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

Contribuţii la un sistem de casă inteligentă comandabilă prin voce

Proiect de diplomă

Prezentat ca cerinţă parţială pentru obţinerea titlului de Inginer
în domeniul Electronică şi Telecomunicaţii
programul de studii de licenţă Electronică Aplicată

Conducător științific:
Conf.Dr.Ing.Horia Cucu

Absolvent:
Manolescu Vlad-Ștefan

București
2018

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

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **MANOLESCU D.L. Vlad-Ştefan , 442B**

1. Titlul temei: Contributii la un sistem de casa inteligenta comandabila prin voce

2. Descrierea contribuţiei originale a studentului (în afara părţii de documentare):
Proiectul de fata reprezinta o extindere a sistemului hardware-software demostrativ pentru o casa inteligenta comandabila prin voce dezvoltat de laboratorul Speech and Dialogue din UPB. Sistemul actual cuprinde: un modul de recunoastere de comenzi vocale in limba romana, un modul de interpretare si executie a comenzilor, un modul de sinteza a vorbirii pornind de la text. Contributia originala a studentului consta in dezvoltarea unui modul software aditional de interpretare si executie a comenzilor. Acest modul va detecta si va intepreta comenzile vocale ce se refera la (i) iluminarea din casa inteligenta si (ii) sistemul audio din casa inteligenta, de exemplu: "casandra, aprinde lumina verde in sufragerie", "casandra, vreau mai mult albastru in dormitor", "casandra, creste intensitatea luminii in bucatarie", "casandra stinge luminile de la parter", "casandra pune-mi Europa FM", "casandra, da muzica mai tare", "casandra, schimba postul de radio", etc. Executia propriu-zisa a comenzilor privind iluminarea va presupune transmiterea unor cereri de tip HTTP (POST sau PUT) catre serviciul REST de comanda a luminii, pus la dispozitie de Philips Hue. Executia propriu-zisa a comenzilor privind sistemul audio va presupunea crearea unui modul software aditional, rezident pe serverul central al casei inteligente, care va putea prelua un flux audio disponibil online, va putea sa redea acest flux prin sistemul audio al casei inteligente si va putea modifica volumul sonor.

3. Resurse folosite la dezvoltarea proiectului:
Sistem hardware-software demostrativ pentru o casa inteligenta comandabila prin voce. Sistemul cuprinde urmatoarele module software: modul de recunoastere de comenzi vocale in limba romana, modul de interpretare si executie a comenzilor, modul de sinteza a vorbirii pornind de la text. Sistemul cuprinde urmatoarele module hardware: sistem de iluminare inteligent Philips Hue (controller Philips Hue

4. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline:
Programarea Calculatoarelor, Structuri de Date si Algoritmi, Microcontrolere

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

6. Data înregistrării temei: 2017-12-05 11:41:53

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: **657acbc1a9**

Declaraţie de onestitate academică

Prin prezenta declar că lucrarea cu titlul "*Contributii la un system de casa inteligenta comandabila prin voce*", prezentată în cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei a Universităţii "Politehnica" din Bucureşti ca cerinţă parţială pentru obţinerea titlului de *Inginer* în domeniul *Electronică şi Telecomunicaţii*, programul de studii *Electronică Aplicată,* este scrisă de mine şi nu a mai fost prezentată niciodată la o facultate sau instituţie de învăţământ superior din ţară sau străinătate.

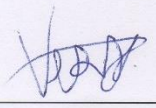
Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referinţe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi fac referinţă la sursă. Reformularea în cuvinte proprii a textelor scrise de către alţi autori face referinţă la sursă. Înţeleg că plagiatul constituie infracţiune şi se sancţionează conform legilor în vigoare.

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

Bucureşti, *20.06.2018*

Absolvent

Manolescu Vlad-Stefan



CUPRINS

LISTA ACRONIMELOR

ACC- Appliance Configuration and Control module-Modulul de control si de configurare al dispozitivului

API- Application Programming Interface- interfață destinată programării aplicațiilor

ASR-Automatic Speech Recognition- recunoaștere automată a vorbirii

BLE-Bluetooth Low Energy-Transmisie Bluetooth folosind energie minimală

CERN- Conseil Européen pour la Recherche Nucléaire -Organizația Europeană pentru Cercetare Nucleară

ETTI- Electronică, Telecomunicații și Tehnologia Informației

FIFO-First In First Out- Primul Venit Primul Ieșit

FM-Frequency Modulation- Modulatie in functie de frecvență

HDMI- High-Definition Multimedia Interface-Interfața multimedia de mare rezoluție

HMM-Hidden Markov Model- Modelul ascuns al lui Markov

HTTP- Hypertext Transfer Protocol- Protocol de transfer al Hypertextelor

IBM- International Business Machines Corporation-

ID-Identification-Identificare

IDE- Integrated development environment-Mediu de dezvoltare

IP-Internet Protocol

IT-Information Technology-Tehnologie si Informație

JSFG- Journal of Structural and Functional Genomics

JSON- JavaScript Object Notation- sintaxa pentru stocarea şi schimbul de informaţii text

LCD- Liquid Crystal Display-Afișaj cu cristale lichide

LED- Light-emitting diode- diodă emițătoare de lumină

MPC-Music Player Control- control al sistemului de redare a muzicii

MPD-Music Player Daemon

NASA- National Aeronautics and Space Administration

NLU-Natural Language Understanding- Modul de înțelegere al limbii vorbite

OS-Operating System-Sistem de operare

POO (OOP)-Programare Orientată pe Obiect

RAM-Random-access memory-Memorie cu acces aleator

REST-Representational State Transfer-Transfer de stare reprezentativ

ROM-Read-only Memory- Memorie strict pentru citire

SBC-Single-Board Computer-Calculator pe o singura placă

SpeeD- Speech and Dialogue Research Laboratory-Laborator de cercetare vorbirii si a dialogului

SSH-Secure Shell-Protocol de Securitate(“coajă securizată”)

SSL-Secure Sockets Layer-Protocol de de Securitate bazat pe Socket-uri

TCP/IP- Protocol de control al transmisiei/Protocol Internet

TLS- Transport Layer Security-Protocol de transfer de date securizat

TTS-Text-to-Speech- Modul de convertire a textului in vorbire.

URL- Uniform Resource Locator- Localizator uniform de resurse

USB- Universal Serial Bus- Magistrală Serială Universală

UTF8-Unicode Transformation Format-Format de transformare din Unicode

VPN- virtual private network-rețea private virtuală

WWW-World Wide Web

LISTA FIGURILOR

LISTA TABELELOR

CAPITOL 1

INTRODUCERE ŞI MOTIVAŢIA PROIECTULUI DE DIPLOMĂ

1.1 MOTIVAȚIE

In ultimii ani, inovațiile tehnologice au cunoscut o creștere exponențială, ceea ce a condus la impunerea unor noi strategii de abordare. În primul rând, s-a pus accent pe dezvoltarea de sisteme autonome pentru a răspunde cerințelor venite din partea utilizatorilor. Cel mai facil si la îndemână mod de interacțiune între om si mașină este reprezentat de vocea umana. Cu ajutorul acestui mod de comunicare se poate comanda un sistem sa execute anumite sarcini, venind in întâmpinarea nevoilor utilizatorului. In al doilea rând, creșterea puterii de calcul integrabile pe dispozitive permite îndeplinirea unor cerințe mult mai complexe într-un interval de timp redus.

Rutina si monotonia la care suntem supuși zi de zi pot afecta gradul de creativitate al unei persoane. Din acest motiv, după o zi stresantă oamenii au nevoie de condiții ce i-ar putea ajuta să se detașeze de mediul de lucru. S-a demonstrat științific faptul ca sistemul de iluminare al unei case si muzica ambientală sunt factori importanți ce influențează starea unei persoane. Un sistem de casă inteligentă presupune faptul că o persoana își poate crea acest mediu într-un mod facil, comandând prin voce [1].

1.2 IoT

Conceptul Internet of Things (IoT) reprezintă o multitudine de dispozitive înzestrate cu componente electronice ,soft-uri , senzori si conexiuni la internet, ce preiau date si le procesează. Acest concept a fost folosit in special in domeniul unei case inteligente capabil sa execute anumite comenzi, in momentul in care utilizatorul ii cere acest lucru. Aceste case inteligente sunt case ecologice ce ajuta la scăderea consumului si in același timp la satisfacerea nevoilor utilizatorului. Spre exemplu, o casa inteligenta va putea controla intensitatea luminii becurilor rezultând într-un consum minim de energie electrica.

Numărul de dispozitive IoT a crescut cu 31% de la an la an la 8.4 miliarde in 2017, și este estimat ca numărul sa crească pană la 30 miliarde pana in 2020. Valoarea pieței este preconizată să atingă o cifră de 7 trilioane de dolari americani pana in 2020 [2].

1.3 Soluție

Proiectul meu de diplomă se bazează pe realizarea un program software ce determină prin comandă vocală un răspuns al unor sisteme hardware. 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. In funcție de comanda utilizatorului, sistemul va executa una din următoarele comenzi: aprinderea/stingerea unor becuri din casă, modificarea intensității luminii becurilor ,modificare culorilor becurilor ,pornirea/stingerea unui radio si alegerea postului de muzica preferat.

Proiectul de față prezintă contribuțiile mele personale aduse proiectului ANVSIB, ce are la bază un concept de casă inteligentă dezvoltată in cadrul laboratorului de cercetare „Speech and Dialogue Research Laboratory” din cadrul Universității Politehnica București. Proiectul anterior nu beneficia de o flexibilitate in ceea ce privește setul de comenzi în sistemul de iluminare și timpul de execuție al sistemului la aceste comenzi era nesatisfăcător. Obiectivele lucrării au fost extinderea setului de comenzi in cadrul sistemului de iluminare obținând astfel particularități din punct de vedere al intensității și culorii luminii și de asemenea implementarea unei funcționalități noi capabile să redea posturi radio.

CAPITOL 2

DESCRIEREA COMPONENTELOR HARDWARE

2.1 DESCRIERE GENERALĂ

Componentele hardware ce au fost folosite pentru realizarea proiectului de licență sunt :

- Raspberry Pi
- Becuri inteligente Philips Hue
- Microfon Samson
- Boxa portabilă

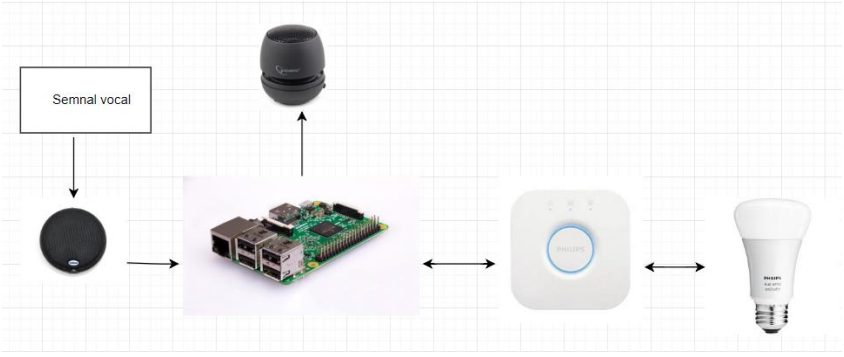


Figura 2-1

2.2 RASPBERRY PI

2.2.1 Informații generale

Plăcuța pe care se încarcă proiectul este de fapt un SBC(Single-Board Computer) și se numește Raspberry Pi 3 Model B și este prima plăcută din a treia generație de Raspberry PI, înlocuind modelul anterior Raspberry Pi 2 Model B în Februarie 2016. Acest model dispune de un procesor quad core pe 64 de biți cu frecvența de 1.2 GHz și are memorie RAM de 1GB.

Raspberry PI 3 dispune de posibilitatea conectării plăcii la o rețea de internet atât prin cablu Ethernet de mare viteză cât și prin rețea wireless. Conectarea se poate face și prin Bluetooth cu ajutorul tehnologiei de ultimă generație BLE(Bluetooth Low Energy). Pe lângă porturile USB discutate anterior placă are port de display DSI pentru conectarea cu un display cu touchscreen. Cardul de memorie MicroSD a fost cumpărat separat și are rolul de a stoca sistemul de operare (OS=operating system) Raspbian și datele pe care dorim să le stocăm în Raspberry. Alimentarea plăcii se face prin intermediul unui MicroUSB și este construit să poată fi folosit atât în SUA pentru tensiuni de 110V/60Hz cât și pentru Europa 220-230V/50Hz. Output-ul este o tensiune aproximativă de 5V la un curent de 2.5 A [3] [4].

2.2.2 Raspbian

Descărcarea sistemului de operare numit Raspbian se face de pe site-ul oficial Raspberry Pi și poate fi ales în varianta cu Desktop sau în varianta Lite în care toate comenzile se fac din consolă.

În cadrul proiectului se va utiliza varianta Lite cu scopul de a învăţa și în special a folosi comenzi Linux care ajută în cele mai multe cazuri în dezvoltarea abilităților unui programator. Se downloadează arhiva ce conține software-ul cu ajutorul unui adaptor MicroSD și se dezarhivează sistemul de operare direct în cardul de memorie.

Se introduce cardul de memorie in slot-ul specific și se conectează alimentarea, nu înainte de a verifica că toate componentele de I/O precum tastatura și ecranul sunt conectate. Procesul de instalare a sistemului de operare este automat. Raspberry Pi nu deține un buton fizic de pornire a plăcii , aceasta pornește când este alimentat.

Odată ce instalarea sistemului de operare Raspbian este finalizată se cere utilizatorului să se logheze prin intermediul unui username și al unei parole. Valorile lor default sunt :

`Username:pi`

`Password:raspberry`

Ele pot fi ulterior modificate dacă utilizatorul dorește acest lucru [5].

2.2.3 Comenzi folositoare în terminal

O provocare frecvent întâlnită în realizarea unui proiect într-un sistem ce operează pe Linux este modul de lucru în linia de comandă. Un inginer software trebuie să fie capabil să efectueze anumite operații în linia de comandă rapid și fără redundantă.

Sudo este o comandă ce oferă libertatea utilizatorului să aibă acces total. Din aceste motive, parola nu mai trebuie introdusă pentru anumite task-uri administrative [6].

Această comandă este folosită foarte des. Un exemplu în care este folosit sudo este oprirea/restartarea corectă a Raspberry-ului prin :

sudo shutdown now/sudo reboot

Un beneficiu major întâlnit în majoritatea sistemelor Linux este salvarea comenzilor pe care utilizatorul le face într-o listă de tipul FIFO (first în first out). Astfel, pentru a vedea ultimele comenzi efectuate proiectantul trebuie să apese butonul de pe tastatură ce este marcat cu o săgeată în sus. Cu toate acestea, dacă proiectantul a executat o comandă foarte lungă și nu își mai amintește sintaxa exactă o metodă eficientă este tastarea Ctrl+R care are ca scop deschiderea unui text box cu rol de căutare în istoricul comenzilor.

Linux este un sistem de operare sigur, cu toate acestea există posibile atacuri și bug-uri. Securitatea este un factor foarte important şi el se poate asigura prin actualizarea constantă a sistemului.

sudo apt-get upgrade- se ocupă cu reîmprospătarea depozitelor

sudo apt-get update- se ocupă cu actualizarea propriu-zisă

Sistemul de operare Raspbian conține numeroase pachete ce sunt disponibile în depozitele oficiale și oferă numeroase beneficii proiectantului. De exemplu, pentru compilarea unui fișier python pe Raspberry PI este necesară instalarea pachetului pip3.

sudo apt-get install pip3, instalarea pachetului

sudo apt-get remove pip3, dezinstalarea pachetului

Alte comenzi de bază [7]:

cp calea_fisierului calea_directorului_unde_se_copiază, copierea unui fişier şi mutarea copiei în alt fișier

mv cale_fişier cale_director_unde_se_mută, mutarea fişierului în altă locaţie

mv nume_fişier_vechi nume_fişier_nou, redenumirea unui fişier

rm nume_fişier, ştergerea unui fişier

cd /home/ANVSIB, se ocupă cu schimbarea directorului în care se lucrează

mkdir new_folder, aceasta creează un director nou

2.2.4 Editoare de text

Deschiderea şi scrierea într-un fişier text se face cu ajutorul unui editor de text ce permite proiectantului să scrie cod sau se modifice diferite fişiere de configurare ale plăcii. Primul editor de text într-un sistem de operare Linux este **vi** şi a apărut la mijlocul anilor 1970.

De atunci au apărut numeroase editoare de text gratis sau cu licenţă printre care se enumeră Joe, Pico şi vim (v improved).

În acest proiect am editat fişierele cu **nano**. Nano este o copie şi în același timp o îmbunătățire a editorului de text Pico GPL, diferența fiind că nano suportă şi limba germană, franceză şi portugheză pe lângă engleză şi italiană. Din punct de vedere al impactului vizual nano a adăugat colorarea textului ce îl percepe ca sintaxa clasică dintr-un limbaj de cod.

De exemplu cuvântul cheie import este scris cu roșu în cod. Nano prezintă îmbunătăţiri majore şi în departamentul de căutare şi de deplasare la o anumită linie într-un fişier de text cu ajutorul combinațiilor de butoane de la tastatură [8] [9] [10].

Combinaţii de la tastatură pentru nano:

Ctrl+O , salvarea fişierului

Ctrl+X , ieşirea din fişier

Ctrl+K , tăierea segmentului selectat din fişier

Ctrl+U , lipirea segmentului de text (paşte)

Ctrl+_ , cursorul se duce la linia dorită

2.2.5 Conectarea Raspberry Pi la internet

Pentru a instala diverse pachete de care proiectantul vrea să beneficieze (de exemplu pip sau mpc) este necesară conectarea plăcuţei la internet în cazul de faţă prin conectarea la o reţea wireless. O metodă de conectare este comanda:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Aceasta deschide un fişier de configurare unde se scrie numele reţelei wi-fi la care se dorește conectarea. In ceea ce privește parola, aceasta este introdusă in câmpul „psk”. Închiderea fişierului se face prin comanda Ctrl-X urmată de Y pentru a salva fişierul de configurare.

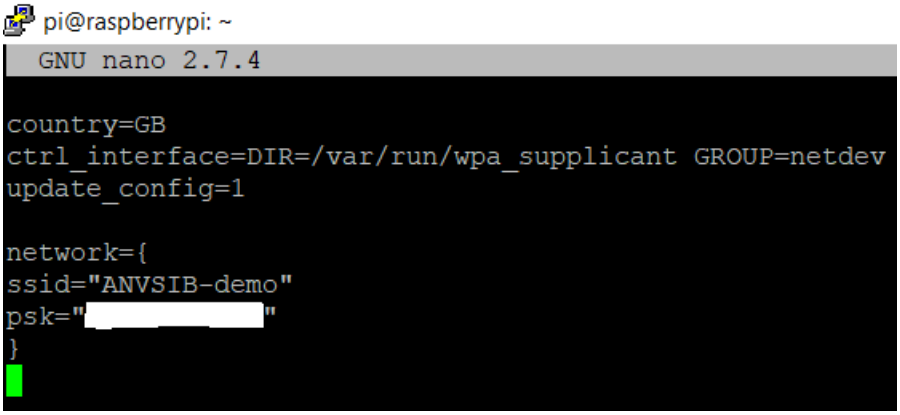


Figura 2-2

Pentru a verifica dacă Raspberry ul este conectat la internet este necesară restartarea Raspberry Pi-ului folosind comanda de reboot iar utilizatorul este nevoit să se logheze din nou. Pentru a verifica că plăcuta poate primi şi trimite pachete pe internet se folosește comanda ping urmată de numele unui site (de exemplu ping google.ro) [11].

2.3 MICROFON SAMSON

În urma succesului remarcabil al microfoanelor USB, Samson introduce microfonul UB1 cu o suprafață plană perfectă pentru recepția undelor sonore din toate directile. Acest microfon omnidirecțional se instalează automat la orice calculator sau în acest caz la Raspberry Pi şi transmite datele direct. 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 slim [12].

Figura 2-3

2.4 BOXA 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 elegant şi funcționează la orice aparat ce are mufă pentru jack de 3.5mm. Bateria are autonomie de până la 6 ore şi o greutate de sub 200 de grame. Puterea de emitere este de 2W şi banda de frecvenţe pe care poate să la emită este 100Hz-20kHz [13].

Figura 2-4

2.5 PHILIPS HUE

2.5.1 Informații generale

Philips Hue este un proiect realizat de cei de la Philips în 2012 şi mai târziu actualizat în 2016 ce se bazează pe comanda unor becuri inteligente LED prin intermediul unei reţele Wi-Fi şi printr-o conexiune de tip Zigbee.

Zigbee este un protocol de comunicaţii conceput în anul 1998 şi revizuit în 2006, de nivel înalt folosit pentru reţele network personale ce nu necesită un volum mare de energie pentru transmiterea datelor şi este funcţional pe distanţe mici de până la 100 m. În general Zigbee este folosit pentru aplicaţii ce nu necesită o rată foarte mare de transfer de date(volum mic de date în general).Rata de biţi cu care sunt transferate datele este de 250kbit/s iar reţelele Zigbee sunt securizate cu chei simetrice de criptare pe 128 biți.

2.5.2 Sistemul Hue

Sistemul de lumini Hue este conceput în general din 3 componente:

1.Aplicaţia (în acest caz comandă vocală)

2.Bridge (podul de conexiune)

3.Becurile Philips Hue

1. Aplicaţia este modul în care utilizatorul va putea interacţiona cu becurile inteligente. O metodă de a te conecta la becurile inteligente este printr-o aplicaţie pe telefon pusă la dispoziție Philips Hue Lights. Aceasta permite utilizatorului să schimbe intensitatea luminii unui bec, saturaţia, nuanţă pe care o doreşte, poate seta o alarmă în care becurile să se aprindă la o anumită oră sau poate crea diferite efecte speciale pentru a crea atmosfera dorită de client. Aplicaţia desigur nu este singurul mod de a te conecta la sistem. Un alt mod este prin intermediul unui site web sau cu ajutorul unui Raspberry Pi .

2. Bridge ul este folosit pentru a conecta becurile Phillips la aplicaţie. Setul principal de API uri este este oferit de bridge şi acesta permite utilizatorului să folosească toate beneficiile becurilor. Pentru a se stabili conexiunea primul pas este conectarea Bridge-ului şi al aplicaţiei utilizatorului să fie conectate la aceeași rețea locală.

Figura 2-5

3. Becurile Philips Hue sunt ieşirea sistemului şi conţin 3 tipuri de LED alese special pentru a oferi o varietate de intensităţi şi nuanţe pentru utilizator. Pe lângă iluminare fiecare bec transmite un mesaj de tip JSON la fiecare schimbare pe care utilizatorul o face .

Figura 2-6

Execuţia propriu-zisă a comenzilor privind iluminarea va presupune transmiterea unor cereri de tip HTTP (POST sau PUT) către serviciul REST de comandă al luminii, pus la dispoziţie Philips Hue. Prin urmare fiecare parametru al fiecărui bec poate fi controlat printr-un URL.

2.5.3 JSON şi variabile de bază în sistemul Hue

Toate răspunsurile şi toate valorile noi pe care le iau becurile sunt trimise şi returnate prin JSON(JavaScript Object Notation) şi codate cu UTF8.

JSON este un format uşor de folosit pentru schimbări de date şi provine din Javascript deşi nu sunt unul şi acelaşi lucru. Acest format este construit pe 2 structuri:

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

Proiectul de licenţă se bazează pe prima structură [14].

Exemplu de JSON folosit în proiectul de diplomă :

```
{"on":true,"bri":255,"sat":255,"hue":0}
```

Variabilă on este de tipul Boolean (acceptă numai valori true sau false) şi cu ajutorul ei se obţine aprinderea sau stingerea becului.

Variabilă bri determina intensitatea luminii becului şi poate avea valori între 1 şi 254 .

Variabilă sat are rolul de a schimba intensitatea nuanţei culorii pe care utilizatorul o doreşte. De exemplu, saturaţia mare accentuează nuanţa dorită în timp ce o saturaţie mică va oferi o culoare palidă si mai apropiată de alb. Valorile saturaţiei sunt cuprinse ca şi la intensitate între 1 şi 254.

Variabilă hue este responsabilă cu nuanţa culorii pe care becul o are şi poate avea valori între 0 şi 66000.

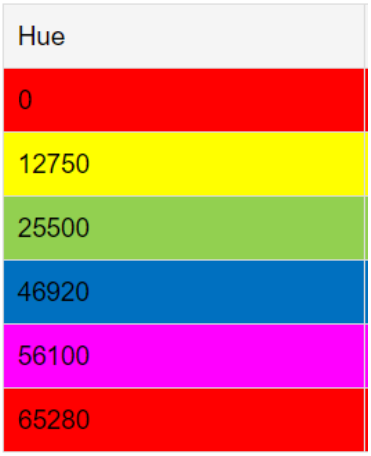


Figura 2-7

Tabelul de mai sus ilustrează culorile cu codul lor pentru o saturaţie maximă [15] [16].

CAPITOL 3

TEHNOLOGII SOFTWARE

3.1 LIMBAJE DE PROGRAMARE

3.1.1 Informaţii generale

Limbajul de programare este caracterizat de un ansamblu alcătuit de un vocabular şi un set de reguli gramaticale utile pentru un computer în vederea realizării anumitor cerinţe.

Scopul acestuia este de a oferi posibilitatea programatorului să poată transmite, prin intermediul programelor, către calculator într-un mod precis şi amănunţit acţiunile pe care acesta trebuie să le realizeze, mai mult decât atât, datele cu care să facă acest lucru cât şi ordinea.

Putem clasifica limbajele de programare în două mari categorii:

1. Limbaje de nivel coborât, dependent de calculator:

limbajul maşină: un set de instrucţiuni diferite de le un computer la altul, fiecare având o combinaţie de 4 biţi realizaţi într-un cod binar cu o succesiune de 0 şi 1 .Pentru a evita erorile ce apar din cauza suprapunerii adreselor de memorie , programatorul trebuie să gestioneze fără greșeală alocarea adreselor pentru un program.

limbajul de asamblare: se realizează prin intermediul unui program ce traduce programele în limbaj maşină şi poartă numele de asamblor(assembler).Acest limbaj conferă programatorului un foarte mare control în ce se întâmplă în computer.

2. Limbaje de nivel înalt, independent de structura calculatorului.

Câteva exemple: - Fortran-1995, IBM - pentru probleme tehnico-stiintifice

- Cobol-1959 - pentru probleme economice

Programarea procedurală din anii 1970 (Pascal,C,etc): a apărut cu scopul de a crea programe capabile să fie sigure în funcţionare cât mai mult timp. Este o modalitate de programare care descompune problemele complexe în subprobleme mai simple numite module. Această abordare poartă numele de **top-down**. Orice algoritm poate fi compus din numai trei structuri de calcul, conform teoremei lui Bohm şi Jacopini şi anume:

- structura secventiala-secventa
- structura alternative-decizia
- structura repetitive-ciclul (buclă)

Programarea orientată pe obiecte din anii 1980 (C++, Java, etc.) îşi propune gruparea datelor şi codurilor într-o singură structură, în unităţi individuale de cod care interacţionează cu altele. Motivul pentru care obiectele POO de cele mai multe ori sunt reprezentări din viaţa reală a condus la o mai bună înţelegere a programelor cât şi către un mod mai accesibil de depanare.

O altă caracteristică prin care se clasifică limbajele de programare este după modul de "traducere":

- Limbaje compilate:

C, C++, Pascal, Java. Prin intermediul unui compilator programul sursa este tradus în limbaj maşină

- Limbaje interpretate:

PHP, Javascript, Python, Matlab. Aceste limbaje de programare au un interpretor prin intermediul căruia codul sursă se transformă în cod maşină şi se execută linie cu linie .

3.1.2 Limbajul C

C este unul dintre cele mai utilizate limbaje de programare pe scară largă din lume pentru scrierea de software de sistem, implementat pe majoritatea platformelor de calcul a fost creat între anii 1969 şi 1973 de către Dennis Ritchie şi Brian Kernigham.

Scopul iniţial acestui limbaj de programare a fost destinat scrierii unei părţi din sistemul de operare Unix. Acest limbaj a fost denumit "C" deoarece este un succesor al limbajului de programare B.The „C Programming Language" reprezintă cartea de referinţă în domeniu, structurată pe 12 capitole ce prezintă într-o manieră unitară limbajul C, a celor doi fondatori , care

impune un standard minimal pentru orice implementare.Cunoscută în rândul programatorilor sub numele de K&R este adesea considerat limbajul de bază pe care orice compilator C trebuie să-l suporte.

Limbajul de programare C este unul structurat și din acest motiv conduce la o bună organizare a programelor. Acest limbaj este potrivit pentru programarea de sistem pentru că are un cod redus ca dimensiune şi rapid în execuție ceea ce face ca acesta să deţină un cod obiect eficient. De asemenea, deţine o complexitate a limbajului de nivel mediu, dă voie la o programare de nivel scăzut faţă de alte limbaje deoarece operaţiile se realizează pe biţi, accesul la memorie fiind direct. Un dezavantaj pentru C îl reprezintă stricteţea în ceea ce priveşte tipurile de date conversiile implicite între tipuri.

Biblioteca string din C conține funcții clasice de modificare a variabilelor ce memorează șiruri de caractere. Aceasta bibliotecă este folosită intens pentru optimizarea structurilor de decizie ce analizează comanda text a utilizatorului.

`strcpy(s1, s2);` –copiază stringul oferit ca al doilea parametru în stringul folosit ca primul parametru

`strcat(s1, s2);`–concatenează cele două string-uri

`strcmp(s1, s2);`–compară cele două string-uri dacă sunt identice

`strstr(s1, s2);`–verifică daca al doilea string se regăsește in primul (incluziune)

Inițial limbajul de programare ales a fost C deoarece rolul temei mele a fost de a contribui la un sistem deja existent ce se ocupa cu recunoaşterea vocală a unor comenzi relativ simple. Cu alte cuvinte, proiectul iniţial numit ANVSIB putea comanda numai aprinderea şi stingerea tuturor luminilor [17].

3.2 PYTHON

3.2.1 Informații generale

Python este un limbaj de programare ce este foarte popular în domeniul IT conform ultimelor statistici oferite de Github. Acest limbaj de programare a apărut în anul 1991 şi a fost creat de Guido van Rossum. În ziua de astăzi Python este foarte folosit de companii precum Google şi Yahoo pentru aplicaţii web dar şi pentru aplicaţii embedded cum este acest proiect. Youtube şi Amazon sunt site-uri ce au fost dezvoltate prin Python. Un mare avantaj care a condus la popularitatea mărită a limbajului de programare este implementarea sa cu ajutorul CPython. Interpretatorul CPython este inclus în sistemele de operare precum Linux şi Mac OS X.

Astfel programatorul poate coda direct din maşină virtuală. Procesul este posibil deoarece, spre deosebire de multe alte limbaje de programare printre care şi C, Python este un limbaj de programare interpretat, etapă de compilare ne mai fiind necesară. Din punct de vedere al timpului, un programator preferă să aibă fişierul deja compilat pentru a îl putea rula în timp ce la Python ieşirea din program determină din nou transformarea codului în biti ce pot fi interpretaţi de calculator. Avantajul clar din această situaţie reiese din posibilitatea de a rula un cod Python pe orice sistem de operare, fără ajustări în plus.

Un concept fundamental ce stă la baza limbajului de programare Python este indentarea. Indentarea este plasarea codului pe linii sub un anumit set de reguli cu ajutorul cărora IDE-ul interpretează codul scris de programator. Acest mod de a scrie cod a ajutat la reducerea volumului de linii pe care un programator trebuie să le scrie pentru acelaşi algoritm(de exemplu buclele nu mai necesită { } ca în C) , dar în acelaşi timp îl obligă să menţină un cod curat şi foarte uşor de înţeles. Un cod ce este lizibil este un argument serios pentru o firmă de a prelua şi a coda în acel limbaj de programare deoarece face muncă în echipă mult mai eficientă.

Din punct de vedere al tipului de programare, Python poate fi folosit atât în programarea orientate pe obiect şi funcţională cât şi în cea procedurală ce este folosită în specială pentru sistemele embbeded. Administrarea memoriei se face că şi la Java prin intermediul unui serviciu de colectare numit "garbage collector" ce se ocupă automat de ştergerea din memorie a variabilelor ce nu mai sunt folosite spre deosebire de C.

Conceptul "Batteries Included" sugerează că orice limbaj de programare, indiferent de modul în care a fost gândit şi proiectat, necesită un set de biblioteci pentru a avea utilitate practică. Popularitatea crescută a Pythonului a determinat un avantaj important al acestui limbaj open-source deoarece a condus la o cantitate foarte mare de biblioteci ce pot fi accesate de orice programator. API urile Python oferă o gamă largă de funcţionalităţi precum cele de bază (modificarea unui string) până la lucrul cu procese şi thread-uri. Un exemplu poate fi implemantarea unui cod Python cu rolul de a executa comenzii ce se fac de obicei în Matlab (software cu licenţă) precum afişarea unor grafice şi lucru cu matrici. Acest lucru este posibil prin API ul Matplotlib. Un alt exemplu este pachetul wxPython ce oferă metode şi structuri de date specifice creării unei intergete grafice.

O altă caracteristică ce are rolul de a face codul scris în Python mai compact este permisivitatea limbajului de a nu specifică tipul unei variabile la declararea acesteia. Cu ajutorul interpretorului, tipul variabilei este dat de conţinutul pe care programatorul îl dă.

```
variabilaX=10
 # interpretorul va considera această variabilă de tip întreg
variabilaY="sunt un string"
 # interpretorul va considera această variabilă de tip string
```

Python nu permite operaţii cu obiecte diferite şi are conceptul de variabile mutabile şi nemutabile. O variabilă nemutabilă este o variabilă ce nu i se poate altera conţinutul după ce au fost iniţializate.

De exemplu :

```
stringul_meu="alex"
stringul_meu[3]="y"
#va rezulta într-o eroare
```

Termenul de "sugar coding" a apărut recent în programare înlocuind funcţiile din C precum strcmp ce a avea rolul de a verifica dacă două string-uri sunt egale cu termenul în folosit astfel în Python

```
      if (string_mic in string_mare==True):
            #verificarea apartenenței primului șir de caractere la cel de al
doilea
```

Funcţiile în Python sunt diferite de cele clasice ce sunt întâlnite în C şi în Java. În C funcţia este fie de tip void ,caz în care nu returnează nimic , sau de un anumit tip caz în care funcţia trebuie să returneze acel tip în programul principal sau în altă funcţie. În Python funcţiile se declară cu **def** urmat de numele funcţiei şi parametrii pe care aceasta trebuie să îi primească tot în interiorul parantezelor că şi la celelalte limbaje de programare. Programatorul nu mai trebuie să se gândească înainte de scrierea codului dacă funcţia lui trebuie să returneze ceva, poate să returneze în unele cazuri şi pe altă structură de decizie (if -else) să fie o funcţie de tip void [18] [19].

3.2.2 Comparație între versiunea Python 2 și Python 3

Versiunea a treia a fost creată în anul 2008 la 8 ani după apariţia celei de a doua versiuni. Principiul de bază al versiunii noi era eliminarea redundanţei ce apăruse la versiunea anterioară.

Această filozofie determina ştergerea structurilor ce erau duplicat.

"There should be one- and preferably only one obvious way to do it"

Cu toate acestea, marele dezavantaj pentru Python 3 ce a creat mari probleme a fost incompatibilitate cu proiectele scrise anterior în Python 2.Comunitatea developerilor de programe software a fost sceptică la agregarea noii versiuni din acest motiv. Trecerea de la Python 2 la Python 3 a durat ani pentru aplicaţii mari precum CherryPy şi Django. Un patch numit 2to3 a fost dezvoltat pentru programele vechi scrise în Python 2 pentru a fi "traduse" în versiunea 3.

În continuare vor fi prezentate diferențele de bază dintre cele 2 versiuni.

O schimbare din punct de vedere al structurării datelor este eliminarea tipului de date întâlnit şi în C numit long. În C şi în Pyhton 2, tipul de variabilă long avea rolul de a memora un număr întreg între [-2000000 ; 2000000] în timp ce int putea memora un număr întreg cu valori între [-32000 ; 32000]. Cu apariţia Python 3 int îşi măreşte domeniul şi înlocuieşte necesitatea long-ului.

O altă deosebire între cele două versiuni stă în modul de afişare al unui text în consola. În versiunea mai veche prinţ nu era considerate o funcţie, ci mai degrabă o comandă. Astfel, print "Hola" ar fi funcţionat corect. În versiunea nouă singurul mod de afişare în consola este cel cu care suntem obişnuiţi şi din alte limbaje de programare în care print este folosit şi privit ca o funcţie, iar parametrul pe care îl primeşte funcţia (print("Hola")) este obiectul ce trebuie afişat.

O diferenţă ce a cauzat multe probleme pentru portare programelor de la Python 2 la Python 3 a fost în comparare. În versiunea 2.0 compararea se făcea fără să se ţină cont de logică că obiectele nu pot fi comparate din simplul motiv că nu aparţin aceleiaşi clase. De exemplu:

```
1<" "  #ar returna valoarea de tip Boolean True în 2.0
1<" "  # ar produce o eroare în 3.0
```

3.2.3 Instalare pachete Python pe Raspberry Pi

Instalarea bibliotecilor pe Raspberry PI este esenţială pentru funcţionarea corectă a programului. De exemplu, pentru a transmite cereri de tip HTTP am folosit biblioteca "requests" introdusă în program cu ajutorul cuvântului cheie "import". Dar, folosirea cuvântului cheie nu este suficientă dacă biblioteca nu se află în sistemul de bază ce vine cu instalare Python pe Raspberry. Raspberry Pi introduce pentru a simplificare pip. Pip este un installer specific pentru Python şi în general se află în sistemul de operare Rasbian. În cazul în care nu se află, sau se află varianta pentru Python 2, se foloseşte comanda obişnuită de instalare a unui nou pachet:

- `sudo apt-get install python3 pip`
- `pip3 install requests`

Fişierul în care se scrie codul este de tip text şi este deschis cu comanda

```
sudo nano  nume_fişier.py
```

Cu toate acestea nano nu are reguli de indentare ce sunt absolut necesare pentru rularea unui program în Python.

Odată instalate toate bibliotecile în terminal şi incluse în cod prin "import", programul este rulat în terminal cu cuvântul cheie python3 urmat de numele fişierului salvat cu extensia .py

python3 Program.py

3.3 PUTTY

Conectarea la internet a Raspberry-ului permite inginerului să se folosească de toate librăriile de care are nevoie, însă codarea se face relativ dificil prin intermediul editorului de text **nano**. De aceea am folosit programul software Putty ce permite conectarea plăcii la laptop (conectarea directă prin cablul HDMI nu este posibilă) prin emularea unui terminal ce se conectează prin SSH.

Conexiunea se face prin conectarea atât laptopului cât şi a plăcii la aceeaşi reţea wi-fi şi prin cunoaşterea IP ului plăcii. Pentru a a afla IP ul se foloseşte comanda **ifconfig**.

Motivul principal pentru care am avut nevoie de acest software, ce face legătura între Raspberry PI şi laptop, este scrierea codului Python într un IDE. Există diferite programe ce sunt specifice pentru scrierea în Python, dar am ales să merg cu PyCharm pentru beneficiile pe care le oferă. Pycharm oferă programatorului posibilitatea de a scrie cod cu indentare automată specifică pentru

limbajul Python. De asemenea programul m-a ajutat să învăţ mult mai repede regulile de bază al sintaxei datorită compilării continue a programului. (verificare linie cu linie pentru posibile erori). Odată ce scriam codul şi verficăm ca acesta să nu aibă erori, îl translatăm cu Copy&Paste într-un fişier text salvat cu extensia specifică .py [20]

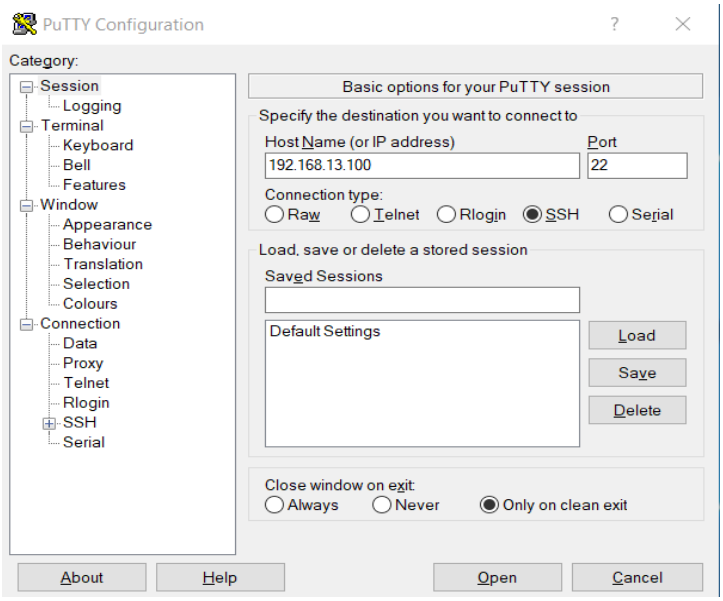


Figura 3-1

3.4 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.

Un wiki este un website şi/sau o bază de date construită de o comunitate de programatori, fiecare user având acces să adauge sau să modifice conţinutul bazei de date. Gitlab oferă două variante, Gitlab Community Edition, Enterprise Edition şi Gitlab-hosted version. Gitlab este o platformă ce este folosită de firme mari precum NASA, CERN şi Alibaba.

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 [21].

Membrul echipei "clonează" proiectul pe calculatorul său şi poate lucra la proiect fără să îşi deranjeze colegii. Odată ce şi-a îndeplinit taskul, 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.

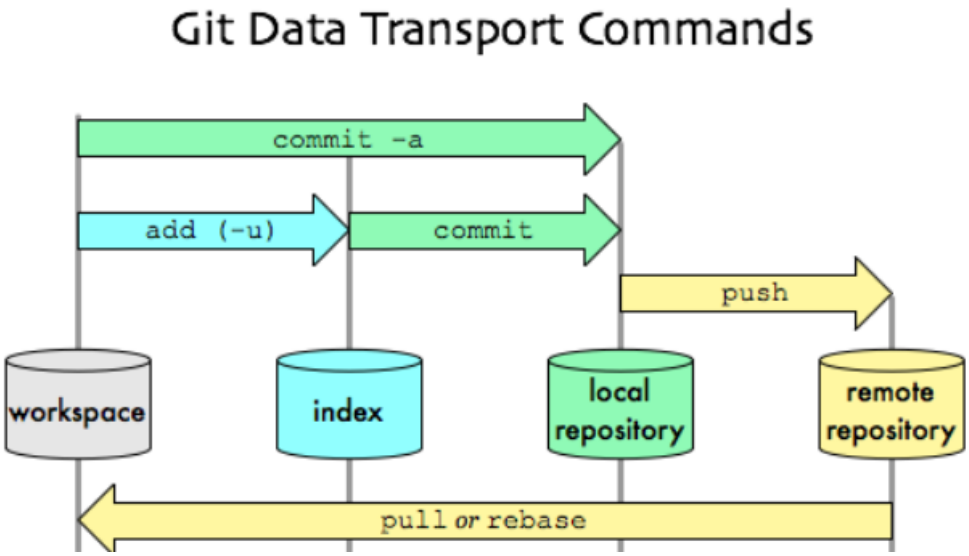


Figura 3-2

Comanda git commit încarcă modificările pe o zonă de depozitare locală. Modificările se observă şi pe Gitlab.com după ce s-a dat comanda push. Din motive de Securitate, comanda push cere autentificarea utilizatorului pe GitLab şi în acelaşi timp verifica dacă utilizatorul are acces la proiectul la care vrea să încarce schimbările efectuate [22] [23].

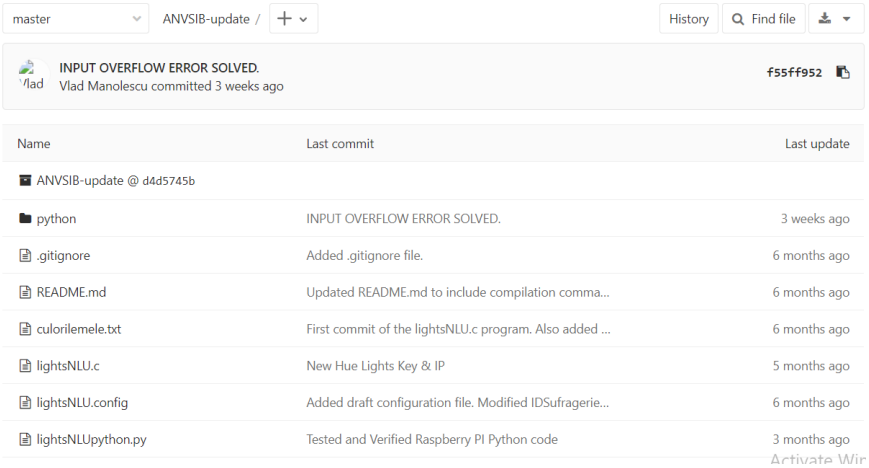


Figura 3-3

3.5 MODUL DE REDARE RADIO

3.5.1 Informații generale

Radioul a fost inventat acum mai bine de un secol de către italianul Guglielmo Marconi la Londra, unde a făcut publică o demonstraţie în care transmitea semnale la distanţă fără fir. Modelul teroretic există deja de 4 ani şi fusese proiectat de faimosul Nikola Tesla.

În România primele comunicaţii radio au avut scop militar în anul 1912 în parcul Carol din Bucureşti. Sistemul avea o putere de 12kW şi era considerat foarte puternic la aceea vreme.

La data de 19 noiembrie 1925 se realizează prima emisiune radio strict pentru divertisment, iar 3 ani mai tarziu s-a creat primul post de radio, Radio România, ce se emitea pe lungimea de undă de 401,5 m. Cu timpul radioul a avut numeroase transformări de la radioul fix la radioul portabil ce beneficia de invenţia tranzistoarelor(ex: Sony 1954) [24].

3.5.2 Radioul digital si transmisia prin internet

Radioul digital a apărut mult mai târziu şi se bazează pe scanarea sunetului, comprimarea şi eşantionarea acestuia prin diferite tehnologii şi apoi transmis. Rată de transfer este mult mai mare ca transmisia analogică a undelor radio (poate ajunge la 1,7 Mbits/s). Radioul transmis pe internet a apărut la sfârşitul anilor 90 având succes în special în America. Astăzi, majoritatea radiourilor din România transmit atât analog (FM) cât şi prin net streaming [25].

Transmiterea radioului prin intermediul internetului necesită 3 componente de bază:

- Sursa de emisie
- Serverul de transmisie
- Un program software de tip mp3 player

Programul software are rolul de a transforma pachetele de date sub formă de biţi în unde sonore redate prin boxe. Conexiunea este de tip TCP/IP şi redă undele sonore cu o întârziere de aproximativ 2 secunde. Pentru redarea radiolui am folosit şi instalat Music Player Daemon pe Raspberry PI.

Music Player Daemon este un program software ce se ocupă redarea fişierelor audio şi cu atât organizarea cât şi reţinerea unor playlisturi. Comanda care se foloseşte în terminal pentru a controla aplicaţia este mpc. Pentru a reda muzica, fişierele audio se introduc în baza de date a programului şi fiecare melodie/post de radio este salvat şi poate fi găsit printr-un indentificator numeric reprezentând ordinea în care melodiile au fost adăugate în playlist [26].

```
sudo apt-get install mpd mpc
```

Pentru a funcţiona radioul linkul adăugat la playlist trebuie să fie strict pentru netstreaming şi în general aceste linkuri se găsesc greu. Pentru a găsi aceste linkuri recomand căutarea playlisturilor pentru Winamp [27].

Odată ce MPD poate decoda muzică din linkurile adăugate utilizatorul poate folosi următoarele comenzi:

`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

```
pi@raspberrypi:~ $ sudo mpc play
Radio ZU Live: Let's do the ZU
[playing] #3/11   0:00/0:00 (0%)
volume: 97%   repeat: off   random: off   single: off   consume: off
pi@raspberrypi:~ $ sudo mpc next
http://stream.dancefm.ro:8032/dancefm.mp3
[playing] #4/11   0:00/0:00 (0%)
volume: 97%   repeat: off   random: off   single: off   consume: off
pi@raspberrypi:~ $ sudo mpc play 11
KissFM Romania: Matthew Koma - Kisses Back ~
[playing] #11/11   0:00/0:00 (0%)
volume: 97%   repeat: off   random: off   single: off   consume: off
pi@raspberrypi:~ $
```

Figura 3-4

3.6 HTTP

3.6.1 Informații generale

HTTP este o aplicaţie protocol ce funcţionează pe lângă TCP/IP şi este probabil şi cea mai folosită. Acest protocol este un set de reguli pentru transferul unor fişiere de tip text, imagini, sunet şi video prin intermediul internetului (World Wide Web). Puţină lume cunoaşte faptul că un Web Browser, precum Google Chrome şi Mozilla Firefox, este de fapt o aplicaţie ce foloseşte indirect protocolul HTTP pentru a executa diferite cereri pe care utilizatorul le comandă. Aceste cereri sunt fundaţia pentru modul de comunicare dintre date pentru internet.

Ideea de bază a acestui protocol de tip cerere-raspuns poate fi exemplificată cu ajutorul unui Web Browser. În cazul de faţă considerăm că utilizatorul foloseşte acest browser pentru a obţine anumite informaţii ce îi sunt necesare de pe internet. Astfel, persoană care doreşte să acceseze

informaţiile este un client în timp ce web-site ul unde se află informaţiile este serverul. În general serverul oferă ca răspuns statusul (dacă cererea a fost acceptată sau nu) şi poate continue şi răspunsul dorit de către utilizator.

O primă problemă ce a apărut în special la început secolului XXI a fost traficul de date deoarece mai mulţi utilizatori făceau cereri HTTP pe un anumit server în acelaşi timp şi acest lucru conducea la blocarea serverului. O metodă de rezolvare a acestei probleme a fost memoria web cache ce avea rolul de a memora temporar informaţii ce sunt des accesate pentru a le putea reutiliza şi a reduce traficul. Acest lucru a fost posibil deoarece HTTP a fost gândit să permită reţele intermediare pentru a îmbunătăţii comunicarea dintre client şi server [28]. O astfel de reţea intermediară se numeşte un server de tip proxy ce facilitează comunicarea cu clienţii. De asemenea, aproape orice server are un HTTP daemon ce are rolul de a aştepta şi a asigura traficul de cereri făcute la serverul respective.

Abrevierea HTTP provine de la HyperText Transfer Protocol. Hypertext este un text ce poate fi accesat de către utilizator pentru a fi condus către un alt website prin intermediul unei cereri ce este transmisă la adresa de protocol indicată de URL ce urmează să fie accesat. Servere web primesc şi transmit informaţii prin porturi . În cazul de faţă HTTP foloseşte portul 80 pentru a transfera resurse [29].

3.6.2 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. De exemplu o eroare pe care orice programator a întâmpinat-o este "Error 404 - File Not Found". Acesta este un tip de răspuns în cazul în care pagina pe care utilizatorul a accesat-o nu există sau URL-ul a fost scris greşit. Pentru a înţelege mai bine răspunsurile HTTP voi oferi un exemplu pentru fiecare cod:

- 1xx această clasă transmite numai un mesaj informativ

  Exemplu : 101 Anunţă schimbarea protocoalelor

- 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

- 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

- 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)

- 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.

Securitatea reprezintă o problemă pentru protocolul HTTP deoarece acesta nu este sigur. O metodă de atac ce este folosită împotriva utilizatorului se foloseşte de cererea TRACE şi se mai numeşte cross-site tracing. Acest tip de cerere verifica dacă răspunsul dorit de la server a fost sau nu modificat pe parcursul reţelelor intermediare. De regulă această cerere este folosită numai dacă sistemul este bine protejat şi nu poate fi atacat [30].

HTTPS este protocolul de comunicare securizat(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 spre deosebire de 80 pentru HTTP [31].

3.6.3 Cereri HTTP specifice pentru transferul de date către sistemul Philips Hue

Conectarea la becurile inteligente la Philips Hue se face în 3 paşi. Primul pas a fost menţionat in capitolul dedicate Philips Hue şi constă în conectarea bridge-ului şi al aplicaţiei cu care utilizatorul va interacţiona la aceeaşi reţea wi-fi. Al doilea pas constă în identificarea IP-ului bridge-ului la reţeaua de internet folosită. Există diverse metode pentru identificarea IP-ului unui sistem conectat la wireless. Al treilea pas este obţinerea unei chei ce oferă accesul utilizatorului la becuri. Din motive de securitate această cheie se obţine numai dacă utilizatorul are acces la singurul buton al bridge ului.

Toate cererile de tip HTTP pentru acest sistem conţin 4 identificatori fiecare cu un rol special:

1.URL-ul. Aceasta este adresa locală a unui obiect specific ce face parte din sistemul Hue.(de exemplu un bec sau un grup de becuri). De asemenea acest URL va conţine cheia ce permite accesul utilizatorului.

2.Un mesaj JSON ce oferă informaţiile necesare sistemului despre ce anume doreşte utilizatorul.

3.Un tip de cerere HTTP. Simplul mesaj JSON nu este suficient pentru a-i spune sistemului dacă utilizatorul doreşte informaţii despre sistem sau vrea să interacţioneze cu el. Sistemul Hue foloseşte 4 astfel de cereri (HTTP Requests):

- GET : este folosit pentru a obţine informaţii despre sistem
- PUT : este o comandă pentru modificarea variabilelor unui bec sau mai multor becuri
- POST: este o comandă ce creează o noua resursă în altă componenţă
- DELETE: comandă care se ocupă evident cu ştergerea unei componente din sistem.

4.Răspunsul: un mesaj de tip JSON care să evidenţieze modificările efectuate de utilizator sau o posibilă eroare.

Cheia de Securitate se obţine după ce numele utilizatorului este trecut în lista bridge ului iar acest lucru se obţine prin intermediul unei cereri de tip POST [16].

Adresa URL	`http://<IP Bridge>/api`
Mesajul JSON	`{"devicetype":"Proiect_de_licență#My Raspberry PI"}`
Tipul de cerere	`POST`

Tabel 1

În momentul în care această comandă este dată va apărea o eroare sub forma "link button not pressed" unde intervine securitatea sistemului sub forma hardware. Noul utilizator trebuie să aibă acces la butonul de pe bridge ca să se emită o nouă cheie de acces. De la apăsare butonului utilizatorul are 30 de secunde să dea comanda de mai sus pentru a obține cheia.

Fiecare bec din sistem îi este atribuit un ID de identificare. Pentru a obține ID urile sistemului se face o cerere de tip GET cu URL ul anterior la care se adaugă "/lights".

Odată obținută cheia de acces și ID-urile becurilor aceastea sunt adăugate la adresa URL pentru ca utilizatorul să înceapă să controleze becurile inteligente prin diverse cereri HTTP de tip PUT [16].

Adresa URL	`http://<IP Bridge>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state`
Mesaj JSON	`{"on":true, "sat":254, "bri":254,"hue":10000}`
Tip de cerere	`PUT`

Tabel 2

```
curl -X PUT -d    '{"on":true,"hue":12000}'
http://192.168.88.245/api/nMvSRBQYknJzKABHhrJrtgbwWDaJKatAJCikdf3b/lights/1/s
tate
```

3.7 REST

REST sau Representational State Transfer este un stil arhitectural ce defineşte un set de constrângeri pentru HTTP. Această idee a fost exprimată pentru prima oară în lucrarea de dizertaţie a lui Roy Fielding pentru a oferi o imagine despre cum ar trebui o aplicaţie web bună să se comporte. În capitolul 5 al lucrării lui intitulat chiar cu numele REST, Roy descrie un număr de condiţii despre cum ar trebui un sistem REST să se comporte. Patru din cele mai importante constrângeri sunt :

Folosirea unei interfeţe uniforme. Sistemele REST se bazează pe idea că orice element din sistem se poate identifica printr-un singur URL şi folosind o anumită metodă din protocolul reţelei .(de exemplu PUT GET pentru HTTP) [32]

Structura Client-Server. Această idee sugerează că un sistem ce se bazează pe REST trebuie să facă o diferenţiere clară între cine este clientul şi cine este serverul într-o reţea. Interfaţa şi cererile utilizatorului intră în categoria clientului, în timp ce accesul datelor şi managementul cum acestea sunt accesate şi transferate se află în domeniul serverului. Această separare clară permite îmbunătăţirea fiecăruia şi de asemenea asigură o cuplare mai bună a celor două

Un protocol "stateless". Ideea de stateless sugerează că operaţiile client server nu trebuie să fie reţinute de server.

Memorie Cache. Sistemul să fie capabil să reţină temporar resursele pentru a reduce latentă sistemului şi îmbunătăţirea performanţei.

Pentru a înţelege idea de bază a acestei arhitecturi putem să ne imaginăm o reţea de pagini web ce sunt interconectate prin hyperlink-uri sau hypertext. Mai întâi utilizatorul se află în starea de index(sau stare iniţială) a paginii web. Prin apăsarea unui hyperlink utilizatorul este transferat către o altă pagină web rezultând în altă stare("state").

Arhitectura REST are avantaje atât din punctul de vedere al clientului cât şi al serverului deoarece foloseşte protocolul HTTP. Astfel , toate interactiuniile se comunică folosind protocolul standard HTTP descris mai sus. (200 este OK).

Din punct de vedere al securităţii, REST se bazează pe criptarea informaţiilor cu ajutorul tehnologiei SSL(Secure Sockets Layer) şi TLS(Transport Layer Security) ce sunt relative bine cunoscute în domeniul IT şi pot fi implementate.SSL este un protocol ce a apărut în anii 90 pentru a preveni atacurile cibernetice ce apăreau din ce în ce mai frecvent pe internet. În 2015 ,cu apariţia TLS , s-a recomandat să se renunţe la SSL din motive de Securitate. TLS este cel mai folosit protocol de comunicaţii din ziua de azi în domeniul IT [33].

Majoritatea web-browserelor ce sunt up-to-date folosesc acest protocol pentru transferul de fişiere, conexiuni VPN şi instant messaging [34].

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).

REST este o arhitectură ce este întâlnită peste tot pe internet în site-uri cu transfer mare de date precum Facebook şi Youtube [35] [36].

CAPITOL 4

EXTINDEREA SISTEMULUI DE CONTROL AL CASEI INTELIGENTE

4.1 INFORMAȚII GENERALE DESPRE SISTEMUL INIȚIAL

Controller-ul de voce împreună cu sistemul de sinteză a vorbirii sunt parte integrată de aceea vor fi prezentate din perspectiva întregului sistem. În prima fază vom expune modul de interconectare al componenetelor şi a modulelor fără a ne axa în acest moment pe vreunul dintre ele.

Aşa cum se poate observa în figură de mai jos, în partea stângă se găsesc toate componentele pe care ni le dorim să le controlăm în cadrul unei case cu alte cuvinte dispozitivele inteligente pe care le deţine o casă (televizor, lumini, geamuri, aer condiţionat, radio, jaluzele). Server-ul constituie creierul întregului sistem, este locul unde sunt interconectate toate componentele sistemului şi în acelaşi timp locul unde se produce procesarea audio. Acesta permite utilizatorului, să-l poată conecta la internet făcând accesul mult mai eficient, confortabil şi rapid. Aşadar, server-ul poate fi utilizat din exterior pentru orice comandă, spre exemplu, dacă utilizatorul doreşte să găsească luminile aprinse în casă când ajunge acasă sau vrea să închidă geamurile pentru că se apropie o furtună, cu ajutorul server-ului conectat la internet toate aceste comenzi pe care utilizatorul şi le doreşte se pot realiza foarte rapid. În partea dreaptă a figurii se regăsesc interfeţele de control,

legătura dintre utilizator şi mediul inteligent. Acestea se aseamănă cu dozele electrice, adică sunt nişte cutii de dimensiuni mici care pot fi conectate prin cablu Ethernet sau fie prin Wireless la punctul central, care au disponibilitatea de a primi o comandă în orice moment, şi au în acelaşi timp o legătură cu difuzoarele din aceeaşi cameră care confirmă vocal comanda realizată cu succes. Aceste dispozitive se clasifică ca fiind dispozitive mobile (tablete/telefoane/telecomandă/etc) sau dispozitive "fixe".

Figura 4-1

În ultima fază sunt prezentate modulele unui server din punct de vedere logic, iar diagrama conţine fiecare modul al sistemului cu accent pe modul în care comunică modulele între ele.

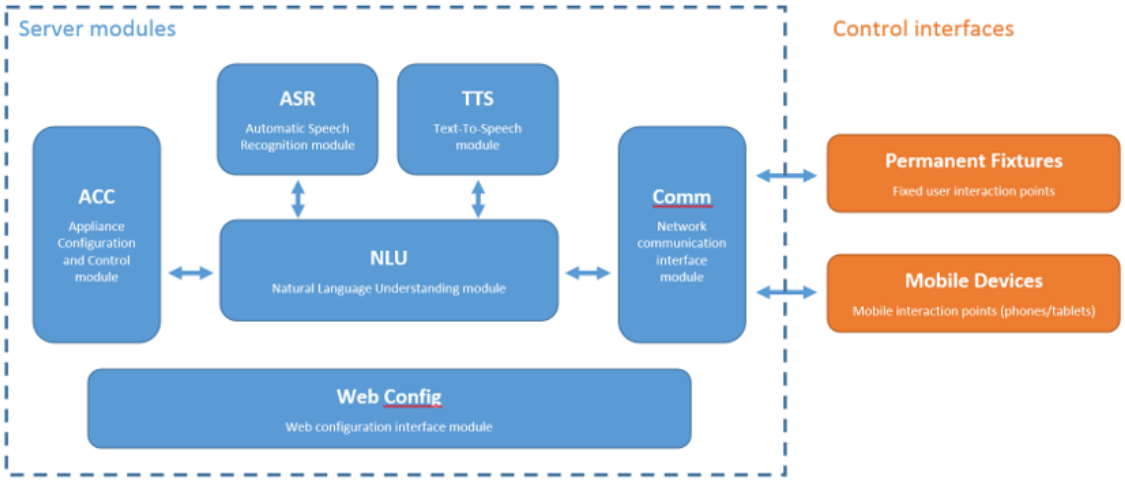


Figura 4-2

Modulul NLU-Natural Language Understanding

Acest modul este considerat a fi elementul central al spaţiului inteligent. Aici este locul unde se procesează mesajul primit (mod 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.

Aşa cum am menţionat şi mai sus interfeţele de control sunt fie dispozitive mobile sau fixe:

Dispozitive mobile

Pot fi definite dispozitive mobile acele dispozitive care beneficiază de avantajul dobândit în urma rulării unei aplicaţii Android şi în acelaşi timp de accesul la server prin Wireless local ori conexiune 3G/4G la server. De exemplu prin intermediul unei tablete sau telefon, deţinătorul unei astfel de sistem inteligent îşi poate controla casă din orice punct, obținând libertatea utilizării dispozitivelor mobile.

Dispozitivele fixe

Un dispozitiv fix este caracterizat prin regăsirea acestuia mereu în acelaşi punct stabilit, punct ce constituie locul de interacţiune plasat de exemplu într-o doză electrică a unei cameră sau chiar direct pe perete la vedere. Întotdeauna, acesta este disponibil pentru o comandă deoarece deţine un microfon ce poate intercepta comanda utilizatorului.Acest dispozitiv fix ascultă un cuvânt cheie prin care îşi dă seama că e momentul în care se începe comandă către casă. În funcţie de configuraţia pe care o deţine în acel moment, fie se procesează semnalul audio local efectuând procesul realizat de modulul ASR local, trimiţând către server doar textul comenzii sau trimițând fluxul audio către server care se ocupă mai departe de procesare [37].

4.2 OBIECTIVELE EXTINDERII SISTEMULUI

Sistemul iniţial nu oferea libertatea utilizatorului de a efectua o gamă largă de comenzi asupra becurilor Phillips Hue. Obiectivele proiectului meu au fost prelucrarea mai multor comenzi ce influenţează becurile unei case inteligente cât şi implementarea unui nou modul capabil sa redea radio. Phillips Hue oferă posibilitatea de a modifica intensitatea luminoasă a unui bec, culoarea luminii şi saturaţia.

Primul task a fost aprinderea şi stingerea unui bec folosind terminalul din Raspberry Pi. Pentru a face acest lucru posibil a fost necesară cheia de acces oferită de Phillips Hue(vezi capitolul Philips Hue) dar şi Id-ul becului cu care vrem să interacţionăm.

Id-ul becurilor s-a aflat cu ajutorul unei cereri HTTP de tip GET la API ul oferit de Hue. Această informaţie venea sub forma unui fişier text în care erau prezentate în mod succinct toate atributele fiecărui bec ce făcea parte din sistemul Hue.

URL ul la care cererea GET va cere informaţii despre sistem trebuie să conţintă IP ul bridge-ului urmat de api,cheia de acces şi secţia lumini deoarece sistemul Hue nu interacţionează numai cu becuri(Hue poate interacţiona şi cu senzori)

http://ip_bridge/ api/cheia_de_acces/lights

Răspunsul va fi de forma :


```
"1": {
        "state": {
            "on": false,
            "bri": 1,
            "hue": 33761,
            "sat": 254,
            "effect": "none",
            "xy": [
                0.3171,
                0.3366
            ],
            "ct": 159,
            "alert": "none",
            "colormode": "xy",
            "mode": "homeautomation",
            "reachable": true
        },
```


Odată cu aflarea ID-urile tuturor becurilor ,următorul pas a fost scrierea unui mesaj JSON ce transmite ce status se dorește a fi modificat şi trimiterea unei cereri de tip PUT către URL-ul descris anterior.

curl -X PUT -d '{"on":false}' http://192.168.88.245/api/nMvSRBQYknJzKABHhrJrtgbwWDaJKatAJCikdf3b/lights/1/state

Un mod de a transmite această cerere pentru a verifica că s-a efectuat conexiunea între Raspberry Pi şi Phillips Hue este prin transmiterea unei comenzi direct din linia de comandă.

Răspunsul la comanda transmisă direct din linia de comanda dacă comanda a fost executata cu succes este:

[{"success":{"/lights/1/state/on":false}}]

Odată ce această comandă este executată cu succes conexiunea dintre cele două componente hardware este stabilită. Cu toate acestea din linia de comandă se poate executa o singură comandă.

Prin urmare pentru a crea un program software ce poate executa mai multe comenzi este necesar crearea unui mesaj JSON ce urmăreşte exact comandă dată de utilizator. Pentru a avea un program

eficient enumerararea unor structuri decizionale pentru fiecare comandă nu reprezintă o soluție optimă. În primul rând din cauza efectului vizual şi al esteticului. În al doilea rand programul primește o latență semnificativă din pricina modulului de recunoaștere a vorbirii si astfel o întarziere suplimentară va provoca un impact negativ asupra promovării produsului. De asemenea, un sistem de casă inteligentă prezintă mai multe module pe langă sistemul de iluminare și radio. Prin urmare taskul meu a fost să găsesc o metodă prin care pot să divizez comanda ce o obţineam sub forma unui string în substringuri ce îmi ofereau informaţii despre ce fel de comandă era [16].

Tipurile de comenzi în cadrul iluminării sunt :

- Aprinderea sau stingerea unui bec
- Aprinderea sau stingerea tuturor becurilor
- Modificarea intensităţii luminoase a unui bec
- Modificarea culorii unui bec

4.2.1 Setul integral de comenzi

Lights - Set on casandra aprinde lumina din $1 casandra porneşte lumina din $1 casandra deschide lumina din $1 Lights - Set off casandra stinge lumina din $1 casandra opreşte lumina din $1 casandra închide lumina din $1 Lights - Set on all casandra aprinde toate luminile casandra porneşte toate luminile casandra deschide toate luminile	Lights - Set absolute intensity casandra setează luminozitatea din $1 la $2 la sută casandra schimbă luminozitatea din $1 la $2 la sută casandra setează intensitatea luminii din $1 la $2 la sută casandra schimbă intensitatea luminii din $1 la $2 la sută Lights - Set increase intensity casandra măreşte luminozitatea din $1 casandra creşte luminozitatea din $1 casandra măreşte intensitatea luminii din $1 casandra creşte intensitatea luminii din $1 Lights - Set decrease intensity casandra micşorează luminozitatea din $1

Lights - Set off all casandra opreşte toate luminile casandra închide toate luminile casandra stinge toate luminile Lights - Set color casandra schimbă culoarea luminii din $1 în $2 casandra aprinde lumina $2 în $1 casandra porneşte lumină $2 în $1 casandra deschide lumina $2 în $1 casandra aprinde lumina $2 din $1 casandra porneşte lumină $2 din $1 casandra deschide lumina $2 din $1	casandra scade luminozitatea din $1 casandra micşorează intensitatea luminii din $1 casandra scade intensitatea luminii din $1 Radio casandra pornește radioul casandra oprește radioul casandra vreau sa ascult radio $1 casandra pune radio $1 casandra schimbă radioul casandra schimbă postul radio

Tabel 3

4.2.2 Organigrama logică a comenzilor pentru sistemul de iluminare

Înainte de a programa trebuie creată o schemă logică pentru a obține un cod fără reduntantă , ce se bazează pe interpretarea comenzilor ilustrate in tabelul anterior.

Comanda vocală dată de utilizator este preluată de modulul de procesarea a comenzilor audio in comenzi string. Comanda string este apoi transmisă funcției parseASRCommand(parsarea comenzii preluată din modulul de recunoaștere vocală). Prin urmare, parametrul funcției este intrarea sistemului. Ieșirea sistemului este mesajul JSON ,ce va fi transmis prin intermediul unei cereri HTTP de tip PUT , care executa comanda prin modificarea parametrilor becurilor.

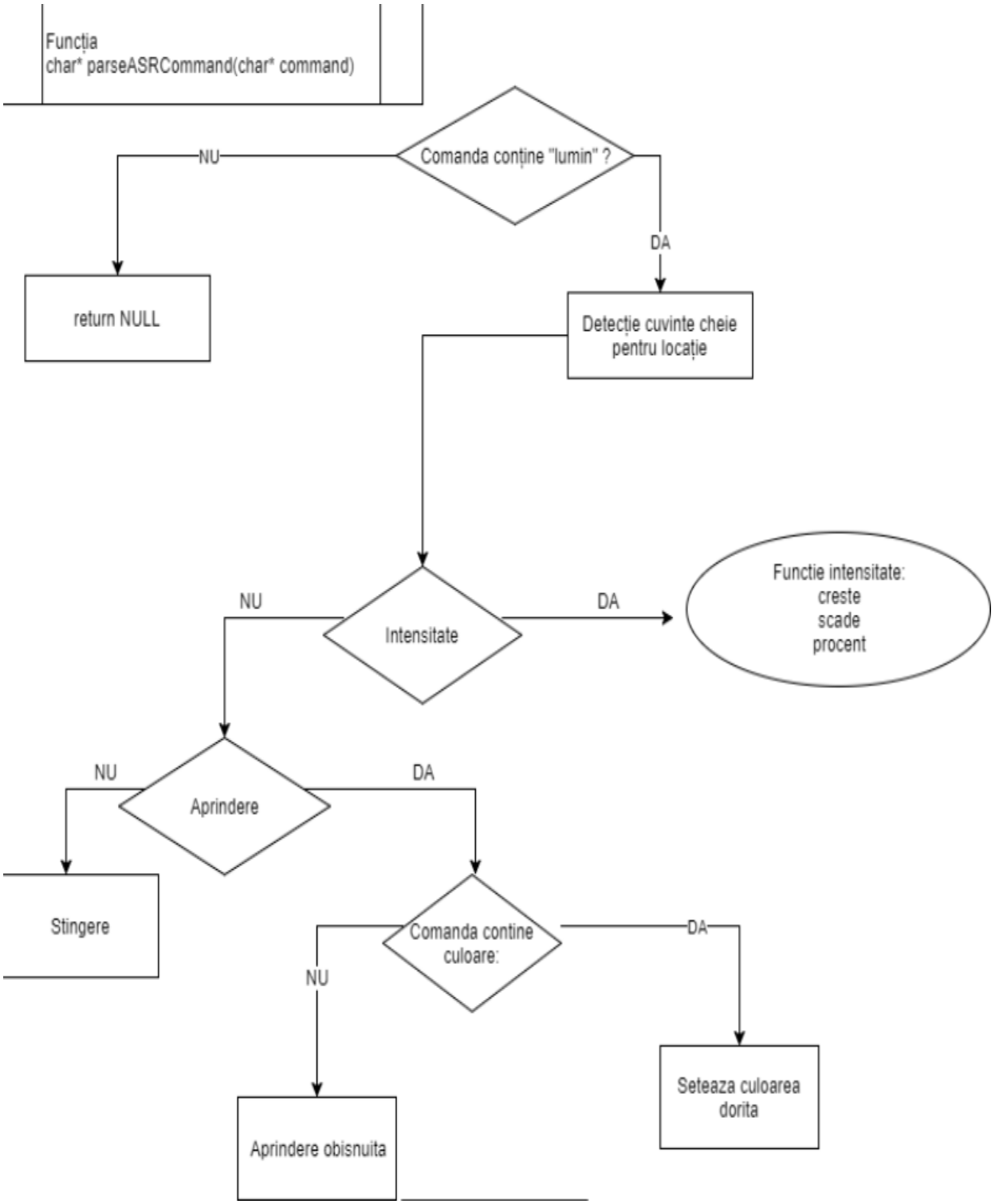

Funcția
char* parseASRCommand(char* command)
Comanda conține "lumin" ?
NU
DA
return NULL
Detecție cuvinte cheie
pentru locație
Intensitate
NU
DA
Functie intensitate:
creste
scade
procent
Aprindere
NU
DA
Stingere
Comanda contine
culoare:
NU
DA
Aprindere obisnuita
Seteaza culoarea
dorita

Figura 4-3

Primul cuvânt cheie ce determină dacă comandă va fi procesată în programul de iluminări este evident substringul "lumin". Cu alte cuvinte toate comenzile ce se încadrau la iluminare aveau în comun acest substring fie prin cuvântul "lumină " fie prin cuvântul "luminozitate". Structura de decizie ce se afla în schemă la această condiție conduce către un bloc numit "Detecție locație" sau către NULL. Ulterior, NULL ul a fost înlocuit cu programul ce se ocupă de comenzile radio.

Blocul de "detecţie locaţie" se bazează pe ideea relativ simplistă că fiecare cameră din casa are un singur bec. Motivul pentru această idee este costul mare al becurilor inteligente şi fiind în stare de prototip acest proiect nu este foarte dificil de implementat nişte grupuri(se numesc groups de către Philips Hue) de becuri pentru fiecare locaţie.

Modul în care am organizat locaţiile în program este în funcţie de ID,asemănător unei baze de date în care tabela se numeşte la modul general cameră.

Figura 4-4

În cazul acestui proiect de licenţă nu era justificată folosirea unei baze de date ce ar fi conţinut tabelă becuri şi tabela camere deoarece avem un bec pentru fiecare cameră.

Pentru identificarea id-ului camerei pe care utilizatorul doreşte să execute o comandă am folosit un fişier de configurare.

Fişierele de configurare se folosesc în programare deoarece nu ţin cont de algoritmul pe care programatorul îl dezvoltă ci de partea hardware. De exemplu , am menţionat în capitolul Philips Hue că fiecare culoare are un cod anume. Aceste informaţii se pun în fişiere de configurare din mai multe motive. Unul din motive este posibilitatea schimbării valorii lor şi posibilitatea de adăugare a noi valorii. În cazul nostru putem să ne imaginăm că dorim să extindem sistemul de iluminare Hue şi la terasă recent construită. Alt motiv este strict de mentenanţă programului şi cât de curat este. Un program ce este simplu, uşor de înţeles, cu puţine linii de cod şi cu un task unic şi specific este mult mai uşor de depanat. Un alt mod de a folosi aceste informaţii în program este de ale declară că variabile globale şi în unele cazuri constante. Programatorul va citi codul mult mai uşor dacă într-o structură de decizie condiţia de egalitate va fi de exemplu ==IDDormitor decât ==4.

Variabile Globale	Fişier de configurare

int BRIMAX=255;
int IDSufragerie=1;
int IDBaie=4;
int IDDormitor=8;
int IDHol=9;
int IDBucatarie=10;
int IDToate=-1;
char text[200];
	roşu 64900
	roşie 64900
	galben 12000
	galbenă 12000
	verde 29000
	albastru 44000
	albastră 44000
	violet 52000
	turcoaz 18000

Tabel 4

Prima coloană din tabel ilustrează modul în care au fost configurate ID urile camerelor în timp ce a doua coloană este un exemplu de fişier de configurare cu extensia .txt a culorilor sistemului Hue şi a codurile sale respective.

Funcţia de detecţie de locaţie are rolul de a returna un ID de tip int şi este alcătuită numai din structuri de decizie.

```
int detectielocatie(char * comandă )
{

	if(strstr(comandă,"sufragerie")!=NULL)
		{
			return IDSufragerie;
		}
	if(strstr(comandă,"baie")!=NULL)
	{
		return IDBaie;
	}

	if(strstr(comandă,"bucătărie")!=NULL)
	{
		return IDBucatarie;
	 }
}
```

În capitolul Philips Hue am menţionat că fiecare bec are ID ul lui şi acest ID va fi inclus în URL ul unic ce va interacţiona cu becul respective. În cazul în care utilizatorul interacţionează cu toate

becurile am folosit un ID aleator negativ deoarece ştiu că toate ID urile pe care sistemul Hue le are sunt pozitive. Acest ID negativ conduce către un URL special pentru a interacţiona cu toate becurile.

Blocul de detecţie locaţie este obligatoriu în sistemul de iluminare al casei deoarece ajută la construirea URL-ului unic.

Următorul bloc verifica dacă utilizatorul doreşte modificarea intensităţii luminoase a unui bec. În cazul în care apar cuvintele cheie "intensitate" ,"luminozitate" sau "la sută" atunci sistemul va verifica dacă utilizatorul doreşte un procent specific al luminozităţii becului sau dacă doreşte să mărească sau să scadă intensitatea becului.

Ultimul bloc sugerează că utilizatorul doreşte fie aprinderea fie stingerea unui bec sau a tuturor becurilor. În cazul aprinderii unui bec există posibilitatea aprinderii becului simplă sau dacă comanda text conţine o culoare din cele care se regăsesc în fişierul de configurare specific atunci becul va fi aprins cu culoarea respectivă.

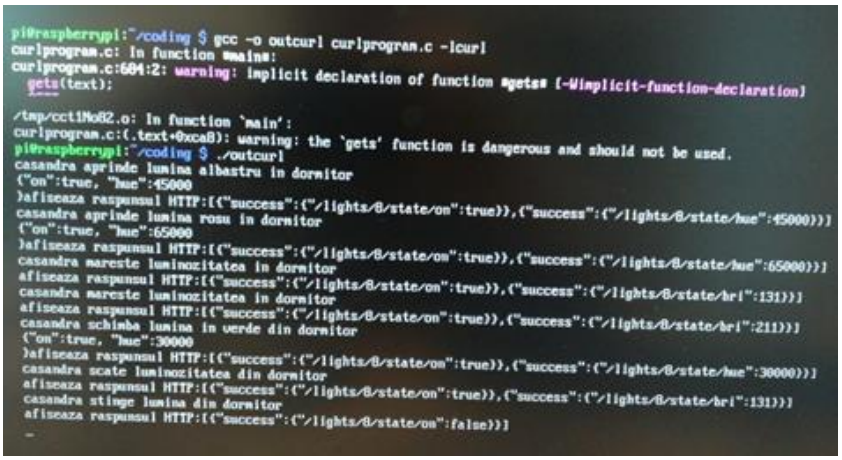


Figura 4-5

Figura de mai sus ilustrează funcționarea cu succes a aplicației scrisă in cod C pentru comenzi oferite de la tastatură. Următorul pas ar fi fost conectarea modulului meu de extindere al comenzilor la modulul de recunoaștere vocală deja implementat. Cu toate acestea am hotărât să portez aplicația in Python.

4.2.3 Portarea aplicației in Python

Motivele pentru care am hotărât să portez aplicaţia în Python sunt exemplificate atât în introducerea lucrării cât şi în subcapitolul dedicat special limbajului Python. Una din regulile pe care se bazează Python este "Simple is better than complex" ce se traduce că "Este mai bine să fie simplu decât complicat". Prin urmare cu cât codul este mai uşor de citit cu atât proiectul Smart Home se poate dezvolta şi altcineva va putea continua munca pe care eu am început-o.

Una din problemele pe care Python le rezolva cu ajutorul "sugar codingului" este chiar lucrul cu stringurile deoarece nu mai foloseşte o funcţie predefinită.

C	Python
strstr(comandă,"lumin")!=NULL	"lumin" in command

Tabel 5

Python a suprascris operatorul "+" astfel încât pus intre doua variabile de tip string acesta nu efectuează o suma ci o concatenare a celor două stringuri.

```
string1="alex are "          string variabila1="alex";
string2="mere"               string variabila2=" are mere";
string3=string1+string3      string variabila3=strcat(variabila1,variabila2);
```

Un alt motiv pentru care portarea este justificată este multitudinea de librării pe care Python le pune la dispoziţie. De exemplu ambele programe (C şi Python) la care am lucrat au funcţia de bază ce are rolul de a trimite cereri HTTP. În timp ce la C am folosit librăria CURL, în Python am folosit pachetul requests. Diferența dintre numărul de linii scrise dintre cele două limbaje are o discrepanță evidentă.

C

```
size_t write_dată_for_http_post(void *ptr , size_t size, size_t nmemb, struct httpRequestData *data)
{
        size_t index=data->size;
        size_t n=(size * nmemb);
        char *tmp;

data->size+=(size *nmemb);
tmp=realloc(data->data,data->size+1);
    if(tmp)
    {
        data->data=tmp;
    }
    else {
        if(data->data)  {    free(data->data);    }
        return 0;
      }
```

```
memcpy((data->data +index),ptr,n);
data->data[data->size]='\0';
return size *nmemb;

}

void send_http(int i,char * JsonObj)
{
	struct httpRequestData httpResponse;
	httpResponse.size=0;
	httpResponse.dată=malloc(4096);
	httpResponse.dată[0]='\0';

	CURL *curl;
	CURLcode httpRequestCode;
	curl_global_inițCURL_GLOBAL_ALL);
	curl=curl_easy_iniț();

	if(curl)
	{

		struct curl_slist *headers=NULL;
		curl_slist_append(headers,"Accept: application/json");
		curl_slist_append(headers,"Content-Type: application/json");
		curl_slist_append(headers,"charsets: utf-8");
if(i==1)
{	//trebuie să reschimbi ip ul pt toate
	curl_easy_setopt(curl,CURLOPT_URL,"http://192.168.88.254/api/1Vuz72w14e-
6U9j0y0IYiFcAu4rVij-Z9w5yO8zo/lights/1/state") ;

		curl_easy_setopt(curl,CURLOPT_CUSTOMREQUEST,"PUT");
		curl_easy_setopt(curl,CURLOPT_HTTPHEADER,headers);
		curl_easy_setopt(curl,CURLOPT_POSTFIELDS,JsonObj);

	curl_easy_setopt(curl,CURLOPT_WRITEFUNCTION,write_dată_for_http_post);
```

```
                curl_easy_setopt(curl,CURLOPT_WRITEDATA, &httpResponse);
                curl_easy_setopt(curl,CURLOPT_USERAGENT,"libcrp/0.1");

        httpRequestCode=curl_easy_perform(curl);

                        if(httpRequestCode!=CURLE_OK)
                        {
                                printf("Error: %s", curl_easy_strerror(httpRequestCode));
                                exit(-1);
                        }
        curl_easy_cleanup(curl);
        } else{
                        exit(-1);
                }
curl_global_cleanup();
printf("afişează răspunsul HTTP:%s \n", httpResponse.data);
free(httpResponse.data);
}
```

Tabel 6

Python

```
def send_http(JsonObj,location):

   if(location==IDBaie or location==IDBucatarie or location==IDDormitor  or location==IDSufragerie ):
      response=requests.put(url+'/lights/'+str(location)+'/state',data=JsonObj)
      print(response.request)
   else :
      response = requests.put(url + '/groups/0/action', data=JsonObj) #sunt toate luminile
      print(JsonObj)
   return
```

Tabel 7

Menționez că funcţia *send_http* din C nu a fost scrisă de mine şi am preluat-o din proiectul deja existent. Am încercat să o înţeleg însă este extrem de dificilă şi greu de parcurs. Un dezavantaj la limbajul scris în C este că programatorul trebuie să aloce si să şteargă memoria pe care o foloseşte în timp ce garbage collectorul la Python se ocupă de asta. Diferenţa dintre cele 2 funcţii este evidentă. De asemenea, un programator îmi va putea prelua codul scris în Python pentru funcţia send_http si să îl poată modifica deoarece este uşor de înţeles. Clasa requests are mai multe metode de tip put,post,get. Din moment ce variabile în Python nu le mai sunt declarate ce fel de tip sunt, variabilă response poate lua orice fel de valoare fără să apară vreo eroare. În cazul de faţă metoda

put are doi parametri şi anume URL ul şi dată ce conţine mesajul JSON ce a fost prelucrat din comandă [38].

Parsare din C în Python se poate face cu ajutorul unor tool-uri ce pot fi luate de pe internet dar Domnul Profesor Coordonator m-a încurajat să îmi "traduc" singur codul pe care deja îl scrisem în C pentru a învăţa un limbaj de programare nou.

4.2.4 Interpretarea unor comenzi text în vederea controlului Philips Hue

În acest subcapitol voi prezenta 2 tipuri de comenzi de iluminare ce necesită mai multă atenţie şi folosesc mai multe funcţii. Primul tip de comandă pe care îl prezint ,utilizatorul doreşte modificarea intensităţii luminoase a unui bec din casă la un procent fix.

Presupunem comandă dată de către utilizator:

casandra setează intensitatea luminii din dormitor la 70 la sută

Urmărind organigrama descrisă mai sus, primul pas este identificarea cuvântului cheie "lumin" urmând apoi să detectăm cameră în care utilizatorul doreşte modificarea intensităţii. Odată, ce variabilă location(locaţie) are un ID ce se afla în sistemul Philips Hue utilizatorul foloseşte cuvântul cheie "intensitate" sau "luminozitate" ce conduce la funcţia brightness.

Funcţia brightness are 3 posibilităţi :

- Setarea intensităţii la un procent fix
- Mărirea intensităţii cu 100 de puncte(aproximativ 40% din intensitatea maximă)
- Scăderea intensităţii cu 100 de puncte

În cazul de faţă comandă dată conţine substringul "la sută" deci programul va urmări prima structura de decizie. Sistemul Philips Hue are o scală de gradare de la 0 la 255 deci numărul ce a fost extras din comandă va fi convertit cu regula de trei simplă în funcţia brightnessconverter. Exemplu 70 se transformă în x=255*70/100=178.5 şi în final cu ajutorul unui cast de tip int numărul va lua partea întreagă a rezultatului şi mesajul JSON va lua forma

```
JsonObj = "{\"on\":true,\"bri\":"+str(178)+"}"
```

Al doilea tip de comandă pe care îl prezint doreşte modificarea culorii unui bec din casă şi o astfel de comandă poate fi enunţată de exemplu în acest mod:

casandra aprinde lumina albastră în sufragerie

Acesta comandă accesează funcţia turn_on_my_lights în care se găseşte o structură de decizie de tip if else. Prima linie a funcţiei accesează funcţia readcolors(citeşte culorile) şi citeşte dintr-un fişier de configurare culoarea şi codul respectiv. Un alt avantaj al limbajului Python este atribuirea prin virgulă a mai multor cuvinte ce au fost separate dintr-un string folosind metoda split.

```
for i în lines:
        namecolor,codecolor=i.split(" ")
        if namecolor în command:
            return codecolor

return 0;
```

Această sintaxa simplă returnează 0 dacă nu se găseşte nicio culoare în comandă sau codul culorii dacă aceasta a fost găsită. Mesajul JSON rezultat va fi de formă :

```
JsonObj="{\"on\":true, \"hue\":"+str(codecolor)+"}"
```

Toate comenzile ce intră în categoria Philips Hue au la bază construirea mesajului JSON şi apoi trimiterea unei cereri de tip PUT prin intermediul funcţiei send_http.

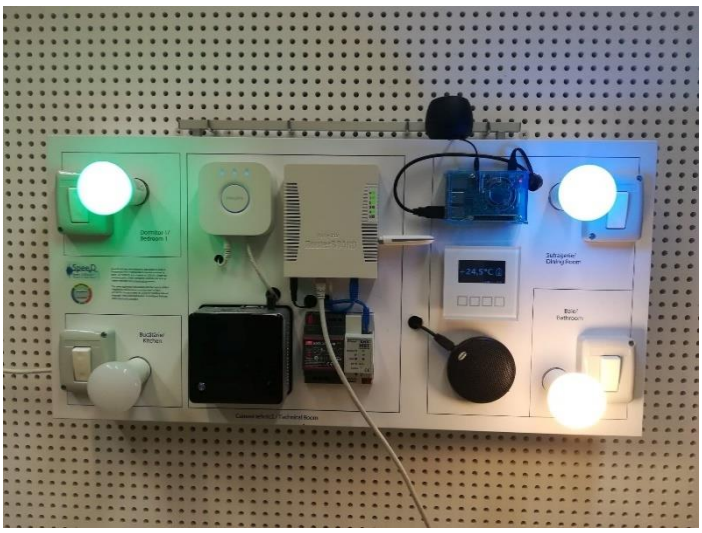


Figura 4-6

4.2.5 Implementarea comenzilor de control radio

Implementarea comenzilor radio în Python a fost făcută cu ajutorul pachetului os pus la dispoziţie de librăriile limbajului. Acest pachet permite executarea unor comenzi în terminal în timp ce programul Python rulează. Reamintesc că pentru redarea unui sistem radio am folosit playerul deja existent Music Player Daemon. Playlistul se salvează în memoria Raspberry Pi asemănător locaţiilor din sistemul Hue [26].

Nume_post_radio Numar_playlist

```
def get_radio_number(command):
    radio_file = open(RADIO_NUMBER_FILE, "r")
    lines = radio_file.readlines()

    for i în lines:
        radio_name, radio_number = i.split(" ")
        if radio_name în command:
            return radio_number
```

Am ales să folosesc din nou un fişier de configurare asemănător culorilor de la sistemul Hue.

Funcţia ce parsează comenzile de radio are 4 structuri de decizie :

- Pornire radio
- Oprire radio
- Schimbare staţie radio în ordinea playlistului
- Setarea unei staţii specifice radio

Dacă utilizatorul oferă o comandă în care specifică postul de radio pe care doreşte să îl asculte atunci comandă va fi interpretată de radio_set_station

```
def radio_set_station(command):
    radio_number=get_radio_number(command)
    os.system("sudo mpc play "+str(radio_number))
```

4.2.6 Implementarea unui program de recunoaștere vocală

Recunoașterea automată a vorbirii (RAV) este un proces ce transforma un semnal vocal într-o succesiune de cuvinte. RAV-ul este împarțit in două etape fundamentale:

- Diarizare (răspunde la întrebarile „cine si cand a vorbit ”)
- Transcriere (răspunde la întrebarea „ce s-a zis”)

Transcrierea este un proces dificil in special din cauza sarcinii de recunoaștere. In acest domeniu este inclus specificitatea limbii, numărul de cuvinte și incertitudinea lingvistică. Limba romană nu conține o bază de date cu resurse acustice dezvoltată. Laboratorul SpeeD a folosit in proiectul anterior o astfel de baza de date la care a adăugat resursele necesare proiectului.

Modelul acustic are rolul de a calcula probabilitatea unui cuvânt vorbit. Un cuvânt este împărțit in foneme care sunt împărțite la rândul lor in unități sub-fonemice numite senone. Rețeaua este antrenată pentru a corecta în cazul in care cuvântul nu a fost rostit corect sau au au apărut interferențe din cauza zgomotului. Modelul acustic din programul anterior folosește modele ascunse Markov (HMM).

Modelul de limbă sau gramatica unui sistem de recunoaștere automată a vorbirii se ocupă cu calcularea probabilităților de apariție a unui cuvânt in funcție de cum a fost antrenată rețeaua. În figura 4-7 de mai jos este exemplificat modelul de gramatică.

```
casandra aprinde luminez în sufragerie
```

se așteaptă să fie cuvantul „lumina” si va transforma în

```
casandra aprinde lumina în sufragerie
```

Modelul fonetic are rolul de a conecta modelul acustic și modelul de limbă. În cele mai multe cazuri, inclusiv cazul de față, modelul fonetic este un model de dictionar. Acest model este ilustrat în figura 4-8.

Programul de recunoaștere vocală se afla deja in sistemul precedent scris in limbaj C , dar era foarte dificil de portat in Python . De aceea am hotărât să construiesc un nou program in Python de recunoaștere a vorbirii. Acest program se folosește de librăria pocketsphinx ca si cel din C dar nu executa keyword-spotting.

Keyword-spotting ,în teorie , obliga programul sa nu asculte conversațiile ce se țin in interiorul casei și să fie activat numai când cuvântul cheie (keyword) a fost folosit. In cazul de fata, cuvântul cheie folosit este evident „casandra”.

Cu toate acestea am implementat un program de recunoaștere vocala ce se folosește de cele trei modele folosite de proiectul anterior [39].

```
config = Decoder.default_config()
config.set_string('-hmm',path.join(MODELDIR+"/am/", 'AM053.cd_cont_4000_64'))
config.set_string('-
jsgf',path.join(MODELDIR+"/grammar/",'FSG_Casandra_v11.gram'))
config.set_string('-
dict',path.join(MODELDIR+"/dictionary/",'homeAutoBIGv3sortUniq.dic'))
```

Acestea sunt configurarile preluate de la proiectul anterior în care folosesc modelele:

- acustice (-hmm) Hidden Markov Model.
- de gramatică (-jsfg).
- de dicționar (-dict) ce conține toate cuvintele si fonemele acestora

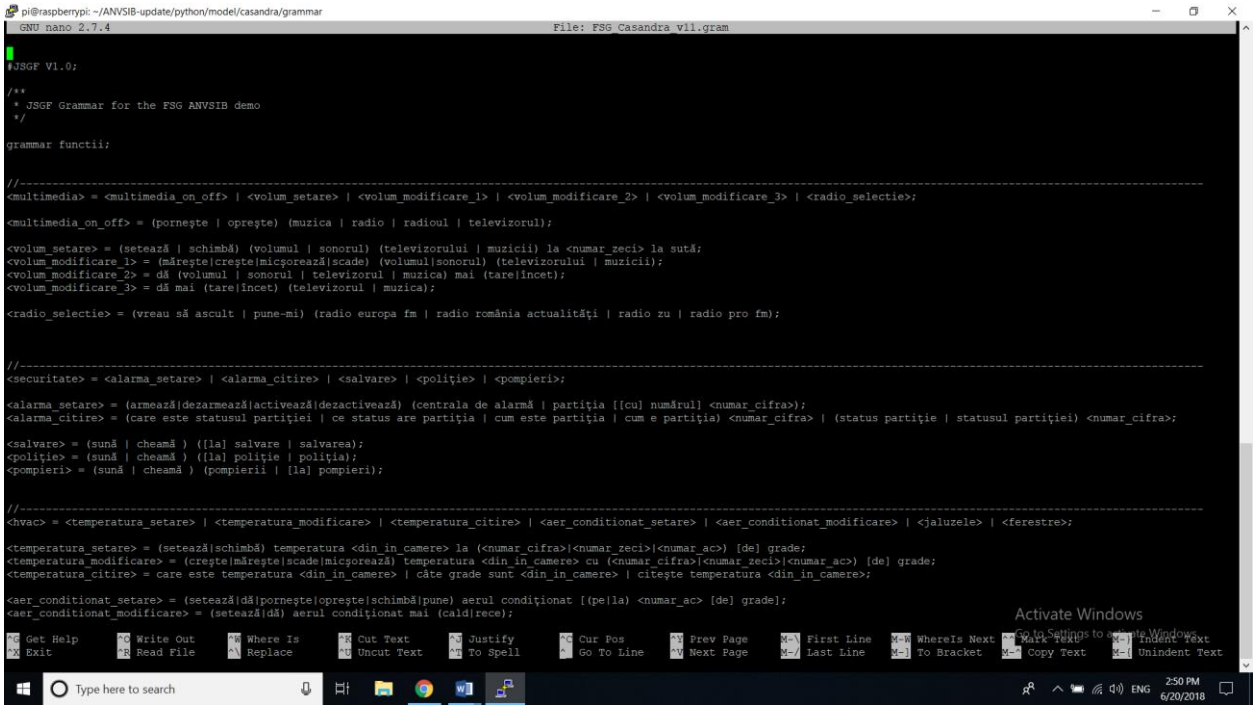


Figura 4-7

Figura 4-8

4.2.7 Implementarea unui program ce accesează modulul de TTS al proiectului anterior

Proiectul anterior era capabil să răspundă utilizatorului printr-un fișier de tip .wav la comenzile date de acestea. Utilizatorul cerea stingerea tuturor luminilor iar modulul TTS prelua comanda , verifica daca aceasta a fost executata cu succes si anunța utilizatorul printr-un mesaj vocal. In cazul in care comanda nu se efectuează deoarece fie modulul de recunoaștere a vorbirii nu a înțeles comanda, fie acesta nu poate sa o execute deoarece nu a fost implementat un modul specific pentru acel tip de comenzi.(exemplu„ casandra pornește televizorul” va rezulta în „Nu pot executa aceasta comandă”).

Configurarea stringului ce este transmis către modulul TTS este făcută in fișierul python intitulat tts.py . Acesta preia comanda dată după ce a fost executată cu succes si o schimbă cu ajutorul unui fișier de configurare. Comanda initială este alterată astfel încât cererea pe care utilizatorul o face este de fapt răspunsul pe care îl primește cu mici diferențe [37].

myanswerwords.txt 204 Bytes

```
porneste pornit
aprinde aprins
schimba schimbat
creste crescut
micsoreaza micsorat
stinge stins
mareste marit
scade scazut
seteaza setat
ascult pus
pune pus
opreste oprit
deschide deschis
inchide inchis
```

Figura 4-9

Stringul inițial conține cuvântul cheie „casandra” ce trebuie șters. Acest lucru se face ușor cu o simpla linie de cod. Partea mai dificilă este schimbarea verbului la persoana a doua in persoana întâi la perfectul compus. Soluția acestei probleme este fișierul de configurare ce are rolul de a verifica daca s-a folosit un verb din baza de date și să îl interschimbe.

```
text_to_speech("am "+str(replacedword)+" "+my_string)
```

Funcția text_to_speech se ocupa cu transferul stringului către modulul TTS. Acest transfer se face asemănător cererilor de la sistemul Philips Hue folosind curl.

```
curl -o audio.wav
"'http://192.168.88.248:8080/synth.php?lang=ro&coder=mlsa&voice=anca&text="
```

Stringul ce este convertit este inserat in URL la final, si apoi noul fisierul audio.wav memorat in Raspberry Pi. Problema pe care am întâmpinat-o în aceasta situație a fost ca URL-urile nu accepta spații. De aceea a fost necesara rezolvarea acestei probleme prin convertirea din nou a stringului pentru a putea fi folosit într-un link URL. Python pune la dispoziție pachetul urllib.parse pentru aceasta problema.

```
html_string=urllib.parse.quote(my_string)
```

Stringul final este dat sub forma unei comenzi din terminal folosind pachetul os si fișierul audio.wav este actualizat si redat.

```
os.system(curl_command+urlsyn+html_string)
os.system("aplay audio.wav")
```

CAPITOL 5
CONCLUZII

5.1 CONCLUZII GENERALE

În urma realizării practice a proiectului de licenţă au fost îndeplinite cu succes obiectivele ce au fost propuse la începutul lucrării. Scopul acestui proiect a fost extinderea comenzilor interpretate de Raspberry Pi în domeniul iluminării unei case inteligente şi construirea unui program ce transmite posturi radio prin intermediul unei boxe conectate la Raspberry Pi.

Am utilizat programul dezvoltat de către colegii mei din cadrul laboratorului de cercetare Speed pentru transformarea unei comanzi sub formă de text într-un fişier wav ce este redat pentru a anunţa utilizatorul că s-a efectuat comanda dată.

Proiectul unei case inteligente comandabila prin voce în română nu există pe piaţă şi acesta poate fi un prototip şi un început foarte bun. Pornind de la un proiect scris în C ce putea să controleze nivelul de iluminare dintr-o încăpere, am extins și optimizat sistemul astfel încât sa adere la standardele moderne IoT. Contribuțiile personale aduse acestui proiect sunt următoarele:

- Am implementat noi comenzi de iluminare
- Am implementat controlul unui radio.
- Portarea codului in Python.

- Optimizarea timpului de răspuns de la 10s la timp real
- Am proiectat un sistem de recunoaștere automata a vorbirii

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. Am învățat sa lucrez organizat si distribuit, fapt care m-a ajutat sa finalizez proiectul. A fost o experiență plăcută să lucrez cu cei din cadrul laboratorului SpeeD care m-au ajutat de fiecare dată când am avut întrebări sau când întâlneam obstacole peste care nu puteam trece fără să îmi fie explicate anumite noțiuni teoretice. Aș dori să mulțumesc laboratorului pentru infrastructura pe care mi-au pus-o la dispoziție pentru dezvoltarea acestui proiect.

5.2 POSIBILITĂȚI VIITOARE DE DEZVOLTARE

Primul concept ce trebuie înţeles la un proiect cu tema casă inteligentă este evoluţia dinamică pe care acesta poate să o aibă. Din punct de vedere al posibilităţilor hardware, sistemul poate fi extins atât în interiorul casei cât şi afară.

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

- Conectarea unui termostat ce se afla în legătură cu sistemul de încălzire(centrală) şi/sau sistemul de ventilaţie(aer condiţionat) ce are rolul menţinerii unei temperaturi ce este setată de utilizator.
- Conectarea unor jaluzele inteligente ce împiedică trecerea luminii la comandă vocală a utilizatorului.
- Conectarea unor geamuri inteligente ce se deschid şi se închid electric prin comandă vocală
- Conectarea unui sistem de telefonie pentru cazuri de urgenţă
- Conectarea la un televizor inteligent
- Conectarea unui sistem de irigaţii pentru grădină

Din punct de vedere al proiectului la care am lucrat personal consider că se pot face mai multe îmbunătăţiri. În primul rând , programul de recunoaştere de vorbire nu caută cuvântul cheie "Casandra" ci ascultă toate tipurile de zgomote sau conversaţii într-o casă.

Un alt dezavantaj este lipsa securităţii atât a sistemului Philips Hue cât şi a Raspberry Pi. Fişierele mele de configurare şi variabilele globale ar trebui să fie securizate şi criptate pentru a împiedica un posibil atac.

O îmbunătăţire importantă ce poate fi adăugată la proiect este dezvoltarea unui filtru capabil să respingă sunetele de o anumită frecvenţă şi să capteze numai frecvenţele ce se găsesc în domeniul vorbirii. Acest defect se observă cel mai bine la pornirea radioului. În momentul în care radioul reda sunetul prin boxa acesta este preluat de microfon şi rezultă fie în incapacitatea microfonului de a mai asculta comenzile utilizatorului fie în întârzieri mari.

BIBLIOGRAFIE

[1] „Anti-Stress lights,” [Interactiv]. Available: https://www.stresslighttherapy.com/technology. [Accesat 20 06 2018].

[2] [Interactiv]. Available: https://en.wikipedia.org/wiki/Internet_of_things. [Accesat 20 06 2018].

[3] „Raspberry Pi,” [Interactiv]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi#Model_B. [Accesat 10 03 2018].

[4] „RaspberryPi3,” [Interactiv]. Available: https://www.robofun.ro/raspberry-pi-v3.

[5] „Raspban,” [Interactiv]. Available: https://www.raspberrypi.org/downloads/raspbian/. [Accesat 20 06 2018].

[6] „sudo,” [Interactiv]. Available: https://en.wikipedia.org/wiki/Sudo. [Accesat 29 06 2018].

[7] „Linux,” [Interactiv]. Available: https://ryanstutorials.net/linuxtutorial/. [Accesat 12 12 2017].

[8] „Editors,” [Interactiv]. Available: https://www.raspberrypi.org/documentation/linux/usage/text-editors.md. [Accesat 28 06 2018].

[9] „Nano,” [Interactiv]. Available: https://www.nano-editor.org/. [Accesat 21 06 2018].

[10] „Textedit,” [Interactiv]. Available: https://www.raspberrypi.org/documentation/linux/usage/text-editors.md. [Accesat 10 05 2018].

[11] „Wireless,” [Interactiv]. Available: https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md. [Accesat 15 11 2017].

[12] „Microfon,” [Interactiv]. Available: www.samsontech.com. [Accesat 15 04 2018].

[13] [Interactiv]. Available: www.dateks.lv. [Accesat 07 02 2018].

[14] „JSON,” [Interactiv]. Available: https://en.wikipedia.org/wiki/JSON. [Accesat 16 04 2018].

[15] „Philips Hue,” [Interactiv]. Available: https://www.developers.meethue.com/philips-hue-api. [Accesat 11 03 2018].

[16] „Hue,” [Interactiv]. Available: https://en.wikipedia.org/wiki/Philips_Hue. [Accesat 14 04 2018].

[17] „Limbaj C,” [Interactiv]. Available: https://ro.wikipedia.org/wiki/C_(limbaj_de_programare). [Accesat 14 12 2017].

[18] „Python,” [Interactiv]. Available: https://www.python.org/. [Accesat 22 12 2017].

[19] „History,” [Interactiv]. Available: https://en.wikipedia.org/wiki/History_of_Python. [Accesat 11 11 2018].

[20] „Putty,” [Interactiv]. Available: https://ro.wikipedia.org/wiki/PuTTY. [Accesat 20 01 2018].

[21] „git2,” [Interactiv]. Available: https://docs.gitlab.com/ee/university/#beginner.

[22] „Git,” [Interactiv]. Available: https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf. [Accesat 18 12 2017].

[23] „gitlab,” [Interactiv]. Available: https://about.gitlab.com/2015/04/21/gitlab-on-raspberry-pi-2/. [Accesat 15 04 2018].

[24] „Istoria radioului,” [Interactiv]. Available: http://www.ziare.com/cultură/istoria-culturii-si-civilizatiei/ziua-cand-s-a-nascut-radioul-acum-mai-bine-de-un-secol-documentar-1206732. [Accesat 12 11 2017].

[25] „Netstreaming,” [Interactiv]. Available: https://ro.wikipedia.org/wiki/Radio. [Accesat 12 04 2018].

[26] „MPD,” [Interactiv]. Available: https://en.wikipedia.org/wiki/Music_Player_Daemon. [Accesat 18 05 2018].

[27] „MPC,” [Interactiv]. Available: https://linux.die.net/man/1/mpc. [Accesat 16 01 2018].

[28] „HTTP1,” [Interactiv]. Available: https://en.wikipedia.org/wiki/List_of_HTTP_header_fields. [Accesat 16 05 2018].

[29] „requests,” [Interactiv]. Available: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods. [Accesat 21 12 2017].

[30] „HTTP,” [Interactiv]. Available: developer.mozilla.org. [Accesat 11 10 2018].

[31] „HTTP,” [Interactiv]. Available: www.codecademy.com. [Accesat 14 10 2017].

[32] „REST11,” [Interactiv]. Available: https://bbvaopen4u.com/en/actualidad/rest-api-what-it-and-what-are-its-advantages-project-development. [Accesat 10 11 2017].

[33] „SSL,” [Interactiv]. Available: https://www.pickaweb.co.uk/kb/what-is-an-ssl-certificate/.

[34] „SSL,” [Interactiv]. Available: https://searchsecurity.techtarget.com/definition/Secure-Sockets-Layer-SSL. [Accesat 12 12 2018].

[35] „REST,” [Interactiv]. Available: https://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer. [Accesat 19 06 2018].

[36] [Interactiv]. Available: https://www.linkedin.com/learning/learning-rest-apis/what-is-a-rest-api. [Accesat 24 12 2017].

[37] „Raport ştiinţific şi ethnic în extenso : Sistem de Asistenta pentru Clădiri Inteligente, controlat prin Voce şi Vorbire Naturala”.

[38] „StackOverFlow,” [Interactiv]. Available: https://stackoverflow.com/questions/3516560/coming-from-c-how-should-i-learn-python. [Accesat 21 12 2018].

[39] „Sphinx,” [Interactiv]. Available: http://speed.pub.ro/speed3/wp-content/uploads/2013/04/Indrumar-de-proiect-PCDTV-v4.pdf. [Accesat 15 12 2017].

ANEXE

Cod Sursă C

```
#include <curl/curl.h>
#include <stdio.h>
#include<stdlib.h>
#include<string.h>


struct httpRequestData{
        size_t size;
        char *dată;

};
struct Culoare{
char nume[25];
char cod[10];

}culori[20];
int BRIMAX=255;

int  IDSufragerie=1;
int  IDBaie=4;
int  IDDormitor=8;
int  IDHol=9;
int  IDBucatarie=10;
int  IDToate=-1;
```

```
int nrculori;
char  text[200];


size_t write_dată_for_http_post(void *ptr , size_t size, size_t nmemb, struct httpRequestData *dată)
{

                size_t index=dată->size;
                size_t n=(size * nmemb);
                char *tmp;

data->size+=(size *nmemb);
tmp=realloc(dată->dată,dată->size+1);

        if(tmp)
        {
                data->dată=tmp;
        }
        else {
                if(dată->dată)      {          free(dată->dată);  }


                return 0;
            }

memcpy((dată->dată +index),ptr,n);
data->dată[dată->size]='\0';
return size *nmemb;

}

void send_http(int i,char * JsonObj)
{
        struct httpRequestData httpResponse;
        httpResponse.size=0;
        httpResponse.dată=malloc(4096);
```

```
        httpResponse.dată[0]='\0';

        CURL *curl;
        CURLcode httpRequestCode;
        curl_global_inițț(CURL_GLOBAL_ALL);
        curl=curl_easy_inițț();



        if(curl)
        {

                struct curl_slist *headers=NULL;
                curl_slist_append(headers,"Accept: application/json");
                curl_slist_append(headers,"Content-Type: application/json");
                curl_slist_append(headers,"charsets: utf-8");




if(i==1)
{       //trebuie să reschimbi ip ul pt toate
        curl_easy_setopt(curl,CURLOPT_URL,"http://192.168.88.254/api/1Vuz72w14e-6U9j0y0IYiFcAu4rVij-Z9w5yO8zo/lights/1/state") ;

}
if(i==4)
{

        curl_easy_setopt(curl,CURLOPT_URL,"http://192.168.88.254/api/1Vuz72w14e-6U9j0y0IYiFcAu4rVij-Z9w5yO8zo/lights/4/state") ;

}



if(i==8)
{
```

```
        curl_easy_setopt(curl,CURLOPT_URL,"http://192.168.88.254/api/NIpXO5s6m83jAL6o2qCmI3WD4YiO
6e9qkj0x2dIZ/lights/8/state") ;


}



if(i==9)

{

        curl_easy_setopt(curl,CURLOPT_URL,"http://192.168.88.254/api/1Vuz72w14e-6U9j0y0IYiFcAu4rVij-
Z9w5yO8zo/lights/9/state") ;

}



if(i==10)

{


        curl_easy_setopt(curl,CURLOPT_URL,"http://192.168.88.254/api/NIpXO5s6m83jAL6o2qCmI3WD4YiO
6e9qkj0x2dIZ/lights/10/state") ;


}

//-1 înseamnă toate

if(i==-1)

{

curl_easy_setopt(curl,CURLOPT_URL,"http://192.168.88.254/api/NIpXO5s6m83jAL6o2qCmI3WD4YiO6e9qkj0
x2dIZ/groups/0/action") ;


}


                curl_easy_setopt(curl,CURLOPT_CUSTOMREQUEST,"PUT");

                curl_easy_setopt(curl,CURLOPT_HTTPHEADER,headers);

                curl_easy_setopt(curl,CURLOPT_POSTFIELDS,JsonObj);

                curl_easy_setopt(curl,CURLOPT_WRITEFUNCTION,write_dată_for_http_post);

                curl_easy_setopt(curl,CURLOPT_WRITEDATA, &httpResponse);

                curl_easy_setopt(curl,CURLOPT_USERAGENT,"libcrp/0.1");


        httpRequestCode=curl_easy_perform(curl);


                        if(httpRequestCode!=CURLE_OK)
```

```
                        {
                                printf("Error: %s", curl_easy_strerror(httpRequestCode));
                                exit(-1);
                        }

        curl_easy_cleanup(curl);

        } else{
                        exit(-1);

                }

curl_global_cleanup();
printf("afişează răspunsul HTTP:%s \n", httpResponse.dată);
free(httpResponse.dată);
}


int citireculori(FILE * ptoFile)
{

        int line=0;
        char input[50];
        int i=0;

        while(fgets(input,50,ptoFile))
        {
                char * token=strtok(input," ");
                strcpy(culori[line].nume,token);
                token=strtok(NULL," ");
                strcpy(culori[line].cod,token);
        line++;
        }

return line;
}
```

```
int detectielocatie(char * comandă )
{

                                        if(strstr(comandă,"sufragerie")!=NULL)
                                        {
                                        return IDSufragerie;
                                        }
                                    if(strstr(comandă,"baie")!=NULL)
                                        {
                                        return IDBaie;
                                        }

                                        if(strstr(comandă,"bucătărie")!=NULL)
                                         {
                                                return IDBucatarie;
                                         }
                                if(strstr(comandă,"dormitor")!=NULL)
                                         {
                                                return IDDormitor;
                                 }
                                        if(strstr(comandă,"hol")!=NULL)
                                    {
                                                return IDHol;
                                    }
                                        if(strstr(comandă,"toate")!=NULL)
                                         {
                                              return IDToate;

                                    }

}

int briconverter(int număr)
{
int rezultat;
rezultat=număr*BRIMAX/100;
```

```
return rezultat;
}



void  intensitate(char * comandă,int locaţie)
{
                if(strstr(comandă,"la sută")!=NULL)
                {
                        int i=0;
                        int număr=0;
                        char aux=i+'0';

                        for(i=0;i<strlen(comandă);i++)
                        {
                         if(comandă[i]>='0'&& comandă[i]<='9')
                         {
                                 număr=număr*10+(comandă[i]-'0');
                          }
                        }
                        printf("%d\n",număr);
                        număr=briconverter(număr);
                        printf("%d\n",număr);


                        char str[5];
                        sprintf(str,"%d",număr);



                        char JsonObj[100]="{\"on\":true,\"bri\":";
                        strcat(JsonObj,str);
                        strcat(JsonObj,"}");
                        printf("\n\n %s",JsonObj);
                        //send_http(locaţie,JsonObj);

    }
```

```
        else if((strstr(comandă,"creşte")!=NULL)||(strstr(comandă,"măreşte")!=NULL))
                        {       char *    JsonObj="{\"on\":true,\"bri_inc\":80}";
                                send_http(locaţie,JsonObj);
                }
                        else{
                                        char *    JsonObj="{\"on\":true,\"bri_inc\":-80}";
                                        send_http(locaţie,JsonObj);

                                }
}

void  turnon(char * comandă,int locaţie)
{       int ok=0;
        int index=0;
        int k;
        int aux=nrculori-1;

 while(aux>=0)
 {      if(strstr(comandă,culori[aux].nume)!=NULL)
        {  ok=1;
           k=aux;
      }
 aux--;
  }

        if(ok==0)
        {       char * JsonObj="{\"on\":true}";
                send_http(locaţie,JsonObj);
        }
        else
                {
                char JsonObj[100] ="{\"on\":true, \"hue\":";
                strcat(JsonObj,culori[k].cod);
                strcat(JsonObj,"}");
                printf("%s",JsonObj);
                send_http(locaţie,JsonObj);
                }
```

```
ok=0;
index=0;
}

void turnoff(char * comandă,int locaţie)
{
	char * JsonObj="{\"on\":false}";
	send_http(locaţie,JsonObj);
}

char * parseASRCommand(char * comandă)
{		char * răspuns;
	if(strstr(comandă,"lumin")!=NULL)
	{
		int locaţie=detectielocatie(comandă);

			if((strstr(comandă,"luminozitate")!=NULL)||(strstr(comandă,"intensitate")!=NULL))
			{
				intensitate(comandă,locaţie);
		}
			else
if((strstr(comandă,"aprinde")!=NULL)||(strstr(comandă,"porneşte")!=NULL)||(strstr(comandă,"deschide")!=NULL)|
|(strstr(comandă,"schimbă")!=NULL))
			{
				turnon(comandă,locaţie);
			}
			else
if((strstr(comandă,"stinge")!=NULL)||(strstr(comandă,"opreşte")!=NULL)||(strstr(comandă,"închide")!=NULL))
				{
				 turnoff(comandă,locaţie);
				}



	}
	else return "Nu pot executa această comandă";
```

```
return comandă;
}

void afisareRezultat(char * mesaj)
{
printf(mesaj);
}


    int main()
{
        int x=1;
        int y=1;
        char * JsonObj="{\"on\":true }";

        // citire fişier de configurare
        FILE * ptoFile=fopen("culorilemele.txt","r");
        nrculori=citireculori(ptoFile);
        fclose(ptoFile);

        while(1==1)
        {
                gets(text);
                char  * comandă=text;
                char * mesaj;

                if(strstr(comandă,"casandra")!=NULL)
                {
                mesaj=parseASRCommand(comandă);
                }

        }
                //afisareRezultat(mesaj);


        return 0;
```

```
}
```

Cod Python

LightsNLU

```
from tts import *
from radioNLU import *
import requests
import re
import json

BRIMAX=255
IDSufragerie=2
IDBaie=4
IDDormitor=3
#IDHol=9
IDBucatarie=1
IDToate=-1
COLORS_FILE_NAME="mycolors.txt"
DORMITOR_STRING="dormitor"



url='http://192.168.88.245/api/nMvSRBQYknJzKABHhrJrtgbwWDaJKatAJCikdf3b'
number=0

#JsonObj = "{\"on\":true}"
#response = requests.PUT('http://192.168.88.245/api/nMvSRBQYknJzKABHhrJrtgbwWDaJKatAJCikdf3b/lights/1/state', dată=JsonObj)

#print (response.url)


def send_http(JsonObj,location):
```

```
    if(location==IDBaie or location==IDBucatarie or location==IDDormitor  or location==IDSufragerie ):
        response=requests.PUT(url+'/lights/'+str(location)+'/state',dată=JsonObj)
        print(JsonObj)
        print(response.request)
        print(response.url)
    else :
        response = requests.PUT(url + '/groups/0/action', dată=JsonObj) #sunt toate luminile
        print(JsonObj)
    return



def readcolors( command):
    text_file = open(COLORS_FILE_NAME, "r")
    lines = text_file.readlines()

    for i în lines:
        namecolor,codecolor=i.split(" ")
        if namecolor în command:
            return codecolor



    return 0;



def detectlocation (command):
    if (DORMITOR_STRING în command):
        return IDDormitor
    if("sufragerie" în command):
        return IDSufragerie
    if ("baie" in command):
        return IDBaie
   # if ("hol" în command):
    #    return IDHol
    if ("bucatărie" în command):
        return IDBucatarie
    if ("toate" în command):
        return IDToate
```

```
def brightnessconverter(number):
    result=number*BRIMAX/100;
    return result

def brightness(command,location):

    if ("la sută" în command):
        number=int(''.join(filter(str.isdigit, command)))    #îmi ia numărul din string
        number=int(brightnessconverter(number))
        JsonObj = "{\"on\":true,\"bri\":"+str(number)+"}"
        send_http(JsonObj,location)
        command_to_answer(command)
    else :
        if("crește" în command or "mărește"în command):
            JsonObj = "{\"on\":true,\"bri_inc\":100"+"}"
            send_http(JsonObj, location)
            command_to_answer(command)
        else :
            if ("scade" în command or "micşorează"în command):
                JsonObj = "{\"on\":true,\"bri_inc\":-100"+"}"
                send_http(JsonObj, location)
                command_to_answer(command)
    return

def turn_on_my_lights(command,location):

    codecolor=readcolors(command)
    if(codecolor!=0):
        JsonObj="{\"on\":true, \"hue\":"+str(codecolor)+"}"
        send_http(JsonObj,location)
        command_to_answer(command)
    else:
        JsonObj="{\"on\":true}"
        send_http(JsonObj,location)
        command_to_answer(command)
```

```
    return

def turn_off_my_lights(command,location):

    JsonObj = "{\"on\":false}"
    send_http( JsonObj,location)
    command_to_answer(command)

    return

def parseASRCommand(command):

    if("lumin" în command):
        location=detectlocation(command)
        if("luminozitate" în command or "intensitate" în command):
            brightness(command,location)
        if("aprinde" în command or "porneşte" în command or "deschide" în command or "schimbă" în command):
            turn_on_my_lights(command,location)
        if("opreşte" în command or "stinge" în command or "închide" în command):
            turn_off_my_lights(command,location)
    else :
        if("radio" în command ):
            parseRadioCommand(command)
        else :
            print("Nu pot executa această comandă")
   # command_to_answer(command)



#while(1==1):
 #command=input()

# readcolors(command)

 #if("casandra" în command):
 #  parseASRCommand(command)
 #else: prinţ("Nu pot executa o comandă fără a începe cu casandra")
```

```
#r = requests.get('https://api.github.com/events')


#req=requests.PUT('http://192.168.88.245/api/nMvSRBQYknJzKABHhrJrtgbwWDaJKatAJCikdf3b/lights/1/state',
dată=JsonObj)
```

RADIONLU

```
import os

RADIO_NUMBER_FILE="myradiostations.txt"
def radio_turn_on():
    os.system("sudo mpc play")

def radio_turn_off():
    os.system("sudo mpc stop")

def radio_change_station():
    os.system("mpc next")

def get_radio_number(command):
    radio_file = open(RADIO_NUMBER_FILE, "r")
    lines = radio_file.readlines()

    for i în lines:
        radio_name, radio_number = i.split(" ")
        if radio_name în command:
            return radio_number

    return 0;

def radio_set_station(command):
    radio_number=get_radio_number(command)
    os.system("sudo mpc play "+str(radio_number))
```

```
def parseRadioCommand(command):
    if "pornește" în command:
        radio_turn_on()
    if "schimbă" în command :
        radio_change_station()
    if "oprește" în command:
        radio_turn_off()
    if "pune" în command or "ascult" în command:
        radio_set_station(command)

#while(1==1):
   # command=input()
   # parseRadioCommand(command)
```

TTS

```
from urllib.parse import urlparse
import os
import urllib

urlsyn='"http://192.168.88.248:8080/synth.php?lang=ro&coder=mlsa&voice=anca&text="
curl_command="curl -o audio.wav "

MY_WORDS_FILE="myanswerwords.txt"

def play_wav():
    os.system("aplay audio.wav")

def text_to_speech(my_string):
    html_string=urllib.parse.quote(my_string)
    html_string +="""
    os.system(curl_command+urlsyn+html_string)
    play_wav()


def switch_word(myword):
```

```
    word_file=open(MY_WORDS_FILE,"r")
    lines=word_file.readlines()

    for i în lines:
        wtochange,wanswer=i.split(" ")
        if wtochange==myword :
            return wanswer


def command_to_answer(command):
    #9 pentru a scoate casandra din comandă
    answer=command[9:]
    v=answer.split(" ",1)
    my_string=str(v[1])
    replacedword=switch_word(v[0])
    replacedword=replacedword.rstrip()
    answer.replace(v[0],replacedword)
    text_to_speech("am "+str(replacedword)+" "+my_string)
```

Program(Main)

```
//recunoaşterea de vorbire

from lightsNLU import *
from os import environ, path
import pyaudio

from pocketsphinx.pocketsphinx import *
from sphinxbase.sphinxbase import *

MODELDIR = "/home/pi/ANVSIB-update/python/model/casandra"

config = Decoder.default_config()
config.set_string('-hmm', path.join(MODELDIR+"/am/", 'AM053.cd_cont_4000_64'))
config.set_string('-jsgf', path.join(MODELDIR+"/grammar/", 'FSG_Casandra_v11.gram'))
config.set_string('-dict', path.join(MODELDIR+"/dictionary/", 'homeAutoBIGv3sortUniq.dic'))
config.set_string('-kws',path.join(MODELDIR,'keywords.txt'))
```

```
decoder = Decoder(config)
p = pyaudio.PyAudio()
stream = p.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True, frames_per_buffer=1024)
stream.start_stream()
în_speech_bf = False
decoder.start_utt()
print("LISTENING...")
#print(keyword)

while True:
    buf=None
    buf = stream.read(1024,exception_on_overflow = False)
    if buf:
        decoder.process_raw(buf, False, False)
        if decoder.get_în_speech() != în_speech_bf:
            în_speech_bf = decoder.get_în_speech()
            if not în_speech_bf:
                decoder.end_utt()
                try:
                    if decoder.hyp().hypstr != None:
                        parseASRCommand (decoder.hyp().hypstr)
                        print(decoder.hyp().hypstr)
                except AttributeError:
                    pass
                except IOError:
                    pass
                decoder.start_utt()
    else:
        break
decoder.end_utt()
```