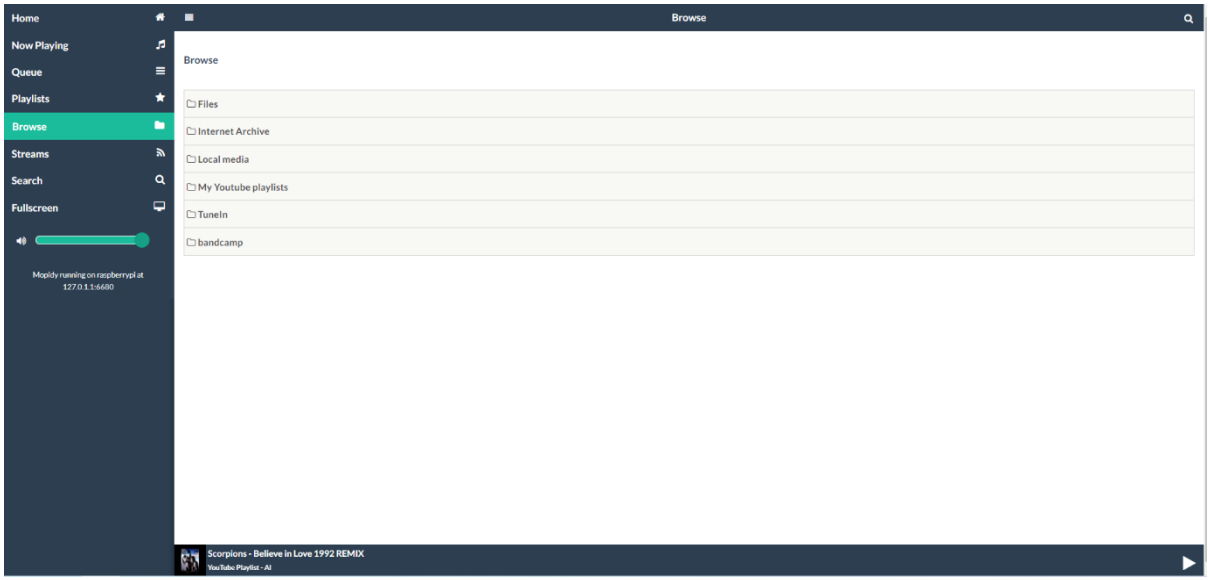
3.2.1 Αναπαραγωγή μουσικής με τον client musicbox

Μεταφερόμαστε στην αρχική σελίδα του musicbox web client και από την λίστα επιλογών στα αριστερά επιλέγουμε την επιθυμητή ενέργεια.



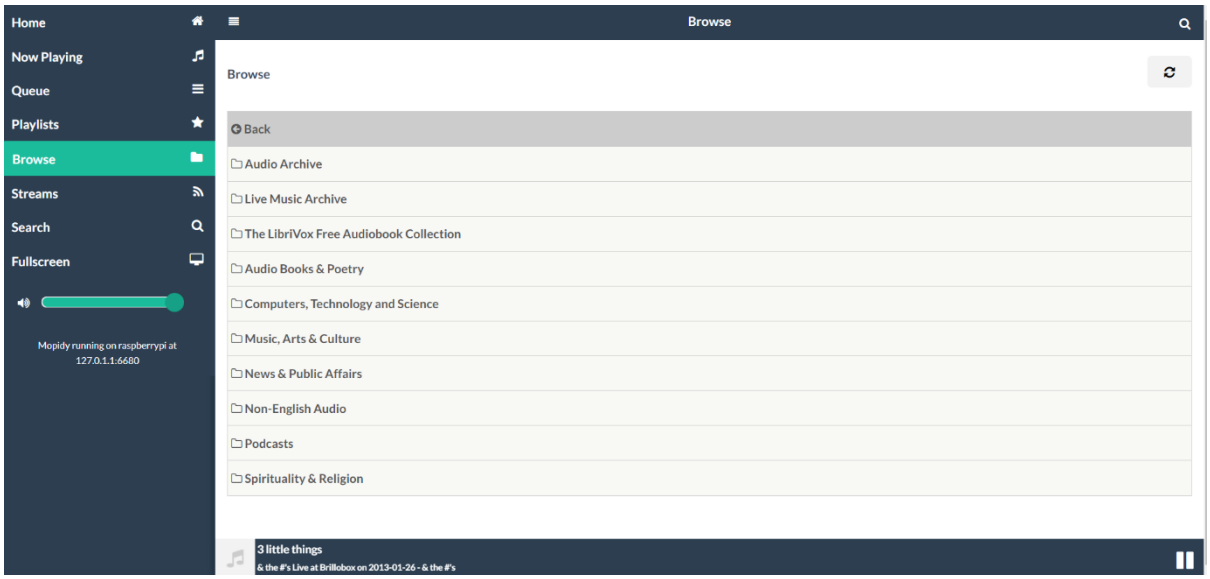
Εικόνα 45: Αρχική σελίδα MusicBox Web Client

Για παράδειγμα επιλέγοντας το μενού Browse, μας δίνεται η δυνατότητα να αναπαράγουμε μουσική από αφαιρούμενα μέσα USB (Local media) και από τους απομακρυσμένους εξυπηρετητές. Ανάλογα με τις επεκτάσεις του Mopidy όπου έχουμε εγκαταστήσει, εμφανίζονται και οι ανάλογες επιλογές.



Εικόνα 46: MusicBox Καρτέλα Περιήγηση Browse

Επιλέγοντας την επιλογή Internet Archive, μπορούμε να περιηγηθούμε στο Internet Archive, και να αναπαράγουμε αρχεία ήχου που είναι αποθηκευμένα σε αυτό και είναι ταξινομημένα με συγκεκριμένα κριτήρια.

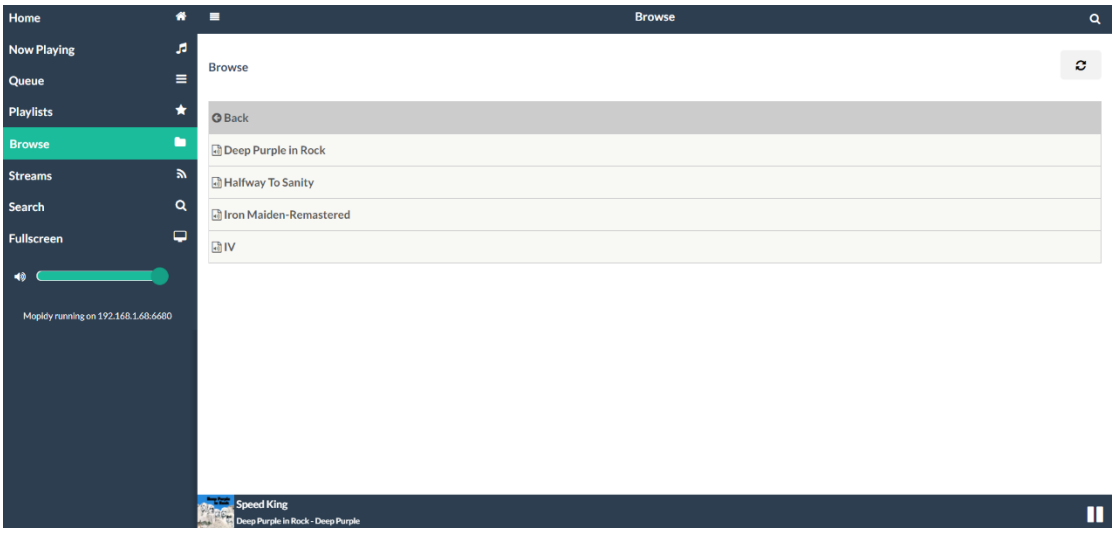


Εικόνα 47: MusicBox Περιήγηση στο Internet Archive

Εάν επιλέξουμε Local media, έχουμε τη δυνατότητα να αναπαράγουμε μουσική από αποθηκευτικά μέσα όπως αφαιρούμενους σκληρούς δίσκους ή USB. Να σημειωθεί πως όταν εισάγουμε άλλο αποθηκευτικό μέσο USB, για να ανανεωθεί η λίστα των μουσικών κομματιών και να είναι ορατά στον web client, θα πρέπει να τρέχουμε την εντολή :

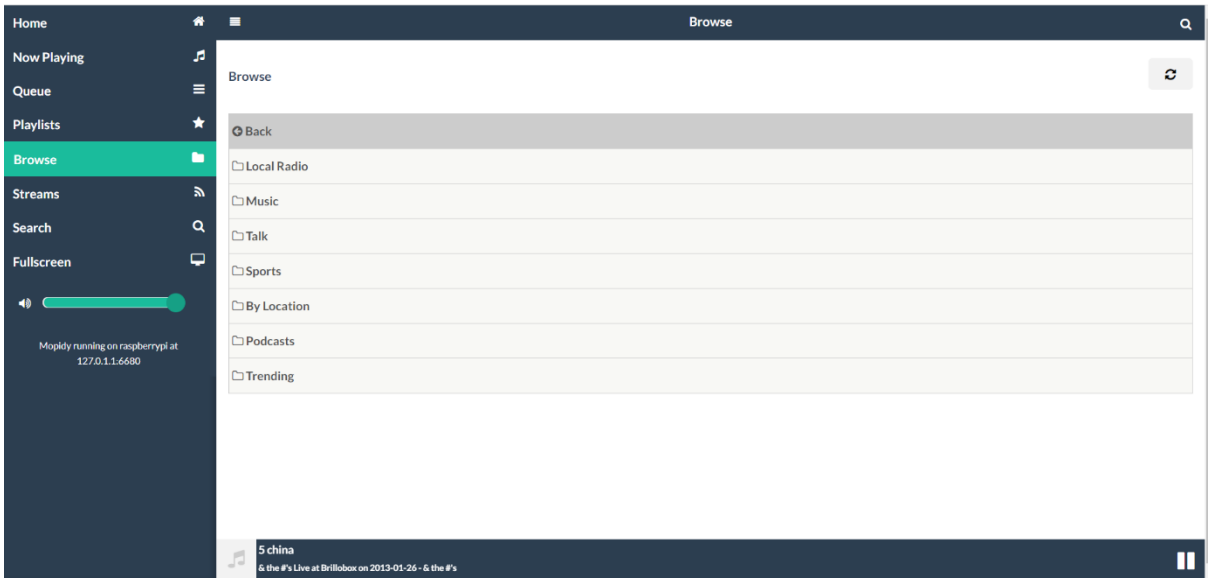
sudo mopidyctl local scan

ή επανεκκινούμε το σύστημά μας.



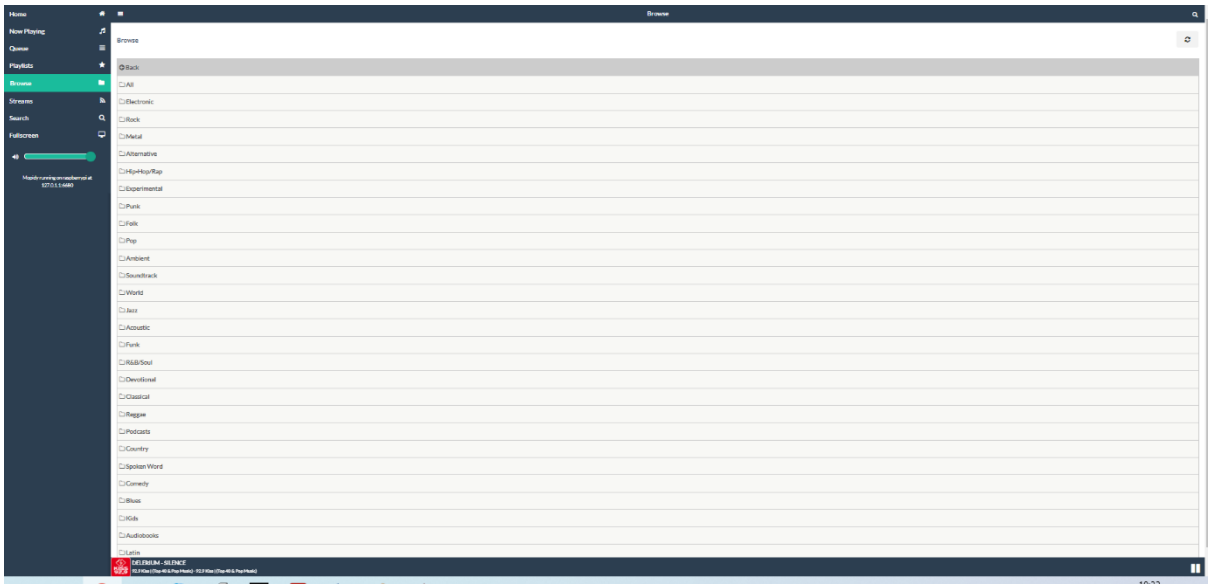
Εικόνα 48: MusicBox Περιήγηση στα τοπικά αρχεία Local

Αν επιλέξουμε TuneIn, μπορούμε να επιλέξουμε τα κορυφαία τραγούδια, μπορούμε να επιλέξουμε ραδιοφωνικούς σταθμούς σύμφωνα με τα κριτήρια που μας δίνονται πχ τοπικούς σταθμούς (local Radio), σταθμούς που παίζουν συγκεκριμένα είδη μουσικής (Music), talk shows (Talk), σταθμούς με αθλητικό περιεχόμενο (Sport), σταθμούς από περιοχή της επιλογής μας (By Location), Podcasts και τέλος δημοφιλείς σταθμούς (Trending).



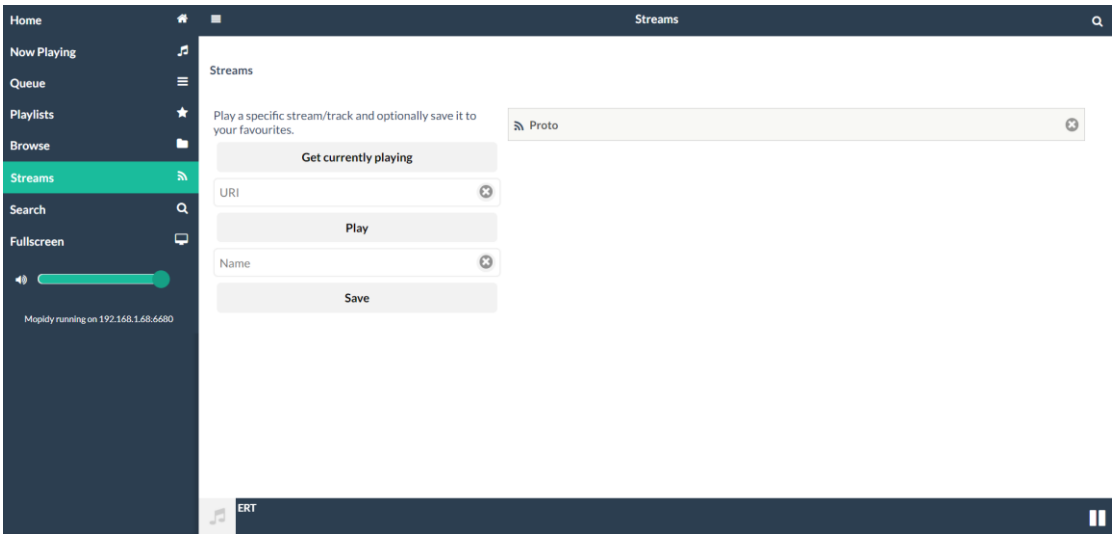
Εικόνα 49: MusicBox Περιήγηση στους ραδιοφωνικούς σταθμούς TuneIn

Επιλέγοντας Bandcamp, μπορούμε να περιηγηθούμε στα περιεχόμενα του Bandcamp και να επιλέξουμε τη μουσική που επιθυμούμε, ανά είδος μουσικής.



Εικόνα 50: MusicBox Περιήγηση στο Bandcamp

Επιλέγοντας Streams, μπορούμε να αναζητήσουμε ραδιοφωνικούς σταθμούς που μεταδίδονται διαδικτυακά με βάση το URI τους, καθώς επίσης να τους δώσουμε ένα όνομα και να τους αποθηκεύσουμε, δημιουργώντας λίστα αγαπημένων σταθμών.



Εικόνα 51: MusicBox Αναπαραγωγή διαδικτυακών ροών Streams

Υπάρχει η δυνατότητα δημιουργίας και επεξεργασίας playlist στο φάκελο /var/lib/mopidy/m3u/όνομα playlist.m3u .

Το περιεχόμενο των αρχείων playlist θα πρέπει να είναι της μορφής :

#EXTM3U

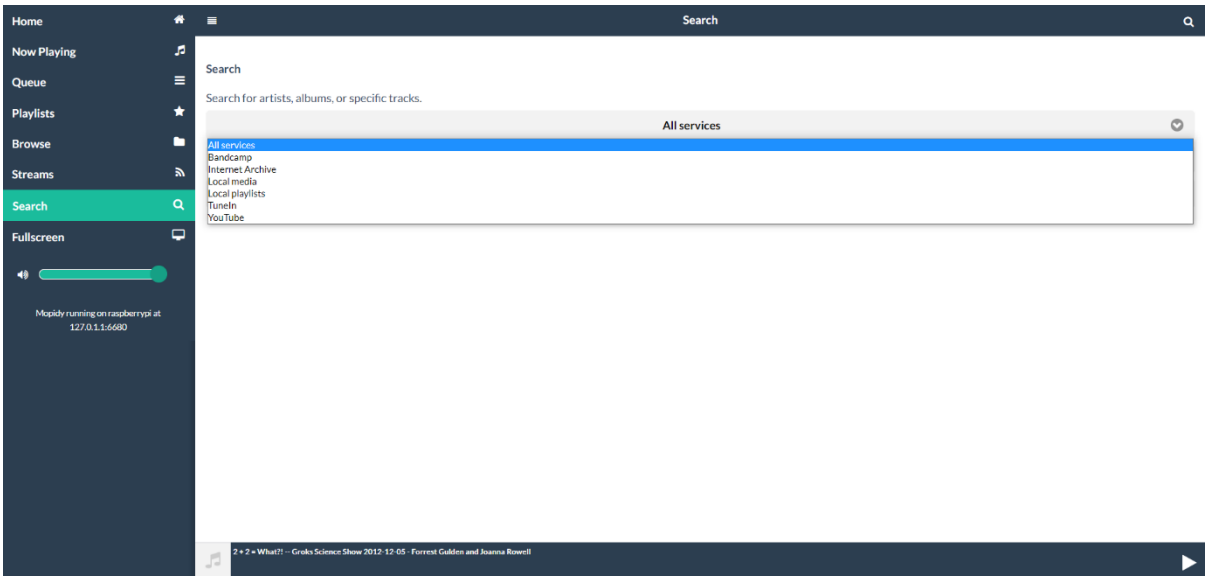
#EXTINF:0, Όνομα του σταθμού

http://url του σταθμού

Αφού αποθηκεύσουμε τις αλλαγές στο αρχείο της playlist, επανεκκινούμε το σύστημα ώστε να γίνει εφαρμογή των αλλαγών.

Οι νέες playlists θα είναι πλέον ορατές πατώντας στην επιλογή Playlists.

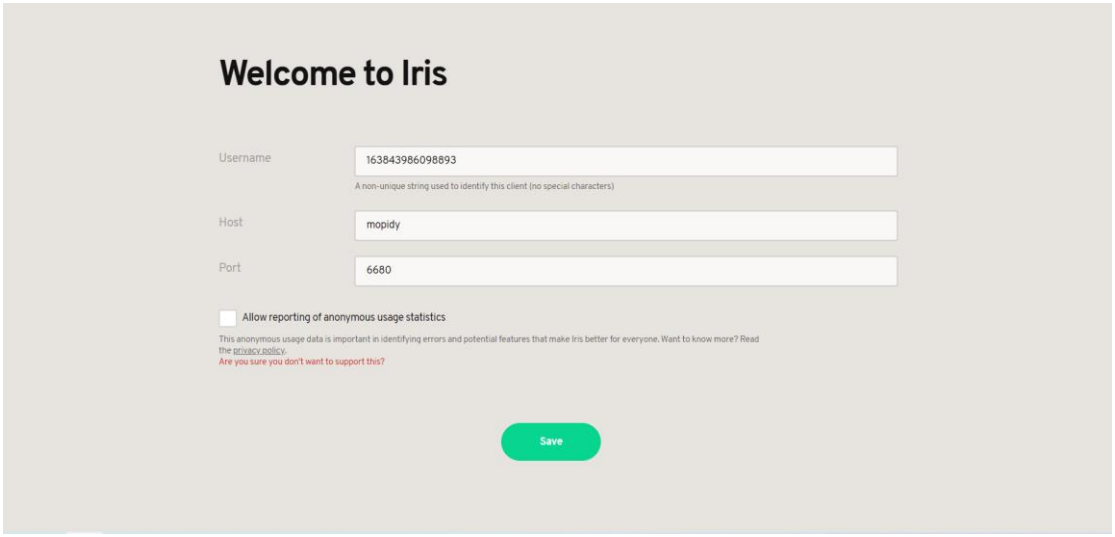
Με την επιλογή Search μπορούμε να αναζητήσουμε μουσικά κομμάτια ανά υπηρεσία ή συνδυάζοντας όλες τις υπηρεσίες, συμπεριλαμβανομένου και του YouTube.



Εικόνα 52: MusicBox Αναζήτηση μουσικών κομματιών

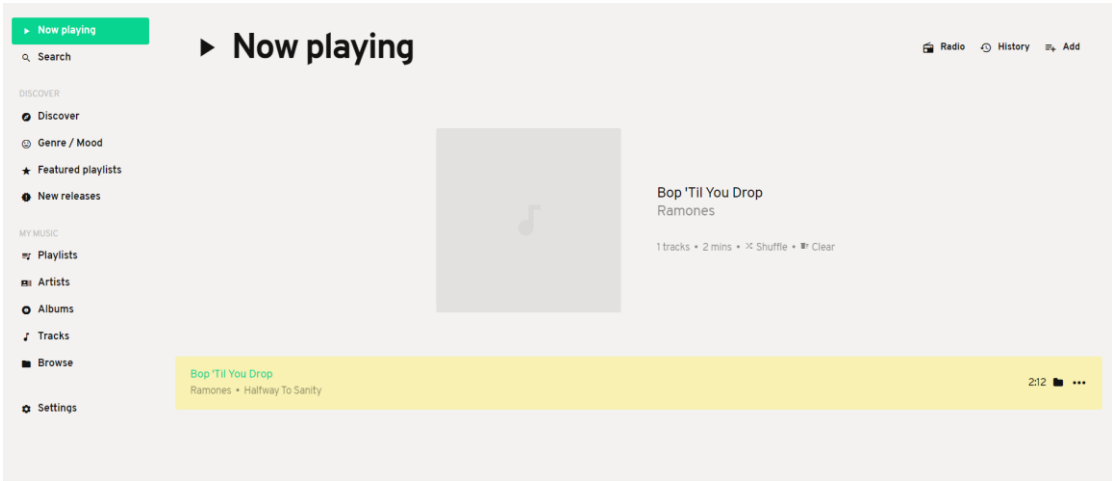
3.2.2 Αναπαραγωγή μουσικής με τον client Iris

Εάν επιλέξουμε ως client τον Iris, αρχικά θα εμφανιστεί μια σελίδα καλωσορίσματος και για να προχωρήσουμε θα πρέπει να πατήσουμε Save. Δεν χρειάζεται να κάνουμε κάποια αλλαγή στα στοιχεία της σελίδας αυτής.



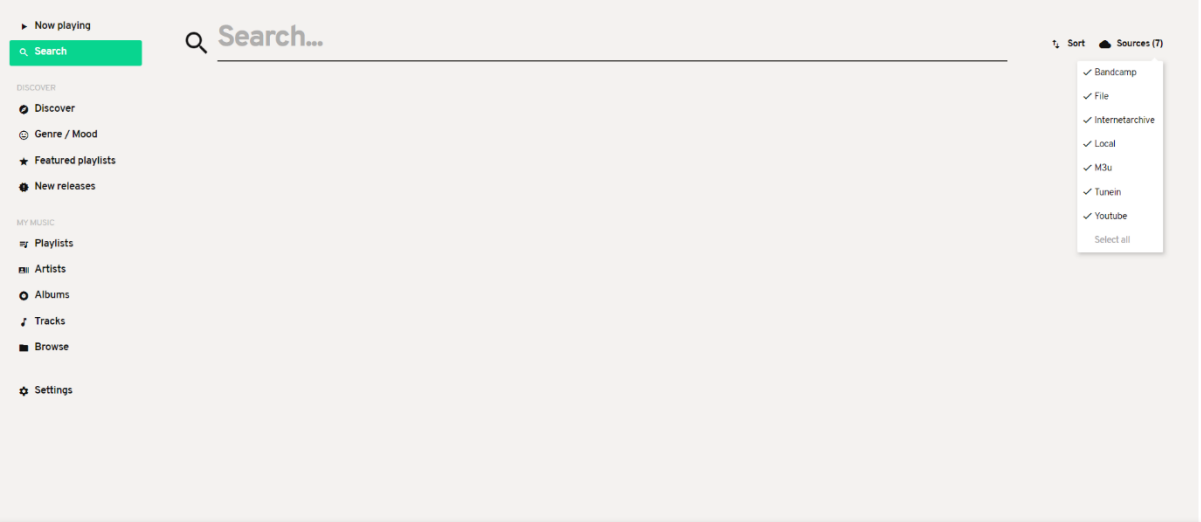
Εικόνα 53: Αρχική σελίδα Iris

Αφού πατήσουμε Save, οδηγούμαστε στη σελίδα Now Playing. Στη σελίδα αυτή βλέπουμε τί παίζει την τρέχουσα στιγμή.



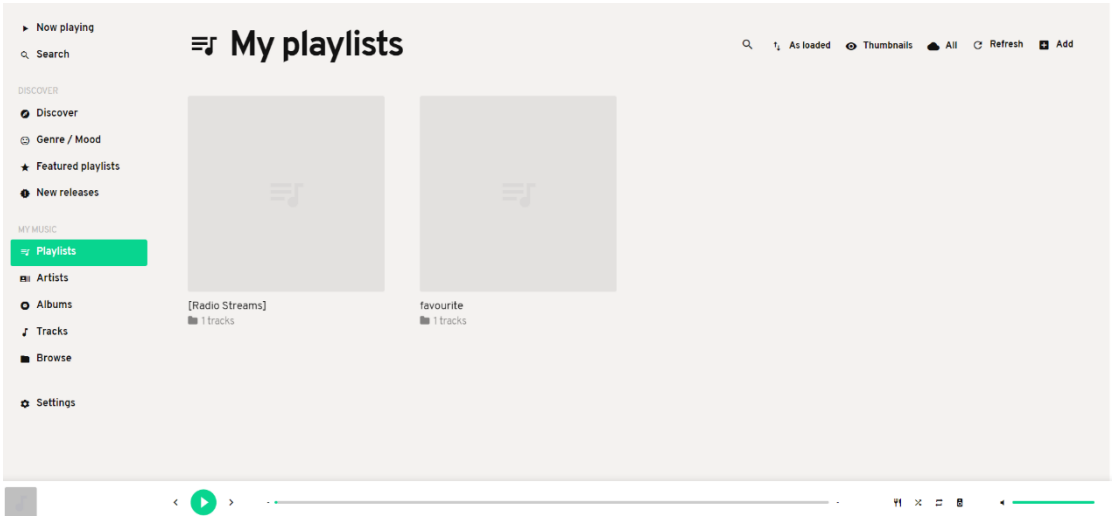
Εικόνα 54: Iris Καρτέλα τρέχουσας αναπαραγωγής

Επιλέγοντας Search μπορούμε να αναζητήσουμε μουσικό περιεχόμενο από όλες τις διαθέσιμες επιλογές όπως μουσική από το Bandcamp, μουσική αποθηκευμένη στο σύστημά μας (File), από το Internet Archive, μουσική αποθηκευμένη σε εξωτερικά μέσα αποθήκευσης (Local), αποθηκευμένες Playlists (M3u), TuneIn ή Youtube ή από επιμέρους πηγές .



Εικόνα 55: Iris Αναζήτηση μουσικών κομματιών

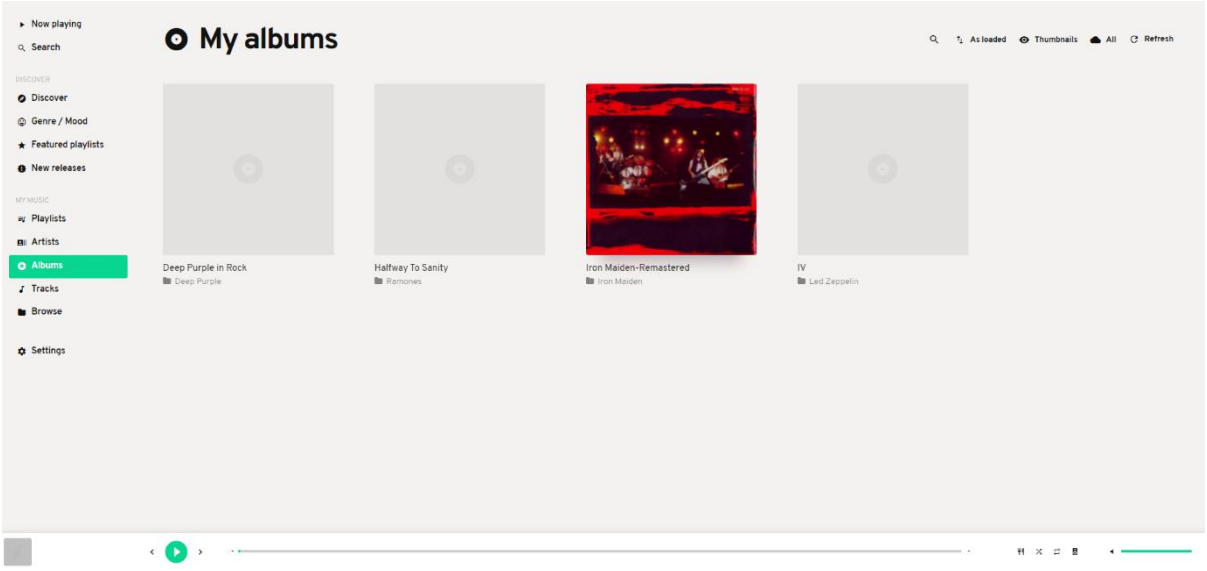
Στην ενότητα MY MUSIC και συγκεκριμένα στην επιλογή Playlists μπορούμε να δούμε τις αποθηκευμένες μας λίστες αναπαραγωγής.



Εικόνα 56: Iris Αποθηκευμένες Playlists

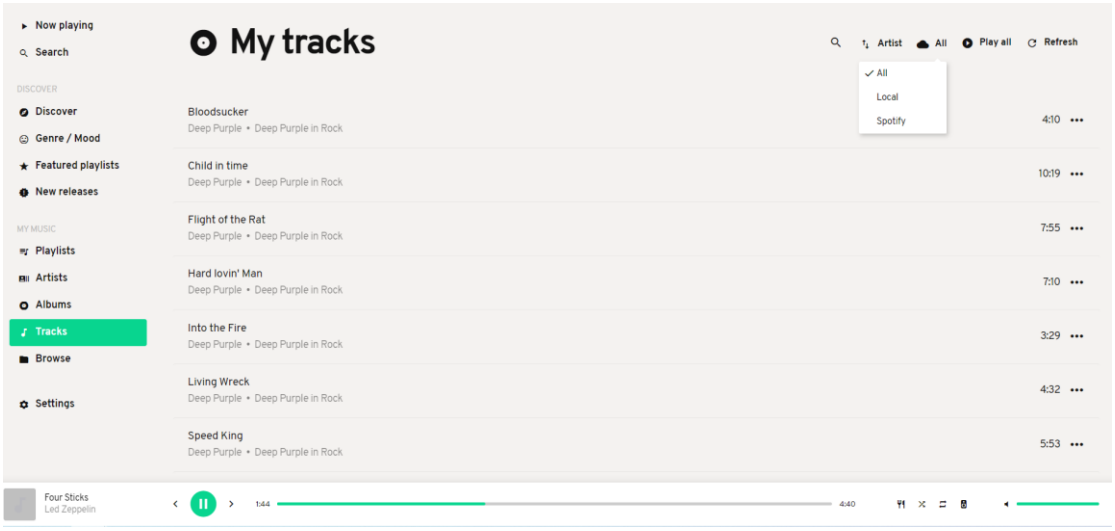
Μετά από κάθε αλλαγή όσον αφορά τις playlists, πατάμε refresh για να ανανεωθεί η λίστα.

Στην επιλογή Albums βλέπουμε τα αποθηκευμένα μας albums στα αφαιρούμενα αποθηκευτικά μέσα.



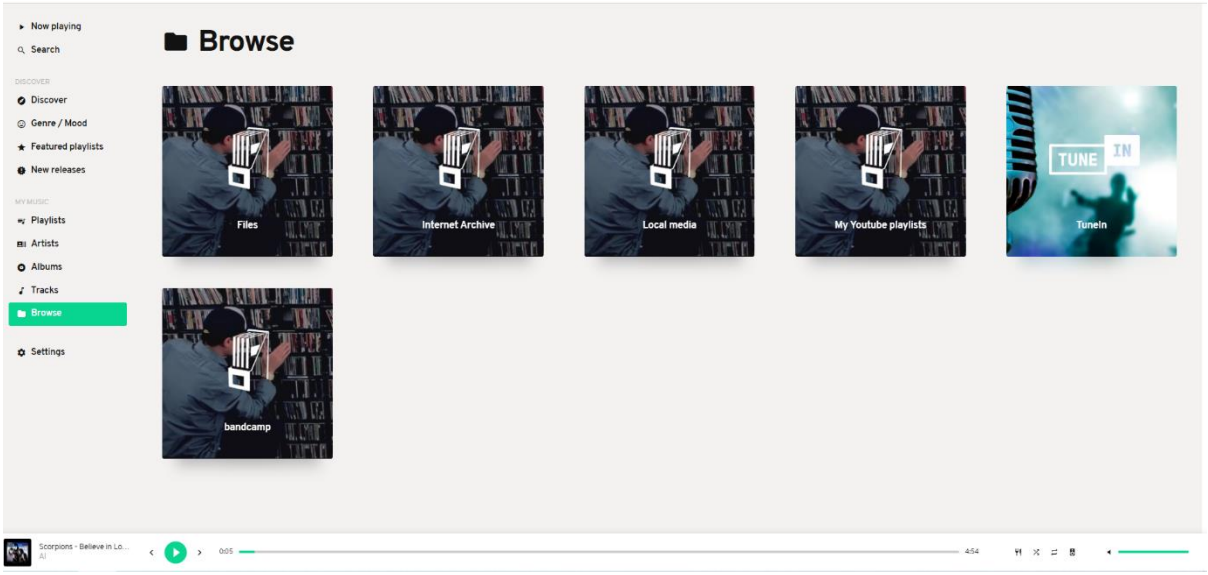
Εικόνα 57: Iris Αποθηκευμένα Albums

Στην επιλογή Tracks βλέπουμε τα τραγούδια.



Εικόνα 58: Iris Αποθηκευμένα τραγούδια

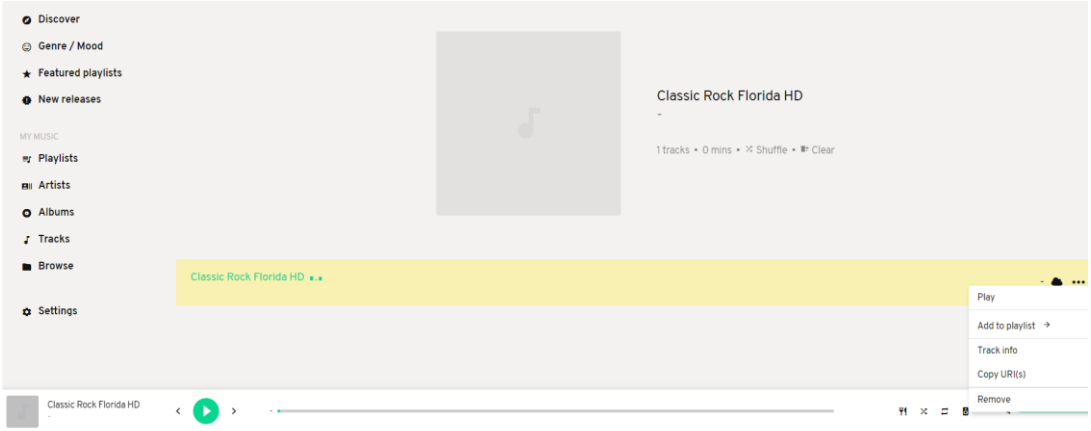
Στην επιλογή Browse μας δίνεται η δυνατότητα να περιηγηθούμε στις επιμέρους μουσικές μας πηγές, ώστε να αναπαράγουμε τη μουσική που επιθυμούμε.



Εικόνα 59: Iris Καρτέλα Περιήγηση Browse

Εάν επιλέξουμε την επιλογή Add πάνω δεξιά, μπορούμε να αναζητήσουμε μουσική με βάση το URI, πχ ένα ραδιοφωνικό σταθμό και να τον προσθέσουμε στην ουρά αναπαραγωγής. Έπειτα εάν επιθυμούμε μπορούμε πατώντας στις τρεις τελείες στα δεξιά

να αποθηκεύσουμε σε μια υπάρχουσα ή σε μια καινούρια λίστα αναπαραγωγής το συγκεκριμένο ραδιοφωνικό σταθμό με την επιλογή Add to playlist.

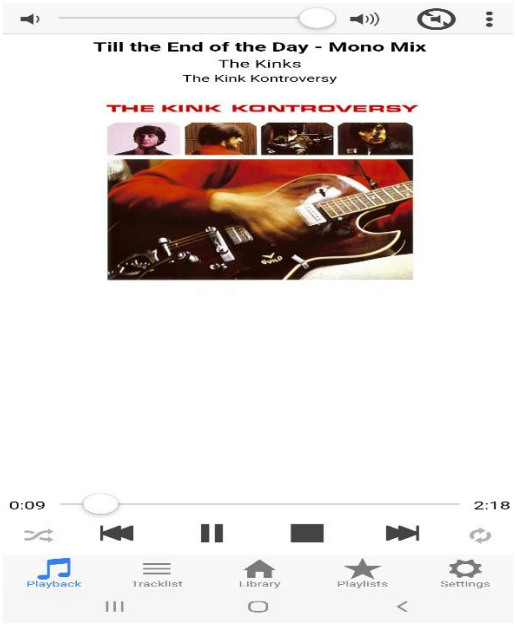


Εικόνα 60: Iris Προσθήκη ραδιοφωνικού σταθμού

3.2.3 Αναπαραγωγή μουσικής με τον client Mopidy-mobile

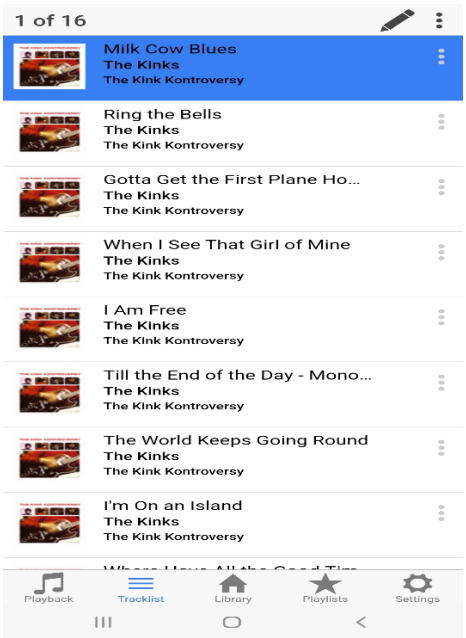
Εκκινώντας την εφαρμογή ερχόμαστε σε επαφή με ένα λιτό αλλά πλήρως λειτουργικό περιβάλλον το οποίο μας επιτρέπει να εκτελέσουμε όλες τις απαραίτητες λειτουργίες του Mopidy.

Η πρώτη καρτέλα είναι αυτή της τρέχουσας αναπαραγωγής (Playing). Μπορούμε να εκτελέσουμε όλες τις βασικές λειτουργίες αναπαραγωγής, να βάλουμε το σύστημα σε κατάσταση σίγασης, να αλλάξουμε τη στάθμη αναπαραγωγής ήχου κλπ.



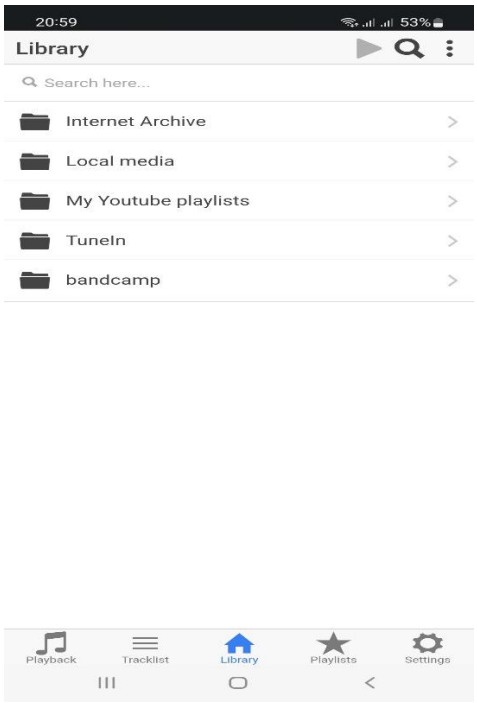
Εικόνα 61: Mopidy-Mobile Καρτέλα τρέχουσας αναπαραγωγής.

Η επόμενη καρτέλα είναι αυτή της λίστας τραγουδιών προς αναπαραγωγή. Μπορούμε να προσθέσουμε ένα ή περισσότερα κομμάτια και να εκτελεστούν με τη σειρά ή με τυχαία σειρά.

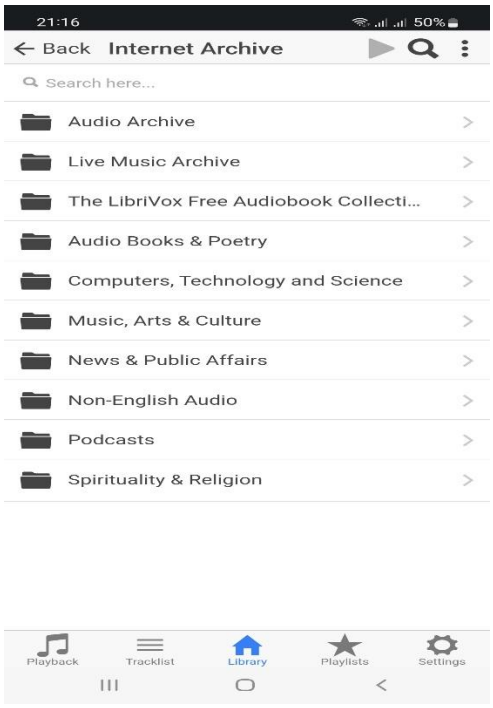


Εικόνα 62: Mopidy-Mobile Καρτέλα Tracklist

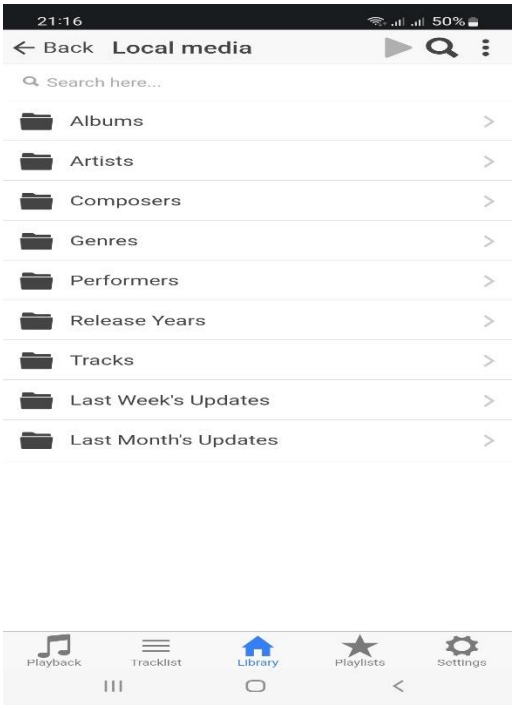
Στην καρτέλα Library μπορούμε να επιλέξουμε μουσική από τις διάφορες διαθέσιμες πηγές των οποίων τις επεκτάσεις έχουμε εγκαταστήσει. Με τη χρήση διαφόρων κριτηρίων μπορούμε να ανακαλύψουμε νέα μουσική ή να κάνουμε αναζήτηση σε μία ή περισσότερες πηγές.



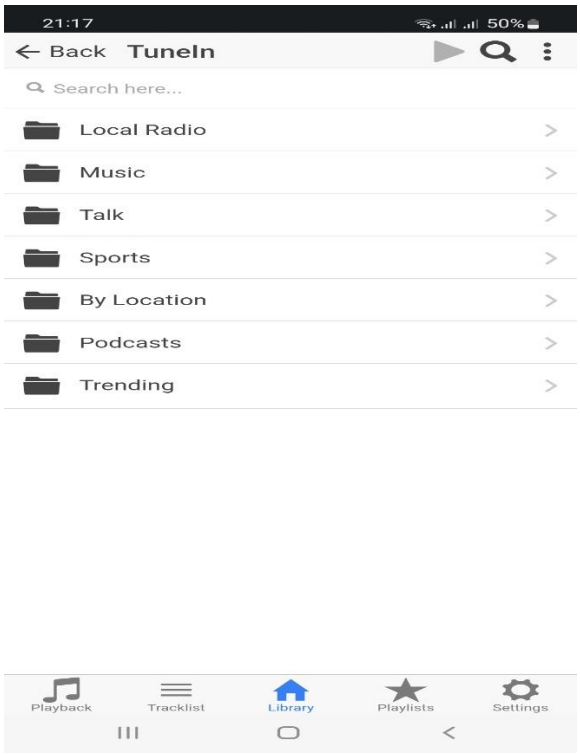
Εικόνα 63: Mopidy-Mobile Καρτέλα Library



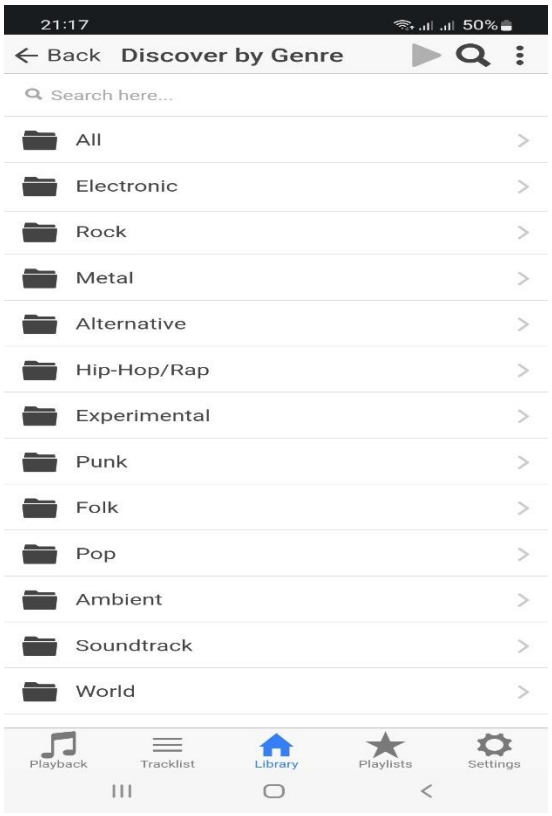
Εικόνα 64: Mopidy-Mobile Καρτέλα Internet Archive



Εικόνα 65: Mopidy-Mobile Καρτέλα Local Media

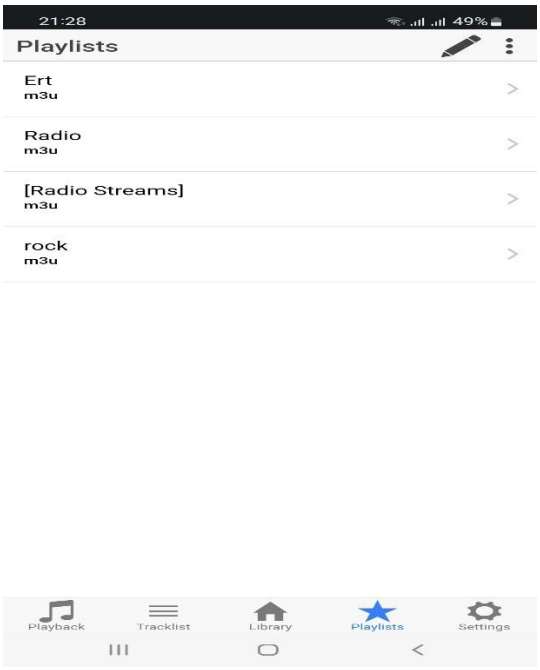


Εικόνα 66: Mopidy-Mobile Καρτέλα TuneIn



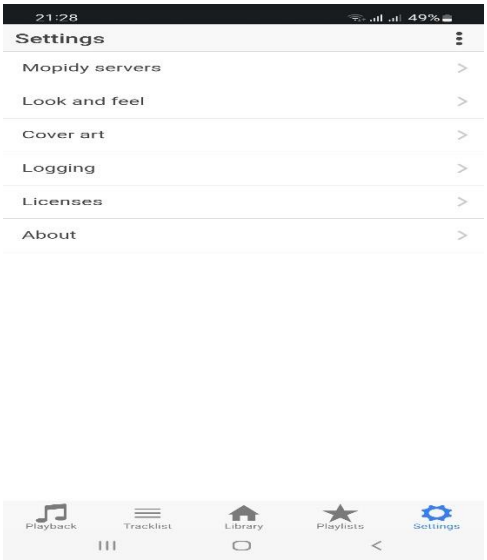
Εικόνα 67: Mopidy-Mobile Καρτέλα Bandcamp

Επόμενη καρτέλα είναι αυτή που περιλαμβάνει τις λίστες με τις επιλογές μας (playlists)



Εικόνα 68: Mopidy-Mobile Καρτέλα με Playlists

Τέλος βλέπουμε την καρτέλα ρυθμίσεων όπου μας δίνεται η δυνατότητα να προβούμε σε ρυθμίσεις του client όπως τη γλώσσα και την εμφάνιση.



Εικόνα 69: Mopidy-Mobile Καρτέλα Ρυθμίσεων

3.3 Επεξήγηση του αρχείου ρυθμίσεων mopidy.conf

Το αρχείο ρυθμίσεων mopidy.conf βρίσκεται στον φάκελο /etc/mopidy στην περίπτωσή που τρέχουμε το Mopidy server σαν service. Για να κάνουμε τροποποιήσεις, ανοίγουμε το αρχείο αυτό με ένα κειμενογράφο όπως το nano ή το vim, και προσθέτουμε τις αλλαγές. Π.χ.

sudo nano /etc/mopidy/mopidy.conf

και για να δούμε το αποτέλεσμα των αλλαγών τρέχουμε :

sudo mopidyctl config

Αυτή η εντολή θα μας δώσει στην οθόνη το αποτέλεσμα των ρυθμίσεων που κάναμε, με τους κωδικούς που τυχόν έχουμε εισάγει καλυμμένους, για λόγους ασφαλείας. Ας δούμε αναλυτικά τις επιμέρους ρυθμίσεις του αρχείου ρυθμίσεων. Παρακάτω παρατίθενται οι προκαθορισμένες ρυθμίσεις του Mopidy [122] :

[core]
cache_dir = var/cache/mopidy
config_dir = /etc/mopidy
data_dir = /var/lib/mopidy
max_tracklist_length = 10000
restore_state = false

[logging]
verbosity = 0
format = %(levelname)-8s %(asctime)s [%(process)d:%(threadName)s] %(name)s\n %(message)s
color = true
config_file =

[audio]
mixer = software
mixer_volume =

output = autoaudiosink
buffer_time =

[proxy]
scheme =
hostname =
port =
username =
password =

Ρυθμίσεις του Core

core/cache_dir

Διαδρομή του βασικού directory για να αποθηκεύονται τα αποθηκευμένα αρχεία (cached data).

Το Mopidy και οι επεκτάσεις του θα χρησιμοποιούν αυτή τη διαδρομή για να αποθηκεύουν δεδομένα τα οποία θα μπορούν να απορριφθούν με ασφάλεια.

core/config_dir

Η διαδρομή που οδηγεί στο αρχείο ρυθμίσεων mopidy.conf

Επειδή μας ενδιαφέρει η λειτουργία σαν service, η διαδρομή είναι η /etc/mopidy.

core/data_dir

Η διαδρομή όπου βρίσκονται τα μόνιμα αρχεία δεδομένων (persistent data). Εδώ αποθηκεύονται δεδομένα που είναι απαραίτητα για το σύστημα μας και η πρόσβαση σε αυτά είναι σπάνια και έχουν μικρή πιθανότητα να χρειαστούν τροποποίηση.

core/max_tracklist_length

Μέγιστο μήκος λίστας αναπαραγωγής. Προεπιλεγμένη τιμή είναι 10000.

core/restore_state

Όταν αυτή η επιλογή είναι ενεργοποιημένη, το Mopidy κατά την εκκίνηση επανακτά την τελευταία του κατάσταση. Αυτή περιλαμβάνει την λίστα αναπαραγωγής, το ιστορικό αναπαραγωγής, την κατάσταση αναπαραγωγής, την στάθμη του ήχου και την κατάσταση σίγασης. Προεπιλεγμένα είναι απενεργοποιημένη.

Ρυθμίσεις Audio

audio/mixer

Ο μίκτης ήχου που θα χρησιμοποιηθεί. Ο προκαθορισμένος είναι 'software', ο οποίος ελέγχει τη στάθμη του ήχου μέσα στο Mopidy, πριν ο ήχος σταλεί στην έξοδο ήχου. Ο μίκτης αυτός δεν επηρεάζει τη στάθμη ήχου άλλων προγραμμάτων του συστήματος, τα οποία κάνουν αναπαραγωγή ήχου. Σε περίπτωση που επιθυμούμε να απενεργοποιήσουμε την μίξη ήχου, δίνουμε την τιμή 'none'.

Αν θέλουμε να χρησιμοποιήσουμε hardware μίκτη, θα πρέπει να εγκαταστήσουμε την κατάλληλη επέκταση Mopidy, πχ για **ALSA**, εγκαθιστούμε την επέκταση **Mopidy-ALSAMixer**.

audio/mixer_volume

Αρχική τιμή στάθμης ήχου του μίκτη ήχου. Οι τιμές ποικίλουν από 0 έως 100. Εάν αφήσουμε την τιμή κενή, η τιμή στάθμης ήχου του μίκτη παραμένει αμετάβλητη, και συγκεκριμένα στο 100.

audio/output

Επιλέγουμε ποια έξοδο ήχου θα χρησιμοποιήσουμε. Τυπικές τιμές είναι διάφορα GStreamer sinks, όπως autoaudiosink, alsasink, osssink, oss4sink, pulsesink και shout2send, καθώς και πρόσθετα ορίσματα κατάλληλα για κάθε sink.

audio/buffer_time

Το μέγεθος ενδιάμεσης μνήμη (buffer) σε milliseconds. Παίρνει ακέραιες τιμές πάνω από το 0. Αν εμφανίζεται buffering πριν την αλλαγή των κομματιών, καλό είναι να αυξήσουμε την τιμή αυτή κατά κάποια δευτερόλεπτα. Η default τιμή είναι το 1000.

Ρυθμίσεις Logging

logging/verbosity

Καθορίζει το πόσες λεπτομέρειες απεικονίζονται κατά την εκτέλεση της εντολής mopidy, η οποία εκκινεί το Mopidy server. Οι τιμές που μπορεί να λάβει είναι από το -1 έως το 4. Όσο μεγαλύτερη είναι η τιμή, τόσο περισσότερες λεπτομέρειες θα βλέπουμε στην οθόνη. Η προκαθορισμένη τιμή είναι το 0,.

logging/color

Αν θα χρωματίζει ή όχι την console log ανάλογα με το log level. Προεπιλεγμένη τιμή είναι ‘true’.

logging/format

Η μορφή μηνύματος που χρησιμοποιείται στο logging.

logging/config_file

Αρχείο ρυθμίσεων το οποίο παρακάμπτει όλες τις τιμές ρυθμίσεων του logging.

Ρυθμίσεις Proxy

Ορισμένες επεκτάσεις του Mopidy μπορούν να χρησιμοποιήσουν proxy server, για αναπαραγωγή μουσικής μέσω διαδικτύου, για παράδειγμα ραδιοφωνικά streams.

proxy/scheme

Uniform Resource Identifier (URI) του proxy server. Συνήθεις τιμές είναι http, https, socks4, ή socks5.

proxy/hostname

Το όνομα του κεντρικού υπολογιστή (hostname) του proxy server.

proxy/port

Ο αριθμός της πόρτας του proxy server.

proxy/username

Το όνομα του χρήστη (username) του proxy server, εάν ζητείται.

proxy/password

Ο κωδικός πρόσβασης (password) για τον proxy server, εάν ζητείται.

Ρυθμίσεις επεκτάσεων (Extension)

Clients

Στην υλοποίησή μας χρησιμοποιούμε τους web clients MusicBox [127] και Iris [139], καθώς και την εφαρμογή Mopidy-Mobile η οποία τρέχει σε έξυπνα κινητά τηλέφωνα με λειτουργικό Android .

MusicBox

[musicbox_webclient]
enabled = true
musicbox = false
websocket_host =

websocket_port =
on_track_click = PLAY_ALL

musicbox_webclient/enabled

Ενεργοποιούμε ή απενεργοποιούμε τον client με τις τιμές true και false αντίστοιχα.

musicbox_webclient/musicbox
musicbox_webclient/websocket_host
musicbox_webclient/websocket_port

Τις παραπάνω ρυθμίσεις τις αφήνουμε ως έχουν διότι είναι προαιρετικές.

musicbox_webclient/ on_track_click

Ορίζουμε τη λειτουργία που θα εκτελεστεί όταν επιλέγουμε ένα κομμάτι. Η εξ ορισμού επιλογή είναι η PLAY_ALL και άλλες επιλογές είναι η PLAY_NOW, PLAY_NEXT, ADD_THIS_BOTTOM, ADD_ALL_BOTTOM και DYNAMIC όπου επαναλαμβάνει την τελευταία λειτουργία.

Iris

[iris]
enabled = true
country = NZ
locale = en_NZ
verify_certificates = true
snapcast_enabled = true
snapcast_host = localhost
snapcast_port = 1780
snapcast_ssl = false
snapcast_stream = Default
spotify_authorization_url = https://jamesbarnsley.co.nz/iris/auth_spotify.php
lastfm_authorization_url = https://jamesbarnsley.co.nz/iris/auth_lastfm.php
genius_authorization_url = https://jamesbarnsley.co.nz/iris/auth_genius.php
data_dir = $XDG_DATA_DIR/iris

Ο client λειτουργεί απρόσκοπτα με τις προκαθορισμένες ρυθμίσεις και δεν χρειάζεται να κάνουμε αλλαγές. Θα αγνοήσουμε τις ρυθμίσεις που σχετίζονται με την αναπαραγωγή του Spotify, του Snapcast, του Last FM και του Genius γιατί δεν χρησιμοποιούμε αυτές τις πηγές στην υλοποίησή μας οπότε μένουν οι παρακάτω ρυθμίσεις :

iris/enabled

ενεργοποιεί και απενεργοποιεί τον client Iris.

iris/ data_dir

Ορίζουμε το path για τα δεδομένα που σχετίζονται με το Iris.

Επέκταση MOPIDY-FILE

Αυτή η επέκταση [122] χρησιμεύει για αναπαραγωγή μουσικών κομματιών αποθηκευμένα σε τοπικούς φακέλους και είναι ενεργοποιημένη κατά την εγκατάστασή της και επιτρέπει να περιηγούμαστε μέσα στους φακέλους του συστήματός μας ή σε αφαιρούμενα αποθηκευτικά μέσα. Δεν παρέχει λειτουργία αναζήτησης (search).

Σε περίπτωση που θέλουμε να δηλώσουμε να διαβάζει το φάκελο /home/pi/Music, θα πρέπει να δώσουμε πλήρη δικαιώματα χρήστη με την εντολή chmod 777 /home/pi/Music ή σε όποιο άλλο φάκελο θέλουμε.

```
[file]
enabled = true
media_dirs =
$XDG_MUSIC_DIR|Music
~/|Home
show_dotfiles = false
excluded_file_extensions =
.directory
.html
.jpeg
.jpg
```

.log
.nfo
.pdf
.png
.txt
.zip
follow_symlinks = false
metadata_timeout = 1000

file/enabled

Ενεργοποιούμε ή απενεργοποιούμε την επέκταση.

file/media_dirs

Λίστα των directories στα οποία θα μπορούμε να περιηγηθούμε. Μπορούμε επίσης προσθέσουμε και το χαρακτήρα | ακολουθούμενο από ένα όνομα το οποίο θα εμφανίζεται για αυτό το path.

file/show_dotfiles

Επιλέγουμε εάν θα είναι ορατά τα κρυφά αρχεία και οι φάκελοι που το όνομά τους ξεκινάει με τελεία. Η προεπιλεγμένη τιμή είναι 'false'.

file/excluded_file_extensions

Εδώ αναφέρονται οι τύποι των αρχείων που θα εξαιρούνται όταν θα γίνεται σάρωση του directory που περιέχει τα μουσικά κομμάτια. Οι τιμές θα πρέπει να χωρίζονται με κόμμα, ή να βρίσκονται σε νέα σειρά.

file/follow_symlinks

Εάν θα εκτελούνται συντομεύσεις (symbolic links) οι οποίες βρίσκονται στο φάκελο file/media_dirs. Η προκαθορισμένη τιμή είναι 'false'.

file/metadata_timeout

Μέγιστος χρόνος σε milliseconds που θα σαρώνεται ένα αρχείο, πριν σαρωθεί το επόμενο αρχείο. Μειώνοντας τον χρόνο αυτό γίνεται πιο γρήγορα η σάρωση των αρχείων, αλλά μπορεί να οδηγήσει σε παράλειψη κάποιων από αυτά με αποτέλεσμα να μην είναι ορατά στη λίστα αναπαραγωγής.

MOPIDY-M3U

Η επέκταση Mopidy-M3U [122] χρησιμοποιείται για την ανάγνωση και δημιουργία λιστών αναπαραγωγής M3U. Εγκαθίσταται και ενεργοποιείται αυτόματα κατά την εγκατάσταση του Mopidy. Η επέκταση αυτή χειρίζεται URIs τα οποία ξεκινούν με m3u: .

[m3u]
enabled = true
playlists_dir =
base_dir = $XDG_MUSIC_DIR
default_encoding = latin-1
default_extension = .m3u8

m3u/enabled

Ενεργοποιούμε ή απενεργοποιούμε την προέκταση M3U.

m3u/playlists_dir

Η διαδρομή για το directory με αρχεία M3U. Η προεπιλεγμένη επιλογή είναι απενεργοποιημένη, οπότε σε αυτή την περίπτωση οι λίστες αναπαραγωγής αποθηκεύονται στον φάκελο που περιέχει τα δεδομένα της προέκτασης.

m3u/base_dir

Διαδρομή για το βασικό directory, από όπου θα φορτώνονται αρχεία M3U. Αν δεν καθοριστεί, οι σχετικές διαδρομές αποκτούνται βάσει της τοποθεσίας των M3U αρχείων.

m3u/default_encoding

Η κωδικοποίηση χαρακτήρων (Text encoding) που χρησιμοποιείται στα αρχεία με επέκταση .m3u. Προεπιλεγμένη επιλογή είναι η κωδικοποίηση latin-1. Αρχεία με επέκταση .m3u8 αναμένονται να έχουν κωδικοποίηση UTF-8.

m3u/default_extension

Η προέκταση των αρχείων για τις λίστες αναπαραγωγής M3U οι οποίες δημιουργήθηκαν με τη χρήση του core playlist API. Η default προέκταση είναι η .m3u8.

MOPIDY-STREAM

Η επέκταση Mopidy-Stream μας δίνει τη δυνατότητα να αναπαράγουμε μουσική η οποία μεταδίδεται (streaming) διαδικτυακά και εγκαθίσταται αυτόματα με την εγκατάσταση του Mopidy και είναι ενεργοποιημένη εξ αρχής. Δεν παρέχει βιβλιοθήκες μουσικής ούτε αποθηκεύει λίστες αναπαραγωγής, απλά δέχεται κάθε URI το οποίο του εισάγουμε και ταιριάζει με κάποιο από τα πρωτόκολλα που αναφέρονται στη ρύθμιση stream/protocols. Τότε προσπαθεί να ανακτήσει μεταδεδομένα και να αναπαράγει το URI χρησιμοποιώντας το GStreamer. Επιπρόσθετα η επέκταση αυτή μπορεί να εξάγει streams από μια πληθώρα διαφορετικών τύπων λιστών αναπαραγωγής, το οποίο είναι χρήσιμο διότι πολλοί ραδιοφωνικοί σταθμοί αντί για την διεύθυνση του stream παρέχουν διεύθυνση με λίστα αναπαραγωγής. [122]

[stream]
enabled = true
protocols =
http
https
mms

rtmp
rtmps
rtsp
timeout = 5000
metadata_blacklist =

stream/enabled

Ενεργοποιούμε ή απενεργοποιούμε την προέκταση

stream/protocols

Λίστα URI που επιτρέπεται να αναπαραχθούν. Οι τιμές θα πρέπει να χωρίζονται με κόμμα ή να είναι η κάθε μία σε νέα σειρά.

stream/timeout

Χρονική διάρκεια σε milliseconds κατά την οποία διαρκεί ο έλεγχος για stream metadata.

stream/metadata_blacklist

Λίστα με URI από τα οποία δεν θα ανακτώνται metadata πριν την αναπαραγωγή. Τυπικά αυτή η επιλογή χρειάζεται για την αναπαραγωγή σε περίπτωση που τα URIs παρέχονται από συγκεκριμένους streaming παρόχους.

MOPIDY-HTTP

Το Mopidy-HTTP είναι μια προέκταση που επιτρέπει τον έλεγχο του Mopidy μέσω HTTP και WebSockets, για παράδειγμα μέσω ενός web client. Συμπεριλαμβάνεται στο Mopidy και είναι ενεργοποιημένο εξ αρχής. Όταν είναι ενεργοποιημένο, εκκινεί ένα web server στην πόρτα που ορίζεται στη ρύθμιση http/port. Για λόγους ασφαλείας ο web server είναι εξ ορισμού διαθέσιμος μόνο για το ίδιο το σύστημα (localhost). Για να είναι διαθέσιμος και από άλλους υπολογιστές ή κινητά τηλέφωνα, θα πρέπει να γίνει αλλαγή στην τιμή της ρύθμισης http/hostname [122].

[http]
enabled = true
hostname = ::
port = 6680
zeroconf = Mopidy HTTP server on $hostname
allowed_origins =
csrf_protection = true
default_app = mopidy

http/enabled

Επιλέγουμε αν ή προέκταση HTTP θα είναι ενεργή ή όχι.

http/hostname

Η διεύθυνση του HTTP server, στην οποία θα συνδέεται ο web client.

127.0.0.1 Ακούει μόνο στο IPv4 loopback interface

::1 Ακούει μόνο στο IPv6 loopback interface

0.0.0.0 Ακούει σε όλα τα IPv4 interfaces

:: Ακούει σε όλα τα interfaces, και IPv4 και IPv6

http/port

Σε ποια TCP πόρτα θα ακούει ο HTTP server.

http/zeroconf

Το όνομα του HTTP service όταν τρέχει μέσω του Zeroconf. Στο όνομα μπορούν να χρησιμοποιηθούν οι τιμές $hostname και $port.

Αν το αφήσουμε κενό το όνομα, το Zeroconf για HTTP απενεργοποιείται.

http/allowed_origins

Λίστα με domains που επιτρέπονται να εκτελούν Cross-Origin Resource Sharing (CORS) αιτήματα. Εφαρμόζεται και σε JSON-RPC και WebSocket αιτήματα. Οι τιμές θα πρέπει να είναι της μορφής hostname:port και να διαχωρίζονται με κόμμα ή να είναι σε ξεχωριστή γραμμή. Επιπρόσθετα, η πόρτα δεν πρέπει να καθορίζεται αν είναι η default (80 για http και 443 για https).

Αιτήματα ίδιας προέλευσης πχ αιτήματα από το web server του Mopidy επιτρέπονται πάντα οπότε δεν χρειάζεται να εισάγουμε κάποια τιμή.

http/csrf_protection

Ενεργοποιεί την προστασία του HTTP server έναντι του Cross-Site Request Forgery (CSRF) από αιτήματα JSON-RPC και WebSocket. Αυτή η ρύθμιση θα πρέπει να απενεργοποιείται με μεγάλη προσοχή, λόγω θεμάτων ασφαλείας.

http/default_app

Ανακατευθύνει το root directory του web server σε μια συγκεκριμένη web εφαρμογή αντί για τη default λίστα web εφαρμογών του Mopidy. Η τιμή θα πρέπει να είναι το όνομα που χρησιμοποιείται από την προέκταση όταν δηλώνονται τα http:static ή http:app extension points. Κατά συνθήκη, είναι το όνομα ext_name της επέκτασης, στην περίπτωσή μας είναι ‘Mopidy’.

MOPIDY-SOFTWAREMIXER

Η επέκταση αυτή χρησιμοποιείται για τον έλεγχο της στάθμης του ήχου με λογισμικό και συγκεκριμένα επηρεάζει εσωτερικά το pipeline του GStreamer που χρησιμοποιεί το Mopidy για την αναπαραγωγή του ήχου. Εγκαθίσταται αυτόματα με την εγκατάσταση του Mopidy και είναι εξ αρχής ενεργοποιημένη. Για να χρησιμοποιηθεί ο συγκεκριμένος mixer, θα πρέπει στη ρύθμιση audio/mixer να δηλώσουμε την επιλογή “software”. [122]

[softwaremixer]
enabled = true

Ενεργοποιεί ή απενεργοποιεί την προέκταση.

Mopidy-YouTube

Η επέκταση Mopidy-Youtube [130] μας επιτρέπει να αναπαράγουμε μουσική από το Youtube. Παρακάτω δίνεται επεξήγηση των επιλογών που πρέπει να συμπεριλάβουμε στο αρχείο ρυθμίσεων.

[youtube]
enabled = true
allow_cache =
youtube_api_key =
search_results = 15
playlist_max_videos = 20
api_enabled = false
channel_id =
musicapi_enabled = false
musicapi_cookie =
autoplay_enabled = false
strict_autoplay = false
max_autoplay_length = 600
max_degrees_of_separation = 3
youtube_dl_package = youtube_dlc

youtube/enabled

Ενεργοποιούμε ή απενεργοποιούμε την δυνατότητα Youtube.

youtube/allow_cache

Επιλέγουμε 'true' εάν θέλουμε να αποθηκεύονται αρχεία στην κρυφή μνήμη. Η τοποθεσία αποθήκευσης ορίζεται στη ρύθμιση [core] στην επιλογή cache_dir.

youtube/youtube_api_key

Εάν θέλουμε να χρησιμοποιήσουμε το YouTube API, εισάγουμε το αντίστοιχο κλειδί του Google API.

youtube/api_enabled

Ενεργοποιούμε ή απενεργοποιούμε το Youtube API.

youtube/musicapi_enabled

Εάν θέλουμε το mopidy-youtube να χρησιμοποιεί το YouTube Music αντί του κλασικού YouTube, δίνουμε την επιλογή 'true'. Δεν είναι απαραίτητο για αυτή τη λειτουργία να χρησιμοποιούμε παράλληλα και το youtube api.

youtube/channel_id

Αν θέλουμε να βλέπουμε τις λίστες αναπαραγωγής ενός καναλιού Youtube στη βιβλιοθήκη του Mopidy, εισάγουμε το ID του καναλιού.
Στην περίπτωση που θέλουμε να βλέπουμε τις λίστες αναπαραγωγής του προσωπικού μας καναλιού Youtube στη βιβλιοθήκη του Mopidy, θα πρέπει να εισάγουμε το ID του δικού μας καναλιού, έπειτα να ενεργοποιήσουμε το music api και τέλος να εισάγουμε ένα music cookie.

youtube/musicapi_cookie

Εισάγουμε το music cookie.

youtube/autoplay_enabled

Το mopidy-youtube μπορεί να παίξει αυτόματα παρόμοια τραγούδια όταν ολοκληρωθεί το τελευταίο τραγούδι στην ουρά αναπαραγωγής. Αν το επιθυμούμε, επιλέγουμε 'true'.
Στην περίπτωση αυτή, έχουμε περαιτέρω επιλογές όπως :

youtube/strict_autoplay

όπου σε περίπτωση που δηλώσουμε ‘true’, το σύστημα θα αγνοήσει την τρέχουσα λίστα αναπαραγωγής και θα παίξει το πιο σχετικό τραγούδι.

youtube/max_autoplay_length

μέγιστη διάρκεια τραγουδιού σε δευτερόλεπτα. Default τιμή είναι τα 600 δευτερόλεπτα, αλλιώς το αφήνουμε κενό.

youtube/max_degrees_of_separation

Ελέγχει πόσο σχετικό θα είναι το προτεινόμενο τραγούδι στο αρχικό τραγούδι, στο οποίο γίνεται η πρόταση. Επιλέγοντας τη default τιμή που είναι το 3, το πρώτο προτεινόμενο τραγούδι θα είναι σχετικό με το τραγούδι για το οποίο γίνεται η πρόταση. Το δεύτερο προτεινόμενο τραγούδι θα σχετίζεται περισσότερο με το πρώτο προτεινόμενο τραγούδι, παρά με το αρχικό. Ομοίως το τρίτο προτεινόμενο θα σχετίζεται περισσότερο με το δεύτερο προτεινόμενο, παρά με τα προηγούμενα. Το τέταρτο προτεινόμενο τραγούδι θα είναι πιο σχετικό με το αρχικό τραγούδι, και θα επαναλαμβάνεται η ίδια διαδικασία.

youtube/threads_max

Ο αριθμός των παράλληλων threads που θα τρέχουν. Default τιμή είναι 16.

youtube/search_results

Ο μέγιστος αριθμός τραγουδιών που θα μας επιστραφεί κατά την αναζήτηση

youtube/playlist_max_videos

Ο μέγιστος αριθμός βίντεο που θα μας επιστρέψει η αναζήτηση.

youtube/youtube_dl_package

Επιλέγουμε ποιόν download manager θα χρησιμοποιήσουμε για κατέβασμα βίντεο και ήχου από το Youtube.

TuneIn [131]

[tunein]
enabled = true
timeout = 5000
filter =

tunein/enabled

Ενεργοποιούμε ή απενεργοποιούμε την δυνατότητα TuneIn

tunein/timeout

Η μέγιστη χρονική διάρκεια σε milliseconds που θα διαρκεί η αναζήτηση. Η προκαθορισμένη τιμή είναι 5000.

tunein/filter

Φιλτράρει τα αποτελέσματα αναζήτησης. Μπορούμε να επιλέξουμε σταθμό (station) ή πρόγραμμα (program). Αν το αφήσουμε κενό, απενεργοποιείται το φιλτράρισμα των αποτελεσμάτων

Bandcamp [132]

[bandcamp]
enabled = true
discover_pages = 1
collection_items = 50
discover_genres =
All
Electronic
Rock
Metal

Alternative
Hip-Hop/Rap
Experimental
Punk
Folk
Pop
Ambient
Soundtrack
World
Jazz
Acoustic
Funk
R&B/Soul
Devotional
Classical
Reggae
Podcasts
Country
Spoken Word
Comedy
Blues
Kids
Audiobooks
Latin
discover_tags =
Outrun
Future Funk
Alternative Hip-Hop
Tokyo, Japan
image_sizes =
10
5
2
identity =

bandcamp/enabled

Ενεργοποιούμε ή απενεργοποιούμε τη δυνατότητα Bandcamp

bandcamp/discover_pages

Ο αριθμός των σελίδων που θα που θα φορτώνονται κατά την αναζήτηση μουσικής. Η προκαθορισμένη τιμή είναι 1.

bandcamp/collection_items

Ορίζεται ο αριθμός των κομματιών ή των άλμπουμ ανά σελίδα που θα ανακαλέσουμε από την συλλογή μας στο Bandcamp, σε περίπτωση που γίνει αυθεντικοποίηση χρήστη. Η προκαθορισμένη τιμή είναι το 50.

bandcamp/discover_genres

Λίστα με τα είδη μουσικής που επιθυμούμε να αναζητούμε.

bandcamp/image_sizes

Λίστα με IDs που αντιστοιχούν σε μεγέθη εικόνας των εξώφυλλων δίσκων, τις οποίες θα επιστρέφει το Mopidy. Προεπιλεγμένες τιμές είναι 10, 5, 2 (1200x1200, 700x700, 350x350).

bandcamp/identity

Το token από τα cookies του Bandcamp που χρησιμοποιείται για την αυθεντικοποίηση χρήστη στο Bandcamp.

Internet Archive [133]

[internetarchive]
enabled = true
base_url = http://archive.org
collections =
audio

etree
librivoxaudio
audio_bookspoetry
audio_tech
audio_music
audio_news
audio_foreign
audio_podcast
audio_religion
audio_formats =
VBR MP3
64Kbps MP3
image_formats =
JPEG
JPEG Thumb
browse_limit = 100
browse_views =
downloads desc|Views
titleSorter asc|Title
publicdate desc|Date Archived
date desc|Date Published
creatorSorter asc|Creator
search_limit = 20
search_order =
cache_size = 128
cache_ttl = 86400
retries = 3
timeout = 10

internetarchive/enabled

Ενεργοποιούμε ή απενεργοποιούμε τη δυνατότητα Internet Archive.

internetarchive/base_url

Το βασικό URL για πρόσβαση στο Internet Archive.

internetarchive/collections

Λίστα με αναγνωριστικά τα οποία θα εμφανίζονται ως κατάλογοι κατά την αναζήτηση.

internetarchive/audio_formats

Λίστα με φορμάτ ήχου τα οποία θα αναζητούνται στο Internet Archive.

internetarchive/image_formats

Λίστα με φορμάτ εικόνας οι οποίες εικόνες θα παρέχονται από το Internet Archive.

internetarchive/browse_limit

Ο μέγιστος αριθμός αποτελεσμάτων κατά την περιήγηση στο Internet Archive.

internetarchive/browse_views

Λίστα με εικονικούς υποκαταλόγους ώστε να ανακτώνται αποτελέσματα βάσει μιας δεδομένης σειράς ταξινόμησης (sort order).

internetarchive/search_limit

Ο μέγιστος αριθμός των αποτελεσμάτων αναζήτησης.

internetarchive/search_order

Η σειρά ταξινόμησης που χρησιμοποιείται όταν κάνουμε αναζήτηση στο Internet Archive.

internetarchive/cache_size

Ο αριθμός των αντικειμένων από το Internet Archive που αποθηκεύονται στην κρυφή μνήμη (cache).

internetarchive/cache_ttl

Ο χρόνος σε δευτερόλεπτα για τον οποίο αποθηκεύεται ένα αντικείμενο στην κρυφή μνήμη. Μετά διαγράφεται.

internetarchive/retries

Ο μέγιστος αριθμός προσπαθειών που θα επιχειρήσει να πραγματοποιήσει κάθε HTTP σύνδεση.

internetarchive/timeout

Το χρονικό όριο σε δευτερόλεπτα όπου διαρκούν τα αιτήματα (requests) HTTP.

Mopidy-Local [134]

Η επέκταση Mopidy-Local μας επιτρέπει να αναπαράγουμε μουσική από εξωτερικούς σκληρούς δίσκους και usb sticks. Σε αντίθεση με την επέκταση Mopidy-File, μπορεί να δημιουργήσει ένα κατάλογο (index) με τα μεταδεδομένα (metadata) των τραγουδιών με αποτέλεσμα να μας παρέχει περαιτέρω δυνατότητες όπως αναζήτηση.

Τα μουσικά μεταδεδομένα αποθηκεύονται δε μια βάση δεδομένων SQLite, και μπορούμε να περιηγηθούμε στη μουσική συλλογή μας βάσει διαφόρων κριτηρίων και χαρακτηριστικών, όπως το όνομα του δίσκου, του καλλιτέχνη, του συνθέτη κ.α.

Για να διαβάσει τα περιεχόμενα των αφαιρούμενων αποθηκευτικών μέσων το Mopidy, θα πρέπει πρώτα να γίνει αλλαγή στα δικαιώματα του φακέλου /media/pi όπου γίνονται mounted τα αφαιρούμενα μέσα και να δώσουμε πλήρη δικαιώματα ανάγνωσης, εγγραφής και εκτέλεσης σε όλους τους χρήστες του συστήματος. Για να επιτευχθεί αυτό δίνουμε στο τερματικό την εντολή :

```
sudo chmod 777 /media/pi
```

Από εκεί και έπειτα να γίνεται η ανανέωση του περιεχομένου, θα πρέπει κάθε φορά που γίνεται κάποια αλλαγή, όπως στην περίπτωση που προσθέτουμε ή αφαιρούμε κάποιο αποθηκευτικό μέσο, θα πρέπει να εκτελούμε την εντολή

sudo mopidyctl local scan

Αυτή η διαδικασία μπορεί να γίνεται χειροκίνητα, ή να ορίσουμε μια αυτόματη εργασία (cron) η οποία να εκτελεί τη σάρωση ανά διαστήματα που θα έχουμε προκαθορίσει. Στην υλοποίησή μας θα τοποθετήσουμε την παραπάνω εντολή στο αρχείο /etc/rc.local , ώστε με κάθε επανεκκίνηση του Raspberry Pi να γίνεται αυτόματα σάρωση για την ανανέωση της μουσικής βιβλιοθήκης.

Αυτό επιτυγχάνεται με το να επεξεργαστούμε το αρχείο /etc/rc.local με την παρακάτω εντολή.

sudo nano /etc/rc.local

και προσθέτουμε την εξής εντολή

sudo mopidyctl local scan

μέσα στο περιεχόμενου κειμένου του αρχείου. Χρειάζεται προσοχή, ώστε η εντολή σάρωσης να μπει πριν το exit 0.

Για να ορίσουμε μια αυτόματη διαδικασία (cron) η οποία να εκτελεί τη σάρωση ανά διαστήματα που θα έχουμε προκαθορίσει, θα δώσουμε την εντολή

crontab -e

Με την εντολή αυτή επεξεργαζόμαστε το αρχείο crontab, ώστε να εισάγουμε λειτουργίες που θα εκτελούνται ανά χρονικά διαστήματα που θα ορίζουμε εμείς. Στην περίπτωσή μας θα ορίσουμε να γίνεται σάρωση κάθε ένα λεπτό. Στην περίπτωση που ερωτηθούμε ποιο κειμενογράφο προτιμούμε για την επεξεργασία, επιλέγουμε τον nano πατώντας την επιλογή 1. Έπειτα δίνουμε την εντολή

*/1 * * * * sudo mopidyctl local scan

και αποθηκεύουμε με ctrl+S και μετά ctrl+X.

```
[local]
enabled = true
max_search_results = 100
media_dir =  /media
scan_timeout = 1000
scan_flush_threshold = 100
scan_follow_symlinks = false
included_file_extensions =
excluded_file_extensions =
  .cue
  .directory
  .html
  .jpeg
  .jpg
  .log
  .nfo
  .pdf
  .png
  .txt
  .zip
directories =
  Albums                local:directory?type=album
  Artists               local:directory?type=artist
  Composers             local:directory?type=artist&role=composer
  Genres                local:directory?type=genre
  Performers            local:directory?type=artist&role=performer
  Release Years         local:directory?type=date&format=%25Y
  Tracks                local:directory?type=track
  Last Week's Updates   local:directory?max-age=604800
  Last Month's Updates  local:directory?max-age=2592000
timeout = 10
use_artist_sortname = false
```

album_art_files =
*.jpg
*.jpeg
*.png

local/enabled

Ενεργοποιήσουμε ή απενεργοποιούμε την επέκταση

local/max_search_results

Ο αριθμός των αποτελεσμάτων που θα επιστρέφονται κατά την αναζήτηση. Προεπιλεγμένη τιμή είναι το 100.

local/media_dir

Το path που οδηγεί στον φάκελο με τα αποθηκευμένα αρχεία ήχου.

local/scan_timeout

Η χρονική διάρκεια σε milliseconds που διαρκεί σάρωση ενός αρχείου, πριν πάει στο επόμενο κομμάτι.

local/scan_flush_threshold

Για πόσα μουσικά κομμάτια να περιμένει πριν να δώσει εντολή στη βιβλιοθήκη να αποθηκεύσει τα μέχρι εκείνη τη στιγμή αποτελέσματα της σάρωσης. Επιλέγοντας 0 απενεργοποιείται η αποθήκευση.

local/scan_follow_symlinks

Αν θα ακολουθούμε αρχεία symlink που βρίσκονται στο φάκελο local/media_dir.

local/ included_file_extensions

Ποιες επεκτάσεις αρχείων να συμπεριλαμβάνονται κατά την σάρωση των τοπικών αρχείων. Η κάθε επέκταση θα πρέπει να χωρίζεται με κόμμα ή να βρίσκεται σε νέα γραμμή και να ξεκινάει με τελεία. Καλύτερα να μην δηλώνουμε κάποια προέκταση σε αυτό το πεδίο γιατί αναιρεί την επόμενη ρύθμιση όπου ορίζουμε ποιες επεκτάσεις αρχείων θα αγνοούνται.

local/excluded_file_extensions

Ποιες επεκτάσεις αρχείων θα αγνοούνται κατά την σάρωση των τοπικών αρχείων. Η κάθε επέκταση θα πρέπει να χωρίζεται με κόμμα ή να βρίσκεται σε νέα γραμμή και να ξεκινάει με τελεία.

local/directories

Λίστα με ονόματα φακέλων του ριζικού καταλόγου (root directory) και URIs όπου θα γίνεται περιήγηση.

local/timeout

Ο χρόνος σε δευτερόλεπτα όπου θα πρέπει να επιτευχθεί σύνδεση με τη βάση δεδομένων.

local/use_artist_sortname

Αφήνουμε απενεργοποιημένη αυτή την εντολή, γιατί χρησιμοποιώντας συντομογραφίες για αναζήτηση, αρκετές φορές οδηγεί σε λανθασμένα αποτελέσματα.

local/album_art_files

Λίστα με επεκτάσεις αρχείων που θα επιστρέφονται κατά την αναζήτηση εξωφύλλων δίσκων.

3.4 Λειτουργίες χειρισμού client

Όλοι οι clients είτε είναι web clients είτε κάποια μεμονωμένη εφαρμογή υπό κανονικές συνθήκες αναπαράγουν τα κομμάτια που περιέχονται στο tracklist διαδοχικά, μέχρι να φτάσουν στο τέλος του tracklist, οπότε και σταματάει η αναπαραγωγή. Όμως μπορούν να ορίσουν τρόπους λειτουργίας αναπαραγωγής όπως οι εξής :

- **random** όπου τα κομμάτια του tracklist αναπαράγονται σε τυχαία σειρά
- **consume** όπου όταν παίξει κάποιο κομμάτι αφαιρείται από το tracklist
- **single** όπου αναπαράγεται μόνο ένα συγκεκριμένο κομμάτι και μετά σταματά η αναπαραγωγή
- και **repeat** όπου το tracklist επαναλαμβάνεται συνέχεια, εκτός αν παράλληλα έχει επιλεγεί και η λειτουργία single, οπότε παίζει συνεχώς ένα συγκεκριμένο κομμάτι

Επίσης οι clients και εκτελούν κάποιες βασικές λειτουργίες αναπαραγωγής. Αυτές είναι οι παρακάτω :

- **play** όπου ξεκινάει η αναπαραγωγή ενός επιλεγμένου κομματιού ή αν δεν επιλεγεί κάποιο κομμάτι, του τρέχοντος κομματιού της λίστας
- **stop** όπου γίνεται διακοπή της αναπαραγωγής
- **pause** όπου γίνεται παύση της αναπαραγωγής του κομματιού
- **resume** όπου γίνεται επανέναρξη της αναπαραγωγής του κομματιού από τη χρονική στιγμή που είχε γίνει η παύση
- **next** όπου γίνεται αναπαραγωγή του επόμενου κομματιού του tracklist ή αν έχει ενεργοποιηθεί η λειτουργία **random** επιλέγεται τυχαία ένα κομμάτι από τη λίστα, το οποίο δεν θα ξαναεπιλεχθεί μέχρι να παίξουν όλα τα κομμάτια της λίστας

- **previous** όπου γίνεται η αναπαραγωγή του προηγούμενου κομματιού του tracklist εκτός αν έχει επιλεγεί παράλληλα και η λειτουργία **random** ή/και η λειτουργία consume, οπότε θα παίξει το τρέχον κομμάτι της λίστας

- **set volume** όπου ορίζεται η στάθμη του ήχου

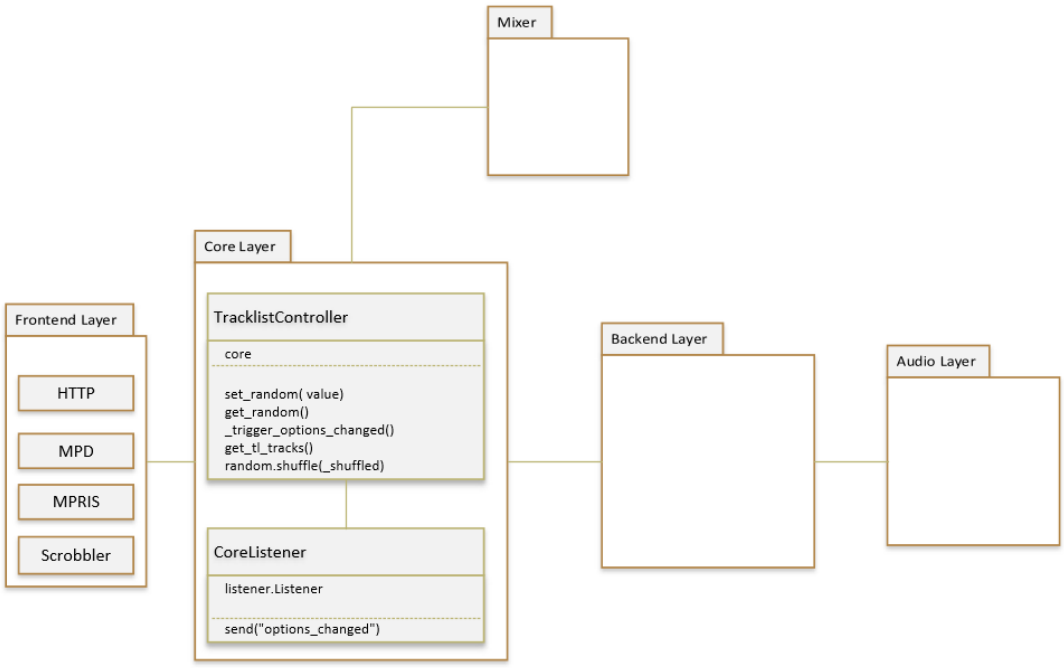
- **mute** όπου γίνεται σίγαση του ήχου

Σε αυτό το κεφάλαιο θα επιχειρήσουμε να αναλύσουμε τον τρόπο που υλοποιούνται οι παραπάνω διαδικασίες σε επίπεδο κώδικα, καθώς και θα γίνει απεικόνιση αυτών με UML διαγράμματα κλάσης. Υπενθυμίζουμε ότι ο κώδικας είναι ανοιχτός και είναι διαθέσιμος στο GitHub. [120]

ΣΗΜΕΙΩΣΗ

Η παρακάτω ανάλυση γίνεται καθαρά για εκπαιδευτικούς / ακαδημαϊκούς σκοπούς και αποτελεί προσωπική εκτίμηση του γράφοντα της διπλωματικής εργασίας αυτής και σε καμία περίπτωση δεν αποτελεί επίσημο εγχειρίδιο ή documentation του Mopidy project και δεν σημαίνει ότι εκφράζει τους δημιουργούς και ιδιοκτήτες του, δηλαδή τον Stein Magnus Jodal και τους συνεισφέροντες (contributors). Όποιος ενδιαφέρεται να εμβαθύνει και να συνεισφέρει στο εγχείρημα του Mopidy, θα πρέπει να συμβουλεύεται αποκλειστικά και μόνο το επίσημο εγχειρίδιο που βρίσκεται στον ιστότοπο https://docs.mopidy.com/en/latest/ ή στο https://docs.mopidy.com/_/downloads/en/latest/pdf/ καθώς και να απευθύνει τυχόν ερωτήματα που προκύπτουν στο ιδιαίτερα ενεργό forum που βρίσκεται στον ιστότοπο https://discourse.mopidy.com/

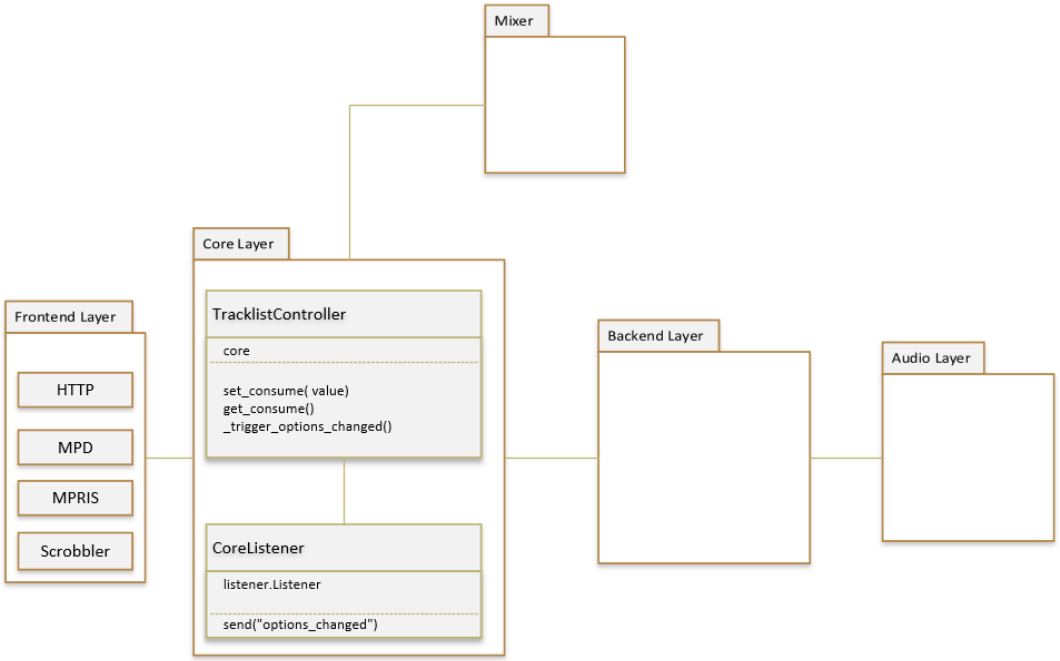
3.4.1 Τυχαία αναπαραγωγή κομματιών random



Εικόνα 70: Τυχαία αναπαραγωγή κομματιών random

Όταν ο client ζητάει να γίνεται τυχαία αναπαραγωγή των κομματιών του tracklist, ενεργοποιεί ή απενεργοποιεί την λειτουργία **random**, δηλαδή αν τα κομμάτια του tracklist θα παίζουν με τυχαία σειρά ή με τη σειρά που έχουν στο tracklist με το να καλέσει την συνάρτηση **set_random**(value) που βρίσκεται στο **TracklistController** [140] του **Core layer**. Αν η τιμή value είναι **True**, σημαίνει ότι το frontend ζήτησε να ενεργοποιηθεί η λειτουργία random, οπότε τα κομμάτια θα εκτελούνται με τυχαία σειρά, και αν είναι **False** θα ακολουθείται η σειρά σύμφωνα με τη σειρά του tracklist. Αρχικά γίνεται σύγκριση της τιμής **value** που δίνεται από τον client με αυτή της τρέχουσας κατάστασης τυχαίας αναπαραγωγής η οποία προκύπτει με τη χρήση της **get_random**() και αν είναι διαφορετική, καλείται η **_trigger_options_changed**() η οποία με τη σειρά της καλεί τη **send**("options_changed") του **CoreListener** [141] η οποία ενημερώνει τους ενδιαφερόμενους listeners ότι η επιλογή της τυχαίας αναπαραγωγής άλλαξε, στέλνοντας ένα event "**options_changed**". Έπειτα με την **get_tl_tracks**() ανακτάται η λίστα των κομματιών και με την συνάρτηση **random**() της **Python** αλλάζει η σειρά της λίστας των κομματιών και η λειτουργία **random** μπαίνει στη νέα κατάσταση παίρνοντας τη νέα τιμή **value**.

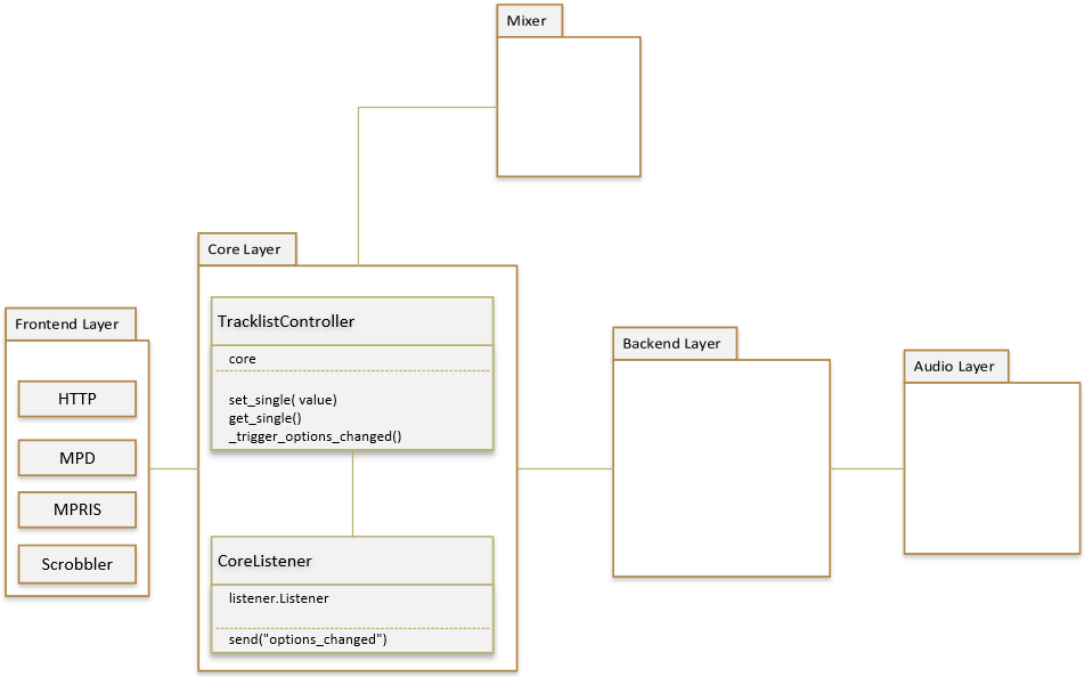
3.4.2 Λειτουργία Consume



Εικόνα 71: Λειτουργία consume

Όταν ο client ζητήσει να αλλάξει η κατάσταση λειτουργίας σε κατάσταση consume, σημαίνει ότι όταν τα κομμάτια παίξουν, θα αφαιρούνται από το tracklist. Οπότε ο client καλεί την **set_consume**(value) από το **TracklistController** του **core layer**, όπου αν η τιμή της **value** είναι **True**, τα κομμάτια θα αφαιρούνται από το tracklist αφού παίξουν, αλλιώς αν η τιμή **value** θα είναι **False**, τα κομμάτια δεν θα αφαιρούνται από το tracklist. Αρχικά γίνεται έλεγχος της τρέχουσας κατάστασης με τη χρήση της **get_consume**() και αν είναι διαφορετική από αυτή που ζητάει ο client, καλείται η **_trigger_options_changed**() η οποία με τη σειρά της καλεί τη **send**() του **CoreListener** η οποία ενημερώνει τους ενδιαφερόμενους listeners ότι η επιλογή της λειτουργίας **consume** άλλαξε με την αποστολή ενός event με όνομα **"options_changed"** και τέλος η λειτουργία **consume** μπαίνει στη νέα κατάσταση παίρνοντας τη νέα τιμή **value**.

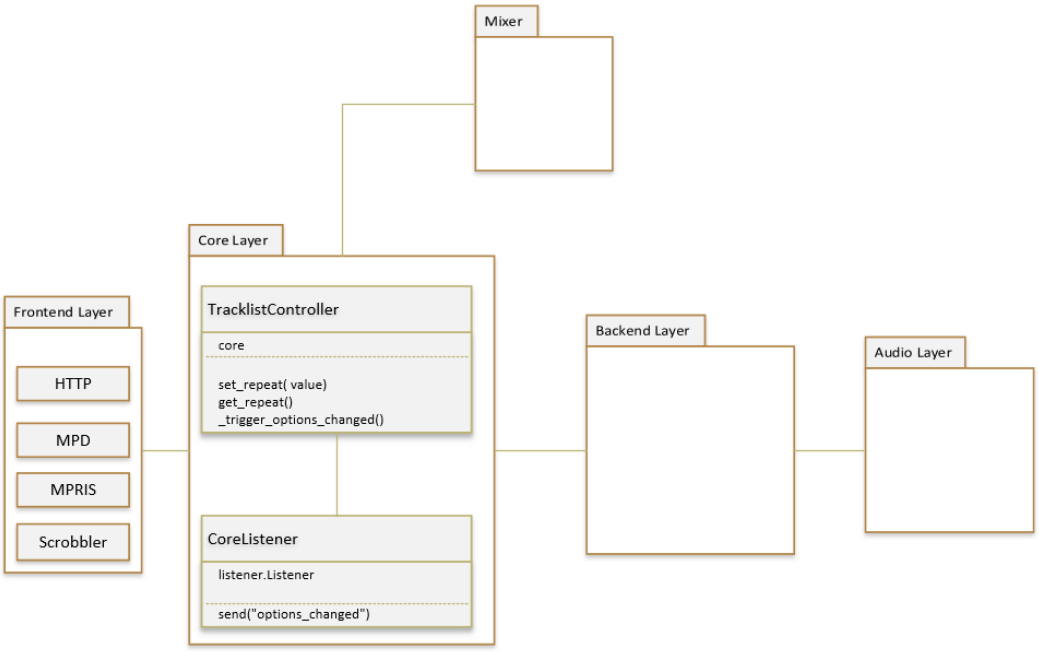
3.4.3 Λειτουργία αναπαραγωγής μόνο ενός κομματιού και μετά διακοπή αναπαραγωγής Single



Εικόνα 72: Λειτουργία Single

Η λειτουργία **single** είναι η λειτουργεία κατά την οποία αναπαράγεται μόνο ένα συγκεκριμένο κομμάτι και μετά σταματά η αναπαραγωγή εκτός αν έχει επιλεγεί παράλληλα και η λειτουργία **repeat**, όποτε θα παίζει συνέχεια το ίδιο κομμάτι. Για να οριστεί η κατάσταση λειτουργίας ως **single**, ο client καλεί την **set_single**(value) από το **TracklistController** του **core layer**, όπου ελέγχεται η τρέχουσα κατάσταση **single** με τη **get_single**() και αν διαπιστωθεί ότι δεν είναι ίδια με τη κατάσταση που ζητάει ο client και αντιστοιχεί στην τιμή **value**, καλείται η **_trigger_options_changed**() η οποία με τη σειρά της καλεί τη **send**() του **CoreListener** η οποία ενημερώνει τους ενδιαφερόμενους listeners ότι η επιλογή της λειτουργίας **single** άλλαξε στέλνοντας event με το όνομα **"options_changed"** και τέλος η αναπαραγωγή μπαίνει στη νέα κατάσταση, σύμφωνα με τη τιμή της **value**.

3.4.4 Λειτουργία επανάληψης αναπαραγωγής repeat

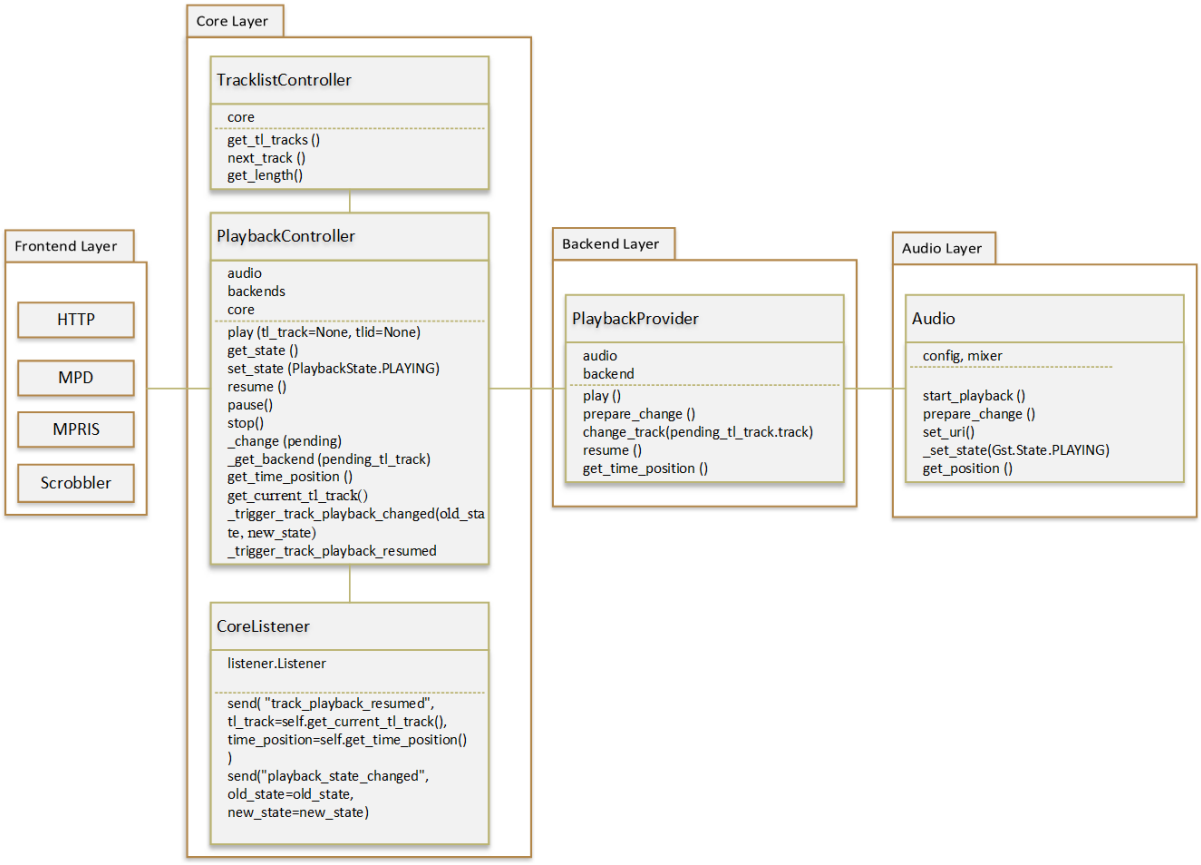


Εικόνα 73: Λειτουργία επανάληψης αναπαραγωγής repeat

Όταν ο client θέλει να τεθεί η λειτουργεία αναπαραγωγής σε κατάσταση επανάληψης, δηλαδή να παίζει συνέχεια το ίδιο tracklist, καλεί τη συνάρτηση **set_repeat**(value) από το **TracklistController** του **core layer** με τιμή της value να είναι True, αλλιώς με False παύει η κατάσταση επανάληψης. Αν θα θέλει να επαναλαμβάνεται συνεχώς το ίδιο κομμάτι, θα πρέπει να ενεργοποιηθεί παράλληλα και η κατάσταση **single**. Με την κλήση της **set_repeat**(value) γίνεται έλεγχος με την **get_repeat**() αν η κατάσταση επανάληψης που ζητείται από τον client είναι ίδια με την τρέχουσα κατάσταση λειτουργίας **repeat** και αν δεν είναι, καλείται η **_trigger_options_changed**() η οποία με τη σειρά της καλεί τη **send**() του **CoreListener** η οποία ενημερώνει τους ενδιαφερόμενους listeners ότι η επιλογή της λειτουργίας **repeat** άλλαξε στέλνοντας event με το όνομα **"options_changed"**και η λειτουργεία **repeat** μπαίνει στη νέα κατάσταση, που αντιστοιχεί στην τιμή **value**.

Παρακάτω παρατίθεται ανάλυση των βασικών λειτουργιών αναπαραγωγής. Λόγω του ότι αρκετές μέθοδοι επιστρέφουν pykka.ThreadingFuture αντικείμενα, στον κώδικα παρατηρούμε ότι προστίθεται στο τέλος των κλήσεων των μεθόδων αυτών και η μέθοδος **get**(), ώστε να λαμβάνουμε την επιστρεφόμενη (actual) τιμή της μεθόδου όταν θα είναι διαθέσιμη.

3.4.5 Αναπαραγωγή κομματιού Play



Εικόνα 74: Αναπαραγωγή κομματιού Play

Ο client ζητάει να αναπαραχθεί ένα συγκεκριμένο κομμάτι βάσει του ονόματος του (tl_track) ή του tracklist ID του (tlid), οπότε στέλνει το αντίστοιχο αίτημα στο core layer, και τη διαχείριση του αιτήματος αυτού την κάνει ο **PlaybackController** [142] με τη χρήση της μεθόδου **play**(*tl_track=None*, *tlid=None).* Το κομμάτι που ζητήθηκε να αναπαραχθεί θα πρέπει να βρίσκεται ήδη στο tracklist και θα πρέπει να ζητηθεί ή το όνομα του κομματιού ή το tracklist ID του, όχι και τα δύο ταυτόχρονα. Το tracklist το ανακτούμε με την κλήση της συνάρτησης **get_tl_tracks**() του **TracklistController**. Αν το κομμάτι που ζητηθεί δεν υπάρχει στο tracklist, θα επιστραφεί μήνυμα λάθους.

Αν δεν δοθεί tl_track και αν διαπιστωθεί με τη χρήση της **get_state**() ότι η παρούσα κατάσταση αναπαραγωγής είναι **PAUSED**, καλείται η συνάρτηση **resume**() η οποία βρίσκει το κατάλληλο backend που αντιστοιχεί στο κομμάτι με την

_get_backend(get_current_tl_track()), καλεί την **resume**() του **Playback Provider** και αυτή με τη σειρά της καλεί την **start_playback**() του **Audio Layer** και αν ξεκινήσει εκ νέου η αναπαραγωγή επιτυχώς, γίνεται αλλαγή της κατάστασης αναπαραγωγής με την **set_state**(PlaybackState.PLAYING) του **Playback Controller** σε κατάσταση PLAYING.

Μετά μέσω της **set_state**(PlaybackState.PLAYING) καλείται η **_trigger_playback_state_changed**(old_state, new_state) και μέσω του **CoreListener** και συγκεκριμένα της **send**("playback_state_changed", old_state=old_state, new_state=new_state) ενημερώνεται το frontend ότι άλλαξε η κατάσταση αναπαραγωγής με την αποστολή του event με όνομα **"playback_state_changed"** και ποια είναι η νέα κατάσταση.

Επίσης με την κλήση της **resume**() καλείται και η **_trigger_track_playback_resumed**(tl_track, get_time_position) και στέλνεται το event **"track_playback_resumed"** μέσω της μεθόδου **send**("track_playback_resumed",tl_track=get_current_tl_track(),time_position=get_time_position()) του **CoreListener** το οποίο ενημερώνει για ποιο κομμάτι πρόκειται καθώς την ακριβή χρονική θέση σε milliseconds στην οποία ξεκινάει εκ νέου η αναπαραγωγή του κομματιού. Η **get_time_position**() του **PlaybackController** καλεί την **get_time_position**() του **PlaybackProvider** από το **Backend layer** [143] και αυτή με τη σειρά της την **get_position**() του **audio actor** [144] και επιστρέφει τη χρονική στιγμή του κομματιού σε milliseconds.

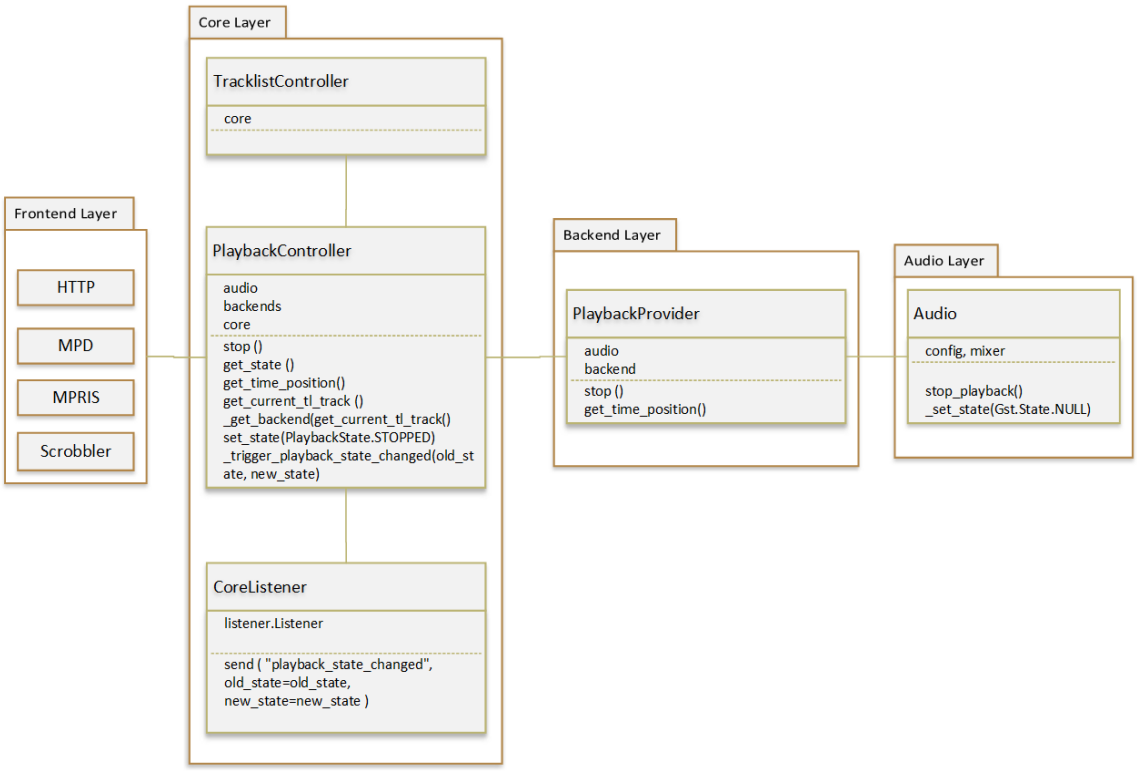
Εάν δεν έχει επιλεγεί λειτουργία επανάληψης εκτέλεσης (**repeat**) ή εκτέλεσης του κομματιού μόνο μια φορά και μετά να διακοπεί η αναπαραγωγή (**single**), με τη βοήθεια ενός βρόχου **while** μετά το τέλος της εκτέλεσης του συγκεκριμένου κομματιού θα εκτελείται κάθε φορά το επόμενο κομμάτι που θα ανακτάται με την **next_track**(tl_track) ώσπου να εκτελεστούν όλα τα κομμάτια του παρόντος tracklist είτε με τη σειρά είτε ανακατεμένα, εάν είναι ενεργοποιημένη η λειτουργία **random** και μετά θα σταματήσει η αναπαραγωγή. Αν δεν υπάρχει επερχόμενο κομμάτι στο tracklist, διακόπτεται η αναπαραγωγή. Για να εξασφαλιστεί ότι θα παίξουν όλα τα κομμάτια, ανακτάται το μήκος του tracklist με την **get_length**() του **TracklistController** και πολλαπλασιάζεται με το 2.

Αναλυτικότερα καλείται η **_change**(pending_tl_track, state) του **PlaybackController**. Όπου pending_tl_track είναι αρχικά το επιθυμητό κομμάτι προς αναπαραγωγή και έπειτα το επόμενο κομμάτι του tracklist, και **state** η κατάσταση αναπαραγωγής και στην περίπτωση αυτή η κατάσταση **PlaybackState.PLAYING**. Αν δεν υπάρχει επόμενο κομμάτι, σταματάει η αναπαραγωγή με τη **stop**().

Μετά γίνεται ανάκτηση του κατάλληλου backend σύμφωνα με το uri που αντιστοιχεί στο όνομα του συγκεκριμένου κομματιού με τη χρήση της **_get_backend**(pending_tl_track) του **PlaybackController**. Αφού βρεθεί κάποιο backend, ανακτάται η ακριβής χρονική στιγμή που βρίσκεται το κομμάτι σε milliseconds με την **get_time_position**() και ενημερώνεται ο **GStreamer** ότι επίκειται αλλαγή με την **prepare_change**() του **PlaybackProvider** του αντίστοιχου backend καλώντας την **prepare_change**() του **audio actor**.

Έπειτα με την **change_track**(pending_tl_track.track) του **PlaybackProvider** από το **backend layer** δίνεται εντολή μετάβασης στο επιθυμητό κομμάτι που ορίζεται με το uri του με την κλήση της **set_uri**() του **Audio Layer** όπου ενημερώνεται ο Gstreamer για το κομμάτι που θα αναπαράγει. Εφόσον η κατάσταση αναπαραγωγής είναι σε κατάσταση **PLAYING**, καλεί την **play**() η οποία ανήκει στο **backend layer** και συγκεκριμένα στον **Playback Provider**, η οποία **play**() καλεί την **start_playback**() που βρίσκεται στο **audio actor** η οποία καλεί την **_set_state**(Gst.State.PLAYING) και που θέτει τον **GStreamer** σε κατάσταση **PLAYING**, με συνέπεια να ξεκινήσει η αναπαραγωγή. Αν για κάποιο λόγο δεν είναι επιτυχής η έναρξη αναπαραγωγής, το κομμάτι θεωρείται μη εκτελέσιμο και επιλέγεται το επόμενο κομμάτι με τη **next_track**(). Η διαδικασία αυτή επαναλαμβάνεται με τη χρήση του βρόχου **while** έως ότου να μην υπάρχει άλλο επόμενο κομμάτι στο tracklist.

3.4.6 Διακοπή Αναπαραγωγής κομματιού Stop

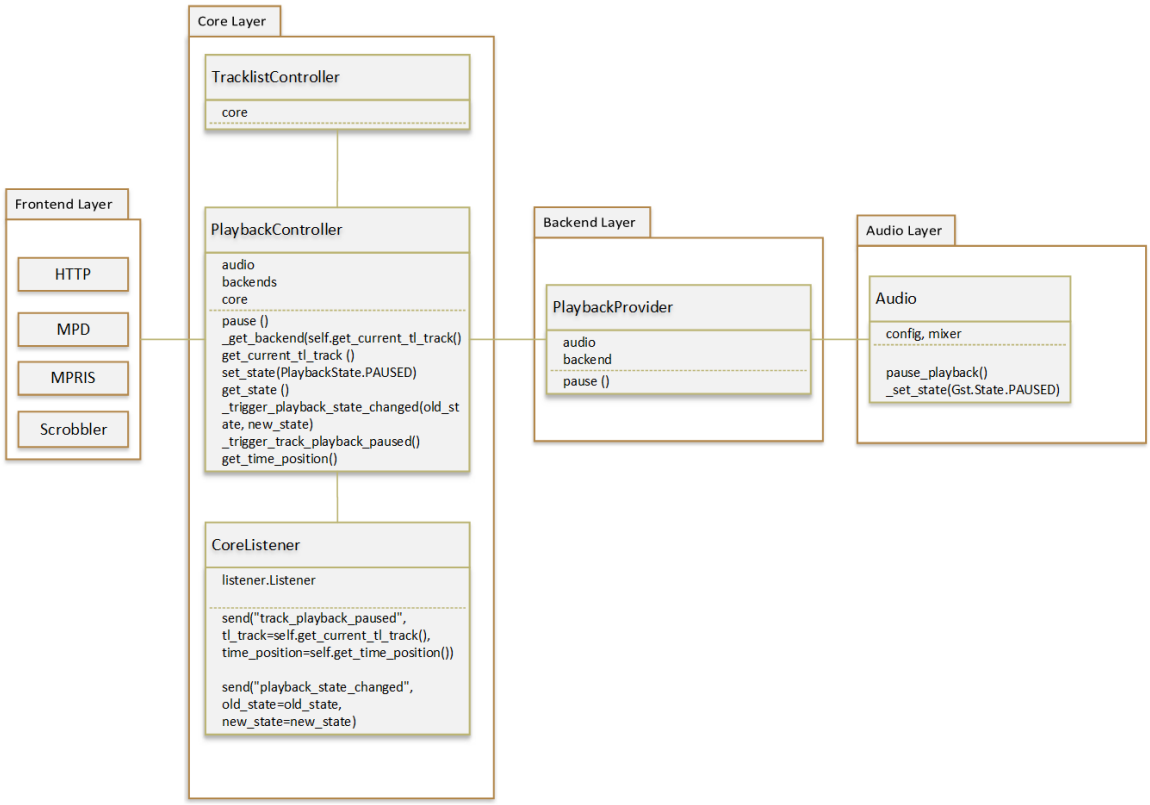


Εικόνα 75: Διακοπή Αναπαραγωγής κομματιού Stop

Ο client ζητάει διακοπή της αναπαραγωγής από το **core layer** οπότε καλείται η **stop**() από τον **PlaybackController**. Γίνεται έλεγχος της κατάστασης αναπαραγωγής με τη χρήσης της **get_state**() και αν η κατάσταση αναπαραγωγής δεν είναι ήδη σε κατάσταση **STOPPED**, με την **get_time_position**() κρατάει την χρονική στιγμή στην οποία βρίσκεται το κομμάτι που αναπαράγεται σε milliseconds, ανακτά το κατάλληλο backend που αντιστοιχεί στο τρέχον κομμάτι με την **_get_backend**(get_current_tl_track()) και καλεί την **stop**() από τον **Playback Provider**. Αν δεν βρεθεί κάποιο backend ή αν αφού κληθεί επιτυχώς ή **stop**() του **Playback Provider**, θέτει την κατάσταση αναπαραγωγής σε κατάσταση **STOPPED** μέσω της **set_state**(PlaybackState.STOPPED) και μέσω του **CoreListener** ενημερώνει το frontend για την αλλαγή αυτή με την κλήση της **_trigger_playback_state_changed**(old_state, new_state) η οποία στέλνει event με όνομα **"playback_state_changed"** με την παλιά και τη νέα κατάσταση.

Η **stop**() του **PlaybackProvider** καλεί την **stop_playback**() του **audio actor**, η οποία καλεί την **_set_state**(Gst.State.NULL) με την οποία ενημερώνει τον **GStreamer** ότι πρέπει να διακόψει την αναπαραγωγή και τον θέτει σε κατάσταση **NULL**.

3.4.7 Παύση Αναπαραγωγής κομματιού Pause

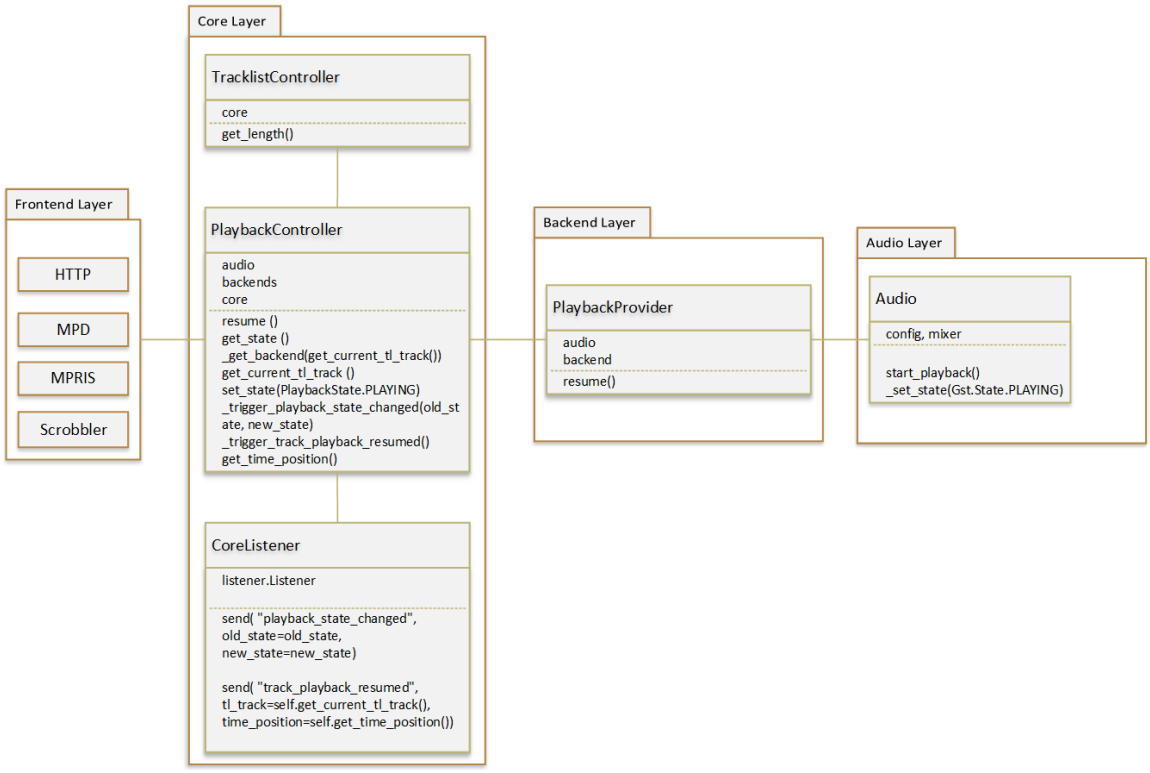


Εικόνα 76: Παύση Αναπαραγωγής κομματιού Pause

Ο client ζητάει **παύση** της αναπαραγωγής από το **core layer** οπότε καλείται η **pause**() από τον **PlaybackController**. Αυτή προσπαθεί να αναγνωρίζει το κατάλληλο backend που αντιστοιχεί στο εκτελούμενο κομμάτι με τη **_get_backend**(get_current_tl_track()) του **PlaybackController** και καλεί την **pause**() από το **Playback Provider** του συγκεκριμένου backend. Το τρέχον κομμάτι ανακτάται με την **get_current_tl_track**() του **PlaybackController**. Η **pause**() του **PlaybackProvider** καλεί την **pause_playback**() του **audio actor**, η οποία με τη συνάρτηση **_set_state**(PlaybackState.PAUSED) ενημερώνει τον **GStreamer** ότι θα πρέπει να κάνει παύση αναπαραγωγής θέτοντάς τον σε κατάσταση **PAUSED**.

Αν δεν βρεθεί κάποιο backend ή η **pause**() εκτελεστεί επιτυχώς, με την **set_state**(PlaybackState.PAUSED) του **Playback Controller** αλλάζει την κατάσταση αναπαραγωγής σε PAUSED και με την **_trigger_playback_state_changed**(old_state, new_state) ενημερώνεται το frontend μέσω του **CoreListener** ότι άλλαξε η κατάσταση αναπαραγωγής του τρέχοντος κομματιού σε **PAUSED** από την προηγούμενη κατάσταση, η οποία ανακτάται με την **get_state**(). Με την **_trigger_track_playback_paused**(), ενημερώνεται το frontend μέσω του **CoreListener** στέλνοντας event με όνομα **"track_playback_paused"** για το ότι έγινε παύση αναπαραγωγής του τρέχοντος κομματιού, για ποιο κομμάτι πρόκειται καθώς και την ακριβή χρονική στιγμή του τρέχοντος κομματιού που έγινε η παύση. Η χρονική αυτή στιγμή προκύπτει με τη χρήση της **get_time_position**().

3.4.8 Συνέχιση αναπαραγωγής resume



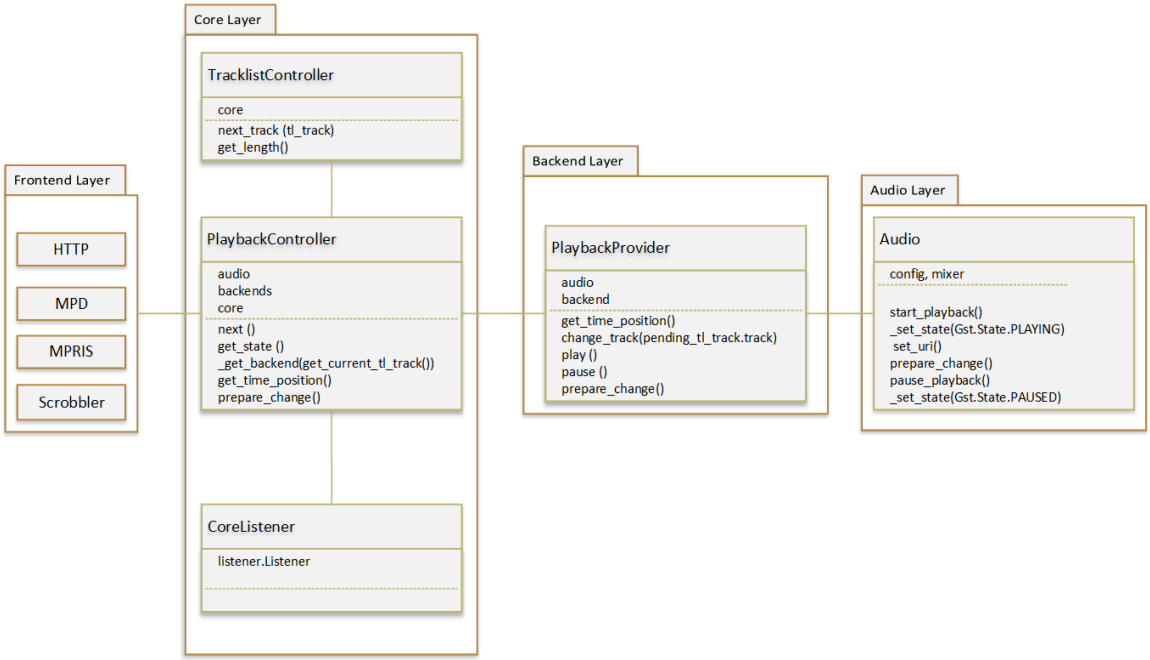
Εικόνα 77: Συνέχιση αναπαραγωγής resume

Ο client στο **frontend layer** στέλνει εντολή στο **core** και συγκεκριμένα στο **PlaybackController** να συνεχίσει η αναπαραγωγή καλώντας τη συνάρτηση **resume**().

Αυτή ελέγχει την τρέχουσα κατάσταση αναπαραγωγής με την **get_state**() και αν είναι σε παύση, προσπαθεί να αναγνωρίσει το backend που αντιστοιχεί στο τρέχον κομμάτι με την χρήση της **_get_backend**(get_current_tl_track()). Το τρέχον κομμάτι ανακτάται με την **get_current_tl_track**() του **PlaybackController**. Έπειτα καλείται η συνάρτηση **resume**() του **PlaybackProvider** του κατάλληλου backend το οποίο ανακτάται με την **_get_backend**(get_current_tl_track()) και αν εκτελεστεί επιτυχώς, θέτει την κατάσταση αναπαραγωγής σε κατάσταση **PLAYING** με την **set_state**(PlaybackState.PLAYING) και ενημερώνεται μέσω του **CoreListener** το frontend ότι η κατάσταση αναπαραγωγής άλλαξε με ένα **event** με όνομα **"playback_state_changed"** με την κλήση της **_trigger_playback_state_changed**(old_state, new_state) και ότι συνεχίζεται η αναπαραγωγή με ένα **event** με όνομα **"track_playback_resumed"**, καθώς και για ποιο κομμάτι πρόκειται και την χρονική στιγμή που είχε γίνει η παύση αναπαραγωγής με τη **_trigger_track_playback_resumed**().

Στο backend επίπεδο και συγκεκριμένα στο **PlaybackProvider** όταν καλείται η **resume**(), αυτή καλεί την **start_playback**() από το **audio actor** η οποία ενημερώνει τον **GStreamer** με την **_set_state**(Gst.State.PLAYING) να μπει σε κατάσταση **PLAYING** και να ξεκινήσει η αναπαραγωγή.

3.4.9 Επόμενο κομμάτι next



Εικόνα 78: Επόμενο κομμάτι next

Ο client ζητάει να παίξει το επόμενο κομμάτι στο tracklist οπότε καλεί την συνάρτηση **next**() του **PlaybackController** στο επίπεδο **core**. Αρχικά ζητείται η τρέχουσα κατάσταση αναπαραγωγής με την **get_state**(). Η κατάσταση αναπαραγωγής θα διατηρηθεί δηλαδή αν η κατάσταση αναπαραγωγής ήταν σε παύση, θα παραμείνει σε παύση, αν ήταν **PLAYING**, θα παραμείνει **PLAYING** κλπ.

Σε περίπτωση που είναι ενεργοποιημένη η λειτουργία **random**, θα παίξει κάποιο τυχαίο κομμάτι του tracklist, και όλα τα κομμάτια θα παίξουν από μια φορά, πριν αρχίσει να επαναλαμβάνεται το tracklist. Αυτό επιτυγχάνεται με την χρήση της **get_length**() του **TracklistController** όπου ανακτάται το μήκος του tracklist, και διπλασιάζεται ώστε να παίξουν σίγουρα όλα τα κομμάτια του tracklist, με τη χρήση ενός βρόχου **while**.

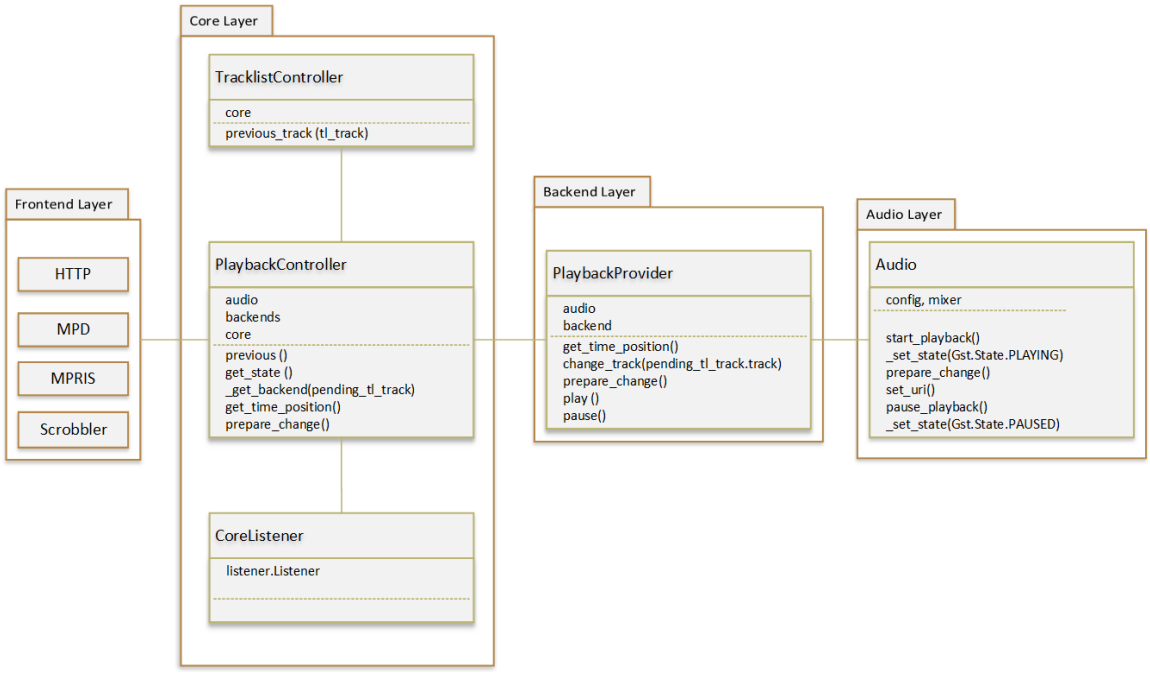
Καλώντας την συνάρτηση **next_track**(tl_track) του **TracklistController** όπου tl_track είναι το τρέχον κομμάτι, επιστρέφεται το επόμενο κομμάτι το οποίο θα αναπαραχθεί. Υπό κανονικές συνθήκες θα είναι το επόμενο κομμάτι στο tracklist.

Με τη χρήση της **_change**(pending, state) εάν δεν υπάρχει επερχόμενο κομμάτι γίνεται διακοπή της αναπαραγωγής, αλλιώς γίνεται κλήση της **_get_backend**(pending_tl_track) ώστε να ανακτηθεί το κατάλληλο **backend** που αντιστοιχεί στο κομμάτι. Έπειτα ανακτάται η ακριβής χρονική στιγμή που βρίσκεται το κομμάτι σε milliseconds με την **get_time_position**(), ειδοποιείται ο **GStreamer** ότι θα γίνει αλλαγή της κατάστασής του με την **prepare_change**() του **PlaybackProvider** και την **prepare_change**() του **audio layer** και δίνεται εντολή μετάβασης στο επόμενο κομμάτι με την **change_track**(pending_tl_track.track) του **PlaybackProvider** και ακολούθως ο Gstreamer ενημερώνεται για το ποιο κομμάτι θα παίξει με τη χρήση της **set_uri**() του **Audio Layer**.

Αν η μετάβαση γίνει επιτυχώς και αν η κατάσταση αναπαραγωγής είναι **PLAYING**, καλείται η **play**() η οποία ανήκει στο **Backend Layer** και συγκεκριμένα στον **Playback Provider**, η οποία **play**() καλεί την **start_playback**() που βρίσκεται στο **audio actor** και που θέτει τον **GStreamer** σε κατάσταση **PLAYING** με την κλήση της μεθόδου **_set_state**(Gst.State.PLAYING), με συνέπεια να ξεκινήσει η αναπαραγωγή. Αν η κατάσταση αναπαραγωγής είναι **PAUSED** καλείται η **pause**() του **Playback Provider** η οποία καλεί την **pause_playback**() του **audio actor** και αυτή με τη σειρά της την **_set_state**(Gst.State.PAUSED) του **audio actor,** θέτοντας το GStreamer σε κατάσταση **PAUSED**. Αν για κάποιο λόγο δεν είναι επιτυχής η έναρξη αναπαραγωγής ή η ενεργοποίηση της κατάστασης παύσης, το κομμάτι θεωρείται μη εκτελέσιμο και επιλέγεται το επόμενο κομμάτι. Η διαδικασία αυτή επαναλαμβάνεται με τη χρήση του βρόχου **while** έως ότου να μην υπάρχει άλλο επόμενο κομμάτι στο tracklist.

Αν η κατάσταση αναπαραγωγής είναι STOPPED, το επόμενο (pending_tl_track) γίνεται και το τρέχον κομμάτι (current_tl_track), και μετά το επόμενο κομμάτι (pending_tl_track) παίρνει την τιμή None, οπότε ο βρόχος **while** φτάνει στο τέλος και έχουμε διακοπή της αναπαραγωγής.

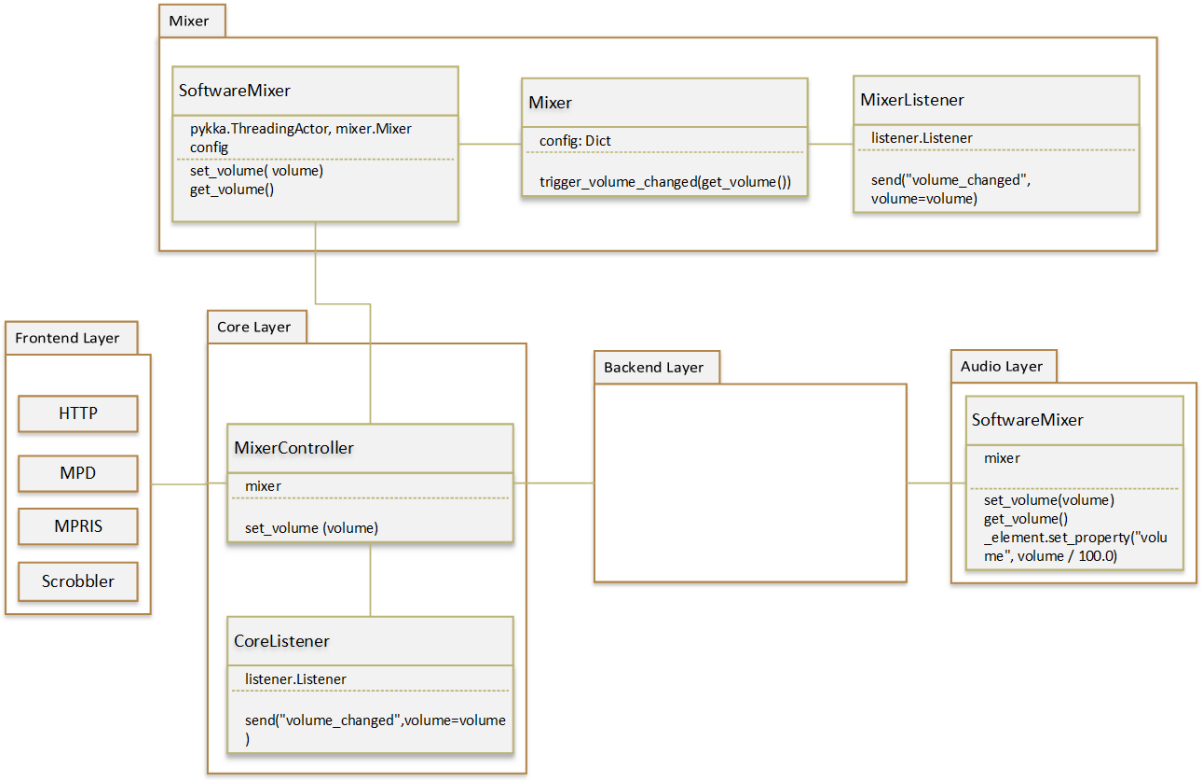
3.4.10 Προηγούμενο κομμάτι previous



Εικόνα 79: Προηγούμενο κομμάτι previous

Ακολουθείται η ίδια διαδικασία με αυτή του επόμενου κομματιού **next**, αλλά αυτή τη φορά ως επερχόμενο κομμάτι (pending) είναι το προηγούμενο κομμάτι τρέχοντος κομματιού (current) του tracklist και ανακτάται με την συνάρτηση **previous_track**(current) από το **TracklistController**. Υπό κανονικές συνθήκες είναι το αμέσως προηγούμενο κομμάτι στο tracklist. Αν είναι ενεργοποιημένες οι λειτουργίες **random** ή/και **consume**, θα παίξει το τρέχον κομμάτι.

3.4.11 Ρύθμιση έντασης ήχου volume



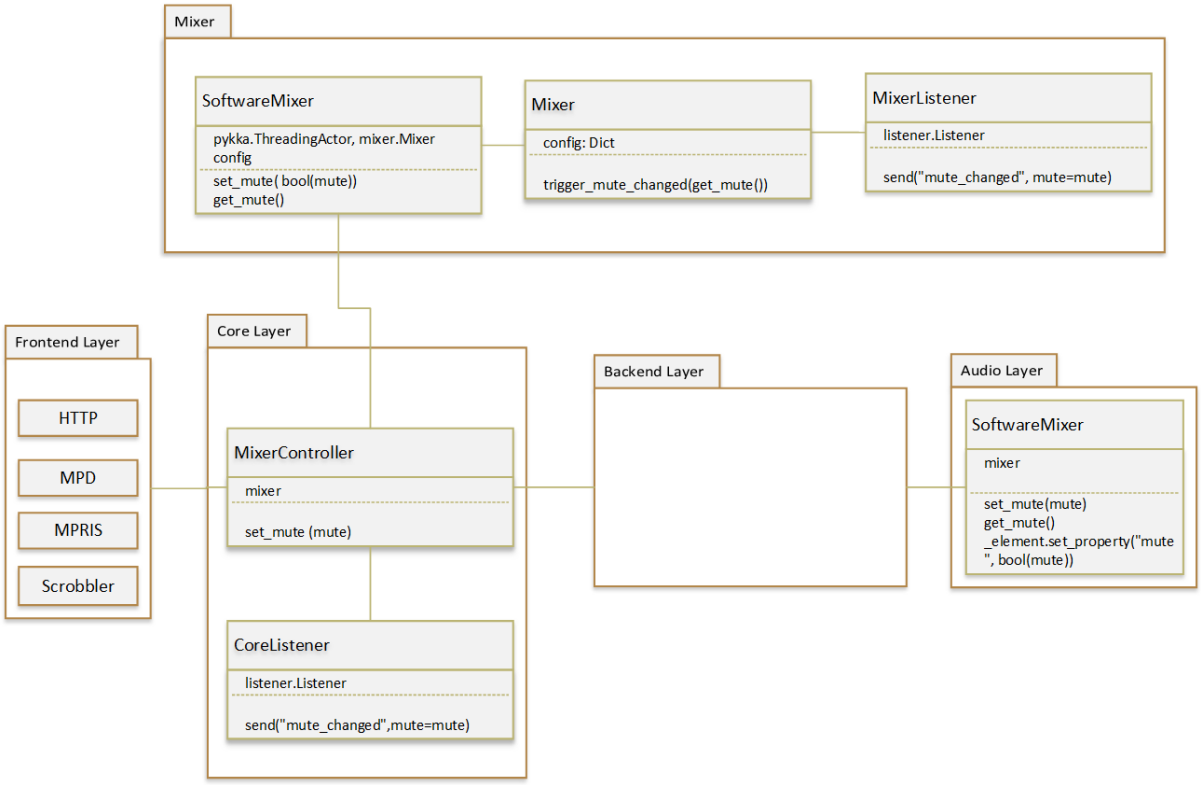
Εικόνα 80: Ρύθμιση έντασης ήχου volume

Ο client ζητάει να αλλάξει η ένταση ήχου σε μία τιμή από 0 έως 100 με γραμμική κλίμακα, όπου 0 είναι σίγαση και 100 η μέγιστη ένταση. Η τιμή αυτή είναι η τιμή της μεταβλητής **volume**. Το **core** λαμβάνει αυτή την εντολή και μέσω του **MixerController** [145] καλείται ή μέθοδος **set_volume**(volume) από το **SoftwareMixer** [146] το οποίο είναι το **default mixer** και το οποίο υλοποιεί και το **Audio Mixer API (mixer.Mixer)** [147] . Η μέθοδος αυτή καλεί την **set_volume**(volume) από το **audio actor**, η οποία θέτει τη στάθμη στο **GStreamer** στην τιμή **volume** που ζητήθηκε από τον client, διαιρεμένη με το 100, γιατί ο **GStreamer** δέχεται τιμές από 0 έως 1, με την χρήση της **_element.set_property**("volume", volume / 100.0).

Έπειτα μέσω του **SoftwareMixer (default mixer)** καλείται η **trigger_volume_changed**(get_volume()) του **Audio Mixer API,** με αποτέλεσμα ο **MixerListener** [147] με τη **send**("volume_changed", volume=volume) να στείλει ένα **event** με όνομα "**volume_changed"** που ενημερώνει τους listeners ότι η στάθμη ήχου άλλαξε και πήρε ως νέα τιμή την ζητούμενη από το client τιμή. Η τιμή της έντασης ήχου ανακτάται

με τη χρήση της **get_volume**(). Ο **core actor** λαμβάνει το event αυτό και μέσω του **CoreListener** στο **core layer** προωθεί το event σχετικά την αλλαγή αυτή με την χρήση της **volume_changed**(volume). Αυτή η μέθοδος καλεί την **send**("volume_changed", volume=volume) του **CoreListener** με αποτέλεσμα να ενημερωθεί και το **frontend** ότι η στάθμη ήχου άλλαξε στην επιθυμητή τιμή.

3.4.12 Κατάσταση σίγασης ήχου mute



Εικόνα 81: Κατάσταση σίγασης ήχου mute

Παρόμοια με τη ρύθμιση έντασης του ήχου, ο client ζητάει από το **core layer** να αλλάξει την σίγαση ήχου (**mute**) οπότε καλεί τη μέθοδο **set_mute**(mute) του **MixerController** του **core layer** όπου η μεταβλητή **mute** παίρνει την τιμή **True** αν θέλουμε να ενεργοποιήσουμε την κατάσταση σίγασης ή **False** αν το σύστημα είναι ήδη σε κατάσταση σίγασης και θέλουμε να το αναιρέσουμε αυτό. Ο **MixerController** καλεί την **set_mute**(bool(mute)) του **SoftwareMixer**, η οποία με τη σειρά της καλεί την **set_mute**(mute) του **audio actor** η οποία

θέτει τον **GStreamer** σε κατάσταση σίγασης με την **_element.set_property**("mute", bool(mute)).

Στη συνέχεια μέσω του **SoftwareMixer** καλείται η **trigger_mute_changed**(get_mute()) του **Audio Mixer API,** με αποτέλεσμα ο **MixerListener** να στείλει event με το όνομα **mute_changed** με την κλήση της **send**("mute_changed", mute=mute) και ενημερώνει τους listeners ότι η κατάσταση σίγασης άλλαξε στη νέα κατάσταση η οποία ανακτάται με την **get_mute**().

Με τη σειρά του ο **core actor** προωθεί το event αυτό με το να κάνει χρήση της **mute_changed**(mute) του όπου καλείται η **send**("mute_changed", mute=mute) του **Corelistener** και προωθεί το event οπότε ενημερώνεται το **frontend** για την αλλαγή της κατάστασης σίγασης.

ΚΕΦΑΛΑΙΟ 4:
Ανάλυση Αποτελεσμάτων – Συζήτηση

4.1 Κόστος υλικών

Στον παρακάτω πίνακα περιγράφονται τα υλικά που χρησιμοποιηθήκανε για την υλοποίηση του μουσικού streamer. Υπενθυμίζουμε ότι ό,τι λογισμικό χρησιμοποιήθηκε είναι ανοιχτού κώδικα οπότε έχει μηδενικό κόστος.

Πίνακας 7: Κόστος υλικών υλοποίησης

Υλικά		**Κόστος (€)**
Raspberry Pi 400		76.50
Raspberry Pi 4 Official μετασχηματιστής στα 3A (15,3W) USB-C		8,90
Integral microSDHC memory card 16 GB MSDHC UHS-1 U1 CL10 V10 A1 with adapter (Raspberry Pi recommended)		7,90
40 Pin GPIO Male to Female Ribbon Cable		2.0
Raspberry Pi IQaudio DAC+		20.90
Συνολικό κόστος		**116.2**

Από τον παραπάνω πίνακα αντιλαμβανόμαστε ότι με 116.2 ευρώ καταφέραμε να κατασκευάσουμε ένα πλήρως λειτουργικό μουσικό streamer, με δυνατότητα εκτέλεσης μουσικής από μια πληθώρα μουσικών πηγών, σε υψηλή ποιότητα.

4.2 Θετικά, αρνητικά στοιχεία και δυσκολίες

Κατά την υλοποίηση και χρήση του μουσικού streamer εντοπίσαμε θετικά αλλά και κάποια αρνητικά στοιχεία και δυσκολίες τα οποία περιγράφονται παρακάτω :

4.2.1 Θετικά στοιχεία

Μπορεί να υλοποιηθεί ως headless σύστημα με έλεγχο απομακρυσμένα με ένα SSH client, με web ή MPD clients ή με εφαρμογή για έξυπνο κινητό τηλέφωνο. Οπότε δεν χρειάζεται απαραίτητα οθόνη, με συνέπεια να μπορούμε να απενεργοποιήσουμε το γραφικό περιβάλλον του Raspberry Pi OS, εξοικονομώντας πόρους του συστήματος, και αυτό μας επιτρέπει να χρησιμοποιήσουμε και παλαιότερα μοντέλα Raspberry Pi. Βέβαια υπάρχει η δυνατότητα να υλοποιηθεί με οθόνη υπολογιστή ή LCD display και να ελέγχεται τοπικά.

Εκτελώντας το Mopidy με οποιοδήποτε client, αντιλαμβανόμαστε ότι μας παρέχει εύκολη αναζήτηση μουσικών κομματιών από όλες τις διαθέσιμες πηγές ή από μεμονωμένες πηγές. Επίσης μας παρέχει δυνατότητα όλων των βασικών λειτουργιών ελέγχου όπως σε ένα συμβατικό ηχοσύστημα, δηλαδή δυνατότητα έναρξης αναπαραγωγής, παύσης, διακοπής, προηγούμενο ή επόμενο κομμάτι, επανάληψη κομματιού, τυχαίας αναπαραγωγής, σίγαση ήχου, έλεγχος στάθμης ήχου από την εφαρμογή και στην περίπτωση του κινητού, ακόμα και από τα κουμπιά ρύθμισης έντασης ήχου του κινητού.

Είναι εύκολα επεκτάσιμο, και μπορούν να προστεθούν νέες επεκτάσεις για αναπαραγωγή μουσικής από άλλες πηγές, καθώς και εύκολη προσθήκη νέων clients. Με την εκτέλεση μιας εντολής στο τερματικό και έπειτα από την απαραίτητη επανεκκίνηση του Mopidy, μπορούμε να προσθαφαιρούμε επεκτάσεις και clients προσαρμόζοντας το σύστημα στις ανάγκες μας, χωρίς να κάνουμε εκ νέου εγκατάσταση του Mopidy από την αρχή. Για παράδειγμα τρέχοντας απλά την εντολή :

```
sudo python3 -m pip install Mopidy-Bandcamp
```

και την επανεκκίνηση του Mopidy, προσθέσαμε την δυνατότητα αναπαραγωγής μουσικής από το Bandcamp και ούτω καθεξής.

Ένα άλλο θετικό της χρήσης του Mopidy είναι ότι υποστηρίζεται η προβολή μεταδεδομένων, δηλαδή πληροφορίες ενσωματωμένες στο κομμάτι, κατά την

αναπαραγωγή της μουσικής, όπως εξώφυλλα δίσκων, όνομα καλλιτέχνη, τίτλο τραγουδιού, όνομα δίσκου κλπ.

Ακόμα έγινε συνειδητά στιγμιαία διακοπή της σύνδεσης στο διαδίκτυο και παρατηρήθηκε ότι η αναπαραγωγή δεν διακόπηκε μέχρι να επανέλθει η σύνδεση, γιατί γίνεται buffering οπότε η αναπαραγωγή δεν επηρεάζεται από προσωρινή διακοπή ίντερνετ ή από διακυμάνσεις στην ποιότητά του. Μόνο στην περίπτωση που η διακοπή διήρκησε αρκετά διακόπηκε η αναπαραγωγή του τραγουδιού.

Επίσης παρέχεται η δυνατότητα ταυτόχρονου ελέγχου του Mopidy από διαφορετικές συσκευές, ακόμα και από διαφορετικούς clients. Αυτό μπορεί να είναι χρήσιμο αν θέλουν να χρησιμοποιούν ταυτόχρονα το Mopidy δύο ή περισσότερα άτομα και να επιλέγουν τα κομμάτια που επιθυμούν να ακούσουν.

Κάτι άλλο που παρατηρήθηκε είναι ότι αν για κάποιο λόγο κλείσει ο client, η αναπαραγωγή συνεχίζεται κανονικά και αν ξανανοίξουμε τον ίδιο client ή ακόμα και κάποιον άλλο, μπορούμε να πάρουμε πάλι τον έλεγχο αναπαραγωγής, κανονικά.

Πολύ χρήσιμο χαρακτηριστικό του Mopidy είναι η υποστήριξη Zeroconf, με το οποίο μπορούμε αντί να χρειάζεται να γνωρίζουμε την IP του Raspberry Pi ώστε να συνδεθεί ο client στο Mopidy, μπορούμε να χρησιμοποιήσουμε το hostname του Raspberry όπως ακριβώς θα χρησιμοποιούσαμε την IP του.

4.2.2 Αρνητικά σημεία και δυσκολίες

Όσον αφορά τα αρνητικά σημεία και τις δυσκολίες που παρουσιάστηκαν κατά την υλοποίηση και λειτουργία του μουσικού streamer, εντοπίστηκαν τα εξής :

Απαιτείται χρήση εντολών Linux και χρήση τερματικού, πράγμα το οποίο μπορεί να είναι αποτρεπτικός παράγοντας για χρήστες που έχουν συνηθίσει σε λειτουργικά όπως τα Windows. Βέβαια αν κάποιος ακολουθήσει πιστά τις οδηγίες που παρέχονται στη σελίδα του Mopidy [125] ή στο πιο πρόσφατο manual του Mopidy [122] καθώς επίσης και στην ιστοσελίδα Mopidy Discourse [148] όπου συζητιούνται ζητήματα σχετικά με το Mopidy, θα συνειδητοποιήσει ότι η διαδικασία εγκατάστασης είναι αρκετά απλή.

Το ίδιο ισχύει και για τις επεκτάσεις και τους clients του Mopidy, τα οποία εγκαθίστανται μόνο με τη χρήση εντολών τερματικού. Αλλά και πάλι στην αντίστοιχη σελίδα στο GitHub

για κάθε επέκταση και client παρέχονται όλες οι πληροφορίες για την εγκατάστασή τους, καθώς και την παραμετροποίησή τους.

Όσον αφορά τις επεκτάσεις, ένα αρνητικό σημείο είναι ότι μπορεί ανά πάσα στιγμή να γίνουν μη λειτουργικές, λόγω του ότι τα άτομα που τις κατασκευάζουν μπορεί να σταματήσουν να παρέχουν υποστήριξη και συντήρηση ή για άλλους λόγους όπως για παράδειγμα στην περίπτωση της λειτουργίας Spotify, όπου λόγω του ότι η εταιρία κατάργησε τη βιβλιοθήκη libspotify την οποία οι προγραμματιστές χρησιμοποιούσαν για να αλληλεπιδρούν με το Spotify και να μεταφέρουν (streaming) ήχο από αυτό και στην οποία στηρίζεται η επέκταση Mopidy-Spotify, η επέκταση αυτή βγήκε εκτός λειτουργίας, οπότε μέχρι να βρεθεί εναλλακτική λύση, είναι αδύνατη η αναπαραγωγή μουσικής του Spotify από το Mopidy. [119]

Ένα σημείο που χρήζει προσοχής είναι ότι ενώ τα εξωτερικά μέσα αποθήκευσης αναγνωρίζονται αυτόματα από το λειτουργικό Raspberry Pi OS και μπορεί κάποιος να διαβάσει και να γράψει σε αυτά χωρίς να χρειαστεί κάποια ρύθμιση, για να μπορεί ο χρήστης του Mopidy να έχει πρόσβαση σε αυτά τα μέσα, θα πρέπει μέσω εντολής τερματικού να δώσει πλήρη δικαιώματα χρήστη όσον αφορά την εγγραφή και την ανάγνωση σε αυτά.

Ένα άλλο αρνητικό που εντοπίζεται είναι ότι οποιαδήποτε αλλαγή στο αρχείο ρυθμίσεων θα πρέπει να γίνεται μέσω εντολών τερματικού, όπως επίσης η διακοπή ή επανεκκίνηση του Mopidy server καθώς και η σάρωση και ανανέωση των αρχείων που βρίσκονται σε τοπικά μέσα αποθήκευσης. Με εντολές από τη γραμμή εντολών γίνεται και η επανεκκίνηση ή η διακοπή λειτουργίας του Raspberry Pi, εάν δεν θέλουμε να αφαιρέσουμε την τροφοδοσία ρεύματος. Στους clients δεν υπάρχει κάποια επιλογή να γίνονται αυτές οι διεργασίες με το πάτημα ενός κουμπιού. Όσον αφορά τη σάρωση των τοπικών μέσων αποθήκευσης όσον αφορά τη χρήση της επέκτασης Mopidy-Local, χρειάστηκε να υλοποιήσουμε μία διεργασία cron, που να εκτελεί σάρωση κάθε ένα λεπτό.

Τέλος, επειδή η εγκατάσταση του λογισμικού γίνεται σε μία SD κάρτα, ελλοχεύει ο κίνδυνος κάποια στιγμή η κάρτα αυτή να καταστραφεί, είτε από τις πολλές αναγνώσεις/εγγραφές, είτε από τις απότομες διακοπές τροφοδοσίας ρεύματος.

4.3 Ορθότητα λειτουργίας

Όσον αφορά την ορθότητα λειτουργίας του, έγιναν δοκιμές με τοπικά αρχεία διαφόρων format και bitrate, απομακρυσμένους εξυπηρετητές και internet radio. Από τις δοκιμές αυτές και για κάθε διαφορετική πηγή παρατηρήσαμε ότι :

4.3.1 Τοπικά αποθηκευτικά μέσα

Στην περίπτωση της αναπαραγωγής μουσικής από τοπικά αποθηκευτικά μέσα, δοκιμάστηκε η αναπαραγωγή διαφόρων αρχείων μουσικής, με διάφορους codecs και bitrate. Χρησιμοποιήθηκαν αποθηκευτικά μέσα διαμορφωμένα σε FAT 32 και NTFS και αναγνωρίστηκαν από το σύστημα με απόλυτη επιτυχία αφού πρώτα δώσαμε πλήρη δικαιώματα χρήστη όσον αφορά την εγγραφή και την ανάγνωση σε αυτά με τη χρήση της εντολής sudo chmod 777 /media/pi. Παρακάτω αναφέρονται οι διάφοροι τύποι αρχείων που δοκιμάστηκαν και η συμπίεση που έχουν υποβληθεί, το bitrate τους και αν υποστηρίζονταν τα μεταδεδομένα τους.

Πίνακας 8: Αρχεία ήχου που αναπαράχθηκαν επιτυχώς

Τύπος αρχείου		Bitrate (Kbps)	Μεταδεδομένα
mp3		320	Ναι
Aiff		1411	Ναι
Flac		930	Ναι
m4a (aac)		256	Ναι
Aac		130	Ναι
Alac		1411	Ναι
Ogg		160	Ναι
Wma		96	Ναι
Wav		1411	Ναι
Opus		1411	Ναι

4.3.2 Απομακρυσμένοι εξυπηρετητές

Στον παρακάτω πίνακα αναφέρονται οι απομακρυσμένοι εξυπηρετητές που δοκιμάσαμε, καθώς και το bitrate τους.

Πίνακας 9: Απομακρυσμένοι εξυπηρετητές που αναπαράχθηκαν επιτυχώς

Πηγή		Bitrate (Kbps)
Youtube		128
Bandcamp		128
TuneIn		-
InternetArchive		64

4.3.3 Internet Radio

Για να δοκιμάσουμε την αναπαραγωγή μουσικής από Internet radio, έγιναν δοκιμές με URL σταθμών από το Radio Paradise [149] [150] το οποίο είναι ραδιοφωνικός σταθμός που παρέχει streams υψηλής ποιότητας, συμπιεσμένα με διάφορα codecs και σε bitrate 320 kbps. Στην περίπτωση αναπαραγωγής του FLAC και του Ogg Vorbis αναπαρήγαγε και μεταδεδομένα.

Πίνακας 10: Διαδικτυακοί σταθμοί που αναπαράχθηκαν επιτυχώς

Όνομα σταθμού	URL	Codec	Bitrate (Kbps)
Rock Mix	https://stream.radioparadise.com/rock-320	AAC	320
Rock Mix	http://stream.radioparadise.com/rock-flacm	FLAC	1411
Main Mix	https://stream.radioparadise.com/mp3-320	MP3	320
Ogg Vorbis	http://stream-dc1.radioparadise.com/rp_192m.ogg	Ogg	192

4.4 Σύγκριση των interfaces

Για τον έλεγχο του Mopidy χρησιμοποιήθηκαν οι Iris και Music Box οι οποίοι είναι web clients και ο Mopidy-Mobile που είναι μία εφαρμογή Android για έξυπνα κινητά τηλέφωνα. Όλοι οι clients που δοκιμάστηκαν είναι εύκολοι στη χρήση και πλήρως λειτουργικοί όσον αφορά τον έλεγχο της αναπαραγωγής και την αναζήτηση μουσικής μέσω των παρεχόμενων μουσικών πηγών. Από τους τρεις αυτούς clients, ο πιο πρακτικός είναι ο Mopidy-Mobile οποίος εκτελεί όλες τις απαραίτητες λειτουργίες μιας μουσικής εφαρμογής, ακόμα και αν η συσκευή είναι κλειδωμένη, δίνει τη δυνατότητα αυξομείωσης της έντασης του ήχου από τα κουμπιά που χρησιμοποιούνται για την αυξομείωση του ήχου του κινητού τηλεφώνου, έχει τη δυνατότητα να βρίσκει αυτόματα την IP του Mopidy Server μέσω του Zeroconf, αλλά παρατηρήθηκε ότι δεν μπορεί ο χρήστης να του εισάγει URL ραδιοφωνικού σταθμού και να τον αναπαράγει. Ένα άλλο σημαντικό πλεονέκτημα είναι ότι κατά τη διάρκεια της ακρόασης, μπορούν να πραγματοποιηθούν κανονικά τηλεφωνικές κλήσεις, χωρίς να επηρεαστεί η ακρόαση. Ο client Iris έχει το πιο εντυπωσιακό και φιλικό προς τον χρήστη περιβάλλον, τα άλμπουμ και οι φάκελοι απεικονίζονται με μεγάλα εικονίδια που απεικονίζουν τα εξώφυλλα των δίσκων και παρέχει τη δυνατότητα να μεταφέρει (streaming) ήχο σε πολλαπλές συσκευές μέσω του Snapcast. Τέλος ο client Music Box είναι πιο απλοϊκός σε σχέση με τον Iris, και για την απεικόνιση των επιλογών μουσικής χρησιμοποιεί λίστες αντί για εικονίδια.

ΚΕΦΑΛΑΙΟ 5:
Συμπεράσματα – Προτάσεις

5.1 Συμπεράσματα

Από τα παραπάνω συμπεραίνουμε πως με ένα Raspberry Pi και την εγκατάσταση του λογισμικού ανοιχτού κώδικα Mopidy μπορούμε να υλοποιήσουμε ένα μουσικό streamer με χαμηλό κόστος και να επιτύχουμε αναπαραγωγή υψηλής ποιότητας. Λόγω του ότι το Mopidy τρέχει στο παρασκήνιο, το Raspberry Pi μπορεί να χρησιμοποιείται παράλληλα και για άλλες εργασίες όπως σερφάρισμα στο internet, διάφορες εφαρμογές και αυτοματισμούς μέσω το GPIO. Άλλες υλοποιήσεις όπως πχ το Volumio [151] χρησιμοποιούν αποκλειστικά το Raspberry Pi για αναπαραγωγή μουσικής, δεσμεύοντάς το για συγκεκριμένη χρήση, μην επιτρέποντας τη χρήση του Raspberry Pi για άλλους σκοπούς.

Στην υλοποίησή μας χρησιμοποιήσαμε το Raspberry Pi 400, το οποίο είναι το πιο ακριβό Raspberry Pi μοντέλο. Μπορούμε κάλλιστα να χρησιμοποιήσουμε οποιοδήποτε μοντέλο Raspberry Pi, ακόμα και τα πρώτα μοντέλα, μειώνοντας το κόστος υλοποίησης. Για να μειώσουμε ακόμα περισσότερο το κόστος, μπορούμε να μη χρησιμοποιήσουμε εξωτερικό DAC, αξιοποιώντας την στερεοφωνική έξοδο ήχου 3,5 mm σε συνδυασμό με απλά ηχεία υπολογιστή, με αντίκτυπο όμως στην ποιότητα του παραγόμενου ήχου. Αυτή η υλοποίηση μπορεί να επιτευχθεί με όλα τα παλαιότερα μοντέλα Raspberry Pi, εκτός από το Raspberry Pi 400 το οποίο δεν διαθέτει στερεοφωνική έξοδο ήχου 3,5 mm.

Επίσης έχει το πλεονέκτημα ότι μπορεί κάποιος να επιλέξει από μια πληθώρα clients οι οποίοι μπορούν να λειτουργούν ταυτόχρονα δίνοντας τη δυνατότητα ελέγχου σε περισσοτέρων του ενός χρηστών, ή ακόμα να κατασκευάσει το δικό του client ανάλογα με τις προτιμήσεις του. Το λογισμικό που χρησιμοποιήσαμε, το Mopidy, είναι δωρεάν, κατασκευασμένο με Python, μια από τις ευκολότερες και φιλικότερες γλώσσες προγραμματισμού οπότε μπορεί κάποιος να κατανοήσει τον τρόπο λειτουργίας του, και να πειραματιστεί ώστε να κάνει τις δικές του διορθώσεις και τροποποιήσεις, αφού είναι ανοιχτού κώδικα. Επίσης διαπιστώσαμε ότι αναπαράγει μουσική από σχεδόν όλα τα γνωστά φορμάτ ήχου, σε μέγιστη ανάλυση. Πολύ ικανοποιητική ήταν και η αναπαραγωγή διαδικτυακών ραδιοφωνικών σταθμών, ακόμα και σε ποιότητα FLAC και Ogg Vorbis.

Η υλοποίηση είναι πολύ μικρή σε διαστάσεις, καθιστώντας τη φορητή και μπορεί κάποιος εύκολα να τη μεταφέρει σε άλλο δωμάτιο ή ακόμα και σε άλλο σπίτι. Επειδή η διαδικασία δεν είναι ιδιαίτερα χρονοβόρα, μπορεί ο χρήστης να εγκαταστήσει το Mopidy σε περισσότερες από μια κάρτες microSD, και να έχει εφεδρεία σε περίπτωση βλάβης της κάρτας. Και λόγω του ότι το κόστος υλοποίησης είναι χαμηλό, μπορεί να κατασκευάσει περισσότερους streamers για κάθε δωμάτιο του σπιτιού.

Μέσω του GPIO μπορεί κάποιος να συνδέσει οθόνη LCD ή και κουμπιά να χειρίζεται τις βασικές λειτουργίες αναπαραγωγής καθώς και να μπορεί να αυξομειώνει την ένταση του ήχου.

5.2 Προτάσεις βελτίωσης

Ως προτάσεις βελτίωσης θα μπορούσαμε να προτείνουμε διάφορες προτάσεις όπως :

Την χρήση ενός καλύτερου DAC το οποίο να αποδίδει υψηλότερης πιστότητας ήχο με ψηφιακές εξόδους και σε συνδυασμό με ένα καλό ηχοσύστημα. Επιπλέον να υπάρχει η δυνατότητα ώστε να μπορεί ο χρήστης να εκτελεί διεργασίες που αφορούν το Mopidy και το Raspberry Pi από το γραφικό περιβάλλον των clients αντί να χρησιμοποιεί τη γραμμή εντολών τερματικού. Για παράδειγμα να μπορεί να τροποποιεί το αρχείο ρυθμίσεων, να υπάρχει η δυνατότητα να μπορεί να επιλέγει και να προσθέτει διαθέσιμες επεκτάσεις, να μπορεί να διακόπτει και να επανεκκινεί το Mopidy και να επανεκκινεί ή να απενεργοποιεί το Raspberry Pi. Επίσης να δημιουργηθούν ακόμα περισσότερες νέες επεκτάσεις για υποστήριξη ακόμα περισσότερων πηγών μουσικής και να κατασκευαστούν clients με λειτουργίες υποστήριξης για ηλικιωμένα άτομα και άτομα που δεν βλέπουν καλά, και να περιλαμβάνουν μεγάλα κουμπιά με τις βασικές λειτουργίες αναπαραγωγής, παράλληλα με φωνητικές εντολές. Τέλος θα μπορούσε η εγκατάσταση του Mopidy να γίνει πιο αυτοματοποιημένη, για παράδειγμα με την εκτέλεση μιας μόνο εντολής.

Συνοψίζοντας, καταλήγουμε ότι η υλοποίηση ενός μουσικού streamer με τη χρήση ενός υλισμικού Raspberry Pi και λογισμικού ανοιχτού κώδικα και συγκεκριμένα του Mopidy, είναι ένας πολύ βολικός τρόπος να ακούει κάποιος την αγαπημένη του μουσική από πληθώρα πηγών. Ακόμα παρέχει δυνατότητες παραμετροποίησης της λειτουργικότητάς του και μπορεί να απευθυνθεί σε ερασιτέχνες και επαγγελματίες ηλεκτρονικούς, μουσικόφιλους και προγραμματιστές.

Η δημιουργία του συγκεκριμένου λογισμικού (που είναι ανοιχτού κώδικα) έχει προκύψει από τη συμβολή πολλών ατόμων ανά την υφήλιο, επομένως δίνει τη δυνατότητα τόσο στην εκμάθηση αλγοριθμικών τεχνικών όσο και στη βελτίωση του με την κατασκευή νέων βελτιωμένων αλγορίθμων. Δηλαδή δίνει τη δυνατότητα τροποποίησης και βελτίωσης των ήδη υπαρχουσών επεκτάσεων και clients και να κατασκευής νέων επεκτάσεων με τη βοήθεια του cookie cutter [152]. Τέλος, οι δυνατότητες πολυεπίπεδης δημιουργικής ενασχόλησης-βελτίωσης τόσο σε επίπεδο λογισμικού όσο και υλισμικού, δίνει δυνατότητες εξατομίκευσης της λειτουργικότητας του streamer, ώστε να μπορούν να δημιουργηθούν εξατομικευμένες εμπειρίες ακρόασης.

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΠΗΓΕΣ

[1] "What is a Music Server?", *Audionexus.com*, 2022. [Online]. Available: https://www.audionexus.com/music_servers.shtml. [Accessed: 04- May- 2022].

[2] "YouTube", *Youtube.com*. [Online]. Available: https://www.youtube.com/. [Accessed: 04- May- 2022].

[3] "Spotify", *Spotify*. [Online]. Available: https://open.spotify.com/. [Accessed: 04- May- 2022].

[4] "Best Music Streamers of 2022 | The Master Switch", *Themasterswitch.com*, 2022. [Online]. Available: https://www.themasterswitch.com/best-music-streamers. [Accessed: 04- May- 2022].

[5] "5 Reasons Why You Must Own A Music Server In Your Smart Home", *Home-automate.co.uk*, 2022. [Online]. Available: https://home-automate.co.uk/blog/five-reasons-music-server-smart-home. [Accessed: 04- May- 2022].

[6] Raspberry Pi Ltd, "Raspberry Pi 400 ", datasheet, Jan. 2022

[7] "IQaudio product guide - raspberry pi." [Online]. Available: https://datasheets.raspberrypi.com/iqaudio/iqaudio-product-brief.pdf. [Accessed: 04-May-2022].

[8] "40 pin GPIO male to female ribbon cable - 200mm," *nettop.gr*. [Online]. Available: https://nettop.gr/index.php/hlektronika/prototyping/gpio-connectors-cables/40-pin-gpio-

[9] *Mopidy*. [Online]. Available: https://mopidy.com/. [Accessed: 05-May-2022].

[10] "Welcome to Python.org," *Python.org*. [Online]. Available: https://www.python.org/. [Accessed: 04-May-2022].

[11] *What is client-server architecture?* [Online]. Available: https://www.w3schools.in/what-is-client-server-architecture/. [Accessed: 04-May-2022].

[12] *What is gstreamer?* [Online]. Available: https://gstreamer.freedesktop.org/documentation/application-development/introduction/gstreamer.html?gi-language=python. [Accessed: 04-May-2022].

[13] *Bandcamp*. [Online]. Available: https://bandcamp.com/. [Accessed: 04-May-2022].

[14] "TuneIn brings together live sports, music, news, and podcasts - hear what matters most to you!," *TuneIn*. [Online]. Available: https://tunein.com/. [Accessed: 04-May-2022].

[15] "Internet archive: Digital Library of Free & Borrowable Books, movies, Music & Wayback Machine," *Internet Archive: Digital Library of Free & Borrowable Books, Movies, Music & Wayback Machine*. [Online]. Available: https://archive.org/. [Accessed: 04-May-2022].

[16] Techopedia, “What is internet radio? - definition from Techopedia,” *Techopedia.com*, 09-Dec-2016. [Online]. Available: https://www.techopedia.com/definition/3108/internet-radio. [Accessed: 04-May-2022].

[17] "History of sound recording - Wikipedia", *En.wikipedia.org*, 2022. [Online]. Available: https://en.wikipedia.org/wiki/History_of_sound_recording. [Accessed: 04- May- 2022].

[18] "1860 'Phonautograph' Is Earliest Known Recording", *Npr.org*, 2022. [Online]. Available: https://www.npr.org/templates/story/story.php?storyId=89380697. [Accessed: 04- May- 2022].

[19] “File:Scott phonautograph.png - Wikimedia Commons.” [Online]. Available: https://commons.wikimedia.org/wiki/File:Scott_phonautograph.png. [Accessed: 04-May-2022].

[20] R. Stross, "PHONOGRAPH The Incredible Talking Machine", *TIME.com*, 2010. [Online]. Available: http://content.time.com/time/specials/packages/article/0,28804,1999143_1999210,00.html. [Accessed: 04- May- 2022].

[21] "Inventing Sound Recording", *Edison.rutgers.edu*. [Online]. Available: https://edison.rutgers.edu/life-of-edison/innovation-series/tinfoil-phonograph/inventing-sound-recording. [Accessed: 04- May- 2022].

[22] R. Beardsley and D. Leech-Wilkinson, "A Brief History of Recording to ca. 1950", *Charm.rhul.ac.uk*. [Online]. Available: https://www.charm.rhul.ac.uk/history/p20_4_1.html. [Accessed: 04- May- 2022].

[23] “File:Edison and phonograph edit1.jpg - Wikimedia Commons.” [Online]. Available: https://commons.wikimedia.org/wiki/File:Edison_and_phonograph_edit1.jpg. [Accessed: 28-May-2022].

[24] “Charm,” *A Brief History of Recording to ca. 1950*, 27-May-2022. [Online]. Available: https://www.charm.rhul.ac.uk/history/p20_4_1.html. [Accessed: 04-May-2022].

[25] “File: Berliner record.jpg - Wikimedia Commons.” [Online]. Available: https://commons.wikimedia.org/wiki/File:Berliner_record.jpg. [Accessed: 04-May-2022].

[26] "History of sound recording - Wikipedia", *En.wikipedia.org*. [Online]. Available: https://en.wikipedia.org/wiki/History_of_sound_recording#Electrical_recording. [Accessed: 04- May- 2022].

[27] "History of sound recording - Wikipedia", *En.wikipedia.org*. [Online]. Available: https://en.wikipedia.org/wiki/History_of_sound_recording#Magnetic_recording. [Accessed: 04- May- 2022].

[28] “File:compact audio cassette 4.jpg - wikimedia commons.” [Online]. Available: https://commons.wikimedia.org/wiki/File:Compact_audio_cassette_4.jpg. [Accessed: 04-May-2022].

[29] “File:DavidOliverTKO.jpg - wikimedia commons.” [Online]. Available: https://commons.wikimedia.org/wiki/File:DavidOliverTKO.jpg. [Accessed: 04-May-2022].

[30] “File:12in-Vinyl-LP-Record-Angle.jpg - Wikimedia Commons.” [Online]. Available: https://commons.wikimedia.org/wiki/File:12in-Vinyl-LP-Record-Angle.jpg. [Accessed: 04-May-2022].

[31] "History of sound recording - Wikipedia", *En.wikipedia.org*. [Online]. Available: https://en.wikipedia.org/wiki/History_of_sound_recording#The_Digital_Era_(1975%E2%80%93present). [Accessed: 04- May- 2022].

[32] “File:blank cd.png - Wikimedia Commons.” [Online]. Available: https://commons.wikimedia.org/wiki/File:Blank_cd.png. [Accessed: 04-May-2022].

[33] "What is Internet Radio? - Definition from Techopedia", *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/3108/internet-radio. [Accessed: 04- May- 2022].

[34] "Stream and listen to music online for free with SoundCloud", *SoundCloud*. [Online]. Available: https://soundcloud.com/. [Accessed: 04- May- 2022].

[35] "Spotify Business Model", *Strategyzer.com*. [Online]. Available: https://www.strategyzer.com/business-model-examples/spotify-business-model. [Accessed: 04- May- 2022].

[36] “Spotify premium,” *Spotify*. [Online]. Available: https://www.spotify.com/us/premium/. [Accessed: 04-May-2022].

[37] “YouTube Premium,” *YouTube*. [Online]. Available: https://www.youtube.com/premium. [Accessed: 04-May-2022].

[38] “IFPI Global Music Report 2022,” *IFPI GLOBAL MUSIC REPORT 2022*. [Online]. Available: https://globalmusicreport.ifpi.org/. [Accessed: 04-May-2022].

[39] D. Gross, “Spotify and pandora don't pay off, musicians say,” *CNN*, 13-Nov-2014. [Online]. Available: https://edition.cnn.com/2014/11/12/tech/web/spotify-pay-musicians/index.html. [Accessed: 04-May-2022].

[40] J. Somers, “Musicians who've demanded their music be removed from streaming sites” *Grunge.com*, 14-Feb-2022. [Online]. Available: https://www.grunge.com/768284/musicians-whove-demanded-their-music-be-removed-from-streaming-sites/. [Accessed: 04-May-2022].

[41] Luis Aguiar & Joel Waldfogel, 2015.

"Streaming Reaches Flood Stage: Does Spotify Stimulate or Depress Music Sales?,"

NBER Working Papers 21653, National Bureau of Economic Research, Inc.

[42] M. Roomi, “4 advantages and disadvantages of bluetooth: Drawbacks and benefits of Bluetooth,” HitechWhizz, 26-Mar-2020. [Online]. Available: https://www.hitechwhizz.com/2020/03/4-advantages-and-disadvantages-drawbacks-benefits-of-bluetooth.html. [Accessed: 04-May-2022].

[43] “What is the maximum range of a bluetooth connection?,” *Samsung levant*, 22-Sep-2020. [Online]. Available: https://www.samsung.com/levant/support/mobile-devices/what-is-the-maximum-range-of-a-bluetooth-connection/. [Accessed: 04-May-2022].

[44] Infographic Overview – Bluetooth Technology See how the global standard for simple, “Bluetooth technology overview,” *Bluetooth® Technology Website*, 2022. [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/. [Accessed: 04-May-2022].

[45] “What causes bluetooth interference in your device?,” *What Causes Bluetooth Interference in Your Device?* [Online]. Available: https://www.audio-technica.com/en-us/support/audio-solutions-question-of-the-week-what-causes-bluetooth-interference/. [Accessed: 04-May-2022].

[46] A. T. Editor, “Advantages and disadvantages of bluetooth,” *Polytechnic Hub*, 29-Apr-2017. [Online]. Available: https://www.polytechnichub.com/advantages-disadvantages-bluetooth/. [Accessed: 04-May-2022].

[47] “Reporting security vulnerabilities,” *Bluetooth® Technology Website*, 2022. [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/reporting-security/. [Accessed: 04-May-2022].

[48] Bekaroo, Girish & Santokhee, Aditya, “Power consumption of the Raspberry Pi: A comparative analysis“. 361-366. 10.1109/EmergiTech.2016.7737367, 2016 [Online]. Available: https://www.researchgate.net/publication/309917878_Power_consumption_of_the_Raspberry_Pi_A_comparative_analysis

[49] J. Park, “Analog knobs on Raspberry Pi 400 with Cyberdeck Hat,” *Adafruit Learning System*. [Online]. Available: https://learn.adafruit.com/analog-knobs-on-raspberrypi-400-with-cyberdeck-hat. [Accessed: 04-May-2022].

[50] Heron et al.: Open source and accessibility: advantages and limitations. Journal of Interaction Science 2013 1:2

[51] Morgan, Lorraine & Finnegan, Patrick, “Benefits and Drawbacks of Open Source Software: An Exploratory Study of Secondary Software Firms“, conference paper, International Federation for Information Processing Digital Library; Open Source Development, Adoption and Innovation;. 234. 10.1007/978-0-387-72486-7_33. 2007 [Online]. Available: https://link.springer.com/content/pdf/10.1007%2F978-0-387-72486-7_33.pdf

[52] Western Digital, “White Paper: Western Digital Flash 101 and flash management“, 2018 [Online]. Available : https://documents.westerndigital.com/content/dam/doc-library/en_us/assets/public/western-digital/collateral/white-paper/white-paper-sandisk-flash101-management.pdf

[53] Raspberry Pi, *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/. [Accessed: 04-May-2022].

[54] "What is a Raspberry Pi?," *Raspberry Pi*, 20-Aug-2015. [Online]. Available: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/. [Accessed: 04-May-2022].

[55] Raspberry Pi, "Buy A raspberry pi 4 model B," *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b. [Accessed: 29-May-2022].

[56] S. Bush, "Dongle computer lets kids discover programming on a TV," *Electronics Weekly*, 25-May-2011. [Online]. Available: https://www.electronicsweekly.com/market-sectors/embedded-systems/dongle-computer-lets-kids-discover-programming-on-a-2011-05/. [Accessed: 04-May-2022].

[57] Raspberry Pi, "Buy A raspberry pi," *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/products/. [Accessed: 04-May-2022].

[58] "RPI hardwarehistory," *RPi HardwareHistory - eLinux.org*. [Online]. Available: https://elinux.org/RPi_HardwareHistory. [Accessed: 04-May-2022].

[59] "Raspberry Pi," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi#Raspberry_Pi. [Accessed: 04-May-2022].

[60] Raspberry Pi, "Raspberry pi 4 model B specifications," *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. [Accessed: 04-May-2022].

[61] Raspberry Pi, "Buy A raspberry pi zero," *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-zero/. [Accessed: 04-May-2022].

[62] Raspberry Pi, "Buy A raspberry pi zero W," *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-zero-w/. [Accessed: 04-May-2022].

[63] Raspberry Pi, "Raspberry Pi Zero 2 W," *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/. [Accessed: 04-May-2022].

[64] Raspberry Pi, "Raspberry pi pico," *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-pico/. [Accessed: 04-May-2022].

[65] "Nettop," *Computers & Electronics*. [Online]. Available: http://www.nettop.gr/. [Accessed: 04-May-2022].

[66] "Raspberry pi documentation," Raspberry Pi OS. [Online]. Available: https://www.raspberrypi.com/documentation/computers/os.html. [Accessed: 04-May-2022].

[67] "Raspberry Pi Gpio Pinout," Raspberry Pi GPIO Pinout. [Online]. Available: https://pinout.xyz/. [Accessed: 04-May-2022].

[68] "GPCLK at raspberry pi gpio pinout," GPCLK at Raspberry Pi GPIO Pinout. [Online]. Available: https://pinout.xyz/pinout/gpclk. [Accessed: 04-May-2022].

[69] Matt, “Simple guide to the raspberry pi GPIO header,” Raspberry Pi Spy, 09-Jun-2012. [Online]. Available: https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/. [Accessed: 04-May-2022].

[70] L. Pounder, “Raspberry Pi Gpio Pinout: What each pin does on PI 4, earlier models,” Tom's Hardware, 16-Jun-2020. [Online]. Available: https://www.tomshardware.com/reviews/raspberry-pi-gpio-pinout,6122.html. [Accessed: 04-May-2022].

[71] G. Halfacree, *The official Raspberry Pi Beginner's Guide: How to use your new computer*. Cambridge: Raspberry Pi Trading Ltd, 2020.

[72] “Raspberry pi os,” *Wikipedia*, 26-Apr-2022. [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi_OS. [Accessed: 04-May-2022].

[73] Raspberry Pi, “Operating system images,” *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/software/operating-systems/. [Accessed: 04-May-2022].

[74] “Raspbian ,” *Raspbian*. [Online]. Available: https://www.raspbian.org/. [Accessed: 04-May-2022].

[75] R. Kabade, “Raspberry pi gets lxde-based Pixel Desktop Environment,” *Open Source For You*, 30-Sep-2016. [Online]. Available: https://www.opensourceforu.com/2016/09/raspberry-pi-gets-lxde-based-pixel-desktop-environment/. [Accessed: 04-May-2022].

[76] peppe8o, “Raspberry pi OS lite VS desktop: Comparison between the 2 distributions,” *peppe8o*, 17-Sep-2019. [Online]. Available: https://peppe8o.com/raspberry-pi-os-lite-vs-desktop/. [Accessed: 04-May-2022].

[77] “Libreelec,” *LibreELEC*. [Online]. Available: https://libreelec.tv/. [Accessed: 04-May-2022].

[78] *OSMC*. [Online]. Available: https://osmc.tv/. [Accessed: 04-May-2022].

[79] “Download ubuntu desktop: Download,” *Ubuntu*. [Online]. Available: https://ubuntu.com/download/desktop. [Accessed: 04-May-2022].

[80] “Get ubuntu server: Download,” *Ubuntu*. [Online]. Available: https://ubuntu.com/download/server. [Accessed: 04-May-2022].

[81] “Ubuntu core,” *Ubuntu*. [Online]. Available: https://ubuntu.com/core. [Accessed: 04-May-2022].

[82] “Arch Linux Arm,” *Arch Linux ARM*. [Online]. Available: https://archlinuxarm.org/. [Accessed: 04-May-2022].

[83] Penthux.NET, “Sarpi Project - Slackware Arm on a Raspberry Pi,” *SARPi RSS Update*. [Online]. Available: https://sarpi.penthux.net/. [Accessed: 04-May-2022].

[84] “Retropie,” *RetroPie*. [Online]. Available: https://retropie.org.uk/. [Accessed: 04-May-2022].

[85] “The open source game console,” *Lakka*. [Online]. Available: https://www.lakka.tv/. [Accessed: 04-May-2022].

[86] “What is client-server? definition and faqs: HEAVY.AI,” *What is Client-Server? Definition and FAQs | HEAVY.AI*. [Online]. Available: https://www.omnisci.com/technical-glossary/client-server. [Accessed: 04-May-2022]

[87] *Client-server computing*. [Online]. Available: http://www.it.uom.gr/project/client_server/theoria1.htm. [Accessed: 04-May-2022].

[88] T. T. Contributor, “What is the client-server model? - definition from whatis.com,” *SearchNetworking*, 09-Nov-2020. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/client-server. [Accessed: 04-May-2022].

[89] “Client–server model,” *Wikipedia*, 29-Mar-2022. [Online]. Available: https://en.wikipedia.org/wiki/Client%E2%80%93server_model. [Accessed: 04-May-2022].

[90] “Frontend and backend,” *Wikipedia*, 16-Jan-2022. [Online]. Available: https://en.wikipedia.org/wiki/Frontend_and_backend. [Accessed: 04-May-2022].

[91] “Sam Newman - backends for frontends,” *Atom*. [Online]. Available: https://samnewman.io/patterns/architectural/bff/. [Accessed: 04-May-2022].

[92] Mayuresh, “What is client server architecture? an overview,” *Jigsaw Academy*, 30-Dec-2020. [Online]. Available: https://www.jigsawacademy.com/blogs/cyber-security/what-is-client-server-architecture/. [Accessed: 04-May-2022].

[93] M. Martin, “N tier(multi-tier), 3-tier, 2-tier architecture with example,” *Guru99*, 23-Apr-2022. [Online]. Available: https://www.guru99.com/n-tier-architecture-system-concepts-tips.html. [Accessed: 04-May-2022].

[94] “What is client-server? definition and faqs,” *What is Client-Server? Definition and FAQs | HEAVY.AI*. [Online]. Available: https://www.heavy.ai/technical-glossary/client-server. [Accessed: 04-May-2022].

[95] “Why is GStreamer written in C? Why not C++/Objective-C,” *GStreamer* . [Online]. Available: https://gstreamer.freedesktop.org/documentation/frequently-asked-questions/general.html?gi-language=python#why-is-gstreamer-written-in-c-why-not-cobjectivec. [Accessed: 04-May-2022].

[96] “What are the exact licensing terms for GStreamer and its plugins?,” *GStreamer*. [Online]. Available: https://gstreamer.freedesktop.org/documentation/. [Accessed: 04-May-2022].

[97] “GNU lesser general public license v2.1 - GNU Project - Free Software Foundation,” *[A GNU head]*. [Online]. Available: https://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html. [Accessed: 04-May-2022].

[98] “Modules,” *GStreamer*. [Online]. Available: https://gstreamer.freedesktop.org/modules/. [Accessed: 04-May-2022].

[99] *GStreamer*. [Online]. Available: https://gstreamer.freedesktop.org/modules/gstreamer.html. [Accessed: 04-May-2022].

[100] "Gstreamer base plug-ins," *GStreamer*. [Online]. Available: https://gstreamer.freedesktop.org/modules/gst-plugins-base.html. [Accessed: 04-May-2022].

[101] "GStreamer good plug-ins," *GStreamer*. [Online]. Available: https://gstreamer.freedesktop.org/modules/gst-plugins-good.html. [Accessed: 04-May-2022].

[102] "Gstreamer ugly plug-ins," *GStreamer*. [Online]. Available: https://gstreamer.freedesktop.org/modules/gst-plugins-ugly.html. [Accessed: 04-May-2022].

[103] "GStreamer bad plug-ins," *GStreamer*. [Online]. Available: https://gstreamer.freedesktop.org/modules/gst-plugins-bad.html. [Accessed: 04-May-2022].

[104] *Foundations*. [Online]. Available: https://gstreamer.freedesktop.org/documentation/application-development/introduction/basics.html?gi-language=python. [Accessed: 04-May-2022].

[105] *Overview*. [Online]. Available: https://gstreamer.freedesktop.org/documentation/additional/design/overview.html?gi-language=python. [Accessed: 04-May-2022].

[106] "Why is it called Python?," *General Python FAQ - Python 3.10.4 documentation*. [Online]. Available: https://docs.python.org/3/faq/general.html#why-is-it-called-python. [Accessed: 04-May-2022].

[107] "Python classes and objects," *GeeksforGeeks*, 10-Jun-2021. [Online]. Available: https://www.geeksforgeeks.org/python-classes-and-objects/. [Accessed: 04-May-2022].

[108] D. Kuhlman, A Python Book. Platypus Global Media, 2011.

[109] M. Berga, "Python vs java: Key differences and code examples," *Blog | Imaginary Cloud*, 18-Mar-2021. [Online]. Available: https://www.imaginarycloud.com/blog/python-vs-java/. [Accessed: 05-May-2022].

[110] Yang Li, "Object-Oriented Design with Python" [Online]. Available: https://home.cs.colorado.edu/~kena/classes/5448/f12/presentation-materials/li.pdf . [Accessed: 05-May-2022]

[111] *Pykka*. [Online]. Available: https://pykka.readthedocs.io/en/stable/. [Accessed: 05-May-2022].

[112] "Actor model," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Actor_model. [Accessed: 04-May-2022].

[113] *Welcome to the Apache Software Foundation!* [Online]. Available: https://www.apache.org/licenses/LICENSE-2.0. [Accessed: 15-May-2022].

[114] "Examples," *Examples - Pykka 3.0.2 documentation*. [Online]. Available: https://pykka.readthedocs.io/en/stable/examples/#mopidy-music-server. [Accessed: 05-May-2022].

[115] "Quickstart," *Quickstart - Pykka 3.0.2 documentation*. [Online]. Available: https://pykka.readthedocs.io/en/stable/quickstart/#rules-of-the-actor-model. [Accessed: 05-May-2022].

[116] "The actor implementations," *Quickstart - Pykka 3.0.2 documentation*. [Online]. Available: https://pykka.readthedocs.io/en/stable/quickstart/#the-actor-implementations. [Accessed: 05-May-2022].

[117] "Actor proxies," *Quickstart - Pykka 3.0.2 documentation*. [Online]. Available: https://pykka.readthedocs.io/en/stable/quickstart/#actor-proxies. [Accessed: 05-May-2022].

[118] "Traversable attributes on proxies," *Quickstart - Pykka 3.0.2 documentation*. [Online]. Available: https://pykka.readthedocs.io/en/stable/quickstart/#traversable-attributes-on-proxies. [Accessed: 05-May-2022].

[119] J. Brown, "Sunset of libspotify on May 16, 2022," *Spotify for Developers*. [Online]. Available: https://developer.spotify.com/community/news/2022/04/12/libspotify-sunset/. [Accessed: 20-May-2022].

[120] Mopidy, "Mopidy/Mopidy: Mopidy is an Extensible Music Server written in Python," *GitHub*. [Online]. Available: https://github.com/mopidy/mopidy. [Accessed: 05-May-2022].

[121] "Mopidy," *GitHub*. [Online]. Available: https://github.com/orgs/mopidy/people. [Accessed: 04-May-2022].

[122] S. M. Jodal and contributors, "Mopidy documentation." [Online]. Available: https://docs.mopidy.com/_/downloads/en/latest/pdf/. [Accessed: 15-May-2022].

[123] "Creative Commons License Deed," *Creative Commons - Attribution-ShareAlike 3.0 Unported - CC BY-SA 3.0*. [Online]. Available: https://creativecommons.org/licenses/by-sa/3.0/. [Accessed: 15-May-2022].

[124] S. M. Jodal, "10 years of Mopidy," *Mopidy*, 23-Dec-2019. [Online]. Available: https://mopidy.com/blog/2019/12/23/10y-of-mopidy/. [Accessed: 05-May-2022].

[125] *Mopidy*. [Online]. Available: https://docs.mopidy.com/en/latest/. [Accessed: 05-May-2022].

[126] "Architecture," *Architecture - Mopidy 3.3.0 documentation*. [Online]. Available: https://docs.mopidy.com/en/latest/api/architecture/. [Accessed: 05-May-2022].

[127] Pimusicbox, “Pimusicbox/Mopidy-musicbox-webclient: Web client for Mopidy Music Server and the Pi Musicbox,” *GitHub*. [Online]. Available: https://github.com/pimusicbox/mopidy-musicbox-webclient. [Accessed: 07-May-2022].

[128] Jaedb, “Jaedb/iris: Discover, explore and manage your music library across multiple sources with this beautiful web-based interface. Iris is a Mopidy frontend extension.,” *GitHub*. [Online]. Available: https://github.com/jaedb/iris. [Accessed: 07-May-2022].

[129] T. Kemmer, “Mopidy mobile - εφαρμογές στο google play,” *Google*. [Online]. Available: https://play.google.com/store/apps/details?id=at.co.kemmer.mopidy_mobile&hl=el&gl=US [Accessed: 08-May-2022].

[130] Natumbri, “Natumbri/mopidy-youtube: Mopidy extension for playing music from YouTube,” *GitHub*. [Online]. Available: https://github.com/natumbri/mopidy-youtube. [Accessed: 04-May-2022].

[131] Kingosticks, “Kingosticks/Mopidy-tunein: Mopidy extension for playing music from tunein,” *GitHub*. [Online]. Available: https://github.com/kingosticks/mopidy-tunein. [Accessed: 04-May-2022].

[132] Impliedchaos, “Impliedchaos/Mopidy-Bandcamp: Mopidy backend for Bandcamp,” *GitHub*. [Online]. Available: https://github.com/impliedchaos/mopidy-bandcamp. [Accessed: 04-May-2022].

[133] “Configuration,” *Mopidy*. [Online]. Available: https://mopidy-internetarchive.readthedocs.io/en/latest/config/. [Accessed: 04-May-2022].

[134] Mopidy, “Mopidy/Mopidy-Local: Mopidy extension for playing music from Your Local Music Archive,” *GitHub*. [Online]. Available: https://github.com/mopidy/mopidy-local. [Accessed: 02-May-2022].

[135] *Putty: A free SSH and telnet client*. [Online]. Available: https://www.chiark.greenend.org.uk/~sgtatham/putty/. [Accessed: 08-May-2022].

[136] *Putty FAQ*. [Online]. Available: https://www.chiark.greenend.org.uk/~sgtatham/putty/faq.html#faq. [Accessed: 08-May-2022].

[137] “Fing App,” *Fing*. [Online]. Available: https://www.fing.com/products/fing-app. [Accessed: 05-May-2022].

[138] Raspberry Pi, “Raspberry pi os,” *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.com/software/. [Accessed: 04-May-2022].

[139] Jaedb, “Getting started · jaedb/iris wiki,” *GitHub*. [Online]. Available: https://github.com/jaedb/Iris/wiki/Getting-started#installing . [Accessed: 04-May-2022].

[140] Mopidy, “Mopidy/tracklist.py at Develop · Mopidy/Mopidy,” *GitHub*, 07-May-2020. [Online]. Available:

https://github.com/mopidy/mopidy/blob/develop/mopidy/core/tracklist.py. [Accessed: 04-May-2022].

[141] Mopidy, "Mopidy/listener.py at 0c80dcab29f23d31d3489c99b89dc5fe3f8c087d · Mopidy/Mopidy," *GitHub*, 08-Nov-2019. [Online]. Available: https://github.com/mopidy/mopidy/blob/0c80dcab29f23d31d3489c99b89dc5fe3f8c087d/mopidy/core/listener.py. [Accessed: 04-May-2022].

[142] Mopidy, "Mopidy/playback.py at Develop · Mopidy/Mopidy," *GitHub*. [Online]. Available: https://github.com/mopidy/mopidy/blob/develop/mopidy/core/playback.py. [Accessed: 04-May-2022].

[143] Mopidy, "Mopidy/backend.py at Develop · Mopidy/Mopidy," *GitHub*. [Online]. Available: https://github.com/mopidy/mopidy/blob/develop/mopidy/backend.py. [Accessed: 04-May-2022].

[144] Mopidy, "Mopidy/actor.py at Develop · Mopidy/Mopidy," *GitHub*, 13-Jan-2022. [Online]. Available: https://github.com/mopidy/mopidy/blob/develop/mopidy/audio/actor.py. [Accessed: 04-May-2022].

[145] Mopidy, "Mopidy/mixer.py at Develop · Mopidy/Mopidy," *GitHub*, 08-Dec-2019. [Online]. Available: https://github.com/mopidy/mopidy/blob/develop/mopidy/core/mixer.py. [Accessed: 04-May-2022].

[146] Mopidy, "Mopidy/mixer.py at Develop · Mopidy/Mopidy," *GitHub*. [Online]. Available: https://github.com/mopidy/mopidy/blob/develop/mopidy/softwaremixer/mixer.py. [Accessed: 04-May-2022].

[147] Mopidy, "Mopidy/mixer.py at Develop · Mopidy/Mopidy," *GitHub*, 09-Nov-2019. [Online]. Available: https://github.com/mopidy/mopidy/blob/develop/mopidy/mixer.py. [Accessed: 04-May-2022].

[148] "Mopidy discourse," *Mopidy Discourse*. [Online]. Available: https://discourse.mopidy.com/. [Accessed: 16-May-2022].

[149] Radio Paradise, "Commercial free - listener supported," *Radio Paradise*. [Online]. Available: http://www.radioparadise.com/. [Accessed: 16-May-2022].

[150] Radio Paradise, "Stream Links", *Radio Paradise*. [Online]. Available: https://radioparadise.com/listen/stream-links . [Accessed: 16-May-2022].

[151] "Volumio - The Audiophile Music Player." https://volumio.com/ (accessed: May 20, 2022).

[152] Mopidy, "Mopidy/Cookiecutter-mopidy-ext: Cookiecutter template for creating a Mopidy extension," *GitHub*. [Online]. Available: https://github.com/mopidy/cookiecutter-mopidy-ext. [Accessed: 10-May-2022].

ΠΑΡΑΡΤΗΜΑΤΑ

ΠΑΡΑΡΤΗΜΑ 1:
Δομή κώδικα Mopidy και περιγραφή βασικών API

Κώδικας Mopidy

Ο πηγαίος κώδικας του Mopidy απαρτίζεται από διάφορα modules, τα οποία παρέχουν μεθόδους και κλάσεις για τις διάφορες λειτουργίες του συστήματος. Τα modules αυτά μπορούν να εισαχθούν σε άλλα modules (import), και με αυτόν τον τρόπο αποφεύγεται να επαναλαμβάνεται πηγαίος κώδικας, για παράδειγμα σε περίπτωση που κάποια συνάρτηση χρησιμοποιείται από περισσότερα προγράμματα.

Σε αυτό το κεφάλαιο θα περιγράψουμε κάποιες από τις κλάσεις και τις μεθόδους που περιλαμβάνονται σε κάποια εκ των modules. Ο κώδικας είναι ανοιχτός οπότε πληροφορίες για τις κλάσεις και τις μεθόδους αυτές βρίσκεται στο Github στον ιστότοπο https://github.com/mopidy/mopidy ή αλλιώς στο https://docs.mopidy.com/en/latest/api/

Core

Στον φάκελο core περιλαμβάνονται τα modules που απαρτίζουν το Core API. Στο αρχείο **core/actor.py** βρίσκουμε την κλάση Core που είναι η κλάση Core(pykka.ThreadingActor, audio.AudioListener, backend.BackendListener, mixer.MixerListener,) που υλοποιείται ο core actor και υλοποιείται με Pykka actor. Στην κλάση αυτή μεταξύ των άλλων βρίσκουμε και τις μεθόδους :

- get_uri_schemes()

Η οποία επιστρέφει λίστα με URI που μπορούμε να διαχειριστούμε.

- get_version()

Επιστρέφει την έκδοση του core API του Mopidy.

core/tracklist.py

Στο αρχείο αυτό βρίσκεται η κλάση **TracklistController(core)** που είναι μια κλάση που είναι υπεύθυνη για τη διαχείριση των κομματιών που θέλουμε να αναπαράγουμε. Με

αυτή την κλάση υλοποιείται ο Tracklist Controller του επιπέδου core, που είναι υπεύθυνος για την διαχείριση της λίστας των κομματιών.

- get_tl_tracks()

Επιστρέφει την tracklist ως λίστα των κομματιών μαζί με το tracklist ID τους, με τη βοήθεια της κλάσης mopidy.models.TlTrack

- get_tracks()

Επιστρέφει την tracklist ως λίστα των κομματιών, με τη βοήθεια της κλάσης mopidy.models.TlTrack

- get_length()

Επιστρέφει το μήκος του tracklist.

- get_version()

Επιστρέφει την έκδοση του tracklist. Πρόκειται για έναν ακέραιο αριθμό ο οποίος αυξάνεται κάθε φορά που αλλάζει το tracklist.

- get_consume()

Επιστρέφει την κατάσταση λειτουργίας consume ως τιμές True και False. Όταν είναι ενεργοποιημένη η κατάσταση consume, τα κομμάτια αφαιρούνται από το tracklist, αλλιώς παραμένουν στη λίστα.

- set_consume(value):

Μέθοδος με την οποία ενεργοποιούμε ή απενεργοποιούμε την λειτουργία consume. Θέτοντας την τιμή value ως True ή False, ενεργοποιούμε ή απενεργοποιούμε την λειτουργία consume.

- set_random(value)

Μέθοδος με την οποία ενεργοποιούμε ή απενεργοποιούμε την λειτουργία random. Θέτοντας την τιμή value ως True ή False, ενεργοποιούμε ή απενεργοποιούμε την λειτουργία random. Όταν η λειτουργία random είναι ενεργοποιημένη, τα κομμάτια του tracklist αναπαράγονται σε τυχαία σειρά, αλλιώς αναπαράγονται με τη σειρά του tracklist.

- get_repeat()

Επιστρέφει την κατάσταση της λειτουργίας επανάληψης (repeat). Αν επιστρέψει True, το tracklist επαναλαμβάνεται συνέχεια, αλλιώς το tracklist παίζει μία φορά.

- set_repeat(value)

Ενεργοποίηση της λειτουργίας επανάληψης. Αν δώσουμε στην παράμετρο value την τιμή True, το tracklist επαναλαμβάνεται συνέχεια, αλλιώς το tracklist παίζει μία φορά.

- get_single()

Επιστρέφει την κατάσταση της λειτουργίας μονής αναπαραγωγής (single). Αν επιστρέψει True, η αναπαραγωγή σταματάει αφού ολοκληρωθεί η αναπαραγωγή του τρέχοντος κομματιού, αλλιώς αν επιστρέψει τιμή False, η αναπαραγωγή συνεχίζεται στα επόμενα κομμάτια.

- set_single(value)

Ενεργοποίηση λειτουργίας μονής αναπαραγωγής. Αν δώσουμε στην παράμετρο value την τιμή True, η αναπαραγωγή του τρέχοντος κομματιού σταματάει, εκτός εάν έχει ενεργοποιηθεί η λειτουργία repeat.

- index(tl_track=None, tlid=None)

Επιστρέφει τη θέση ενός συγκεκριμένου κομματιού, στο tracklist. Οι παράμετροι που δέχεται είναι η tl_track που αντιστοιχεί στο όνομα του κομματιού που θέλουμε να μάθουμε τη θέση του, και tlid είναι το TLID του κομματιού που θέλουμε να μάθουμε τη θέση του στο tracklist. Εάν δεν δώσουμε τις παραμέτρους αυτές, θα επιστραφεί η θέση στο tracklist του κομματιού που αναπαράγεται την τρέχουσα στιγμή.

- get_eot_tlid()

Επιστρέφει το TLID του κομματιού που θα αναπαραχθεί μετά από το τρέχον κομμάτι.

- eot_track(tl_track)

Επιστρέφει το κομμάτι που θα αναπαραχθεί μετά από το συγκεκριμένο κομμάτι (tl_track)

- get_next_tlid()

Επιστρέφει το TLID του κομματιού που θα αναπαραχθεί εάν καλέσουμε την μέθοδο mopidy.core.PlaybackController.next(). Κανονικά είναι το επόμενο κομμάτι στο tracklist. Εάν είναι ενεργοποιημένη η λειτουργία repeat, το επόμενο κομμάτι θα είναι το ίδιο με το τρέχον. Εάν είναι ενεργοποιημένη η λειτουργία random, θα επιστρέψει το TLID ενός τυχαίου κομματιού.

- next_track(tl_track)

Επιστρέφει το όνομα του κομματιού που θα αναπαραχθεί εάν καλέσουμε την μέθοδο mopidy.core.PlaybackController.next().

- get_previous_tlid()

Επιστρέφει το TLID του κομματιού που θα αναπαραχθεί εάν καλέσουμε την μέθοδο mopidy.core.PlaybackController.previous(). Κανονικά είναι το προηγούμενο κομμάτι του tracklist. Αν είναι ενεργοποιημένη η λειτουργία random ή/και η λειτουργία consume, θα επιστρέψει το τρέχον κομμάτι.

- previous_track(tl_track)

Επιστρέφει το όνομα του κομματιού που θα αναπαραχθεί εάν καλέσουμε την μέθοδο mopidy.core.PlaybackController.previous(). Κανονικά είναι το προηγούμενο κομμάτι του tracklist. Αν είναι ενεργοποιημένη η λειτουργία random ή/και η λειτουργία consume, θα επιστρέψει το τρέχον κομμάτι.

- add(tracks=None, at_position=None, uris=None)

Μέθοδος που προσθέτει κομμάτια στο tracklist. Εάν αντί για όνομα κομματιού (παράμετρος tracks) δίνεται παράμετρος uris, τα URI θα αναζητηθούν στη βιβλιοθήκη και θα προστεθούν στο tracklist. Εάν δίνεται η παράμετρος at position, τα κομμάτια θα προστεθούν στη συγκεκριμένη θέση του tracklist. Αλλιώς θα προστεθούν στο τέλος του tracklist.

- clear()

Κάνει εκκαθάριση του tracklist

- filter(criteria)

Φιλτράρει το tracklist βάσει δεδομένων κριτηρίων.

- move(start, end, to_position)

Μετακινεί συγκεκριμένο τμήμα του tracklist σε νέα θέση. Η παράμετρος start είναι η θέση στο tracklist του πρώτου κομματιού που θα μετακινηθεί. Η παράμετρος end είναι η θέση στο tracklist μετά από το τελευταίο κομμάτι που θα μετακινηθεί. Τέλος η παράμετρος to position είναι η νέα θέση των κομματιών που θα μετακινηθούν.

- remove(criteria)

Αναζητεί κομμάτια βάσει κριτηρίων και τα αφαιρεί από το tracklist.

- shuffle(start=None, end=None)

Ανακατεύει ολόκληρο το tracklist. Εάν δίνονται οι παράμετροι start και end, ανακατεύει τα κομμάτια μεταξύ των δύο αυτών θέσεων.

- slice(start, end)

Επιστρέφει τα κομμάτια μεταξύ των δύο θέσεων του tracklist που δίνονται στις παραμέτρους start και end.

core/playlists.py

Στο αρχείο αυτό βρίσκουμε την κλάση :

class PlaylistsController

Η κλάση αυτή είναι υπεύθυνη για τη διαχείριση των αποθηκευμένων playlists και με αυτή υλοποιείται ο Playlist Controller. Περιέχει διάφορες μεθόδους όπως :

- get_uri_schemes()

Η μέθοδος αυτή επιστρέφει λίστα με URI schemes τα οποία υποστηρίζουν playlists.

- as_list()

Επιστρέφει λίστα με τις διαθέσιμες playlists, χρησιμοποιώντας τη mopidy.models.Ref.

- get_items(uri)

Επιστρέφει τα χαρακτηριστικά μιας playlist. Η παράμετρος uri αντιστοιχεί στο URI της playlist αυτής και καλεί την κλάση mopidy.models.Ref

- create(name, uri_scheme=None)

Δημιουργεί μία νέα playlist. Εάν το uri scheme είναι συμβατό με τα URI scheme που μπορεί να διαχειριστεί ένα συγκεκριμένο backend, το backend καλείται να δημιουργήσει την playlist. Εάν δεν δοθεί κάποιο uri scheme ή αν δεν είναι συμβατό με το συγκεκριμένο backend, θα ζητηθεί από το πρώτο backend να δημιουργήσει την playlist.

- delete(uri)

  Με τη μέθοδο αυτή γίνεται διαγραφή μιας playlist, βάσει του URI της. Αν το URI δεν είναι έγκυρο δεν συμβαίνει τίποτα. Αν γίνει η διαγραφή μας επιστρέφει την τιμή True, αλλιώς False.

- lookup(uri)

Κάνει αναζήτηση playlist με βάση το URI τους σε όλες τις πιθανές πηγές που μπορεί να παρέχουν playlists. Αν τελικά δεν βρεθεί καμία playlist, επιστρέφει None.

- refresh(uri_scheme=None):

Κάνει ανανέωση (refresh) των playlists. Αν το uri scheme έχει τιμή None, θα κάνουν ανανέωση όλα τα backends. Αν το uri scheme που θα δοθεί αναγνωρίζεται από το backend, τότε μόνο το συγκεκριμένο backend θα κάνει ανανέωση playlist. Αν το uri scheme δεν αναγνωρίζεται από κανένα backend, δεν γίνεται τίποτα.

- save(playlist)

Μέθοδος που χρησιμεύει στο να αποθηκεύουμε playlists. Η παράμετρος playlist αντιστοιχεί στην playlist που θέλουμε να αποθηκεύσουμε, και μας την επιστρέφει η κλάση mopidy.models.Playlist.

core/library.py

class LibraryController

Η κλάση αυτή είναι υπεύθυνη για τη διαχείριση της μουσικής βιβλιοθήκης όπως για παράδειγμα για την αναζήτηση μουσικής και περιήγηση στις βιβλιοθήκες και υλοποιεί τον Library Controller.

- browse(uri)

Χρησιμοποιείται για περιήγηση σε directories και κομμάτια, τα οποία directories αναπαρίστανται με ένα URI. Αν στην παράμετρο uri εισαχθεί η τιμή None, θα είναι δυνατή η περιήγηση στο root directory του backend. Επιστρέφει μία λίστα αντικειμένων (objects) της κλάσης mopidy.models.Ref με τα directories και κομμάτια που περιέχονται στο συγκεκριμένο URI.

- get_images(uris)

Αναζητά τις εικόνες για συγκεκριμένα URIs. Τα backends μπορούν να χρησιμοποιήσουν τη μέθοδο αυτή ώστε να επιστρέφουν URI εικόνων βάσει συγκεκριμένων URI που αφορούν κομμάτια, άλμπουμ ή playlists. Εάν η παράμετρος uri αποτελείται από άγνωστα URI ή URI που δεν είναι συμβατά με το backend, επιστρέφει μία κενή λίστα για αυτό το URI.

- lookup(uris)

Αναζητά συγκεκριμένα URI. Εάν τα URI επεκτείνονται σε πολλαπλά κομμάτια, θα επιστρέψει λίστα τα οποία θα τα περιέχει όλα.

- refresh(uri=None)

Κάνει ανανέωση της βιβλιοθήκης. Δέχεται την παράμετρο uri η οποία είναι το URI ενός directory ή ενός κομματιού.

- search(query, uris=None, exact=False)

Κάνει αναζήτηση στη βιβλιοθήκη για κομμάτια με βάση συγκεκριμένα κριτήρια (query) όπως uri, όνομα κομματιού (track_name), άλμπουμ (album), καλλιτέχνη (artist), συνθέτη (composer), εκτελεστή (perfomer), αριθμό κομματιού (track_no), είδος μουσικής (genre), ημερομηνία (date), σχόλια (comment) ή με όλα τα κριτήρια (any). Μπορούμε να κάνουμε αναζήτηση με URI ώστε να περιορίσουμε την αναζήτηση ή χωρίς URI και με την

παράμετρο exact μπορούμε να ορίσουμε εάν η αναζήτηση θα γίνει με βάση τα ακριβή κριτήρια ή όχι. Η μέθοδος αυτή επιστρέφει τιμές με τη χρήση της κλάσης mopidy.models.SearchResult.

core/playback.py

Περιέχει την κλάση PlaybackController, η οποία διαχειρίζεται την κατάσταση αναπαραγωγής και το τρέχον κομμάτι που αναπαράγεται και υλοποιεί τον Playback Controller.

- get_current_tl_track

Επιστρέφει το τρέχον κομμάτι που αναπαράγεται και επιστρέφει τιμές με τη χρήση της κλάσης mopidy.models.TlTrack.

- get_current_track()

Επιστρέφει το τρέχον κομμάτι που αναπαράγεται και επιστρέφει τιμές με τη χρήση της μεθόδου get_current_tl_track, η οποία καλεί την κλάση mopidy.models.TlTrack.

- get_current_tlid()

Επιστρέφει το tracklist ID (TLID) του τρέχοντος κομματιού, με τη χρήση της μεθόδου get_current_tl_track.

- get_state ()

Επιστρέφει την κατάσταση αναπαραγωγής.

- set_state (new_state)

Εισάγει την κατάσταση αναπαραγωγής, με παραμέτρους Αναπαραγωγή (Playing), Παύση (Paused) ή Διακοπή (Stopped).

- get_time_position

Επιστρέφει την χρονική θέση της αναπαραγωγής ενός κομματιού σε milliseconds.

- next()

Δίνει εντολή αναπαραγωγής του επόμενου κομματιού, χωρίς να αλλάξει η κατάσταση αναπαραγωγής.

- pause()

Δίνει εντολή για παύση της αναπαραγωγής.

- play(tl_track=None, tlid=None)

Αναπαράγει το κομμάτι που το όνομά του που περιγράφεται στην παράμετρο tl_track και ανακτάται από την κλάση class:`mopidy.models.TlTrack , ή σύμφωνα με το tracklist ID του (tlid).

- previous()

Δίνεται εντολή αναπαραγωγής του προηγούμενου κομματιού, χωρίς να αλλάξει η κατάσταση αναπαραγωγής

- resume()

Σε περίπτωση παύσης της αναπαραγωγής, συνεχίζεται η αναπαραγωγή του τρέχοντος κομματιού.

- seek(time_position)

Βρίσκει τη συγκεκριμένη χρονική στιγμή του κομματιού σε milliseconds, σύμφωνα με την παράμετρο time_position και επιστρέφει True εάν βρεθεί η συγκεκριμένη χρονική στιγμή, αλλιώς False.

- stop()

Διακόπτει την αναπαραγωγή.

core/history.py

class HistoryController

Διατηρεί ιστορικό με το ποια κομμάτια έχουν ήδη παίξει και υλοποιεί τον History Controller.

- get_length()

Επιστρέφει των αριθμό των κομματιών που περιέχονται στο ιστορικό.

- get_history()

Επιστρέφει τη λίστα κομματιών που περιέχονται στο ιστορικό, με τη βοήθεια της κλάσης mopidy.models.Ref.

core/mixer.py

class MixerController

Η κλάση αυτή είναι υπεύθυνη για τη διαχείριση της στάθμης του ήχου (volume) καθώς και για τη λειτουργία σίγασης (muting) και υλοποιεί τον Mixer Controller.

- get_volume()

Επιστρέφει την τιμή της στάθμης ήχου σε γραμμική κλίμακα, με τιμές από 0 έως 100.

- set_volume(volume)

Θέτει την τιμή της στάθμης ήχου σε γραμμική κλίμακα, με τιμές από 0 έως 100 και επιστρέφει τιμή True αν η τιμή είναι έγκυρη, αλλιώς False.

- get_mute()

Επιστρέφει την κατάσταση σίγασης, εάν η σίγαση είναι ενεργοποιημένη επιστρέφει την τιμή True, αλλιώς False. Αν η κατάσταση είναι άγνωστη, επιστρέφει None.

- set_mute(mute)

Αν η παράμετρος mute πάρει την τιμή True, το σύστημα μπαίνει σε κατάσταση σίγασης, αλλιώς αν πάρει την τιμή False, βγαίνει από την κατάσταση σίγασης. Επιστρέφει True εάν η ρύθμιση είναι επιτυχής, αλλιώς False.

CoreListener.py

class CoreListener(listener.Listener)

Είναι μία διεπαφή για λήψη συμβάντων (events) από το core actor και υλοποιεί τον CoreListener.

- send(event, **kwargs)

Είναι ένας βοηθός (helper) ώστε να καλούμε συμβάντα του core listener.

- on_event(event, **kwargs)

Καλείται σε όλα τα συμβάντα. Εξ' ορισμού αυτή η μέθοδος προωθεί το συμβάν στις συγκεκριμένες μεθόδους του συμβάντος. Η παράμετρος event είναι το όνομα του συμβάντος που μας δίνεται από την κλάση mopidy.models.TlTrack, και η kwargs είναι ορίσματα σχετικά με τη διαχείριση του συμβάντος.

- track_playback_paused(tl_track, time_position)

Η μέθοδος αυτή καλείται όταν γίνεται παύση της αναπαραγωγής ενός κομματιού, όπου tl_track είναι το όνομα του κομματιού και time position είναι ο χρόνος που έγινε παύση σε milliseconds.

- track_playback_resumed(tl_track, time_position)

Μέθοδος που καλείται όταν ξαναρχίζει η αναπαραγωγή του κομματιού που είχε γίνει παύση.

- track_playback_started(tl_track)

Καλείται όταν ξεκινάει η αναπαραγωγή ενός καινούριου κομματιού (tl_track).

- track_playback_ended(tl_track, time_position)

Καλείται όταν η αναπαραγωγή ενός κομματιού τελειώνει.

- playback_state_changed(old_state, new_state)

Καλείται όταν η κατάσταση αναπαραγωγής αλλάζει. Η παράμετρος old_state είναι η κατάσταση πριν την αλλαγή, και η new_state η κατάσταση μετά την αλλαγή. Οι καταστάσεις αυτές ανακτώνται με τη χρήση του πεδίου mopidy.core.PlaybackState.

- tracklist_changed()

Καλείται κάθε φορά που η tracklist αλλάζει.

- playlists_loaded()

Καλείται κάθε φορά που οι playlists φορτώνονται ή ανανεώνονται.

- playlist_changed(playlist)

Καλείται κάθε φορά που η playlist αλλάζει. Η playlist που αλλάζει ανακτάται από την κλάση mopidy.models.Playlist.

- playlist_deleted(uri)

Καλείται κάθε φορά που μια playlist διαγράφεται. Η παράμετρος uri είναι το URI της διεγραμμένης playlist.

- options_changed()

Καλείται κάθε φορά που μια επιλογή αλλάζει.

- volume_changed(volume)

Καλείται κάθε φορά που ή στάθμη του ήχου αλλάζει. Η παράμετρος volume είναι η νέα τιμή της έντασης ήχου, με τιμές από 0 έως 100.

- mute_changed(mute)

Καλείται κάθε φορά που η κατάσταση σίγασης (mute) αλλάζει. Η παράμετρος mute είναι η νέα κατάσταση σίγασης και είναι δυαδικής (Boolean) μορφής.

- seeked(time_position)

Καλείται κάθε φορά που η χρονική θέση του κομματιού αλλάζει για κάποιο λόγο όπως λόγω της αναζήτησης νέας χρονικής θέσης. Η pπαράμετρος time_position είναι ή χρονική θέση που αναζητήθηκε σε milliseconds.

- stream_title_changed(title)

Καλείται κάθε φορά που ο τίτλος του αναπαραγόμενου stream αλλάζει, όπου title είναι το όνομα του νέου stream.

mopidy/backend.py

Σε αυτό το module περιλαμβάνεται κώδικας που αποτελεί το Backend API. Συγκεκριμένα συναντάμε την κλάση Backend η οποία λαμβάνει ως παραμέτρους την παράμετρο config(dict) η οποία αντιστοιχεί στο περιεχόμενο ολόκληρου του αρχείου ρυθμίσεων του Mopidy, και την παράμετρο audio που είναι ο actor proxy (pykka.ActorProxy) για το υποσύστημα ήχου (mopidy.audio.Audio).

- ping()

Μέθοδος που καλείται για να ελέγξουμε εάν ο actor είναι ενεργός. Επιστρέφει True εάν ο actor είναι ενεργός, αλλιώς False.

Library provider

Περιέχει την κλάση LibraryProvider

Παίρνει ως παράμετρο το backend (mopidy.backend.Backend) στο οποίο ανήκει ο controller.

- root_directory: Optional[Ref] = None

Ο αρχικός κατάλογος (root directory) του browse tree της συγκεκριμένης βιβλιοθήκης. Είναι ένα instance της κλάσης mopidy.models.Ref.directory.

- browse(uri: Uri) -> List[Ref]

Επιστρέφει αποτελέσματα στη μέθοδο mopidy.core.LibraryController.browse. Απαραίτητη προϋπόθεση είναι να έχει οριστεί πρώτα ο αρχικός κατάλογος (root_directory).

- get_images(uris: List[Uri]) -> Dict[Uri, List[Image]]

Επιστρέφει αποτελέσματα στη μέθοδο mopidy.core.LibraryController.get_images

- lookup(uri: Uri) -> Dict[Uri, List[Track]]

  Επιστρέφει αποτελέσματα στη μέθοδο mopidy.core.LibraryController.lookup.

- refresh(uri: Optional[Uri] = None) -> None

Επιστρέφει αποτελέσματα στη μέθοδο mopidy.core.LibraryController.refresh.

- search(query: Query[SearchField], uris: Optional[List[Uri]] = None,exact: bool = False) -> List[SearchResult]

Επιστρέφει αποτελέσματα στη μέθοδο mopidy.core.LibraryController.search.

Playback provider

Η κλάση PlaybackProvider είναι η κλάση με την οποία υλοποιείται ο Playback Provider και δέχεται την παράμετρο audio η οποία είναι ένας actor proxy ενός instance της κλάσης mopidy.audio.Audio και την παράμετρο backend η οποία αντιστοιχεί στο backend και προκύπτει από την κλάση mopidy.backend.Backend.

- pause() -> bool

Προκαλεί παύση της αναπαραγωγής. Επιστρέφει True αν η παύση εκτελεστεί επιτυχώς, και σε αντίθετη περίπτωση False.

- play() -> bool

Ξεκινά την αναπαραγωγή. Επιστρέφει True αν η εκκίνηση της αναπαραγωγή εκτελεστεί επιτυχώς, και σε αντίθετη περίπτωση False.

- prepare_change() -> None

Υποδεικνύει ότι αναμένεται αλλαγή ενός URI.

- translate_uri(uri: Uri) -> Optional[Uri]

Μετατρέπει ειδικά κατασκευασμένα URI σε υπάρχοντα εκτελέσιμα URI.

- is_live(uri: Uri) -> bool

Αποφασίζει εάν το URI θα πρέπει να αντιμετωπιστεί ως live stream ή όχι. Αναπαράγοντας μία μουσική πηγή ως live stream, απενεργοποιείται το buffering, με αποτέλεσμα να μειώνεται η καθυστέρηση πριν αρχίσει η αναπαραγωγή. Δέχεται ως παράμετρο ένα URI και επιστρέφει True ή False, ανάλογα με το αν θα αντιμετωπιστεί ως live stream ή όχι.

- should_download(uri: Uri) -> bool:

Δέχεται ως παράμετρο ένα URI και χρησιμοποιείται για να επιλέξουμε αν θα πραγματοποιηθεί κατέβασμα του URI σε τοπικό μέσο αποθήκευσης με σκοπό την απρόσκοπτη αναπαραγωγή του ή όχι.

- change_track(track: Track) -> bool

Η μέθοδος αυτή εξυπηρετεί στο να γίνει μετάβαση στο συγκεκριμένο κομμάτι. Η default υλοποίηση θα καλέσει τη μέθοδο translate_uri. Δέχεται την παράμετρο track που είναι το όνομα του κομματιού στο οποίο θέλουμε να μεταβούμε, και επιστρέφει True αν η μετάβαση είναι επιτυχής, αλλιώς False.

- resume() -> bool

Συνεχίζει την αναπαραγωγή από το χρονικό σημείο που είχε γίνει η παύση αναπαραγωγής. Επιστρέφει True αν η ενέργεια είναι επιτυχής, αλλιώς False.

- seek(time_position: int) -> bool

Αναζητεί συγκεκριμένη χρονική στιγμή στο κομμάτι, σε milliseconds. Επιστρέφει True αν η ενέργεια είναι επιτυχής, αλλιώς False.

- stop() -> bool

Διακόπτει την αναπαραγωγή. Επιστρέφει True αν η ενέργεια είναι επιτυχής, αλλιώς False.

- get_time_position() -> int

Επιστρέφει το τρέχον χρονικό σημείο του κομματιού που αναπαράγεται, σε milliseconds.

Playlists provider

class PlaylistsProvider

Η κλάση αυτή παρέχει μεθόδους που επιτρέπουν την αναζήτηση, την ανανέωση, τη δημιουργία, την τροποποίηση και διαγραφή playlists.

- as_list() -> List[Ref]

Επιστρέφει μια λίστα από αντικείμενα της κλάσης mopidy.models.Ref τα οποία σχετίζονται με τις playlists. Δεν επιστρέφει πληροφορίες σχετικά με το περιεχόμενο των playlists.

- get_items(uri: Uri) -> Optional[List[Ref]]

Επιστρέφει τα χαρακτηριστικά μίας playlist, η οποία αντιστοιχεί σε κάποιο δεδομένο URI. Επιστρέφει μια λίστα με αντικείμενα της κλάσης mopidy.models.Ref, τα οποία αναφέρονται στα χαρακτηριστικά αυτά ή None εάν το URI δεν υπάρχει.

- create(name: str) -> Optional[Playlist]

Δημιουργεί μια κενή playlist με συγκεκριμένο όνομα και με ένα URI. Σε περίπτωση αδυναμίας, επιστρέφει None.

- delete(uri: Uri) -> bool

Διαγράφει την playlist που αντιστοιχεί στο δεδομένο URI. Εάν η διαγραφή είναι επιτυχής επιστρέφει True, αλλιώς False.

- lookup(uri: Uri) -> Optional[Playlist]

Αναζητεί μια playlist με βάση το URI της σε ένα σετ από playlists ή σε οποιαδήποτε άλλη πηγή που περιέχει playlists. Επιστρέφει τις playlists που βρήκε, αλλιώς None.

- refresh() -> None

Ανανεώνει τις playlists.

- save(playlist: Playlist) -> Optional[Playlist]

Αποθηκεύει τη δεδομένη playlist. Επιστρέφει την αποθηκευμένη playlist, αλλιώς σε περίπτωση αποτυχίας None.

Backend Listener

Περιέχει την κλάση BackendListener η οποία είναι μια διεπαφή που τη χρησιμοποιούν οι Pykka actors ώστε να καλούν τις μεθόδους που περιέχονται σε αυτή όταν οι αντίστοιχες αλλαγές συμβαίνουν σε ένα backend actor. Η διεπαφή αυτή χρησιμοποιείται για να αναζητεί ποιους actors θα ειδοποιεί όταν γίνονται αλλαγές σε ένα backend actor, καθώς και να παρέχει default υλοποιήσεις για τους listeners που δεν ενδιαφέρονται για όλα τα events. Συνήθως μόνο ο core actor χρησιμοποιεί αυτή την κλάση.

- static send(event: str, **kwargs: Any) -> None:

- playlists_loaded(self) -> None:

Audio API

mopidy/audio

Στον φάκελο αυτό περιλαμβάνονται όλα τα modules που αποτελούν το audio API.

mopidy/audio/actor.py

Το module αυτό περιέχει δύο κλάσεις που μας ενδιαφέρουν. Η μια εξ αυτών είναι η class Audio(config, mixer) η οποία αποτελεί το audio actor και είναι ένας pykka actor, που υλοποιείται ώστε να γίνει αναπαραγωγή του ήχου από το GStreamer και παίρνει ως παραμέτρους τις ρυθμίσεις του Mopidy και το mixer, που στην περίπτωσή μας είναι το software mixer. Μερικές από τις μεθόδους που περιλαμβάνει είναι οι παρακάτω :

- on_start()

Πρόκειται για μια μέθοδο και συγκεκριμένα ένα hook, το οποίο είναι υπεύθυνο για το setup αφού ξεκινήσει ο actor, αλλά πριν αρχίσει να επεξεργάζεται τα μηνύματα. Αν εγερθεί κάποια εξαίρεση (exception), το stack trace θα καταγραφεί και ο actor θα σταματήσει.

- on_stop()

Ακόμα ένα hook το οποίο είναι υπεύθυνο για το cleanup που πρέπει να γίνει αφού ο actor διεκπεραιώσει το τελευταίο μήνυμα και πριν αυτός σταματήσει. Αν εγερθεί κάποια εξαίρεση (exception), το stack trace θα καταγραφεί και ο actor θα σταματήσει.

- set_uri(uri, live_stream=False, download=False)

Με τη μέθοδο αυτή θέτουμε το URI του κομματιού που θέλουμε να εκτελεστεί και ως παραμέτρους δέχεται το URI αυτό, την παράμετρο live_stream η οποία παίρνει Boolean τιμές και αν την θέσουμε False, απενεργοποιείται το buffering δηλαδή την συγκέντρωση αρκετής πληροφορίας ώστε η αναπαραγωγή να γίνει χωρίς διακοπές, κυρίως σε περιπτώσεις που το δίκτυο δεν είναι τόσο καλό, οπότε μειώνεται η καθυστέρηση μεταφοράς (latency), αλλά όταν γίνει παύση, δεν συγκρατούνται τα δεδομένα. Τέλος η παράμετρος download παίρνει και αυτή Boolean τιμές και ενεργοποιεί ή απενεργοποιεί τη

δυνατότητα να κατεβάσουμε το κομμάτι σε κάποιο τοπικό δίσκο, ώστε μετά να αναπαραχθεί απρόσκοπτα.

- get_position()

Επιστρέφει την θέση αναπαραγωγής του κομματιού σε milliseconds.

- set_position(position)

Με την μεταβλητή position μπορούμε να θέσουμε την θέση αναπαραγωγής του κομματιού σε milliseconds.

- start_playback()

Ειδοποιεί το GStreamer ότι πρέπει να αρχίσει την αναπαραγωγή. Σε περίπτωση η που έναρξη αναπαραγωγής γίνει επιτυχώς, επιστρέφει True, αλλιώς False.

- pause_playback()

Ειδοποιεί το GStreamer ότι θα πρέπει να σταματήσει την αναπαραγωγή. Σε περίπτωση που παύση γίνει επιτυχώς, επιστρέφει True, αλλιώς False.

- prepare_change()

Με τη μέθοδο αυτή ενημερώνεται το GStreamer ότι επίκειται αλλαγή της κατάστασης αναπαραγωγής και πρέπει να καλείται όποτε αλλάζουν τα URI που θα αναπαραχθούν γιατί το GStreamer διαγράφει την κατάσταση του όταν μπαίνει σε κατάσταση READY.

- stop_playback()

Ειδοποιεί το GStreamer ότι πρέπει να σταματήσει την αναπαραγωγή. Σε περίπτωση η που έναρξη αναπαραγωγής γίνει επιτυχώς, επιστρέφει True, αλλιώς False.

- wait_for_state_change()
- enable_sync_handler()
- set_metadata(track)
- get_current_tags()

mopidy/audio/listener.py

Module που περιλαμβάνει κώδικα για την υλοποίηση του Audio Listener. Η κλάση αυτή είναι η AudioListener() και περιλαμβάνει τις παρακάτω μεθόδους :

- static send(event, **kwargs)

Είναι ένας βοηθός (helper) που επιτρέπει την κλήση γεγονότων του audio listener.

- reached_end_of_stream()

Καλείται όταν το stream ήχου φτάνει στο τέλος.

- stream_changed(uri)

Καλείται όταν το stream του ήχου αλλάζει και ως παράμετρο παίρνει το URI του stream που αλλάζει.

- position_changed(position)

Καλείται όταν η χρονική θέση του stream αλλάζει και δέχεται ως παράμετρο ,μια τιμή σε milliseconds.

- state_changed(old_state, new_state, target_state)

Μέθοδος που καλείται αφού έχει αλλάξει η κατάσταση αναπαραγωγής από την πρότερη κατάσταση (old_state) στη νέα κατάσταση (new_state) και η παράμετρος target_state είναι μια ενδιάμεση κατάσταση που χρησιμοποιείται σε περιπτώσεις όπως το buffering.

- tags_changed(tags)

mopidy/audio/scan.py

Το module αυτό περιλαμβάνει την κλάση Scanner(timeout=1000, proxy_config=None) και είναι ένας βοηθός (helper) που εξυπηρετεί στην ανάκτηση πληροφοριών από τα URI και ως παραμέτρους δέχεται μια τιμή σε milliseconds που αντιστοιχεί στο χρόνο που θα κάνει

αναζήτηση πληροφοριών στο URI και ένα λεξικό (dictionary) με ρυθμίσεις proxy. Εκτός των άλλων, περιλαμβάνει τη μέθοδο

- scan(uri, timeout=None)

η οποία ανακτά μεταδεδομένα από ένα δεδομένο URI και παίρνει ως παράμετρο το URI αυτό και μια παράμετρο timeout η οποία αντιστοιχεί στο χρόνο που θα γίνεται η αναζήτηση σε milliseconds. Επιστρέφει πληροφορίες σχετικά με το URI καθώς και τη χρονική διάρκεια του URI σε milliseconds.

mopidy/audio/utils

Στο module αυτό συναντάμε μεθόδους που σχετίζονται με τη λειτουργία του GStreamer όπως :

- calculate_duration(num_samples, sample_rate)

Υπολογίζει τη διάρκεια των δειγμάτων με τη χρήση ενός GStreamer helper.

- create_buffer(data, timestamp=None, duration=None)

Δημιουργεί ένα νέο GStreamer buffer βάσει δεδομένων πληροφοριών.

- millisecond_to_clocktime(value)

Μετατρέπει ένα χρόνο ο οποίος είναι σε milliseconds σε μορφή που να κατανοεί ο GStreamer εσωτερικά.

- clocktime_to_millisecond(value)

Μετατρέπει μια τιμή χρόνου που κατανοεί εσωτερικά ο GStreamer σε milliseconds.

- supported_uri_schemes(uri_schemes)

Με τη μέθοδο αυτή καθορίζονται ποια URI υποστηρίζονται από το GStreamer.

- setup_proxy(element, config)

Η μέθοδος αυτή δίνει τη δυνατότητα ρυθμίσεως ενός GStreamer στοιχείου (element) με ρυθμίσεις proxy.

mopidy/mixer.py

Το module αυτό περιέχει την κλάση Mixer(config: Dict) η οποία είναι το API του audio mixer και παίρνει ως παράμετρο ολόκληρο το configuration του Mopidy.

- get_volume() -> Optional[int]

Η μέθοδος αυτή επιστρέφει την τιμή της στάθμης του ήχου σε γραμμική κλίμακα από 0 έως 100, όπου 0 είναι μηδενική στάθμη ήχου και 100 η μέγιστη στάθμη ήχου.

- set_volume(volume: int) -> bool

Με τη μέθοδο αυτή ορίζεται η στάθμη του ήχου του mixer και παίρνει τιμές από 0 έως 100.

- trigger_volume_changed(volume: int) -> None

Η μέθοδος αυτή καλείται όταν γίνεται αλλαγή του επιπέδου της στάθμης του ήχου και στέλνει ένα event με όνομα “volume_changed” σε όλους τους mixer listeners.

- get_mute() -> Optional[bool]

Επιστρέφει True όταν ο mixer βρίσκεται σε κατάσταση σίγασης και False στην αντίθετη περίπτωση.

- set_mute(mute: bool) -> bool

Δίνοντας ως παράμετρο την τιμή True, θέτει το Mixer σε κατάσταση σίγασης, αλλιώς με False ο mixer βγαίνει από την κατάσταση σίγασης.

- trigger_mute_changed(mute: bool) -> None

Αυτή η μέθοδος καλείται όταν αλλάζει η κατάσταση σίγασης και στέλνει ένα event με όνομα “mute_changed” σε όλους τους mixer listeners.

- ping() -> bool

Με αυτή τη μέθοδο γίνεται έλεγχος αν ο actor λειτουργεί ακόμα ή έχει σταματήσει να λειτουργεί.

Στο module αυτό περιλαμβάνεται και η κλάση MixerListener() η οποία είναι μια διεπαφή που τη χρησιμοποιούν οι Pykka actors ώστε να καλούν τις μεθόδους που περιέχονται σε αυτή όταν οι αντίστοιχες αλλαγές συμβαίνουν στο mixer actor.

- send(event: MixerEvent, **kwargs: Any) -> None

Είναι ένας βοηθός (helper) που επιτρέπει την κλήση γεγονότων (events) του mixer listener.

- volume_changed(volume: int) -> None

Η μέθοδος αυτή καλείται όταν γίνεται αλλαγή στο επίπεδο της στάθμης του ήχου και επιστρέφει τιμές από 0 έως 100.

- mute_changed(mute: bool) -> None

Μέθοδος που καλείται όταν αλλάζει η κατάσταση σίγασης και επιστρέφει True όταν η κατάσταση τίθεται σε κατάσταση σίγασης, και False όταν δεν είναι σε κατάσταση σίγασης.

Άλλα διαθέσιμα APIs είναι τα παρακάτω :

- **mopidy/models.py** --- Data model API
- **mopidy/commands.py** --- Commands API
- **mopidy/config** --- Config API
- **mopidy/httpclient.py** --- HTTP client helpers
- **mopidy/zeroconf.py** --- Zeroconf API

ΠΑΡΑΡΤΗΜΑ 2:
Άδεια Mopidy

Τα Mopidy προστατεύεται από την άδεια Apache License, Version 2.0 το κείμενο της οποίας είναι το παρακάτω :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of

this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or,

within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,

whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at