UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Integració vocal amb tecnologia SPHINX. Aplicació pràctica a vivendes intel ligents.

Projecte Final de Carrera

Enginyeria Tècnica d'Informàtica Aplicada de Sistemes

Autor: Manel Mellado Romero

Director: Joan Fons i Cors

Curs 2013-2014

Resum

Aquest projecte tracta el desenvolupament d'una interfície vocal per a un habitatge domotitzat, mitjançant el programari lliure SPHINX com a base del reconeixement de la veu. El reconeixement de veu es processa per poder dictar les ordres d'una forma intuïtiva i amb llenguatge humà.

Paraules clau: Domòtica, reconeixement de veu, Sphinx, codi obert, speech to text, STT, smarthome

.

Taula de continguts

1. Introducció

Este document és la memòria de realització del projecte final de carrera per l'alumne Manel Mellado Romero de l'Escola Tècnica Superior d'Informàtica Aplicada de la Universitat Politècnica de València.

En aquest capítol comentarem la motivació que ha portat a la realització d'aquest projecte, així com els seus propòsits i objectius a complir.

1.1 Motivació

La selecció d'aquest projecte és motivada per l'interés que em desperta l'estudi del reconeixement de veu. La fascinació per aquest àmbit ve donada sobretot perquè és una tecnologia de nova introducció i que encara no ha sigut del tot desenvolupada, amb la qual cosa encara queda molt per treballar i descobrir respecte d'ella.

Com a to anecdòtic, em crida el caire futurista del projecte, en el qual s'aconsegueix la comunicació verbal amb màquines que responen mitjançant accions concretes, ja que representa un primer pas cap a la intel·ligència artificial. Però principalment m'atreu pel fet que es tracta d'una tecnologia en part per desenvolupar, i que pot tindre un gran nombre d'aplicacions com ara facilitar la vida d'aquelles persones que no es poden valdre per si mateixes, com poden ser els discapacitats físics.

A més a més, la utilització de l'API SPHINX com a capturador de veu em fou realment atractiva donat que és un programari de codi obert.

1.2 Propòsit

El propòsit principal d'aquest projecte naix de la necessitat de poder controlar una selecció de dispositius sense una interfície gràfica o un dispositiu de control remot. A més a més, la intenció és crear un intèrpret que no sols reconega les ordres vocals, sinó que siga fàcil d'utilitzar per l'usuari amb un llenguatge natural.

Com a cas pràctic s'utilitzarà l'entorn d'una casa domotitzada, amb un servidor funcional que controla els diferents dispositius. Fins el moment, les ordres són donades a través d'una interfície gràfica o d'estímuls provinents de diferents sensors.

El projecte, per tant, es centrarà en la creació d'un intèrpret d'ordres, capturades en llenguatge natural, que s'encarregarà de la seua traducció i posterior comunicació al servidor central de l'habitatge.

1.3 Objectius

L'objectiu principal d'aquest projecte és implementar una interfície vocal de comunicació entre l'usuari i l'habitatge domotitzat d'una forma senzilla i intuïtiva per a qualsevol persona, sense necessitat d'utilitzar i recordar ordres predefinides, sols amb el llenguatge natural humà.

Altre objectiu secundari es crear-lo amb un programari de codi obert com a base, evitant així la dependència i cost que pogués crear-se amb la utilització d'un altre de tipus comercial.

Per últim objectiu, es proposa crear la interfície de control de veu en llengua valenciana.

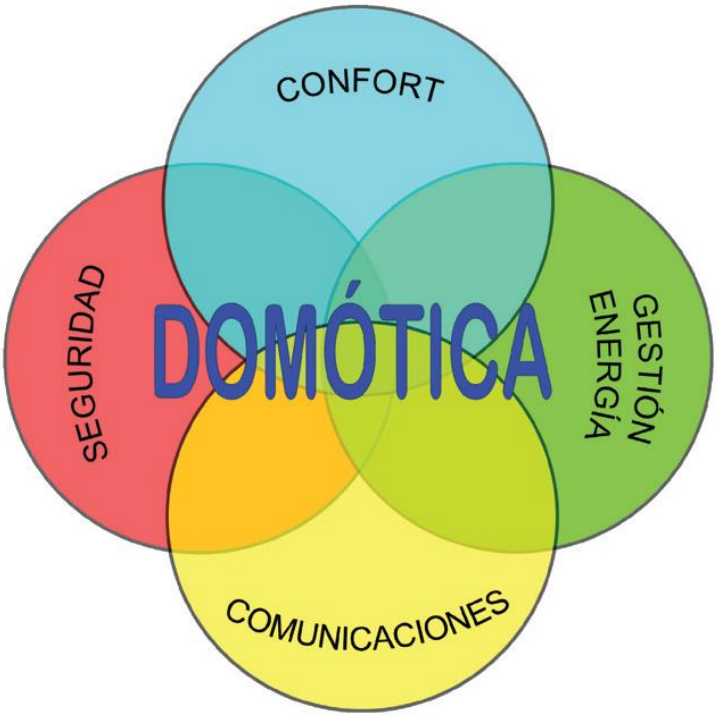
2. Context tecnològic

2.1 Tecnologia actual

En aquest apartat es tractarà d'oferir una visió global de la tecnologia més moderna i actual. Així mateix, s'oferix una xicoteta descripció del seu funcionament i de la seua repercussió a la societat.

2.1.1 Domòtica

La domòtica consisteix en l'automatització dels processos que tenen succés dins d'un habitatge, com puguen ser la gestió de l'energia, el benestar, la seguretat, la comunicació i l'accessibilitat; i que poden ser controlats tant des de dins com des de fora de l'habitatge. Aquesta disciplina permet, per exemple, que quan es faça fosc s'encenguen automàticament els llums de les habitacions on hi ha persones i es baixen les persianes.



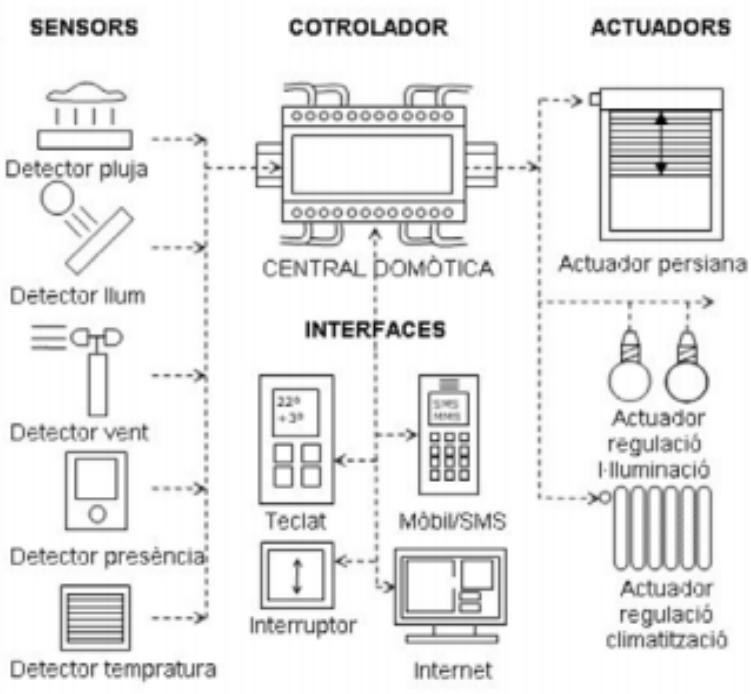
Imatge 1: Diagrama Domòtica.

Cada dia és més comuna la presencia de la domòtica i altres sistemes de control als habitatges. La domòtica pretén fer més fàcil i còmode el dia a dia de les persones que viuen a una casa, així mateix també es útil per millorar la seguretat d'edificis, ja que es poden combinar sensors i actuadors per dur a terme accions o avisos implicats en la protecció de l'edifici.

A un sistema domòtic podem trobar diversos elements que el componen:

- **Controladors**: Són els dispositius encarregats de gestionar tota la informació sobre la resta d'elements del sistema. També s'inclouen les **interfícies** per interactuar amb l'usuari. Un exemple de controlador seria el servidor central, o bé un comandament a distància en cas d'una interfície.

- **Sensors**: La funció d'aquest element és obtindre informació de l'entorn on es troba. Aquesta informació és rebuda pel controlador per dur a terme les accions necessàries. Poden ser sensors de temperatura, de lluminositat, de presència...



Imatge 2: Esquema domòtica domèstica

- **Actuadors**: Són els dispositius que reben les ordres del controlador i realitzen les accions. Aquests poden ser de 4 tipus diferents:

    1. Binari: tenen dos estats, encès/apagat, i són els més senzills que anem a trobar. Per exemple, una bombeta.

    2. Regulable: són aquells que poden ser controlats de manera regulada en qualsevol dels seus aspectes de forma percentual, es a dir, expressant en forma de percentatge la potència del dispositiu que es vol utilitzar . Per exemple, una bombeta la qual podem regular la seua intensitat al 50%, expressant-la com un percentatge.

    3. Bidireccional: tenen dues interaccions oposades i opcionalment una tercera d'estat en repòs. Per exemple, una persiana que pot pujar-se, baixar-se o quedar-se esperant a un punt entremig.

    4. De valor: aquests són similars als regulables, però amb un concepte més ampli, poden adquirir qualsevol valor numèric, depenent de la seua programació i actuaran en conseqüència, però no representen el percentatge de potència del dispositiu. Per exemple, un climatitzador al que se li designa una temperatura de 21º.

2.1.2 Dispositius mòbils

Un dispositiu mòbil és un aparell de mides reduïdes i lleuger que pot ser utilitzat mentre es transporta. La gran majoria a més tenen la capacitat de comunicar-se a través de Internet, com puguen ser un telèfon mòbil o una tableta.

2.2 Auge de les tecnologies mòbils i el control per veu

Les tecnologies mòbils són, cada vegada més, una part molt important a la vida quotidiana. Des de l'aparició dels "smartphones", aquests formen part íntegra de la rutina de qualsevol persona. És fàcil observar a la gent consultar el seu correu, comunicar-se per missatgeria, realitzar compres on-line, utilitzar-lo com agenda o com a GPS.

A aquest fenomen s'agrega també l'aparició de programes com SIRI, un dels assistents personals controlats per veu més famosos, desenvolupat per Apple. Aquesta combinació, encara que no té una utilització massa estesa, té un gran potencial per aprofitar en aquelles situacions on, per dificultat, perill o incomoditat, no resulta viable l'ús d'un entorn gràfic.

2.3 Impacte social

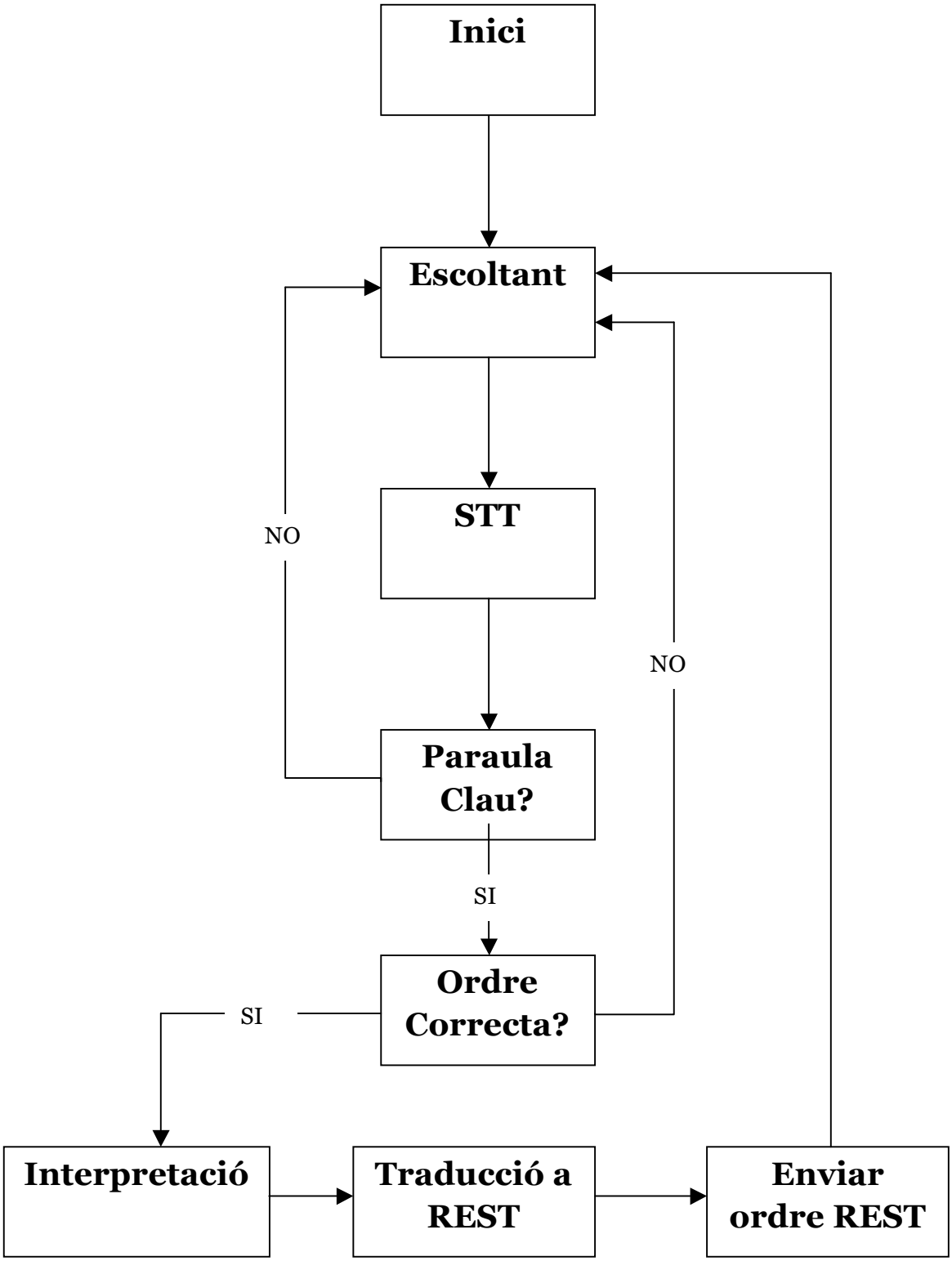
L'impacte social que podria arribar a tenir aquesta tecnologia podria ser enorme, ajudant a eliminar grans barreres si el desenvolupament es realitza seguint uns bons criteris d'accessibilitat. Aquesta tecnologia pot oferir a tota la gent, sobretot a aquells que pateixen alguna discapacitat, una considerable millora de la qualitat de vida. Per exemple, persones cegues no necessitarien recórrer la vivenda per conèixer l'estat en que es troba. Persones amb la mobilitat reduïda, en major o menor grau, serien capaços de dur a terme moltes de les accions diàries, com pogueren ser encendre o apagar llums, canviar el canal del televisor, obrir la porta, baixar les persianes a la nit, etcètera d'una forma còmoda i senzilla, sense necessitat de comandaments ni grans dispositius.

El major desavantatge que pot tenir aquest tipus de tecnologia és l'alt cost actual que implica instal·lar-la a un habitatge, fent que siguen poc assequibles per a un gran nombre de persones. No obstant això, amb el temps podria resultar una solució i una millora de vida per a moltes persones.

3. Anàlisi de requeriments

En aquest apartat es realitzarà una descripció del que ha de complir l'aplicació i en base a aquesta es desenvoluparà la implementació. L'anàlisi inicial, com a tota aplicació, ha sigut modificat amb l'avançament de la resolució del projecte per ajustar-se al que s'esperava de l'aplicació.

- L'aplicació haurà de ser capaç de reconèixer i configurar els micròfons d'entrada de veu estàndard del dispositiu.
- L'aplicació haurà de capturar les paraules clau definides a la gramàtica i convertir-les a text.
- L'aplicació ha de ser capaç de reconèixer quan l'usuari es dirigeix a ella i quan no.
- A l'entorn de control es podrà visualitzar l'estat d'escolta.
- L'aplicació capturarà les paraules clau d'una oració en llenguatge natural.
- L'aplicació capturarà les paraules clau independentment de l'estructura en que apareguen a l'ordre.
- Les paraules capturades seran mostrades a la interfície de text i guardades a un arxiu de text com a control.
- L'aplicació classificarà les paraules capturades i les interpretarà per convertir-les en les accions pertinents.
- Les accions es mostraran a la interfície de text i es guardaran a un arxiu de text, juntament amb les paraules capturades, com a control.
- L'aplicació traduirà les accions a llenguatge REST per comunicar-se amb el servidor.



Imatge 3: Diagrama d'estats de l'intèrpret.

3.1 Casos d'ús

Per realitzar el disseny dels casos d'us que es duran a terme a l'aplicació, s'ha utilitzat com a element base, una llista de les funcions de les que es disposava al servidor, la qual fou entregada amb el mateix. D'aquesta llista, han sigut seleccionats els més representatius i generals, deixant de banda alguns més complexos que hagueren rellentit el desenvolupament de l'aplicació al termini previst. Així mateix, es pot observar a la següent taula l'existència de sensors entre els dispositius, aquestes característiques també s'han obviat, degut que la seua funció es limita a comprovacions de l'estat ambiental per modificar el comportament de les ordres, per exemple no encendre la llum si la lluminositat és suficient.

Funcionalidad	ID Funcionalidad	Descripción
bistate	DF-CUINA.NEVERA.EN	Nevera
	DF-CUINA.CAFETERA.EN	Cafetera
	DF-ENT.SE.DETMOV.SENSOR	Detector de movimiento (Recibidor)
	DF-BC.SE.DETPRES.SENSOR	Detector presencia (Baño Común)
	DF-ESTACIO.METEO.PLUTJA	Sensor lluvia (Exterior)
	DF-PORTA.SENSOR.TANCADA	Sensor Puerta Ppal Cerrada
togglebistate	DF-CUINA.IL.CENTRAL	Luz central (Cocina)
	DF-CUINA.IL.AUXILIAR	Luz auxiliar (Cocina)
	DF-CUINA.IL.BANCADA	Luz encimera (Cocina)
	DF-MENJ.CL.AC	Aire Acondicionado (Comedor)
	DF-HAB1.CL.AC	Aire Acondicionado (Habitación 1)
	DF-BC.CL.CALEFACTOR	Calefactor (Baño Común)
	DF-BM.CL.CALEFACTOR	Calefactor (Baño Matrimonio)
	DF-MENJ.EN.ENDOLLS	Enchufes (Comedor)
movement	DF-MENJ.FINESTRA.PERSIANES12	Persianas 1 y 2 (Comedor)
	DF-MENJ.FINESTRA.PERSIANES34	Persianas 3 y 4 (Comedor)
	DF-MENJ.FINESTRA.PERSIANA5	Persiana 5 (Comedor)
	DF-MENJ.FINESTRA.PERSIANES	Persianas Todas (Comedor)
	DF-HAB3.FINESTRA.PERSIANA	Persiana (Habitación 3)
	DF-HAB3.FINESTRA.ESTOR	Estor (Habitación 3)
numeric	DF-ESTACIO.METEO.TEMPERATURA	Temperatura (Exterior)
	DF-ESTACIO.METEO.LUMINOSITAT	Luminosidad (Exterior)
	DF-ESTACIO.METEO.VENT	Velocidad Viento (Exterior)
	DF-BC.SE.TEMPERATURA.SENSOR	Temperatura (Baño Común)
	DF-MENJ.SE.TEMPERATURA.SENSOR	Temperatura (Comedor)

Finalment la selecció de casos d'ús i per tant, les crides a funcions que seran realitzades pel servidor de la smarthome són les que segueixen.

Per una banda tenim la selecció de les ubicacions i els dispositius.

Llocs	**Dispositius**
CUINA	**ILUMINACIÓ**(IL.CENTRAL)
MENJADOR(MENJ)	**PERSIANES**(FINESTRA.PERSIANA/ES)
CAMBRA DE BANY(BC)/(Bany Comú)	**CLIMATITZADOR**(CL.AC)
DORMITORI(HAB1)	**CALEFACTOR**(CL.CALEFACTOR)
	PORTA(PORTA.SENSOR.TANCADA)
	NEVERA(CUINA.NEVERA.EN)

Amb aquesta selecció feta, es pot ja desenvolupar la llista d'accions que deurà ser capaç d'interpretar i dur a terme l'aplicació. A la taula queden ressaltades les paraules claus que es tindran en compte més endavant per al reconeixement vocal. Cal tenir en compte que no tots els habitacles de la smarthome disposen de tots els dispositius, per exemple, la cuina no disposa de persianes o el menjador no disposa de calefactor.

Acció	**URL dispositiu***	**Ordre Vocal****	**Codi d'acció**
Encendre/apagar la llum de la cuina.	DF-CUINA.IL.CENTRAL	Turn **on/off** | **plug/unplug** the **kitchen light**	action:biaON action:biaOFF
Encendre/apagar la nevera	DF-CUINA-NEVERA.EN	Turn **on/off** | **plug/unplug** the **fridge**	action:biaON action:biaOFF
Pujar/baixar les persianes del menjador	DF-MENJ.FINESTRA.PERSIANES	**Open/close** | **up/down** the **livingroom window/blind**	action:movaOPEN action:movaCLOSE

Encendre/apagar el climatitzador del menjador	DF-MENJ.CL.AC	Turn **on/off | plug/unplug** the **livingroom climate**	action:bia:ON action:biaOFF
Encendre/apagar la llum del menjador	DF-MENJ.IL.CENTRAL	Turn **on/off | plug/unplug** the **livingroom light**	action:biaON action:biaOFF
Encendre/apagar la llum del dormitori	DF-HAB1.IL.CENTRAL	Turn **on/off | plug/unplug** the **bedroom light**	action:biaON action:biaOFF
Pujar/baixar les persianes del dormitori	DF-HAB1.FINESTRA.PERSIANA	**Open/close | up/down** the **bedroom window/blind**	action:movaOPEN action:movaCLOSE
Encendre/apagar el climatitzador del dormitori	DF-HAB1.CL.AC	Turn **on/off | plug|unplug** the **bedroom climate**	action:biaON action:biaOFF
Encendre/apagar el calefactor del dormitori	DF-HAB1.CL.CALEFACTOR	Turn **on/off | plug/unplug** the **bedroom heater**	Action:biaON action:biaOFF
Encendre/apagar la llum de la cambra de bany	DF-BC.IL.CENTRAL	Turn **on/off | plug/unplug** the **bathroom/toilet light**	Action:biaON action:biaOFF
Encendre/apagar el calefactor de la cambra de bany	DF-BC.CL.CALEFACTOR	Turn **on/off | plug/unplug** the **bathroom/toilet heater**	Action:biaON action:biaOFF

*Durant la crida al servidor tota URL anirà precedida amb la direcció del servidor, en aquest cas: "http://localhost:8182/devFunc/".

**Les ordres vocals no han de mantindre eixa estructura, poden variar lliurement la posició dels elements oracionals, per exemple: "Isis, the light in the kitchen, unplug it."

A més d'aquestes funcions hi haurà un tipus específic de funció que sols podrà ocórrer immediatament després de que una funció de moviment, en aquest cas sols les funcions de pujar o baixar les persianes, tinga lloc. Aquestes accions tenen un comportament lleugerament diferent degut a què utilitzen l'element que ha sigut cridat anteriorment per modificar el seu estat, no obstant l'execució segueix els mateixos passos que en el cas de les ordres anteriors.

Parar el moviment	Url anterior	**Stop/ok/okay**	Codi d'acció anterior
Més moviment	Url anterior	**more**	Codi d'acció anterior
Menys moviment	Url anterior	**less**	Codi d'acció contrari a l'anterior

4. Disseny

En aquest apartat es desenvoluparà el procés de disseny de l'aplicació, que enviarà al servidor la informació per executar les ordres. Es realitzarà una descripció de les funcionalitats dels mètodes implementats i de les seues propietats.

4.1 Visió general

Tal i com s'ha comentat en l'inici, el punt de partida serà una API STT ("Speech To Text") què complirà la funció de captar el so de la veu, analitzar-lo i convertir-lo en una cadena de String. Per realitzar aquesta funció, s'ha optat per un model de codi obert, alternativament als models presentats per empreses com ara Google, degut a l'avantatge que suposa poder adaptar lleugerament l'API a les necessitats del projecte, a més de les característiques i avantatges que suposa el codi obert:

- Fiabilitat: Al poder ser millorat per qualsevol usuari, els errors que puguen existir són ràpidament localitzats i solucionats.
- Estabilitat: Una vegada apareix una versió "estable oficial", és a dir, que està aprovada per l'equip de desenvolupament, sol tindre una llarga durada sense necessitat de cap tipus d'actualització, mantenint la compatibilitat.
- Cost: La major part del codi obert és també gratuït, no obstant, no tot el codi obert és distribuit gratuitament, ja que qui el distribuieix es lliure d'elegir regalar-lo o vendre'l, depenent del seu criteri
- Llibertat: L'usuari o la empresa és lliure de modificar qualsevol part del codi per adaptar-la a les seues necessitats.

Una vegada obtingut el text amb l'STT, aquest és analitzat i interpretat pel programari desenvolupat i traduït a un llenguatge REST, que serà interpretat al servidor que s'utilitzarà en aquest cas pràctic. Ja amb la interpretació es converteix l'ordre capturada en un missatge entendible per al servidor, què activarà el dispositiu corresponent.

4.2 Components del sistema

El sistema de reconeixement d'ordres de veu es compon de 5 parts fonamentals:

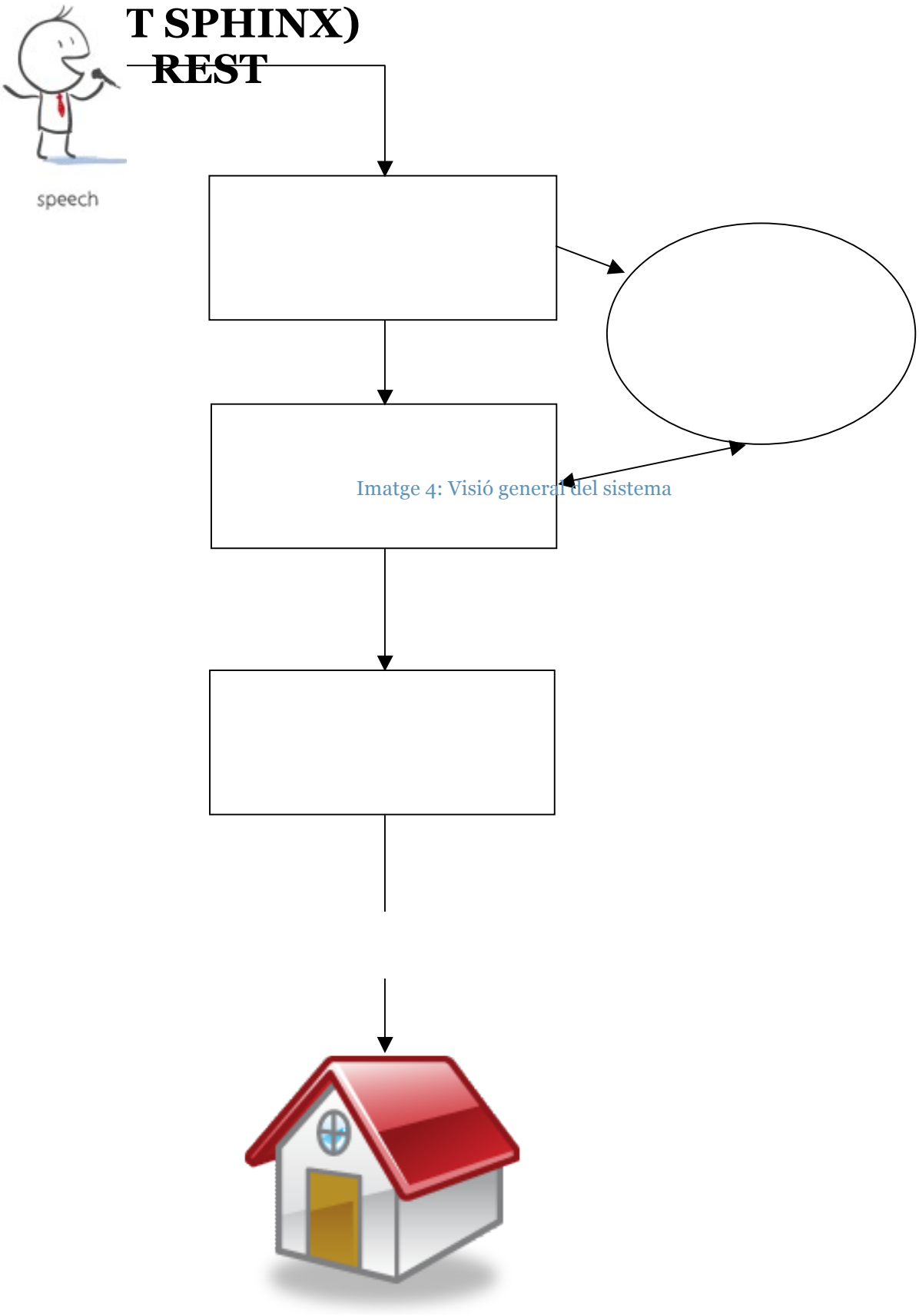
1- **Micròfon.** Es tracta del dispositiu bàsic amb el qual es treballarà i la seua funció és captar la veu de l'usuari. S'utilitzarà un model de micròfon dinàmic, el qual consta d'una xicoteta bobina mòbil, unida a un diafragma, situada en el camp magnètic d'un imant. Quan les ones sonores fan vibrar el diafragma, aquest mou la bobina creant un corrent elèctric proporcional al so. Aquest tipus de micròfons són robustos i barats, i poden ser utilitzats a una gran varietat de situacions.

2- **Speech To Text (STT).** És el programari que s'utilitzarà per gestionar l'àudio capturat. La funció d'aquest és transcriure la parla capturada a text pla. Les captures estaran limitades a un conjunt de paraules clau prèviament definides a una gramàtica. Més endavant, a l'apartat d'implementació, es desenvoluparà la informació referent a l'STT.

3- **Analitzador semàntic.** Aquest és el primer punt a desenvolupar del projecte. Una vegada capturat el text, s'analitzarà per determinar quin tipus d'ordre s'ha declarat i si és correcta o lògica per dur-la a terme. L'analitzador semàntic també es desenvoluparà més extensament al següent apartat d'implementació.

4- **Intèrpret d'accions.** Una vegada comprovada la semàntica de les ordres aquestes passaran a l'intèrpret. Ací seran novament analitzades per determinar quina acció és l'adequada per dur a terme. Una vegada assignada l'acció, aquesta serà traduïda per enviar-la al servidor. L'intèrpret d'accions es desenvoluparà, a l'igual que la resta, més extensament al següent apartat d'implementació.

5- **Servidor i Smarthome.** Aquest és l'últim component del nostre sistema. L'smarthome és un habitatge el qual està previst d'un sistema domòtic. Com s'ha explicat anteriorment de forma detallada, aquest sistema domòtic consta d'un servidor o central domòtica, que serà l'encarregat de gestionar els actuadors. Aquest servidor central rebrà les ordres emeses per l'intèrpret en llenguatge REST i accionarà els actuadors corresponents per materialitzar l'acció desitjada.



Imatge 4: Visió general del sistema

5. Implementació

En aquest apartat es nomenaran i explicaran les tecnologies utilitzades per al desenvolupament del projecte, així com la implementació del codi programat per al mateix a les funcions de l'analitzador i l'intèrpret.

5.1 Tecnologies utilitzades

A continuació es mostren les tecnologies utilitzades per desenvolupar les diferents parts del projecte.

5.1.1 L'SPHINX (Speech To Text)

La idea original era crear el programa amb un reconeixement de veu en valencià, però després de les primeres investigacions sobre el funcionament de l'SPHINX, es va desestimar. La raó fou que l'API funciona amb el suport de VoxForge per als models acústics, i el model català, que seria el necessari per poder desenvolupar el programari en llengua valenciana, no està desenvolupat degut a la falta de hores de gravació.

Imatge 5: Percentatges contribució llengua catalana a VoxForge

Al resultar impossible la realització del reconeixement en llengua valenciana, s'optà per buscar el model acústic de llengua castellana. Novament l'equip de VoxForge indica que no hi ha cap model desenvolupat per a aquesta llengua, es a dir, cap model acústic de parla castellana. No obstant, el temps de gravació és molt superior al model de llengua catalana.

****

Imatge 6: Percentatges contribució llengua castellana a VoxForge

Tot i que les hores de gravació de llengua castellana no estan completes, sí que existeix un model de veu elaborat i funcional per a aquesta llengua.

Home / Acoustic and Language Models / Spanish Voxforge

Name	Modified	Size	Downloads / Week
Parent folder			
voxforge-es-0.1.1.tar.gz	2014-05-06	3.4 MB	22
voxforge-es-0.1.tar.gz	2014-05-06	6.0 MB	4
LICENSE	2012-07-11	475 Bytes	1
Totals: 3 Items		**9.4 MB**	**27**

Imatge 7: Model de llengua espanyola de voxforge.

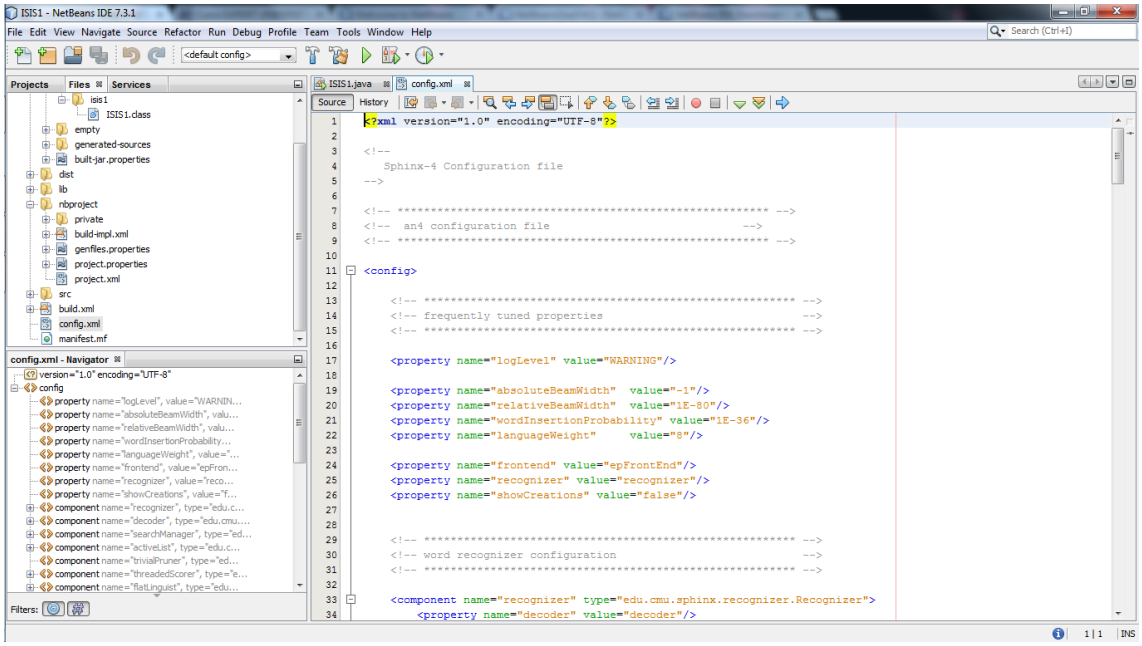
Després d'implementar el model acústic castellà al projecte de prova i realitzar algunes proves preliminars, els resultats no van ser els esperats degut a diverses causes. La més significativa era la falta d'hores de gravació, que no permet un model de veu fiable. Aquesta va acompanyada d'altres com la gran diferència d'accents que hi ha a la llengua castellana, com són l'espanyol i sud-americà, que feia cometre diversos errors en el reconeixement de veu a l'hora de captar i realitzar la conversió a text de les paraules, donant un gran nombre de falsos resultats i sobre tot, resultats nuls. Aquesta causa porta, novament, a desestimar la llengua castellana com a model de veu per al programari, ja que la intenció és realitzar una versió que siga funcional i fiable.

Finalment, després de tot els contratemps i amb els models de veu disponibles, es va optar per utilitzar com a llengua vehicular l'anglès. Aquesta elecció es deu a que, d'aquesta manera, la fiabilitat de les ordres i paraules que puguen ser interpretades és major sumat al fet què amb un nivell mitjanament bo d'anglès, és possible realitzar les proves i captures pertinents. La diferència més notable rau en la semàntica, que en l'anglés és molt més tancada i estricta que en valencià o castellà, fent més simple la seua tasca.

5.1.2 Entorn i llenguatge de programació

En primer lloc es parlarà de l'entorn de programació utilitzat per al desenvolupament del codi del programa. L'entorn seleccionat per realitzar el programa ha sigut NetBeans, amb Ubuntu com a sistema operatiu. L'elecció d'aquest entorn de programació ha sigut condicionada per dos factors, el primer la seua disposició gratuïta, ja que es tracta d'un software de codi obert, i la qual cosa implica una llibertat d'ús de les llicències necessàries per desenvolupar qualsevol tipus de projecte sense cap cost addicional.

El segon factor pel qual s'ha elegit ha sigut el coneixement previ de l'entorn, ja que s'havien realitzat treballs de programació amb anterioritat baix aquesta interfície, cosa que facilita i agilitza el desenvolupament del projecte, evitant haver d'aprendre a utilitzar una nova interfície o funcions d'un programa diferent.



Imatge 9: Entorn de programació NetBeans.

En segon lloc, es parla del llenguatge de programació utilitzat. El que s'ha triat per desenvolupar el codi ha sigut Java. L'elecció d'aquest llenguatge ha sigut influïda primerament degut a la seua capacitat portable. Java permet escriure el programari en una plataforma per més tard executar-la virtualment a altra plataforma totalment diferent. Aquesta versatilitat resulta d'una gran utilitat per a aquest projecte ja que, per exemple, la seua disposició a les diferents parts d'un habitatge, pot estar formada per diferents dispositius de control independents, com pogueren ser ordinadors de baix cost RaspBerry Pi, connectats al servidor central de processos.

L'elecció d'aquest llenguatge ha estat condicionada també a que l'API SPHINX, utilitzada per l'STT, està desenvolupada amb aquest llenguatge, a més a més, la prèvia coneixença del mateix per part del programador, ha influït també a l'elecció perquè facilita el desenvolupament de l'aplicació.

5.1.3 Comunicació amb el servidor (REST)

- **Visió general.**

  El llenguatge REST (Representation State Transfer) deriva d'una tesi doctoral de Roy Fielding. Aquest llenguatge és un tipus de servei web. Es referix a un tipus d'arquitectura client/servidor sense estats, on els serveis web es veuen com a recursos que poden ser identificats per les seues URIs (Uniform Resource Identifiers). Els serveis estan limitats als mètodes estàndard GET, POST, PUT y DELETE HTTP. En la seua forma bàsica, l'API REST es una URI que es capaç de realitzar una única operació. Les APIs REST són independents als llenguatges de programació i són accessibles des de qualsevol plataforma que tinga una llibreria de client HTTP, incloent Java

- **API REST de la Smarthome**

  El servidor que es proporciona a aquesta implementació únicament conté una capa de serveis REST molt senzilla per interactuar sols amb les funcions DeviceFunctionalities. Aquestes realment són les que proporcionen les funcionalitats dins d'una vivenda, per exemple sobre les llums, les persianes, climatitzadors, endolls, electrodomèstics, sensors, etc. Pel que, encara que solament s'oferisquen aquests tipus de recursos, son més que suficients, ja que són els més versàtils.

- **Operacions permeses**

  Les úniques operacions permeses a la versió que s'utilitzarà a aquest projecte son:

  - GET: per obtindre informació del recurs.
  - PUT: per sol·licitar l'execució d'alguna operació del recurs.

  Les operacions estan implementades de forma que tant les respostes per part del servidor com les Payloads que són enviades, estan en format JSON. El JSON (JavaScript Object notation) és un estàndard basat en text dissenyat per a intercanvi de dades llegible per humans.

  Ja que la operació GET no envia Payload, per invocar-la no és necessari indicar una URI de destí.

  En canvi, per invocar una petició PUT, és necessari passar un Payload en el següent format JSON.

  { action : codi-acció, value : valor }

- **Tipus de recursos i codis d'acció**

  Els codis d'acció que s'utilitzaran depenen del tipus de recurs que s'utilitze, així com el valor, que serà opcional en funció de l'acció indicada. Hi ha diferents tipus de DeviceFunctionality. Es pot observar a la propietat devfunc-type, al realitzar un GET, o com a resultat d'un PUT, sobre una DeviceFunctionality.

  A continuació s'enumeren els tipus més representatius:

  - **Bistate:** Representa un dispositiu que pot estar en dos estats, apagat / encès, connectat / desconnectat, etc. Per exemple es pot aplicar als estats de les llums, les portes o les finestres.

  - **ToggleBistate:** Representen un tipus de dispositiu molt semblant al bistate però permetent l'acció "toggle", es a dir, permeten commutar l'estat en què es troba el dispositiu, encendre si està apagat o apagar-lo si es hi és encès. Poden ser ToggleBistate tots aquells dispositius bistate als quals se'ls dona l'opció de commutar.

  - **Dimmer:** Es tracta d'un tipus de bistate regulable, això vol dir que a més de tenir les posicions encès o apagat, poden situar-se a un percentatge concret entre ambdós. Un exemple seria una llum gradual o regulable.

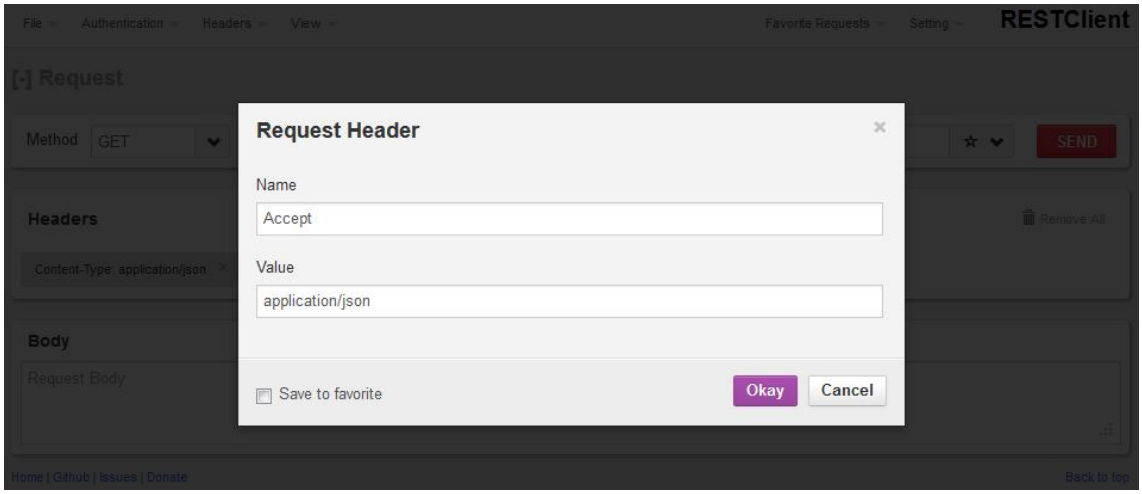
  - **Numeric Value:** Representen tots aquells dispositius que tenen relació amb valors numèrics, es a dir, que se'ls pot establir o poden indicar un valor. Per exemple, són numeric value tots aquells sensors que indiquen numèricament temperatura, lluminositat, velocitat del vent... o també un climatitzador on se li indica la temperatura a aconseguir.

  - **Pulse:** Representen tots aquells dispositius que poden ser polsats. Aquests dispositius manquen d'estat, o el seu estat es troba en inactiu fins que són polsats. Solen estar associats a accionament de dispositius, com ara interruptors o polsadors, sensors i esdeveniments puntuals, el fet que comence a ploure o es faça de nit.

- **Presa de contacte amb l'API REST**

  Per realitzar la primera presa de contacte amb l'API, ha sigut utilitzada una interfície que treballa sobre un navegador web, concretament una extensió de firefox anomenada RESTClient. Aquesta va ser aconsellada i facilitada pel tutor d'aquest projecte.

  El primer pas a realitzar per executar les primeres proves sobre la interfície és configurar les peticions. Açò ho férem afegint les capçaleres o "headers" corresponents, que en aquest cas necessitaven acceptar el format JSON.



Imatge 10: Capçaleres RESTClient.

Per realitzar les proves pertinents, juntament amb el sistema de RESTClient, es va utilitzar un servidor basat en un projecte real d'una smarthome, facilitat també pel tutor, i adaptat per al seu ús sense necessitat del sistema real de dispositius i sensors. Per començar, tan sols va ser necessari executar el programa i donar l'ordre d'activació de tots els dispositius perquè estigueren disponibles per començar les primeres proves.

5.1.4 Client (REST)

El client és, en essència, la part del programa que s'encarrega de la comunicació amb el servidor de la smarthome. Ha sigut necessari importar dues llibreries externes al programa per desenvolupar aquesta part del codi. Les llibreries utilitzades han sigut **Restlet**, per realitzar les comunicacions pertinents amb el servidor, i **Jackson**, per interpretar i deserialitzar la informació rebuda.

El client estarà implementat al propi codi del programa amb dues funcions bàsiques:

- GET: encarregat d'enviar peticions de dades, és a dir, la funció GET realitza una petició al servidor sobre un dispositiu i el servidor li envia la informació.

- PUT: encarregat d'enviar les ordres, és a dir, la funció PUT envia una ordre al servidor, conjuntament amb la direcció d'un dels dispositius, perquè siga executada.

Per desenvolupar el client, cal tindre en compte dues coses, en primer lloc que el servidor treballa amb JSON, que com ja s'ha comentat és un estàndard d'intercanvi de dades. I en segon lloc l'estructura de les dades que es troben al servidor. En el nostre cas l'estructura que trobem és la següent, aquest seria el resultat per a la petició GET de "DF-BC.IL.CENTRAL":


```
"tags": [
    {
      "id": "PerPresencia",
      "link": "/tag/PerPresencia"
    },
    {
      "id": "AutoApagar",
      "link": "/tag/AutoApagar"
    }
  ],
  "classifiers": [
    {
      "id": "ED:IL",
      "link": "/classifier/ED:IL"
    }
  ],
  "current-state": "bisON",
  "link": "/devFunc/DF-BC.IL.CENTRAL",
  "last-action": "biaON",
  "isEnabled": true,
  "id": "DF-BC.IL.CENTRAL",
  "isQuiescent": false,
```

```
  "devfunc-subtype": "ON_OFF",
  "locations": [
    {
      "id": "LOC-BC",
      "link": "/location/LOC-BC"
    }
  ],
  "name": "Llum Bany Comú Central",
  "device": {
    "id": "DEV-BC.IL",
    "link": "/device/DEV-BC.IL",
    "name": "Llum Bany Comu"
  },
  "previous-state": "UNKNOWN",
  "isStarted": true,
  "devfunc-type": "togglebistate"
```


No obstant aquesta vista no és ni més ni menys la rebuda pel client, sinó que és una vista processada i reestructurada per fer-la més amigable. La informació rebuda pel client, en format JSON, en raw és idéntica *a la mostrada* però sense salts de línia, és a dir, tota la informació es troba a una mateixa línia.


```
{"tags":[{"id":"PerPresencia","link":"/tag/PerPresencia"},{"id":
"AutoApagar","link":"/tag/AutoApagar"}],"classifiers":[{"id":"ED
:IL","link":"/classifier/ED:IL"}],"current-
state":"bisON","link":"/devFunc/DF-BC.IL.CENTRAL","last-
action":"biaON","isEnabled":true,"id":"DF-
BC.IL.CENTRAL","isQuiescent":false,"devfunc-
subtype":"ON_OFF","locations":[{"id":"LOC-
BC","link":"/location/LOC-BC"}],"name":"Llum Bany Comú
Central","device":{"id":"DEV-BC.IL","link":"/device/DEV-
BC.IL","name":"Llum Bany Comu"},"previous-
state":"UNKNOWN","isStarted":true,"devfunc-
type":"togglebistate"}
```


A continuació s'explica de forma més detallada el funcionament d'aquestes dues funcions, així com l'ús que se'ls ha donat conjuntament amb la resta del codi. Cal remarcar que a la resposta es poden observar variables amb guions unint-les, açò en la declaració de variables de java és impossible així que ha sigut necessari modificar la línia rebuda per substituir aquest elements, per exemple "**previous-state**" queda modificat a "**previousstate**" simplement amb la eliminació del "-" amb la funció "**replace("-","");**".

5.2 Desenvolupament del programari

En aquest apartat es desenvoluparà tota la informació referent a la implementació i el procés de desenvolupament que s'ha dut a terme amb l'analitzador semàntic i l'intèrpret d'ordres.

5.2.1 Analitzador

Com ja s'ha comentat a l'apartat anterior, la funció de l'analitzador semàntic és assegurar-se que les ordres donades tenen sentit i són lògiques. És a dir, s'encarrega de detectar ordres il·lògiques com ara encendre una porta o obrir el climatitzador.

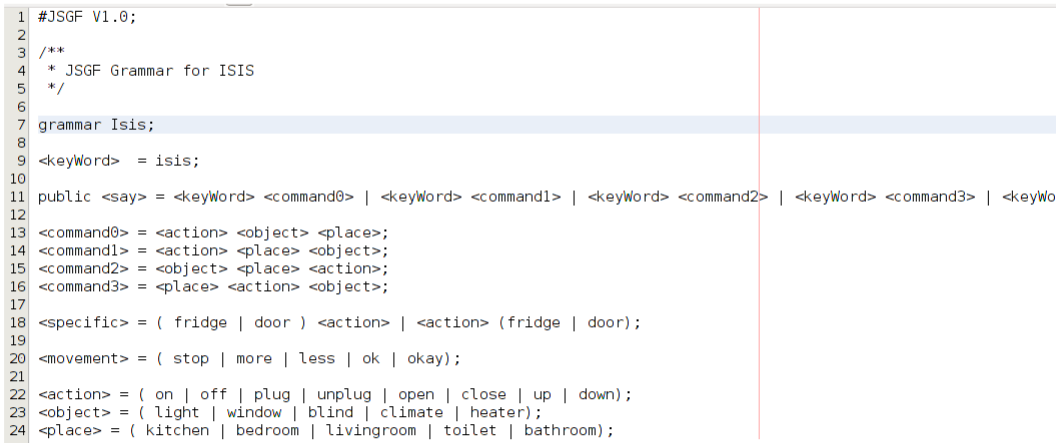
També es comentà durant el disseny la possibilitat de que el programa interpretara expressions com ara "tinc fred", "tinc calor", "vaig a dormir", etc. Però aquest apartat, degut a la seua complexitat per l'àmplia varietat d'expressions existents, a la qual cosa se li afegeix la dificultat de ser en anglés, ha sigut desestimada. Per poder realitzar-ho, la seua implementació hauria de ser semblant a la realitzada per comprovar la coherència de la sintaxi, però amb una gran quantitat de variables extra.

Per començar a implementar l'analitzador, és necessari definir correctament la gramàtica abans. Com s'ha comentat anteriorment també, la gramàtica utilitzada és una gramàtica tancada, és a dir, sols reconeixerà certes paraules clau, mentre obvia les altres. No obstant això, el programa continuarà reconeguent-les encara que aquestes es troben a una oració completament estructurada.

És important afegir que el programa sols enregistrarà l'ordre si és cridat pel seu nom. En aquest cas se li ha donat el nom d'ISIS.

Recordar també que la llengua amb la qual s'ha implementat finalment el programari és la llengua anglesa, per la qual cosa els exemples estaran en aquesta llengua.

A continuació, s'explicarà de manera detallada la gramàtica implementada, com i per què ha resultat d'aquesta manera, així com les raons que han portat a realitzar-la així.



```
#JSGF V1.0;

/**
 * JSGF Grammar for ISIS
 */

grammar Isis;

<keyWord>  = isis;

public <say> = <keyWord> <command0> | <keyWord> <command1> | <keyWord> <command2> | <keyWord> <command3> | <keyWo

<command0> = <action> <object> <place>;
<command1> = <action> <place> <object>;
<command2> = <object> <place> <action>;
<command3> = <place> <action> <object>;

<specific> = ( fridge | door ) <action> | <action> (fridge | door);

<movement> = ( stop | more | less | ok | okay);

<action> = ( on | off | plug | unplug | open | close | up | down);
<object> = ( light | window | blind | climate | heater);
<place> = ( kitchen | bedroom | livingroom | toilet | bathroom);
```

Imatge 11: Gramàtica.

Primerament apareix l'estructura <KeyWord>, aquesta és la paraula definida que s'utilitzarà com a comandament de crida al programa. Com ja s'ha comentat la paraula elegida ha sigut ISIS, per tant les ordres donades tindran la forma "Isis, do something", ja que és necessari incloure la paraula clau per a que el programari reconega l'ordre.

Tal i com es pot observar a la imatge 11, la gramàtica està formada per diversos tipus de comandaments. Hi ha 6 tipus basics de comandaments, quatre d'ells són combinacions de les habitacions <place>, accions <action> i dispositius <object>. Aquestes combinacions són necessàries per poder realitzar les ordres en qualsevol ordre mínimament lògic. En aquest cas, com que l'idioma utilitzat és l'anglès, té menys variacions gramaticals al ser una llengua més senzilla en aquest sentit. Els altres dos tipus de comandaments són per a accions més especifiques com ara les referents a la porta principal o la nevera, que únicament es troben a un lloc concret o com totes les referents a un moviment realitzat amb anterioritat, per exemple obrir una persiana, es a dir, obri més la persiana o para d'obrir la persiana.

Donant per conclosa la gramàtica, i una vegada capturat el text mitjançant el sistema STT cal, com a primer pas, separar les ordres de la cadena de caràcters obtinguda amb la funció nativa de les cadenes de caràcters String.split(). Obtinguda la col·lecció de paraules, es crea, abans de continuar amb l'anàlisi, un nou array, que és una nova col·lecció d'elements buits.

Seguidament cal emplenar el nou array amb la classificació dels elements obtinguts. Cadascuna de les paraules claus té assignada un tipus de ordre, ja siga on/off, moviment, moure o altra molt més específica. Per emplenar el nou array, cada paraula és processada a traves de la funció switch, que comprova la paraula donada i assigna les funcions del cas coincident. Per exemple, en un cas simplificat on paraula = on:

```
switch(paraula){
        case "on":{
                    Tipo ="on/off";
                    break; }
```

Aquest mètode per a l'assignació podria haver-se dut a terme de diferents maneres, una d'elles utilitzant la funció Map(k,v) que actua com a diccionari, però degut al reduït nombre de variables en aquest cas pràctic en concret, es va optar per la utilització de swith. El resultat de la col·lecció de paraules quedaria de la següent manera: [Tipo1 | Tipo2 | Tipo3 | Tipo4]. El Tipo1 sempre correspondria amb el nom Isis i la resta com a Tipus del dispositiu, Tipus de l'acció o Tipus d'habitació, assignada sempre com a default.

Una vegada assignats els tipus, és hora de comprovar si aquests es combinen correctament. Per realitzar aquesta acció, cal executar la funció nomenada "comprovaSintaxi" que s'ha creat. Aquesta funció retorna un tipus boolean, és a dir, un element verdader o fals, sobre la col·lecció de tipus que se li ha passat a la funció.

El procediment seguit per realitzar les comprovacions sintàctiques ha sigut similar a aquell utilitzat a la funció anterior, és a dir, el mètode de selecció ha sigut novament un switch. Aquesta vegada el sistema comprova si l'element de la col·lecció és compatible sintàcticament amb els altres existents dins la col·lecció. En cas de trobar durant el procés alguna combinació vàlida, la funció acabarà, retornant així la variable com a verdadera. En cas contrari, és a dir, que les combinacions no resulten en cap moment correctes, el procés acabarà i retornarà un resultat fals, fent que la funció principal detinga l'execució de l'ordre, donant pas a un missatge d'error sintàctica. Un exemple del procés de selecció seria aquest, siguent el fragment de comprovació del tipo=on/off;

```
switch (tipo[1]){
        case "on/off":switch (tipo[2]){
                case "on/off": return true;
                case "climate": return true;
                case "conmuta": return true;
                case "fridge": return true;
                default: switch (tipo[3]){
                        case "on/off": return true;
                        case "climate": return true;
                        case "conmuta": return true;
                        case "fridge": return true;
                        default:
                }
        }
```

Una vegada superat el filtre sintàctic sols queda una acció més a fer per l'analitzador. Aquesta és designar si el tipus d'acció que s'està duent a terme és de tipus moviment per marcar la variable booleana, la qual és anomenada 'movent', amb l'estat vertader, si l'última acció ha sigut de tipus moviment. En cas contrari, si ens trobem amb l'estat fals, on l'última acció no ha sigut de moviment, es reconeixerà que l'ordre donada no ha sigut correcta.

Aquest procediment és requerit per a la pròpia anàlisi sintàctica de la propera acció. Açò és degut què les accions que anomenades al punt 4.1 Casos d'ús modifiquen l'estat d'un dispositiu en moviment, com ara una persiana que esta obrint-se, es fan servir d'aquesta variable per validar o no la sintaxi. A continuació es mostra amb un fragment de codi:

```
case "stop":{
                    Tipo[i]="moure";
                    if (movent) return true;
                    else return false;
        }
```

5.2.2 Intèrpret

Després de superar el filtre de l'analitzador, açò és, després d'assegurar-se que l'ordre donada és sintàcticament correcta, es donarà pas a l'intèrpret. La funció d'aquest és, com el seu nom indica, interpretar les paraules clau rebudes i traduir-les perquè puguen ser enviades al servidor mitjançant la funció Put. Aquesta ja s'ha comentat amb anterioritat al punt 6.1.4, i serà explicada totalment al punt 6.2.4. El funcionament de l'algoritme radica en l'assignació de la paraula corresponent a la seua variable, que pot ser:

```
String lloc="";
String accio="";
String dispisitiu="";
```

Per realitzar la traducció s'ha utilitzat novament un switch. L'elecció d'aquest mètode front a un diccionari, que pot semblar més útil en aquest cas, ha sigut que amb el diccionari era inevitable realitzar una assignació als tipus de variables, fent que fóra necessari realitzar de nou una comprovació extra. Açò portava a processos innecessaris, per la qual cosa s'ha optat pel switch. Amb aquest, aprofitant el fet que es treballa amb un número reduït de variables, s'han realitzat les dues accions, la comprovació i la transcripció, simultàniament, assignar directament la transcripció de la paraula clau a la seua variable. A continuació es mostra el fragment del codi:

```
for(int i=0; i<resultatsDividits.length; i++){
            //El switch treballa com un classificador i un
diccionari al mateix temps,  per una banda ordena en lloc,
dispositiu o acció la paraula reconeguda i  la tradueix al
seu posterior ID al servidor.

            switch(resultatsDividits[i]){
            //Llocs/habitacions de l'habitatge
                    case "kitchen" :{
                    lloc="CUINA";
                    break;
                }
                    case "livingroom" :{
```

```
            lloc="MENJ";
            break;
        }
            case "bathroom" :{
            lloc="BC";
            break;
        }
            case "bedroom" :{
            lloc="HAB1";
            break;
        }
            case "toilet":{
            lloc="BC";
            break;
        }
        //Dispositius de l'habitatge.
            case "climate" :{
            dispositiu="CL.AC";
            break;
            }
            case "heater": {
            dispositiu ="CL.CALEFACTOR";
            break;
            }
            case "light":{
            dispositiu="IL.CENTRAL";
            break;
            }
            case "door":{
            dispositiu="PORTA.SENSOR.TANCADA";
            break;
            }
            case "fridge":{
            dispositiu="CUINA.NEVERA.EN";
            break;
            }
            case "window":{
            dispositiu ="FINESTRA.PERSIANES";
            break;
            }
            case "blind":{
            dispositiu = "FINESTRA.PERSIANES";
            break;
            }
        //Accions dels dispositius.
            case "on":{
            accio="biaON";
            break;
            }
            case "off":{
            accio="biaOFF";
            break;
```

```
                    }
                    case "plug":{
                    accio="biaON";
                    break;
                    }
                    case "unplug":{
                    accio="biaOFF";
                    break;
                    }
                    case "open":{
                    accio="movaOPEN";
                    break;
                    }
                    case "close":{
                    accio="movaCLOSE";
                    break;
                    }
                    case "up":{
                    accio="movaOPEN";
                    break;
                    }
                    case "down":{
                    accio="movaCLOSE";
                    break;
                    }
                    //Accions de moviment.
                    case "stop":{
                    accio="stop";
                    break;
                    }
                    case "ok":{
                    accio="stop";
                    break;
                    }
                    case "okay":{
                    accio="stop";
                    break;
                    }
                    case "more":{
                    accio="more";
                    break;
                    }
                    case "less":{
                    accio="less";
                    break;
                    }
                    default:
        }
    }
```

Després d'executar el fragment anterior sols queda un pas, crear la URL per poder fer la crida adequada a la funció Put. Tot açò, tenint en compte que

totes les variables hauran sigut emplenades, ja es treballa amb una gramàtica tancada i es comprova prèviament amb l'analitzador l'absència d'errors de sintaxi (entre els quals s'inclou la falta d'una part de l'rodre).

La crida a la funció Put requereix d'una String amb la url i d'una altra amb l'acció a executar per part del dispositiu seleccionat, seguint aquestes l'estructura que s'ha mostrat als casos d'us a l'apartat 4.1.

Es pot observar al codi que les anomenades accions de moviment no tenen cap tipus de transcripció. Açò és degut a què aquestes són ordres específiques que treballen sobre l'acció que ha sigut executada amb anterioritat a elles, és per això que se'ls assigna més endavant unes variables generals. Hi haurà les LastAction com a acció a realitzar (exceptuant el cas de l'ordre "less" que serà l'oposada) i LastDevice com a url. Llevat d'aquesta singularitat de casos i els de la cuina i la porta (que rebran també un tractament especial) la creació de la url segueix un mètode genèric i senzill:

```
url = "DF-"+lloc+"."+dispositiu;
```

Una vegada aconseguida la url, sols resta realitzar la crida a Put(url, accio). Amb aquesta crida i l'assignació de les noves variables LastAction i LastDevice, conclou l'execució de la funció intèrpret.

5.2.3 GET

Com ja s'ha comentat anteriorment, la funció que du a terme GET és la de demanar i rebre les dades del servidor, així com la de processar la informació rebuda i classificar-la.

Per tal de rebre i classificar la informació rebuda cal, primerament, definir una classe 'objecte', que en aquest cas anomenarem Dispositiu. La classe dispositiu serà la que organitze i classifique tota la informació rebuda en una estructura ordenada. La classe Dispositiu té definides totes i cadascuna de les variables que podem trobar a la línia rebuda, quedant la classe amb la següent definició :

```
public class Dispositiu {
     public ArrayList <Tags> tags;
     public ArrayList <Classifiers> classifiers;
     public String currentstate;
     public String link;
     public String lastaction;
     public boolean isEnabled;
```

```
        public String id;
        public boolean isQuiescent;
        public String devfuncsubtype;
        public ArrayList<Locations> locations;
        public String name;
        @JsonProperty ("device") private Device device;
        public String previousstate;
        public String isStarted;
        public String devfunctype;
}
```

Una vegada definit l'objecte Dispositiu com a contenidor, ha de ser emplenat. Per emplenar-lo ha de realitzar-se la crida al servidor amb l'element del qual es desitja la informació. Açò ho fem creant un client "**ClientResource**" apuntant a l'element desitjat, cridant a la funció interna de "**ClientResource.get()**" i transformant-la a text amb la funció "**getText()**", obtenint així la línia de informació comentada a l'apartat anterior.

Arribats a aquest punt, sols queda deserialitzar el contingut de la informació rebuda. Per realitzar aquesta funció s'utilitzarà un mapa d'objecte "**ObjectMapper**" de la llibreria restlet, cridant a la funció interna readValue(String,Object) i donant-li com a String la línia obtinguda per part del servidor i com a Object la classe dispositiu, Dispositiu.class.

El resultat total de la funció GET queda, finalment, així:

```
@Get("json")
public static Dispositiu Get(String disp)throws
IOException{
String n;
Dispositiu d = new Dispositiu();
n=new
ClientResource("http://localhost:8182/devFunc/"+disp).get()
.getText();
n = n.replace("-","");
ObjectMapper om = new ObjectMapper();
om.configure(DeserializationConfig.Feature.FAIL_ON_UNKNOWN_
PROPERTIES, false);
d = om.readValue(n, Dispositiu.class);
return d;
}
```

Com a últim punt, comentar que l'ús real d'aquesta funció no s'utilitza al programari final, sinó que ha sigut únicament utilitzat per realitzar comprovacions. No obstant, es tracta d'una funció vital en cas d'utilitzar qualsevol tipus de sensor o dispositiu del qual siga necessari extraure informació.

5.2.4 PUT

La funció que realitza PUT és, com s'ha comentat al començament de l'apartat, la d'enviar les ordres al servidor per tal que les porte a terme el dispositiu indicat.

En aquesta funció sols requerim la creació de l'objecte JSON a enviar i la corresponent connexió al servidor, ja que l'acció i el dispositiu al qual va dirigit, són donades en la crida: “Put(String disp, String action)”.

1. El primer pas a realitzar és crear l'objecte JSON:
   JSONObject object = new JSONObject();

2. En segon lloc cal afegir l'acció desitjada a l'objecte creat:
   object.put("action", action);

3. Una vegada assignada l'acció a l'objecte s’ha de crear la sol·licitud al servidor, amb el tipus de petició i el destinatari:
   Request request = new Request(Method.PUT, new Reference("http://localhost:8182/devFunc/"+disp));

4. En quart lloc s'assigna l'objecte creat anteriorment a la sol·licitud:
   request.setEntity(new JsonRepresentation(object.toString()));

5. Es crea el nou client amb el protocol dessitjat, en aquest cas HTTP:
   Client cliente = new Client(Protocol.HTTP);

6. I finalment, es gestiona la sol·licitud amb la funció interna
   Client.handle(Request);

El resultat total de la funció PUT queda, finalment, així:

```
@Put("json")
public static void Put(String disp, String action)throws
IOException{
    System.out.println("url"+disp+"accio"+action);
    JSONObject object = new JSONObject();
    try{
    object.put("action", action);
    }catch(Exception e){}
    Request request = new Request(Method.PUT, new
Reference("http://localhost:8182/devFunc/"+disp));
    request.setEntity(new
JsonRepresentation(object.toString()));
    Client cliente = new Client(Protocol.HTTP);
    cliente.handle(request);
    }
```

6. Conclusions

6.1 Fins on hem arribat

Una vegada finalitzat el projecte, és l'hora de veure què s'ha aconseguit. El programa és capaç de reconèixer les paraules clau disposades a la gramàtica, independentment de l'oració en la qual s'expressen i de l'ordre d'aquesta.

La llengua utilitzada per interactuar amb el programa és l'anglès.

I finalment es comunica de forma correcta amb el servidor, el qual executa les accions pertinents de forma, també, correcta.

6.2 Perspectiva de la idea inicial i el final assolit

L'objectiu principal d'aquest projecte era crear una interfície vocal mitjançant SPHINX, senzilla i intuïtiva, que poguera ser utilitzada amb un llenguatge humà. L'objectiu secundari era realitzar-lo amb programari de codi lliure, és a dir, sense cap tipus de llicencies. A aquest també se li sumava el fet de realitzar el programa en llengua valenciana.

Si s'observen els resultats obtinguts amb el programa, la major part dels requisits imposats inicialment han sigut complits. En primer lloc, la interfície és capaç de reconèixer les paraules clau independentment de l'ordre amb què són situades en una oració estructurada en llenguatge humà. A més a més, tot el programari utilitzat pel seu desenvolupament és lliure, com ara l'SPHINX o les llibreries Restlet i Jackson utilitzades per comunicar-se amb el servidor. L'única part que no ha pogut acomplir-se ha sigut la de la llengua a emprar, ja que després dels problemes trobats amb els models acústics en valencià, es va optar per utilitzar l'anglés.

Per tant es pot dir que, quasi en la seua totalitat, el projecte ha complit els objectius proposats. L'únic que ha quedat sense ser cobert ha sigut la part idiomàtica, que s'espera que amb el temps o utilitzant una altra tecnologia per realitzar l'STT (diferent a SPHINX) poguera ser fàcilment salvable.

6.3 Què queda per fer

Després d'analitzar la perspectiva inicial i final del projecte, es pot observar que aquest s'ha assolit quasi amb la seua totalitat.

No obstant, al tractar-se d'una aplicació pràctica, i degut al temps de què es disposava, ha hagut certs aspectes que han sigut deixats de banda i han quedat reclosos com a propostes que han quedat per fer.

Una de les més destacades era la idea inicial què el programa fóra capaç d'interpretar una sèrie d'expressions com ara 'tinc fred', 'tinc calor', 'tinc son', 'no puc vore', etcètera. i amb açò realitzara les funcions corresponents. Però per dur a terme aquesta implementació, seria necessari augmentar la complexitat de la gramàtica i tenir en compte un o més tipus de variables i accions, al ser aquestes més difícils d'interpretar.

També haguera sigut interessant utilitzar diferents sensors per captar les variacions del medi extern (canvis de temperatura, lluminositat, etc.), que enviaren una senyal i el programa fóra capaç de respondre modificant els dispositius de la smarthome per adaptar-se a aquestes variacions. Per tant ha quedat també pendent l'obtenció de la informació brindada per aquests dispositius, la qual sols requereix una crida amb la funció Get, que sí que està implementada, i fer ús d'eixa informació.

A més d'aquests elements, també ha quedat per implementar certs dispositius, que degut al temps d'entrega foren omesos, com persianes independents i habitacions secundaries.

Finalment, el més important si cap que ha quedat per fer, ha sigut la disposició de poder donar les ordres en llengua valenciana. Per aconseguir-ho, caldria esperar què VoxForge desenvolupara els models acústics pertinents, o pel contrari, renunciar a l'ús de Sphinx i utilitzar alguna altra alternativa.

6.4 Propostes de millora

La visió principal d'aquest projecte era realitzar el reconeixement de veu amb un programari de codi lliure per comprovar la seua viabilitat. És per això que moltes característiques de l'entorn s'han deixat de banda.

Per millorar el programa seria necessari implementar al sistema els sensors i altres dispositius o ordres molt precises que s'han deixat per implementar.

A més la llengua no ha sigut la desitjada, una millora seria poder parlar-li en valencià, o inclòs poder elegir la llengua en què vols parlar-li. Per fer açò, primerament caldria esperar que Voxforge disposara dels models acústics pertinents, o en el seu defecte buscar un reconeixedor de veu alternatiu que acomplira el perfil.

Finalment, cal comentar que el projecte s'ha dut a terme amb un servidor i un sistema donats, i per tant s'ha fet a mida. Un important pas seria realitzar-lo de manera que pogués adaptar-se a qualsevol model i inclòs pogués modificar-se per a afegir o retirar dispositius.

6.5 Valoració

Durant el període de desenvolupament d'aquest projecte ha sigut necessari investigar i aprendre noves competències.

Una de les parts més interessants del projecte fou investigar i aprendre el funcionament del reconeixedor de veu SPHINX juntament amb VoxForge. El món del reconeixement vocal no havia despertat en mi més que simple curiositat, però el fet de poder convertir-lo en un projecte real ha sigut una oportunitat única per certament aprendre el seu funcionament i el seu potencial.

Però no sols això ha sigut l'únic nou a investigar i aprendre. La comunicació via JSON tampoc l'havia treballat, i per tant em va fer falta aprendre a utilitzar-la. Aquest pas fou molt més difícil, frustrant i finalment satisfactori que el primer, ja que durant la recerca trobí una gran quantitat de processadors diferents amb els quals treballar. Després d'un temps amb proves i errors amb diversos processadors, amb l'ajuda inestimable del tutor i d'un amic, fiu capaç de realitzar correctament les connexions amb el servidor i accedir al programa des del codi.

Més en general, i ja per concloure, dir que ha sigut un projecte interessant i satisfactori, amb el qual m'agradaria continuar treballant en un futur.

7. Bibliografia

La informació requerida per a dur a terme aquest projecte ha sigut, en la seua major part, obtinguda d'Internet. A continuació s'adjunta per ampliar tota la informació que puga ser necessària per a aquest PFC. Per desenvolupar aquesta bibliografia s'ha seguit l'estàndard ISO690-2.

- Benefits of Using Open Source Software [en línia] Disponible a Internet: "http://open-source.gbdirect.co.uk/migration/benefit.html".

- CMUSphinx Wiki [en línia]. Start User Documentation última actualització 2014/06/12 [ref. de 2014/02/10]. Disponible a internet: " http://cmusphinx.sourceforge.net/wiki/".

- CMU Sphinx [en línia]. CMU Sphinx: Joan Quintana, novembre 2011, última actualització 7 de novembre de 2011 [ref. de 15 de febrer de 2014]. Disponible a internet:"http://wiki.joanillo.org/index.php/CMU_Sphinx".

- Java Platform, Standard Edition 7 API Specification [en línia].Última actualització 2014 [ref. 2-6/2014]. Disponible a internet:"http://docs.oracle.com/javase/7/docs/api/".

- Wikipedia [En línia]. [ref. 2-7/2014] Disponible en internet:"http://ca.wikipedia.org/wiki/Portada".

- VoxForge [en línia]. Última actualització 2014 [ref. 3-4/2014] Disponible a internet:"http://www.voxforge.org".

- Introducing JSON [en línia]. [ref. 6/2014]. Disponible a internet:" http://json.org/".

- REST Application Programming [en línia]. Última actualització 21/9/2010 [ref. 6/2014]. Disponible a internet:" http://www.ibm.com/developerworks/aix/library/au-aem_rest/".

- Learn Restlet Framework [en línia] . Última actualització 2014 [ref. 6/2014]. Disponible a internet:" http://restlet.com/learn/tutorial/2.2/".

- Jackson JSON Processor Wiki [en línia]. Última actualització 30/4/2013[ref. 6/2014]. Disponible a internet:" http://wiki.fasterxml.com/JacksonHome".

8. Annex de Codi Font

8.1 Main

```
package isis3;


import edu.cmu.sphinx.frontend.util.Microphone;
import edu.cmu.sphinx.recognizer.Recognizer;
import edu.cmu.sphinx.result.Result;
import edu.cmu.sphinx.util.props.ConfigurationManager;
import java.io.IOException;
import org.codehaus.jackson.map.DeserializationConfig;
import org.codehaus.jackson.map.ObjectMapper;
import org.restlet.resource.ClientResource;
import org.restlet.Request;//
import org.restlet.data.Method;
import org.restlet.data.Reference;
import org.restlet.resource.Get;
import org.restlet.resource.Put;
import org.json.JSONObject;//
import org.restlet.Client;//
import org.restlet.Response;//
import org.restlet.data.Protocol;//
import org.restlet.ext.json.JsonRepresentation;//
//import java.util.Scanner;//Per llegir els fitxers



public class Isis3 {

static boolean movent;//variable global per controlar si
l'última acció ha sigut de moviment o no.
static String[] resultatsDividits;
static String[] Tipos;
static String LastDevice;
static String LastAction;
     public static void main(String[] args) {
        ConfigurationManager cm;

        //Carrega la configuració del programa i els periferics.
        if (args.length > 0) {
            cm = new ConfigurationManager(args[0]);
        } else {
            cm = new
ConfigurationManager(Isis3.class.getResource("IsisConfig.xml"));
        }


        Recognizer recognizer = (Recognizer)
cm.lookup("recognizer");
        recognizer.allocate();
```

```
        //Engega el microfon o tanca el programa en cas d'error.
        Microphone microphone = (Microphone)
cm.lookup("microphone");
        if (!microphone.startRecording()) {
            System.out.println("No es possible engegar el
microfon.");
            recognizer.deallocate();
            System.exit(1);
        }


        //Comença el STT fins que el programa s'atura.
        movent = false;//s'inicialitza el controlador de accions
de moviment.
        while (true){
            //Isis esta preparat per escoltar.
            System.out.println("Parla'm ara.\n");// Pressiona
Ctrl+C per aturar.\n");


            Result result = recognizer.recognize();//Ara per ara
el reconeixement de veu es en anglés.

            if (result != null) {
                String resultText =
result.getBestFinalResultNoFiller();
                //String[] resultatsDividits= resultText.split("
");//Divideix les paraules capturades a un array.
                System.out.println("Has dit: " + resultText +
'\n');   //Comprovació del STT.

                    try{if (Analitzador (resultText,movent)){
                    Interpret(resultText);
                }else System.out.println("La sintÃxi de la
ordre no era correcta. \n Per favor, repeteix l'ordre.");
                    }catch(Exception e){System.out.println("La
sintàxi de l'ordre no era correcta. \n Per favor, repeteix
l'ordre.");}
            }
        }
    }

public static boolean Analitzador(String captura, boolean
movent){
    //analitza les paraules capturades i els asigna un tipus
d'acciÃ³
    Isis3.resultatsDividits= captura.split(" "); //divideix la
captura per paraules

    int l = resultatsDividits.length;
    //comprovació resultat capturat
    for(int x=0; x<l;x++){
    System.out.println(resultatsDividits[x]);
    }
    String[] Tipo = new String[l];
    for(int i=0;i<l;i++){
    switch(resultatsDividits[i]){
```

```
case "on":{
            Tipo[i]="on/off";
            break;
}
case "off":{
            Tipo[i]="on/off";
            break;
}
case "change":{
            Tipo[i]="conmuta";
            break;
}
case "put":{
            Tipo[i]="put";
            break;
}
case "plug":{
            Tipo[i]="on/off";
            break;
}
case "unplug":{
            Tipo[i]="on/off";
            break;
}
case "open":{
            Tipo[i]="moviment";
            break;
}
case "close":{
            Tipo[i]="moviment";
            break;
}
case "up":{
            Tipo[i]="moviment";
            break;
}
case "down":{
            Tipo[i]="moviment";
            break;
}
case "light":{
            Tipo[i]="on/off";
            break;
}
case "window":{
            Tipo[i]="moviment";
            break;
}
case "door":{
            Tipo[i]="moviment";
            break;
}
case "blind":{
            Tipo[i]="blind";
            break;
}
case "climate":{
```

```
                    Tipo[i]="climate";
                    break;
            }
            case "heater":{
                    Tipo[i]="climate";
                    break;
            }
            case "stop":{
                    Tipo[i]="moure";
                    if (movent) return true;
                    else return false;
            }
            case "more":{
                    Tipo[i]="moure";
                    if (movent) return true;
                    else return false;
            }
            case "less":{
                    Tipo[i]="moure";
                    if (movent) return true;
                    else return false;
            }
            case "ok":{
                    Tipo[i]="moure";
                    if (movent) return true;
                    else return false;
            }
            case "okay":{
                    Tipo[i]="moure";
                    if (movent) return true;
                    else return false;
            }
            case "fridge":{
                    Tipo[i]="fridge";
                    break;
            }
            default: {
                    Tipo[i] ="default";
                    break;
            }

        }
        }
        boolean result= comprovaSintaxi(Tipo);
        if (result=true)
            if(Tipo[1].equals("moviment") |
Tipo[2].equals("moviment") | Tipo[3].equals("moviment" ))
Isis3.movent=true; //marca la ultima accio com a en moviment
            else Isis3.movent=false;

        Isis3.Tipos=Tipo;
        return result;
}
public static boolean comprovaSintaxi(String [] tipo){
//comprova que els tipus de paraules siguen compatibles.

System.out.println("Tipos:"+tipo[0]+tipo[1]+tipo[2]+tipo[3]);
```

```
switch (tipo[1]){
        case "on/off":switch (tipo[2]){
                case "on/off": return true;
                case "climate": return true;
                case "conmuta": return true;
                case "fridge": return true;
                default: switch (tipo[3]){
                        case "on/off": return true;
                        case "climate": return true;
                        case "conmuta": return true;
                        case "fridge": return true;
                        default:
                }
        }
        case "clima": switch (tipo[2]){
                case "on/off": return true;
                case "numero": return true;
                default: switch (tipo[3]){
                        case "on/off" : return true;
                        case "numero": return true;
                        default:
                }
        }
        case "moviment" : switch (tipo[2]){
                case "moviment": return true;
                case "blind": return true;
                case "conmuta": return true;
                default: switch(tipo[3]){
                        case "moviment": return true;
                        case "blind": return true;
                        case "conmuta": return true;
                        default:
            }
        }
        case "conmuta": switch (tipo[2]){
            case "on/off": return true;
            case "moviment": return true;
            default: switch (tipo[3]){
                        case "on/off": return true;
                        case "moviment": return true;
                        default:
            }
        }
        case "blind": switch (tipo[2]){
            case "moviment":return true;
            default: switch (tipo[3]){
                case "moviment": return true;
                default:
            }
        }
        case "put": switch(tipo[2]){
            case "number":return true;
            case "climate":return true;
            default:switch (tipo[3]){
                case "number": return true;
                case "climate": return true;
                default:
```

```
            }
        }
        case "fridge": switch (tipo[2]){
            case "on/off": return true;
            default:switch (tipo[3]){
                case "on/off": return true;
                default:
            }
        }
        default: switch(tipo[2]){
            case "on/off":switch (tipo[3]){
                case "on/off": return true;
                case "climate": return true;
                case "conmuta": return true;
                case "fridge": return true;
                default:
            }
            case "clima": switch (tipo[3]){
                case "on/off": return true;
                case "numero": return true;
                default:
            }
            case "moviment" : switch (tipo[3]){
                case "moviment": return true;
                case "blind": return true;
                case "conmuta": return true;
                default:
            }
            case "conmuta": switch (tipo[3]){
                case "on/off": return true;
                case "moviment": return true;
                default:
            }
            case "blind": switch (tipo[3]){
                case "moviment":return true;
                default:
            }
            case "fridge": switch (tipo[3]){
                case "on/off" : return true;
                default:
            }
            default:
        }
 }
return false;
}

public static void Interpret(String captura){
        //String[] resultatsDividits= captura.split(" ");
//divideix la captura per paraules
        String lloc="";
        String accio="";
        String dispositiu="";

        for(int i=0; i<resultatsDividits.length; i++){
            //El switch treballa com un clasificador i un
diccionari al mateix temps, per una banda ordena en lloc,
```

dispositiu o acció la paraula reconeguda i la tradueix al seu posterior ID al servidor.

```
            switch(resultatsDividits[i]){
            //Llocs/habitacions de l'habitatge
                    case "kitchen" :{
                    lloc="CUINA";
                    break;
                }
                    case "livingroom" :{
                    lloc="MENJ";
                    break;
                }
                    case "bathroom" :{
                    lloc="BC";
                    break;
                }
                    case "bedroom" :{
                    lloc="HAB1";
                    break;
                }
                    case "toilet":{
                    lloc="BC";
                    break;
                }
                //Dispositius de l'habitatge. El climatitzador i la porta principal tenen un tractament diferent ja que la seua ID es diferent.
                    case "climate" :{
                    dispositiu="CL.AC";
                    break;
                    }
                    case "heater": {
                    dispositiu ="CL.CALEFACTOR";
                    break;
                    }
                    case "light":{
                    dispositiu="IL.CENTRAL";
                    break;
                    }
                    case "door":{
                    dispositiu="PORTA.SENSOR.TANCADA";
                    break;
                    }
                    case "fridge":{
                    dispositiu="CUINA.NEVERA.EN";
                    break;
                    }
                    case "window":{
                    dispositiu ="FINESTRA.PERSIANES";
                    break;
                    }
                    case "blind":{
                    dispositiu = "FINESTRA.PERSIANES";
                    break;
                    }
                    //Accions dels dispositius.
                    case "on":{
```

```
                    accio="biaON";
                    break;
                    }
                    case "off":{
                    accio="biaOFF";
                    break;
                    }
                    case "plug":{
                    accio="biaON";
                    break;
                    }
                    case "unplug":{
                    accio="biaOFF";
                    break;
                    }
                    case "open":{
                    accio="movaOPEN";
                    break;
                    }
                    case "close":{
                    accio="movaCLOSE";
                    break;
                    }
                    case "up":{
                    accio="movaOPEN";
                    break;
                    }
                    case "down":{
                    accio="movaCLOSE";
                    break;
                    }
                    //Accions de moviment.
                    case "stop":{
                    accio="stop";
                    break;
                    }
                    case "ok":{
                    accio="stop";
                    break;
                    }
                    case "okay":{
                    accio="stop";
                    break;
                    }
                    case "more":{
                    accio="more";
                    break;
                    }
                    case "less":{
                    accio="less";
                    break;
                    }
                    default:
                }
            }
            // Si l'accio es una de les accions de moviment utilitza
la ultima acció realitzada per modificar-la.
```

```
        String url;
                switch(accio){
            case "stop":{
                accio="movaSTOP";
                url=Isis3.LastDevice;
            }
            case "more":{
                accio=Isis3.LastAction;
                url=Isis3.LastDevice;
            }
            case "less":{
                if(accio.equals("movaOPEN"))accio="movaCLOSE";
                else accio="movaOPEN";
                url=Isis3.LastDevice;
            }
            default:{

if(dispositiu.equals("PORTA.SENSOR.TANCADA")|dispositiu.equals("
CUINA.NEVERA.EN")){
                url = "DF-"+dispositiu;
        }
        else url = "DF-"+lloc+"."+dispositiu;


        }
            }
        try {
            Put(url,accio);
            Isis3.LastAction=accio;
            Isis3.LastDevice=url;
        }catch(Exception e){}
}

@Get("json")
public static Dispositiu Get(String disp)throws IOException{

  String n;
  Dispositiu d = new Dispositiu();

  n = new
ClientResource("http://localhost:8182/devFunc/"+disp).get().getT
ext();

  System.out.println(n);
  n = n.replace("-","");
  ObjectMapper om = new ObjectMapper();

om.configure(DeserializationConfig.Feature.FAIL_ON_UNKNOWN_PROPE
RTIES, false);

  d = om.readValue(n, Dispositiu.class);

  return d;
    }

@Put("json")
```

```
public static void Put(String disp, String action)throws
IOException{
    System.out.println("url"+disp+"accio"+action);

    JSONObject object = new JSONObject();
    try{
    object.put("action", action);
    }catch(Exception e){}

    Request request = new Request(Method.PUT, new
Reference("http://localhost:8182/devFunc/"+disp));
    request.setEntity(new
JsonRepresentation(object.toString()));
    Client cliente = new Client(Protocol.HTTP);
    Response response = cliente.handle(request);
    //String text = response.getEntity().getText();
    }

}
```

8.2 Configuració de l'SPHINX

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
   Sphinx-4 Configuration file
-->

<!-- ********************************************************* --
>
<!--  an4 configuration file                                   -->
<!-- ********************************************************* --
>

<config>

    <!--
************************************************* -->
    <!-- frequently tuned properties
-->
    <!--
************************************************* -->

    <property name="logLevel" value="WARNING"/>

    <property name="absoluteBeamWidth"  value="-1"/>
    <property name="relativeBeamWidth"  value="1E-80"/>
    <property name="wordInsertionProbability" value="1E-36"/>
    <property name="languageWeight"     value="8"/>

    <property name="frontend" value="epFrontEnd"/>
    <property name="recognizer" value="recognizer"/>
    <property name="showCreations" value="false"/>


    <!--
************************************************* -->
    <!-- word recognizer configuration
-->
    <!--
************************************************* -->

    <component name="recognizer"
type="edu.cmu.sphinx.recognizer.Recognizer">
        <property name="decoder" value="decoder"/>
        <propertylist name="monitors">
            <item>accuracyTracker </item>
            <item>speedTracker </item>
            <item>memoryTracker </item>
        </propertylist>
    </component>

    <!--
************************************************* -->
```

```
    <!-- The Decoder   configuration
-->
    <!--
********************************************************* -->

    <component name="decoder"
type="edu.cmu.sphinx.decoder.Decoder">
        <property name="searchManager" value="searchManager"/>
    </component>

    <component name="searchManager"

type="edu.cmu.sphinx.decoder.search.SimpleBreadthFirstSearchMana
ger">
        <property name="logMath" value="logMath"/>
        <property name="linguist" value="flatLinguist"/>
        <property name="pruner" value="trivialPruner"/>
        <property name="scorer" value="threadedScorer"/>
        <property name="activeListFactory" value="activeList"/>
    </component>


    <component name="activeList"

type="edu.cmu.sphinx.decoder.search.PartitionActiveListFactory">
        <property name="logMath" value="logMath"/>
        <property name="absoluteBeamWidth"
value="${absoluteBeamWidth}"/>
        <property name="relativeBeamWidth"
value="${relativeBeamWidth}"/>
    </component>

    <component name="trivialPruner"

type="edu.cmu.sphinx.decoder.pruner.SimplePruner"/>

    <component name="threadedScorer"

type="edu.cmu.sphinx.decoder.scorer.ThreadedAcousticScorer">
        <property name="frontend" value="${frontend}"/>
    </component>

    <!--
********************************************************* -->
    <!-- The linguist  configuration
-->
    <!--
********************************************************* -->

    <component name="flatLinguist"

type="edu.cmu.sphinx.linguist.flat.FlatLinguist">
        <property name="logMath" value="logMath"/>
        <property name="grammar" value="jsgfGrammar"/>
        <property name="acousticModel" value="wsj"/>
        <property name="wordInsertionProbability"
                value="${wordInsertionProbability}"/>
```

```
        <property name="languageWeight"
value="${languageWeight}"/>
        <property name="unitManager" value="unitManager"/>
    </component>


    <!--
********************************************************* -->
    <!-- The Grammar  configuration
-->
    <!--
********************************************************* -->

    <component name="jsgfGrammar"
type="edu.cmu.sphinx.jsgf.JSGFGrammar">
        <property name="dictionary" value="dictionary"/>
        <property name="grammarLocation"
             value="/home/manel/Projecte/Isis3/"/>
        <property name="grammarName" value="Isis"/>
      <property name="logMath" value="logMath"/>
    </component>


    <!--
********************************************************* -->
    <!-- The Dictionary configuration
-->
    <!--
********************************************************* -->

    <component name="dictionary"

type="edu.cmu.sphinx.linguist.dictionary.FastDictionary">
        <property name="dictionaryPath"

value="resource:/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz/dict/cmu
dict.0.6d"/>
        <property name="fillerPath"

value="resource:/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz/noisedic
t"/>
        <property name="addSilEndingPronunciation"
value="false"/>
        <property name="allowMissingWords" value="false"/>
        <property name="unitManager" value="unitManager"/>
    </component>

    <!--
********************************************************* -->
    <!-- The acoustic model configuration
-->
    <!--
********************************************************* -->
    <component name="wsj"

type="edu.cmu.sphinx.linguist.acoustic.tiedstate.TiedStateAcoust
icModel">
```

```
        <property name="loader" value="wsjLoader"/>
        <property name="unitManager" value="unitManager"/>
    </component>

    <component name="wsjLoader"
type="edu.cmu.sphinx.linguist.acoustic.tiedstate.Sphinx3Loader">
        <property name="logMath" value="logMath"/>
        <property name="unitManager" value="unitManager"/>
        <property name="location"
value="resource:/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz"/>
    </component>


    <!-- ********************************************************
-->
    <!-- The unit manager configuration
-->
    <!--
******************************************************** -->

    <component name="unitManager"
        type="edu.cmu.sphinx.linguist.acoustic.UnitManager"/>

    <!--
******************************************************** -->
    <!-- The frontend configuration
-->
    <!--
******************************************************** -->

    <component name="frontEnd"
type="edu.cmu.sphinx.frontend.FrontEnd">
        <propertylist name="pipeline">
            <item>microphone </item>
            <item>preemphasizer </item>
            <item>windower </item>
            <item>fft </item>
            <item>melFilterBank </item>
            <item>dct </item>
            <item>liveCMN </item>
            <item>featureExtraction </item>
        </propertylist>
    </component>

    <!--
******************************************************** -->
    <!-- The live frontend configuration
-->
    <!--
******************************************************** -->
    <component name="epFrontEnd"
type="edu.cmu.sphinx.frontend.FrontEnd">
        <propertylist name="pipeline">
            <item>microphone </item>
            <item>dataBlocker </item>
            <item>speechClassifier </item>
            <item>speechMarker </item>
```

```
            <item>nonSpeechDataFilter </item>
            <item>preemphasizer </item>
            <item>windower </item>
            <item>fft </item>
            <item>melFilterBank </item>
            <item>dct </item>
            <item>liveCMN </item>
            <item>featureExtraction </item>
        </propertylist>
    </component>

    <!--
********************************************************* -->
    <!-- The frontend pipelines
-->
    <!--
********************************************************* -->

    <component name="dataBlocker"
type="edu.cmu.sphinx.frontend.DataBlocker">
        <!--<property name="blockSizeMs" value="10"/>-->
    </component>

    <component name="speechClassifier"

type="edu.cmu.sphinx.frontend.endpoint.SpeechClassifier">
        <property name="threshold" value="13"/>
    </component>

    <component name="nonSpeechDataFilter"

type="edu.cmu.sphinx.frontend.endpoint.NonSpeechDataFilter"/>

    <component name="speechMarker"

type="edu.cmu.sphinx.frontend.endpoint.SpeechMarker" >
        <property name="speechTrailer" value="50"/>
    </component>


    <component name="preemphasizer"

type="edu.cmu.sphinx.frontend.filter.Preemphasizer"/>

    <component name="windower"

type="edu.cmu.sphinx.frontend.window.RaisedCosineWindower">
    </component>

    <component name="fft"

type="edu.cmu.sphinx.frontend.transform.DiscreteFourierTransform
">
    </component>

    <component name="melFilterBank"
```

```
type="edu.cmu.sphinx.frontend.frequencywarp.MelFrequencyFilterBa
nk">
    </component>

    <component name="dct"

type="edu.cmu.sphinx.frontend.transform.DiscreteCosineTransform"
/>

    <component name="liveCMN"
               type="edu.cmu.sphinx.frontend.feature.LiveCMN"/>

    <component name="featureExtraction"

type="edu.cmu.sphinx.frontend.feature.DeltasFeatureExtractor"/>

    <component name="microphone"
               type="edu.cmu.sphinx.frontend.util.Microphone">
        <property name="closeBetweenUtterances" value="false"/>
    </component>


    <!-- ********************************************************
-->
    <!--  monitors
-->
    <!-- ********************************************************
-->

    <component name="accuracyTracker"

type="edu.cmu.sphinx.instrumentation.BestPathAccuracyTracker">
        <property name="recognizer" value="${recognizer}"/>
        <property name="showAlignedResults" value="false"/>
        <property name="showRawResults" value="false"/>
    </component>

    <component name="memoryTracker"

type="edu.cmu.sphinx.instrumentation.MemoryTracker">
        <property name="recognizer" value="${recognizer}"/>
      <property name="showSummary" value="false"/>
      <property name="showDetails" value="false"/>
    </component>

    <component name="speedTracker"

type="edu.cmu.sphinx.instrumentation.SpeedTracker">
        <property name="recognizer" value="${recognizer}"/>
        <property name="frontend" value="${frontend}"/>
      <property name="showSummary" value="true"/>
      <property name="showDetails" value="false"/>
    </component>
```

```
    <!-- ********************************************************
-->
    <!--  Miscellaneous components
-->
    <!-- ********************************************************
-->

    <component name="logMath"
type="edu.cmu.sphinx.util.LogMath">
        <property name="logBase" value="1.0001"/>
        <property name="useAddTable" value="true"/>
    </component>

</config>
```

8.3 Clase dispositiu

```
package isis3;

import java.sql.Array;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import org.codehaus.jackson.annotate.JsonProperty;


/**
 *
 * @author manel
 */
import org.codehaus.jackson.annotate.JsonIgnoreProperties;
import org.codehaus.jackson.map.annotate.JsonSerialize;

@JsonIgnoreProperties(ignoreUnknown = true)
@JsonSerialize(include=JsonSerialize.Inclusion.NON_NULL)
    public class Dispositiu {

    //@JsonProperty("tags") private Tags tags;
    //@JsonProperty("classifiers") private Classifiers
classifiers;
    public ArrayList <Tags> tags;
    public ArrayList <Classifiers> classifiers;
    public String currentstate;
    public String link;
    public String lastaction;
    public boolean isEnabled;
    public String id;
    public boolean isQuiescent;
    public String devfuncsubtype;
   // @JsonProperty ("locations") private Locations locations;
    public ArrayList<Locations> locations;
    public String name;
```

```
    @JsonProperty ("device") private Device device;
    //protected List <Device> device;

    public String previousstate;
    public String isStarted;
    public String devfunctype;

    public String getName(){
    return this.name;
}
    /**public String getEnabled(){
        return ""+this.isEnabled;
    }*/

    public String getState(){
    return this.currentstate;
}
   // public Tags getTag(){
   //     return this.tags;
   // }
}

    class Tags {
    public String id;
    public String link;

    public void constructor(String id, String link){
        this.id= id;
        this.link = link;
    }
    public String getId(){
    return this.id;
}
public String getLink(){
    return this.link;
}
}

    class Classifiers{
    public String id;
    public String link;

    public String getID(){
        return this.id;
    }
    public String getLink(){
        return this.link;
    }
    }

    class Locations{
    public String id;
    public String link;

    public String getID(){
        return this.id;
        }
```

```
public String getLink(){
    return this.link;
    }
}

class Device{
public String id;
public String link;
public String name;
}
```

8.4 Gramàtica

```
#JSGF V1.0;

/**
 * JSGF Grammar for ISIS
 */

grammar Isis;

<keyWord>  = isis;

public <say> = <keyWord> <command0> | <keyWord> <command1> |
<keyWord> <command2> | <keyWord> <command3> | <keyWord>
<movement> | <keyWord> <specific>;

<command0> = <action> <object> <place>;
<command1> = <action> <place> <object>;
<command2> = <object> <place> <action>;
<command3> = <place> <action> <object>;

<specific> = ( fridge | door ) <action> | <action> (fridge |
door);

<movement> = ( stop | more | less | ok | okay);

<action> = ( on | off | plug | unplug | open | close | up |
down);
<object> = ( light | window | blind | climate | heater);
<place> = ( kitchen | bedroom | livingroom | toilet | bathroom);
```