RESUM

Aquest projecte presenta el disseny de una aplicació de reconeixement de veu. Aquesta aplicació té dos utilitats una és la conversió de veu a text i l'altre de text a veu.

En primer lloc s'han analitzat els diferents elements o motors de reconeixement de veu actuals del mercat amb la posterior decisió de quin és el més adequat per a una aplicació de reconeixement de veu capaç de controlar gairebé qualsevol dispositiu, senzillament realitzant petites modificacions de codi. És una aplicació que permet ús de dispositius de àudio inalàmbrics, per poder utilitzar-lo a una distància raonable del dispositiu on s'executi l'aplicació. A més a més és una aplicació robusta davant de qualsevol usuari. Independentment del tipus de veu o de la persona, té un funcionament correcte. Finalment, a part de que és personalitzable segons les necessitats del programador, també permet establir connexió amb dispositius a través de internet.

A continuació s'ha desenvolupat l'aplicació tant la part gràfica, com la part de codi. S'ha compilat en un arxiu executable i a partir de l'aplicació funcional s'ha realitzat un test de robustesa per a veure que amb diferents condicions de contorn tant internes com externes, el motor de reconeixement de veu respon segons l'esperat.

Finalment per a validar el funcionament de reconeixement de veu s'ha desenvolupat un petit dispositiu de prototipatge. Aquest dispositiu es basa en una televisió i una làmpada els quals es poden controlar a través de la veu amb unes petites modificacions de les connexions i utilitzant un petit ordinador de pont entre la interfície sobre la que corre l'aplicació i els dispositius a controlar.

ÌNDEX

1. Objectiu

L'objectiu d'aquest projecte consisteix en mitjançant un prototip, demostrar que sobre un computador amb interfície Windows, que és un dispositiu que actualment es troba a la majoria de llars, es poden crear aplicacions de reconeixement de veu per a controlar qualsevol dispositiu mecànic o electrònic. Tot això de forma inalàmbrica, mitjançant la connexió a internet, i l'establiment de una connexió client-servidor. En aquest cas s'utilitzarà una Raspberry Pi com a servidor informàtic, el qual servirà de pont entre la aplicació i els dispositius mecànics. Gràcies a aquest sistema es podran controlar els dispositius des de la aplicació de reconeixement de veu, corrent sobre un sistema operatiu Windows.

2. Motivació

Tota idea parteix de una experiència. En aquest cas el projecte ha sorgit del programa de practiques de la universitat juntament amb la empresa de tecnologia HP Inc. que ha permès al autor obtenir la experiència laboral. Gràcies al coneixements adquirits en el departament de I+D d'aquesta empresa multinacional va sorgir la idea de crear aquesta aplicació de reconeixement de veu.

Una de les inclinacions sempre ha estat la programació i aquest projecte permetia entrar completament a aquest món vist des de un altre punt de vista que el que havia obtingut de les assignatures de informàtica a la universitat. Aquest nou punt de vista és, que existeixen varis llenguatges de programació i varies interfícies o programes que permeten crear aplicacions segons el sistema operatiu que s'utilitzi per a córrer la aplicació. A la universitat tota la programació estava centrada en Linux, mentre que aquí l'autor s'ha aventurat en el món de Windows.

A més a més, el lloc de treball que m'han ofert a l'empresa em proveïa de totes les facilitats que eren necessàries per a dur a terme la aplicació. La oportunitat de disposar de software avançat que facilitava la programació del projecte i el suport a nivell de coneixement ja que es pot recórrer a professionals especialitzats, amb molta experiència i coneixements. A més a més, la companyia també disposava del material electrònic necessari per al prototipatge.

Gràcies a tot el comentat anteriorment es va començar a tirar endavant aquest projecte.

3. Abast

L'abast del projecte engloba el desenvolupament d'una aplicació de reconeixement de veu inclòs la creació d'un prototip per a poder demostrar que el dispositiu funciona i que es pot utilitzar en altres aplicacións que apareguin. Tot i això el projecte no contempla més que la creació del primer prototip de demostració. Això significa que faltarà validar el concepte, sotmetent el prototip a una gran varietat de veus, masculines i femenines, per poder confirmar que aquest dispositiu es pot comercialitzar o integrar en algun producte. L'abast detallat del projecte és el següent:

- Anàlisis d'alguns dels actuals hardwares que incorporen reconeixement de veu a les seves plaques i que es poden adaptar segons les necessitats del usuari.
- Anàlisis d'algunes de les llibreries de reconeixement de veu existents que permeten personalitzar el software a gust del usuari.
- Selecció del millor motor de veu que ofereix el mercat per a poder complir amb l'objectiu del projecte.
- Investigació de les aplicacions actuals del reconeixement de veu.
- Disseny de un prototip per a la demostració de l'aplicació de reconeixement sobre el sistema operatiu Windows.
- Disseny de un servidor Linux per poder executar-lo a la Raspberry Pi que servirà per a la connexió entre l'aplicació i el dispositiu mecànic, la làmpada i la televisió.
- Prototipatge i connexió entre el servidor (Raspberry Pi) i el client (l'aplicació), juntament amb les adequades connexions elèctriques per a controlar els dispositius electrònics, per a demostrar les oportunitats de l'aplicació.
- Pressupost de la realització del projecte.
- Estudi d'altres oportunitats que ofereix l'aplicació desenvolupada en el projecte.

El punt que no es tindrà en compte és la definició del procés de fabricació ni de l'aplicació ni de tot el muntatge de la televisió o de el control de un llum. Per tant el projecte constarà de veure quines són les possibilitats d'aquest software de reconeixement de veu juntament a un prototip per veure que el sistema és funcional.

4. Estat del art

4.1. Aplicacions de reconeixement de veu

Els sistemes de reconeixement de veu és un tipus de tecnologia molt nova que encara està en una etapa molt prematura de desenvolupament. Aquests tipus de tecnologia es va inventar el 1940 amb la col·laboració de dues empreses americanes, Bell i AT&T. Aquestes dues companyies van desenvolupar un aparell molt senzill que era capaç de reconèixer la veu. A partir d'aquí es va crear un camí cap al progrés d'aquests sistemes, centrat en les millores de precisió i robustesa d'aquesta tecnologia.

No és fins el 1960 que una altre gran empresa americana, Texas Instruments, crea una tecnologia de reconeixement de veu a la que se l'hi pot donar una utilitat. Com a tot primer dispositiu tecnològic, sempre tenen opcions de millora. És un dispositiu que requereix de fer pauses entre paraula i paraula per a que el sistema sigui capaç de reconèixer el que l'usuari està dient.

El 1970 apareix el primer producte de reconeixement de veu, el VIP100 de Threshold Technology Inc., un software que tenia un diccionari petit, que depenia del usuari i que només reconeixia paraules discretes. Durant aquesta dècada els governs comencen a investigar sobre aquest tipus de tecnologia, sobretot la agencia ARPA[1], una divisió del departament de defensa dels Estats Units.

Figura 1: Logo agencia ARPA

[1] ARPA (Advanced Research Projects Agency), agència creada el 1958 als EE.UU. en resposta al desenvolupament militar i tecnològic de la URSS durant la Guerra freda. També és l'empresa que va assentar les bases de la red global de computadors.

Finalment el 1980 apareixen ja sistemes de reconeixement de veu més avançats, amb una gran quantitat de paraules en les seves llibreries, i que fan que aquesta tecnologia sigui més accessible. Algunes d'aquestes empreses són Philips, Speechworks, Intel, Dragon Systems o Microsoft.

El reconeixement de veu és un sector que porta uns anys de creixement exponencial com la majoria de les tecnologies. Cada cop són més les aplicacions que s'han donat en aquests sistemes, degut a que també milloren les tecnologies referents a aquesta funcionalitat. Actualment les aplicacions on més s'està utilitzat el reconeixement de veu són:

- **Contestadors automàtics.** Una de les primeres utilitats de reconeixement de veu va ser la de distribuir les trucades entrants en el servei d'assistència telefònica de varies companyies, com Telefònica, Vodafone o Orange en el nostre país, mitjançant preguntes automàtiques i on es reconeixen les respostes dels usuaris utilitzant un motor de veu, de manera que segons les respostes dels usuaris la trucada es dirigeix automàticament a un departament o un altre dins de la mateixa companyia.

Figura 2: Empreses que utilitzen un sistema de reconeixement de veu.

- **Assistents personals.** Són un servei que s'utilitzen en la majoria de dispositius de computació, ordinadors personals, ordinadors dels cotxes o els *smartphones,* on es controlen les aplicacions que formen part d'aquests dispositius mitjançant el reconeixement de veu. Aquest tipus d'aplicacions s'anomenen assistents personals.

  Actualment s'estan centrant en els dispositius mòbils amb les aportacions de *Siri*, *Cortana* o *Google Now*, els tres formen part de diferents plataformes però tenen en comú que són motors de veu amb els quals pots conversar i realitzen les accions que se'ls ordena.

  Tot i això també existeixen assistents personals per a computadors personals, com poden ser *Jarvis* o *Julius*, que són projectes en desenvolupament que es van actualitzant amb l'aparició de nous algoritmes, cada cop més precisos, de reconeixement de veu. Utilitzant aquestes aplicacions s'han realitzat molts projectes personals de control de veu de robots, com controlar el moviment de un petit vehicle robot o el moviment d'un braç robòtic.

Figura 3: Asistents de reconeixement de veu per a smartphones actuals, Cortana i Siri.

- **Automatització de les vivendes (casa domòtica).** Dins aquest conjunt, en formen part, la il·luminació de les diferents habitacions, un sistema intel·ligent de calefacció on es pot controlar la temperatura simplement amb la veu, dispositius electrònics com poden ser la televisió, un despertador o un sistema electrònic de música i, fins i tot, sistemes mecànics automatitzats com pot ser el moviment vertical d'una taula - que ho permet-hi - o el moviment de compartiments amagats en els sostre de la casa i, fins i tot, el control del fluids dins de les tuberes, modificant el comportament de les vàlvules de les tuberes de gas o les tuberes d'aigua.

Per exemple, hi ha fabricants com *Siemens* que estan desenvolupant un element que no ha experimentat millores significatives els últims anys com és un forn o microones que es podran controlar a través de la veu, per a canviar la temperatura o configurar el temps de cocció.

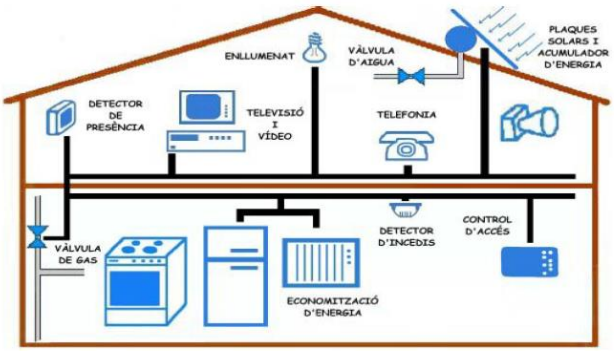

Figura 4: Disseny esquemàtic de una casa domòtica.

- **Control de robots.** [2]En el àmbit aeroespacial s'utilitza el reconeixement de veu, com per exemple el dispositiu *Mars Polar Lander* de la NASA, que és un robot explorador al que se li poden donar ordres mitjançant la veu. Aquest robot utilitza tecnologia de reconeixement de veu de la companyia *Sensory, Inc* .

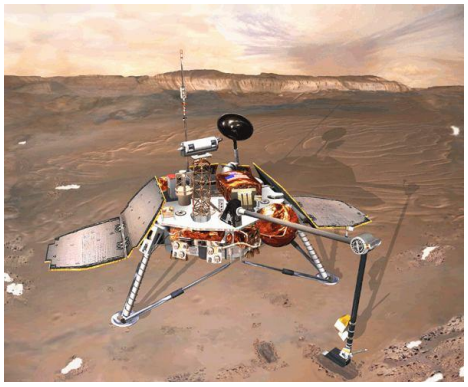

Figura 5: Robot Mars Polar Lander, NASA.

[2] http://nssdc.gsfc.nasa.gov/image/spacecraft/mars_polar_lander.jpg

- **Subtitulació autònoma de vídeos.** Per a la subtitulació automàtica de vídeos també s'utilitzen softwares de reconeixement de veu a on es capta l'àudio dels vídeos, identificant les paraules reproduïdes i es converteixen en text. Aquest sistema l'utilitza la subtitulació automàtica de *Youtube*.

Figura 6: Sistema de subtitulació automàtica del Youtube.

- **Aplicació a les xarxes socials.** Altres aplicacions que es volen portar a la pràctica és incorporar reconeixement de veu a les xarxes socials com poden ser *Facebook* o *Twitter* que tot i que actualment no està implementat, hi ha indicis que aquestes xarxes socials podrien, en un futur, utilitzar també el reconeixement de veu per a navegar dins d'elles. Això obriria una nova porta que és el utilitzar el reconeixement de veu, no només en programes i aplicacions directament, sinó que també dins de la pròpia web de internet.

- **Aplicació al sector militar.** [3]En el sector més avançat tecnològicament, el sector militar, també s'estan creant projectes de reconeixement de veu. A part de control de robots i vehicles utilitzant la veu - que s'han comentat anteriorment – hi ha moltes més aplicacions. Algunes d'aquestes aplicacions són, el control de les comunicacions que es poden interceptar d'altres països, així es pot automatitzar el procés o el reconeixement de veu per a l'accés a dispositius informàtics, utilitzant-ho de contrasenya biomètrica. I com a ultima aplicació també s'utilitza com a traductor en temps real, per comunicar-se amb altres països, de manera que tot el que diu un persona es reconegut per el sistema de reconeixement de veu i posteriorment traduït per un programa informàtic.

[3] http://www.stephanepigeon.com/Docs/TR-IST-037-ALL.pdf

4.1. Tecnologies de reconeixement de veu

Actualment el mercat és molt gran i ofereix moltes possibilitats relacionades amb les tecnologies de reconeixement de veu. Tots els sistemes de reconeixement de veu tenen les seves especificacions però dins d'aquestes característiques es distingiran en dos grups.

Primer la tecnologia basada en components de hardware, on es parteix d'una placa de circuït imprès i per tant tot el firmware que s'encarrega el reconeixement de veu i la conversió de text a veu es munta sobre aquesta. Aquesta tecnologia usualment no necessita de coneixements de programació sinó aprendre a fer funcionar el software complementari que ofereix.

Per altre banda tenim tecnologia basada en software. Aquesta concedeix la possibilitat de utilitzar llibreries i eines de desenvolupament, per a construir un arxiu executable. Aquest arxiu pot presentar la funcionalitat de reconeixement de veu i el post processat un cop reconegut. Normalment aquests segons tipus de tecnologies també presenten la conversió de text a veu, de manera que són sistemes bidireccionals.

Els dispositius de hardware consisteixen en una petita placa (PCB[4]) que conté totes les llibreries i sistemes necessaris per a que es pugui implementar un reconeixement de veu de forma senzilla i eficaç. Aquests dispositius en general disposen de diferents connectors, per a micròfon ja sigui en forma de pin o de port mini JACK, suport per a dispositius USB, o connexió WIFI/Buetooth per a connectar dispositius sense fils. Aquests sistemes són interesants per a veure si es pot implementar una entrada d'àudio adient. Cal comentar que molts dispositius ja incorporen un micròfon en ells i per tant aquestes interfícies no són necessàries, però aquests micròfons normalment són amb cable, cosa que no interessa seguint els requisits comentats posteriorment.

[4] http://www.pcb.electrosoft.cl/04-articulos-circuitos-impresos-desarrollo-sistemas/01-conceptos-circuitos-impresos/conceptos-circuitos-impresos-pcb.html

La majoria d'aquests dispositius són complementaris, es a dir, necessiten d'un altre dispositiu, un mini computador, per a realitzar les tasques de càlcul que no són capaços de processar en la pròpia placa del dispositiu que incorporen un processador amb poca potència. Una altre característica única de les propostes de hardware és el tipus d'interfície per a connectar-la al mini computador, depenent d'aquesta interfície la velocitat de transmissió de les dades i la distància a la que poden estar un dispositiu respecte l'altre queda completament limitat.

Pel que fa a les solucions tipus software es miren unes altres característiques, com són el nombre de instruccions que són capaços d'emmagatzemar, que determinen el nombre de paraules o frases que és capaç de identificar el sistema de reconeixement de veu. Una altre característica important d'aquests elements és el tipus de llicència, si disposen de una llicencia de codi obert (*open source)* o de una llicència privada i a més a més si es tracta d'una llicència gratuïta o no, per saber si és possible implementar-la, o no, en aquest projecte. Finalment l'ultima característica a tenir en compte és sobre quin dispositiu o sistema operatiu es pot treballar amb aquestes distribucions de software. El sistema operatiu escollit és molt important per a l'experiència de l'usuari.

Hi ha característiques que són importants a tenir en compte en els dos casos. Una és el tipus de sistema de reconeixement de veu. Hi ha dos tipus de reconeixement de veu segons el usuari, el SI (speaker independent) i el SD (speaker dependent). SI significa que és independent del usuari de la aplicació i per tant reconeix a qualsevol persona que vulgui utilitzar-la sense necessitat de cap configuració. En canvi SD significa que té guardat el perfil de tots els usuaris que han de utilitzar la aplicació només funciona correctament amb aquest usuari. Això significa que cada usuari ha de passar per un procés de aprenentatge (*trainning*) que ensenya a la aplicació la pronunciació de cada usuari i es guarda en un perfil. Aquest perfil s'ha d'anar canviant cada cop que es canvia el usuari de la màquina.

La llista dels dispositius que actualment estan al mercat i que poden ser els adequats per a l'objectiu d'aquest projecte, que és aconseguir controlar un dispositiu mitjançant la veu, és la següent:

- **Elechouse Voice Recognition Module**

És un dispositiu de reconeixement de veu basat en hardware, concretament una placa electrònica que disposa de un processador que s'encarrega de processar el àudio que li arriba per un port mini JAC que incorpora en la placa.

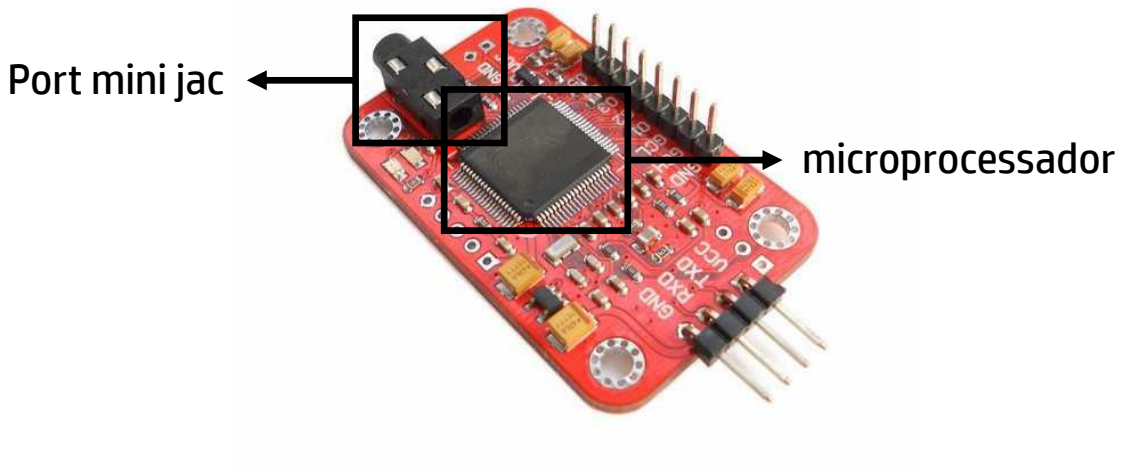

Figura 7: Placa PCB elechouse voice recognition module.

Aquest dispositiu és complementari de un altre dispositiu que és una altre placa amb un processador però programable, una Arduino. És necessari que un cop el mòdul de Elechouse ha processat la veu del usuari, decidir que es fa amb aquesta informació. Per decidir que es fa amb la informació s'utilitza la Arduino que és fàcilment programable. Aquest mòdul està preparat amb varies llibreries d'Arduino per a poder establir aquesta connexió de forma fàcil i eficient.

Aquest mòdul té una petita desavantatge i és que només és capaç de guardar com a molt 80 comandes de veu i aquestes s'han de gravar anteriorment a utilitzar-les. És per això que el sistema és dependent del usuari, que significa que cada usuari que volgués utilitzar aquest sistema hauria de gravar les paraules que necessités utilitzar, per això aquest sistema és poc robust per a diferents usuaris. El tipus de connexió que estableix el mòdul amb qualsevol altre dispositiu utilitza un protocol serial[5] com els USB. Finalment comentar que el preu per unitat d'aquest mòdul és aproximadament de 23,5 $.

[5] El protocol serial és un tipus de connexió que permet la comunicación entre varis dispositius i permet la transferència de dades de forma rápida.

- **Easy Vr Shield 3.0**

Aquest dispositiu és molt semblant al anterior però amb unes quantes millores. També es tracta de una placa electrònica amb un microprocessador encarregat del tractament de l'àudio però en aquest cas no es disposa de un port mini JAC per a connectar el micròfon sinó que la pròpia placa porta un micròfon incorporat. A més a més aquesta placa està directament pensada per Arduino, ja que està feta per poder acoblar directament aquesta placa a la de l'Arduino aconseguint que sembli un únic dispositiu.

Figura 8: Placa PCB easy vr shield 3.0 amb el micròfon

Aquest mòdul en concret consta de 28 comandes que es poden configurar independentment de qui sigui l'usuari (SI) del sistema que les hauria de reconèixer igualment. A més a més també, es poden configurar 32 comandes més, però en aquest cas són úniques per a un usuari (SD) i per tant s'han de gravar en àudio per a cada usuari i són exclusives per aquella persona que les ha gravat.

Aquest dispositiu suporta varis llenguatges, el anglès americà, el japonès, l'espanyol, l'italià, el alemany i el francès, per a les comandes SI. Per a configurar aquestes comandes és necessari que el mòdul es configuri mitjançant un PC Windows amb un software la llicència que està inclòs en la compra del dispositiu. Aquest software té una interfície molt intuïtiva i permet seleccionar tant el llenguatge com les comandes.

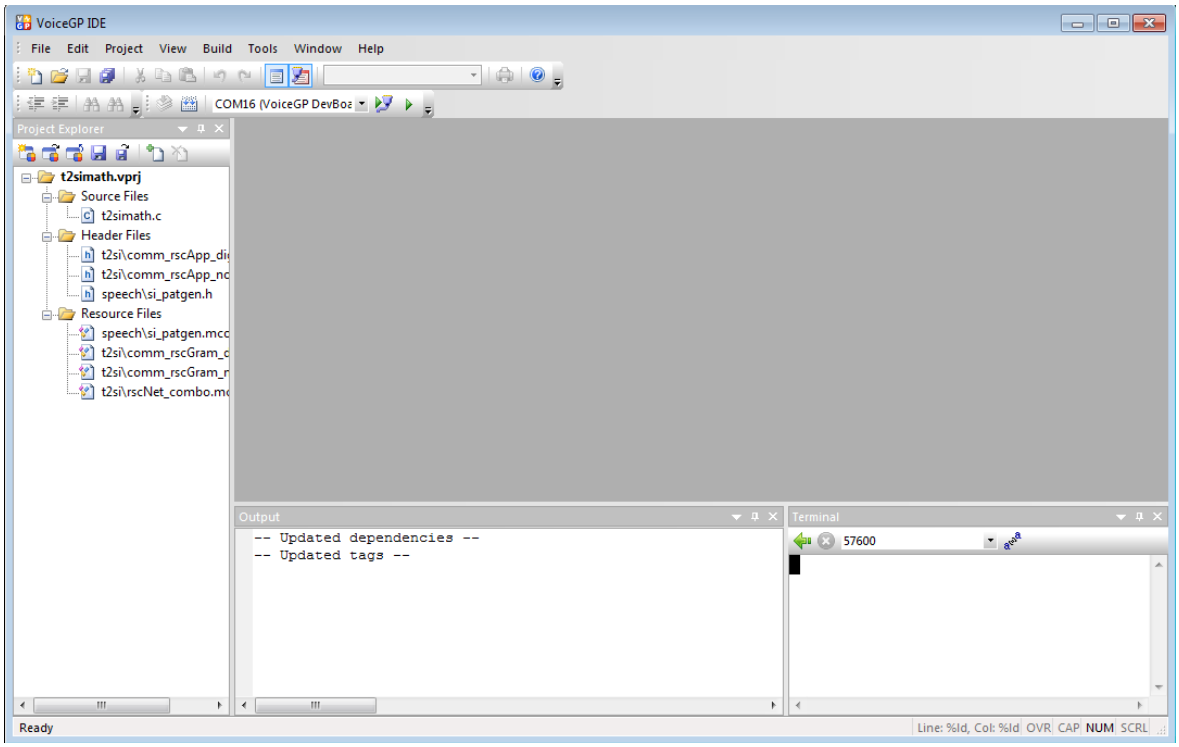

Figura 9: Finestra de inici del software .

Un cop generades les comandes funciona exactament igual que el anterior, és el mòdul l'encarregat del reconeixement de veu i l'Arduino s'encarrega del post procés. El cost d'aquest dispositiu és major que l'anterior, aproximadament 43,5$ per unitat. Però incorpora la llicència del software de configuració.

- **LC LD3320 ASR**

Aquest dispositiu també és una placa PCB com les altres. Però té una virtut respecte les altres i és que no necessita configuració prèvia. Senzillament converteix la veu que rep a text i la expulsa per un dels ports de connexió que suporta. Aquesta placa es pot controlar mitjançant qualsevol dispositiu que suporta interfície de connexió en paral·lel[6] o SPI[7], dos protocols diferents. La Arduino o altres *single board computers* com la Raspberry Pi (de la qual parlarem més endavant) suporten aquests protocols. Per tant seria necessari un altre dispositiu per al post processat del reconeixement de veu.

En aquest dispositiu, al no necessitar una preconfiguració, totes les comandes són SI i per tant el pot usar qualsevol usuari i hauria de reconèixer la veu d'aquests sense cap problema.

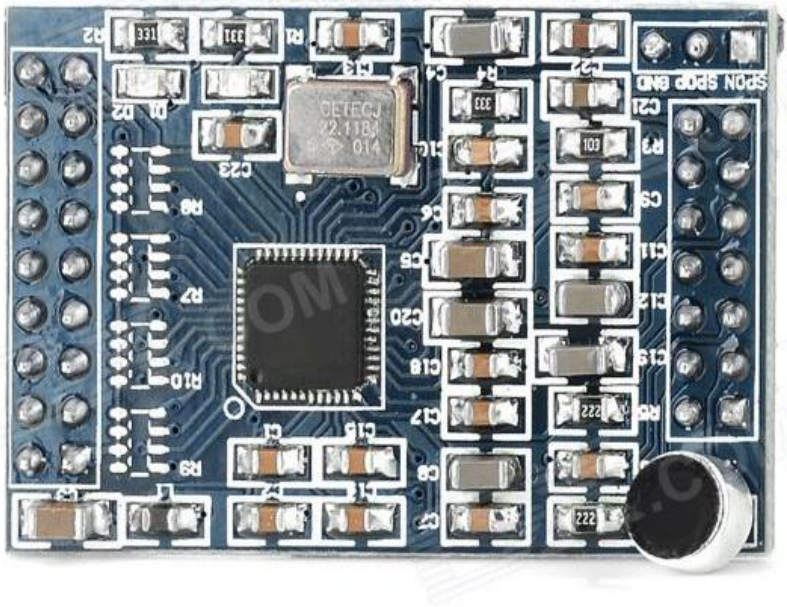

Figura 10: Placa PCB del sistema de reconeixement de veu LC LD3320 ASR

[6] La interfície paral·lel és una connexió entre un ordinador i un perifèric amb la peculiaritat que s'envien paquets de bits de dades en forma de bytes, en comptes de enviar la información bit a bit.

[7] La inerfície SPI (Serial Peripheral Interface) és un estàndard de comunicació que s'utilitza per a la transferència de información entre circuits integrts electrònics, té dos peculiaritats, una que és la interfície més rápida d'aquest tipus i laltre que no és estable si s'utilitzen cables llargs.

- **BitVoicer**

Un cop contemplades totes les opcions de hardware on el reconeixement de veu està integrat en un dispositiu físic, es compararan amb les opcions de software/firmware. El primer sistema de reconeixement de veu basat en software és el BitVoicer. És un software que permet incorporar en un microcontrolador un sistema de reconeixement de veu. El punt fort d'aquest software és que és totalment personalitzable. És capaç de capturar àudio i processar-lo en un microcontrolador de manera que el resultat és un text. Suporta molts idiomes, els més utilitzats en el món i altres com el danès o el finlandès. Aquest tipus de reconeixement de veu és totalment configurable:

- El volum mínim a partir del qual comença a ser efectiu el reconeixement de veu, deixant a part l'àudio amb un volum menor.
- El nivell de confiança entre la paraula pertanyent a la llibreria i la paraula reconeguda per el software.
- El nivell de latència entre que l'usuari diu alguna cosa fins que el microcontrolador reporta el text.
- Una paraula clau a partir de la qual s'activa el reconeixement de veu.

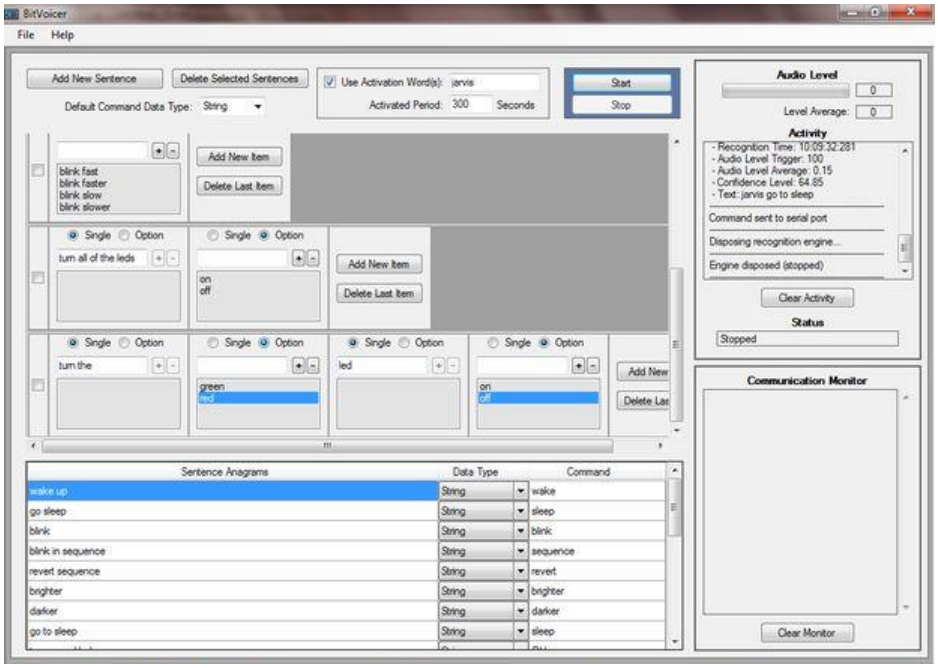

Figura 11: Software BitVoicer

Un altre avantatge és que es pot comunicar amb altres dispositius, tant per USB com a través de una xarxa de internet, de manera que no són necessaris fils si es disposa de WIFI. Disposa d'una interfície gràfica que permet tenir controlats tots aquests elements i el estatus actual del reconeixement de veu a través de un PC que tingui aquest software instal·lat. A més a més aquest software permet dotar de reconeixement de veu una Arduino de la qual s'ha parlat anteriorment. Aquest software no disposa de llicencia lliure i té un cost mensual de 4,5$.

- **Dragon Naturally Speaking 13**

Aquest és el software de reconeixement de veu més treballat sobre Windows. És un software capaç de comunicar-se amb qualsevol aplicació de Windows i a més a més té la virtut que no és necessari cap tipus de coneixement de programació ja que és *plug&play*. El problema principal és que aquest software ja està creat, i té una empresa darrere. Per tant no es pot accedir al codi i modificar-lo per a adaptar-lo a aquest projecte concret.

Les especificació més limitant d'aquest software és que només es pot instal·lar sobre el sistema operatiu Microsoft Windows. Sobre aquest sistema operatiu és capaç de controlar la majoria d'aplicacions. Té una resposta molt ràpida a la veu i gran precisió, comandes de veu configurables o també és capaç de interpretar el dictat de text. Té uns requisits de l'ordinador sobre el que s'instal·la molt elevats comparat amb el software anterior. Aquest software també incorpora la funcionalitat de assistent personal de l'ordinador, permet controlar totes les funcions de l'ordinador per veu, una característica innecessària però que suposa un cost afegit en el preu final del producte.

Figura 12: Producte Dragon Naturally Speaking 13.

- **Windows speech platform**

Finalment actualment hi ha una altre opció de software en la que es necessiten coneixements de programació no molt elevats però sí que s'ha de conèixer un llenguatge típic en programació com és el C, C++ o el C#. Tots 3 variants de un mateix llenguatge.

Aquest software consisteix en varies llibreries organitzades en un SDK (*software devolopment kit*) que són les eines necessàries per a crear un software en aquest cas de reconeixement de veu. Amb aquest kit de eines juntament amb les eines bàsiques de programació ja es pot crear una aplicació de Windows totalment personalitzada i capaç de realitzar totes les funcions inherents al sistema operatiu. Tot això amb un sistema de reconeixement de veu incorporat.

El punt fort d'aquest software és que la llicència d'aquestes eines va lligat a la del sistema operatiu, és a dir, la pròpia empresa Microsoft allibera la possibilitat de tenir problemes legals sempre i quan s'executi aquesta aplicació sobre una copia del sistema operatiu original.

Més d'un 80% de la població mundial disposa de un dispositiu amb aquest sistema operatiu. Això significa que és una bona plataforma sobre la que treballar el sistema de reconeixement de veu. Aquesta forma de concebre el concepte de una aplicació de reconeixement de veu permet modificar la aplicació a petició del usuari. Per tant no té una única aplicació directe al món real sinó una gran varietat.

Figura 13: Logo de Microsoft Windows.

5. Especificacions bàsiques

Abans de poder decidir quina és la millor opció de les comentades anteriorment , s'han de valorar quines són les necessitats i els requeriments d'aquest projecte. Al inici del projecte s'han presentat les següents necessitats:

- Un sistema de reconeixement robust per què es pugui utilitzar en moltes aplicacions.
- Un sistema fàcil de configurar i de utilitzar. És important que les llibreries de paraules capaces de reconèixer tinguin una fàcil configuració.
- Incorporació dels fonemes i els sons més habituals per a un millor reconeixement.
- Suport en els llenguatges més utilitzats arreu del món per a una possible personalització de l'aplicació segons la localització. L'anglès, el castellà, el francès, el xinès, l'alemany, el japonès i el català.
- Tenir el suport de una companyia que garanteixi que les llibreries utilitzades no es quedin obsoletes en poc temps.
- Suport de auriculars inalàmbric ja sigui per *bluetooth*, radiofreqüència, *WIFI* o utilitzant un *dongle* (llapis USB), per a poder utilitzar auriculars amb micròfon a distància i no dependre d'estar a prop de l'ordinador o la placa.
- Reconeixement de veu independent de la persona que l'estigui utilitzant, és a dir un sol reconeixement global per a la majoria de usuaris segons el dialecte, de manera que no s'hagi de fer la àrdua tasca de configurar per a cada usuari l'aplicació.
- Evitar que les llicencies suposin una barrera a l'hora de implementar el dispositiu escollit. La millor opció és buscar una solució *open source* de manera que no hi hagi cap problema legal o de Copyright al implementar el hardware o software en el producte. En cas que no sigui possible aquesta opció, l'altre cosa que s'ha de mirar és que la llicència sigui comercial i que es pugui disposar de ella fàcilment.

- Finalment l'últim requeriment, però no menys important, és que la aplicació es pugui personalitzar segons les necessitats de cadascú.

Un cop establerts els requeriments del projecte només falta decidir quina és la opció de les comentades anteriorment que les compleix totes.

Fent una comparació de totes les solucions als problemes que hi ha en el mercat, i que s'han detallat anteriorment, s'ha escollit el sistema de reconeixement de veu Microsoft Windows Platform, ja que té un valor afegit i és que permet crear una aplicació totalment des de zero i personalitzar-la i configurar-la tant gràficament com les funcionalitats.

Requeriments	Elechouse Voice Recognition Module	Easy Vr Shield 3.0	LC LD3320 ASR	BitVoicer	Dragon Naturally Speaking 13	Windows speech platform
Cost	6	4	7	8	1	**10**
Requeriments obligatoris						
Speaker dependent (SD)	NO	NO	SÍ	SÍ	SÍ	SÍ
Àudio WIFI	NO	NO	NO	SÍ	SÍ	SÍ
Llicència oberta	NO	NO	NO	NO	NO	SÍ

Taula 1: Comparació dels sistemes de reconeixement de veu.

6. Desenvolupament del projecte

6.1. Introducció

Primer de tot es s'ha procedit a programar l'aplicació de reconeixement de veu utilitzant un software de desenvolupament d'aplicacions per a Windows, el Visual Studio 2013, amb el llenguatge de programació C#, una variant del conegut llenguatge C, però més intuïtiu per a la gent que està començant en aquesta branca de coneixement, la programació. Aquesta aplicació s'anomenarà *Mock Up*. Un cop feta l'aplicació el següent pas és establir la connexió entre la aplicació de Windows i el mini computador (la Raspberry Pi). S'ha instal·lat sobre el ordinador de placa única el sistema operatiu Raspbian el qual és una variant del sistema operatiu lliure Linux optimitzat i adaptat per a el correcte funcionament sobre una Raspberry Pi. Per a aconseguir això s'ha desenvolupat un programa servidor per a establir la connexió mitjançant una xarxa de internet. Aquest servidor s'ha programat utilitzant el llenguatge de programació TCL, que és un dels més utilitzats per a crear servidors.

El segon pas ha sigut crear les funcions que donen ordres a la Raspberry Pi, de manera que serveix per a controlar els dispositius electrònics que van connectats a aquesta placa.

Finalment el que s'ha fet a sigut crear un prototip de l'aplicació, juntament amb l'ordinador amb sistema operatiu Windows i el ordinador amb sistema operatiu Raspbian, per a demostrar que el projecte és una realitat i no només una teoria. Aquesta fase és la més important ja que és quan es reflecteix tot l'esforç realitzat durant el projecte i si tot el que s'ha estudiat en aquest projecte concorda amb el funcionament real.

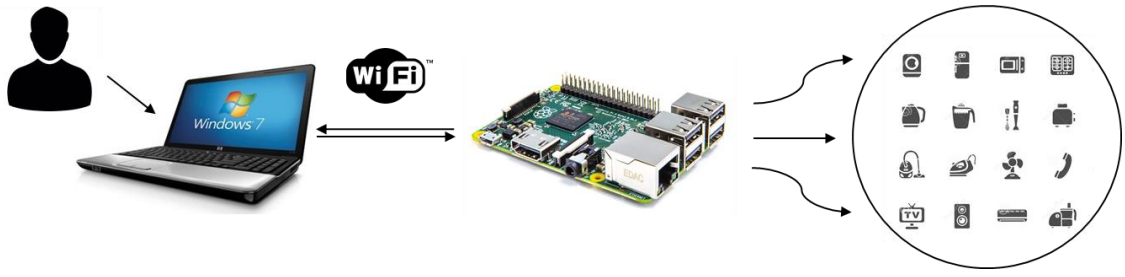

Figura 14: Esquemàtic de funcionament general del prototip.

6.2. Visual Studio 2013

L'aplicació Visual Studio és una eina per a poder programar de manera senzilla dins d'una plataforma que simplifica molt les coses al usuari sobretot quan es tracta de fer aplicacions per a la interfície Windows. Aquest programa disposa de una infinitat de eines útils, però en aquest projecte només se'n han utilitzat una desena. En aquest programa es pot utilitzar varis llenguatges de programació, el Visual Basic, Visual C#, Visual C++, Visual F# o JScript. Entre tots aquests s'ha escollit el Visual C# que és molt senzill d'entendre i programar.

Per a crear el programa de reconeixement de veu s'ha creat un projecte de Visual Studio del tipus *Windows Form Aplication* a on tot el codi va referenciat a una finestra o interfície gràfica. L'arbre de l'aplicació consisteix en varies formes les quals contenen un codi que es crea automàticament i que defineix la posició, forma, text i color de l'element que conté la finestra principal i a més a més inclou un altre arxiu de codi (.cs) que és la part programable.

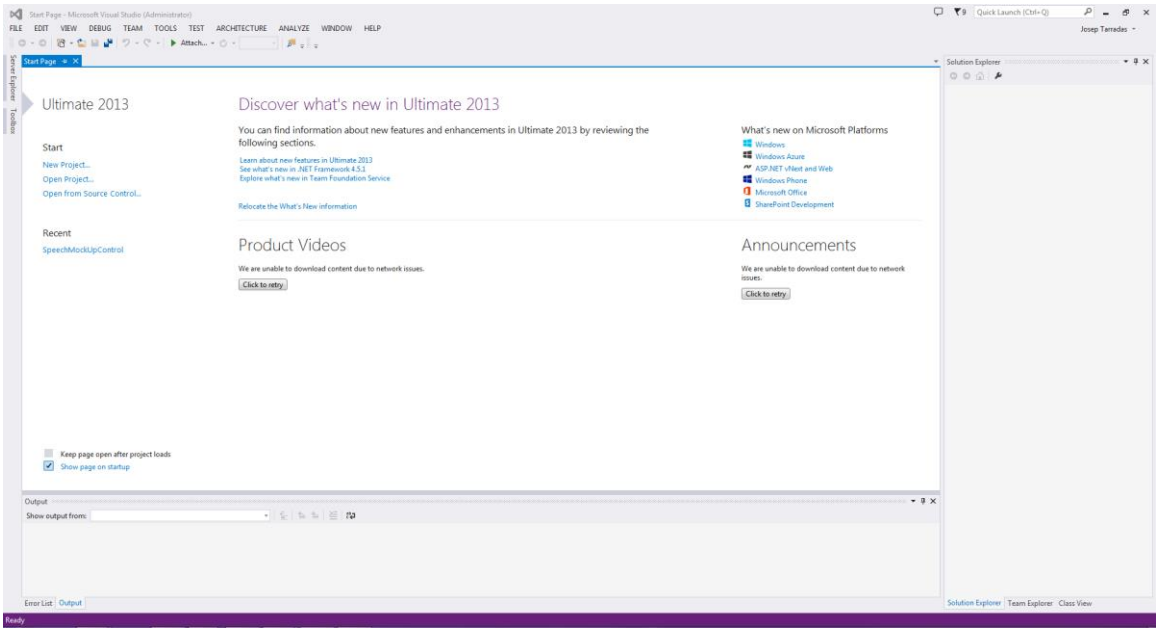

Figura 15: Finestra inicial del Visual Studio 2013 Ultimate

A part de les formes, del programa principal també pengen altres finestres com les propietats del projecte o les referències. En aquest projecte s'han deixat les propietats per defecte. En canvi les referencies si que s'han modificat, ja que aquesta pestanya conté les llibreries que es volen utilitzar en el projecte (SDK[8]). Per l'aplicació de reconeixement de veu s'ha utilitzat el software de desenvolupament anomenat "Microsoft Speech SDK 5.3 (SAPI)" juntament amb el "Microsoft Speech platform runtime 11". La llibreria "Microsoft Speech SDK 5.3 (SAPI)" és la que s'ha inclòs a la pestanya referències i que permet executar funcions especifiques de reconeixement de veu com executar el motor de reconeixement de veu. El "Microsoft Speech platform runtime 11" és l'encarregat de buscar i executar les llibreries dels diferents llenguatges tant de reconeixement de veu com la conversió de veu a text, quan el programa principal de l'aplicació en fa referencia. La SAPI s'ha de incloure a les referències mitjançant l'acció *add references*.

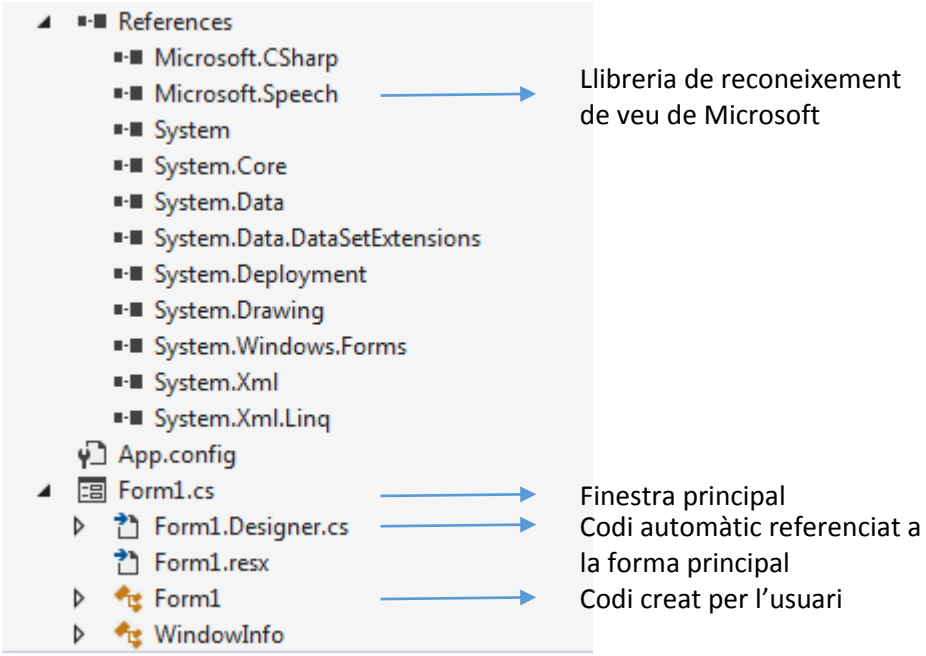

Figura 16: Arbre de funcions del projecte en Visual Studio

[8] SDK (software development kit) és un conjunct de eines de desenvolupament de software que permet al programador crear softwares o aplicacions per a un sistema concret.

La major virtut d'aquest programa és que permet realitzar tant la part gràfica com la part de de programació del codi en un mateix projecte sense haver de crear varis arxius diferents i tot això de forma fàcil i intuïtiva. Un cop creats els arxius el Visual Studio et dóna la opció de compilar tot el codi amb un simple clic.

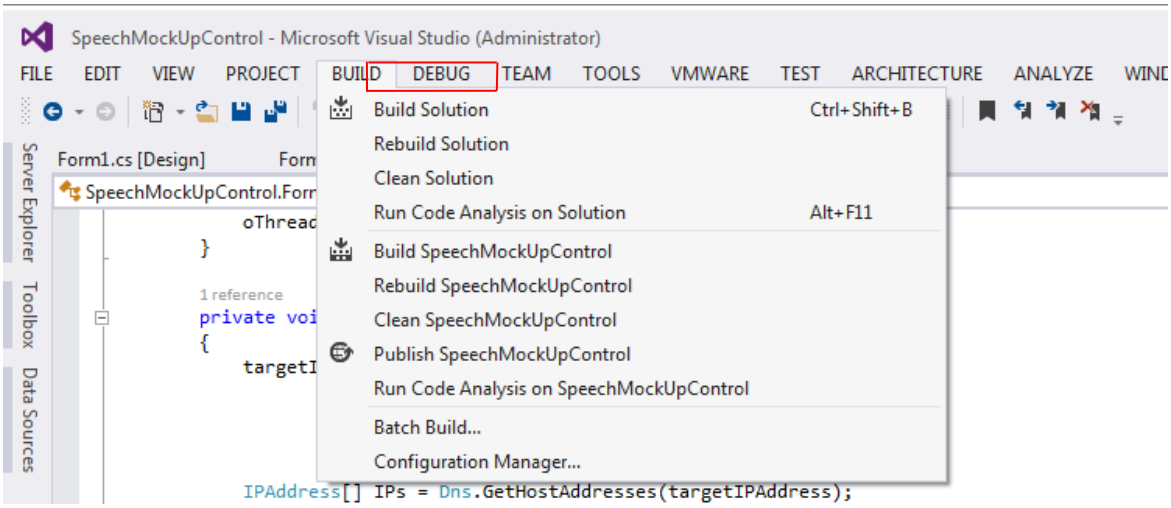

Figura 17: Pestanya de compilació del Visual Studio

Com s'ha comentat anteriorment un dels punts forts d'aquesta aplicació és la part de programació de la interfície gràfica de l'aplicació.

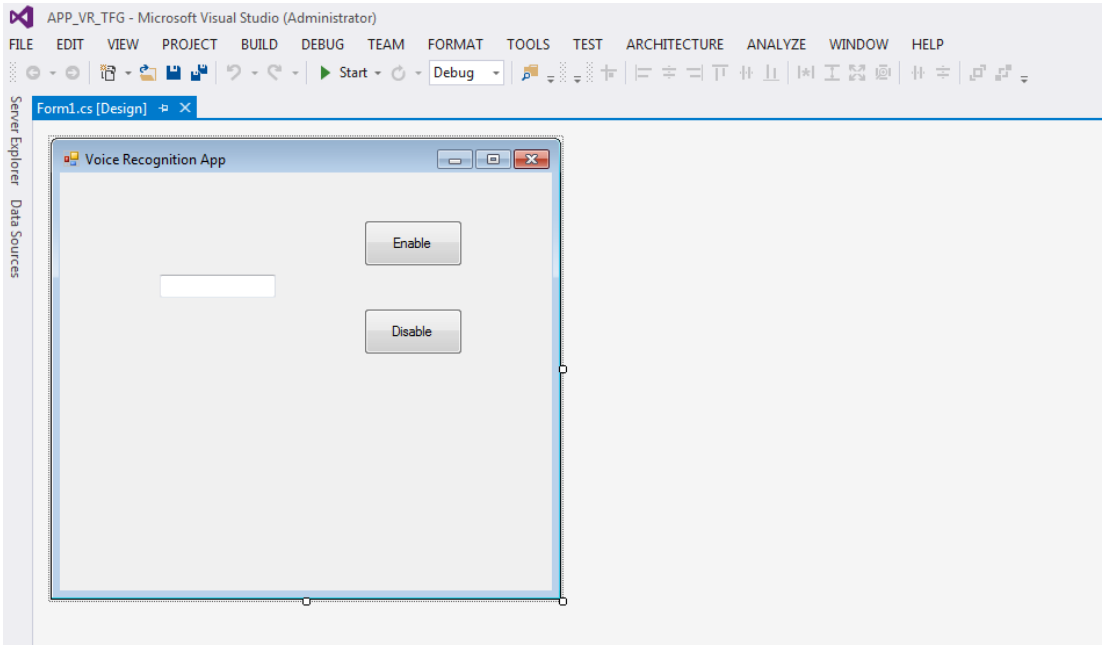

Figura 18: Interfície gràfica del programa Visual Studio.

A diferència d'altres programes o llibreríes tota la part del disseny està representat graficament, és a dir, que és pot colocar tots els elements sense haber de introduïr en el codi les coordenades de cada element com a la majoria de programes, senzillament arrastrant l'element amb el botó dret del ratolí. Gràcies a això es pot veure el resultat final abans d'executar l'aplicació. D'aquesta manera és molt senzill colocar els diferents els elements, un botó o un quadre de text, a la posició desitjada visualment. Tots els elements que es poden incorporar a la interficie gràfica es troben a la pestanya *Toolbox* del programa.

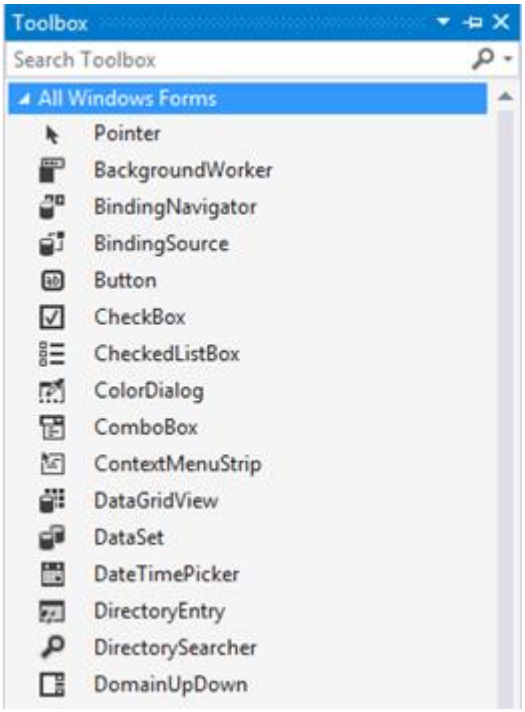

Figura 19: Eines de Windows per a la interfície gràfica

6.3. Raspberry Pi

Un element necessari per a posar en pràctica el reconeixement de veu és una SBC (*Single Board Computer*[9]), un ordinador de placa única. Aquest element només funciona de pont entre l'aplicació de reconeixement de veu i els dispositius a controlar en el prototipatge. Obviament, la majoria de electrodomèstics actuals no disposen de un dispositiu amb un port TCP[10] obert, per tant per a poder connectar directament la aplicació és necessari aquest ordinador pont.

La funció d'aquest dispositiu és interpretar la informació que envia el *Mock Up* després de reconèixer el missatge de veu. La informació s'envia codificada en una sèrie de bytes que fan referència a una funció de la SBC que la descodifica. Aquesta funció genera una senyal elèctrica per un dels seus ports GPIO[11] de manera que si tot està ben connectat és capaç de controlar un dispositiu extern.

D'aquest tipus de ordinadors n'hi ha varis però els més coneguts són la Raspberry Pi, l'Arduino i la BeagleBone.

La Arduino és la més senzilla de totes, és la que té menys potencia de càlcul i té poques funcionalitats. Aquesta es va descartar ja que no disposa de connexió a una xarxa de internet a través de WIFI i no dóna tanta llibertat a la hora de programar com els altres ordinadors ja que s'ha de fer en un llenguatge concret sobre una aplicació pròpia de Arduino. Tampoc ofereix la possibilitat de escollir un sistema operatiu, de manera que no és el dispositiu el que s'adapta a les necessitats del programador sinó que tot el contrari.

---

[9] Una *Single Board Computer* no és més que un petit ordinador poc potent i sense cap perifèric, no disposa de pantalla ni de teclat, però amb moltes connexions diferents per a poder connectar-se a qualsevol dispositiu.

[10] Un port de connexió TCP és un port que permet, mitjançant una adreça IP i un número de port, establir connexions entre dispositius dins de una mateixa xarxa de internet. Podent enviar informació entre els dispositius.

[11] Un port GPIO és un pin que funciona com un interruptor, o està obert o està tancat, que es pot controlar en temps real.

De les altres dos interfícies existents s'ha de dir que a nivell de hardware no hi ha cap diferència entre les dos, les diferències però venen a nivell de sistema operatiu. En el cas de la Raspberry existeix un sistema operatiu Linux específic per a aquesta interfície, anomenat Raspbian i que no només disposa de terminal sinó que també de interfície gràfica. En canvi el sistema operatiu de BeagleBone no disposa de interfície gràfica i per tant és menys intuïtiva. A part d'això a Espanya només es distribueix la Raspberry Pi i per tant és més fàcil d'aconseguir.

La decisió final ha estat utilitzar una Rasberry Pi amb el sistema operatiu basat en Linux, Raspbian. A on s'executarà un programa servidor per a poder connectar-s'hi mitjançant l'aplicació de reconeixement de veu.

Figura 20: Raspberry Pi

6.4. Diagrama de funcionament

Per a entendre com funciona tot el sistema complert de funcionament de l'aplicació de reconeixement de veu a continuació es mostra un diagrama del flux de treball del sistema incloses les diferents funcions del reconeixement de veu.

El flux de treball té dos branques. Una que reconeix la veu del operador del *Mock Up* i l'altre on el *Mock Up* converteix la veu en text i per tant és la aplicació la que informa al operador. Cada branca té el seu mode de funcionament.

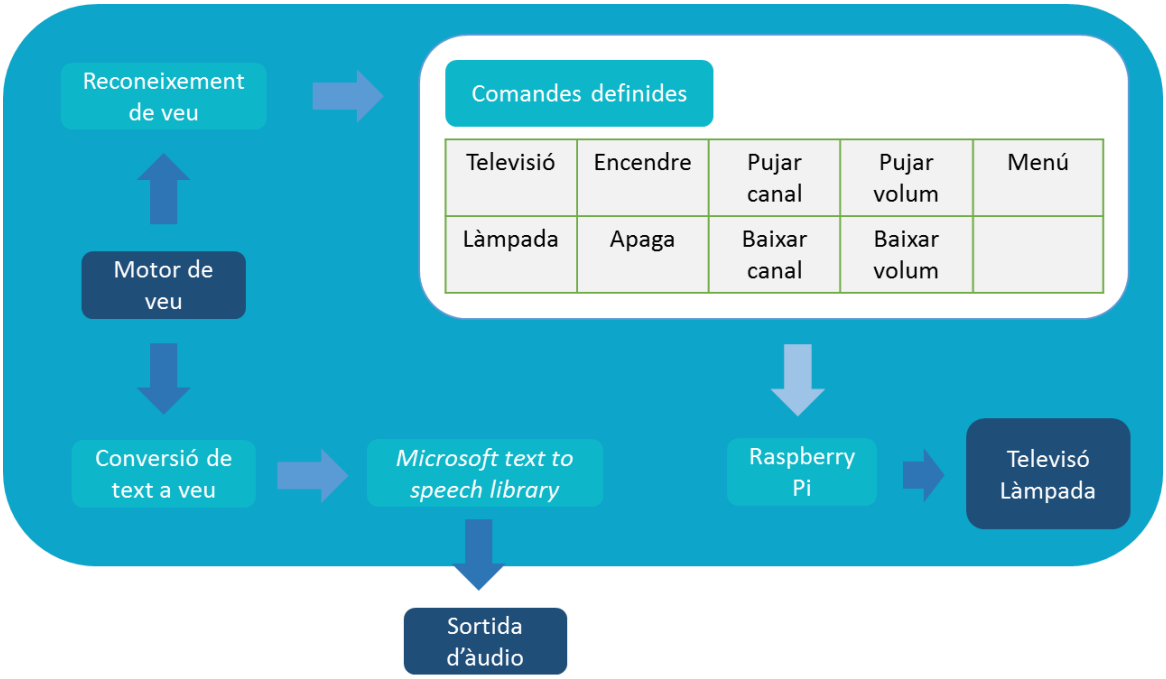

Figura 21: Esquemàtic de funcionament de l'aplicació de reconeixement de veu.

Pel que fa a el sistema de reconeixement de veu segueix el següents passos. Primer s'executa el mode escolta. A continuació es separa el flux d'àudio en segments. Un cop es té l'àudio segmentat, s'agafen aquests segments i es comparen amb els models acústics, lèxics i gramàtics de les llibreries corresponents al llenguatge escollit. Es converteix el àudio en un text. El *Mock Up* recorre aquest text i comprova quin és l'identificador per a saber quin dispositiu a de complir la ordre. En el prototip un exemple és "televisió". Finalment es mira si la resta de text que no forma part de d'identificació té assignat alguna comanda i si és així executa aquesta comanda en el dispositiu seleccionat.

El mode de funcionament de la conversió de text a veu és lleugerament diferent al anterior. Igual que en el cas anterior s'inicialitza el mode de conversió de text en veu. Segons el procés en el que està el dispositiu aquest envia una frase de text a l'aplicació. En el cas que es disposi de més d'un dispositiu s'afegeix davant del text anterior la paraula identificació del dispositiu que està enviant el text. Aquest text és processat per el motor de conversió de text a veu i l'expulsa per el sistema de sortida d'àudio que està per defecte en el ordinador on s'executa l'aplicació. Aquest procés el realitza per complet el *Mock Up* de manera que no és la Raspberry Pi la que dóna informació del estat dels dispositius. Senzillament quan la aplicació envia la comanda encén i no hi ha cap error de connexió, el pròpi *Mock Up* interpreta quin és el estat del dispositiu.

Per a entendre el funcionament de les aplicacions que es dissenyen a continuació, a continuació es mostra un diagrama de blocs del recorregut de cada programa.

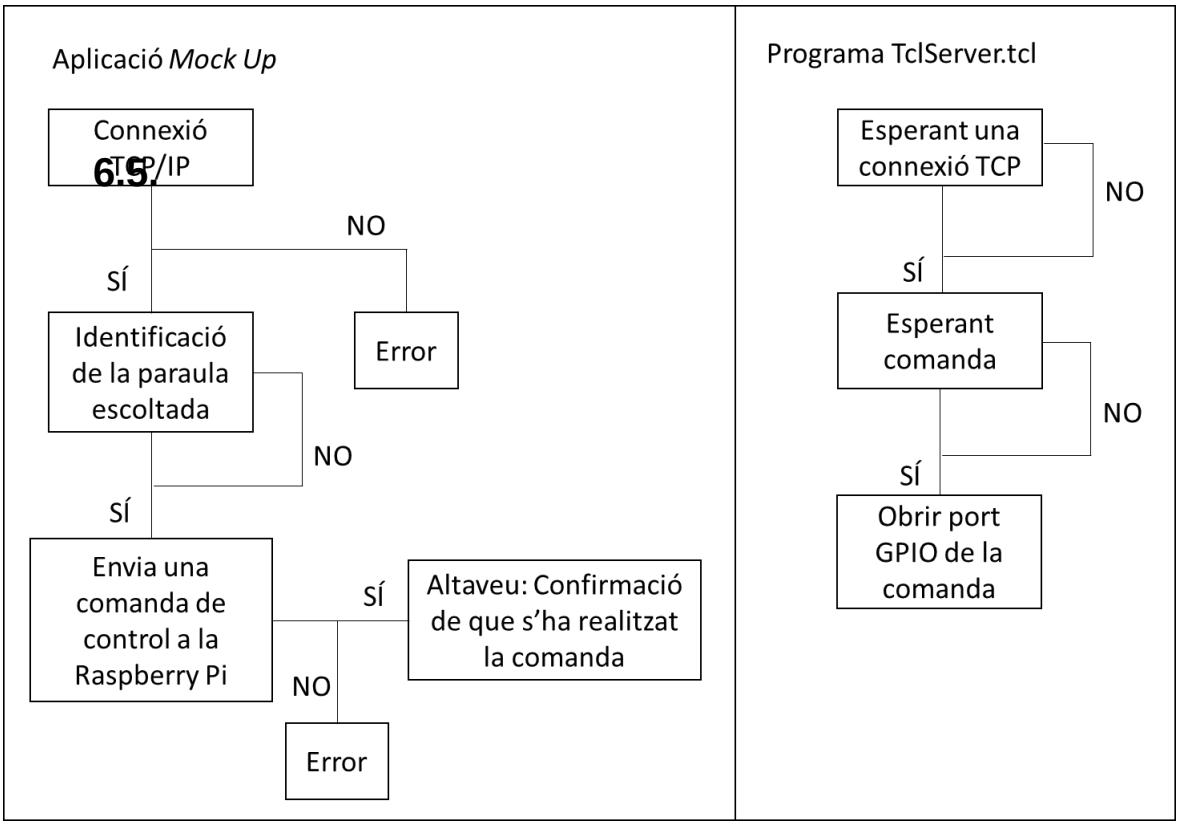

Figura 22: Diagrama de funcionament dels programes Mock Up i TclServer.tcl.

Programa servidor

Com s'ha comentat anteriorment per a que el sistema sigui funcional és necessari que hi hagi una plataforma on es pugui connectar la aplicació per enviar les ordres necessàries als dispositius. Aquesta plataforma s'anomena servidor. Un servidor és un programa informàtic capaç de emmagatzemar i executar les ordres de més d'un computador a la vegada mitjançant una connexió de xarxa.

En aquest projecte s'ha dissenyat un senzill servidor TCP, protocol de comunicació que s'utilitza per a les connexions dins de una mateixa xarxa que funcionen a través de internet. Aquest protocol funciona a través de un *socket*, que és un port de connexió no tangible on un ordinador es pot connectar mitjançant una adreça IP, número de identificació de cada dispositiu connectat a una mateixa xarxa de internet, i un numero de port que és la interfície per on es rep i s'envia la informació. Aquest servidor ha estat programat en TCL un llenguatge de programació que permet crear un servidor amb les seves funcions executables en el mateix paquet d'arxius de manera que des de un altre ordinador (client) es pot cridar directament a aquestes funcions amb la única condició d'haver establert anteriorment la connexió amb el servidor.

Per a crear el servidor el primer que s'ha fet ha sigut crear un arxiu amb extensió TCL. Concretament s'ha creat el TclServer.tcl arxiu en el qual s'ha construït el programa per establir connexió.

Inicialment es crida el paquet per executar arxius en format TCL i a la vegada es càrrega l'arxiu tclserverVR.so que conté les funcions que posteriorment es poden cridar mitjançant comandes des de el client.

```
package require Tclx

load ./tclserverVR.so
```

Un cop cridat tots els arxius necessaris, es procedeix a declarar les variables que s'utilitzaran a les següents funcions. Com es pot veure s'inicialitza el port amb un valor de 9104, número el qual s'ha de posar en el client per a crear la connexió. I també s'inicialitza una variable *timeout* que estableix el temps màxim que pot tardar en establir-se la

connexió, en cas de que una connexió amb aquest client respongués més tard de 10 segons, la connexió es perdria i sortiria un error.

```
global APPort
global APChan
global timeout_Connection

# Global Initialization
set APPort 9106
set timeout_Connection 10000
```

Un cop definit tot el necessari, es continua amb una subrutina, que s'encarrega de comprovar l'estat de la connexió client-servidor. En cas de que el client tanqui la connexió del seu port, el servidor imprimeix en pantalla el missatge "CLOSE!!!". Si per altre banda el client envia una comanda el servidor imprimeix en pantalla la comanda que ha rebut encara que no coincideixi amb cap de les funcions incloses al arxiu tclserverVR.so. Si realment la comanda no coincideix amb cap funció a continuació el servidor envia un missatge d'error.

```
proc  svcHandler {sock} {
  dup $sock stdout
  set comando [gets $sock]  ;# get the client packet
  if {[eof $sock]} {        ;# client gone or finished
     close $sock            ;# release the servers client channel
     puts stderr "CLOSE!!!"
  } else {
    puts stderr "info Command from client:$comando"
    set errorcode [catch $comando msg]
    if {$errorcode != 0} {
        puts stderr "error $errorcode Mensage $msg"
    }
  }
}
```

Seguidament es crea una altre subrutina anomenada "accept" que com indica el nom de la rutina es dedica a acceptar les connexions dels clients. Quan un client intenta connectar-se al *socket* d'aquest servidor, el propi servidor identifica la adreça IP del client i la imprimeix per pantalla juntament amb el missatge "Accepted connection". A part un cop establerta la connexió la rutina crida a la funció descrita anteriorment per comprovar l'estat de la connexió i si rep algun missatge del client.

```
proc accept {sock addr port} {
  puts $sock "$addr:$port, You are connected to application AP."
  flush $sock
  fconfigure $sock -buffering line -blocking 0
  fileevent $sock readable [list svcHandler $sock]
  dup $sock stdout
  puts stderr "info: Accepted connection from $addr"
}
```

Finalment es crea la rutina principal del servidor el qual està en espera mentre no rep una petició de connexió, però quan rep una petició, crida la subrutina "accept" que s'ha explicat anteriorment. Quan una de les subrutines finalitzen com pot ser el cas de que es tanqui una connexió, el programa torna a la rutina principal per esperar una nova connexió d'algun client.

```
set errorcode [catch { set APChan [socket -server accept $APPort]} msg]
if {$errorcode != 0} {
    puts stderr "error $errorcode Mensage $msg"
} else {
    puts stderr "info server started and waiting for connections..."
    vwait forever
}
```

Aquest codi és el que quan s'executa l'arxiu genera un servidor que es capaç de enviar i rebre missatges i interpretar-los. Però per això es necessari disposar de les funcions a executar que com s'ha comentat està en el arxiu tclserverVR.so que prové de dos arxius compilats en un únic arxiu. Per a fer la compilació es disposa d'un arxiu "Makefile" que incorpora a dintre totes les comandes de compilació necessàries per a generar el fitxer amb extensió .so que ens interessa. Per a poder executar el "Makefile" es necessari disposar de dos fitxers, un escrit en C++ i l'altre on només s'han de llistar els noms de les diferents funcions, amb extensió .i (del anglès input).

El primer fitxer escrit en llenguatge C++, anomenat tclserverVR.cpp consisteix en el següent. És on s'escriuen les funcions que després el client vol executar. En el cas que ens ateny les funcions que s'han afegit al fitxer ha estat la funció encendre i la funció apaga.

La primera funció s'encarrega de precisament fixar la direcció i el valor de un dels ports de la Raspberry Pi (GPIO), concretament en aquest cas es fixa el port numero 17 com a port de sortida i amb un valor de sortida de 0 Volts. Això ho fa modificant els fitxers interns de la Raspberry Pi que controlen aquests ports i que es troben en el camí root/sys/class/gpio.

Per a executar aquest programa senzillament cal introduir en el *prompt* del terminal de la Raspberry Pi la següent línia de text: >>> ./TclServer.tcl .

```
void encendre(){
    string gpionum = "17";
    string export_str = "/sys/class/gpio/export";
    string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
    string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
    ofstream exportgpio(export_str.c_str());
    ofstream setdirgpio(setdir_str.c_str());
    ofstream setvalgpio(setval_str.c_str());

    if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
        cout << "OPEATION FAILED: Unable to export GPIO";
    }
    exportgpio << gpionum;
    setdirgpio << "out";
    setvalgpio << "0";
    exportgpio.close();
    setdirgpio.close();
    setvalgpio.close();
}
```

En l'altre funció que s'anomena apaga tenim un codi igual que l'anterior però en comptes de fixar el valor de sortida del port a 0 Volts el fixa a 3,3 Volts que és el valor màxim al que poden treballar aquests ports.

```
void apaga(){
    string gpionum = "17";
    string export_str = "/sys/class/gpio/export";
    string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
    string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
    ofstream exportgpio(export_str.c_str());
    ofstream setdirgpio(setdir_str.c_str());
    ofstream setvalgpio(setval_str.c_str());

    if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
        cout << "OPEATION FAILED: Unable to export GPIO";
    }
    exportgpio << gpionum;
    setdirgpio << "out";
    setvalgpio << "1";
    exportgpio.close();
    setdirgpio.close();
    setvalgpio.close();
}
```

Per a executar aquest programa senzillament cal introduir en el *prompt* del terminal de la Raspberry Pi la següent línia de text: >>> ./TclServer.tcl .

6.6. Aplicació de reconeixement de veu

Un cop fets tots els elements complementaris, es comença amb el nucli del projecte que és la pròpia aplicació de reconeixement de veu, el *Mock Up*. Aquesta aplicació està programada en llenguatge Visual C# un llenguatge molt senzill i molt fàcil d'entendre. A més a més també s'utilitzen varies funcions de la llibreria Microsoft.speech que es pot descarregar de la següent pagina web de forma gratuïta, https://www.microsoft.com/en-us/download/details.aspx?id=11670 .

Un cop es disposa d'aquesta llibreria s'ha de incloure tant en el projecte de Visual estudio, com en el codi de l'aplicació com es veu en la imatge següent. El primer pas ja s'ha explicat quan s'ha parlat de les referències en el punt 5.2., però per afegir-ho a el codi encara és més senzill, només s'han d'afegir les següents línies al inici de tot.

```
using Microsoft.Speech.Recognition;
using Microsoft.Speech.Synthesis;
```

Aquest codi està dins d'un fitxer, el fitxer principal, anomenat Form1.cs a on està el nucli del programa. Seguidament un cop iniciada la llibreria de Microsoft tant de reconeixement de veu com de síntesis de text a veu també s'afegeixen les llibreries que permeten establir una connexió TCP amb el servidor que s'ha explicat al punt anterior a través d'un *socket*.

```
using System.Threading;
using System.Net;
using System.Net.Sockets;
using System.IO;
```

Definides ja les llibreries característiques d'aquest projecte es comença ja amb el programa principal. El llenguatge C# és un llenguatge centrat en la creació de classes. Aquestes classes tenen funcions internes anomenades *void*, de funcions *void* n'hi ha de diferents classes però en aquesta aplicació només s'utilitzen funcions publiques (*private void*) que permeten cridar-les només dins de la classe on han estat creades.

Per tant el primer que és fa es crear la classe principal anomenada igual que l'arxiu, es a dir Form1. Aquesta classe va lligada a una interfície gràfica que té el mateix nom, ja que es tracta d'un projecte amb suport gràfic.

```
public partial class Form1 : Form
{
```

Dins d'aquesta classe el primer que es fa és inicialitzar les variables. S'inicialitzen 3 variables importants, primer la "cultureInfoConfig" que conté el nom informàtic del llenguatge seleccionat, després el "syn" a on s'inicialitza el sintetitzador de veu que es capaç de transformar el text a veu i finalment la variable "s" que correspon al *socket* que s'utilitzarà per connectar-se al servidor.

```
string cultureInfoConfig;
SpeechSynthesizer syn;
Socket s;
```

Un cop definits els elements a la classe principal, dins d'aquesta es crea una subrutina que fa referència a la selecció del llenguatge. Aquesta subrutina conté un petit diccionari amb les paraules o frases que ha de ser capaç de reconèixer el motor de reconeixement de veu i també de les paraules o frases que es vol que el programa enviï via àudio obtenint la bidireccionalitat de l'aplicació. Segons l'idioma seleccionat per a l'usuari s'escull un o altre diccionari i la variable "cultureInfoConfig" ja definida obté el valor del nom de l'idioma que l'ordinador es capaç d'entendre.

```
switch (langId)
{
    case 0:

        speakCommands[0] = "You can speak";
        speakCommands[1] = "Waiting";
        speakCommands[2] = "You said on";
        speakCommands[3] = "You said off";
        speakCommands[4] = "Connected";
        speakCommands[5] = "Disconected";
        speakCommands[6] = "Couldn't connect to the device";

        voiceCommands[0] = "on";
        voiceCommands[1] = "off";

        cultureInfoConfig = "en-US";

        break;

    case 1:
        speakCommands[0] = "Estoy escuchando";
        speakCommands[1] = "Esperando";
        speakCommands[2] = "Has dicho encender";
        speakCommands[3] = "Has dicho apagar";
        speakCommands[4] = "Conectado";
        speakCommands[5] = "Desconectado";
        speakCommands[6] = "No se pudo conectar con el dispositivo";


        voiceCommands[0] = "encender";
        voiceCommands[1] = "apagar";

        cultureInfoConfig = "es-ES";

        break;
```

A continuació el programa inicialitza tots els elements necessaris per a que funcioni el reconeixement de veu i el sintetitzador de text a veu. Primer es configura el llenguatge del sistema utilitzant el identificador definit en el diccionari, de manera que quan comencin a funcionar els motors de veu, agafin com a idioma aquest i busquin dins de les llibreries instal·lades tant la gramàtica com la pronunciació com el lèxic d'aquesta llengua.

```
// Set up culture info
Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo(cultureInfoConfig);
Thread.CurrentThread.CurrentUICulture = new System.Globalization.CultureInfo(cultureInfoConfig);
System.Globalization.CultureInfo ci = new System.Globalization.CultureInfo(cultureInfoConfig);
```

Seguidament s'inicialitza el sintetitzador de veu, per a fer això es mira en el computador quins són els llenguatges instal·lats i a partir d'aquí es compara si alguna part del nom dels

llenguatges coincideix amb l'identificador de llengua "cultureInfoConfig" i el llenguatge amb el que coincideix es selecciona i utilitzant la funció "selectVoice" es tria quina veu i en quin llenguatge es vol utilitzar. A més a més també es selecciona que tot l'àudio que expulsi la aplicació la reprodueix-hi a través del sistema de àudio predeterminat.

```
//Initialize the synthesizer
syn = new SpeechSynthesizer();
syn.SetOutputToDefaultAudioDevice();
foreach (InstalledVoice voice in syn.GetInstalledVoices())
{
    if (voice.VoiceInfo.Name.Substring(46, 5) == cultureInfoConfig)
    {
        syn.SelectVoice(voice.VoiceInfo.Name);
    }
}
```

Finalment s'inicialitza el motor de reconeixement de veu, només cal passar-li l'idioma en que es vol que s'executi i com en el sintetitzador quin és el dispositiu d'entrada de veu, que en aquest cas també és el predeterminat del ordinador a on s'executi. Un cop ja definit el motor també es crea un esdeveniment on es posaran per a cada paraula distingida quina acció ha de realitzar la aplicació.

```
//Initialize the recognition engine
SpeechRecognitionEngine sre = new SpeechRecognitionEngine(ci);
sre.SetInputToDefaultAudioDevice();
sre.SpeechRecognized += new EventHandler<SpeechRecognizedEventArgs>(sre_SpeechRecognized);
```

Tot i que ja estan iniciats tots els elements necessaris per a començar el reconeixement de les paraules obtingudes per el motor gràfic falta introduir en el motor els diccionaris definits al inici d'aquesta classe. Això es fa creant un element "Choices" a on s'afegeix un a un totes les paraules del apartat "voiceCommands" del diccionari. Un cop es tenen totes les paraules dins d'aquest element "Choices" s'afegeixen en un constructor de gramàtica (GrammarBuilder) i posteriorment en una categoria de gramàtica (Grammar), finalment aquesta gramàtica s'introdueix a dins del motor de reconeixement de veu utilitzant la funció "LoadGrmmarAsync".

```
//Get culture info
Choices actionsChoices = new Choices();
for (int i = 0; i < voice_commands; i++)
{
    actionsChoices.Add(voiceCommands[i]);
}
Console.WriteLine(cultureInfoConfig);

GrammarBuilder actionsGrammarBuilder = new GrammarBuilder();
actionsGrammarBuilder.Append(actionsChoices);
Grammar WordsGrammar = new Grammar(actionsGrammarBuilder);
sre.LoadGrammarAsync(WordsGrammar);
```

Un cop tot està configurant només manca que el programa es posi a escoltar i mirar de reconèixer tot el que entra per el sistema d'entrada d'àudio. Això es fa amb la funció *RecognizeAsync*, que quan rep àudio per el canal d'entrada escolta de manera asíncrona fins que considera que l'usuari ha parat de parlar degut a un silenci llarg o que ha passat més d'un cert temps internament establert. Tot aquest arxiu de àudio el processa el motor de reconeixement de veu per transformar-lo en text.

```
//Begin recognition
sre.RecognizeAsync(RecognizeMode.Multiple);
```

La funció *RecognizeAsync* un cop té un àudio que es capaç de processar envia la informació en forma de una classe anomenada *Result* dins del esdeveniment creat durant la inicialització del motor de reconeixement de veu, aquesta classe conté varies funcions específiques dues de les quals s'utilitzen en aquest programa. La primera és la funció *Result*. Text que retorna en forma de text la paraula o frase que ha entès el motor i que ha de coincidir amb una paraula del diccionari ja que el motor quan s'executa compara el fitxer d'àudio amb totes les paraules del diccionari per veure a quina s'assembla més. L'altre funció és la *Result.Confidence* que et diu amb quin percentatge la paraula que ha dit l'usuari s'assembla a la paraula que retorna el motor de reconeixement de veu, aquest percentatge pren un valor del 0.00 al 1.00. Per a obtenir certa robustesa es fixa un valor llindar a partir del valor aquest de confiança, de manera que només es considera com a bona la paraula que té un valor superior a aquest llindar. En el cas que sigui així el programa mira a quina paraula correspon el que ha dit l'usuari i executa una funció o una altre, en aquest cas la funció on() o la funció off().

```
//Recognition event
1 reference
private void sre_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
    if (e.Result.Confidence > 0.7)
    {
        if (e.Result.Text == "encender")
        {
            on();
        }
        if (e.Result.Text == "apagar")
        {
            off();
        }
    }
}
```

Les funcions a les que fan referència en el tros de codi anterior estan descrites posteriorment. Aquestes funcions el que fan es agafar la comanda adequada per al servidor TCL, codificar aquesta comanda en format ACSII (Codi estàndard Americà per a l'intercanvi de informació), que converteix el text en una sèrie de 7 bits per representar els caràcters que s'envien al servidor el qual descodifica la informació i la executa.

```
//TCL
1 reference
void on()
{
    byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [encendre]" + '\n');
    s.Send(sendBytes);
}
1 reference
void off()
{
    byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [apaga]" + '\n');
    s.Send(sendBytes);
}
```

L'últim tros de codi d'aquest script es basa en la creació del client per a crear i establir la connexió d'aquest amb el servidor. S'han creat dos botons per a fer tant la connexió com la desconnexió de es dues interfícies.

Començant per el botó "Enable" el qual només està actiu quan el programa no ha establert connexió amb cap altre dispositiu i es desactiva quan s'ha establert una connexió. La opció de clicar el botó el que fa es crear el client i el *socket* a través del qual el client es pot comunicar. Com en volem connectar al servidor TCL creat a la Rapberry Pi, hem de posar el mateix número de port i la seva adreça IP. Com que aquesta última pot canviar segons la

xarxa a la que es connecti el dispositiu i també de forma aleatòria dins de la mateixa xarxa, doncs no s'ha fixat cap adreça IP sinó que s'ha posat un quadre de text a on es pot escriure la adreça adequada i quan es fa *click* al botó "Enable" crea el *socket* amb la adreça IP escrita en el quadre de text. Un cop connectat amb el servidor el sintetitzador de veu reporta a la sortida d'àudio si la connexió ha tingut èxit o si hi ha hagut algun error.

```
//EnableButton
1 reference
private void EnableButton_Click(object sender, EventArgs e)
{
    s = new Socket(AddressFamily.InterNetwork,
        SocketType.Stream,
        ProtocolType.Tcp);
    IPAddress[] IPs = Dns.GetHostAddresses(textBox1.Text);
    try
    {

        TcpClient tcpclnt = new TcpClient();
        Console.WriteLine("Connecting.....");
        s.Connect(IPs[0], 9106);

        // use the ipaddress as in the server program

    }
    catch (Exception)
    {
        syn.Speak(speakCommands[6]);
        return;
    }
    EnableButton.Enabled = false;
    DisableButton.Enabled = true;
    syn.Speak(speakCommands[4]);

}
```

També es disposa d'un botó "Disable" que s'encarrega de tancar el *socket* del client de manera que es tanca la connexió però només per part d'aquest programa. En el servidor el *socket* segueix estant funcional i per tant es pot establir una nova connexió des de aquesta aplicació o programa o des de qualsevol altre que funcioni amb protocol TCP.

```
//DisableButton
1 reference
private void DisableButton_Click(object sender, EventArgs e)
{
    s.Disconnect(false);

    EnableButton.Enabled = true;
    DisableButton.Enabled = false;
    syn.Speak(speakCommands[5]);
}
```

6.7. Robustesa

El principal problema del sistema de reconeixement de veu és que el motor no reconegui bé la paraula del usuari i per tant realitzi una acció que no és la voluntat del usuari o per contra que no entengui la paraula i per tant no faci res. Per evitar això es disposa d'un valor de confiança entre la coincidència de la paraula dita respecte la paraula escrita. Mitjançant aquest valor es pot aconseguir obtenir un sistema robust davant de varies variables externes com pot ser el soroll ambiental, l'ús d'un idioma no usual per al usuari o senzillament que sigui usable per a tots els usuaris independentment del accent o el tipus de veu.

Primer de tot s'ha mirat quin és quines variables poden afectar a la precisió del sistema de reconeixement de veu, s'han localitzat 4 variables externes que poden tenir un impacte en aquest sistema. Aquestes 4 variables són el tipus de micròfon, el soroll ambiental, el to de veu i el llenguatge.

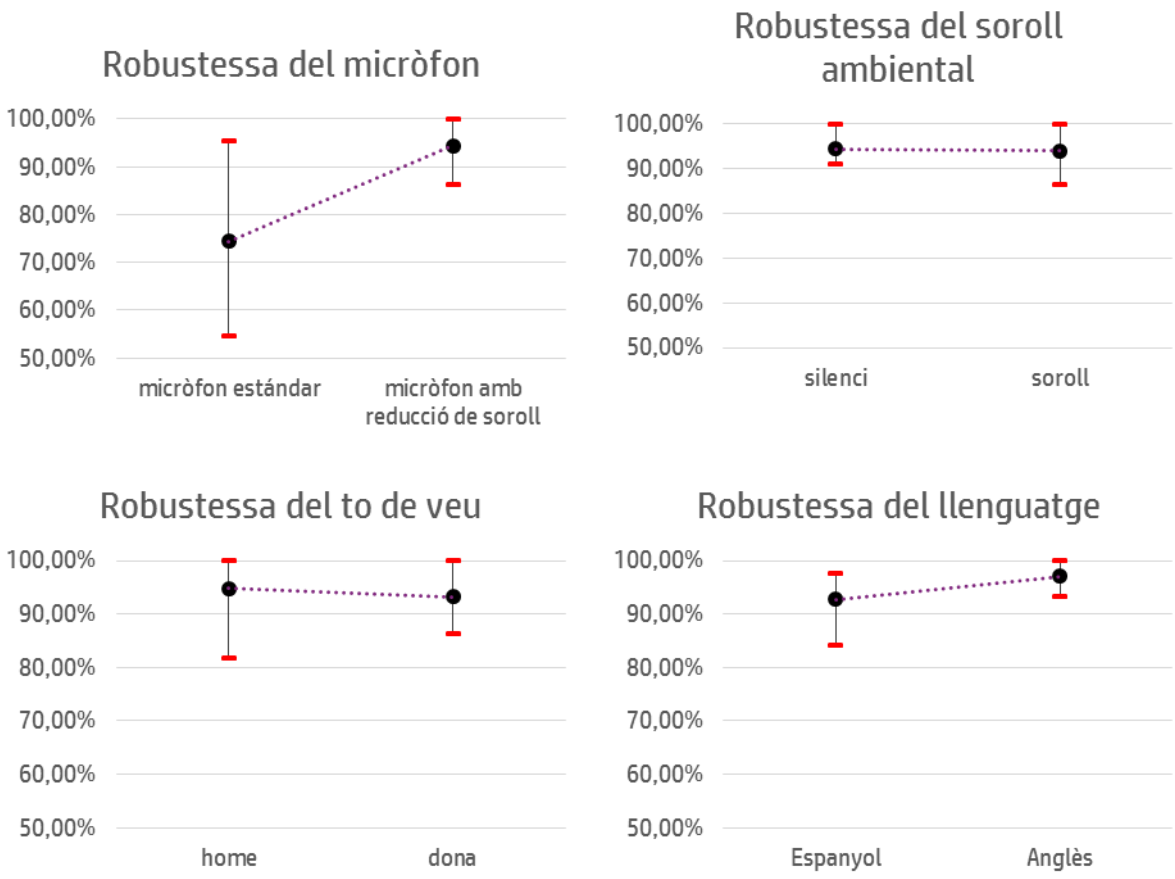

Figura 23: Test de robustesa de les variables que afecten al sistema.

S'ha realitzat un test (figura anterior) on s'han comparat dins les variables, les possibles configuracions. Aquest test

En el primer cas tenim la comparació entre un micròfon estàndard i un micròfon amb cancel·lació de soroll activa, veien que clarament hi ha gran diferencia entre les diferents configuracions, tant en variabilitat com en mitjana. Aquest efecte s'estudiarà a continuació detalladament en la selecció del micròfon.

En segon lloc s'han fixat la resta de variables i s'ha estudiat l'efecte del soroll a la aplicació de reconeixement de veu respecte a una zona amb sense soroll, obtenint un resultat bastant semblant en ambdues configuracions, per tant es pot decidir que la variable soroll no afecta a la precisió o robustesa de la aplicació de reconeixement de veu.

En tercer lloc s'ha comprovat si el tipus de veu representa un inconvenient per a que varis usuaris facin ús del *Mock Up* sense necessitat d'un aprenentatge anterior. Per a estudiar dos tipus de veus diferent s'ha fet la prova amb una veu de home respecte una veu de dona mirant el percentatge d'encerts de l'aplicació. S'ha observat que la variabilitat dels homes és menor que el de les dones, sense cap valor representatiu ja que no s'han fet suficients experiments. Es descarta la variable veu com a variable soroll del sistema.

Finalment i en quart lloc s'han comparat si el *Mock Up* funciona igual de bé utilitzant dos llenguatges diferents (en aquest cas s'han agafat el anglès americà i l'espanyol com a llenguatges a testejar). El usuari que ha realitzat el test era natiu americà, per això considerarem que la llengua mare és el aquesta i la estrangera és el espanyol. En el test es pot veure com es millor el percentatge d'encerts de la llengua mare respecte la llengua estrangera, cosa normal, que només certifica que s'ha de tenir en compte a l'hora de configurar la aplicació quina llengua s'està utilitzant. Aquesta característica no significa un problema per a obtenir una aplicació de reconeixement robusta.

Per a aconseguir aquesta robustesa també s'ha mirat quines eren les necessitats dels cascs de àudio a utilitzar per poder seleccionar segons les especificacions. Llavors dins del catàleg de la mateixa marca de dispositius de àudio s'han seleccionat possibles candidats i s'han descartat els que no donaven tanta funcionalitat al projecte, obtenint com a resultat un únic dispositiu que és el que s'ha utilitzat per a els test de comprovació del sistema.

Dins del catàleg de cascs de àudio que es disposava s'ha fet una taula comparativa seleccionant el que més s'adaptava a les necessitats del projecte. Ja que disposa de les tecnologies comentades anteriorment i dins dels que complien les especificacions és el més senzill i fàcil de usar. **Tots els tests de robustesa s'han realitzat amb aquest dispositiu d'àudio**.

	Jabra Supreme UC MS	Plantronics Voyager Legend (Mono) B235-M	Plantronics Savi W420-M	Plantronics Savi W720
				
Price [€]	93.08	99	145.6	189
Aural	On ear	On ear	Over the head (binaural)	Over the head (binaural)
Wireless tech.	Bluetooth	Bluetooth	Radio frequency (DECT)	Radio frequency (DECT)
Distance [m]	10	10	120	120
Connectivity	PC (USB-Dongle)	PC (USB-Dongle)	PC (USB-Dongle)	PC (USB-Cable) Phone (Cable) Mobile (Bluetooth)
Active noise reduction	Yes	Yes	Yes	Yes
Mute available	Yes	Yes	Yes	Yes
Converation time [h]	6	7	9	9

Figura 24: Taula comparativa dels diferents dispositius d'àudio estudiats.

El primer requeriments d'aquest dispositiu és que disposi tant de altaveu com de micròfon ja que el *Mock Up* és bidireccional. Un altre detall important és que pugui utilitzar-se a una distancia gran del ordinador principal per a poder controlar els dispositius des de qualsevol lloc de la casa. Per tant a de tenir un bon funcionament a uns 10-20m de distància. Per aconseguir això el dispositiu d'àudio ha de disposar de tecnologia de radiofreqüència així la transmissió a aquestes distancies és més neta i per tant el reconeixement és més precís. La última característica important és que el micròfon incorpori la tecnologia de cancel·lació de soroll, de manera que el propi micròfon elimina el soroll exterior del ambient i així s'estalvia un post processat del arxiu de àudio que sinó seria necessari.

Per veure com afecta el ús d'aquestes tecnologies a l'aplicació de reconeixement de veu, s'ha fet un test utilitzant unes 30 paraules i un micròfon amb cancel·lació de soroll comparat amb el mateix test utilitzant un micròfon convencional sense cap característica especial. El resultat en percentatge d'encerts és el següent:

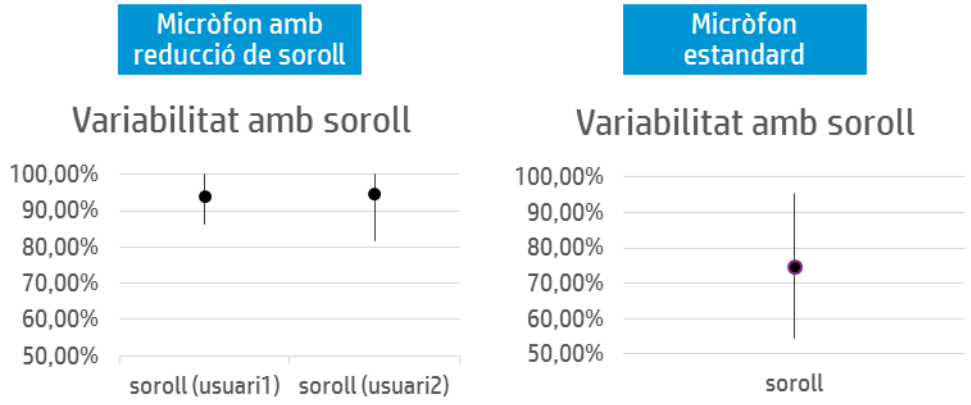

Figura 25: Comparativa de un micròfon estàndard respecte un micròfon amb cancel·lació de soroll.

Com es pot observar en els gràfics anteriors, utilitzant el micròfon amb les característiques recomanades s'obté més del 90% d'encert en un entorn amb soroll i amb poca variabilitat. En canvi amb un micròfon estàndard el percentatge d'encert és menor del 80% i amb molta variabilitat, per tant clarament el software de reconeixement de veu és més robust utilitzant el hardware adequat.

L'altre eina important que permet obtenir una millora en la precisió del sistema de reconeixement de veu i així evitar errors és un valor intern del sotware de l'aplicació, que defineix el percentatge de semblança entre la paraula interpretada per l'àudio i la paraula escrita en text de la biblioteca de paraules. Aquest valor s'anomena *treshold* i s'introdueix dintre de la funció *RecognizedPhrase.Result.Confidence* en percentatge sobre 1.

Sobre aquest valor s'ha fet un estudi per veure quin és el més òptim utilitzant paraules usualment utilitzades en el àmbit domòtic com poden ser encendre, apagar, cancel·lar, parar, comença, encendre llum, apagar llum, etc... fins a un total de unes 30 paraules o frases diferents. D'aquest estudi s'han obtingut els resultats de la següent figura.

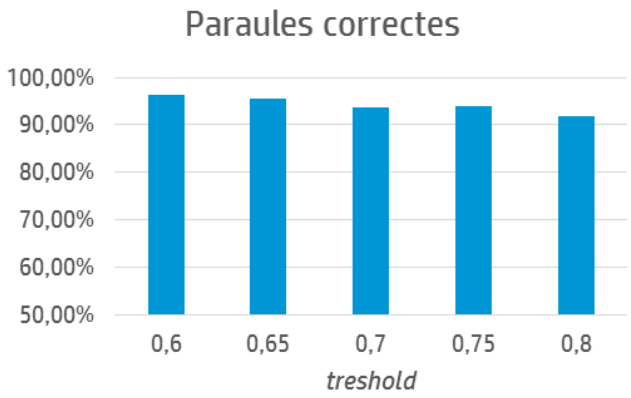

Figura 26: Test de comandes correctes respecte el treshold.

Com es pot veure en les figures anteriors l'estudi només contempla el rang de valors del *treshold*[12] entre el 60% fins al 80% ja que un valor més elevat donaria la majoria de paraules incorrectes i en canvi un valor menor donaria com a vàlides paraules totalment diferents de les que composen la biblioteca.

El nombre de paraules correctes en funció del valor del *treshold* no varia més que la variabilitat del àudio. Llavors per poder escollir un valor de *treshold* s'ha dissenyat un model lineal tenint en compte les paraules que s'han fallat a cada test i les paraules del test que s'han confós, es a dir, l'usuari volia dir una paraula i el sistema de reconeixement n'ha reconegut com a bona una altre.

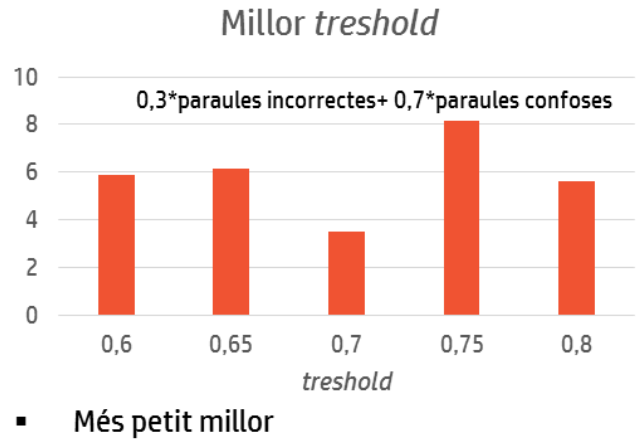

Figura 27: Model lineal del test per a determinar el millor valor del treshold.

[12] *Treshold* és el valor llindar a partir del qual la aplicación considera com a bona la paraula

El model segueix la següent equació, 0,3*paraules incorrectes + 0,7*paraules confoses. D'aquest model s'ha vist que el valor òptim del *treshold* és el 0,7 i per tant s'escollirà aquest valor com a predeterminat en la aplicació. Tot i això es posarà una pestanya per a poder canviar el valor segons convingui depenent si s'utilitza una llengua nativa o una llengua estrangera.

6.7.1. Connexions de hardware

Per a demostrar la utilitat de una aplicació de reconeixement de veu personalitzada s'ha construït un prototip. En aquest cas el projecte s'ha centrat en demostrar que la aplicació funciona utilitzant un llum i una televisió establint varies connexions entre la Raspberry Pi i els dispositius amb varis elements electrònics necessaris enmig. A continuació es mostra quin ha estat el esquema de connexions entre la Raspberry Pi i el llum i la televisió. Utilitzant aquestes connexions i els programes anteriors es pot controlar un televisor i una làmpada a través de la veu.

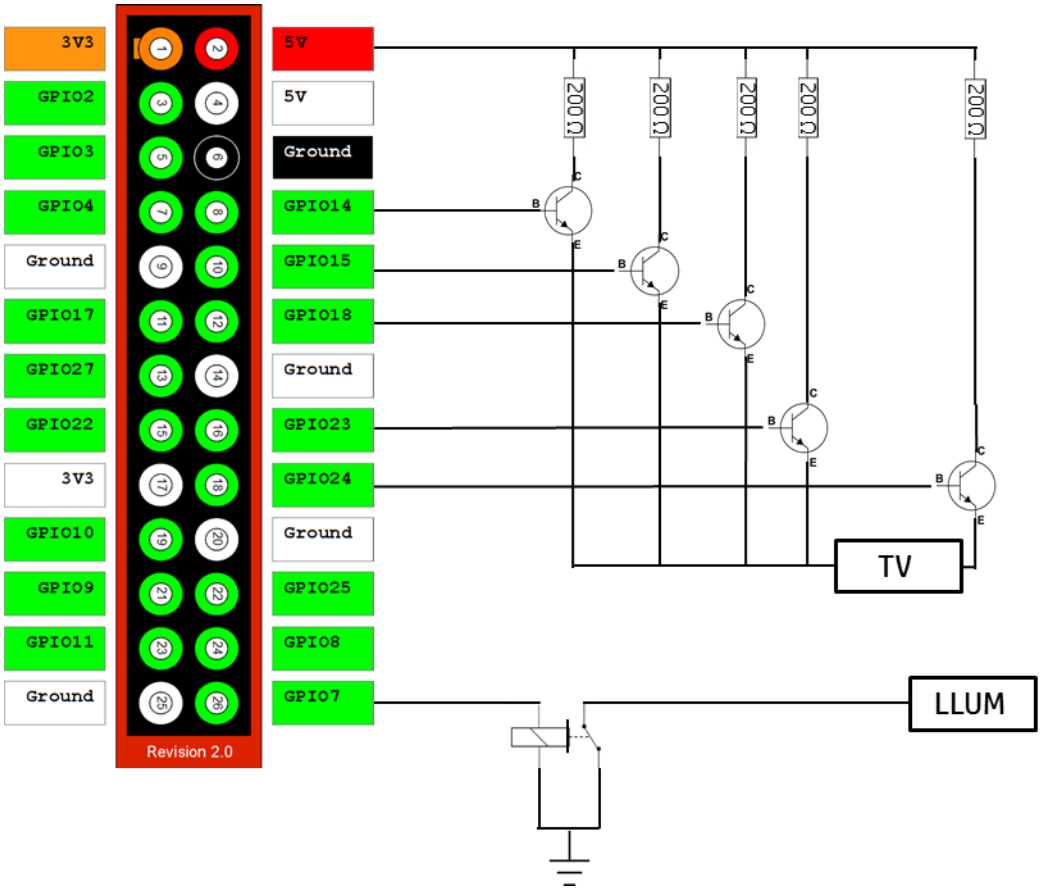

Figura 28: Diagrama de connexions del circuit elèctric del prototip.

Per a permetre el control de encès i apagat de les funcions de la televisió s'han utilitzat transistors substituint els polsadors de existents en la televisió. La diferència entre un polsador i un transistor es senzillament que un tanca un circuit elèctric de forma mecànica (en el polsador l'usuari prem el botó per a tancar el circuit) i en canvi el transistor es controla elèctricament (la Raspberry Pi dóna una ordre a un dels seus pins de que circuli corrent, aquest pin està connectat a la base del transistor i quan es dóna corrent a la base del transistor llavors es tanca el circuit entre el col·lector i el emissor).

Per a controlar la làmpada s'ha utilitzat un altre sistema que utilitza un relé enlloc de transistors, ja que els relés són dispositius mecànics capaços de treballar a tensions i corrents més elevats que els transistors. Tot i això la funció és la mateixa que els transistors, la funció de interruptor, treballant a 230 V.

.

7. Pressupost

En aquest projecte per a fer el pressupost, s'han dividit les despeses que s'han produït durant el recorregut del projecte en dos apartats, unes son les despeses referents a els recursos humans i les altres despeses fent referència a les despeses materials necessàries per al projecte

7.1. Pressupost de recursos humans

S'ha definit una taula on es contemplen les despeses de recursos humans referents als costs generats per la participació de les persones en aquest projecte. Tenint en compte que la duració del projecte ha estat de aproximadament de 6 mesos.

Cost de recursos humans			
Descripció	**Temps [h]**	**Preu [€/h]**	**Cost [€]**
1 Enginyer en pràctiques	**300**	**7**	**2100**
Documentació i aprenentatge del Visual Studio	60		
Desenvolupament de la aplicació	100		
Tests de robustesa	60		
Selecció del hardware	20		
Disseny del prototip	60		
Cost de S.S. estudiant en pràctiques			
1 Enginyer superior	**30**	**30**	**900**
Cost de S.S. enginyer superior(33%)			297
Total			3297

Taula 2: Pressupost dels recursos humans del projecte.

7.2. Pressupost recursos materials

En aquest apartat es desglossarà les despeses del recursos materials utilitzats durant el projecte, dividint-los en dos categories diferents, les llicències de software i els components físics utilitzats tant per a crear la aplicació com per a construir el prototip del projecte. Per a avaluar les despeses dels elements que disposen duna amortització més llarga que el temps del projecte s'ha considerat només el cost dels 6 mesos que ha durat el projecte.

Cost de recursos materials					
Descripció	**Cost inicial [€]**	**Temps d'amortització [anys]**	**Amortització anual [€/any]**	**Duració [anys]**	**Amortització total [€]**
Software					**97,5**
Llicència Visual Studio 2013	650	4	163	0,5	81,3
Llicència sistema operatiu Raspbian	0	4	0	0,5	0
Llicència Microsoft Office 2013	130	4	32,5	0,5	16,3
Hardware					**479**
Televisió	200				200
Làmpada	30				30
Ordinador portàtil	1000	5	200	0,5	100
Raspberry Pi 2	60				60
Cablejat	5				5
Resistències	2				2
Transistors	2				2
Impressió	80				80
Total					**576,5**

Taula 3: Pressupost dels recursos materials del projecte.

7.3. Pressupost total del projecte

Finalment un cop estudiats quins són les despeses particulars i detallades del projecte a la següent taula es fa un balanç final del projecte, tenint en compte els impostos que el contractista ha de abonar en la realització de qualsevol projecte. Aquests impostos depenen del lloc de facturació, en aquest cas España.

Cost de recursos humans	
Descripció	**Cost [€]**
Cost de recursos humans	3297
Cost de recursos materials	576,5
PRESUPOST TOTAL SENSE IVA	**3873,5**
IVA (21%)	**813,435**
PRESSUPOST TOTAL IVA INCLUIT	**4686,935**

Taula 4: Pressupost total del projecte, amb el IVA inclòs.

En resum el pressupost total del projecte és de un valor de **4388 €**.

8. Impacte Mediambiental

Al tractar-se d'un projecte centrat en una aplicació informàtica l'impacte mediambiental és escàs i es podria considerar gairebé nul. Ja que no es produeix cap consum energètic afegit més enllà de tenir unes hores més engegat el ordinador centrar per a poder tenir constantment en funcionament la aplicació.

Tot i això els dispositius que s'utilitzen en el prototip, la televisió i la làmpada tenen un impacte mediambiental. Aquests dos dispositius en conjunt tenen un consum aproximat entre 200-250 W d'energia elèctrica. També hi ha altres dispositius com la Raspberry Pi o el ordinador que s'ha estat utilitzant durant la programació de l'aplicació i que també són necessaris per a controlar el prototip. Aquests dos dispositius tenen un consum energètic de 3 i 100 Watts respectivament. Com es pot veure el consum de la Raspberry Pi es menyspreable davant del consum de un ordinador.

Suposant que utilitzar un Watt d'energia elèctrica suposa una emissió de 0,7 grams de CO_2 a la hora, resulta que l'impacte mediambiental en unitats de gas contaminant del projecte és del voltant de 245 g de CO_2 per hora. Considerant que un cotxe genera aproximadament 100 g/km i que la mitjana de velocitat és de 50 km/h, un cotxe arriba a consumir 5000 g de CO_2 per hora. Respecte aquest valor el impacte mediambiental del projecte és relativament petit.

Per tant concloure que a nivell mediambiental aquesta aplicació no suposa cap avantatge, però tampoc implica un consum afegit. Només es necessita energia en el cas que s'utilitzi el televisor o la làmpada, cost independent de si s'està fent ús de la aplicació de reconeixement de veu. En alguns casos concrets fins i tot pot suposar un estalvi energètic ja que es pot consultar l'estat de la làmpada o de la televisió sense estar en la mateixa sala d'un mateix domicili i apagar-la des de allà.

9. Planificació

Task Name	Duration	Start	Finish
Detecció de problema	7 days	Mon 16/02/15	Tue 24/02/15
Solució al problema	10 days	Fri 20/02/15	Thu 05/03/15
Investigació d'aplicacions del reconeixement de veu	5 days	Mon 02/03/15	Fri 06/03/15
Investigació de tecnologies de reconeixement de veu	5 days	Mon 02/03/15	Fri 06/03/15
Documentació de llibreries de Microsoft Speech Platform	8 days	Fri 06/03/15	Tue 17/03/15
Disseny de l'aplicació	30 days	Tue 17/03/15	Mon 27/04/15
Construcció d'un servidor tcl	15 days	Mon 06/04/15	Fri 24/04/15
Desenvolupament de funcions	4 days	Mon 20/04/15	Thu 23/04/15
Configuració de la Raspberry Pi	8 days	Mon 11/05/15	Wed 20/05/15
Desenvolupament del prototip	18 days	Mon 13/04/15	Wed 06/05/15
Pressupost	3 days	Mon 18/05/15	Wed 20/05/15
Consclusió	6 days	Mon 25/05/15	Mon 01/06/15

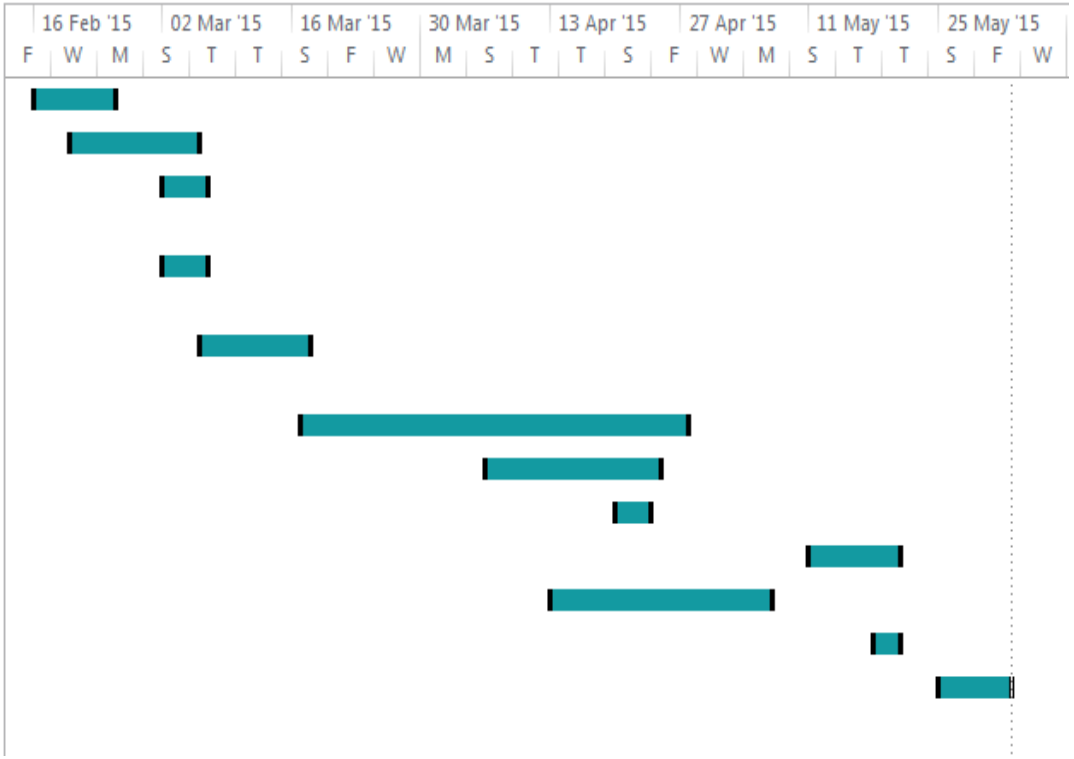

Figura 29: Diagrama de Gantt del projecte.

CONCLUSIONS

Un cop acabat el Treball de Fi de Grau de Enginyeria en Tecnologies Industrials, el qual s'ha basat en la programació de una aplicació de reconeixement de veu i la demostració de que les eines de reconeixement de veu actuals tenen molt de potencial per a aplicacions del dia a dia, s'han obtingut varies conclusions.

En aquest cas s'han aprofitat i ampliat els coneixements obtinguts durant el grau, més concretament coneixements de electrònica, de programació i en certa part de estadística. Tot això gracies al aprenentatge relatiu a la cerca de informació i a la presa de contacte amb noves eines de programació com el Visual Studio i l'ús de un nou llenguatge, el llenguatge C#.

Inicialment s'han estudiat quines eren les aplicacions actuals dels sistemes de reconeixement de veu per a poder veure que aquest projecte no és una invenció sinò una innovació. És a dir busca una aplicació nova i diferent de la resta. En aquest projecte la innovació no és que el sistema de reconeixement de veu sigui capaç de controlar una làmpada o una televisió, ja que hi ha moltes aplicacions d'aquest tipus actualment en el mercat. Sinó que és una aplicació fàcilment adaptable i que a través d'una xarxa interna de internet i algun dispositiu afegit (un mini computador) amb les connexions adequades, pot controlar gairebé qualsevol dispositiu.

En resum, s'han aconseguit acomplir tots els objectius que s'havien proposat al inici del projecte satisfactòriament. Gràcies a la selecció de la tecnologia de reconeixement de veu més adequada per al projecte s'ha aconseguit crear una aplicació de reconeixement de veu executable sobre un ordinador Windows capaç de controlar varis dispositius. Tot això queda reflectit en el funcionament del prototip que permet a un usuari controlar només amb ordres de veu una televisió i una làmpada a la vegada.

AGRAÏMENTS

S'agraeix al tutor Emilio Angulo per el suport ofert durant el procés d'aquest projecte. Sobretot destacar la seva disponibilitat i interès mostrat per a guiar-me, aconsellar-me i corregir-me sempre que ho he necessitat.

També agrair a la empresa Hewlett Packard Incorporate per a la oportunitat de tenir la experiència gràcies a la beca de pràctiques i a la ajuda oferta durant el projecte sobretot als inicis del projecte, per a pensar la idea, i al final del projecte durant el prototipatge, per la ajuda de coneixements i de material.

BIBLIOGRAFÍA

Referències bibliogràfiques

[1] ROBERT B. DUNAWAY, *The book of visual studio .NET: A Guide for Developers*, 2002.

[2] KULDIP K. PALIWAL, *Automatic Speech and Speaker Recognition: Advanced Topics,* Springer Science & Business Media, 31 mar. 1996.

[3] BRENT B. WELCH, *Practical Programming in Tcl and Tk*, Prentice Hall, 2000.

[4] Calculador de codi de colors de resistències elèctriques:

- http://www.digikey.com/es/resources/conversion-calculators/conversion-calculator-resistor-color-code-5-band

[5] Llibrería *Microsoft speech recognition*:

- https://msdn.microsoft.com/en-us/library/system.speech.recognition(v=vs.110).aspx

[6] Aplicacións de la tecnologia de reconeixement de veu:

- http://www.paralibros.com/passim/p20-tec/pg2050ci.htm

[7] Tecnologies de reconeixement de veu:

- http://www.elechouse.com/elechouse/index.php?main_page=product_info&cPath=&products_id=2151
- https://www.sparkfun.com/products/13316
- http://www.bitsophia.com/BitVoicer.aspx
- http://www.nuance.es/dragon/index.htm
- http://www.dx.com/es/p/lc-ld3320-asr-non-specific-speech-recognition-module-w-microphone-source-crystal-blue-silver-234560#.VeQVP_ntlBc
- http://www.microsoft.com/en-us/download/details.aspx?id=27225

ANNEX A: Codi de la aplicació *Mock Up*

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Globalization;
using Microsoft.Speech.Recognition;
using Microsoft.Speech.Synthesis;
using System.Threading;
using System.Net;
using System.Net.Sockets;
using System.IO;


namespace APP_VR_TFG
{
    8 references
    public partial class Form1 : Form
    {
        const int speak_commands = 7;
        const int voice_commands = 9;
        string[] speakCommands = new string[speak_commands];
        string[] voiceCommands = new string[voice_commands];
        string cultureInfoConfig;
        bool tv_state = false;
        bool light_state = false;
        SpeechSynthesizer syn;
        Socket s;

        //Language select
        3 references
        public Form1(uint langId)
        {
            InitializeComponent();
            switch (langId)
            {
                case 0:

                    speakCommands[0] = "Tv is opened";
                    speakCommands[1] = "Tv is shutted down";
                    speakCommands[2] = "Light on";
                    speakCommands[3] = "Light off";
                    speakCommands[4] = "Connected";
                    speakCommands[5] = "Disconected";
                    speakCommands[6] = "Couldn't connect to the device";

                    voiceCommands[0] = "tv on";
                    voiceCommands[1] = "tv off";
                    voiceCommands[2] = "light on";
                    voiceCommands[3] = "light off";
                    voiceCommands[4] = "menu";
                    voiceCommands[5] = "prgram up";
                    voiceCommands[6] = "program down";
                    voiceCommands[7] = "volume up";
                    voiceCommands[8] = "volume down";

                    cultureInfoConfig = "en-US";

                    break;

                case 1:
                    speakCommands[0] = "televisión encendida";
                    speakCommands[1] = "televisión apagada";
                    speakCommands[2] = "luz encendida";
                    speakCommands[3] = "luz apagada";
                    speakCommands[4] = "conectado";
                    speakCommands[5] = "desconectado";
                    speakCommands[6] = "no se pudo conectar con el dispositivo";

                    voiceCommands[0] = "encender televisión";
                    voiceCommands[1] = "apagar televisión";
                    voiceCommands[2] = "encender luz";
                    voiceCommands[3] = "apagar luz";
                    voiceCommands[4] = "menú";
                    voiceCommands[5] = "subir programa";
                    voiceCommands[6] = "bajar programa";
                    voiceCommands[7] = "subir volumen";
                    voiceCommands[8] = "bajar volumen";

                    cultureInfoConfig = "es-ES";

                    break;
```

```
            case 2:
                speakCommands[0] = "televisió encesa";
                speakCommands[1] = "televisió apagada";
                speakCommands[2] = "llum encès";
                speakCommands[3] = "llum apagat";
                speakCommands[4] = "conectat";
                speakCommands[5] = "desconectat";
                speakCommands[6] = "no s'ha pogut conectar amb el dispositiu";

                voiceCommands[0] = "encendre televisor";
                voiceCommands[1] = "apagar televisor";
                voiceCommands[2] = "encendre llum";
                voiceCommands[3] = "apagar llum";
                voiceCommands[4] = "menú";
                voiceCommands[5] = "pujar programa";
                voiceCommands[6] = "baixar programa";
                voiceCommands[7] = "pujar volum";
                voiceCommands[8] = "baixar volum";

                cultureInfoConfig = "ca-ES";

                break;

        }

        // Set up culture info
        Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo(cultureInfoConfig);
        Thread.CurrentThread.CurrentUICulture = new System.Globalization.CultureInfo(cultureInfoConfig);
        System.Globalization.CultureInfo ci = new System.Globalization.CultureInfo(cultureInfoConfig);

        //Initialize the synthesizer
        syn = new SpeechSynthesizer();
        syn.SetOutputToDefaultAudioDevice();
        foreach (InstalledVoice voice in syn.GetInstalledVoices())
        {
            if (voice.VoiceInfo.Name.Substring(46, 5) == cultureInfoConfig)
            {
                syn.SelectVoice(voice.VoiceInfo.Name);
            }
        }

        //Initialize the recognition engine
        SpeechRecognitionEngine sre = new SpeechRecognitionEngine(ci);
        sre.SetInputToDefaultAudioDevice();
        sre.SpeechRecognized += new EventHandler<SpeechRecognizedEventArgs>(sre_SpeechRecognized);



        //Get culture info
        Choices actionsChoices = new Choices();
        for (int i = 0; i < voice_commands; i++)
        {
            actionsChoices.Add(voiceCommands[i]);
        }
        Console.WriteLine(cultureInfoConfig);

        GrammarBuilder actionsGrammarBuilder = new GrammarBuilder();
        actionsGrammarBuilder.Append(actionsChoices);
        Grammar WordsGrammar = new Grammar(actionsGrammarBuilder);
        sre.LoadGrammarAsync(WordsGrammar);

        //Begin recognition
        sre.RecognizeAsync(RecognizeMode.Multiple);
    }

    //Recognition event
    1 reference
    private void sre_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
    {
        if (e.Result.Confidence > 0.7)
        {
            if (e.Result.Text == voiceCommands[2])
            {
                LED_on();
                light_state = true;
            }
            if (e.Result.Text == voiceCommands[3])
            {
                LED_off();
                light_state = false;
            }
            if (e.Result.Text == voiceCommands[0])
            {
                TV_on();
                tv_state = true;
            }
            if (e.Result.Text == voiceCommands[1])
            {
                TV_off();
                tv_state = false;
            }
```

```
            if (tv_state == true)
            {
                if (e.Result.Text == voiceCommands[8])
                {
                    vol_down();
                }
                if (e.Result.Text == voiceCommands[7])
                {
                    vol_up();
                }
                if (e.Result.Text == voiceCommands[6])
                {
                    prog_down();
                }
                if (e.Result.Text == voiceCommands[5])
                {
                    prog_up();
                }
                if (e.Result.Text == voiceCommands[4])
                {
                    menu();
                }
            }
        }
    }

    //EnableButton
    1 reference
    private void EnableButton_Click(object sender, EventArgs e)
    {
        s = new Socket(AddressFamily.InterNetwork,
            SocketType.Stream,
            ProtocolType.Tcp);
        IPAddress[] IPs = Dns.GetHostAddresses(textBox1.Text);
        try
        {

            TcpClient tcpclnt = new TcpClient();
            Console.WriteLine("Connecting.....");
            s.Connect(IPs[0], 9106);

            // use the ipaddress as in the server program

        }
        catch (Exception)
        {
            syn.Speak(speakCommands[6]);
            return;
        }
        EnableButton.Enabled = false;
        DisableButton.Enabled = true;
        syn.Speak(speakCommands[4]);

    }

    //DisableButton
    1 reference
    private void DisableButton_Click(object sender, EventArgs e)
    {
        s.Disconnect(false);

        EnableButton.Enabled = true;
        DisableButton.Enabled = false;
        syn.Speak(speakCommands[5]);
    }

    //LED
    1 reference
    void LED_on()
    {
        byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [encendre_llum]" + '\n');
        s.Send(sendBytes);
        syn.Speak(speakCommands[2]);
    }
    1 reference
    void LED_off()
    {
        byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [apaga_llum]" + '\n');
        s.Send(sendBytes);
        syn.Speak(speakCommands[3]);
    }
```

```
        1 reference
        void TV_on()
        {
            byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [encendre_televisor]" + '\n');
            s.Send(sendBytes);
            syn.Speak(speakCommands[0]);
        }
        1 reference
        void TV_off()
        {
            byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [apaga_televisor]" + '\n');
            s.Send(sendBytes);
            syn.Speak(speakCommands[1]);
        }
        1 reference
        void vol_down()
        {
            byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [volum-]" + '\n');
            s.Send(sendBytes);
        }
        1 reference
        void vol_up()
        {
            byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [volum+]" + '\n');
            s.Send(sendBytes);
        }
        1 reference
        void prog_down()
        {
            byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [prog-]" + '\n');
            s.Send(sendBytes);
        }
        1 reference
        void prog_up()
        {
            byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [prog+]" + '\n');
            s.Send(sendBytes);
        }
        1 reference
        void menu()
        {
            byte[] sendBytes = Encoding.ASCII.GetBytes("puts $sock [menu]" + '\n');
            s.Send(sendBytes);
        }

        //IP conection
        1 reference
        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {

        }

    }//Class
}//APP_VR_TFG
```

S'han utilitzat imatges del codi, per a que es pogui distinguir el codi de colors del sript de la aplicació o *Mock Up*, que dóna informació més visual que no posant senzillament el text del codi.

ANNEX B: Codi del programa en TCL servidor.

Makefile

```
all: tclserverVR.x tclserverVR.so

tclserverVR.x: tclserverVR.cpp
	g++ -O3 -std=c++11 -o tclserverVR.x tclserverVR.cpp

tclserverVR_wrap.cxx: tclserverVR.i
	swig -c++ -tcl tclserverVR.i

tclserverVR_wrap.o: tclserverVR_wrap.cxx
	g++ -O3 -fPIC -c tclserverVR_wrap.cxx -I/usr/include/tcl8.5/

tclserverVR.o: tclserverVR.cpp
	g++ -O3 -std=c++11 -fPIC -c tclserverVR.cpp

tclserverVR.so: tclserverVR.o tclserverVR_wrap.o
	g++ -O3 -o tclserverVR.so -shared tclserverVR.o tclserverVR_wrap.o

clean:
	rm -rf *.cxx *.o *.x *.so
```

TclServer.tcl

```
#!/bin/sh
# TclServer.tcl \
exec tclsh "$0" ${1+"$@"}


package require Tclx


load ./tclserverVR.so

# Global variables
global APPort
global APChan
global midasLcdIP
global midasLcdPort
global midasLcdChan
global timeout_Connection


# Global Initialization
set APPort 9106
set midasLcdIP 0
set midasLcdPort 9104
set midasLcdChan 0
set timeout_Connection 10000


proc  svcHandler {sock} {
  dup $sock stdout
  set comando [gets $sock]  ;# get the client packet
  if {[eof $sock]} {            ;# client gone or finished
     close $sock                ;# release the servers client channel
     puts stderr "CLOSE!!!"
  } else {
        puts stderr "info Command from client:$comando"
        set errorcode [catch $comando msg]
        if {$errorcode != 0} {
            puts stderr "error $errorcode Mensage $msg"
        }
  }
}


proc accept {sock addr port} {
  puts $sock "$addr:$port, You are connected to application AP."
  flush $sock
  fconfigure $sock -buffering line -blocking 0
  fileevent $sock readable [list svcHandler $sock]
  dup $sock stdout
  puts stderr "info: Accepted connection from $addr"
}


set errorcode [catch { set APChan [socket -server accept $APPort]} msg]
if {$errorcode != 0} {
        puts stderr "error $errorcode Mensage $msg"
} else {
        puts stderr "info server started and waiting for connections..."
        vwait forever
}
```

Tclserver.cpp

```
/*tclserverVR.cpp*/

#include<iostream>
#include<fstream>
#include<math.h>
#include<sstream>
#include<string>
#include<stdio.h>
#include<thread>
#include<mutex>
#include<chrono>
#include<unistd.h>
using namespace std;

void encendre_llum(){
        string gpionum = "7";
        string export_str = "/sys/class/gpio/export";
        string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
        string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
        ofstream exportgpio(export_str.c_str());
        ofstream setdirgpio(setdir_str.c_str());
        ofstream setvalgpio(setval_str.c_str());

        if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
                cout << "OPEATION FAILED: Unable to export GPIO";
        }
        exportgpio << gpionum;
        setdirgpio << "out";
        setvalgpio << "0";
        exportgpio.close();
        setdirgpio.close();
        setvalgpio.close();
}

void apaga_llum(){
        string gpionum = "7";
        string export_str = "/sys/class/gpio/export";
        string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
        string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
        ofstream exportgpio(export_str.c_str());
        ofstream setdirgpio(setdir_str.c_str());
        ofstream setvalgpio(setval_str.c_str());

        if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
                cout << "OPEATION FAILED: Unable to export GPIO";
        }
        exportgpio << gpionum;
        setdirgpio << "out";
        setvalgpio << "1";
        exportgpio.close();
        setdirgpio.close();
        setvalgpio.close();
}
```

```
void encendre_tv(){
        string gpionum = "14";
        string export_str = "/sys/class/gpio/export";
        string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
        string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
        ofstream exportgpio(export_str.c_str());
        ofstream setdirgpio(setdir_str.c_str());
        ofstream setvalgpio(setval_str.c_str());

        if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
                cout << "OPEATION FAILED: Unable to export GPIO";
        }
        exportgpio << gpionum;
        setdirgpio << "out";
        setvalgpio << "0";
        exportgpio.close();
        setdirgpio.close();
        setvalgpio.close();
}


void apaga_tv(){
        string gpionum = "15";
        string export_str = "/sys/class/gpio/export";
        string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
        string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
        ofstream exportgpio(export_str.c_str());
        ofstream setdirgpio(setdir_str.c_str());
        ofstream setvalgpio(setval_str.c_str());

        if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
                cout << "OPEATION FAILED: Unable to export GPIO";
        }
        exportgpio << gpionum;
        setdirgpio << "out";
        setvalgpio << "1";
        exportgpio.close();
        setdirgpio.close();
        setvalgpio.close();
}

void menu(){
        string gpionum = "18";
        string export_str = "/sys/class/gpio/export";
        string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
        string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
        ofstream exportgpio(export_str.c_str());
        ofstream setdirgpio(setdir_str.c_str());
        ofstream setvalgpio(setval_str.c_str());

        if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
                cout << "OPEATION FAILED: Unable to export GPIO";
        }
        exportgpio << gpionum;
        setdirgpio << "out";
        setvalgpio << "1";
        usleep(1000000);
        setvalgpio << "0";
        exportgpio.close();
        setdirgpio.close();
        setvalgpio.close();
}
```

```
void prog+(){
        string gpionum = "23";
        string export_str = "/sys/class/gpio/export";
        string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
        string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
        ofstream exportgpio(export_str.c_str());
        ofstream setdirgpio(setdir_str.c_str());
        ofstream setvalgpio(setval_str.c_str());

        if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
                cout << "OPEATION FAILED: Unable to export GPIO";
        }
        exportgpio << gpionum;
        setdirgpio << "out";
        setvalgpio << "1";
        usleep(1000000);
        setvalgpio << "0";
        exportgpio.close();
        setdirgpio.close();
        setvalgpio.close();
}

void prog-(){
        string gpionum = "24";
        string export_str = "/sys/class/gpio/export";
        string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
        string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
        ofstream exportgpio(export_str.c_str());
        ofstream setdirgpio(setdir_str.c_str());
        ofstream setvalgpio(setval_str.c_str());

        if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
                cout << "OPEATION FAILED: Unable to export GPIO";
        }
        exportgpio << gpionum;
        setdirgpio << "out";
        setvalgpio << "1";
        usleep(1000000);
        setvalgpio << "0";
        exportgpio.close();
        setdirgpio.close();
        setvalgpio.close();
}


void volum+(){
        string gpionum = "25";
        string export_str = "/sys/class/gpio/export";
        string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
        string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
        ofstream exportgpio(export_str.c_str());
        ofstream setdirgpio(setdir_str.c_str());
        ofstream setvalgpio(setval_str.c_str());

        if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
                cout << "OPEATION FAILED: Unable to export GPIO";
        }
        exportgpio << gpionum;
        setdirgpio << "out";
        setvalgpio << "1";
        usleep(1000000);
        setvalgpio << "0";
        exportgpio.close();
        setdirgpio.close();
        setvalgpio.close();
}
```

```
void volum-(){
        string gpionum = "8";
        string export_str = "/sys/class/gpio/export";
        string setdir_str = "/sys/class/gpio/gpio" + gpionum + "/direction";
        string setval_str = "/sys/class/gpio/gpio" + gpionum + "/value";
        ofstream exportgpio(export_str.c_str());
        ofstream setdirgpio(setdir_str.c_str());
        ofstream setvalgpio(setval_str.c_str());

        if (exportgpio < 0 || setdirgpio < 0 || setvalgpio < 0){
                cout << "OPEATION FAILED: Unable to export GPIO";
        }
        exportgpio << gpionum;
        setdirgpio << "out";
        setvalgpio << "1";
        usleep(1000000);
        setvalgpio << "0";
        exportgpio.close();
        setdirgpio.close();
        setvalgpio.close();
}

/* Main
 */
int main() {
    return 0;
}
```