UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

CONTRIBUȚII LA SISTEMUL DE CASĂ INTELIGENTĂ CASANDRA

PROIECT DE DIPLOMĂ

Prezentat ca cerință parțială pentru obținerea titlului de Inginer în domeniul Electronică și Telecomunicații programul de studii de lincență Calculatoare și Tehnologia Informției

Conducător științific:
Conf. Dr. Ing. Horia Cucu

Absolvent:
Simion Andrei

București
2020

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Departamentul **EAII**

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **SIMION C. Andrei , 442A**

1. Titlul temei: Contribuţii la sistemul de casă inteligentă Casandra

2. Descrierea contribuţiei originale a studentului (în afara părţii de documentare) şi specificaţii de proiectare:
O dată cu avansarea tehnologie și a modulelor de automatizare clădirile inteligente încep să devină un standard în construcții, mai ales în cele de interes public, precum sedii ale instituțiilor statului, școli, spitale etc., dar și multe construcții private, precum sedii de companii sau vile personale, sunt echipate cu sisteme inteligente din ce în ce mai sofisticate.Astfel că in cadrul proiectului de cercetare intitulat "Natural-language, Voice-controlled Assistive System for Intelligent Buildings" , coordonat de grupul de cercetare SpeeD, a fost dezvoltat și un prototip de casă inteligentă, demonstrabil în acest moment în Centrul CAMPUS. Prototipul conține urmatoarele module software :o componentă de detecție de cuvinte cheie "casandra" și transcrierea de comenzi vocale, componenta de înțelegere si execuție a comenzilor transcrise, componenta de sinteza de vorbire.Iar din punct de vedere hardware sistemul conține : Kit Philips Hue controlat prin intermediul uni API de tip REST,Sursă KNX + Router IP KNX,Termostat KNX cu afișaj digital, Intel NUC N3700 WEB (serverul central al aplicației demonstrative),Router WiFi Mikrotik, Raspberry Pi 3 Model B,Boxe pentru Raspberry Pi,Microfon USB.Proiectul Casandra a fost ulterior preluat și portat in Python ca proiect de diplomă apoi a fost continuat in cadrul disciplinei Proiect 2 propunându-se să se extindă sistemul deja existent.Sistemul are câteva limitări și anume :algoritmul de detecție a cuvintelor cheie avea o sensibilitate mai mică astfel că se activa fară a primi o comandă, modul de interacțiune cu sistemul inteligent de iluminat și imposibilitatea interacțiunii cu sistemul inteligent după ce se dă comanda de pornire a radioului.Soluția problemei sensibilității algoritmului ar fi modificarea parametrului ce defineste valoarea de prag ce reprezintă cât de repede se rostesc cuvintele.Acestă valoare o voi seta în funcție de numărul de silabe a cuvântului cheie ("casandra").Soluția pentru modul de interacțiune cu sistemul de iluminat este ca de fiecare dată când utilizatorul va rosti o comandă ce se referă la starea becului să se facă un request care să preia valoarea stării acestuia și apoi să se excute comanda dacă este cazul altfel sistemul să răspundă cu un mesaj în care să comunice starea actuală a becului.Prima abordare a problemei reducerii muzicii de pe fundalul unei comenzi date de utilizator ar fi reducerea zgomotului în mod direct din semnalul în timp și anume: dintr-un sinusoidal generat voi scade același semnal dar înregistrat pe boxa, dupa ce o să determin DT-ul cu care acesta este deplasat și voi alinia cele doua semnale.Urmatorea abordarea este să creez un filtru adaptiv și să pun la intrarile acestuia semnalul audio de la radio și semnalul audio cu comanda vocală peste care suprapun muzica de la radio și să determin semnalul vocal al comenzii utilizatorului.

3. Resurse folosite la dezvoltarea proiectului:
Limbaj de programare : Python3 ; Hardware : Raspberry Pi 3, microfon, difuzor, kit becuri rgb Phillips, termostat knx, router wifi, sursa knx, router knx, Intel NUC ; Software : o componentă de detecție de cuvinte cheie și transcrierea de comenzi vocale, componenta de înțelegere si execuție a comenzilor transcrise, componenta de sinteza de vorbire

4. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline:
Programarea Calculatoarelor, Structuri de Date și Algoritmi, Prelucrarea Digitală a Semnalelor

5. Proprietatea intelectuală asupra proiectului aparţine: U.P.B.

6. Data înregistrării temei: 2019-11-27 17:06:21

Conducător(i) lucrare,
Conf. dr. ing. Horia CUCU

semnătura: .....

Student,

semnătura:.....

Director departament,
Prof. dr. ing Sever PAŞCA

semnătura:.....

Decan,
Prof. dr. ing. Cristian NEGRESCU

semnătura:.....

Cod Validare: **f21049f54e**

Anexa 5

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul " Contribuții la sistemul de casă inteligentă Casandra", prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Ingineria Informației*, programul de studii *Calculatoare și Tehnologia Informației* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, *1.09.2020*

Absolvent *Andrei SIMION*

(semnătura în original)

CUPRINS

LISTA ACRONIMELOR

ACC - Appliance Configuration and Control module
API - Application Programming Interface
ASR - Automatic Speech Recognition
BLE - Bluetooth Low Energy
CLI - Command Line Interface
DSI - Display Serial Interface
GPIO - General-purpose input/outup
GRUB - Grand Unified Bootloader
HDMI - High-Definition Multimedia Interface
HTTP - Hypertext Transfer Protocol
IDE - Integrated development environment
IoT - Internet of Things
IP - Internet Protocol
JSFG - Journal of Structural and Functional Genomics
JSON - JavaScript Object Notation
LED - Light-emitting diode
MPC - Music Player Control
MPD - Music Player Daemon
NLMS - Normalized Least Means Squares
NLU - Natural Language Understanding
OS - Operating System
POO - Programare Orientată pe Obiect
RAM - Random-access memory
REST- Representational State Transfer
SBC - Single-Board Compute

SNR - Signal Noise Ratio

SpeeD - Speech and Dialogue Research Laboratory

SCP - Secure Copy

SSH - Secure Shell-Protocol

SSL - Secure Socket Layer

TSL - Transport Layer Security

TTS - Text-to-Speech

URL - Uniform Resource Locator

USB - Universal Serial Bus

UTF-8 - Unicode Transformation Format

WWW - World Wide Web

LISTA FIGURILOR

LISTA TABELELOR

CAPITOL 1

INTRODUCERE ȘI MOTIVAȚIA PROIECTULUI DE DIPLOMĂ

1.1 MOTIVAȚIE

O dată cu avansarea tehnologiei și a modulelor de automatizare clădirile inteligente încep să devină un standard în construcții, mai ales în cele de interes public, precum sedii ale instituțiilor statului, școli, spitale etc., dar și multe construcții private, precum sedii de companii sau vile personale, sunt echipate cu sisteme inteligente din ce în ce mai sofisticate. Cel mai facil si la îndemână mod de interacțiune între om si mașină este reprezentat de vocea umană. Cu ajutorul acestui mod de comunicare se poate comanda un sistem sa execute anumite sarcini, venind în întâmpinarea nevoilor utilizatorului. Astfel că în cadrul proiectului de cercetare intitulat "Natural-language, Voice-controlled Assistive System for Intelligent Buildings", coordonat de grupul de cercetare SpeeD, a fost dezvoltat și un prototip de casă inteligentă, demonstrabil în acest moment în Centrul CAMPUS. Prototipul conține urmatoarele module software : o component de detecție a cuvintelor cheie "Casandra" și transcrierea comenzilor vocale, componenta de înțelegere și execuție a comenzilor transcrise, componenta de siteza de vorbire. Proiectul Casandra a fost ulterior preluat și poratat în python ca proiect de diploma apoi a fost continuat în cadrul disciplinei Proiect 2 propunându-se să se extindă sistemul deja existent refactorizandu-se codul de Python

deja existent. În momentul preluării proiectului sistemul avea câteva limitări :algoritmul de detecție a cuvintului cheie avea o problema la sensibilitate astfel că se activă fară a primi o comandă, modul de interacțiune cu sistemul inteligent de iluminat și imposibilitatea interacțiunii cu sistemul inteligent după ce se dă comanda de pornire a radioului. De aceea în proiectul de diplomă îmi propun să reduc aceste limitări și astfel sistemul de casă inteligentă Casandra să devină o soluție fiabilă pentru ați crea mediu locuibil usor de automatizat comandat prin voce.

1.2 IoT

Conceptul Internet of Things (IoT) presupune folosirea Internetului pentru a conecta între ele diferite dispozitive, servicii și sisteme automate formându-se astfel o rețea de obiecte. Aceste obiecte pot fi o mutitudine de dispozitive înzestrate cu componente electronice, soft-uri și senzori ce preiau date ce urmează să fie procesate de sistem. Aceste case inteligente sunt case ecologice ce ajuta la scăderea consumului și în același timp la satisfacerea nevoilor utilizatorului. Spre exemplu, o casa inteligenta va putea controla consumul fiecărui dispozitiv rezultând într-un consum minim de energie electrică.

1.3 Soluție

Proiectul meu de diplomă se bazează pe rezolvarea limitărilor deja precizate la punctul (1.1) . Casa inteligentă dispune de un dispozitiv, in cadrul lucrării mele un Raspberry Pi , ce are incorporat un microfon cu rolul de a prelua semnalul vocal de la utilizator. Soluțiile pentru problemele prezentate : soluția problemei sensibilitații algoritmului ar fi modificarea parametrului ce defineste valoarea de prag ce reprezintă cât de repede se rostesc cuvintele. Acestă valoare o voi seta în funcție de numărul de silabe a cuvântului cheie ("casandra"), soluția pentru a crea un mod inteligent de interacțiune cu sistemul de iluminat este ca de fiecare dată când utilizatorul va rosti o comandă ce se referă la starea becului să se facă un request care să preia valoarea stării acestuia și apoi să se excute comanda dacă este cazul altfel sistemul să răspundă cu un mesaj în care să comunice starea actuală a becului. Pentru a rezolva imposibilitatea de a mai da comenzi vocale în timp ce se redă muzica de la radio voi încerca două prin abordări. Prima dată voi încerca ar fi reducerea zgomotului în mod direct din semnalul în timp colculand întârzierea dintre cele două semnale. A doua abordare ar fi să folosesc un filtru adaptiv ce funcționează în timp real .

CAPITOL 2

SISTEMUL CASANDRA ACTUAL

2.1 DESCRIEREA COMPONENTELOR HARDWARE

Componentele hardware ale sistemului actual demonstrativ sunt :

- Raspberry Pi 3 Model B
- Microfon digital Samson USB
- Boxa portabila Gembird pentru Raspberry Pi
- Becuri inteligente Philips Hue
- Sursa KNX + Router IP KNX
- Termostat KNX cu afișaj digital
- Intel NUC N3700
- Router Wi-Fi Mikrotik

2.2 RASPBERRY PI 3 MODEL B

2.2.1 Informații generale

Pe placa de dezvoltare Raspberry Pi 3 B rulează aplicația de recunoaștere a cuvintelor cheie "Casandra" și a comenzilor. Raspberry Pi este o serie de SBC(Single Board Computer) dezvoltată în United Kingdom de Raspberry Pi Foundation[1]. Placa menționată face parte din generația a treia fiind primul model scos. Iar aceasta dispune de următoarele specificații [3]:

- Procesor quad core pe 64 de biți cu frecvența de 1.2 GHz
- 1GB memorie RAM
- Wireless Lan și Bluetooth Low Energy(BLE)
- Ethernet
- GPIO extins cu 40 pini
- 4 USB
- HDMI
- Port de display DSI pentru conectarea cu un display cu touchscreen
- Port MicroSD pentru a încărca sistemul de operare dar și pentru a stoca date
- Port Micro USB pentru alimentare

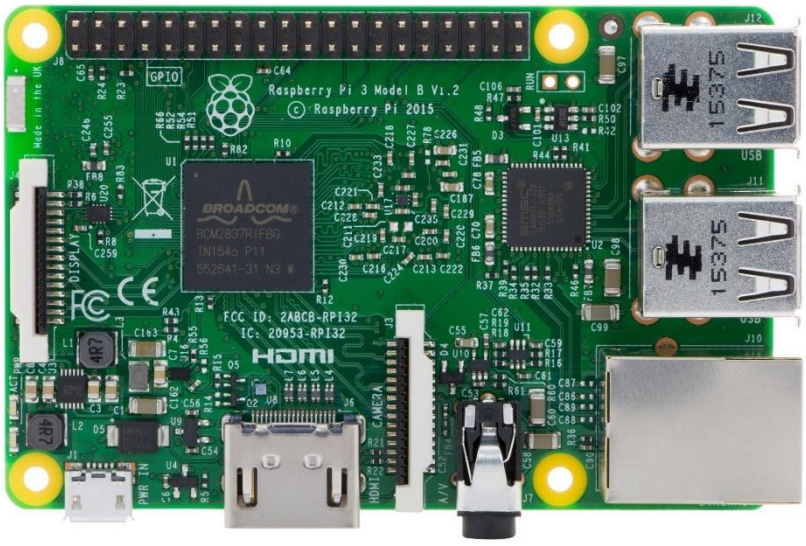


Fig. 2.1 Placă Raspberry Pi 3 B

2.2.2 Sistemul de operare

Sistemul de operare ce rulează pe Raspberry Pi este o verisune mai veche de Raspberry Pi OS numită Raspbian. Raspbian este un sistem de operare pentru care nu trebuie să plătești să îl poți folosi. Raspbian după cum îi spune și numele este bazat pe distribuția de linux Debian optimizată pentru componentele fizice de care dispune Raspberry Pi. Astfel că acesta oferă mai mult decât un sistem de operare pur ci vine și cu numeroase pachete precompilate și împachetate frumos pentru o instalare ușoară pe placă[4].

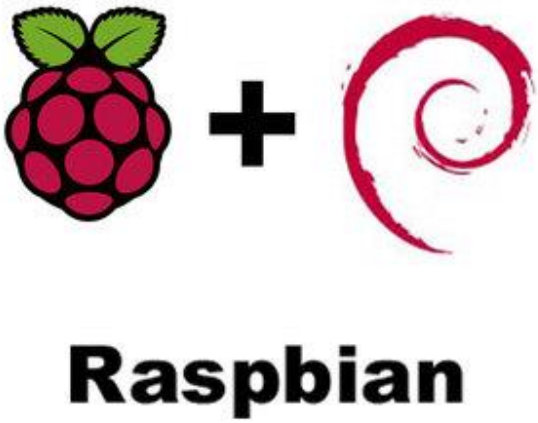


Fig. 2.2 Logo Raspbian

2.3 MICROFON DIGITAL SAMSON USB

Pentru a capta comenzile vocale date sistemului s-a mers pe utilizarea unui microfon omindirecțional. Ca soluție s-a ales un microfon digital de la Samson și anume microfonul UB1 cu o suprafață plană perfectă pentru recepția undelor sonore din toate directile. Acest microfon se instalează automat pe orice calculator sau în acest caz la placa de dezvoltare Raspberry Pi 3 B şi transmite datele direct prin cablul (miniUSB tata la USB tata). Cablul USB are atât rol de alimentare cât şi de transmitere a sunetului recepţionat sub formă de biți. Acest microfon se poate masca ușor într-o cameră datorită caracterului său minimal[5].

Fig. 2.3 Microfon Samson

2.4 BOXA PORTABILĂ GEMBIRD

Boxa Gembird SPK-103 este o boxă portabilă ce se alimentează printr-un cablu USB şi prin baterie proprie. Aceasta boxa are un design minimalist, elegant şi funcționează prin conectarea la orice aparat ce are port pentru jack de 3.5mm. Bateria are autonomie de până la 6 ore şi o greutate de sub 200 de grame având o puterea de emitere este de 2W şi bandă de frecvenţe pe care poate să emită este 100Hz-20kHz. Aceasta fiind utilizată pentru a reda răspunsurile sintetizate ale sistemului inteligent. Răspunsuri ce au fost create după îndeplinirea sarcinilor[6].

Fig. 2.4 Boxă Portabilă Gembird

2.5 BECURI INTELIGENTE PHILIPS HUE

2.5.1 Informații generale

Proiectul Philips Hue a fost realizat de cei de la Philips în anul 2012 și actualizat cu patru ani mai târziu adică în 2016. Philips Hue se bazează pe comanda unor becuri inteligente LED prin conectarea lor la o rețea Wi-Fi și intermediul unei conexiuni Zigbee.

Zigbee dezvoltat de Zigbee Alliance este un protocol de comunicații de nivel înalt folosit pentru rețele personale ce nu necesită un volum mare de energie pentru transmiterea datelor și este functional pe distanțe mici între zece și douăzeci de metri. Rata de biți cu care sunt transferate datele este de 250kbit/s iar rețelele Zigbee sunt securizate cu chei simetrice de criptare pe 128 biți[7].

2.5.2 Sistemul Hue

Sistemul de lumini Hue este alcătuit din trei componente principale:

1. Aplicație Client-Server bazată pe REST API
2. Bridge(podul de conexiune)
3. Becurile LED inteligente (Philips Hue)

1.Aplicația Client-Server este modul prin care utilizatorul va putea interacționa cu becurile inteligente. Metoda implementă pe sistemul actual Casandra pentru a comunica cu becurile este următoarea : se face o cerere către server pentru a primi un token cu care să te autentifici și să ai acces la caracteristicile becului apoi folosind tokenul obținut se trimit cereri de POST pentru a da ca date de intrare caracteristicile schimbate, ale becului sau becurilor, către server ce comandă automat și in timp real LED-urile.

2.Bridge-ul este folosit pentru a conecta becurile Philips la aplicație. Prin itermediul bridge-ului utilizatorul poate să folosească toate beneficiile becurilor. Pentru a stabili conexiunea este imperios necesar ca bridge-ul și raspberry pi-ul de pe care se face controlul becurilor să fie in aceeași rețea locală.

Fig. 2.5 Bridge Philips

3.Becurile inteligente Philips Hue sunt ieșirea sistemului și conțin 3 tipuri de LED alese special pentru a oferi o varietate de intensități și nuanțe de lumină pentru utilizator. Pe lângă toate acestea de fiecare dată când se produce o schimbare acestea transmit un mesaj de tip JSON.

Execuția comenzilor privind iluminarea va presupune transmiterea unor cereri de tip HTTP (GET / POST) către serverul REST pus la dispoziție de Philips Hue.

Fig. 2.6 Becuri Philips Hue

2.5.3 JSON în sistemul Hue

Toate răspunsurile și toate valorile noi pe care le iau becurile sunt trimise și returnate prin JSON(JavaScript Object Notation) și codate cu UTF-8. JSON este un format ușor de folosit pentru transmiterea de date. Pentru oameni este ușor de citit și de scris iar pentru computere și siteme este ușor de analizat si generat. JSON deși în nume face referință la javascript el este un format de text independent de acesta însă folosește anumite convețiii. Acest format este construit pe 2 structuri[8]:

- colecție de perechi de nume ale variabilelor și valorile lor
- listă ordonată a valorilor, în cele mai multe cazuri un vector

În sistemul actual este folosită prima structură ce presupune o colecție de caracteristici a becului având asociată o valoare . Sistemul folosește acest model pentru a păstra simplitatea utilizării lui.

2.6 Sursă KNX și Router IP KNX

KNX este un standard pentru automatizarea clădirilor. Dispozitivele KNX pot gestiona iluminatul, jaluzelel și obloanele, sistemele de securitate, gestiunea energiei etc.

Prin urmare aceste două componente hardware ale sistemului sunt folosite pentru a gestiona controlul termostatului amplasat pe macheta prototipului Casandra[9].

Fig. 2.7 Sursă KNX și Router IP KNX

2.7 Termostat KNX cu afișaj digital

Acest dispozitiv realizat tot în standardul KNX este folosit pentru pentru reglarea automată a temperaturii iar afișajul digital este utilizat pentru a avea confirmarea reală a modificării temperaturii.

Fig. 2.8 Termostat KNX cu afișaj digital

2.8 Intel NUC N3700

Intel NUC N3700 are următoarele specificații importante:

- 8GB RAM
- Spațiu de stocare SSD de 120GB

Pe acest sistem rulează componentele software NLU, TTS și WEB (serverul central al aplicației demonstrative)

Fig. 2.9 Intel NUC N3700

2.9 ROUTER WI-FI MIKROTIK

Router Wi-Fi Mikrotik RB951Ui-2HnD, cu suport USB, ce permite conectarea la modem-uri 3G / 4G, pentru a oferi conectivitate IP sistemului, independent de locație .

Fig. 2.10 Router Wi-Fi Mikrotik

CAPITOL 3

TEHNOLOGII SOFTWARE

3.1 LINUX

3.1.1 Informații generale

"La fel ca Windows, iOS și Mac OS, Linux este un sistem de operare. De fapt, una dintre cele mai populare platforme de pe planetă, Android, are la bază sistemul de operare Linux. Un sistem de operare este un software care gestionează toate resursele hardware asociate cu sistemul dumneavoastră desktop sau laptop. Pur și simplu, sistemul de operare gestionează comunicarea dintre partea software și partea hardware. Fără sistemul de operare (OS), software-ul nu ar funcționa"[10].

Sistemul de operare Linux este format din[10]:

1. Bootloader : software-ul care gestionează procesul de încărcare a computerului. ex: Das U-Boot, GRUB [10].
2. Nucleu (Kernel) : nucleul sistemului și gestionează procesorul, memoria și dispozitivele periferice. El este cel mai de jos nivel al sistemului de operare [10].
3. Sistemul de inițializare(Init system) : cu el gestionezi serviciile ce rulează în fundal. ex : systemd [10].
4. Serviciile ce rulează în fundal (Daemons) : serviciile pentru placa de sunet [10].
5. Server-ul grafic (Graphical server) : X11 server resposabil cu afișarea graficii pe monitor [10].
6. Mediul grafic (Desktop environment) : GNOME, Xfce, KDE etc. [10].
7. Aplicațiile [10].

3.1.2 Ubuntu 20.04 LTS și Raspbian

Pentru a lucra la proiectul de diplomă într-un mod mai profesionist și pentru a dobândi dexteritate în a lucra cu terminalul de linux mi-am instalat pe computerul personal Ubuntu 20.04 LTS. Un motiv în plus pentru care am optat pentru aceasta distribuție de linux pe langă faptul că este una prietenoasă și ușor de folosit este că la fel ca și Raspbian, Ubuntu are la bază Debian. Deci aceste doua distribuții sunt derivații ale lui Debian ceea ce înseamnă că structură(arhitectura, pachete) și setul de comenzi sunt asemănatoare.

3.1.3 Comenzi utile

Pe mini calculatorul Raspberry Pi pe care se afla portat proiectul Casandra rulează Raspbian Lite adică o versiune minimala ce nu are un mediu grafic instalat deci singura armă ramasă la îndemână pentru a putea implementa anumite funcții deci efectiv a acționa asupra fișierelor sistemului este utilizarea interfeței din linie de comanda adică CLI (Command Line Interface). Pe sistemul meu cu Ubuntu 20.04 LTS am ca mediu grafic GNOME 3.6 dar pentru a accesa si lucra pe Raspberry Pi folosesc aplicația gnome-terminal ce îmi deschide CLI. Din acest motiv am întocmit o listă cu comenzi utilizate :

1. `$ pwd`

   Această comandă afișează directorul current în care te afli[11].

2. `$ echo "<text>"`

   Această comandă scrie argumentul sau argumentele date pe streamul standard output[11].

3. `$ su`

   Această comandă este folosită pentru a te înregistra pe utilizatorul administrator(root)[11].

4. `$ sudo <comandă>`

   Comanda se va executa doar dacă utilizatorul are drepturi de administrator(root) pe acel calculator[11].

5. `$ clear`

   Comanda este utilizată pentru a curăța ecranul liniei de comandă.Conținutul nu este șters ci doar ecranul terminalului de mută mai jos[11].

6. `$ cp <flag> {numefișier} /calea/către/directorul/destinație/`

   Cu această comanda putem să copiem fișiere sau directoare dacă în câmpul `<flag>` punem `-r` sau `-R[11].`

7. `$ mv <flag> {numefișier} /cale/către/destinație/`

   Această comandă ne permite să mutăm fișierele sau directoarele (pentru asta avem nevoie sa punem ca `<flag>` -r) în alte directoare și în același timp ne permite să folosim ca și comandă de redenumire deoarece după ce s-a făcut mutarea fișierele sau directoarele sunt șterse[11].

8. `$ rm <flag> {numefișier}`

   Această comandă șterge fișierul specificat din directorul current[11].

9. `$ cat {numefișier}`

   Această comadă ne permite să afișăm conținutul fișierului în console[11].

10. `$ ls`

    Această comandă ne permite să afișăm coținutul directorului curent[11].

11. `$ cd /cale/către/destinație/`

    Această comandă îți permite să navighezi prin directoarele sistemului[11]

12. `$ mkdir {numedirector}`

    Această comandă îți oferă posibilitatea de a crea directoare[11].

13. `$ ssh user@ip_host_destinație`

    Această comandă îți permite să te conectezi la distanță (remote) la computerul cu adresa ip specificăta printr-o conexiune securizată, criptată[11].

14. `$ scp fișier_sursă user@ip_host_destinație:/cale/destinație`

    Această comandă îți permite să copiezi fișiere pe un calculator ce se află la distanța folosind o conexiune securizată , ssh. Practic avem comanda cp peste ssh[12]

15. `$ sudo apt install (Ubuntu 20.04) / $ sudo apt-get install (Raspbian)`

    Acestea sunt cele două variații ale comenzii cu care poți instala pachete[11]

16. `$ tail <flag> {numefișier}`

    Această comandă folosită fără nici un flag îți va afișa ultimele zece rânduri dintr-un fișier. Folosind diverse flaguri putem afișa utlimele n linii din fișier sau putem prelua ultimii n bytes[13]

17. `$ dd if=/cale/sursă of=/cale/destinație [opțiuni]`

    Cu această comandă putem face copii de rezervă a mediului de stocare și a imaginii linuxului în caz că stricăm ceva ca apoi să restaurăm tot ce am copiat [11]

3.1.4 VIM

Vim este un editor de text extrem de configurabil construit pentru a face crearea și editarea oricărui fel de text într-un mod foarte eficient. În vim există două moduri de utilizare. Unul este modul de comandă, iar altul este modul de inserare. În modul de comandă, utilizatorul se poate deplasa prin fișier, șterge text, etc.[14]. În modul inserare, utilizatorul poate adăuga text. Pentru a intra în modul inserare trebuie apăsată tasta **I** iar pentru a comuta în modul de comandă trebuie apăsată tasta **Esc**. Pentru a salva modificările făcute și a ieși din editorul de text trebuie ca în modul de comandă să tastăm **:wq** iar pentru a ieși dar fară să salvezi modificările **:q!** sau **:q**, evident tot în modul de comandă. Acest editor l-am folosit pentru a modifica fișierele din proiectul Casandra[15].

3.1.5 GITLAB

Gitlab este un sistem ce se ocupă cu depozitarea proiectelor software pentru a putea fi accesate de membrii unei echipe cu scopul de a asigura un mod eficient de lucru în echipa. Acest manager de depozitare permite duplicarea codului pentru a refolosi părţi din proiecte mai vechi pentru proiecte noi. Limbajul în care a fost scris GitLab este Ruby şi programul continue un Wiki.[16] Un wiki este un website şi/sau o bază de date construită de o comunitate de programatori, fiecare utilizator având acces să adauge sau să modifice conţinutul bazei de date. Gitlab oferă două variante, Gitlab Community Edition, Enterprise Edition şi Gitlab-hosted version. Unul din marele avantaje pe care le prezintă Gitlab este posibilitatea de a avea acces la fiecare update care a fost efectuat. Astfel, orice utilizator are acces la istoricul complet al proiectului echipei sale iar platforma îi permite să

observe ce modificări au fost făcute de la versiunea anterioară. Varianta Community Edition este gratis şi open-source[17].

Membrul echipei "clonează" proiectul pe calculatorul său şi poate lucra la proiect fără să îşi deranjeze colegii. Odată ce şi-a îndeplinit obiectivul şi codul rulează fară erori, utilizatorul îşi încarcă schimbările pe Gitlab unde colegii lui vor avea acces la noul proiect. Acest proces care a fost prezentat anterior continue mai multe subprocese. În momentul în care programatorul a modificat fişiere din proiect sistemul îl anunţă ce fişiere trebuie reîncărcate pe depozit. Acest proces se numeşte commit. Pentru a executa acest subproces, programatorul este obligat să lase un mesaj ce are rolul de a descrie pe scurt modificările aduse proiectului. În cazul de faţă, zona de depozitare remote este de tip online şi anume Gitlab.com[18].

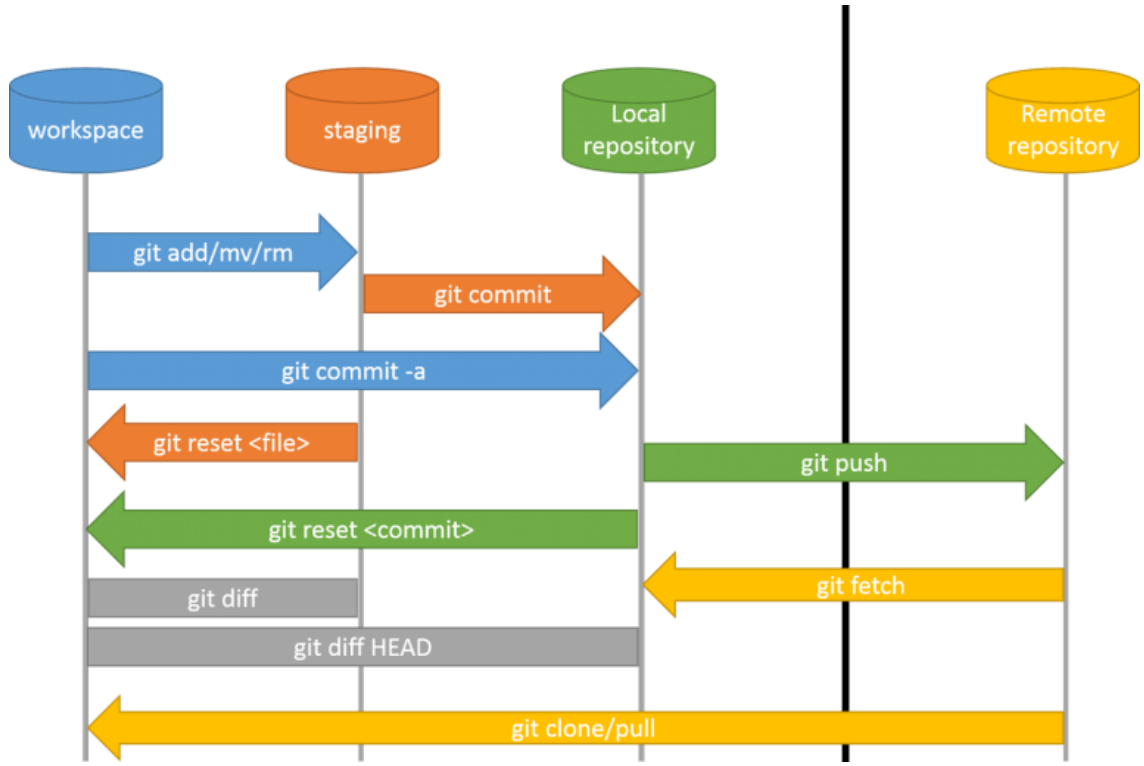


Fig. 3.1 Diagrama modului de funcționare git

3.2 PYTHON 3

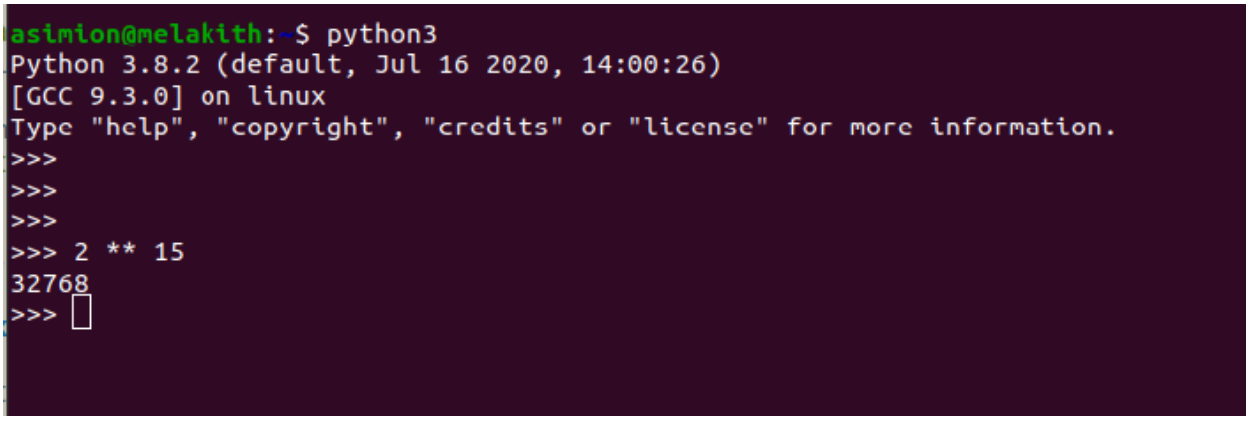
3.2.1 Informații generale

Python este un limbaj de programare de nivel înalt și are o aplicabilitate în mai multe domenii.

Deoarece codul este compilat direct în cod de octeți (byte code) și executat el este potrivit pentru a fi utilizat ca limbaj de scripting, limbaj pentru implementarea aplicațiilor Web, etc. Pentru că Python poate fi extins în C si C++, el ne poate oferi viteza necesară pentru a calcula și sarcini intensive. Deoarece este un limbaj bine structurat și cu o putere mare de abstractizare el fiind un limbaj ce se bazeaza pe princiile programării orientate pe obiecte(POO), Python ne permite să scriem aplicații clare, logice pentru sarcini mici și mari.

Dacă nu avem un mediu de dezvoltare software pentru Python și vrem să încercăm fragmente mici de cod avem posibilitatea doar tastând `pyhton` în linia de comadă să accesăm iterpretorul interactiv[19].

Fig 3.2 Consola Interactivă Python3



3.2.2 Instalarea bibliotecilor în Python 3

Python vine cu un set prestabilit de biblioteci standard dar în proiectele mai mari nu te poți baza doar pe acestea pentru că practic ar trebui să scrii și să concepi de la zero lucruri ce deja au fost scrise, adică să stai să reinventezi roata de fiecare data. De aceea s-au create biblioteci specializate pe anumite funcții. Așa că Python vine cu un instrument ce îți permite să descarci din baza de date biblioteca dorită pentru ca mai apoi să ai posibilitatea să o imporți dacă ai nevoie de ea în proiectul tău. Această comandă se numeste pip3 pentru versiune de Python 3.

Exemplu de utilizare: `$ pip3 install numpy --user`

Comanda este una intuitivă, după cum se observă dacă am scrie în teminal exemplul de mai sus am instala biblioteca `numpy`.

3.2.3 Spyder

“Spyder este un mediu științific puternic scris în Python, pentru Python, și proiectat de și pentru oameni de știință, ingineri și analiști de date. Acesta oferă o combinație unică de editare avansată, analiză, depanare, și funcționalitatea de profilare cuprins întru-un mediu de dezvoltare software ce-ti permite să vizualizezi variabilele create, execuție interactivă, inspecție profundă, și capacități de vizualizare frumos al unui pachet științific. Dincolo de numeroasele sale caracteristici încorporate, abilitățile sale pot fi extinse și mai mult prin intermediul sistemului său de plugin-uri și API. În plus, Spyder poate fi, de asemenea, utilizat ca o bibliotecă de extensie PyQt5, permițând dezvoltatorilor să se bazeze pe funcționalitatea sa și să încorporeze componentele sale, ar fi consola interactivă, în propriul software PyQt”[20].

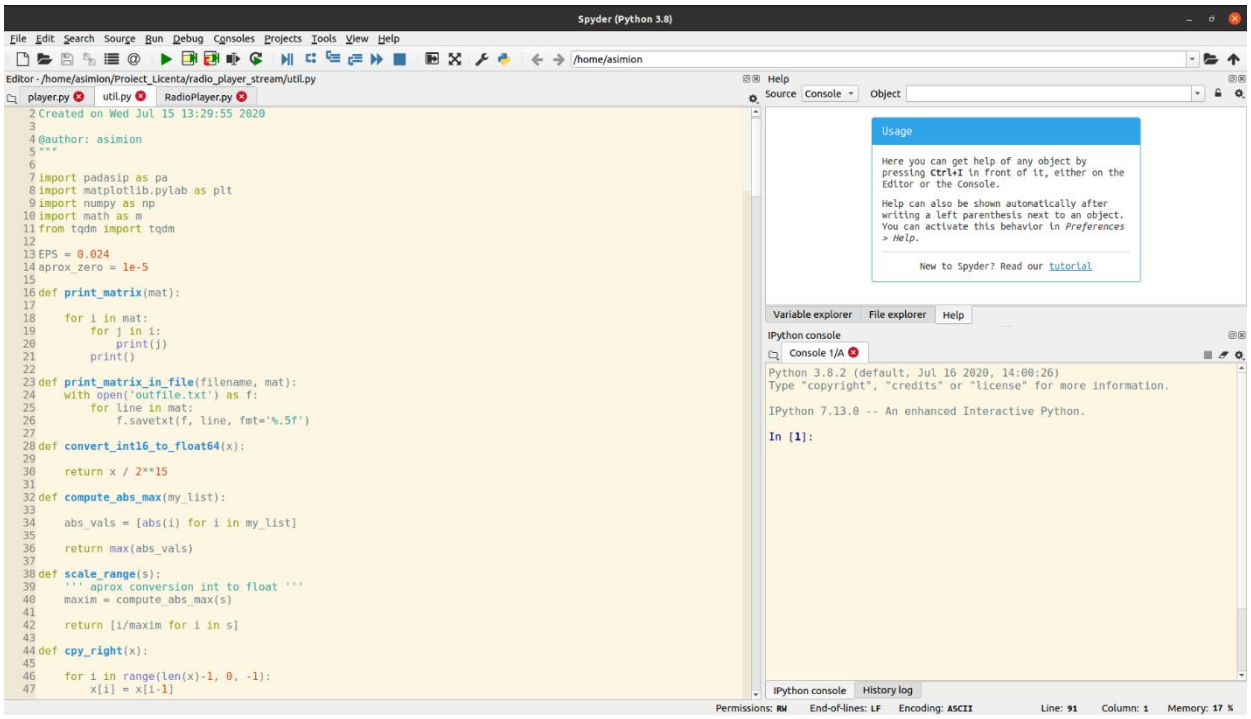


Fig. 3.3 Captură Ecran Din Spyder3-IDE

3.3 HTTP

3.3.1 Informații generale

Protocolul de transfer hypertext (HTTP) este un protocol la nivel de aplicație pentru sistemele informatice distribuite. HTTP a fost dat în uz de către World-Wide Web inițiativa globală de informare din 1990. Prima versiune de HTTP, denumită HTTP/0.9, a fost un protocol simplu pentru transferul de date brute pe Internet. HTTP/1.0, astfel este definit de RFC 1945, a îmbunătățit protocolul permițând mesajelor să fie în formatul mesajelor de tip MIME, conținând metainformare cu privire la datele transferate și modificatori pe semantica solicitării/răspunsului[21].

Protocolul HTTP este un protocol de solicitare/răspuns. Un client trimite o solicitare către server sub forma unei metode de solicitare, URI și versiune de protocol, urmată de un mesaj de tip MIME care conține modificatori de solicitări, informații despre client și conținut posibil printr-o conexiune cu un server. Serverul răspunde cu o linie de stare, inclusiv versiunea de protocol a mesajului și un cod de succes sau de eroare, urmat de un mesaj de tip MIME care conține informații despre server[22].

Multipurpose Internet Mail Extensions (MIME) este un standard de Internet care extinde formatul de e-mail el fiind proiectat pentru SMTP dar a fost introdus și pentru HTTP practic serverele introduc anteturile MIME la începutul oricărei transmisii Web. Programul client folosește acest antet MIME pentru a selecta aplicația de vizualizare corespunzătoare tipului de conținut sau tipului de media indicat în antet. Unele dintre aceste programe de vizualizare sunt integrate direct în programul-client Web sau în browser (de exemplu, aproape toate browserele pot afișa imagini GIF și JPEG în afara capacității acestora de a gestiona fișiere HTML)[23].

3.3.2 Tipuri de coduri status HTTP

Răspunsul serverului respectă un anumit set de reguli numite "status code" ce oferă informaţii cu privire la statusul cererii utilizatorului. Pentru înţelegerea răspunsurile HTTP voi oferi un exemplu pentru fiecare cod[24]:

- 1xx această clasă transmite numai un mesaj informativ Exemplu : 101 Anunţă schimbarea protocoalelor[24].
- 2xx această clasă anunţă că cererea a fost înţeleasă şi acceptată Exemplu: 200 OK (răspuns standard pentru cerere HTTP) 204 "No Content" cererea a fost înţeleasă dar nu serverul nu returnează nicio informaţie[24].
- 3xx această clasă se ocupă cu redirecţionarea sugerând că utilizatorul trebuie să mai facă nişte paşi pentru a transmite cererea. Exemplu: 301 "Moved Permanently" această cerere va fi redirecţionată către noul URL [24].
- 4xx această clasă sugerează că eroarea provine din cauza clientului Exemplu: 404 "File Not Found" 403 "Forbidden" accesul la această pagină este interzis deoarece utilizatorul nu are acces la aceste informaţii (poate nu este logat)[24].
- 5xx această clasă anunţă că problema a apărut din cauza serverului Exemplu: 502 "Bad Gateway" sugerează că serverul accesat se comportă ca o "poartă" eventual spre un virus[24].

Pentru ca HTTP este un serviciu ce este nesecurizat având portul 80 asignat pentru a citi traficul de aceea s-a creat protocolul de comunicare securzat HTTPS (Secure HTTP) şi aceasta nu permite transferul între client şi server dacă serverul nu are un certificate de Securitate SSL. De asemenea toate transferurile de date sunt criptate. În eventualitatea în care mesajul este intercepționat acesta nu va putea fi descifrat. Portul folosit pentru HTTPS este 443[25].

3.4 REST

Representation State Transfer sau pe scurt REST a fost propus de Roy Fielding și a fost sursă de inpirație atât pentru documentul arhitectural pentru servicii Web al grupului tehnic W3C cât și pentru cei ce îl iau drept model pentru dezvoltarea serviciilor Web. REST este subsetul WWW (bazat pe HTTP) în care furnizează print intermediul agenților (client sau server) semantică de interfață uniformă; în esență creează, recuperează, actualizează și șterg astfel manipuland resursele numai prin schimbul de reprezentări. Mai mult, interacțiunile REST sunt „stateless” în sensul că înțelesul unui mesaj nu depinde de starea conversației. Cu alte cuvinte stateless sugereaza faptul că operațiile client server nu trebuie reținute de server, cookie-uri, etc. [26].Primele constrângeri adăugate sunt cele ale stilului arhitectural client-server. Separarea sarcinilor este principiul din spatele constrângerilor de tip client-server. Prin separarea atribuțiilor interfeței de utilizator de problemele de stocare a datelor, se îmbunătățește portabilitatea interfeței de utilizator pe mai multe platforme și totodată se îmbunătățește scalabilitatea prin simplificarea componentelor serverului. Poate cel mai semnificativ pentru Web, cu toate acestea, este faptul că separarea permite componentelor să evolueze independent, sprijinind astfel cerința la scară mare mai exact la nivel de rețea internațională a mai multor domenii organizaționale[26].La interacțiunea client-server adăugăm proprietatea de stateless: comunicarea trebuie să fie de natură independentă, ca în client-stateless-server, astfel încât fiecare cerere de la client la server trebuie să conțină toate informațiile necesare pentru a înțelege cererea. și nu poate profita de niciun context memorat pe server. Prin urmare, starea sesiunii este păstrată în întregime pe client.Un alt avantaj ce face arhitectură REST accesibilă este posibilitatea programatorului de a-o implementa în orice limbaj de programare (JavaScript,Java,Python,AngularJS) atâta timp cât limbajul îţi

permite să faci cereri HTTP la un API RESTful(un API ce îndeplineşte condiţiile unei arhitecturi REST)[27].Există mai multe modalități de securizare a unei API RESTful, de ex. auth basic, OAuth etc., dar un lucru este sigur că API-urile RESTful ar trebui să fie stateless - deci cererea de autentificare / autorizare nu ar trebui să depindă de cookie-uri sau sesiuni. În schimb, fiecare cerere API ar trebui să vină cu o serie de acreditări de autentificare care trebuie validate pe server pentru fiecare solicitare. Dar pe lânga aceste metode arhitectura REST folosește protocolul HTTPS plus tehnologiile de criptare SSL/TSL[28].

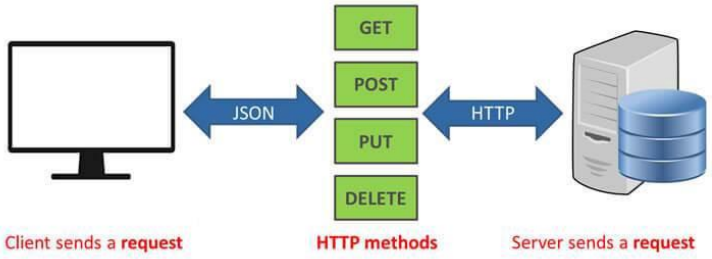


Fig. 3.4 Model REST

3.5 MODULUL RADIO

Sistemul inteligent Casandra dispune și de un modul radio digital cu care poți asculta posturi de radio de pe stream. Acest modul folosește la bază un serviciu numit Music Player Daemon cu care poți da play la muzica. Cu el se poate reda o varietate de fișiere audio în timp ce sunt controlate de protocolul său de rețea.[29].Tot cu MPD se pot organiza fișierele audio in playlist-uri deci fiecare stream radio se poate introduce într-o astfel de bază de date și apoi accesate prin intermediul unui index de către un program client, ce se folosțe de acest serviciu MPD, numit intuitiv Music Player Client . MPC functionează exact ca un dispozitiv ce redă muzică cum ar fi un MP3 Player sau casetofon. Practic are următoarele comenzi [30] :

- `$ mpc add` adaugă o melodie la playlist
- `$ mpc play` reda muzică (această comandă poate să aibă şi un număr pentru a începe redarea de la o anumită melodie/staţie radio)
- `$ mpc pause`
- `$ mpc stop`
- `$ mpc next` reda următoarea melodie din playlist
- `$ mpc volume` modifica volumul
- `$ mpc clear` șterge playlistul

CAPITOL 4

EXTENSIA SISTEMULUI CASANDRA

4.1 INFORMAȚII GENERALE DESPRE SISTEMUL INIȚIAL

În cadrul proiectului de cercetare intitulat “Natural-language, Voice-controlled Assistive System for Intelligent Buildings” , coordonat de grupul de cercetare SpeeD și finalizat în septembrie 2017, a fost dezvoltat și un prototip de casă inteligentă, demonstrabil în acest moment în Centrul CAMPUS. Prototipul include următoarele module software:

- Componentă de detecție de cuvinte cheie (“Casandra”) și transcriere de comenzi vocale;
- Componentă de înțelegere și execuție a comenzilor transcrise;
- Componentă de sinteză text-vorbire;
- Altele.

Acest sistem se bazează pe un server la care se leagă în reațea toate compomentele auxiliare comunicării cu sistemul inteligent dar și dispozitivele pe care vrem se controlam prin voce. Serverul permite utilizatorului să îl conecteze la internet adică la rețeaua WAN și să poată controla de la distanță întreg sistemul.

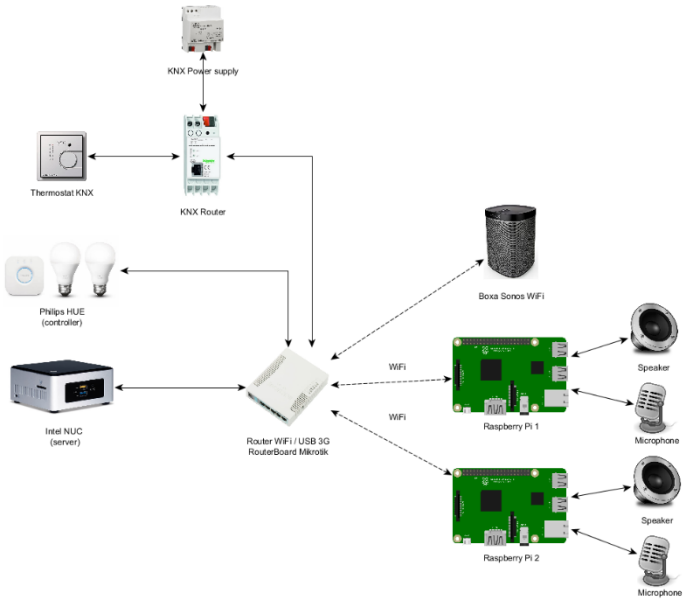


Fig. 4.1 Diagrama Prototip a Sistemului Inteligent Casandra

După cum se poate observa în Fig. 4.1 serverul este computerul Intel NUC 3700. Pe acest sistem rulează componentele software NLU, TTS și WEB (serverul central al aplicației demonstrative) iar pe Raspberry Pi 3 Model B rulează aplicația de recunoaștere a cuvintelor cheie ”Casandra” şi a comenzilor.

Modulul NLU-Natural Language Understanding

Acest modul este considerat a fi elementul central al spaţiului inteligent. Aici este locul unde se procesează mesajul primit sub formă text de la toate celelalte module, având în plan "traducerea" acestora pentru a putea fi transmise mai departe către modulul ACC(Appliance Configuration and Control).

Modulul TTS- Text-to-Speech

Rolul pe care îl are TTS este de a întoarce un stream wave audio direcţionat către punctul din care s-a primit mesajul text în limba în care se doreşte sinteza cu scopul de a obţine un mesaj vocal.

Modul ASR-Automatic Speech Recognition

Aici are loc exact procesul invers faţă de cel întâlnit în modulul TTS adică are loc procesul de recunoaştere a vocii primind un stream wave audio şi întorcând un mesaj text.

Modulul ACC-Appliance Configuration and Control module

Modulul acesta este locul unde are loc configurarea iniţială a fiecărui dispozitiv existent.

Modulul Web

Utilizatorul poate să configureze metodă proprie de control asupra fiecărui dispozitiv deoarece acest modul este de fapt un server standard Apache.

Pe lângă aceste module ce se află pe server există și programe/module care sunt pe dispozitivul Raspberry Pi 3 B. Acolo găsim: un program ce interfațează modulu TTS de pe server, program ce se ocupă cu recunoașterea vorbirii diferit de cel de pe server ce era scris în C. Aceasta versiune de cod scrisă în python folosește cele trei modelele utilizate în proiectul inițial și anume : model acustic (-hmm) Hidden Markov Model, model de gramatică (-jsfg) și de dicționar (-dict) ce conține toate cuvintele si fonemele acestora. Programul de recunoaștere folosește biblioteca pocketsphinx. Programul ce se ocupa cu recunoașterea vorbirii folosește keyword-spotting ,în teorie, obligă

programul să nu asculte conversațiile ce se țin în interiorul casei și să fie activat numai când cuvântul cheie (keyword) a fost folosit. În cazul de față, cuvântul cheie folosit este evident „casandra”. Tot pe Raspberry Pi se află și modulele ce comdandă sistemul de iluminat, sistemul de temperatură, sistemul de alarmă și modulul radio.

4.2 Obiectivele Extinderii Sistemului

Sistemul preluat de mine avea o problemă la algoritmul de keyword spotting și anume sistemul Casandra se activa când nu trebuia sau nu se activa când se rostea cuvantul cheie. Primul caz se întâmpla atunci când prin microfon se înregistra la rând câteva silabe din componența cuvântului cheie iar al doilea caz se întâmpla când modul în care spuneai cuvântul cheie repede/lent . Pentru a rezolva această limitare a trebuit să mă documentez despre arhitectura software a bibliotecii pocketsphinx și implicit a sistemului Casandra ce se afla pe Raspberry Pi. După ce am aflat în ce fișier se află setată acea variabilă am schimbat valoarea acesteia în funcție de numărul de silabe din cuvântul cheie.

```
pi@raspberrypi:~/PiASR/python $ ls
ColorsDigitsAnswers.py  Config           HttpRequests.py
CommandsEnum.py         emergencyNLU.py  keywords.list
pi@raspberrypi:~/PiASR/python $ cat keywords.list
casandra /1e-30/
```

Fig. 4.2 Variabila pentru keyword spotting

O altă îmbunătățire ar fi aceea că Proiectul Casandra ce se afla pe gitlab nu putea fi clonat oriunde ci doar într-un director anume de pe raspberry pi sau pe alt calculator deoarece acesta nu funcționa negăsindu-si fișierele de care avea nevoie. Acest lucru se datora faptului că în proiect erau folosite căi explicite către directoarele proiectului ca `/home/raspberry/PiASR` , directoare necesare pentru rularea propice a sistemului inteligent Casandra. Prin urmare am rescris bucățile de cod unde se făcea referire explicită la directoarele și fișierele utilizate cu o cale relativă sau cu bucăți de cod în python ce determina prin apelul unei funcții directorul curent în care se află și pe baza lui să determine unde se află celelalte fișiere. Funcția fiind `getcwd()` din biblioteca `os` .

```
from SpeechParser import *
from os import environ, path, system, getcwd
import pyaudio
from pocketsphinx import LiveSpeech
from pocketsphinx.pocketsphinx import *
from tts import save_transcription, default_response
from sphinxbase.sphinxbase import *
import time
CURRENTDIR = getcwd()
MODELDIR = CURRENTDIR.replace("/python", "/models/casandra")
```

Fig. 4.3 Setare Cale Relativa

Pentru a se realiza recunoașterea vocală și înțelegerea comenzilor sistemul Casandra înregistra tot ce se auzea în camera din momentul când a pornit și până se rostea cuvântul cheie ce active sistemul apoi se salva pe cardul microSD fișierul cu tot ce s-a vorbit în încăpere plus cuvantul cheie la final. Pentru a nu ocupa spațiu pe cardul de memorie unde se află scris și sistemul de

operare am implementat o funcție care să păstreze doar ultimii n octeți din toată înregistrearea adică doar cuvântul cheie sau comanda data. În aceeași funcție fac și o redenumire a fișierelor mai simplă, mai exact le redenumesc după timpul în milisecunde masurat de la 1970 pană în zilele noastre apelul funcției fiind `time.time()`, pentru a le accesa ușor dacă este nevoie.

```
def tail_bytes(path, file_name, numof_bytes, time_stamp):
   assert isinstance(path, str) , "FATAL path : That's not a string !"
   assert isinstance(file_name, str) , "FATAL file_name : That's not a string !"
   assert isinstance(numof_bytes, int) , "FATAL numof_bytes : That's not a integer !"
   assert isinstance(time_stamp, int) , "FATAL time_stamp : That's not a integer !"
   _ , file_format = file_name.split(".")
   sizeof_current_f = os.path.getsize(path + "/" + file_name)
   if sizeof_current_f > numof_bytes:
      os.system("tail -c " + str(numof_bytes) + " " + path + "/" + file_name + " > " + path + "/" + str(time_stamp) + "." + file_format)
      os.system("rm " + path + "/" + file_name)
   else:
      print("fisierul " + file_name + "are dimensiunea optima!")
      replaced_string = "mv " + path + "/" + file_name + " " + path + "/" + str(time_stamp)+ "." + file_format
      os.system(replaced_string)
   return
```

Fig. 4.4 Funcția tail_bytes()

Programul ce controla sistemul de iluminat Philips Hue avea o limitare și anume dacă ordonai sistemului Casandra să îți stingă/aprindă lumina într-o anumită cameră sau în toată casa iar lumina sau luminile erau deja stinsă/stinse respective aprinsă/aprinse sistemul seta din nou becurile în modul în care tu îi spuneai la fel și în cazul setării culorilor. Practic modul de interacțiune cu sistemul de iluminat se realiza într-un mod defectuos și nefolositor răspunsul primit fiind unul jenant, neinteligent. De aceea am modificat programul ce interacționează cu acest sistem de iluminat și anume când se cere de către utilizator să se execute o anumită comandă ce priveste strea sitemului de iluminat să se facă o cerere pentru a se verifica starea actuală . Dacă starea actuala nu corespunde cu ce se cere în comandă atunci să se execute comanda dată dacă nu sistemul inteligent Casandra să răspundă cu un mesaj care să indice că starea dorita este deja folosită în casă.

```
def turn_off_my_lights(command, location):

    response = requests.get(url)

    if location != locations.ALL.value[1]:
        if light_status(location, response) == True:
            JsonObj = "{\"on\":false}"
            send_http(JsonObj, location)
            # mod_ans == False -> init command process
            command_to_answer(command, False)
        else:
            # print("Lumina este deja stinsa")
            # mod_ans == True -> exceptional case was thrown
            command_to_answer("lumina este deja stinsă", True)
    elif light_status(locations.KITCHEN.value[1], response) == False and light_status(locations.LIVING_ROOM.value[1], response) == False and
        #print("Toate luminile sunt stinse!")
        # mod_ans == True -> exceptional case was thrown
        command_to_answer("toate luminile in casa sunt stinse", True)
    else:
        JsonObj = "{\"on\":false}"
        send_http(JsonObj, location)
        # mod_ans == False -> init command process
        command_to_answer(command, False)

    return
```

Fig. 4.5 Exmplu de cod în care controlez stingerea luminilor

O altă problemă a sistemului inteligent Casandra este imposibilitatea comunicării cu sistemul Casandra în timp ce modulul radio redă muzică pe boxa. Microfonul fiind aproape de boxă și boxa având un nivel sonor ridicat Casandra nu mai poate înțelege comenzile date de utilizator. De aceea mi-am propus să rezolv această limitare.

Prima abordare în rezolvarea acestei probleme a fost determinarea întârzierii în timp folosind intercorelația dintre cele două semnale . Cel redat pe boxă, semnal curat netrecut prin calea de ecou, și cel înregistrat de microfon. Am rulat zece astfel de experimente cu scopul de a determina și demonstra că avem o întârziere constantă. Ca și semnal de test am generat un semnal audio sinusoidal ce porneste de la frecvența 1000hz și crește în mod constat până la 7000hz (cu un generator de tonuri online). Testul a constat într-un program scris în python ce pornea două fire de execuție în paralel. Un fir de execuție pentru a reda sinusul pe boxă și unul care înregistrează ceea ce se aude pe boxă.
În următoarea fază a experimentului am desenat cu o funcție în python formele de undă a celeor doua semnale si am observat ca semanlul inițial era defazat cu 180 grade pe lângă aceasta am mai observat că semnalul înregistrat avea o perioda în care statea pe zero la începutul înregistrării ceea ce m-a pus în dificultate. Nu știam dacă înregistrase câteva perioade de liniște inițială sau microfonul nu înregistreaza primele esantioane ale semnalui.
În urma unor verificari am dedus că era vorba despre liniște ; practic firul de execuție pe care rula inregistrarea pornea mai repede ca thread-ul pe care rula functia play.
Apoi am făcut funcția de intercorelație (între semnalul generat si cel înregistrat)ca să determin(abscisa primului maxim din funcția de intercorelație). Astfel aflată acea întârziere am suprapus perfect cele două semnale și am făcut diferența dintre cele ele sperând să ajung la liniște sau la un semnal cu o amplitudine considerabil mai mică față de semnalul original.
În urma scăderii nu am obțiut liniște constantă peste tot ci doar obțineam acest lucru pe anumite intervale. Deci în problema întervine și calea de ecou formată din funcția de transfer a boxei , funcția de transfer a aerului și funcția de transfer a microfonului. În final după ce am rulat de zece ori acest experiment nu am obținut întârzierea constantă ea trebuie calculate de fiecare data când se pornește sistemul inteligent Casandra.

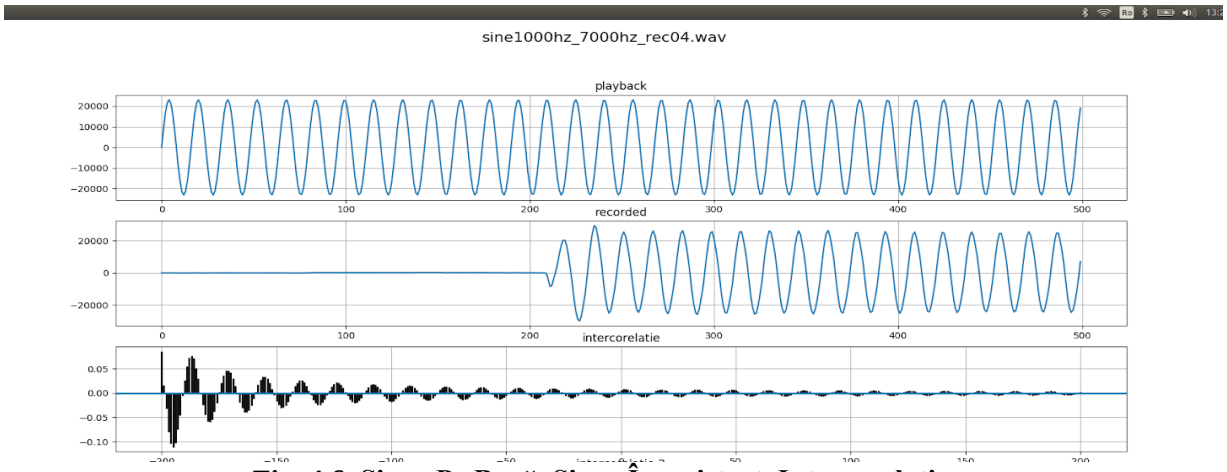


Fig. 4.6 Sinus Pe Boxă, Sinus Înregistrat, Intercorelație

A doua abordare este folosirea unui filtru NLMS care să reducă muzica din fundal. Pentru aceasta avem nevoie să determinăm parametrii filtrului cu care obținem cele mai bune rezultate în majoritatea cazurilor. De aceea am creat o bază de date formată din combinarea a cincizeci de comenzi vocale ce se aflau salvate pe cardul microSD cu cincizeci de fișiere audio salvate de pe streamu-ul a cinci posturi de radio diferite în diferite momente alea zilei. Acest set preluat de la radio de cincizeci de fișiere avea trei variații și anume SNR egal 1 între comenzi si radio, SNR egal cu 0.5 între comenzi și radio și SNR egal 2 între comenzi și radio. Astfel am create 150 cazuri cu care să testez NLMS-ul și să determine cei mai buni parametrii deoarece folosind diferite rapoarte semnal zgomot simulez volumul ce poate fi setat la radio mai tare / mai încet și pe langă acestea în baza de date sunt și interviuri deci alte voci practic am acoperit un nivel destul de vast de secenarii ce se pot întampla la o funcționare obișnuită.

Ca să preiau octeții de pe stream astfel încât să am fișiere de zece secunde am folosit comanda :

```
$ curl --connect-timeout “<stream.radio>” --output post_radio_ora.wav
```

```
65 def create_far_end_signal(s, d, VMR):
66     '''
67         s - cmd
68         d - clear music radio
69         MVR - voice music ratio == (SNR)
70     '''
71     start = 3*16000
72     Px = np.sqrt(np.sum(s * s) / len(s))
73     Py = Px / np.power(10, VMR / 10) / np.sqrt(np.sum(d * d) / len(d))
74     d = d.copy() * Py
75     combined = d.copy()
76     combined[start:start + len(s)] += s
77 
78     return combined, d
79 
```

Fig. 4.7 Funcția cu care combin fișierele audio

După ce am realizat baza de date am căutat o bibliotecă în python care să conțină implementat algoritmul NLMS și am găsit `padasip 1.1.1` . Am instalat biblioteca folosind:

```
$ pip3 install padasip
```

Trecând de acest pas intermediar am implementat diverse metrici care să îmi determine cei mai buni coeficienți ai filtrului și anume pasul de adaptare respectiv ordinul filtrului.

Pentru a ști sigur că metrica aleasă mă va conduce către rezultatele dorite am decis să fac câteva teste inițiale pe un set de date (pe o singură combinație de fișiere ce aveau rapotul SNR egal 1).

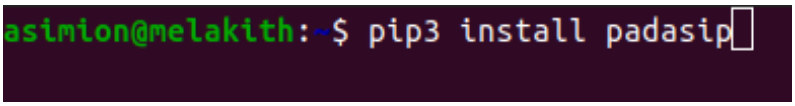


Figure 4.8 Exemplu instalare padasip

Metrica 1 :

$$\Delta P = \left| P(s) - P(\hat{s}) \right| \quad (4.1)$$

unde s este comanda vocală iar $\hat{s}$ semanlul filtrat de NLMS.

Fișier folosit : guerrilla_9_35^1564389290.wav

Formulă Metrică : formula 4.1

ΔP	**Ordin**					
mu	**16**	**32**	**64**	**128**	**256**	**512**
0.001	8.14E-04	5.57E-05	1.48E-03	3.01E-03	5.61E-03	7.15E-03
0.025	0.0010	0.0011	0.0011	0.0012	0.0011	0.0008
0.050	0.0011	0.0010	0.0010	0.0013	0.0012	0.0010
0.075	0.0014	0.0012	0.0010	0.0013	0.0012	0.0011
0.100	0.0016	0.0015	0.0011	0.0013	0.0012	0.0013
0.125	0.0017	0.0016	0.0012	0.0013	0.0012	0.0014
0.150	0.0017	0.0017	0.0013	0.0013	0.0013	0.0015
0.175	0.0018	0.0017	0.0013	0.0012	0.0014	0.0016
0.200	0.0018	0.0017	0.0014	0.0011	0.0014	0.0016
0.225	0.0017	0.0017	0.0014	0.0010	0.0014	0.0016
0.250	0.0017	0.0017	0.0015	0.0009	0.0014	0.0016

Tabel 4.1 Metrica 1

În urma generării fișierului audio cu coeficienții unde valoarea este cea mai mică (adică cel mai bun scor) am observat că această metrică nu ne conduce la rezultatul dorit adică în stânga și în dreapta comenzii să avem valori mici ale eșantioanelor iar acolo unde utilizatorul rostește comanda să se audă clar. Analizând în detaliu ceea ce presupune formula 4.1 am ajuns la concluzia că metrica te poate păcălii deoarece filtrul poate scoate pe toată durata semnalului eșantioane care pot fi de valori mici dar ca valoare a puterii lor sau energiei lor să dea cât valoarea puterii sau energiei eșantioanelor comenzii vocale curate, notat cu s în formula 4.1 .

Deci această metrică o vom abandona.

Așa că am abordat metrica 2 :

$$\frac{\max\{\sum_k^n s(k) * \hat{s}(n-k)\}}{\max\{\sum_k^n s(k) * s(k)\}} \quad (4.2)$$

unde s este comanda vocală iar $\hat{s}$ semanlul filtrat de NLMS.

Fișier utilizat : digifm_11_22^1563877717.wav

Formulă metrică : formula 4.2

Metrica 2	**ordin**					
μ	**16**	**32**	**64**	**128**	**256**	**512**
0.001	0.96	0.94	0.95	0.93	0.89	0.87
0.025	0.96	1.03	1.04	1.02	1.02	1.01
0.050	0.92	1.03	1.06	1.04	1.06	1.01
0.075	0.89	1.006	1.01	1.04	1.03	1.07
0.100	0.87	0.98	0.97	1.04	0.99	1.04
0.125	0.85	0.97	0.95	1.04	0.99	1.02
0.150	0.84	0.95	0.92	1.03	0.93	1.00
0.175	0.80	0.94	0.90	1.03	0.91	0.98
0.200	0.74	0.92	0.89	1.03	0.89	0.96
0.225	0.69	0.91	0.88	1.02	0.87	0.96
0.250	0.65	0.90	0.87	1.02	0.86	0.94

Tabel 4.2 Metrica 2

Prin metrica 2 vreau să determin pentru ce combinație de ordin și pas de adaptare μ obțin un fișier audio filtrat al cărei calitate audio a comenzii este cea mai bună acest lucru presupunând ca eșantioanele cu muzica de la radio din stânga și din dreapta comenzii să fie semnificativ mai mici. După cum se observă valorile depășesc valoarea 1 (aceasta fiind maximul) ceea ce înseamnă că se poate ca fitrul să nu e fie destul de bun încât să nu se adapteze cum trebuie sau formula metricii să nu evalueze corect eșantionale. De aceea pentru ordinul 128 metrica a dus la rezultate ce te induc în eroare. Pentru ordinele 32 si 64 observăm valori bune în medie. Deci metrica 2 singură nu poate să ne conducă spre o singură soluție.Din punctul meu de vedere cel mia bun scor se află la ordin=32 și μ = 0.075 dar nici scorul obținut la ordin=16 și μ=0.025 este de luat în seamă.

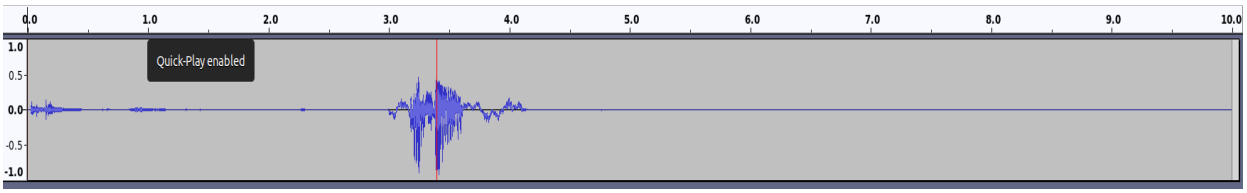


Fig. 4.9 Semnal filtrat cu configurația ordin=16 și μ=0.025

În figura de mai sus se poate observa că scorul de 0.96 este un scor bun ce ne indică faptul că am obținut ceea ce ne dorim în mod teoretic adică în stânga și în dreapta comenzii liniște iar claritatea mesajului vocal este una bună.

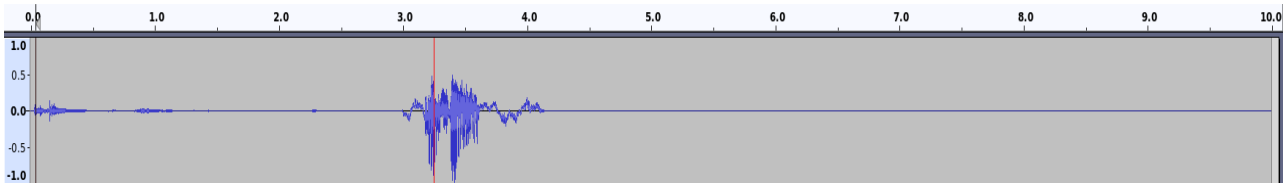


Fig. 4.10 Semnal filtrat cu configurația ordin=32 și μ=0.075

Din punctul meu de vedere cel mai bun scor este 1.006 (este cel mai aproape de 1 chiar dacă valoarea sa depășește limita maximă) obțiut în urma setării filtrului cu parametrii ordin = 32 și μ = 0.075. Dar după cum se observă nu exită diferențe sesizabile între Fig. 4.9 șă Fig. 4.10 . Din punct de vedere auditive există o diferență mică ce mă face să aleg configurația aceasta, probabil doar pentru cazul acesta.

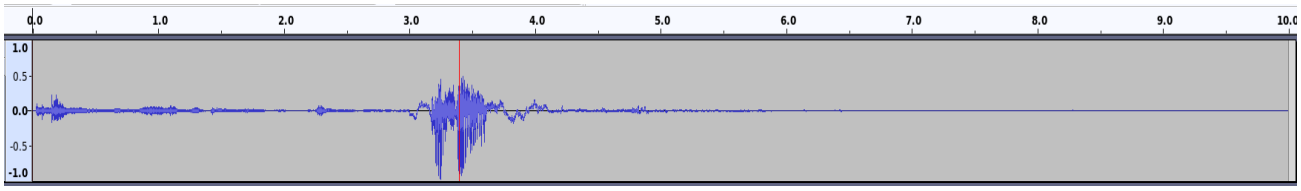


Fig. 4.11 Semnal filtrat cu configurația ordin=128 și μ=0.075

Pentru configurația ordin=128 și μ=0.075 am obținut scorul de 1.04 un scor ce ne-ar putea induce în eroare spunând că este aprox. 1 dar aceasta eroare de 4% se reflectă și în forma de undă a semnalului filtrat obținut în Fig. 4.11 . Muzica este sesizabilă pe fundal chiar dacă se aude mult mai încet ca vocea utilizatorului.

În concluzie această metrică este utilă în combinație cu altă formula, cu alte cuvinte nu poți folosi această metrică ca și criteriu de sine stătător ci trebuie folosită o a doua formulă ce se bazează doar pe valorile eșantioanelor (să fie zero în stânga și în dreapta comenzii vocale).

De aceea am gândit următoarea formulă pentru a crea acel criteriu combinat și am ajuns la metrica 3 :

$$\frac{\sum_{k=start}^{start+N} \hat{s}(k) \cdot \hat{s}(k)}{\sum_{k}^{N} \hat{s}(k) \cdot \hat{s}(k)} \qquad (4.3)$$

unde $\hat{s}$ este semnalul filtrat de NLMS , start reprezintă de la ce eșantion pornește comanda vocală (toate comenzile vocale le-am inserat de la secunda 3) iar N reprezintă lungimea comenzii vocale în eșantioane.

Formula 4.3 se traduce raportul dintre energiei comenzii vocale după filtrarea NLMS-ului și energia întregului semnal filtrat. Astfel că voi obține maxim 1 în cazul în care am zerouri stânga dreapta. Însă această formula îți indică cât de agresiv este filtrul NLMS. Pentru că se obtine aproximativ 1 dar cu pierderea calității audio a comenzii vocale.

De aceea am obținut urmatorul table cu scoruri :

Fișier utilizat : digifm_11_22^1563877717.wav

Formulă metrică : formula 4.3

Metrica 3	ordin					
μ	16	32	64	128	256	512
0.001	0.776	0.656	0.551	0.497	0.474	0.462
0.025	0.994	0.982	0.957	0.916	0.863	0.800
0.050	0.998	0.993	0.982	0.962	0.931	0.889
0.075	0.999	0.996	0.990	0.977	0.957	0.927
0.100	0.999	0.997	0.993	0.985	0.970	0.947
0.125	0.999	0.998	0.995	0.989	0.977	0.959
0.150	0.999	0.998	0.996	0.991	0.982	0.967
0.175	0.999	0.999	0.997	0.993	0.985	0.973
0.200	0.999	0.999	0.997	0.994	0.988	0.977
0.225	0.999	0.999	0.998	0.995	0.990	0.981
0.250	0.999	0.999	0.998	0.996	0.991	0.983

Tabel 4.3 Metrica 3

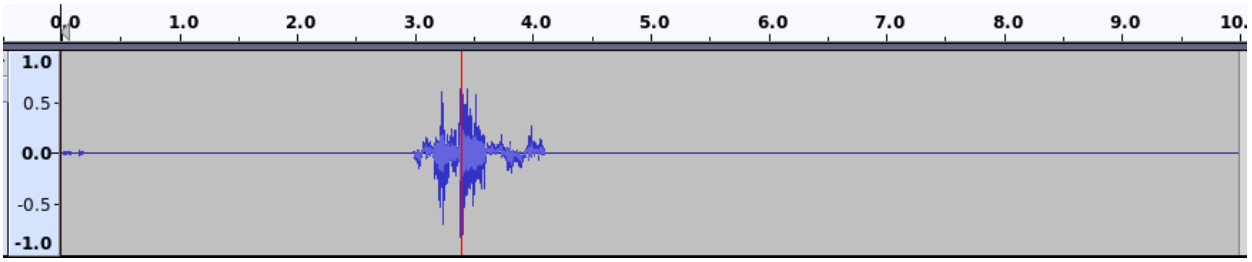


Fig. 4.12 Semnal filtrat cu configurația ordin=16 și μ=0.25

În Fig. 4.12 se observă cât de mult reduce filtrul, aici scorul fiind cel mai bun. După cum am spus acestă metrică te conduce către valori cu care obții liniște dar este afectată calitatea semnalului vocal. Muzica se reduce aproape instant filtru având puțini coeficienți de actualizat la care se mai adaugă și pasul de adaptare mare.

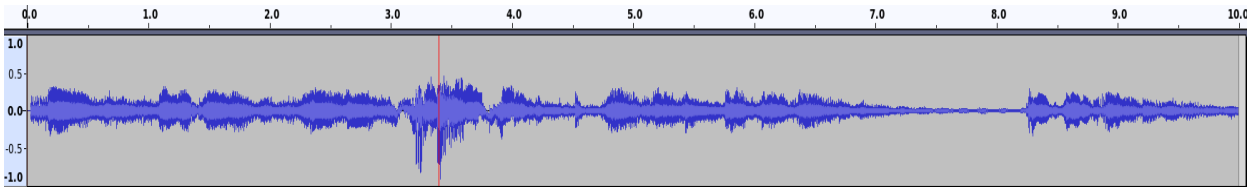


Fig. 4.13 Semnal filtrat cu configurația ordin=512 și μ=0.001

La polul opus avem configurația ce are ca răspuns forma de undă din figura de mai sus. Având cel mai mic pas de adaptare și cel mai mare număr de coeficienți filtrul lucrează mai greu de aceea nu a reușit să reducă semnificativ muzica.

După aceste două încercări am combinat cele două metrici : metrica 1 și metrica 2 . Practic am făcut media aritmetică a scorurilor obținute în cazu celor două metrici.

Metrica 4 la modul general arată :

$$\frac{metrica\ 2 + metrica\ 3}{2} \tag{4.4}$$

unde metrica 2 reprezintă formula 4.2 iar metrica 3 reprezintă formula 4.3 .

Fișier utilizat : digifm_11_22^1563877717.wav

Formulă metrică : formula 4.4

Metrica 4	ordin					
μ	**16**	**32**	**64**	**128**	**256**	**512**
0.001	0.87	0.80	0.75	0.71	0.68	0.67
0.025	0.97	1.01	1.00	0.97	0.94	0.91
0.050	0.96	1.01	1.02	1.00	1.00	0.95
0.075	0.95	1.00	1.00	1.01	0.99	1.00
0.100	0.94	0.99	0.98	1.01	0.98	0.99
0.125	0.93	0.98	0.97	1.01	0.98	0.99
0.150	0.92	0.97	0.96	1.01	0.96	0.98
0.175	0.90	0.97	0.95	1.01	0.95	0.98
0.200	0.87	0.96	0.94	1.01	0.94	0.97
0.225	0.84	0.96	0.94	1.01	0.93	0.97
0.250	0.82	0.95	0.93	1.01	0.93	0.96

Tabel 4.4 Metrica 4

Tabelul 4.4 arată identic cu tabelul 4.2 practic metrica 2 cântarește mai mult iar configurația tabelului rămâne aceeași. Totuși se observă ca obținem în locuri diferite valoarea maximă 1 ceea ce ar însemna că avem mai multe configurații perfecte pentru filtrul NLMS ceea ce este fals. Pentru valori mari ale ordinului începând cu 128 metrica aceasta oferă valori eronate. Însă valorile de la ordinl 32 și 64 care sunt marcate cu verde conduc la rezultate bune la ieșirea filtrului. Cele mai bune rezultate găsindu-se la perchile(ordin, μ) : 32 și 0.025, 32 și 0.05, 32 și 0.075, 64 și 0.025, 64 și 0.05, 64 și 0.075.

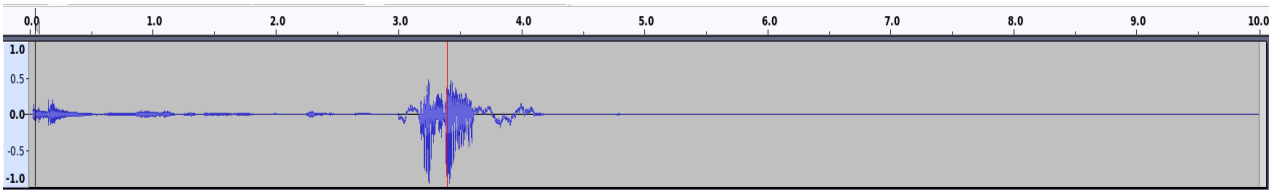


Fig. 4.14 Semnal filtrat cu configurația ordin=32 și μ=0.025

După cum am spus chiar dacă valoarea depașeste cu cu 1% valoarea maximă. Numărul 32 este un ordin rezonabil pentru un filtru NLMS deoarece se observă că în medie valorile obținute se încadrează în plaja de scoruri real posibile. Același lucru îl observăm și la ordinal 64 care pentru pasul de adaptare 0.025 se obține valoarea 1.

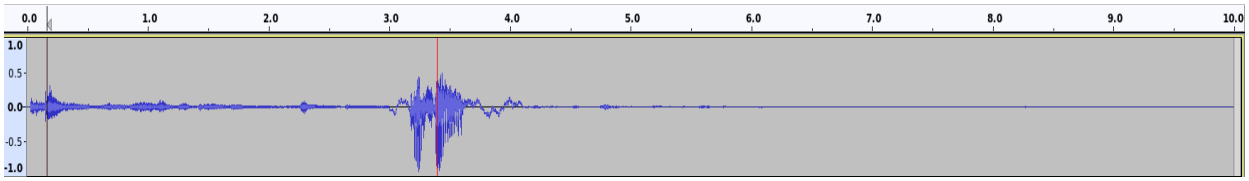


Fig. 4.15 Semnal filtrat cu configurația ordin=64 și μ=0.025

În concluzie metrica 4 nu m-a condus către un singur rezultat pentru această combinație de fișiere. Poate pe un set mare de valori poate să fie eficientă dar momentan va rămâne doar o încercare.

Metrica 5:

$$\frac{\sqrt{\sum(\hat{s}-s)\cdot(\hat{s}-s)}}{\sum s\cdot s} \tag{4.5}$$

unde s este comanda vocală iar $\hat{s}$ este semnalul filtrat de NLMS.

Fișier utilizat : digifm_11_22^1563877717.wav

Formulă metrică : formula 4.5

Metrica 5	ordin					
μ	**16**	**32**	**64**	**128**	**256**	**512**
0.001	0.0357	0.0499	0.0702	0.0870	0.1004	0.1123
0.025	0.0189	0.0175	0.0208	0.0262	0.0328	0.0412
0.050	0.0249	0.0217	0.0229	0.0266	0.0303	0.0358
0.075	0.0289	0.0249	0.0250	0.0297	0.0305	0.0353
0.100	0.0320	0.0276	0.0273	0.0328	0.0317	0.0358
0.125	0.0345	0.0300	0.0294	0.0357	0.0332	0.0370
0.150	0.0366	0.0322	0.0314	0.0383	0.0348	0.0384
0.175	0.0379	0.0343	0.0333	0.0408	0.0364	0.0399
0.200	0.0386	0.0363	0.0350	0.0432	0.0379	0.0414
0.225	0.0396	0.0382	0.0366	0.0455	0.0394	0.0429
0.250	0.0406	0.0399	0.0382	0.0477	0.0394	0.0445

Tabel 4.5 Metrica 5

Privind rezultatele din tabelul 4.5 se observă că metrica 5 coduce la un singur rezultat optim și anume pentru configurația ordin egal 32 și μ egal cu 0.025. A cărei formă de undă am prezentat-o și anticipat-o în Fig. 4.14 în cadrul analizei metricii 4. Totuși se observă ca avem valori apropriate pentru ordinele 16, 32 și 64 pentru un μ cu valoarea 0.025. Așa că am luat o altă combinație de fișiere pentru a test-o cu metrica 5 și să obțin un nou tabel de scoruri.

Fișier utilizat : eurofm_21_21^1563956775.wav

Formulă metrică : formula 4.5

Metrica 5	ordin					
μ	16	32	64	128	256	512
0.001	0.0136	0.0174	0.0221	0.0276	0.0319	0.0356
0.025	0.0169	0.0166	0.0168	0.0171	0.0179	0.0205
0.050	0.0196	0.0213	0.0214	0.0201	0.0204	0.0217
0.075	0.0218	0.0221	0.0238	0.0225	0.0225	0.0238
0.100	0.0232	0.0236	0.0252	0.0244	0.0244	0.0259
0.125	0.0243	0.0249	0.0260	0.0259	0.0260	0.0277
0.150	0.0251	0.0260	0.0267	0.0272	0.0275	0.0292
0.175	0.0258	0.0271	0.0274	0.0283	0.0288	0.0303
0.200	0.0265	0.0281	0.0280	0.0291	0.0299	0.0313
0.225	0.0271	0.0287	0.0286	0.0299	0.0310	0.0322
0.250	0.0277	0.0291	0.0292	0.0306	0.0321	0.0330

Tabel 4.6 Metrica 5 fișier EuropaFm

Aici observăm că s-a modificat valoarea unde avem cel mai bun răspuns al filtrului. Iar forma de undă arată ca în figura de mai jos.

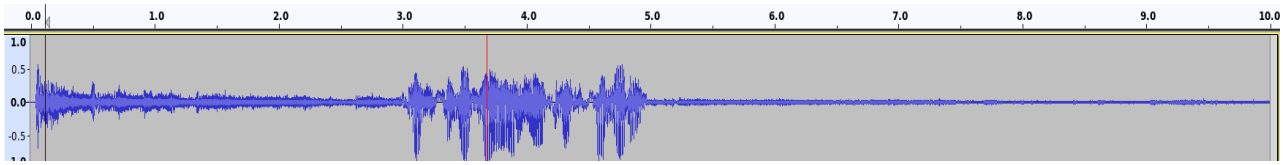


Fig. 4.16 Semnal filtrat cu configurația ordin=16 și μ=0.001

În Fig. 4.16 se observă că filtrul este lent din cauza pasului mic de adaptare iar reducerea muzicii se face mai lent. Dar dupa cum se observă nu valoarea indicate de scor nu artă că acest fișier este cel mai bun. Următoarea cea mai mica valoarea se află la ordin 32 și μ 0.025 .

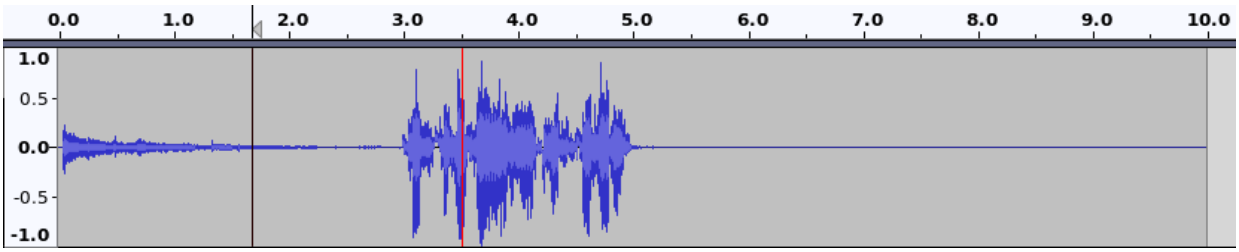


Figure 4.17 Semnal filtrat cu configurația ordin=32 și μ=0.025

Prin acest fapt se observă ca nici această metrică este ideală dar conduce la rezultate relative bune. De aceea aleg să testez toată baza de date cu metrica 5 și să fac o medie a sumei valorilor obținute pentru fiecare configurație de parametrii.

Metrica 5	ordin					
μ	**16**	**32**	**64**	**128**	**256**	**512**
0.001	0.0211	0.0280	0.0365	0.0434	0.0492	0.0540
0.025	0.0170	0.0176	0.0188	0.0207	0.0238	0.0281
0.050	0.0206	0.0208	0.0215	0.0223	0.0239	0.0271
0.075	0.0230	0.0234	0.0239	0.0244	0.0256	0.0281
0.100	0.0249	0.0254	0.0261	0.0263	0.0274	0.0295
0.125	0.0265	0.0270	0.0275	0.0278	0.0289	0.0308
0.150	0.0277	0.0284	0.0288	0.0292	0.0303	0.0321
0.175	0.0286	0.0294	0.0298	0.0303	0.0315	0.0331
0.200	0.0294	0.0303	0.0307	0.0314	0.0325	0.0341
0.225	0.0301	0.0311	0.0315	0.0322	0.0334	0.0349
0.250	0.0307	0.0317	0.0322	0.0330	0.0341	0.0357

Tabel 4.7 Metrica 5 peste baza de date

Rezultatele generate în tabelul 4.7 arată că în medie cea mai bună configurație pentru parametrii ai filtrului NLMS se află la ordin=16 și μ=0.025 . Pentru metrica 2 această configurație avea o valoarea bună 0.96 iar în Fig. 4.9 vedem și rezultatele pentru fișirul de la DigiFm fișier audio ce exprimă rezultate favorabile pentru ceea ce doresc să obțin.

În figura de mai jos voi prezenta fișierul de ieșire al NLMS-ului folosind fișierul de intrare de la EuroFM dar pentru configurația determinată în urma rulării metricii 5 peste tot setul de fișiere.

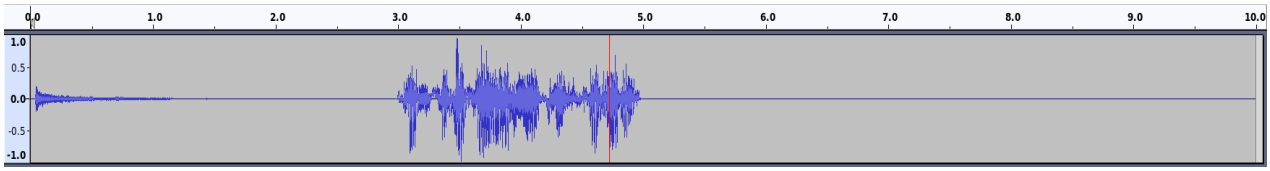


Fig. 4.18 Semnal filtrat cu configurația ordin=32 și μ=0.025

După cum se observă rezultatul acesta este mai bun decât configurația ordin=16 și μ=0.001 obținând o reducere rapidă a muzicii de fundal și păstrând o calitate audio bună pe porțiunea comenzii vocale.

Pentru a face o comparație am generat scoruri pentru întreaga bază de date folosind metrica 4. Când am analizat metrica 4 am zis că generează valori bune dar pentru anumite fișiere te poate induce in eroare de aceea aleg să o rulez pe întreg setul de date pentru a determina dacă mă conduce către un rezultat final valid și pentru a verifica dacă metrica 5 îmi generează cea mai bună soluție.

Metrica 4	ordin					
μ	16	32	64	128	256	512
0.001	0.8756	0.8218	0.7695	0.7259	0.6860	0.6593
0.025	0.8887	0.8800	0.8844	0.8566	0.8455	0.8229
0.050	0.8462	0.8510	0.8645	0.8435	0.8400	0.8276
0.075	0.8150	0.8294	0.8407	0.8215	0.8236	0.8154
0.100	0.7918	0.8059	0.8180	0.8020	0.8059	0.8006
0.125	0.7728	0.7860	0.7937	0.7838	0.7890	0.7858
0.150	0.7552	0.7696	0.7735	0.7687	0.7745	0.7725
0.175	0.7395	0.7552	0.7569	0.7557	0.7624	0.7587
0.200	0.7263	0.7424	0.7426	0.7447	0.7507	0.7461
0.225	0.7152	0.7310	0.7301	0.7346	0.7397	0.7351
0.250	0.7055	0.7206	0.7195	0.7255	0.7300	0.7249

Tabel 4.8 Metrica 4 peste baza de date

Comparând tabelul 4.7 și tabelul 4.8 se observă că maximul se găsește în același loc însă următorul maxim apropiat se află la ordin=64 și μ=0.025 între cele două fiind o diferență 0.0043. Deci este o diferență fină între ele. Același fenomen se întâmplă și în tabelul generat de metrica 5. Făcând și o privire de ansamblu, configurațiile celor două tabele sunt asemănătoare dar tind să aleg să mă ghidez după metrica 5 deoarece de la bun început a generat scorul care m-a condus către rezultatul favorabil. Rezultatul favorabil înseamnă liniște în stânga și în dreapta comenzii vocale iar pe porțiunea unde se află vocea utilizatorului să am o calitate audio bună adică să se audă clar vocea.

În urma acestei învestigații am ajuns la următoarele concluzii:

1) Din figurile cu răspunsul filtrului rezultă că cea mai bună soluție ca sistemul să răspundă în continuare la comenzi în timpul în care utilizatorul ascultă radioul este implementarea unui filtrul NLMS în timp real.
2) Cea mai bună configurație pentru parametrii filtrului NLMS (ordin, pas de adaptare) este 16, 0.025

CAPITOL 5

CONCLUZII

5.1 CONCLUZII GENERALE

În urma realizării practice a proiectului de diploma am dus la final o parte din o parte din ceea ce mi-am propus. Dificultați punându-mi modulul NLMS în timp real. Pentru acesta realizând doar o investigație ce m-a condus la faptul că implementarea lui în sistemul Casandra va elimina limitarea ce apare în timpul rulării modului radio, limitare ce este reprezentată de incapacitatea sistemului sistemului de a asculta comenzi.

Scopul extinderii sistemului deja existent fiind îmbunătățirea algorimului de detecție a cuvântului cheie pentru ca sistemul să nu se mai activeze din senin ,să nu se activeze la timp sau sa nu se activeze. Astfel comunicarea cu sistemul inteligent să fie una coerentă iar controlul casei să se realizeze doar la dorința utilizatorului. Să existe un mod inteligent de a interacționa cu sistemul de iluminat astfel când se dorește să se schimbe starea becurilor (aprins/stins) sau culoarea acestora să se verifice dacă deja cerința utilizatorului este deja îndeplinită iar sistemul să anunțe acest fapt. Și pe lângă acestea să refactorizez codul de python deja existent pentru a rezolva bug-uri de sistem.

Contribuții la sistemul de casă inteligentă Casandra

Contribuțiile personale aduse acestui proiect sunt :

- Refactorizarea codului de python deja existent
- Îmbunătățirea algoritmului de detecție a cuvântului cheie
- Îmbunătățirea intereacțiunii cu sistemului de iluminat astfel încât să se realizeze acțiuni ascupra lui doar dacă este cazul

Pe lângă acestea se adaugă investigația și testarea unui filtrul NLMS pentru a determina parametrii filtrului ce trebuie folosiți în timpul funcționării modului radio pentru ca sistemul să poată să răspundă în continuare la comenzi.

Acest proiect a reprezentat o provocare pentru mine și m-a ajutat să îmi dezvolt aptitudini noi pe care o să le pot folosi mai departe în cariera. Cu ocazia acestui proiect am învățat mai multe depre : cum să utilizez o distribuție de linux și prelucrarea digitală a semnalelelor. Pe lângă acestea am acumulat informații depre tehnici avansate de prelucrarea semnalelor digitale.

5.2 POSIBILITĂȚI VIITOARE DE DEZVOLTARE

La categoria posibilități de dezvoltare imediate ar implementarea filtrului NLMS în timp real ce trebuie atașat modului radio. Pentru asta ai nevoie de octeții ce vin de pe fluxul de date online radioului, acesta fiind folosit ca semnal dorit (desire), iar pentru semnalul de intrare x al filtrului NLMS octeții ce sunt înregistrați de microfon. Apoi răspunsul să fie dat mai departe sistemului să prelucreze comenzile.

Voi prezenta pe scurt posibilităţile de extindere ale unei case inteligente comandată prin voce:

- Conectarea sistemului la un server REST care să ofere informații meteo
- Conectarea sistemului la un server REST care să ofere informații despre competiții sportive (meciuri de fotbal, meciuri de baschet) cum ar fi scorurile înregistrate.
- Conectarea sistemului la platforma YouTube pentru ca utilizatorul să asculte ce melodie nu ce melodii sunt în listele radiourilor.

Pe lângă acestea o îmbunătățire majoră de maximă necesitate ar fi securizarea sistemului inteligent Casandra importiva atacurilor cibernetice. În primul rând ar trebui testate vulnerabilitățile sistemului Casandra cu o distribuție de testare și anume Khali Linux. O dată ce vulnerabilitățile sunt detectate va trebuii găsirea unor soluții de securizare.

BIBLIOGRAFIE

[1] R. P. Foundation, "About Us," [Online]. Available: https://www.raspberrypi.org/about/.

[2] E. Upton, "Ten millionth Raspberry Pi, and a new kit," 8 September 2016. [Online]. Available: https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/.

[3] R. P. Foundation, "Product Raspberry Pi 3 B," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b.

[4] R. P. Foundation, "Welcome to Raspbian," [Online]. Available: https://www.raspbian.org/.

[5] S. Technologie, "UB1 - USB Boundary Microphone," [Online]. Available: http://www.samsontech.com/samson/products/microphones/usb-microphones/ub1/.

[6] Melarox, "Boxa portabila Gembird SPK-103-W, 2W, Alb," [Online]. Available: https://www.melarox.ro/boxa-portabila-gembird-spk-103-w.html.

[7] Z. Alliance, "Zigbee," [Online]. Available: https://zigbeealliance.org/solution/zigbee/.

[8] json.org, "Introducing JSON," [Online]. Available: https://www.json.org/json-en.html.

[9] KNX, "A brief introduction to KNX," [Online]. Available: https://www.knx.org/knx-en/for-professionals/What-is-KNX/A-brief-introduction/index.php.

[10] Linux.Com, "What is Linux?," [Online]. Available: https://www.linux.com/what-is-linux/.

[11] Upasana, "20 Linux Commands You'll Actually Use In Your Life," [Online]. Available: https://www.edureka.co/blog/linux-commands/.

[12] C. Hope, "Linux scp command," 16 11 2019. [Online]. Available: https://www.computerhope.com/unix/scp.htm.

[13] D. McKay, "How to use tail command on Linux," [Online]. Available: https://www.howtogeek.com/481766/how-to-use-the-tail-command-on-linux/.

[14] Vim, "Vim the editor," [Online]. Available: www.vim.org.

[15] V. Commands, "Vim editor commands," [Online]. Available: https://www.radford.edu/~mhtay/CPSC120/VIM_Editor_Commands.htm.

[16] Gitlab, "Gitlab University," [Online]. Available: https://docs.gitlab.com/ee/university/#beginner.

[17] git, "Reference," [Online]. Available: https://git-scm.com/docs.

[18] Github, "GitHub GIT Cheat Sheet," [Online]. Available: https://education.github.com/git-cheat-sheet-education.pdf.

[19] W. Machine, "Important Features Of Python," [Online]. Available: https://web.archive.org/web/20120623165941/http://cutter.rexx.com/~dkuhlman/python_book_01.html#introduction-python-101-beginning-python.

[20] Spyder, "Spyder-Ide Overview," [Online]. Available: https://www.spyder-ide.org/.

[21] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach and T. (. 1. Berners-Lee, "RFC2616 - Hypertext Transfer Protocol -- HTTP/1.1," [Online]. Available: https://tools.ietf.org/html/rfc2616#page-7.

[22] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach and T. Berners-Lee, "RFC2616 - Hypertext Transfer Protocol – HTTP/1.1," 1999. [Online]. Available: https://tools.ietf.org/html/rfc2616#section-1.4.

[23] "MIME," [Online]. Available: https://en.wikipedia.org/wiki/MIME.

[24] M. W. Doc, "HTTP response status codes," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status.

[25] N. W. Group, "HTTP Over TLS," May 2000. [Online]. Available: https://tools.ietf.org/html/rfc2818.

[26] W3.ORG, "Web Services Architecture," [Online]. Available: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest.

[27] R. T. Fielding, "Representational State Transfer (REST)," [Online]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

[28] R. A. Tutorial, " REST API Security Essentials," [Online]. Available: https://restfulapi.net/security-essentials/.

[29] MDP, "User’s Manual - Music Daemon Player," [Online]. Available: https://www.musicpd.org/doc/html/user.html#the-client.

[30] "mpc(1) - Linux manual," [Online]. Available: https://linux.die.net/man/1/mpc.

ANEXE

```
"""
File : test_NLMS.py

@author : asimion
"""
from scipy.io import wavfile
import numpy as np
import time
import util

PATH_RADIO_WAVs =
"/home/asimion/Proiect_Licenta/database/d_signal/radio10s_1/"
PATH_CMD_WAVs =
"/home/asimion/Proiect_Licenta/database/cmd_clean_wav/"
PATH_COMBINED = "/home/asimion/Proiect_Licenta/database/combined/"
PATH_D = "/home/asimion/Proiect_Licenta/database/D_signals/"
FILE_OUT = "out.txt"

N = 3 * 16000
MU_ARR = [0.001 , 0.025, 0.050, 0.075, 0.100, 0.125, 0.150, 0.175,
0.200, 0.225, 0.250]
ORDER_ARR = [16, 32, 64, 128, 256, 512]
mat_scor_db = len(MU_ARR) * [[0] * len(ORDER_ARR)]
mat_scor_db = np.matrix(mat_scor_db, dtype="float")

def compute_NLMS_table_score(fnc_test, cmd, x, d):

    e = 0
    mat_scor = []
    for mu in MU_ARR:
        row_scor = []
        for order in ORDER_ARR:
            print(f'mu : {mu} -- Order : {order} ')
            _, e = util.NLMS_padasip(d, x, order, util.aprox_zero, mu)
            scor = fnc_test(cmd, e, len(cmd))
            row_scor.append(scor[0])
            print(f'scor : {scor}')
        mat_scor.append(row_scor)
    mat_scor = np.matrix(mat_scor)

    return mat_scor

'''----- main ------'''

#### matrix score database sequentially #####
```

```
cmd_files = "cmd.txt"
d_files = "radio.txt"
cmd_signals = []
d_signals = []

with open(cmd_files, "r") as filehandler:
    filecontent = filehandler.readlines()
    for line in filecontent:
        cmd_signals.append(line.rstrip())

with open(d_files, "r") as filehandler:
     filecontent = filehandler.readlines()
     for line in filecontent:
         d_signals.append(line.rstrip())

num_of_files = len(cmd_signals)
t0 = time.time()
for i in  range(num_of_files):
    for vmr in [-3, 0, 3]:
        fs_d, data_d = wavfile.read(PATH_RADIO_WAVs +
d_signals[i])#radio
        fs_cmd, cmd = wavfile.read(PATH_CMD_WAVs +
cmd_signals[i])#vocal cmd
        d_name = d_signals[i].split(".")[0]
        cmd_name = cmd_signals[i].split(".")[0]
        data_d = util.convert_int16_to_float64(data_d)
        data_x, data_d = util.create_far_end_signal(cmd , data_d ,
vmr)#radio+voice
        print(i)
        wavfile.write(filename=PATH_COMBINED + d_name + "^" + cmd_name
+ "_VMR_" + str(vmr) + ".wav", rate=16000, data=data_x)
        wavfile.write(filename=PATH_D + d_name + "_VMR_" + str(vmr) +
".wav", rate=16000, data=data_d)
        mat_scor_db +=
compute_NLMS_table_score(util.compute_measure_l2_mod, cmd, data_x,
data_d)

mat_scor_db = mat_scor_db / 150
t1 = time.time()
print(f"time : {(t1-t0) // 60} min")
util.print_matrix(mat_scor_db)
util.print_matrix_in_file(mat_scor_db, FILE_OUT)
```

```
"""
File : util.py

Created on Wed Jul 15 13:29:55 2020

@author: asimion
"""

import padasip as pa
import numpy as np
import math as m
from tqdm import tqdm

aprox_zero = 1e-5

def print_matrix(mat):

    for i in mat:
        for j in i:
            print(j)
        print()

def print_matrix_in_file(filename, mat):

    with open(filename) as f:
        for line in mat:
            f.savetxt(f, line, fmt='%.5f')

def convert_int16_to_float64(x):

    return x / 2**15

def compute_abs_max(my_list):

    abs_vals = [abs(i) for i in my_list]

    return max(abs_vals)

def scale_range(s):
    ''' aprox conversion int to float '''
    maxim = compute_abs_max(s)

    return [i/maxim for i in s]
```

```
def cpy_right(x):
    '''
        this function simulates a circular buffer
    '''
    for i in range(len(x)-1, 0, -1):
        x[i] = x[i-1]

    return x

def right_padding_signal(x0, length):

    padding = [0] * length
    x0 = np.concatenate((x0, padding))

    return x0

def left_padding_signal(x0, length):

    padding = [0] * length
    x0 = np.concatenate((padding, x0))

    return x0

def create_far_end_signal(s, d, VMR):
    '''
        s - cmd
        d - clear music radio
        MVR - voice music ratio == (SNR)
    '''
    start = 3*16000
    Px = np.sqrt(np.sum(s * s) / len(s))
    Py = Px / np.power(10, VMR / 10) / np.sqrt(np.sum(d * d) / len(d))
    d = d.copy() * Py
    combined = d.copy()
    combined[start:start + len(s)] += s

    return combined, d
```

```
def nlms(x, d, L ,alpha, delta):
    '''
        x - input NLMS
        d - desire NLMS
        L - filter order
        alpha - learning rate (adaption step)
        delta - variable that controls the division to zero
    '''
    N = len(x)
    e = np.zeros(N)
    y_est = np.zeros(N)
    h_est = np.zeros(L)
    xn = np.zeros(L)

    for n in tqdm(range(0, N)):
        xn = np.concatenate(([x[n]], xn[0:L - 1]))
        y_est[n] = np.dot(xn, h_est)
        e[n] = d[n] - y_est[n]

        step = alpha / (np.dot(xn, xn) + delta)
        h_est = h_est + step * e[n] * xn

    return -e, h_est

def nlms_realtime(x, d, xn, h_est, L, alpha, delta):
    '''
        x - input NLMS
        d - desire NLMS
        L - filter order
        alpha - learning rate (adaption step)
        delta - variable that controls the division to zero
        xn – circular buffer
        h_est – filter coef.
    '''
    e = 0
    y_est = 0

    xn = np.concatenate(([x], xn[0:L - 1]))
    y_est = np.dot(xn, h_est)
    e = d - y_est

    step = alpha / (np.dot(xn, xn) + delta)
    h_est = h_est + step * e * xn

    return -e, h_est, xn
```

```
def NLMS_padasip(x, d, N, a_eps, u):
    '''
        x - input NLMS
        d - desire NLMS
        N - filter order
        u - learning rate (adaption step)
        a_eps - variable that controls the division to zero
    '''
    x_len = len(x)
    d_len = len(d)

    if(x_len < d_len):
        dif = d_len - x_len
        padding = [a_eps] * dif
        x = np.concatenate((x, padding))
    elif x_len > d_len:
        dif = x_len - d_len
        padding = [a_eps] * dif
        d = np.concatenate((d, padding))

    f = pa.filters.FilterNLMS(n = N, mu = u, eps = a_eps, w = "zeros")
    x_f = [N * [0]]
    idx = 0
    out = []
    err = []
    for x_i in x:
        x_f[0] = np.concatenate(([x_i], x_f[0][1:N]))
        di = np.array([d[idx]])
        y, e, w = f.run(di, x_f)
        out.append(y)
        err.append(e)
        x_f[0] = cpy_right(x_f[0])
        idx = idx + 1

    err = 1/compute_abs_max(err) * err
    out = 1/compute_abs_max(out) * out
    err=np.array(err, dtype="float")
    out=np.array(out, dtype="float")

    return out, err

def compute_energy(signal):

    return sum(np.power(signal, 2))
```

```
def compute_power(signal):

    return compute_energy(signal) / len(signal)

def delta_power(s, s_hat):
    '''
        s - cmd
        s_hat - out nlms err
    '''
    s = convert_int16_to_float64(s)
    s_hat = np.transpose(s_hat)
    s = left_padding_signal(s, len(s))
    s = right_padding_signal(s, len(s_hat[0]) - len(s))

    return abs( compute_power( s ) - compute_power( s_hat[0]) )

def delta_power_windowing(s, s_hat):
    '''
        s - cmd
        s_hat - out nlms err
    '''
    delta_pow = []
    s_len = len(s)
    s_h_len = len(s_hat)
    s_h = []
    s_hat = np.transpose(s_hat)
    if s_h_len % s_len != 0:
        zeros_padding = ( (m.ceil(s_h_len / s_len) * s_len ) -
s_h_len) * [0]
        print(s_hat.shape)
        s_h = np.concatenate((s_hat[0], zeros_padding))
        s_h_len = len(s_h)
    else:
        print(s_hat.shape)
        s_h = s_hat[0]
        s_h_len = len(s_h)
    i = s_len
    start = 0
    print(f's_h_len : {s_h_len}')
    while i <= s_h_len:
        window_s_hat = []
        print(f'{start} # {i}')
        for j in range(start, i, 1):
            window_s_hat.append(s_h[j])
        delta_pow.append( delta_power(s, window_s_hat ) )
        start = i
```

```
        i = i + s_len

    return min(delta_pow)

def max_corr(s, s_hat):
    '''
        s - cmd
        s_hat - out nlms err
    '''
    cross_corr = np.correlate(s, s_hat, "full")
    print(f" max cross corr  : {max(cross_corr)}")
    print(f" min cross corr  : {min(cross_corr)}")
    max_val = compute_abs_max(cross_corr)
    #max_val_neg = (-1) * max_val
    #idx_max_val = [i for i in range(len(cross_corr)) if cross_corr[i]
== max_val_neg]
    #print(idx_max_val)

    return max_val

def xcorr_windowing(s, s_hat):
    '''
        s - cmd
        s_hat - out nlms err
    '''
    x_corr = []
    s_len = len(s)
    s_h_len = len(s_hat)
    s = convert_int16_to_float64(s)
    s_h = []
    s_hat = np.transpose(s_hat)
    if s_h_len % s_len != 0:
        zeros_padding = ( (m.ceil(s_h_len / s_len) * s_len ) -
s_h_len) * [0]
        s_h = np.concatenate((s_hat[0], zeros_padding))
    else:
        s_h = s_hat[0]
        s_h_len = len(s_h)
    i = s_len
    start = 0
    while i <= s_h_len:
        window_s_hat = []
        for j in  range(start, i, 1):
            window_s_hat.append(s_h[j])
        plt.xcorr(s, window_s_hat)
        max_val = max_corr(s, window_s_hat)
```

```
        x_corr.append(max_val)
        start = i
        i = i + s_len

    return max(x_corr)

def get_max_xcorr(s, s_hat, n_padd_left):
    '''
        s - cmd
        s_hat - out nlms err
        n_padd_left - num of samples for left padding s
    '''
    #s = convert_int16_to_float64(s)
    acorr = np.correlate(s, s)
    s_hat = np.transpose(s_hat)
#    if len(s) != len(s_hat[0]):
#        print("Modific lungimea lui s")
#        s = left_padding_signal(s, n_padd_left)
#        s = right_padding_signal(s, len(s_hat[0]) - len(s))
    max_val = max_corr(s, s_hat[0]) / acorr

    return max_val

def division_cpv_cpr(s_hat, N):
    '''
        s_hat - out nlms err
        N - num samples ( corresponding for 3s)
    '''
    N_start = 3 * 16000
    s_hat = np.transpose(s_hat)
    cpv = compute_power(s_hat[0][N_start:N_start + N])#compute power
voice
    cpr = compute_power(s_hat[0][0:N_start]) +
compute_power(s_hat[0][N_start + N : ])#compute power rest
    div_scor = cpv / cpr

    return div_scor

def division_pvoice_pshat(s_hat, N):
    '''
        s_hat - out nlms err
        N - num samples ( corresponding for 3s)
    '''
    N_start = 3 * 16000
    s_hat = np.transpose(s_hat)
    pvoice =  compute_power(s_hat[0][N_start:N_start + N])
```

```
    pshat = pvoice + compute_power(s_hat[0][0:N_start]) +
compute_power(s_hat[0][N_start + N : ])

    return pvoice / pshat

def division_evoice_eshat(s_hat, N):
    '''
        s_hat - out nlms err
        N - num samples ( corresponding for 3s)
    '''
    N_start = 3 * 16000
    s_hat = np.transpose(s_hat)
    evoice = compute_energy( s_hat[0][N_start:N_start + N] )#compute
power voice
    eshat = compute_energy( s_hat[0][:] )

    return evoice / eshat

def compute_measure_l2(s, s_hat):
    '''
        s - clean cmd
        s_hat - out nlms err
    '''
    n_padd_left = 16000 * 3#3s
    #s = convert_int16_to_float64(s)
    s_hat = np.transpose(s_hat)
    s = left_padding_signal(s, n_padd_left)
    s = right_padding_signal(s, len(s_hat[0]) - len(s))
    diff = s_hat[0][:] - s

    return np.sqrt(sum(diff ** 2))

def compute_measure_l2_mod(s, s_hat):
    '''
        s - cmd
        s_hat - out nlms err
    '''
    p_s = compute_energy(s)
    l2_ecmd_norm = compute_measure_l2(s, s_hat) / p_s

    return l2_ecmd_norm
def compute_m23(s, s_hat, len_cmd):
    ''' s – cmd ; s_hat - out nlms err '''
    return (get_max_xcorr(s, s_hat, 48000) +
division_evoice_eshat(s_hat, len_cmd)) / 2
```

```
"""
File : filter_nlms.py

@author : asimion
"""
from scipy.io import wavfile
import util

n = 16
mu = 0.025

fsx, data_x =
wavfile.read('/home/asimion/Proiect_Licenta/combined.wav')#combined
fs_d, data_d =
wavfile.read('/home/asimion/Proiect_Licenta/radio1_5min.wav')#radio
#fs_cmd, cmd =
wavfile.read('/home/asimion/Proiect_Licenta/database/cmd_clean_wav/156
3956775.wav')#vocal cmd
#data_d = util.convert_int16_to_float64(data_d)
#data_x, data_d = util.create_far_end_signal(cmd , data_d , 0)

_ , e = util.NLMS_padasip(data_d, data_x, n, util.aprox_zero, mu)

wavfile.write(f"/home/asimion/e_mu{mu}_order{n}_offline.wav",
rate=16000, data=e)
```