Grau en Enginyeria Informàtica

Treball de Final de Grau

Dispositiu capaç de reconèixer ordres de veu, que permet a persones amb diversitat funcional utilitzar un ascensor

Autor:
Philippe González Miralles

Supervisor:
Diego Centelles Beltrán
Tutor acadèmic:
Raúl Marín Prades

Data de lectura: 16 de Març de 2022
Curs acadèmic 2021/2022

Resum

El projecte té com a objectiu fer un viatge en ascensor sense necessitat de prendre cap interruptor ni element físic, per exemple, per a donar suport a persones amb discapacitat. Hem investigat diferents eines i tecnologies com: *TensorFLow, Jupyter/Colab, C++, protocol CAN, Eepressif.* El projecte es pot dividir en 3 parts: captació d'ordres de veu, identificació de les ordres mitjançant un model entrenat, i comunicació CAN per indicar les accions a l'ascensor.

En aquest projecte s'intenta cobrir la necessitat generada per la covid-19, ja que em hagut d'evitar tocar superfícies, i a banda cobrir les necessitats de la gent amb mobilitat reduïda. El funcionament del dispositiu és simple: funciona amb una paraula que activa el sistema en aquest cas *Nexy* i després espera l'orde de veu, així com *planta baja, uno, dos, terraza....* D'aquesta forma intentem evitar els falsos positius amb el sistema. Els resultats que hem obtingut han sigut bons, sempre que utilitzem un nombre de paraules reduïdes, ja que es produeixen més falsos positius a l'hora d'ampliar el nombre de paraules. La implantació d'aquest sistema en un edifici de 6 plantes, es realitzaria de la següent forma: un dispositiu instal·lat en la cabina, i després un en cada planta per poder cridar-lo. Així que per calcular quants dispositius necessitem per fer-hi una instal·lació completa, caldrà saber el número de plantes i sumar-li 1. La conclusió general que he obtingut, és que aquest projecte m'ha permés aprendre com treballen en una empresa tecnològica. També he reconegut possibles millores del projecte amb el qual seria una versió 2.0, però una de les coses més importants seria desenvolupar un model amb un 'set' de paraules més gran capaç de reconéixer una major varietat de paraules a l'hora, ja que açò és primordial per a la seua implantació.

Resumen

El proyecto tiene como objetivo hacer un viaje en ascensor sin necesidad de presionar ningún interruptor ni elemento físico, para así dar adecuado soporte a personas con discapacidad. Hemos investigado diferentes herramientas y tecnologías como: *Tensor Flow, Jupyter/Colab, C++, protocolo CAN, Espressif.* El proyecto se puede dividir en 3 partes: captación de órdenes de voz, identificación de los comandos de voz mediante un modelo previamente entrenado, y comunicación CAN para indicar los movimientos al ascensor.

En este proyecto se intenta cubrir la necesidad generada por el covid-19 que nos ha llevado a evitar tocar superficies y además cubrir las necesidades de la gente con movilidad reducida. El funcionamiento del dispositívo es sencillo, funciona con una palabra que lo activa, en este caso *Nexy* y después queda a la espera de una segunda palabra como: *planta baja, uno, dos, terraza....* De esta forma evitamos tener falsos positivos en el sistema. Los resultados que hemos obtenido han sido buenos, siempre y cuando utilizamos un número reducido de palabras, ya que si utilizamos un conjunto grande, se producen falsos positivos entre las palabras a detectar. La puesta en marcha del sistema, en un edificio, por ejemplo, de 6 plantas se realizaría de la siguiente manera, un dispositivo en la cabina y otro en cada planta, para así de esta manera poder llamarlo desde cualquier lugar. Para la instalación se parte del número de plantas más uno(1). Así que para calcular cuantos necesitamos es tan fácil como el número de plantas más uno. La conclusión general que he obtenido, es que este proyecto me ha permitido aprender

la manera de trabajar en una empresa tecnológica. Además, he pensado algunas mejoras que se podrían implantar en el proyecto que pasaría a ser la versión 2.0, y una de las mejoras más importantes sería un modelo de palabras capaz de reconocer una mayor variedad de palabras, ya que esto nos facilitaría su implantación.

Paraules clau / Palabras clave

M5Stack, ESP32, TensorFlow, C++, AA, CNN, Encastat, Colab, Jupyter.

Keywords

M5Stack, ESP32, TensorFlow, C++, AA, CNN, Embedded, Colab, Jupyter.

Agraïments

Arribar fins a aquest projecte no seria possible sense el suport dels meus pares i avis que han estat presents durant tota aquesta etapa en els bons i els mals moments. A més a més, donar les gràcies a tots els amics de la universitat que hem passat junts molt de temps, fent treballs i estudiant a les nits en la biblioteca. En especial a la meua parella, Laia Martínez, que ha estat al meu costat durant tota aquesta etapa.

Agrair l'ajuda del tutor Raul Marín per la seua mentorització, ajuda i consells durant el projecte, també agrair a Javier Estruch Garcia la seua ajuda en aquest projecte i per últim, però no menys important a dos companys de Nayar Systems, David Castellano i Diego Centelles per la seua ajuda i guia en el projecte.

Índex

Capítol 1

Introducció

Índex

En aquest capítol contextualitzem el projecte que hem dut a terme en l'empresa **Nayar Systems** en les pràctiques curriculars del grau. Especificarem els objectius del projecte tenint en compte els requisits que es deuen complir.

1.1 Estructura de la memòria

Aquesta memòria està dividida en 5 capítols.

- El capítol 1, Introdueix el projecte en el context i motivació, objectiu i abast.
- El capítol 2, es comenta la planificació del projecte així com la seua metodologia, recursos i costos.
- El capítol 3, es comenta l'anàlisi del sistema i el disseny de l'arquitectura.
- El capítol 4, es comenta els detalls de la implementació amb la seua validació i verificació.
- El capítol 5, es comenta les conclusions obtingudes en desenvolupar el projecte.

1.2 Motivació i context del projecte

El projecte que es proposa es realitza durant les pràctiques formatives com a projecte de final de grau. En les pràctiques s'apliquen els coneixements apresos durant el grau, ja que sense la base dels primers cursos i l'especialització de l'últim no haguera pogut desenvolupar aquest projecte.

La companyia **Nayar Systems**, està situada en Castelló de la Plana, i es dedica principalment a crear solucions de *IoT*, Internet de les coses, per als camps de les elevacions i la indústria.

La missió de la companyia és crear solucions IoT lleugeres i competitives per acompanyar als clients en la seua evolució tecnològica, des del coneixement expert en ascensors i sistemes industrials.

Nayar Systems és una companyia mitjana, ja que fins fa poc no ha superat el llindar dels 60 treballadors. A més a més, compta amb dues oficines, la de Castelló de la Plana i una altra que està a la Xina. Han triat expandir-se allí, perquè és un dels països on l'urbanisme creix més verticalment del món i això suposa una oferta molt gran en el mercat dels ascensors. La companyia compta amb diversos departaments i està organitzada de la següent forma.

- **Direcció:** Dissenya i implementa en totes les àrees l'estratègia anual i la missió, que fan possible apropar-nos fins a la visió estratègica de la companyia.
- **Administració i finances:** Aquest departament és l'encarregat d'estudiar i gestionar la realitat econòmica i financera de **Nayar Systems**, clau per prendre decisions a llarg termini.
- **Producte:** Aquest departament està format per les àrees d'informàtica i enginyeria. La primera d'aquestes és el I+D de **Nayar Systems**, qui posa a punt els productes i serveis, des de la implantació del software i el manteniment fins a la supervisió i infraestructura de sistemes d'informació. Ací dins existeixen dos departaments diferenciats per la seua feina: *Embedded i Frondend/Backend*. Aquest últim afronta nous projectes basats en les necessitats del client i/o mercat, des del disseny conceptual fins a la seua posada en funcionament, passant per controlar la seua fabricació i qualitat.
- **Cotxera:** Ací és on es creen nous productes, tecnologies i vies de negoci. El seu objectiu és convertir en realitat les idees junt amb Producte.
- **Projectes:** Està dirigit a oferir un servei personalitzat als clients per obtenir solucions, acord a les seues necessitats en l'àmbit de la digitalització i IoT.
- **Marketing i vendes:** És el departament d'investigar el mercat, elaborar estratègies que donen a conéixer els productes i serveis i la comercialització. Està dividit en dues àrees geogràfiques: Europa i la Xina. Dins d'aquest departament es troba també **Comunicació**, des d'on es gestionen totes les àrees vinculades a la comunicació de la companyia, tant interna com externa.
- **Operacions:** Està integrat dins d'**Àrea Tècnica**, responsable de gestionar cada incidència o consulta; **Qualitat**, qui controla tant els processos interns com els externs,

perquè aquests tinguen resultats òptims; i **Logística** i **Compres/Producció**, àrees en els que es gestiona i planifica el transport, emmagatzemament i distribució.

- **Capital Humà:** s'encarrega des de la selecció de personal fins al desenvolupament i evolució de la trajectòria professional de cadascun dels components de l'organització.

El projecte que es presenta, està dins del departament de **Producte**, en la secció d'**Informàtica** en la secció d'**Embedded**, on treballen en dispositius creats per l'empresa, amb un nucli de ESP32 a més de donar suport i fer actualitzacions a tots els productes. Aquest treball està relacionat amb l'automatització del funcionament d'ús d'un ascensor, per evitar el procés de pressionar físicament cap element de l'ascensor. Tenint en compte açò, el projecte es centra en la captació d'ordes de veu, per identificar-les com a comandes i poder indicar a l'ascensor cap a on es té que moure.

Com s'ha de desenvolupar un reconeixement de comandes de veu mitjançant una *IA*, aquest projecte es pot utilitzar per a diferents usos, ja siga per donar ordres a un ascensor com per automatitzar qualsevol sistema propi de l'empresa; a més que és fàcilment escalable per afegir-hi diversos idiomes.

1.3 Objectiu del projecte

L'objectiu principal del projecte és poder utilitzar l'ascensor sense haver de tocar cap element. Per complir l'objectiu, s'ha d'aconseguir poder captar comandes de veu i comunicar-se amb l'ascensor per poder moure's d'una planta a una altra.

Aquest objectiu es pot dividir en tres sub-objectius:

- En primer lloc, s'ha d'aconseguir poder captar l'àudio de forma contínua per després poder analitzar-lo.
- Després poder interpretar les comandes de veu per poder interpretar-lo.
- I comunicar el dispositiu i l'ascensor perquè es moga d'un lloc a un altre.

1.3.1 Abast del projecte

L'abast del projecte es divideix en abast funcional, organitzatiu i informàtic.

Abast funcional

EL producte que es desenvolupa ha de ser capaç de reconéixer les comandes de veu de diferents persones mitjançant un model entrenat per *tensorFlow* amb *Júpiter* i poder enviar la informació per *CAN* a l'ascensor per poder fer un viatge entre diferents plantes.

Abast organitzatiu

El projecte que es desenvolupa, està localitzat en la secció d'**Embedded** que està dins del departament d'**informàtica** en el departament de **Producte**.

I l'objectiu més ampli d'aquest projecte, és fer una investigació per veure les tecnologies existents per poder implementar-ho i en un futur poder incloure'l com a un producte propi de l'empresa.

Abast informàtic

El dispositiu ha de ser capaç de captar àudio de forma contínua mitjançant *I2S* per poder interpretar comandes de veu amb un model entrenat i trametre la informació correctament per un *CAN Bus* per poder fer un viatge en ascensor.

1.4 Descripció del projecte

L'objectiu - recordem - del projecte de l'empresa és crear un prototip que siga capaç de fer un viatge en ascensor sense pressionar cap element. A partir d'aquest prototip es pretén fer una investigació i poder adaptar-lo als productes ja existents de l'empresa com una funcionalitat nova, tot per poder vendre-lo als clients.

Així doncs, s'ha de començar des de zero fent una investigació de les tecnologies existents per poder implementar-lo.

En primer lloc, s'ha investigat el hardware que existeix en el mercat per a la implementació del projecte. Existeixen diferents microcontroladors com ès el cas d'*Arduino* i *ESP32* amb els diferents models que tenen. S'ha seleccionat *ESP32* perquè té més capacitat de processament que un *Arduino* i ens permet majors funcionalitats a un preu més reduït. Hem trobat un dispositiu que compleix tots els requisits. A més a més, porta diferents components integrats que ens dona facilitat, és el dispositiu *M5 Echo* que compta amb un *ESP32 pico*, que porta incorporat un micròfon i un altaveu pel protocol *I2S*, per tant, això ens soluciona el tema de fer un estudi de protocol de comunicació.

També hem hagut de fer una investigació sobre el reconeixement de veu, hem vist que existeixen diferents formes de IA, però la seleccionada ha sigut TensorFLow Lite, ja que ens permet una implementació més senzilla en el dispositiu ESP32. Per "entrenar"el model s'ha utilitzat Jupyter amb Tensorflow. Per comunicar-nos en els ascensors, partim de les comunicacions que accepten la majoria d'ascensors, sempre pensant en poder fer-ho lo més escalable possible. Així que, com tots els ascensors tenen una comunicació CAN, hem seleccionat aquest mètode per comunicar-nos en l'ascensor. Ja que perquè siga escalable sols es té que canviar un fitxer de configuració on s'especifica la velocitat i les trames corresponents que s'han d'enviar.

En resum, el dispositiu estarà format per un Atom Echo, que està format per un ESP32, que farà servir I2S per a la captura de comandes de veu que seran interpretades per un model entrenat amb TensorFLow Lite i es comunica amb l'ascensor mitjançant el protocol CAN Bus.

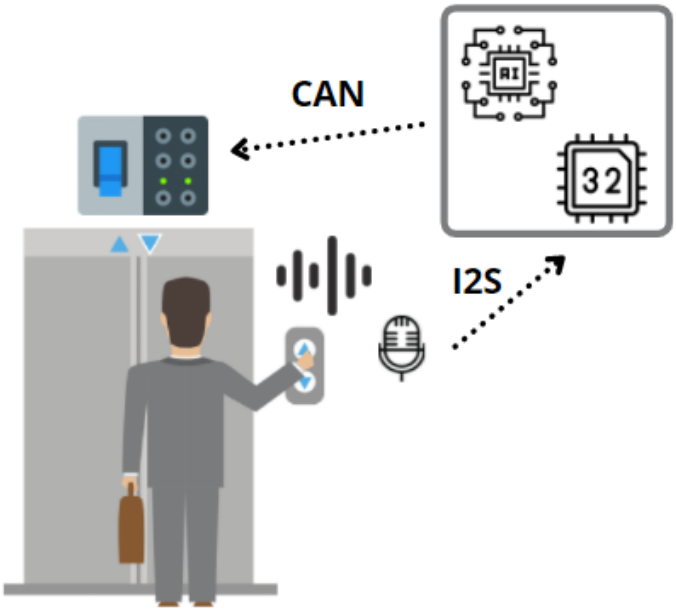


Figura 1.1: Descripció general del sistema

Capítol 2

Planificació del projecte

Índex

2.1 Estat de l'art

Aquest projecte s'ha desenvolupat com un prototip de l'empresa **Nayars Systems**. El dispositiu serà capaç de funcionar amb els ascensors que compten amb connectivitat *CAN*. Abans de dur a terme aquest projecte, l'empresa ja comptava amb un dispositiu similar per enviar ordres a un ascensor, un dispositiu que es connecta a l'ascensor i mitjançant una aplicació mòbil amb *Bluetooth* es comunica i realitza l'acció determinada [1]. Però aquest prototip serà capaç de funcionar sense cap tipus de connectivitat i sense dispositius externs, com el telèfon mòbil, sols amb ordres de veu.

Fent un estudi del mercat s'ha trobat un producte i un treball de recerca similars:

- La marca **Hyundai Elevator Co**, és un fabricant d'ascensor de Corea del sud que amb l'empresa **KT Corp**, estan desenvolupant un producte d'Intel·ligència Artificial, per al reconeixement de veu i poder interactuar amb l'ascensor.[2]
- En la conferència del 10 de novembre del 2021 en *Lipetsk, Russian Federation.* Es presenta un estudi similar al que es tracta en aquest treball, amb algunes diferències com la forma d'interactuar amb l'ascensor, en la conferència es proposen interruptors electromagnètics per a comunicar-se amb l'ascensor i en aquest treball s'utilitza el protocol *CAN* per comunicar-se amb ell.[3]

Com es pot observar, la idea de fer servir la combinació d'assessors amb *IA* està a l'alça i que en un futur estarà prou demandat pels clients.

2.2 Metodologia

La metodologia utilitzada per a la planificació del projecte és la predictiva. Aquesta metodologia estableix una planificació de tasques inicials i estima el temps que durarà. Diàriament, s'ha comentat amb el tutor l'estat del projecte i setmanalment s'ha fet una reunió per analitzar l'estat actual i revisar els objectius de la setmana següent.

El seguiment del projecte s'ha dut a terme mitjançant Trello on hem definit les tasques i fet un control de les realitzades i pendent.

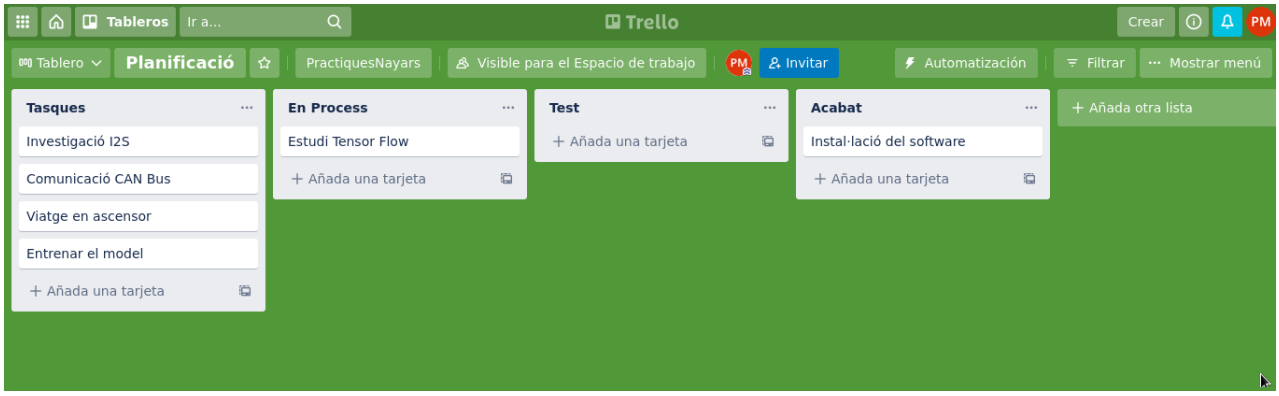


Figura 2.1: Ferramenta d'organització de tasques, trello.

2.3 Planificació

En aquest apartat es descriuen les tasques en què s'ha dividit el projecte. Aquest projecte consta d'unes 300h per a la realització de les pràctiques en l'empresa. En la figura 2.2 podem veure l'esquema.

- **Estudi previ de les diferents tecnologies i dispositius:** Es té en compte els casos d'usos i requisits per fer una anàlisi dels diferents protocols de comunicació i dispositius que hi ha en el mercat.
- **Instal·lació del software necessari per al projecte:** Instal·lar l'entorn de programació en l'ordinador de l'empresa, la qual ja proporciona un equip per a l'estància de pràctiques.
- **Estudi i creació del model de *TensorFLow*:** S'investiga amb Tensor Flow per veure com funciona i com es pot utilitzar en el projecte per crear un model.
- **Entrenament de diferents models:** S'asseguen diferents models per obtenir el que més ens pot interessar i s'adapte a les necessitats.

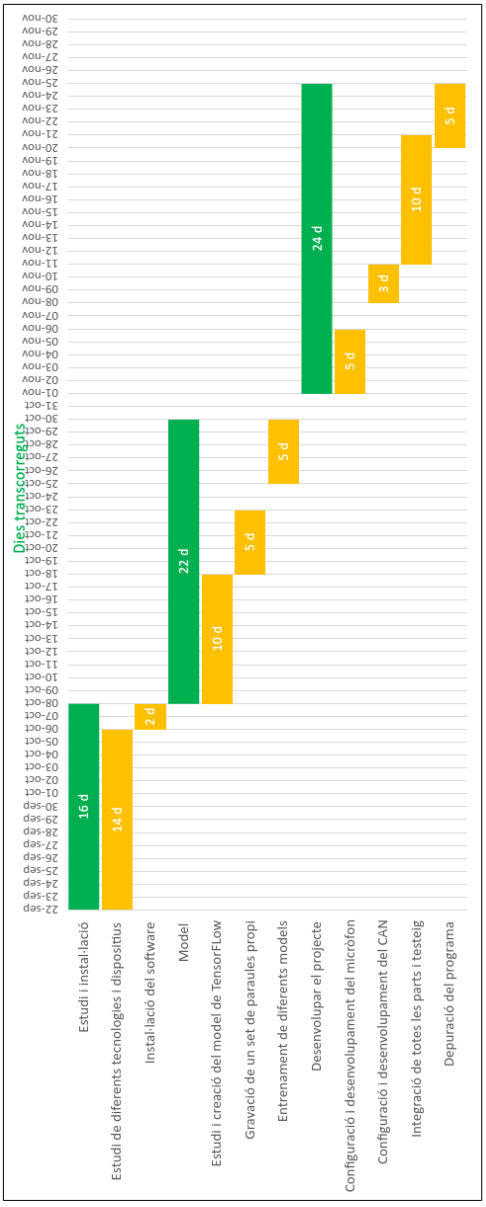
- **Gravació d'un "set"de paraules propi:** Es planteja gravar un "set"de paraules propi per estudiar els resultats. Açò comporta la gravació de les paraules i després adaptar - totes elles - en el model.

- **Configuració i desenvolupament del micròfon:** Es configura el protocol de comunicació del micròfon i es realitzen diferents proves per veure com funciona.

- **Configuració i desenvolupament del sistema de comunicació de l'ascensor:** Analitze'm el sistema de configuració seleccionat i experimentem com funciona per utilitzar-lo en l'ascensor.

- **Integració de totes les parts i testeig:** Una vegada provades totes les parts, s'unifiquen per veure el seu funcionament i provar si funcionen correctament entre elles.

- **Depuració del programa:** Analitzem el programa per a veure si està ben organitzat i si es pot optimitzar per a corregir alguns errors, si és que n'han sorgit.



Tasca	Inici	Duració
Estudi i instal·lació	22-sep	16
Estudi de diferents tecnologies i dispositius	22-sep	14
Instal·lació del software	06-oct	2
Model	08-oct	22
Estudi i creació del model de TensorFLow	08-oct	10
Gravació de un set de paraules propi	18-oct	5
Entrenament de diferents models	25-oct	5
Desenvolupar el projecte	01-nov	24
Configuració i desenvolupament del micròfon	01-nov	5
Configuració i desenvolupament del CAN	08-nov	3
Integració de totes les parts i testelg	11-nov	10
Depuració del programa	20-nov	5

Figura 2.2: Diagrama de Gantt.

2.4 Estimació de recursos i costos del projecte

En aquest apartat es calcula el cost de totes les parts del projecte. Les hem dividit en diverses parts: costos de software i de hardware per a desenvolupar el projecte; costos del prototip, i costos de recursos humans (on es comptabilitzen les hores de feina i la gestió). En la taula 2.1 podem veure el cost del software i del hardware, necessaris per a poder fer el prototip.

Recursos			
Software	**Quantitat**	**Cost (€)**	**Total (€)**
IDE Arduino	1	0	
ESP-IDF	1	0	
Visual Studio	1	0	
Jupyter	1	0	
GitHub	1	0	0
Hardware			
Atom Echo	1	9.29	9.29
Atom CAN Bus	1	19.34	19.34
Total			**28.63**

Taula 2.1: Costos dels recursos de software i de hardware.

Per poder tirar endavant el prototip s'ha utilitzat el següent equip informàtic, i amb detall dels costos dels seus components, vegeu la taula 2.2.

Hardware		
Software	**Quantitat**	**Cost (€)**
Lenovo 80XH	1	509
Monitor samsung s22d300hy	1	114
Microsoft Keyboard 400	1	20
Gigabayte mouse GM-M63000	1	13
Xiaomi Redmi 8x	1	139
Total		**795.00**

Taula 2.2: Costos del recursos de hardware per dur a terme les pràctiques.

Les despeses dels recursos humans segons el conveni de les TIC en 2021, les concretem com segueix: un perfil junior, es troba en l'Àrea 3 del grup E Nivell 1 segons les taules salarials, així que el salari és de 12.701,05€ anuals, el que equival a 1.058,42€ al mes.

En la taula 2.3 podem veure la suma de tots els costos que ha suposat dur a terme aquest prototip per a l'empresa.

Recursos	**total (€)**
Recursos software i hardware	28.63
Recursos equip	795
Recursos humanos	3175.26
Total	**3998.89**

Taula 2.3: Costos totals del projecte.

2.5 Seguiment del projecte

El seguiment del projecte s'ha realitzat setmanalment per veure el seu estat i alhora assegurar que es realitzen les tasques programades, per a complir els objectius. A més a més, diàriament s'han anat discutint els entrebancs i avanços en la realització del projecte, constant així millor quin és el seu estat. I no s'ha utilitzat cap eina de comunicació entre les parts, ja que tant el supervisor com jo ens trobàvem en l'empresa i en la mateixa planta. Per mantenir les reunions s'ha utilitzat el Google Calendar per fer la planificació.

Capítol 3

Anàlisi i disseny del sistema

Índex

En aquest capítol es realitza l'estudi del sistema, que s'ha desenvolupat amb l'especificació de les tecnologies utilitzades. En primer lloc, es fa un estudi dels protocols i les tecnologies que es poden usar per enviament de dades i comunicació en el món de les "elevacions". A més, s'estudia com assajar un model que reconega les comandes de veu, i millorar-lo (el model) a base de tasques d'entrenament del programa que li dona suport. La idea, per dur a terme aquest projecte, és no fer servir una connexió remota a Internet. És per això, que es du a terme un model de dades que puga reconéixer comandes de veu.

3.1 Anàlisi del sistema

Aquest projecte el podem dividir principalment en 3 faces, i a cadascuna d'elles s'utilitza una tecnologia diferent. La primera és la comunicació mitjançant *I2S* amb el micròfon; després, l'ús de *Tensor Flow* per poder interpretar les comandes de veu, i finalment l'ús de *CAN Bus* per comunicar-nos amb l'ascensor.

3.1.1 Protocol d'enviament de dades

En aquest apartat es realitza un estudi dels diferents protocols que utilitzen els microcontroladors per enviar informació. Existeixen dos tipus de comunicació, la paral·lela i la serial[4]. La paral·lela té més desavantatges que avantatges, i consisteix a enviar 1 bit per 8 línies diferents més una referència al *GND*. Açò fa que s'incremente el cost, i a més no serveix per a llargues distàncies, ja que com major és la distància major són les interferències que es generen entre les línies.

Un altre dels aspectes negatius és que té l'ús de 8 pins, cosa que és un problema, perquè ens pot ocupar la majoria de pins.

En canvi, una de les coses bones que te és la velocitat, ja que ens permet enviar 8 bits alhora. Per altra banda, la sèrie envia la informació de forma seqüencial, un bit de dades per un canal. Encara que és més lent que el paral·lel, ens permet enviar dades a una major distancia utilitzant un menor número de línies. Aquest és el més usat actualment. Un exemple d'aquesta comunicació són els USB.

Val afegir que, dins de la comunicació serial existeixen dos tipus, *síncrona i asíncrona*. La síncrona fa servir dues línies per l'enviament de dades i una altra per al rellotge. I s'usa per a sincronitzar la transferència de dades. Un protocol síncron és l'*I2C*. En canvi, l'*asíncrona*, no fa servir el senyal de rellotge. I és que la *sincronització* l'estableix l'emissor enviant uns bits especials al principi i al final de cada missatge.

D'altra banda, un dels paràmetres més importants és la velocitat de transmissió, ja que tant l'emissor com el receptor han de tenir la mateixa. Aquesta velocitat s'expressa en bauds per segons. Un protocol asíncron és el *CanBus*.

CAN Bus

El protocol *CAN Bus* [5] (*Controller Area Network*), és un bus automotriu desenvolupat per Bosch, que permet als microcontroladors i dispositius comunicar-se entre si. És un protocol basat en missatges, i dissenyat específicament per aplicacions automotrius. Però també s'utilitza amb àrees com l'aeroespacial, l'automatització industrial i en equips mèdics. Permet interconnectar dispositius amb un menor nombre de cables, ja que sols fa servir 2, i per això és un protocol en serie i asíncron.

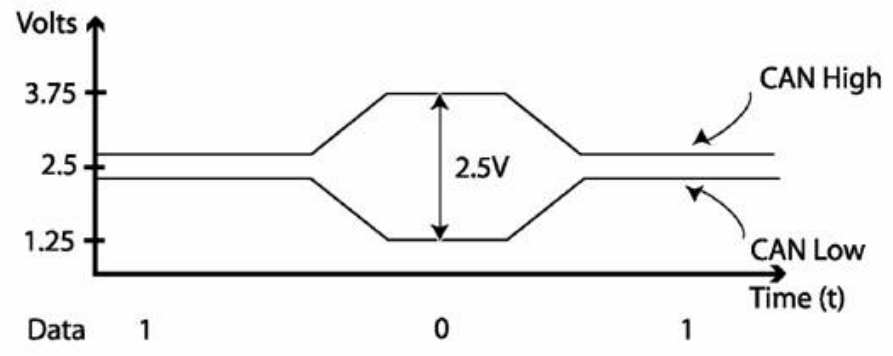


Figura 3.1: Diferència de voltatge de la comunicació CAN

El seu funcionament és mitjançant dos cables, un CAN alt (CAN High) i una altra CAN baix (CAN Low). Quan està inactiu dues línies transporten 2.5v i quan estan transmitent l'alta ho fa 3.75v i la baixa a 1.25v, generant una diferència de potencial de 2.5v. Les dues línies estan referenciades entre elles, no sent necessari una connexió a terra, ja que la connexió es basa en una diferència de voltatge entre les dues línies de bus; el *CAN* no és sensible a pics inductius, ni a camps elèctrics ni a soroll. Açò la converteix en l'opció favorita per a les comunicacions entre equips mòbils, com per exemple un ascensor.

UART

El protocol *UART* (*Universal Asynchronous Receiver - Transmiter*) és un protocol de tipus asíncron, i, per tant, no requereix un senyal de rellotge. Utilitza 3 línies, una per enviar dades, una altra per a rebre-les i l'altra per a connectar a terra. Els pins usats són els *RX* i *TX*. Per a poder fer servir aquest protocol s'han d'establir uns paràmetres en els dos dispositius: la velocitat, els bits de parada i els bits de paritat.

- **Velocitat de transmissió:** Aquesta s'expressa en bauds per segon i és la quantitat de bits que es transmetran en un segon.
- **Bits de parada:** La quantitat de bits indica al receptor que la transmissió ha acabat.
- **Bit de paritat (bit de control):** Aquest bit s'usa per a determinar si hi ha algun error en la transmissió.

SPI

Aquest protocol de comunicació és síncron, per tant, utilitza un senyal de rellotge. Utilitza dues línies per a la transferència de dades, i, per tant, treballa en full dúplex. Pot enviar i rebre dades al mateix temps. Els perifèrics poden connectar-se al bus de dades *I2C*, però el *SPI* no treballa en direcció del perifèric, sinó que té una línia de crida *SS (Slave Select)* que permet informar l'esclau quan té que trametre informació. El protocol té 4 línies, *MISO, MOSI, SCK, SS*.

- **MISO (Master In Slave Out):** És la línia usada per l'esclau per trametre les dades.
- **MOSI (Master In Slave Out):** Aquesta és la línia usada pel màster per a trametre dades a l'esclau.
- **SCK (Serial Clock):** Aquesta és el senyal de rellotge per sincronitzar la comunicació.
- **SS (Slave Select):** És la línia que indica quan el màster vol parlar en l'esclau. Pot existir més d'una línia d'aquest tipus per connectar diversos esclaus.

I2C

L'*I2C (Inter-Integred Circuit)* és un protocol de comunicació serial i síncron, que utilitza el senyal de rellotge. Fa servir dues línies, una de dades i l'altra de rellotge. A diferència de *UART*, on el dispositiu és emissor i receptor al mateix temps, ací sols un dispositiu és màster i els altres són esclaus. El màster inicia la comunicació.

Aquest protocol pot treballar en forma de bus, on els dispositius que es connecten a les mateixes línies poden intercanviar dades. Cada dispositiu disposa d'una direcció hexadecimal que l'identifica dins del bus. I un aspecte important és que el protocol treballa de forma *half duplex*, la comunicació sols va en una sola direcció en cada moment.

I2S

El protocol *I2S (Integrated Interchip Sound)*, és de tipus síncron i s'utilitza per a connectar dispositius d'àudio digital. La connexió és similar a *I2C*, ja que per connectar-se a diversos busos de dades fa servir 3 línies: *SCK, FS, SD*

- **SCK (Serial Clock):** Aquesta és la línia de senyal de rellotge.
- **FS (Frame Select):** Aquesta línia s'usa per a seleccionar el canal esquerre o dret.
- **SD (Serial Data):** Aquesta és la línia per on s'envien les dades.

Igual que en *I2C* es deu tenir un dispositiu màster. Aquest protocol es fa servir majorment per a treballar amb àudi estèreo i es pot aconseguir en components d'àudio que utilitzen convertidors *ADC i DAC.*

3.1.2 Assaig del model de *IA*

En la *intel·ligència artificial*[6], no existeix una definició acceptada per tots els experts , ja que és una ciència nova i que va canviant; és experimental. Però una definició simple seria que, la IA és un intent d'imitar la intel·ligència humana utilitzant robots o software. En el 2009 *Stuart Russell i Peter Norvig*, diferenciaren 4 tipus de sistemes informàtics al voltant de la IA:

- **Sistemes que pensen com humans:** Xarxes neuronals artificials.
- **Sistemes que actuen com humans:** Robots.
- **Sistemes que fan servir la lògica racional:** Sistemes experts.
- **Sistemes que actuen racionalment:** Com els agents intel·ligents

La IA no és una ciència nova. *Alan Turing* un expert matemàtic que desxifrà els codic secrets dels nazis amb la màquina Enigma, va iniciar el procés d'intel·ligència artificial moderna.

En 1950 es formalitza l'inici de la IA amb el test de Turing, una prova que defineix si una màquina és intel·ligent o no. Consisteix a fer que un humà i una IA s'enfronten a les preguntes d'un interrogador i si aquest interrogador, no pot distingir si les respostes provenen de l'humà o de la IA , aleshores la IA és intel·ligent.

No va ser fins a 2014 quan se supera per primera vegada el test de Turing[7].

Un altre expert és *Marvin Minsky* que utilitza per primera vegada el termini d'intel·ligència artificial, en 1956. I Minsky va crear el primer simulador de xarxes neuronals per la qual cosa, se'l considera un dels pares de la **Intel·ligència Artificial.**

En primer lloc, una ***IA*** deu aprendre a dur a terme una tasca en concret, si té que reconéixer comandos de veu, deu de processar milers d'àudios. Per començar l'ensinistrament, assaig, li hem de donar diferents àudios de paraules perquè puga determinar com es diferencien, i per això és necessari fer un espectrograma d'un àudio que és com fer una foto a un so. L'objectiu principal és que la **IA** puga treballar a soles, sols passant-li un comando de veu, i que ens diga si el reconeix i quin és o si és un desconegut.

Aquest és un tipus d'estructura d'aprenentatge, l'ensinistrament i resultats, que és el més comú per dur a terme tasques repetitives i mecàniques. Existeixen diferents tipus d'aplicacions d'aquesta teoria, depenent del tipus de IA i la tasca a realitzar, així com: Sistema Expert, Maching Learning, Xarxes Neuronals, Deep Learning.

Tensor FLow

Tensor FLow [8] és una plataforma de codi obert d'extrem a extrem que ens permet l'aprenentatge automàtic. Compta amb un *ecosistema integrat i flexible* d'eines, biblioteques i recursos de la comunitat que permeten innovar amb l'aprenentatge automàtic, i on tot el món pot crear aplicacions amb la tecnologia de AA fàcilment. Ens permet la creació de models d'aprenentatge automàtic, AA, amb una API intuïtiva d'alt nivell com *Keras* que ens permet una interacció de models a l'instant i una depuració fàcil.

Alguna de les coses bones que té *Tensor FLow* és la gran quantitat d'exemples de referència que té, i que et permet començar a investigar per poder desenvolupar el teu model. Una vegada aconseguit un model assajat, ens faltaria l'últim pas, i és fer una conversió de *Tensor FLow a Tensor FLow Lite* que és un *freamwork* per a IoT i dispositius mòbils.

El gran problema que té *Tensor Flow* és la incompatibilitat que té entre versions; d'una versió a una altra existeixen molts canvis que les fan incompatibles, així que s'ha de seleccionar la versió correcta a l'hora de començar a crear i ensinistrar un model. Nosaltres utilitzarem un sistema de xarxes neuronals mitjançant l'ajuda de *Tensor FLow i Tensor FLow Lite.*

3.1.3 Hardware utilitzat

Microcontrolador

Un dels elements més importants és el microcontrolador que hem seleccionat, ja que aquest ens pot limitar a l'hora d'executar el programa, siga per falta de CPU o de memòria. En el mercat existeixen diversos dispositius, des de més econòmics fins als més potents i cars.

ESP32

El *ESP32* [9] és un xip de baix cost i alta integració, ja que compta amb un processador multi nucli, *TCP/IP*, connectivitat *Wi-Fi, Bluetooth* i diverses entrades i eixides disponibles. Està dissenyat per l'empresa xinesa **Espressif Systems** i fabricat per **TSMC** en tecnologia de 40 nm. El seu antecessor es el *ESP8266.*

Especificació	XIPS ESP8266	ESP32
MCU	**ingle-Core 32-bit L106**	**Dual-Core 32-bit LX6**
802.11 b/g/n Wi-Fi	Sí, HT20 (banda de 20 MHz)	Sí, HT40 (banda de 40 MHz)
Bluetooth	No	Sí, versió 4.2 i anteriors
Freqüència típica	80 MHz	160 MHz
SRAM	160 KBytes	512 KBytes
Flaix	SPI, fins a 16 MBytes	SPI, fins a 16 MBytes
GPIO	17	36 volts
Hardware - Software PWM	No / 8 canals	1 / 16 canals
SPI-I2C-I2S-UART	2-1-2-2	4-2-2-2
ADC	de 10 bits	de 12 bits
CAN	No	1
Ethernet MAC	No	1
Sensors Touch	No	Sí
Sensor de Temperatura	No	Sí
Temperatura de treball	-40 °C a 125 °C	-40 °C a 125 °C

Taula 3.1: Taula d'especificacions principals dels xips ESP32 i ESP8266 [9]

ESP32 és el xip processador; això vol dir que tenen diferents plaques amb més *E/S* o menys i amb funcionalitats extres, ja siga mitjançant *Lora*, un *LCD* integrat o diferents elements. Hem investigat, i una empresa anomenada **m5Stack** [10] que ofereix diverses solucions que porten integrat un ESP32, el que ens resulta bastant interessant l*'Atom Echo* [11] que porta incorporat un led *RGB*, altaveu, micròfon i polsador. A més aquesta mateixa empresa ven un mòdul *CAN* per fer una connexió de *plug and play* a falta de fer la programació pertinent.

Arduino

Arduino és un dispositiu basat en codi obert. El microcontrolador que té consisteix en un disseny simple amb un processador *Atmel AVR* amb diferents pins de E/S. A continuació, en la taula 3.2 podem veure la configuració del *Arduino Atmega168*.

Característiques	Atmega168
Voltatge operatiu	**5 volts**
Voltatge d'entrada (recomanat)	7-12 volts
Voltatge d'entrada (límit)	6-20 volts
Pins d'entrada/sortida digital	14(proporcionen PWM)
Pins d'entrada analògics	16
DC pels pins d'entrada/sortida	40mA
Memòria Flash	216KB (2KB pel bootloader)
SRAM	1KB
EEROM	512 bytes
Velocitat de rellotge	16 Mhz

Taula 3.2: Taula d'especificacions principals de **Arduino Atmega168** [12]

Arduino compta amb moltes versions del seu producte, cada una especificada per a un ús específic, amb més o menys E/S, capacitat de processament, memòria, dimensions. Algunes són, *Yún, Due, Mega ADK, Ethernet, Robot, Leonard...* Tots els **Arduino** tenen en comú que es programen mitjançant el *freamwork* propi d'*Arduino IDE*, que és una eina clau a l'hora de desenvolupar software en aquestes targetes.

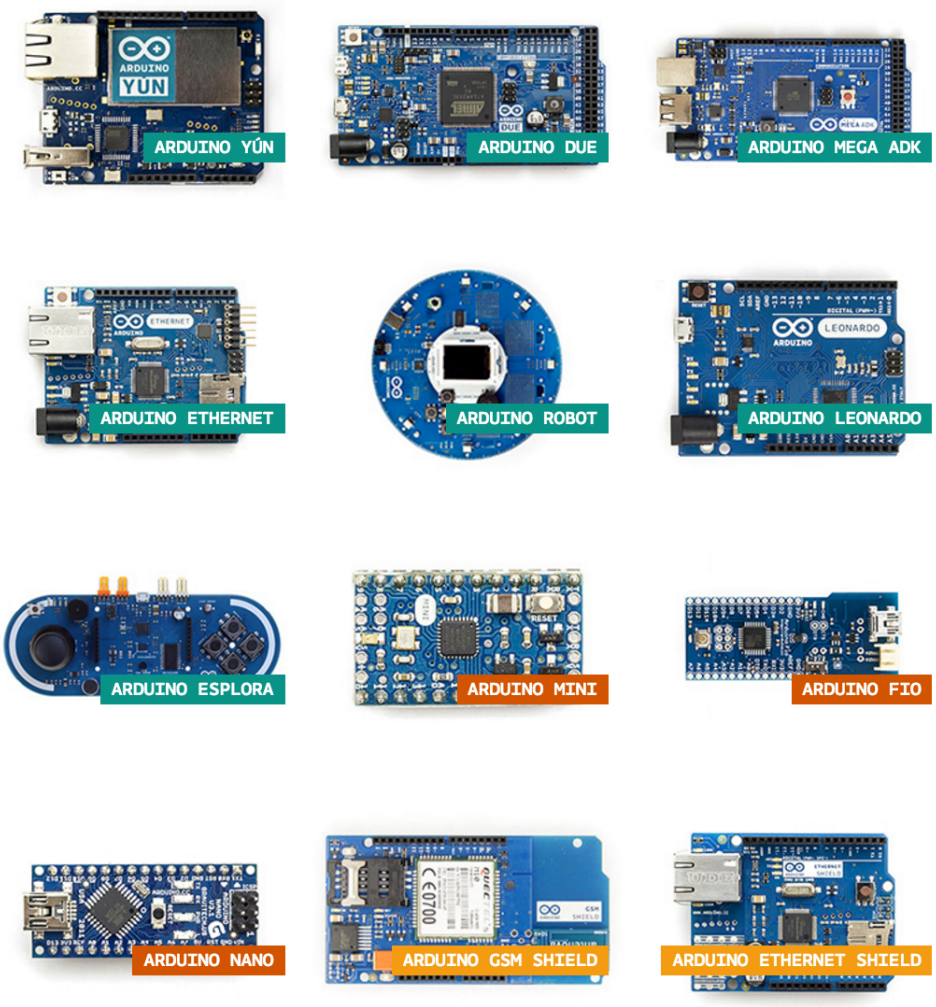


Figura 3.2: Diferents tipus de **targetes Arduino**.

Raspberry

La *Rasberry Pi* és un ordinador de placa reduida o *SBC* (Single-Board Computer) de baix cost desenvolupat al Regne Unit per la fundació Raspberry Pi[13].

L'objectiu d'aquest projecte és estimular l'ensenyament de les ciències de la computació a les escoles. I en àmbit docent, destacaríem que és des de la seua sortida al mercat 2012 ha evolucionat bastant i actualment van per al model ***Raspberry Pi 4 B***. És tracta d'un dispositiu molt potent, ja que està considerat un mini ordinador. No pot realitzar totes les tasques d'un ordinador actual, encara que, a l'hora de considerar-lo com a possible candidat, el que ens agrada és la seva capacitat de processament, que ens a segura que serà capaç de portar endavant qualsevol projecte. També, s'ha de tenir en compte que té un consum molt més elevat que un Arduino o ESP323.

Característiques	Raspberry Pi 4 B
Chip	Broadcom BCM2711
CPU	Quad-Core 1,5 GHz con brazo Cortex-A72
GPU	VideoCore VI
Memòria	11/2/4GB LPDDR4 RAM
Connectivitat	802.11ac Wi-Fi / Bluetooth 5.0, Gigabit Ethernet
Ports	2x USB 3.0, 2x USB 2.0
Alimentació	5V/3A
Expansió	Capçal GPIO de 40 pins

Taula 3.3: Taula d'especificacions principals de **Raspberry Pi 4** [14]

Després de fer un estudi dels microcontroladors actuals, seleccionem l'**ESP32**, però en la versió **ESP32 pico**, que ens proporciona l'empresa **M5Stack**, que té a la venda un dispositiu que compta amb diferents elements integrats com, micròfon, altaveu, interruptor, led RGB i diferents pins de connexió. En la taula següent podem veure totes les característiques.

Característiques	
SoC	ESP-PICO-D4,240MHz,Dual Core,BLE,Wi-Fi
Flash	4MB
Interface	1x IR-TX,1x Function Button,1x Reset Button
PinOut	G21/G25/5V/GND, 3V3/G22/G19/G23/G33
RGB LED	SK6812
Speaker	0.5W/NS4168 I2S
Microphone	SPM1423 PDM
Product Size	24*24*17mm

Taula 3.4: Taula d'especificacions principals d'**ATOM Echo Smart** [15]

Aquest dispositiu connectat amb un mòdul de *CAN* de la mateixa empresa, ens permet poder desenvolupar el projecte.

3.1.4 Requisits

En aquesta secció es defineixen els requisits que deu complir el prototip per a garantir el compliment dels objectius del projecte. Són els següents:

- Captar la informació que grava el micròfon.
- Interpretar el comando de veu, en conformitat amb l'ensinistrat.
- Comunicar-se amb l'ascensor per poder fer un viatge.

Una vegada definits els requisits, podem analitzar les especificacions que hi hem triat, a través d'una taula, tot perquè quede més clar.

Requisits	
Nom	Dispositiu de reconeixement de veu i comunicació CAN
Propòsit	Poder interpretar comandos de veu i realitzar una acció depenent de l'orde.
Entrada	1 micròfon
Eixida	Comunicació CAN
Funcions	Captar la informació que grava el micròfon Interpretar el comando de veu, amb el model entrenat Comunicar-se amb l'ascensor per poder fer una maniobra.
Costos de fabricació	28,63 €
Consum d'energia	Sense especificar
Dimensions físiques	24*48*40mm

Taula 3.5: Taula de les especificacions escollides

3.1.5 Especificacions

Una vegada ja tenim els requeriments hem pogut definir les especificacions. Aquestes ens serveixen com un contracte entre l'empresa i l'alumne per a establir les especificacions, i després validar les accions del dispositiu. En aquest punt és quan es varen definir els objectius del projecte comentats anteriorment, i que - recordem - són: la ***captació de comandos de veu***, la ***identificació de les ordes*** i la ***comunicació CAN amb l'ascensor***.

3.1.6 Casos d'ús

En aquest apartat es tenen en compte els requisits de forma de casos d'ús. En la figura 3.3 els hem recollit esquemàticament.

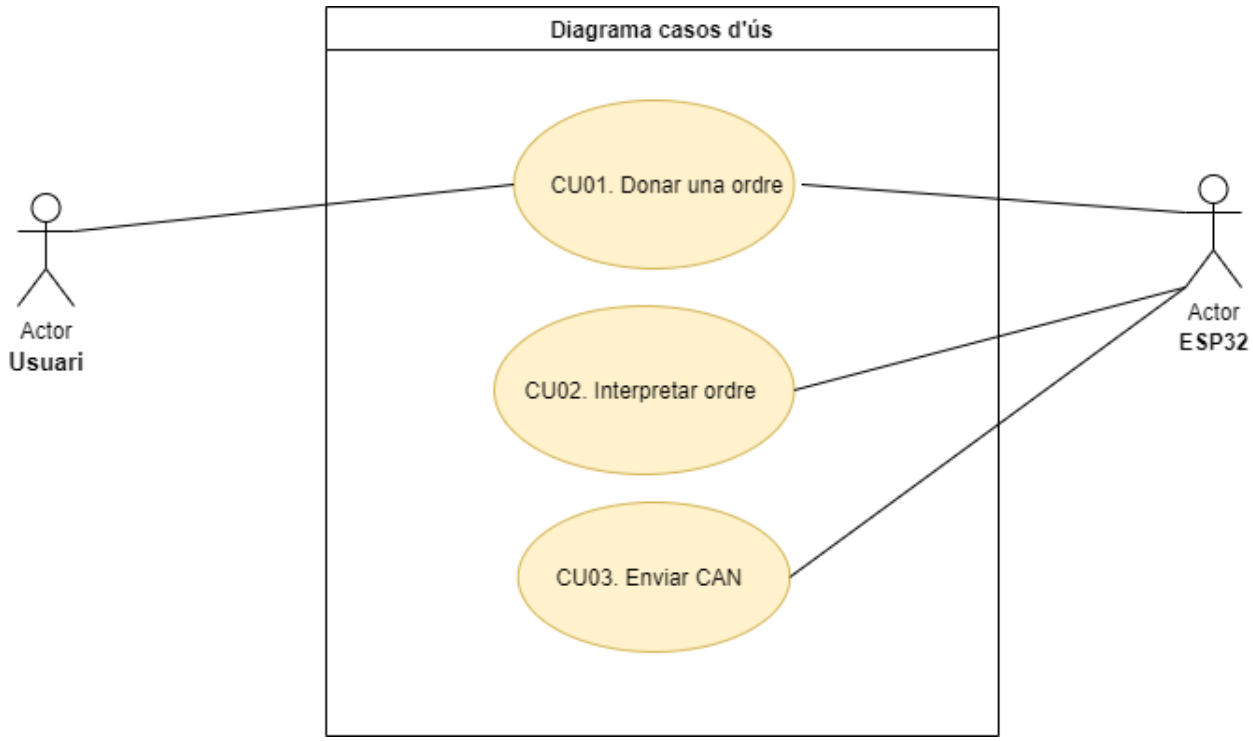


Figura 3.3: Casos d'ús

A continuació podem veure més detallat cada cas d'ús en les següents taules 3.6, 3.7, 3.8.

CU01. Donar una ordre	
Autor: Philippe Gonzalez Miralles Revisor: Diego Centelles Font:Nayar Systems	Data creació: 1/11/2021 Data de revisió: 3/11/2021 Data d'aprovació: 3/11/2021 Versió: 0.1
Descripció: L'usuari li parla diguen la paraula *Nexy* i després la planta on vol anar. Ex: *Clandestino, Uno, Dos, Llamar.* Com començar: Dir *Nexy*, i una vegada el led es fica en verd, diguem la següent paraula.	
Passos: 1. Dir la paraula NEXY. 2. Dir una de les següents paraules: *Clandestino, Planta Baja, Uno, Dos, Tres, Cuatro, Llamar.*	
Observacions: Hem de fer una pronunciació ben correcta, ja que si no pot donar lloc a falsos positius.	

Taula 3.6: Cas d'ús 01.

CU02. Interpretar ordre	
Autor: Philippe Gonzalez Miralles Revisor: Diego Centelles Font:Nayar Systems	Data creació: 1/11/2021 Data de revisió: 3/11/2021 Data d'aprovació: 3/11/2021 Versió: 0.1
Descripció: Una vegada detectat un so, és passa el so pel model prèviament detectat, i el resultat indicarà l'ordre que és. Com començar: Una vegada està el buffer d'informació ple, li passa al model per poder identificar-lo.	
Passos: 1. L'*ESP* detecta un so i l'emmagatzema fins a omplir el buffer. 2. Una vegada aquest ple, el model identifica quina ordre és la capturada.	
Observacions: Si una paraula dura més d'un segon no serà detectada.	

Taula 3.7: Cas d'ús 02.

CU03. Enviar CAN	
Autor: Philippe Gonzalez Miralles Revisor: Diego Centelles Fuente:Nayar Systems	Data creació: 1/11/2021 Data de revisió: 3/11/2021 Data d'aprovació: 3/11/2021 Versió: 0.1
Descripció: Una vegada identificat l'ordre, s'indica a la planta que es vol anar. Com començar: Identificant l'ordre de veu.	
Passos: 1. S'indica la planta on es vol anar. 2. S'envia la trama per CAN. 3. S'espera a que la trama finalitze.	
Observacions: Si el sistema envia una trama i el dispositiu no està connectat es bloqueja.	

Taula 3.8: Cas d'ús 03.

3.2 Disseny de l'arquitectura del sistema

En aquest apartat, es dissenya l'arquitectura en la qual comentem tots els components i la seua relació.

3.2.1 Justificació de la tecnologia utilitzada

Hem seleccionat la targeta **Atom Echo**, que compta amb un *ESP32 pico* perquè era la que més s'adaptava a les necessitats del projecte, ja que compte amb un nucli potent i integra un micròfon. Això ens permet tenir ja seleccionat eixe component i assegurar-nos que és compatible.

A més, i com hem comentat abans, compta amb un mòdul per a la connectivitat *CAN*, que ens facilitarà la comunicació amb l'ascensor. Com que l'ascensor accepta la comunicació *CAN*, hem seleccionat aquesta, ja que és de les més fiables en maquinàries en moviment, *(ja ho hem dit abans)*.

Hem utilitzat el llenguatge de *C++* amb el "framework"d'**Arduino**. Hem seleccionat aquest encara que no és el més potent, però en seleccionar un hardware amb connexions I2S i llibreries pròpies per a fer funcionar diferents elements que incorpora (led RGB, polsador, altaveu i micròfon), ja que ens ha facilitat la feina d'haver de fer la configuració i veure com treballàvem amb ells. Tot i que en el projecte uns dels perifèrics no és el *RGB*, és un element molt útil que ens ha permès verificar el funcionament del dispositiu en diverses vegades.

Per verificar el correcte funcionament de la comunicació *CAN*, s'ha utilitzat un hardware propi de l'empresa que és un convertidor de *CAN* a *USB* amb un cost aproximat de 22€. A baix mostrem un exemple que revela que funciona. Les trames mostrades són dades inventades que no tenen relació, per a res, amb cap informació de l'empresa. De forma esquemàtica:

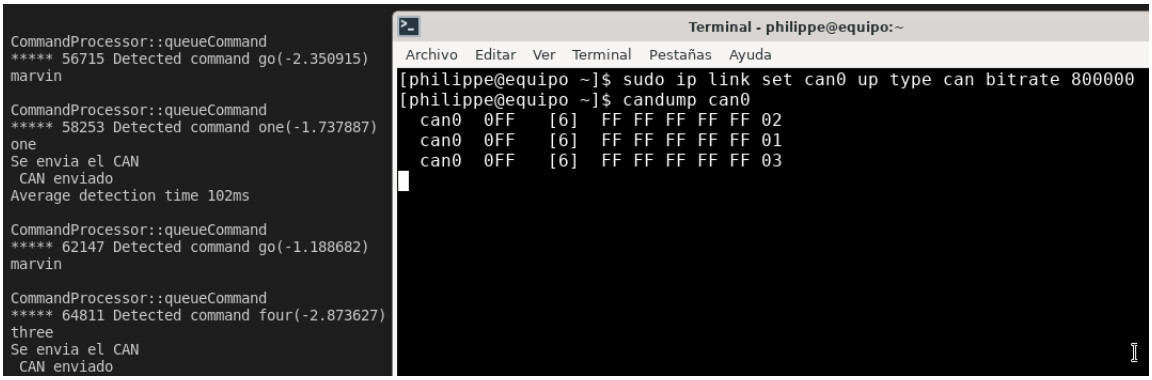


Figura 3.4: Exemple del funcionament CAN

3.2.2 Arquitectura del sistema

L'arquitectura del sistema està organitzada en diferents mòduls, on cadascun s'encarrega de diferents aspectes. Tenim el *Main* que és qui crida a les diferents parts: La part que detecta els comandos, i la part que les executa.

Qui detecta els comandos utilitza el mòdul de captació de sons, mitjançant un *I2S* a més de passar els sons que detecta al mòdul de la *AA* que compara el que rep amb el model i fa una predicció. Depén del model podrà ser més encertada o menys.

Una vegada s'ha detectat el comando, es crida a l'encarregat d'executar les ordes, que és qui es comunica amb el mòdul *CAN* que, depenent d'allò que li passem, farà una acció o una altra.

Aquest sistema està orientat a ser escalable, ja que podem modificar el model molt fàcilment amb un altre set de paraules d'un altre idioma. Sols ha de seguir el mateix ordre que l'original. A més, per la comunicació *CAN* fa servir un fitxer on es troba la configuració necessària per

a configurar i enviar els missatges corresponents, que sols incloent una nova configuració o substituint un fitxer per un altre seria útil per a un altre ascensor de marca i model diferents, o inclús es pot modificar i, en lloc d'utilitzar *CAN*, utilitzar un altre sistema [16].

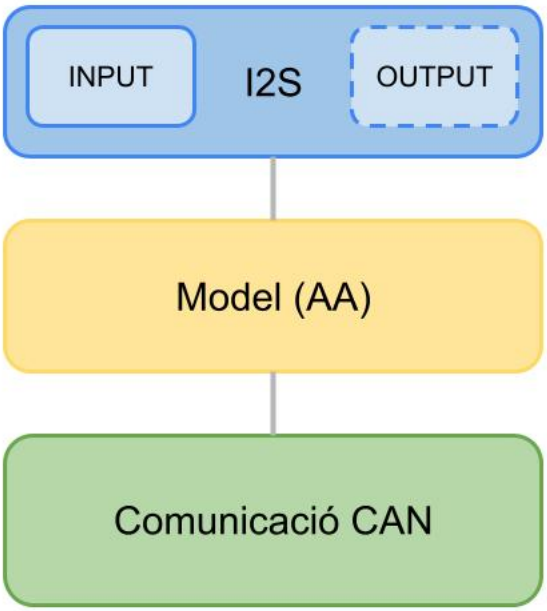


Figura 3.5: Diagrama de l'arquitectura del sistema en blocs

El mòdul de **Detectar àudio**, és l'encarregat d'enllaçar tots els mòduls, primerament amb l'*I2S* per a la comunicació amb el micròfon, i una vegada el sistema detecta un so, és quan captura el so en un buffer fins a omplir-lo. Després aquest buffer li passa al mòdul de **AA**, on és ací on s'interpretarà i ens comunicarà, mitjançant unes puntuacions, si l'àudio és bo i si es tracta d'una de les paraules clau o bé una altra paraula. I detectar que és una de les paraules clau, es comunica amb el mòdul d'acció d'àudio, on, depenent de l'índex que li passem, farà una acció o una altra. En aquest cas es comunica amb el mòdul de *CAN*, i envia un missatge *CAN* depenent de la paraula.

La forma que està organitzada l'arquitectura ens permet fer les següents modificacions. Podem substituir el tipus de micròfon i amb una breu configuració pot funcionar. Respecte a la *AA*, si nosaltres ensinistrem diferents models amb el mateix ordre de les paraules, ens permet substituir el model per un altre, sent així apte per a diversos idiomes. I, per altra banda, en tenir separada l'acció que realitza, ens permet, en un futur, poder substituir l'enviament CAN per una altra acció, siga accionar uns relés o un altre tipus de comunicació.

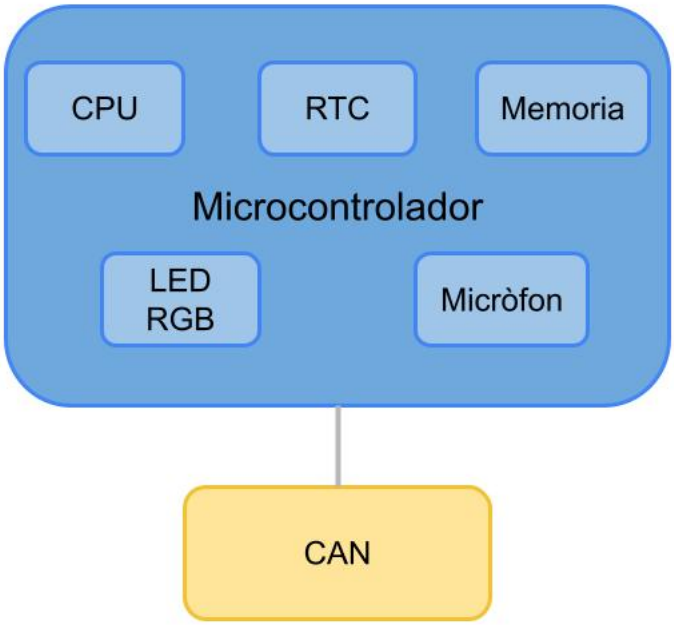


Figura 3.6: Diagrama del HW de l'arquitectura en blocs

Capítol 4

Implementació i proves

Índex

4.1 Detalls d'implementació

4.1.1 Entorn de desenvolupament

Per desenvolupar el projecte s'ha utilitzat el següent software, un entorn de desenvolupament integrat *IDE*, **Visual Studio Code**, versió 1.60.1, ja que és un *IDE* que disposa de molts complements que ajuden a l'hora de programar. Per crear el projecte per a l'ESP32, hem utilitzat el *pluging* de **platformio**, que ens ajuda a crear el projecte des de zero i ens facilita la configuració i la importació de llibreries. El *framework* usat per a programar és el d'**Arduino**, ja que moltes de les llibreries que disposa ens són d'utilitat i ens ha facilitat programar el dispositiu per la seua facilitat comparat amb el *framework* d'**espressif**. A continuació 4.1 tenim la configuració feta servir per a executar el projecte.

```
1 [env:m5stack-atom]
  platform = espressif32
3 board = m5stack-atom
  framework = arduino
5 monitor_speed = 115200
```

Listing 4.1: Configuració basica de **platformio** per executar *m5Atom*

Les llibreries utilitzades per desenvolupar el projecte són les següents:

- *TensorFlow-lite*
- *M5Atom*
- *FreeRTOS*
- *FastLED*
- *CAN*
- *kissfft*
- *I2S*

4.1.2 *TensorFLow*

¿Què és *TensorLFow*? *TensorFlow* és una plataforma de codi obert d'extrem a extrem per a l'aprenentatge automàtic. Compta amb un ecosistema integral i flexible d'eines, biblioteques i recursos de la comunitat que permet que els investigadors innoven amb l'aprenentatge automàtic (AA) i els desenvolupadors creen i implementen aplicacions amb tecnologia d'AA fàcilment.[8].

Aquesta plataforma compta amb diversos exemples i tutorials per aprendre a utilitzar-la, i poder així crear un model propi. A més compta amb un set de dades que pots utilitzar per a provar el teu model. Compta amb un apartat on classifica els diferents projectes que pots realitzar i la seua dificultat, des de xarxes neuronals més simples, fins xarxes generatives adverses, que seria de nivell expert.

Per poder començar a crear un model propi ens permet usar diferents eines, *Jupyter o Colab*. La diferència principal entre aquestes dos, és que *Jupyter* l'executes en local i el *Colab* en remot des d'un servidor de *Google*, cosa que ens pot ser d'utilitat si comptem amb un ordinador modest amb poca capacitat de processament i memòria ram.

En *TensorFLow* hi ha molta varietat d'exemples, un d'ells és per a la classificació d'àudios, on podem veure quins comandos poder fer servir i com hem d'entrenar el nostre model. [17].

```
import tensorflow as tf
import numpy as np
from tensorflow.io import gfile
import tensorflow_io as tfio
from tensorflow.python.ops import gen_audio_ops as audio_ops
from tqdm.notebook import tqdm
import matplotlib.pyplot as plt
```

Listing 4.2: Importacions necessàries per l'ensinistrament del model

L'ensinistrament mitjançant àudios es basa en la comparació d'imatges a través d'un espectrograma, que és el resultat de calcular l'espectre d'un senyal al llarg d'una seqüència temporal. Resulta una imatge que representa l'energia del contingut freqüencial dels senyals segons va variant, al llarg del temps[18]. Obtenim l'espectrograma aplicant la transformada de Fourier en temps curt (STFT) per convertir l'àudio en el domini de freqüència de temps.

La transformada de Fourier converteix un senyal en les seues freqüències de component, però perd tota l'altra informació del temps. El mètode STFT executa la transformada de Fourier retornant una imatge en 2D. EL STFT produeix una sèrie de números complexos que representen la magnitud i la fase, però nosaltres sols necessitem la magnitud que la podem derivar aplicant .tf.abs a la sortida de tf.signal.sstft[19]. En el següent codi 4.3 tenim un exemple del mètode que utilitzem per aconseguir l'espectrograma.

```
def get_spectrogram(waveform):
  # Padding for files with less than 16000 samples
  zero_padding = tf.zeros([16000] - tf.shape(waveform), dtype=tf.float32)

  # Concatenate audio with padding so that all audio clips will be of the
  # same length
  waveform = tf.cast(waveform, tf.float32)
  equal_length = tf.concat([waveform, zero_padding], 0)
  spectrogram = tf.signal.stft(
      equal_length, frame_length=255, frame_step=128)

  spectrogram = tf.abs(spectrogram)

  return spectrogram
```

Listing 4.3: Mètode per obtenir un espectrograma

A continuació podem veure l'espectrograma 4.1 resultant de la paraula dos, aquest so ha sigut gravat en l'empresa **NayarSystems** per a fer una avaluació de l'ensinistrament d'un model amb àudios propis que explicarem més endavant.

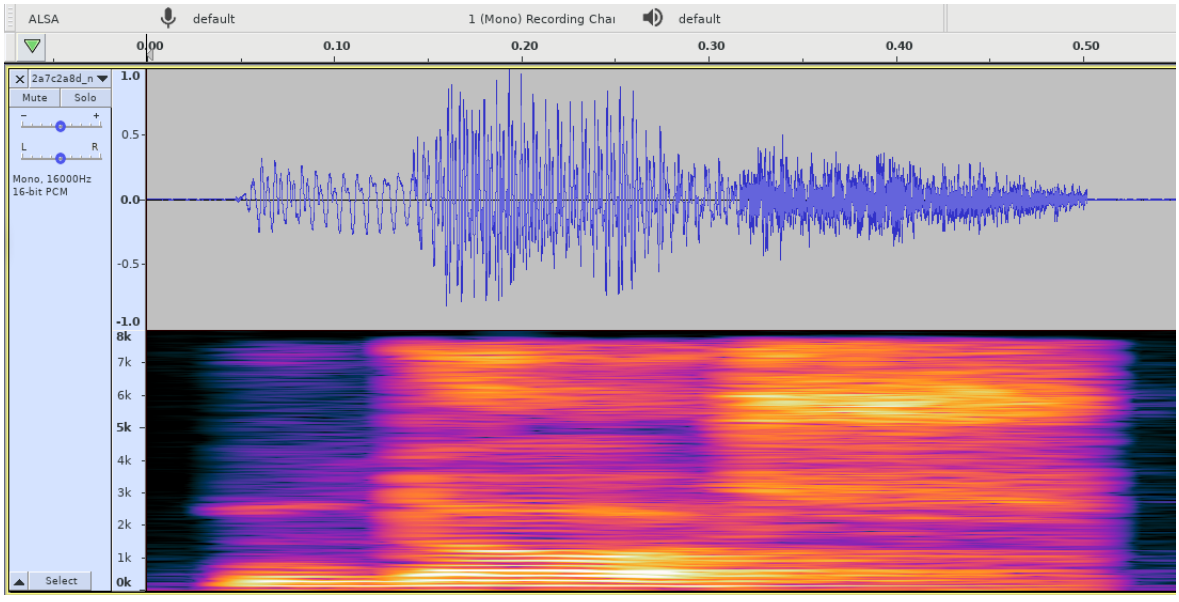


Figura 4.1: Espectrograma generat en pronunciar la paraula **dos** amb el programa *Audacity*.

Per al nostre model utilitzem una xarxa neuronal convolucional (CNN). Aquest tipus de xarxes són semblants a les multicanal, i la seua principal venteja és que cada part de la xarxa s'encarrega d'una tasca; per lo tant redueix significativament el nombre de capes ocultes, motiu pel qual és un model més ràpid. Aquest tipus de xarxa neuronal se solen fer servir principalment a l'anàlisi d'imatges, ja que poden detectar característiques similars d'una a altra[20].

La CNN[21] fa servir diferents tipus de capes o layers. La capa més important és la que li dona nom a aquesta xarxa; és la convolucional. Aquesta capa funciona a partir d'uns filtres de tres dimensions de petit tamany, i que van movent-se per la imatge obtenint així les eixides de la capa. L'objectiu és que la capa recórrega la imatge de dalt a baix reduint la mida de la imatge. En la figura 4.2

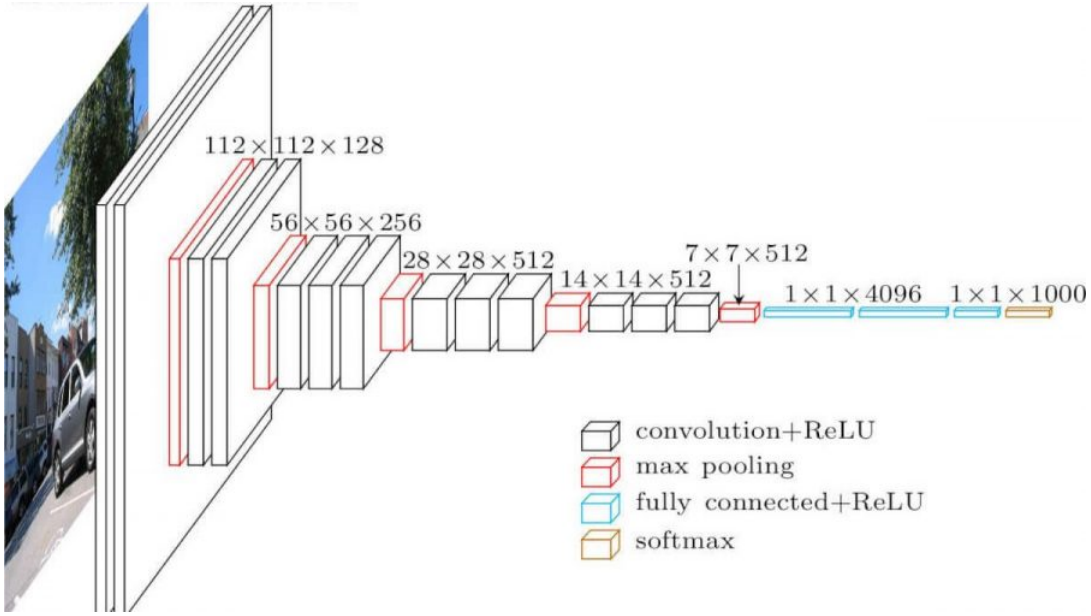


Figura 4.2: Exemple de xarxa convolucional

El més comú és aplicar capes convolucionals seguides de funcions d'activació anomenades *ReLU*. La unitat *ReLu* s'aplica sobre cada valor d'activació obtingut d'un filtre sobre una àrea.

L'objectiu és que, en configurar una xarxa d'aquesta forma, les diferents capes van obtenint una representació jeràrquica de les "features". I com les primeres capes reconeguen elements més simples en una imatge, i les següents obtenint representacions d'alt nivell a partir d'aquests elements simples [22].

La capa de *pooling* és un tipus de capa que està present en una gran quantitat d'arquitectures CNN. La seua utilitat consisteix a reduir les representacions obtingudes, de manera que cada vegada vagen reduint computacionalment el número de paràmetres necessaris. L'última capa de la xarxa és la *fully connected layer*, ja que necessitem una neurona d'eixida per cada objecte que ha de detectar. Normalment, s'utilitza mitjançant una capa *softmax*. Alguna CNN fa servir vàries *fully connected layer* com últimes capes per obtenir les representacions finals després d'una capa de *convolutions + pooling*.

Per poder fer l'ensinistrament del model necessitem un conjunt de dades, i *tensorFLow* ens proporciona un conjunt de dades de diverses paraules gravades a partir de més de 3.000 gravacions per paraula. Amb aquest model hem pogut fer un ensinistrament molt satisfactori amb 20 repeticions per paraula, comptant així amb un conjunt molt gran d'àudios per poder fer l'ensinistrament. Hem utilitzat la següent relació per fer l'ensinistrament, el test i la validació *80, 10, 10.*

Amb aquest mètode 4.4 creem un model convolucional simple amb dues capes de convolució seguides i una capa completament connectada i a la que segueix la capa d'eixida. Aquest és el model que farem servir.

```
model = Sequential([
    Conv2D(4, 3,
           padding='same',
           activation='relu',
           kernel_regularizer=regularizers.l2(0.001),
           name='conv_layer1',
           input_shape=(IMG_WIDTH, IMG_HEIGHT, 1)),
    MaxPooling2D(name='max_pooling1', pool_size=(2,2)),
    Conv2D(4, 3,
           padding='same',
           activation='relu',
           kernel_regularizer=regularizers.l2(0.001),
           name='conv_layer2'),
    MaxPooling2D(name='max_pooling3', pool_size=(2,2)),
    Flatten(),
    Dropout(0.1),
    Dense(
        80,
        activation='relu',
        kernel_regularizer=regularizers.l2(0.001),
        name='hidden_layer1'
    ),
    Dropout(0.1),
    Dense(
        len(command_words),
        activation='softmax',
        kernel_regularizer=regularizers.l2(0.001),
        name='output'
    )
])
```

Listing 4.4: Codi per crear un model convolucional simple

Una vegada ja tenim el model creat, necessitem fer una configuració bàsica abans de fer l'ensinistrament. Amb el model.compile 4.5 fem una configuració, per a optimitzar i reduir les pèrdues.

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'],
)
```

Listing 4.5: Codi per a configurar un model i optimitzar - reduint pèrdues- al model.

Amb el següent codi 4.6 podem fer l'ensinistrament del model, on el primer paràmetre que rep el mètode són les dades, i el segon les dades per a fer les validacions.

```

EPOCHS = 10
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS,
    callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=2),
)
```

Listing 4.6: Codi per a ensinistrar el model

Una vegada ensinistrat el model, podem veure la *matriu de confusió* 4.3 que ens serveix per avaluar el seu ensinistrament. La generem de la següent forma 4.7.

```

confusion_mtx = tf.math.confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mtx, xticklabels=commands, yticklabels=commands,
            annot=True, fmt='g')
plt.xlabel('Prediction')
plt.ylabel('Label')
plt.show()
```

Listing 4.7: Codi per a generar una *matriu de confusió*

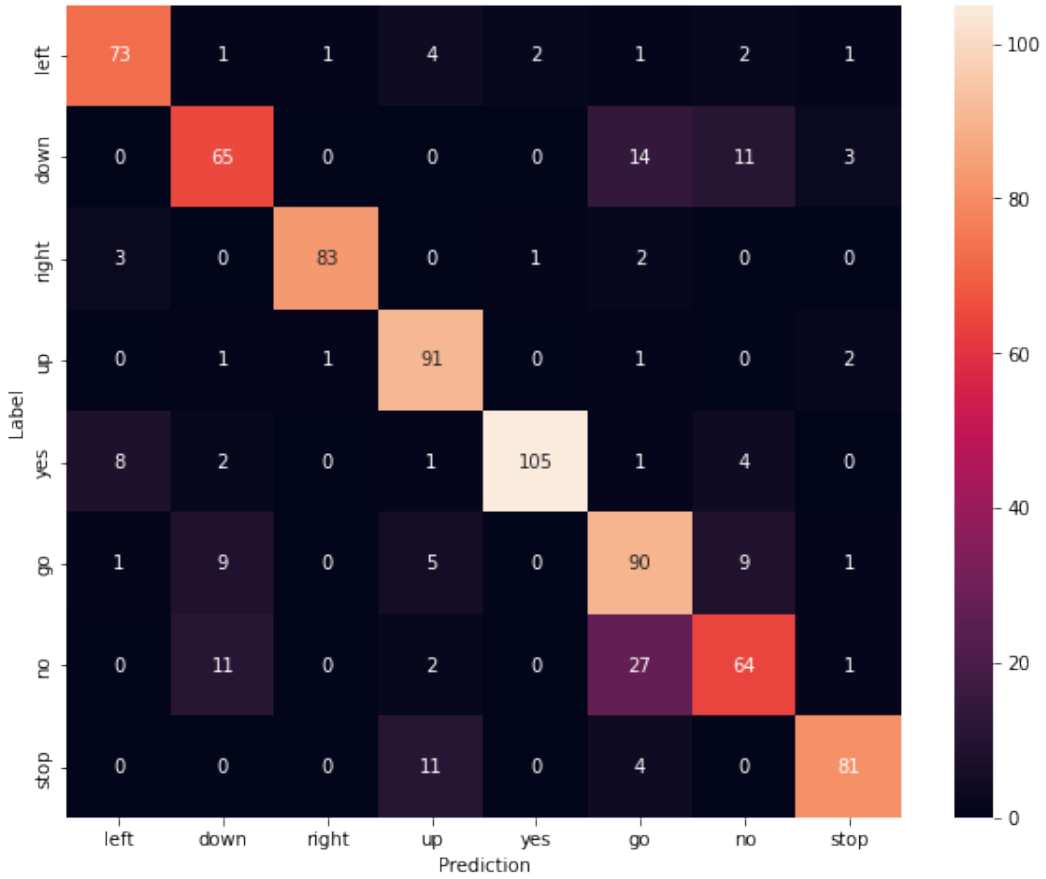


Figura 4.3: *Matriu de confusió* de l'exemple de *TensorFLow*

Una vegada tenim el model ensinistrat sols queda l'últim pas; convertir un model entrenat a *TensorFLow Lite* 4.8. *TensorFlow Lite* és una solució lleugera de *TensorFlow* per a dispositius mòbils i sistemes encastats, que permet implementar models *AA* en dispositius amb una mida binària petita i que suporten l'acceleració del maquinari. En aquestes línies mostrem la conversió, i el fitxer que ens genera el convertim a un fitxer **.cc** pel terminal, com es veu a l'última línia.

```
converter2 = tf.lite.TFLiteConverter.from_saved_model("fully_trained.model")
converter2.optimizations = [tf.lite.Optimize.DEFAULT]

def representative_dataset_gen():
    for i in range(0, len(complete_train_X), 100):
        yield [complete_train_X[i:i+100]]

converter2.representative_dataset = representative_dataset_gen
converter2.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
tflite_quant_model = converter2.convert()
open("converted_model.tflite", "wb").write(tflite_quant_model)

xdd -i fitxer.tflile > fitxer.cc
```

Listing 4.8: Codi per a convertir un model de *TF a TFL*

Una vegada ja tenim el model ensinistrat i convertit a TFL, és quan ja el podem utilitzar en el nostre programa.

Ens hem basat en un exemple simple, on li passem dos números, per tal que ens indique si el segon número és més gran o més menut que el primer. Consisteix en el fet que importem una classe **TensorFLowLite**, al que li indiquem - mitjançant *tflite::GetModel(modelEntrenant)*- el vector d'informació que es troba en el fitxer *model.cc*. Més tard li passem dos números aleatoris, en un vector de n números i en les dos primeres posicions assignem els números, i després amb el mètode *Invoke()* es genera la predicció, i el valor el tenim en l'eixida *output.data.f[0]*. I és així com el model torna un valor float entre 0.0 i 1.0, que ens indica si el resultat és major que 0.5 ens indica que és major que el primer número.

4.1.3 *I2S*

L'*I2S* és el protocol utilitzat per a la comunicació interna entre el micròfon i el dispositiu, ja que el microcontrolador seleccionat el porta incorporat. Per fer servir el micròfon s'ha configurat el mòdul de *I2S* indicant el dispositiu amb el seu identificador, en aquest cas el 0, i a més indicant el *sample-rate a 1600*.

Una vegada configurat amb el mètode, *i2s-read(i2s-port-t i2s-num, void *dest, size-t size, size-t *bytes-read, TickType-t ticks-to-wait)*, comença a captar sons fins a omplir el buffer on després és processat per un altre mòdul en aquest cas pel mòdul comentat anteriorment de *AA*.

En el següent codi 4.9 podem veure un exemple d'un Repetidor proporcionat per l'empresa **M5Stack**. En mètode *loop*, podem veure on es crida al *i2s* i podem observar la forma com ho fa: primer per capturar el so una vegada està pressionat l'interruptor, i després una vegada és soltat, es reprodueix el so. Per canviar d'un mode a un altre, cada vegada es crida la instrucció *InitI2SSpeakerOrMic*, la qual, depenent de la variable entrant, que farà que es configure d'una forma o altra.

```
1  void loop() {
       if (M5.Btn.isPressed())
3      {
           data_offset = 0;
5          InitI2SSpeakerOrMic(MODE_MIC);
           size_t byte_read;
7
           while (1)
9          {
               i2s_read(SPEAKER_I2S_NUMBER, (char *)(microphonedata0 + data_offset
       ), 1024, &byte_read, (100 / portTICK_RATE_MS));
11             data_offset += 1024;
               M5.update();
13             if (M5.Btn.isReleased())
                   break;
15             //delay(60);
           }
17         size_t bytes_written;
           //MODE_MIC 0 & MODE_SPK 1
19         InitI2SSpeakerOrMic(MODE_SPK);
           i2s_write(SPEAKER_I2S_NUMBER, microphonedata0, data_offset, &
       bytes_written, portMAX_DELAY);
```

```
21      }
        M5.update();
23  }
```

Listing 4.9: Codi d'exemple de I2S

En aquest fragment de codi 4.10, tenim el mètode de configuració per al micròfon i altaveu. La majoria de paràmetres són iguals, excepte el mode de configuració que tenen alguns paràmetres diferents.

```
1   void InitI2SSpeakerOrMic(int mode)
    {
3       esp_err_t err = ESP_OK;

5       i2s_driver_uninstall(SPEAKER_I2S_NUMBER);
        i2s_config_t i2s_config = {
7           .mode = (i2s_mode_t)(I2S_MODE_MASTER),
            .sample_rate = 16000,
9           .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT, // is fixed at 12bit,
    stereo, MSB
            .channel_format = I2S_CHANNEL_FMT_ALL_RIGHT,
11          .communication_format = I2S_COMM_FORMAT_I2S,
            .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
13          .dma_buf_count = 6,
            .dma_buf_len = 60,
15      };
        if (mode == MODE_MIC)
17      {
            i2s_config.mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX |
    I2S_MODE_PDM);
19      }
        else
21      {
            i2s_config.mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_TX);
23          i2s_config.use_apll = false;
            i2s_config.tx_desc_auto_clear = true;
25      }

27      err += i2s_driver_install(SPEAKER_I2S_NUMBER, &i2s_config, 0, NULL);
        i2s_pin_config_t tx_pin_config;
29
        tx_pin_config.bck_io_num = CONFIG_I2S_BCK_PIN;
31      tx_pin_config.ws_io_num = CONFIG_I2S_LRCK_PIN;
        tx_pin_config.data_out_num = CONFIG_I2S_DATA_PIN;
33      tx_pin_config.data_in_num = CONFIG_I2S_DATA_IN_PIN;

35      //Serial.println("Init i2s_set_pin");
        err += i2s_set_pin(SPEAKER_I2S_NUMBER, &tx_pin_config);
37      //Serial.println("Init i2s_set_clk");
        err += i2s_set_clk(SPEAKER_I2S_NUMBER, 16000, I2S_BITS_PER_SAMPLE_16BIT,
    I2S_CHANNEL_MONO);
39  }
```

Listing 4.10: Mètode per a la configuració de *I2S*

4.1.4 *CAN*

CAN és el sistema de comunicació utilitzat per a comunicar-nos amb l'ascensor.

Depenent de cada marca i model d'ascensor canvien les dades, la mida i la velocitat a la qual s'han d'enviar els missatges.

Per trametre les dades correctament, s'ha proporcionat la configuració correcta dels ascensors de l'empresa; aquesta informació s'ha obtingut mitjançant enginyeria inversa, i, per tant, és confidencial de l'empresa. Les dades mostrades a continuació són fictícies. I per a fer servir el **CAN** s'ha usat la llibreria que proporciona *m5Stack* per a usar el seu mòdul. A més, per a poder enviar dades cal atendre als 4 paràmetres següents: **l'identificador**, **la mida** de l'identificador perquè existeixen dues mides l'estàndard o l'extensa, **la velocitat** a la qual volem transmetre les dades, i **el missatge**. Amb tot açò ja podem transmetre les dades.

En el programa següent 4.11 podem veure el programari necessari per a poder trametre una trama de mida 6, en l'identificador 0xFF, i la trama 0xFF,0xFF,0xFF,0xFF,0xFF,0x02

```
1     CAN_device_t CAN_cfg;
      CAN_frame_t rx_frame;
3     ESP32Can.CANInit();
      uint8_t *message = {0xFF,0xFF,0xFF,0xFF,0xFF,0x02};
5     rx_frame.MsgID = 0xFF;
      rx_frame.FIR.B.DLC = 6;
7     for (int i = 0; i < tam; i++)
      {
9         rx_frame.data.u8[i] = message[i];
      }
11    ESP32Can.CANWriteFrame(&rx_frame);
```

Listing 4.11: Codi per a enviar una trama CAN

En l'exemple que aquí mostrem 4.4, la comunicació té lloc a una velocitat de 800 KBPS; a més l'identificador s'ha triat que sigue *0FF*, amb una mida estàndard, i un missatge *0xFF,0xFF,0xFF,0xFF,0xFF,0x02* de mida de 6.

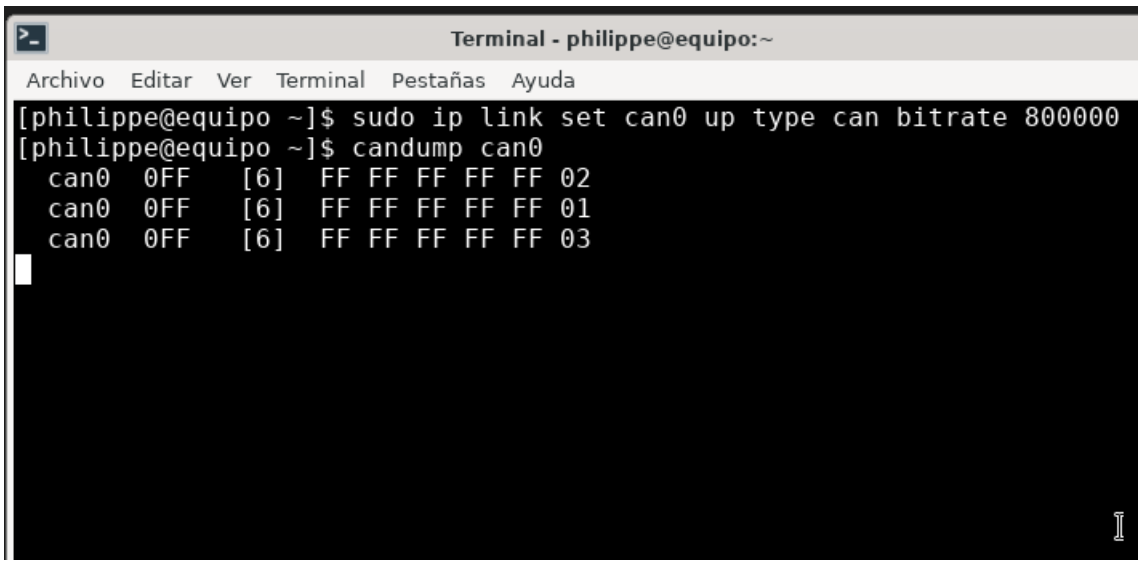


Figura 4.4: Enviament de dades per CAN

Podem veure en la imatge 4.5 el dispositiu blanc que es el *Echo*, que esta damunt del *modul CAN*, el qual està connectat per tres cables a una targeta verda que es el conversor de **CAN** a **USB**

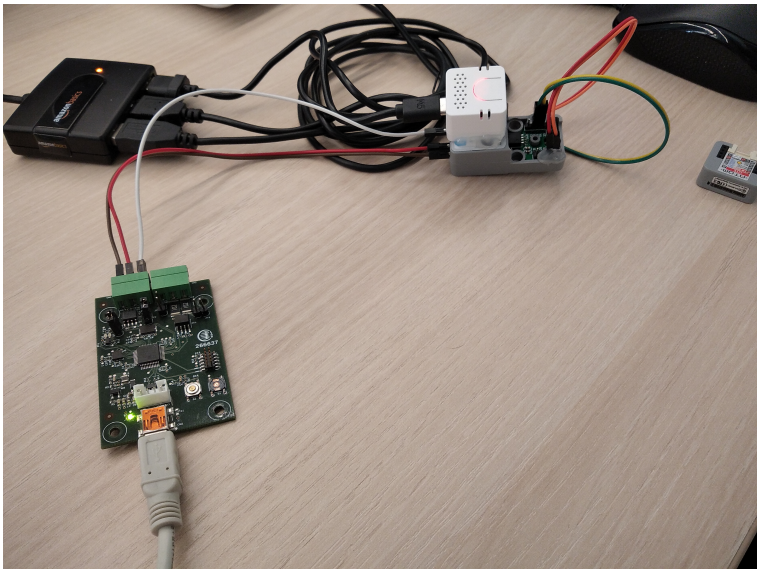


Figura 4.5: Hardware per a la comunicació CAN

Per a poder obtenir aquesta imatge s'ha utilitzat un dispositiu propi de l'empresa, que es un convertidor de **CAN** a **USB**, i que ens permet veure les trames que s'envien sempre que li indiquem a quina velocitat ho volem estar escoltant.

4.1.5 Programa principal

El programa principal on ens permet unificar tots els mòduls del progràma. Per a executar els mòduls s'utilitza *FreeRTOS* [23], que és un sistema operatiu que ens permet la programació multitasca, ja que *Arduino* no ens proporciona aquesta opció, (ja que realment no és multitasca). És recomanable executar les tasques en el nucli 1, ja que el 0 està reservat per a la connectivitat *Bluetooth* i *wifi*, i ja que es poden generar interrupcions i excepcions.

En el nostre projecte s'usen les *Xtask* , en concret la *xTaskCreatePinnedToCore()* 4.12, que rep els paràmetres següents en aquest ordre:

- **Funció** que executarà la tasca.
- **Nom de la tasca**, per poder fer una depuració.
- **Mida de la pila per a la funció**, intercanvi de dades.
- **Paràmetres** de la funció.
- **Prioritat** que tindrà la funció.
- **Handler de la tasca** que ens permetrà treballar amb ella. (Aquest paràmetre, és la direcció de memòria de la variable; hem d'incloure un *ampersand*.)
- **Nucli** en el que volem executar la tasca, que és opcional.

```
xTaskCreatePinnedToCore(
  TaskFunction, // Function to call
  "Taskname", // Name for this task, mainly for debug
  1024, // Stack size
  NULL, // pvParameters to pass to the function
  1, // Priority
  NULL, // Task handler to use
  1 // Core where to run
);
```

Listing 4.12: Constructor d'una *xTaskCreatePinnedToCore*

El programa té el següent cicle d'execució 4.6

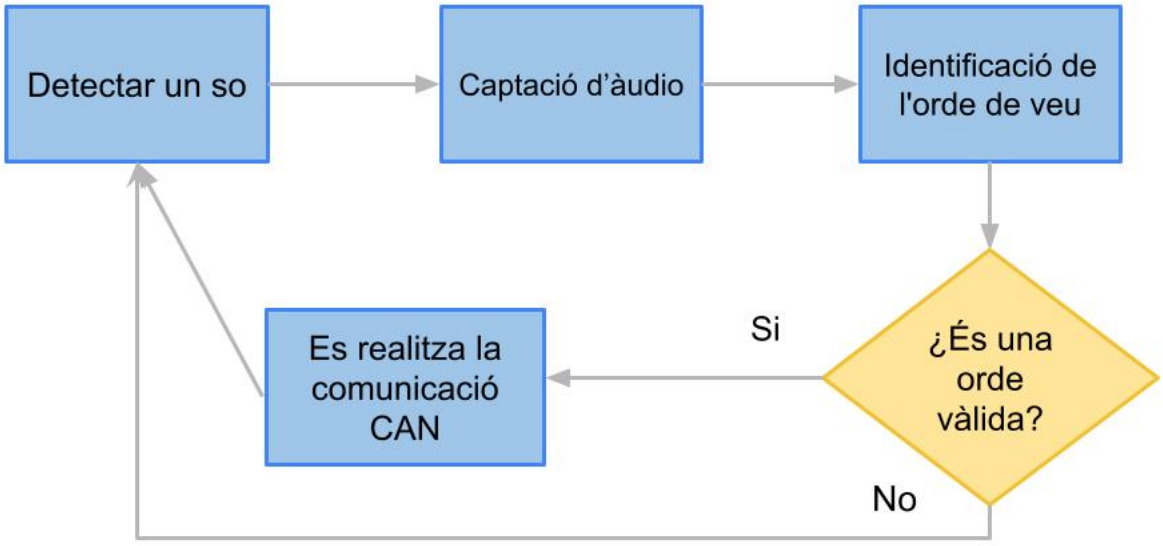


Figura 4.6: Diagrama de flux del programa principal

Com podem veure en la figura 4.6, on representem el cicle d'execució del programa, al primer pas s'espera a detectar un so per activar la tasca que està enllaçada al mètode. Una vegada es detecta un so, automàticament es capta el so fins a omplir el buffer. I una vegada està ple, el model rep la informació del buffer i la interpreta determinant si és una ordre de veu valida o no. En el cas de ser-ho s'envia la seua trama corresponent per CAN; i si no ho és, tornem al principi de l'execució.

Una vegada desenvolupat el programa, podem ensinistrar diferents models, sempre substituint la paraula per un sinònim per a realitzar l'acció; com per exemple, en lloc de dir *uno*, dir *planta uno*. Tot açò ens permet poder entrenar el model en diversos idiomes tenint en compte l'ordre de les paraules, per a facilitar l'escalabilitat de les paraules.

Òbviament, si volem modificar l'acció que realitza una paraula, ja hem d'entrar a modificar el codi.

De les dues maneres, una vegada modificat el programa hem de compilar-lo i pujar-lo al dispositiu perquè funcione amb la nova configuració.

En el *setup()* definim una tasca, *xTaskCreatePinnedToCore*, i aquesta activa el mètode *aplicationTask* que està a l'espera que li es notifique una tasca, amb el *ulTaskNotifyTake* [24] per a poder cridar al mètode de què identificarà l'ordre de veu amb el model i realitzarà l'acció. Des del mètode de captació de so - en el *I2Sampler* una vegada ha detectat un so i ha captat l'ordre en el buffer -, envia una notificació amb *xTaskNotify* [25] que és el que esta esperant el *applicatonTask*.

Després de detectar l'ordre i que el mòdul identifica que el so s'active, és a dir, és quan es crida al model per a poder fer un *predic* del so capturat. Si el model té ensinistrat 4 paraules per a detectar, el *predic* ens tornarà un vector de 5 elements on, en cada element, ens indicara la puntuació de predicció que obté per a identificar quina paraula és la correcta. La cinquena posició és per indicar si no és cap de les paraules que té per a detectar.

Una vegada obtingut el vector de solucions, es fa un estudi per a veure quina posició és la que té major puntuació. L'índex que tinga major puntuació serà la posició de la paraula que està determinant. Exemple, si tenim 4 paraules [*clandestino, uno, nexy , llamar*], i el sistema ens indica l'índex 4, vol dir que el so que ha analitzat no és cap de les paraules vàlides; però si, en canvi, indica la posició 2, significa que ha detectat la paraula *nexy*. Una vegada detectat el so, i si és un índex vàlid, se li transmet al mòdul l'ordre de realitzar una acció, així com quin és l'índex detectat, i el mòdul realitza l'acció; en aquest cas, enviar una trama CAN. Tot açò està representat en la següent figura 4.7

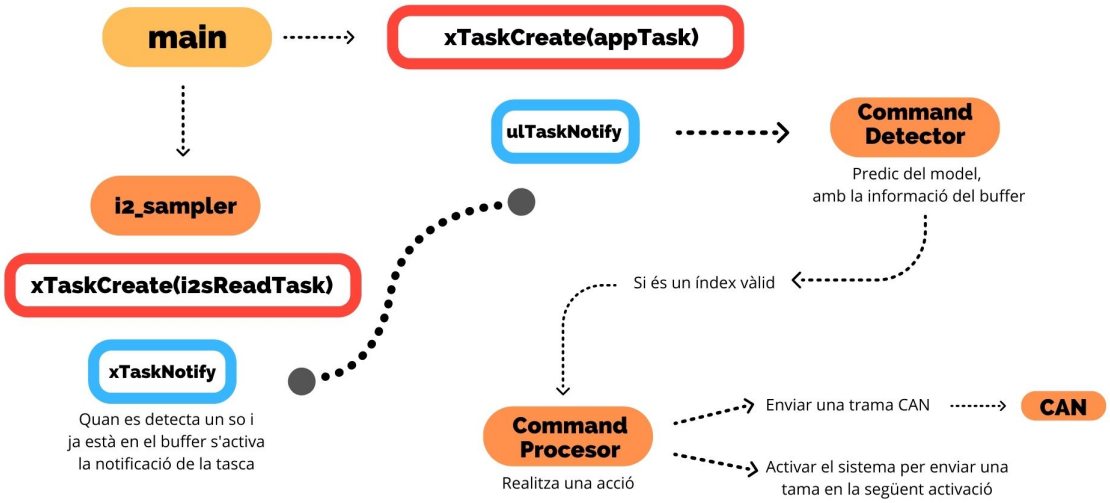


Figura 4.7: Esquema del funcionament del programari

4.2 Verificació i validació

Per a verificar el projecte i validar-lo, s'ha tractat de fer de la manera següent: a través dels tres mòduls que n'hem parlat. Com hem comentat abans, el projecte compta amb tres mòduls principals que s'han desenvolupat de forma individual i que després s'han connectat. Els dividim en: *micròfon, el model, i la comunicació CAN.*

4.2.1 Micròfon

Aquest mòdul compta amb la configuració prèvia del *I2S*. Una vegada configurat s'ha provat el seu funcionament amb programes simples de gravar un so i després reproduir-lo. Aquest programa està desenvolupat per la mateixa empresa del dispositiu [10]. Tot açò ens ha permés comprovar la qualitat del micròfon, un *SPM1423 PDM* , i del altaveu per a veure si el primer és suficient per a la captació d'ordres de veu.

4.2.2 Comunicació CAN

La comunicació **CAN** s'ha verificat de dues formes, la primera mitjançant el dispositiu propi de l'empresa **Nayar Systems**, que es pot veure en la figura 4.5, per comprovar les trames que s'envien. Amb aquesta prova vàrem verificar la configuració i que tot estava funcionant conforme a la configuració correcta.

I la segona prova fou realitzada per a verificar el seu funcionament, i es realitza amb una torre d'una cabina d'ascensor de la marca **Edel**. Amb dita cabina 4.8 vàrem poder verificar que les trames eren correctes, a l'acceptar la comunicació i fer un viatge entre plantes.

Figura 4.8: Torre d'una cabina de la marca Edel

4.2.3 Model

El model resultant de l'ensinistrament es pot verificar de dues formes: mitjançant *la matriu de confusió* i *provant el model veient si detecta les ordres de veu.* Per a realitzar aquesta última comprovació ha sigut necessari tenir tot el sistema desenvolupat, o almenys la part del micròfon per a poder comptar amb una entrada d'informació.

Amb *La matriu de confusió*, hem realitzat diverses proves, però ens hem trobat amb un inconvenient i és el temps d'execució per a aconseguir un model ensinistrat. I és que amb un ordinador modest no hem estat capaços d'ensinistrar un model gran. Quan parlem de gran ens referim a un set de paraules variades, amb unes 3.000 gravacions per paraules per a poder disposar d'una gran varietat de mostres.

Així que en primera instància vàrem optar per ensinistrar el model amb un set de paraules de Google que compta amb 35 paraules: *five, left, tree, backward, follow, right, two, bed, forward, marvin, seven, up, bird, four, nine, sheila, cat, go, no, six, visual, dog, happy, off, stop, wow, down, house, on, yes, eight, learn, one, three, zero.*

Com que per a ensinistrar un model amb tantes paraules necessitàvem molts recursos, vàrem decidir fer-ne una selecció, i ens vàrem limitar a 20 paraules. Les seleccionades foren: *one, two, three, four* i després hem agafat unes 16 paraules de comparació per poder fer un bon ensinistrament, i que són aquestes: *zero, five, six, seven, eight, nine, marvin, forward, backward, left, right, stop, go, on, off,up.*

Per fer aquest ensinistrament hem repetit el set de paraules unes 20 vegades, per tal de poder obtenir més mostres encara; Així vàrem aconseguir 60.000 àudios per paraula perquè siga suficient per a tenir un model eficient.

I una vegada ensinistrat, vàrem comprovar, amb la matriu de confusió 4.9 que els resultats foren molt acceptables: tots tingueren una puntuació del 0.95 com a màxim, i com a mínim del 0.93.

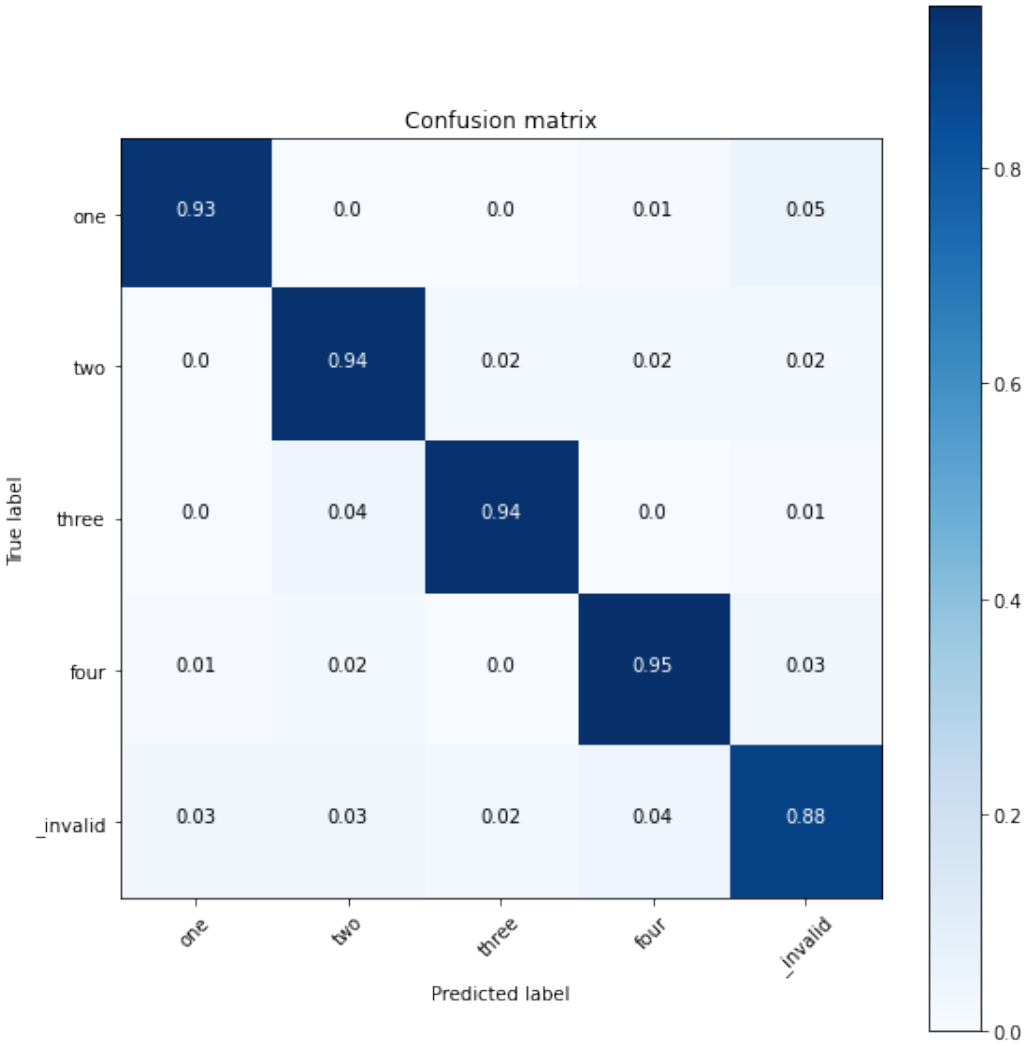


Figura 4.9: Matriu de confusió de les paraules one, two, three, four

Una vegada convertit el model i exportat al projecte, vàrem provar aquest model per a veure si ens reconeixia les ordres de veu. Ho vàrem provar amb un usuari amb un nivell baix de pronunciació, i el model fou capaç de reconéixer les ordres de veu i funcionar correctament, veient que com millor és la pronunciació millor funciona.

Vàrem deduir, amb aquestes proves que si les paraules a reconéixer són fonèticament prou diferents és més fàcil reconéixer-les, i també si es comparàvem paraules monosíl·labes i bisíl·labes. Tot açò ens plantejà la idea de gravar un set de paraules pròpies per a veure què passaria, quin seria el número de gravacions mínim necessari i què passaria, ¿El model seria capaç de funcionar? ¿Quin és el numero mínim de repeticions? Així que una vegada plantejat aquestes incògnites, vàrem pensar en un model de 21 paraules per tindre suficient varietat per a poder fer un ensinistrament i que tinga èxit.

Per a determinar aquestes 21 paraules vàrem considerar les qüestions que anunciem tot seguit: primer que, tot pensant en l'empresa, l'ascensor té concretament 6 parades, i, per tant, que necessitàrem una paraula per a cada planta; a més, en caldria una altra que fes d'activador;

que la funcionalitat podria ser similar a *Alexa* d'Amazon; i , per últim, que era precisa una paraula per a cridar l'ascensor perquè vingues a la planta on ens trobéssim. Tot açò suposava establir un model de 9 paraules, així que vàrem realitzar una recopilació de diferents per a veure quines podíem utilitzar després. Set de paraules resultant fou:

- Planta -1
  - Planta menos uno
  - Menos uno
  - Clandestino
  - Planta Clandestino
  - Futbolin
- Plana 0
  - Planta bajà
  - Planta cero
  - Cero
- Planta 1
  - Planta uno
  - Uno
- Planta 2
  - Planta dos
  - Dos
- Planta 3
  - Planta tres
  - Tres
- Planta 4
  - Planta cuatro
  - Cuatro
  - Terraza
  - Roof
- Paraula d'activació
  - Nexy
- Paraula per a cridar-lo
  - Ven
  - Ven aquí

– Llamar

La idea era disposar de diverses paraules per a identificar una planta i realitzar una acció, i tindre la paraula *Nexy* com activador per impedir falsos positius, ja que un fals positiu és fàcil que es produeixi en fer servir l'ascensor. En canvi, que es produeixin dos o més falsos positius en un període curt de temps és més difícil. De moment sols utilitzarem una paraula per a cada acció, partint d'un model de 9 paraules.

La gravació de totes les paraules es realitza mitjançant un telèfon mòbil **Xiaomi Redmi 8x** per l'aplicació de la gravadora. Els usuaris gravats foren els treballadors de l'empresa que es trobaven físicament en les oficines, en total unes 32 persones: de 23 homes entre les edats de 22 i 45 anys i 9 dones entre les edats de 24 i 46 anys. Tot això ens ha permès obtindre uns 32 àudios per paraula amb un total de 672 àudios.

Una vegada classificats els àudios, vàrem realitzar l'ensinistrament del model. I de sobte ens trobarem, amb diferents problemes que no ho havíem considerat prèviament; el nom dels arxius no té per contenir espais, i a més, els àudios que calia a processar havien de ser exactament d'1 segon de temps i de tipus mono, no poden ser estéreo i a 16000 Hz. Tot açò és per a reduir la mida i proporcionar un número de *mostres* similars per a més tard fer l'espectrograma. Açò fora un contratemps, ja que ens porta molt de temps adaptar els àudios per fer l'ensinistrament.

Tot seguit podem veure la matriu de confusió 4.10 de les paraules *uno, dos, res, cuatro*, podem apreciar que ens trau resultats prou elevats, encara que comptem amb poques mostres per fer l'ensinistrament.

En la figura 4.9 podem observar la matriu de les mateixes paraules en anglés - que tenen una mostra de dades molt superior a aquesta - i els resultats són similars. Per tant, després de veure la matriu s'ha provat en el dispositiu i encara que es produeixen falsos positius, la majoria de vegades el sistema prediu la paraula correcta.

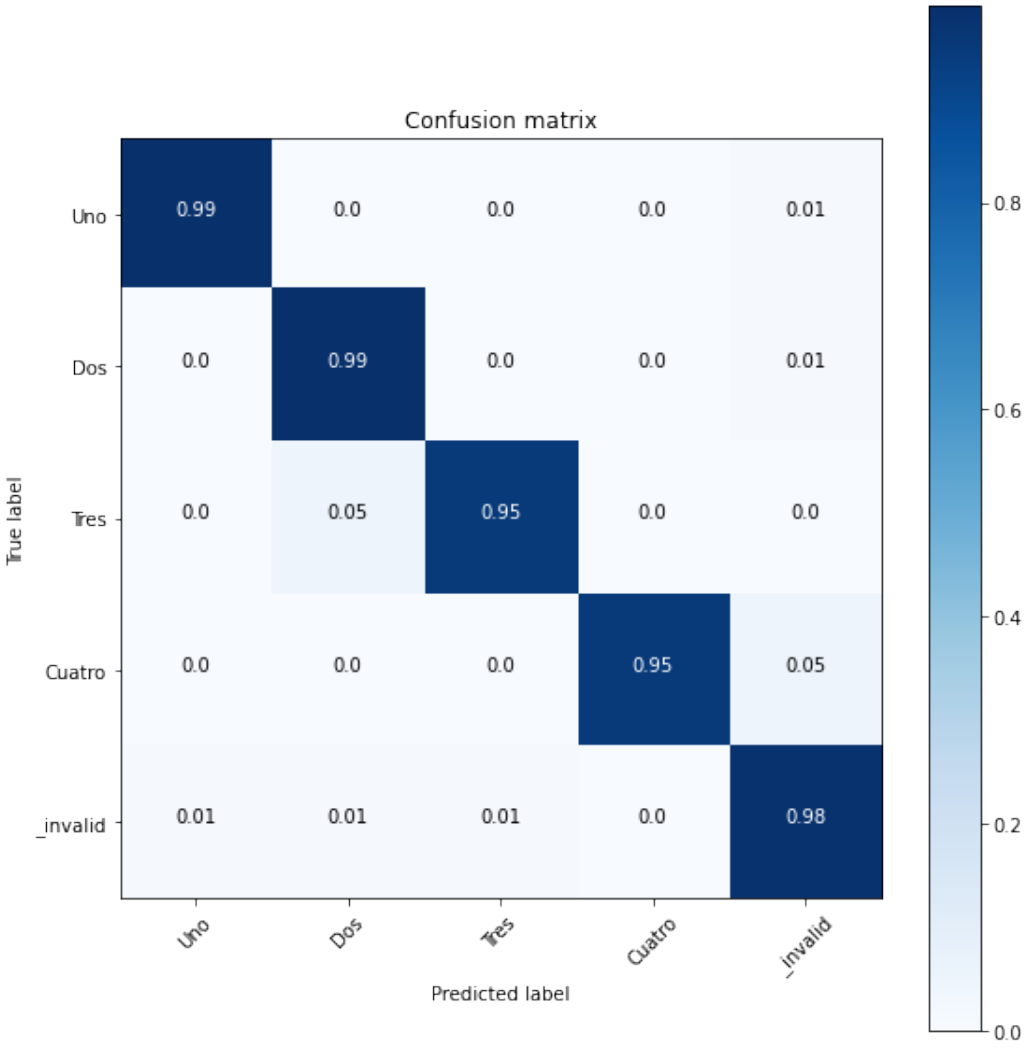


Figura 4.10: Matriu de confusió de les paraules uno, dos, tres, cuatro

Una vegada provat el sistema, intentarem ensinistrar una xarxa més gran, amb el set de paraules següent *MenosUno, PlantaBaja, Uno, Dos, Tres, Cuatro, Marvin, Nexy i ven*, amb repeticions de 300 vegades, i obtenint uns 6400 àudios. Per a fer l'ensinistrament, remarquem que són menys que el model en anglès anteriorment comentat. El resultat el tenim en la següent matriu de confusió 4.11.

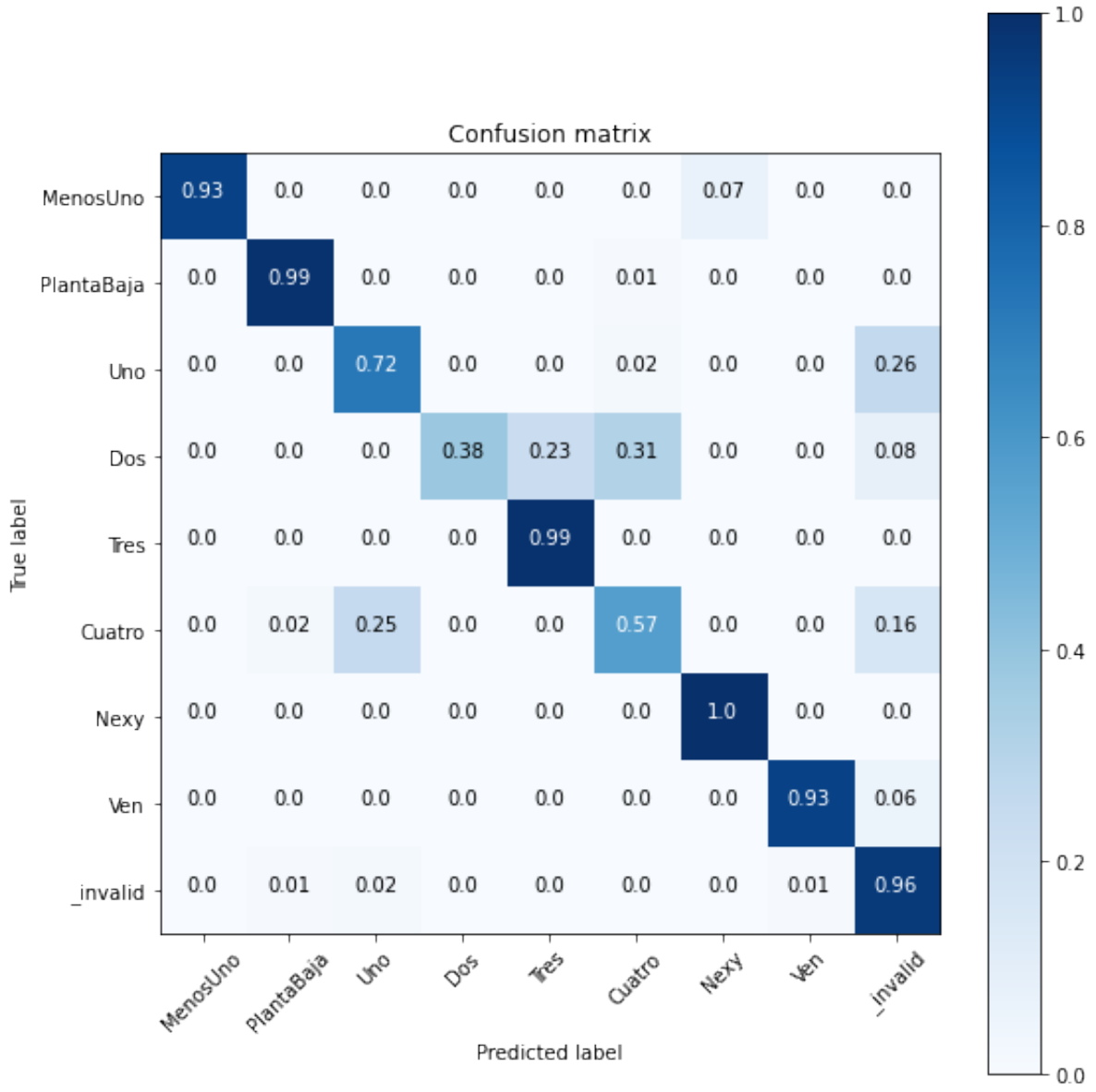
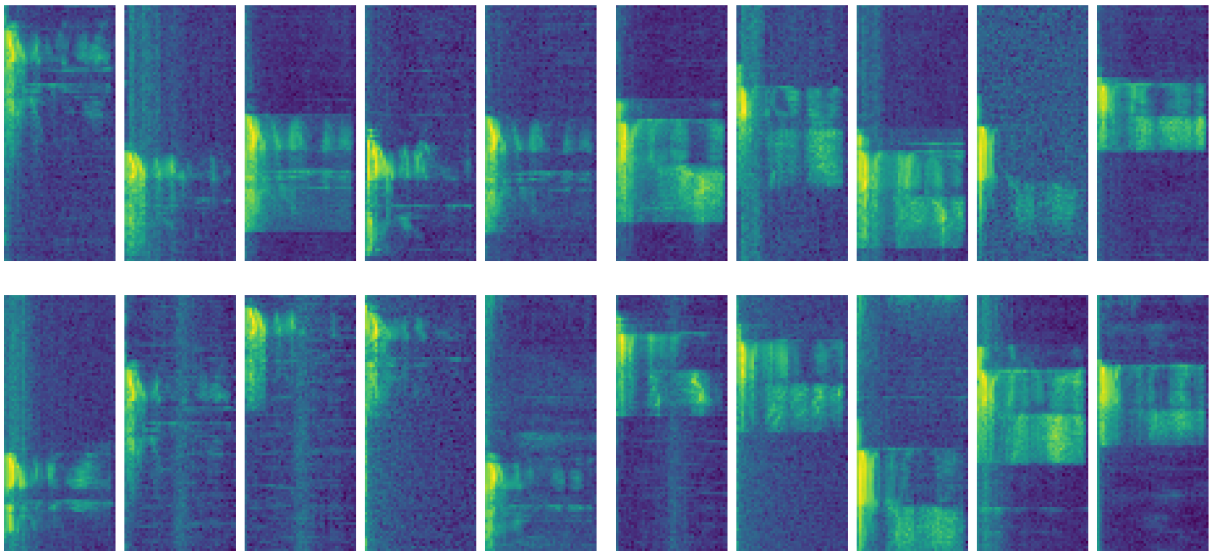


Figura 4.11: Matriu de confusió del model propi

Podem observar diverses coses en la matriu, al ser un model gran és més difícil obtenir bons resultats per a tots els encreuaments de paraules idèntiques. És el cas de la paraula *dos*, que la confon amb la *cuatro*, ja que els espectrogrames d'aquestes dues acaben de manera molt semblant. Ho podem veure en la figura. 4.12.



(a) Espectrograma paraula cuatro (b) Espectrograma paraula dos

Figura 4.12: Comparació entre els espectrogrames de les paraules *dos* i *cuatro*

Capítol 5

Conclusions

Índex

Aquest projecte està dins del departament d'informàtica en la secció d'*Embedded* de **Nayar Systems**, té com a objectiu ser un prototip d'un sistema que en un futur es puguen implantar en algun dels seus productes actuals.

El prototip és capaç de detectar diferents paraules i comunicar-se mitjançant connexió **CAN** amb l'ascensor.

5.1 Àmbit formatiu

En l'àmbit formatiu, amb el projecte presentat he pogut aplicar els coneixements adquirits durant el grau d'Enginyeria Informàtica i en especial a les assignatures de *EI1028 Sistemes Intel·ligents* i a la *EI1062 disseny de sistemes encastats en temps real.*

El desenvolupament d'aquest projecte m'ha permés aprendre com es treballa en una empresa tecnològica, i adaptar-me al seu funcionament per a veure com es facen les diferents tasques d'organització i desenvolupaments en els projectes.

Utilitzar una *ESP32* és una cosa que 'no em pilla de nou', ja que prèviament a les pràctiques ja havia trastejat amb ell, en l'associació d'estudiants d'*UJI MotorSport* per a la captació de dades de diferents sensors per al seu estudi.

5.2 Àmbit professional

En l'àmbit professional ha sigut una experiència molt bona i recomanable a tot el món. He aprés com hi funcionen internament i m'hi han ensenyat tecnologies que desconeixia així com la forma correcta d'estructurar el projecte.

Per a l'optimització del projecte s'ha arribat a la conclusió que dona igual si hi treballem amb un set gran de gravacions o no, ja que mentre només en calgui classificar unes poques paraules - com el 4 dispositiu - funciona molt bé, amb un encert del 0.92 per cent. Ara, quan volem incrementar aquest número de paraules i les volem que reconega és quan comencen els problemes perquè el sistema es confon algunes paraules que té a reconéixer i causa falsos positius. En aquests casos sí que és important comptar amb un nombre gran de gravacions per paraula.

5.3 Àmbit personal

En aquest àmbit recomane realitzar les pràctiques en **Nayar Systems**, ja que m'han pogut tutoritzar molt bé, i resolt els dubtes de seguida, per a poder avançar en el projecte. En la companyia hem sentit molt ben acollit, així que des del meu punt de vista és una molt bona empresa per realitzar aquest tipus de pràctiques formatives. Pel tema de la covid-19, no és un aspecte a tenir en compte, ja que actualment les restriccions sols ens obliguen a portar mascareta als interiors. Açò, per tant, ens ha permés poder dur a terme les pràctiques de forma presencial i reunions setmanals sense problema.

5.4 Millores del projecte

Les millores d'aquest projecte entenem que són, les separem en dos blocs: respecte a les millores dels programes que articulen el model; i pel que fa a la comunicació d'aquest amb l'ascensor.

Les millores del programari, serien considerant sempre un model amb més mostres gravades de cada paraula, per a permetre que funcione un model més gran, possibilitant, sobre tot, l'ús de diversos idiomes a la vegada i una confirmació tant auditiva com visual a part de la que pot proporcionar el mateix ascensor.

Quant a les millores respecte a la comunicació amb l'ascensor, una molt considerable seria la implementació en el sistema, d'un producte propi de l'empresa, ***botonera virtual***, que és una *esp32* amb diferents relés que va connectada directament als botons de la cabina. Si en aquest producte s'incorporara, a més, un micròfon, es podria fer una migració del sistema a aquest producte. Seria una versió 2.0 del producte que ja tenen amb noves funcionalitats. Així on tenen instal·lat aquest producte seria molt fàcil fer un *upgrade* del sistema.

Una altra millora seria modificar el sistema perquè es connecte a una xarxa wifi que podria generar el *GSR*, un producte propi de l'empresa que instal·len en els ascensors per a verificar el funcionament i obtenir dades per telemetria, i que es comunicara per poder indicar-li la planta

on vols anar. Així quan es detecta un so i es detecte l'ordre es comunique amb el *GSR*, i aquest realitze l'acció. Tot açò permet que el sistema fora molt fàcil d'instal·lar, ja que sols requeriria una xarxa wifi.

Bibliografia

[1] *Botonera Virtual Accesible*. [Consulta: 11 de Març 2022]. URL: https://www.nayarsystems.com/que-ofrecemos/botonera-virtual-accesible/.

[2] *Hyundai: IA y reconocimiento de voz en sus ascensores*. [Consulta: 11 de Març 2022]. URL: https://ascensores-montacargas.com/hyundai-ia-en-sus-ascensores/.

[3] Yan Liu Wenqing Wang Yanwei Li. *Realization of Contactless Elevator Control Panel System Based on Voice Interaction Technology*. 2021 3rd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA). IEE, 2021. ISBN: 978-1-6654-3981-7.

[4] *Protocolos de comunicación usados en microcontroladores*. [Consulta: 27 de Octubre 2021]. URL: https://www.rjconcepcion.com/podcast/protocolos-de-comunicacion-usados-en-microcontroladores/.

[5] *¿Qué es el CAN BUS y cómo funciona?* [Consulta: 27 de Octubre 2021]. URL: https://www.ingenieriaymecanicaautomotriz.com/que-es-el-can-bus-y-como-funciona/.

[6] *Inteligencia artificial: qué es, cómo funciona y para qué se utiliza en la actualidad*. [Consulta: 28 de Octubre 2021]. URL: https://computerhoy.com/reportajes/tecnologia/inteligencia-artificial-469917.

[7] *Inteligencia artificial supera tras 65 años Test de Turing*. [Consulta: 28 de Octubre 2021]. URL: https://computerhoy.com/noticias/software/inteligencia-artificial-supera-65-anos-test-turing-14043.

[8] *Tensor Flow*. [Consulta: 28 de Octubre 2021]. URL: https://www.tensorflow.org/.

[9] *Documentació ESP32*. [Consulta: 28 de Octubre 2021]. URL: https://www.espressif.com/en/products/socs/esp32/resources.

[10] *m5stack*. [Consulta: 28 de Octubre 2021]. URL: https://m5stack.com/.

[11] *Atom Echo*. [Consulta: 28 de Octubre 2021]. URL: https://shop.m5stack.com/collections/atom-series/products/atom-echo-smart-speaker-dev-kit.

[12] *Documentació Atmega168*. [Consulta: 28 de Octubre 2021]. URL: https://docs.rs-online.com/c852/0900766b8166ec43.pdf.

[13] *Raspberry Pi Fundation*. [Consulta: 28 de Octubre 2021]. URL: https://www.raspberrypi.org/.

[14] *Raspberry Pi 4 ya a la venta: características y precio oficial de sus 3 versiones*. [Consulta: 28 de Octubre 2021]. URL: https://hardzone.es/2019/06/24/raspberry-pi-4-caracteristicas-precio/.

[15] *ATOM Echo Smart Speaker Development Kit.* [Consulta: 29 de Octubre 2021]. URL: https://shop.m5stack.com/collections/atom-series/products/atom-echo-smart-speaker-dev-kit.

[16] Brito-Loeza Carlos Espinosa-Romero Arturo Martin-Gonzalez Anabel Safi Asad. *Intelligent Computing Systems Third International Symposium.* Cham : Springer International Publishing. Springer, 2020. ISBN: 3-030-43364-1.

[17] *Reconeixement d'àudio simple: reconeixement de paraules clau.* [Consulta: 15 de Novembre 2021]. URL: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/audio/simple_audio.ipyn.

[18] *¿Que es un Espectrograma?* [Consulta: 15 de Novembre 2021]. URL: https://ccrma.stanford.edu/~jos/mdft/Spectrograms.html.

[19] *tf.signal.stft.* [Consulta: 15 de Novembre 2021]. URL: https://www.tensorflow.org/api_docs/python/tf/signal/stft.

[20] *Red Neuronal Convolucional CNN.* [Consulta: 15 de Novembre 2021]. URL: https://www.diegocalvo.es/red-neuronal-convolucional/.

[21] *Qué son las Redes Neuronales Convolucionales.* [Consulta: 16 de Novembre 2021]. URL: https://www.iebschool.com/blog/redes-neuronales-convolucionales/.

[22] Chua-Leon-O Roska-T. *Cellular neural networks and visual computing : foundation and applications.* Cambridge : Cambridge University Press. Cambridge, 2015. ISBN: 0-521-65247-2.

[23] *¿Qué me ofrece FREERTOS?* [Consulta: 19 de Novembre 2021]. URL: https://www.electrosoftcloud.com/freertos-en-esp32-esp8266-multi-tarea/.

[24] *Documentació de FreeRTOS de ulTaskNotifyTake.* [Consulta: 29 de Novembre 2021]. URL: hhttps://www.freertos.org/ulTaskNotifyTake.html.

[25] *Documentació de FreeRTOS de xTaskNotify.* [Consulta: 29 de Novembre 2021]. URL: https://www.freertos.org/xTaskNotify.html.

Índex de taules

Índex de figures