TREBALL DE RECERCA:
INTEL·LIGÈNCIA ARTIFICIAL

Toni López
2n Batxillerat A
Tecnologia
Tutor: Jordi Orts
INS Príncep de Viana
Gener 2020

Resumen

Mi TR trata sobre *Inteligencia Artificial*, un concepto que cada vez está cobrando más importancia en nuestra sociedad y que está llamado a revolucionar nuestro futuro. Este hecho, junto a mi gran interés en el mundo de la tecnologia y todo lo relacionado con ella, me hizo decidirme a realizar este trabajo, en el qual he investigado y he aprendido acerca de conceptos como el *Machine Learning*, los paradigmas de aprendizaje, las redes neuronales y, como no podía ser de otra forma, acerca de la inteligencia artificial en sí misma.

Toda esta parte teórica del trabajo viene acompañada por su correspondiente parte práctica, en la cual he aplicado los conocimientos adquiridos con anterioridad con el objetivo de crear un proyecto que tuviera una aplicación en el mundo real o que, con una serie de modificaciones, pudiera llegar a tenerla. Y después de mucho trabajo y de muchas pruebas fallidas, creo que puedo afirmar que, si bien con ciertos errores, he conseguido desarrollar un proyecto que cumple esos requerimientos.

Se trata de un vehículo de dos ruedas capaz de evitar los obstáculos en su camino gracias a los sensores que lleva incorporados y que, además, es capaz de reconocer algunas señales de tráfico (reinventadas como caras humanas debido a ciertas limitaciones) y actuar en consecuencia.

Sin embargo, como hemos comentado, tiene algunos errores: el programa solo actúa correctamente cuando tiene que reconocer una única “señal”, ya que al trabajar con más se genera un error que le impide funcionar bien. A parte de ese fallo, hay muchas mejoras que, con tiempo, podríamos aplicar, siendo éstas, no obstante, de mucha menor importancia para el proyecto.

Abstract

My TR's topic is *Artificial Intelligence*, a concept which is increasing every day its importance in our society and which is said to cause big changes in our future. This fact, alongside my great interest in technology and everything related to it, made me decide to choose this specific subject, in which I have researched and learned about *Machine Learning*, neural networks and, of course, about artificial intelligence itself.

This whole theoretical part is accompanied by its corresponding practical part, where I have put everything I had learned together in order to try to develop a project that had an actual real-life use or, at least, that it could have it with a few modifications. And, after a lot of hard work and multiple failures, I think I can state that, even with a few errors, I have been successful in making a project that fulfills those requirements.

It is a two-wheel vehicle capable of avoiding obstacles that are put in its way, thanks to some incorporated sensors. It can also recognize different faces as they were traffic signals (it would had been really tough to make it recognize actual signals due to our limitations) and act in consequence.

However, as we have said before, it has some mistakes: the program works correctly only when it has to recognize a single "signal". When there are more, some problems appear and the vehicle can not work properly. Apart from this main mistake, there are plenty of possible ways to improve, with time, our project, although this upgrades are a lot less important.

Índex

Capítol 1. Introducció

1.1. Antecedents

Al llarg de tota la història, la humanitat ha anat evolucionant juntament amb la tecnologia, utilitzada per a solucionar tot tipus de problemes i fer la nostra vida més còmoda. No tots els elements tecnològics, però, han tingut la mateixa importància i repercussió: hi ha invents que han marcat un abans i un després en la forma d'entendre el món.

Actualment, s'està començant a desenvolupar una d'aquestes tecnologies revolucionàries que, en uns anys, de ben segur canviarà completament la nostra societat. Es tracta de la **intel·ligència artificial**.

Des de ben petit m'ha agradat la tecnologia i estar sempre enterat dels darrers avenços tecnològics, imaginant invents que canviessin la nostra vida. Per aquest motiu, al arribar a batxillerat vaig decidir triar aquesta modalitat. Va passar el mateix a l'hora d'escollir l'àmbit del treball de recerca, vaig triar la tecnologia per poder dur a terme algun projecte interessant i pràctic.

Quan vaig haver de decidir un tema vaig tenir força clar que volia que tingués alguna relació amb la intel·ligència artificial, ja que sempre m'ha semblat un camp molt interessant (influenciat, entre d'altres coses, per pel·lícules amb elements de ciència ficció) i, com dèiem abans, amb molt de futur però del que no en sabem gairebé res, degut a la seva complexitat.

Concretar un tema va ser més complicat, ja que havíem d'ajustar-nos als recursos disponibles i el camp en el que havia decidit treballar sol requerir de, per exemple, una gran capacitat computacional.

Al final, després de múltiples converses amb el meu tutor, vam decidir enforcar-nos més en aplicacions d'intel·ligència artificial més relacionades amb la robòtica i l'electrònica que podrien utilitzar-se en camps com ara la conducció autònoma.

1.2. Objectius

Els principals objectius a l'hora de dur a terme aquest projecte són:

- Investigar i aprendre sobre l'extens camp de la intel·ligència artificial aplicada a diferents aspectes de la tecnologia i de la societat moderna.
- Millorar els meus coneixements sobre altres aspectes relacionats amb el treball (programació, electrònica, impressió 3D...).
- Realitzar un projecte final aplicant els coneixements apresos que sigui útil o pugui arribar a ser-ho.
- Elaborar una memòria clara, entenedora i coherent.

Capítol 2. Fonament teòric

2.1. Intel·ligència Artificial

Per començar a parlar d'intel·ligència artificial hem de definir primer què és. Fer aquesta definició, però, no és gens fàcil ja que depèn directament del que entenguem per intel·ligència com a tal, concepte que, a dia d'avui, té múltiples significats. Per aquest motiu, diferents autors fan interpretacions ben diverses, però si n'extraiem les idees comunes, trobem que la intel·ligència artificial (IA) és ***la subdisciplina del camp de la informàtica que busca la creació de màquines que puguin imitar comportaments intel·ligents***.

Aquesta tecnologia, tot i ser força novedosa, ja ha demostrat grans resultats, arribant a superar el rendiment humà en moltes situacions. Potser la més coneguda és la victòria de la intel·ligència artificial ***Deep Blue*** en una partida d'escacs contra el rus Garri Kaspàrov, campió del món en aquesta modalitat, l'any 1997 [1].

Figura 2.1. Deep Blue vs Kaspàrov

La màquina no només havia derrotat un humà, sinó que a més ho havia fet en una disciplina que havia estat considerada sempre impossible de dominar per als ordinadors, degut a la seva complexitat i a la gran quantitat d'entrenament i aprenentatge necessàries per, simplement jugar-hi. Però, vol dir això que els algoritmes superen la capacitat cerebral humana?

La resposta és que no. Si agafem qualsevol IA que destaca en un camp determinat i intentem que realitzi una tasca per a la que no ha estat programada, veurem que el resultat que obtenim és desastrós. La capacitat de realitzar múltiples tasques és la que ens permet fer tot el que fem: caminar, parlar, jugar, llegir… I això és el que s'està buscant des de fa molt de temps en els laboratoris dedicats a investigar la intel·ligència artificial. Amb això podem fer una primera classificació d'algoritmes de IA: dèbils i forts.

La intel·ligència artificial dèbil és aquella que només pot realitzar correctament una tasca o un grup molt reduït d'elles, sent força inútil a la resta.
D'altra banda, una intel·ligència artificial forta és aquella que pot generalitzar el coneixement après per aplicar-ho a altres àrees per a les quals no ha estat programada, tal i com fem els humans.
Avui en dia, però, tots els algoritmes es troben en el primer grup.

Figura 2.2. Robot “dèbil” intentant xutar una pilota.

Tornant a l’inici, els sistemes d’intel·ligència artificial tenen moltíssims usos, i cada cop es milloren més per aplicar a diferents situacions. Per això, existeixen diferents subcategories per classificar els algoritmes segons el seu comportament. Per exemple, si parlem de la capacitat de moure’s i adaptar-se a l’entorn, en referim a la **robòtica** que, tot i no semblar un comportament intel·ligent, si que ho és. Si programem un braç robòtic perquè faci un determinat moviment, tot i haver-ho programat nosaltres el comportament del braç imita una capacitat humana, per tant entra en la definició que hem donat anteriorment.

Figura 2.3. Robot “gos” de l’empresa Boston Dynamics[2]

Una altra habilitat molt interessant és el ***Natural Language Processing*** o ***NLP*** (en anglès, processament de llenguatge natural) que s’encarrega d’interpretar i entendre el llenguatge

humà. Recentment s'han fet grans avenços en aquest camp arribant a crear un programa que, a partir d'un text inicial pot continuar escrivint amb força sentit, l'algoritme **GPT-2**. [3]
Altres capacitats són la **conversió de text a veu** (com al traductor de Google) i la **conversió de veu a text** (com en gairebé tots els telèfons actuals amb Siri, Ok Google, etc.), entre moltes altres. Però si una subdisciplina del camp de la intel·ligència artificial destaca per sobre de les altres és la capacitat d'aprendre, el **Machine Learning**.

2.2. Machine Learning

Per molt que hi hagi diferents habilitats que puguin desenvolupar els agents intel·ligents per ser reconeguts com a tal, la més important tant en les màquines com en els humans és la capacitat d'aprendre, el **Machine Learning (ML)**. El Machine Learning el podem definir com la **branca dins del camp de la intel·ligència artificial que busca la forma de dotar les màquines de capacitat d'aprenentatge**, entès aquest com la generalització del coneixement en base a una sèrie d'experiències.

El Machine Learning, doncs, és una subdisciplina dins del camp de la intel·ligència artificial. Però no n'és una de qualsevol, sinó que és un component fonamental dins d'aquest camp que es relaciona amb tota la resta de categories. Això té una explicació molt clara: podem programar un algoritme perquè faci una determinada cosa (com vèiem en l'apartat anterior) però també podem programar l'algoritme perquè aprengui a realitzar aquesta mateixa tasca. Això és el que fa que el ML tingui una importància tan gran actualment i que s'estigui treballant molt sobre aquest concepte.

Per fer aprendre a aquests algoritmes, els investigadors van fixar-se en la natura, principalment en el cervell humà per observar com aprenem nosaltres. Així es van idear els **paradigmes d'aprenentatge**, models segons els quals les nostres màquines poden aprendre. Els més coneguts i utilitzats a dia d'avui són 3: l'**aprenentatge supervisat**, l'**aprenentatge no supervisat** i l'**aprenentatge reforçat**.

El primer paradigma, l'**aprenentatge supervisat** és segurament el més senzill d'entendre. Consisteix en proporcionar a un algoritme suficients dades d'entrada i de sortida per tal que pugui trobar la relació que hi ha entre elles i aplicar-la a casos que no hagi vist amb anterioritat. L'adjectiu "supervisat" ve de la forma en que "ajudem" a l'algoritme a aprendre a partir de dir-li quins són els resultats que volem obtenir, supervisant així el seu entrenament.

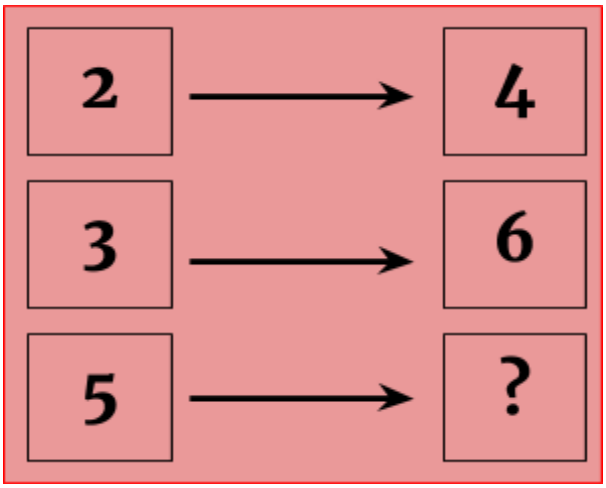


FIgura 2.4. Exemplificació de l'aprenentatge supervisat

Aquests algoritmes, tot i semblar força senzills, tenen una gran quantitat d'usos i han provat ser molt efectius en certes situacions. Per exemple, un cas d'aprenentatge supervisat que utilitzem a diari sense adonar-nos és el filtre de spam del nostre correu electrònic. Aquest ha estat entrenat mitjançant correus etiquetats com a spam i d'altres que no ho eren i ha estat capaç d'aprendre les semblances i diferències entre ells. Per això, quan li arriba un correu nou, que no ha vist durant l'entrenament, pot saber amb un grau d'encert molt elevat si es tracta o no de spam.

El problema és que, perquè l'entrenament sigui efectiu, necessites una gran quantitat de dades que, depenent del cas, poden arribar a ser milers o centenars de milers de parells de dades (entrada i sortida). Un exemple força clar és un algoritme que ha de diferenciar entre imatges de gossos i de gats. Primer s'ha d'obtenir moltes imatges d'ambós animals i posteriorment etiquetar-les segons l'animal del que es tracti perquè l'algoritme pugui buscar-ne relacions i diferenciar-les. Aquest procés titànic d'etiquetatge acostuma a ser realitzat per persones que han d'estar hores treballant-hi.
Per aquest motiu, existeix un altre gran paradigma d'aprenentatge cada cop més rellevant: l'**aprenentatge no supervisat**.

L'aprenentatge no supervisat es basa en entrenar un algoritme perquè aprengui a dur a terme una certa tasca proporcionant-li únicament valors d'entrada. És a dir, l'algoritme aprendrà buscant les relacions entre unes dades d'entrada, sense cap coneixement previ. L'aplicació més clara d'aquest paradigma la trobem en els algoritmes de classificació o

clustering (en anglès), encarregats de separar les dades que introduïm al programa en diferents grups segons les seves característiques.

Podem veure un clar exemple en la imatge següent:

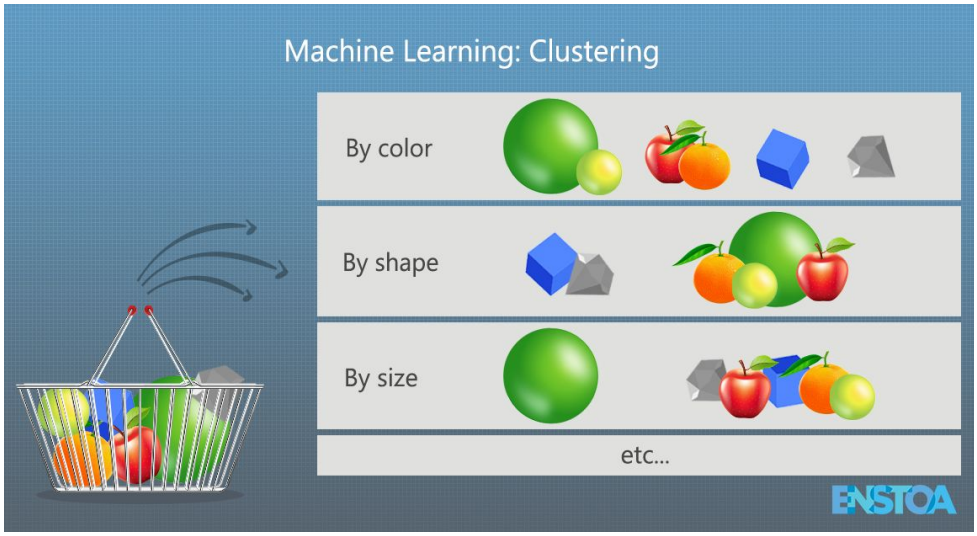


Figura 2.5. Exemple de clustering

En aquest exemple podem veure com un algoritme podria separar els diferents elements que hi ha en la cistella de supermercat segons el color, la forma o la mida, entre d'altres.

Aquest tipus d'aprenentatge és cada cop més important, arribant a tenir aplicacions realment sorprenents. Si imaginem una cadira o una taula, per exemple, ens poden venir al cap molts tipus de mobles diferents, de diferents materials, tamanys, formes, etc. Tot i això, nosaltres automàticament podem reconèixer les imatges i dir si l'objecte és o no és una taula, una cadira o una prestatgeria sense saber exactament perquè. Doncs, alguns algoritmes, només buscant les relacions entre els valors d'entrada, podrà aprendre a distingir i classificar les imatges, tal i com fem nosaltres.

Pel seu gran potencial i el fet que realitzar un entrenament en aquest paradigma és molt menys costós, molts experts afirmen que el futur del camp passa pel desenvolupament de l'aprenentatge no supervisat.

El tercer paradigma d'aprenentatge és certament molt diferent als altres dos. L'**aprenentatge reforçat** consisteix en l'aprenentatge d'una tasca per part d'un algoritme mitjançant un sistema de recompenses. Aquesta forma d'aprendre és molt semblant a la que dels humans (especialment quan som petits), tot i que també la trobem en altres

animals com els gossos. Quan volem ensenyar al nostre gos a fer algun “truc”, segons els seu progrés l’anirem recompensant i així l’animal sabrà que allò que està fent és el correcte.

Al pensar en aquestes característiques ràpidament podem relacionar tot això amb els videojocs, simulacions en les quals hem de fer un seguit d’accions per obtenir certes recompenses. De fet els videojocs s’utilitzen per entrenar aquests models d’aprenentatge. Però els videojocs només són mitjans per ensenyar els algoritmes i després poder utilitzar-los en situacions del món real: un programa podrà aprendre a superar els diferents nivells d’un videojoc (o un altre tipus de simulació) amb molt poc coneixement previ i després, un cop entrenat, podrà ser utilitzat per programar, per exemple, un robot en el món real.

Per exemplificar l’aprenentatge reforçat i la implicació dels videojocs en aquest, utilitzarem un joc molt conegut, el *Super Mario Bros*:

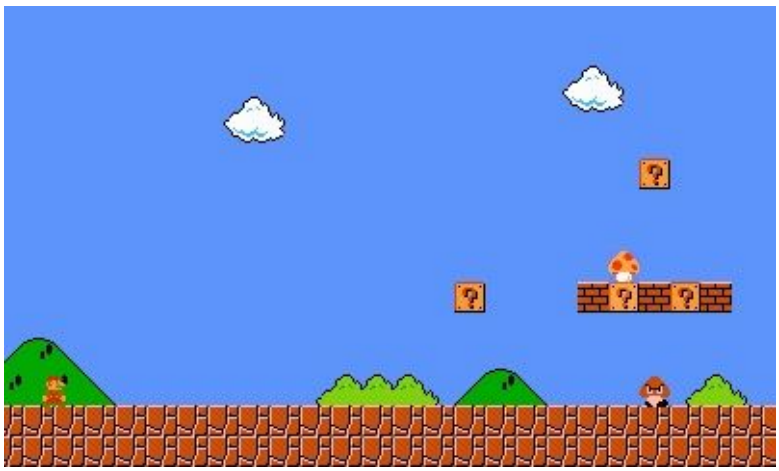


Figura 2.6. Nivell de *Super Mario Bros*.

Per entrenar un algoritme li podem “dir” que ha de completar el nivell, el joc, etc. A més, podem donar-li més o menys informació sobre com es pot moure, la seva posició, etc. (com més informació tingui l’agent intel·ligent més fàcilment resoldrà la tasca que li ha estat encomanada). Un cop fet això, l’agent intentarà completar la tasca a base de prova i error. Nosaltres com a éssers humans segurament haurem tingut algun tipus d’experiència amb algun tipus de videojoc que ens permeti deduir cap a on ens hem de moure o què hem de fer amb els enemics, però l’algoritme no, per tant, s’anirà movent a l’atzar fins que rebi algun senyal de recompensa o una penalització. Per exemple, quan l’agent es trobi amb l’enemic (el *goomba*), aquest el “matarà”, penalitzant-lo. En canvi, quan, per casualitat, es trobi amb el bolet, augmentarà la seva puntuació (recompensa). De totes aquestes experiències anirà

aprenent fins que sigui capaç de resoldre tot el joc amb el mateix rendiment que una persona i, posteriorment, amb un rendiment molt superior.

Una de les empreses més conegudes per treballar en aquest camp és *Deep Mind* (actualment en propietat de *Google*). Aquesta empresa va crear un algoritme capaç de superar el rendiment humà en una sèrie de jocs de la videoconsola *Atari 2600* com podem veure a la gràfica següent:

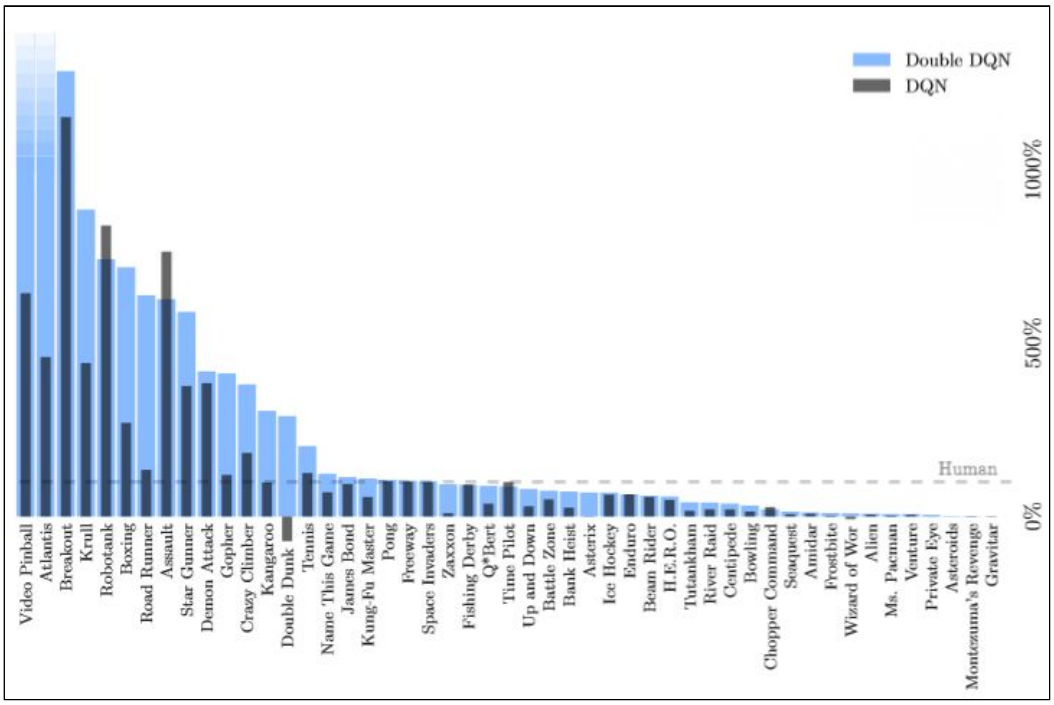


Figura 2.7. Rendiment d'una IA en 49 jocs de l'*Atari 2600*.[4]

2.3. Xarxes Neuronals

Hi ha diferents formes de crear un algoritme d'intel·ligència artificial que faci ús del Machine Learning per aprendre a dur a terme una tasca. La més coneguda, i que ha donat gran fama al camp de la intel·ligència artificial, són les **xarxes neuronals**.

Les xarxes neuronals són una família d'algoritmes inspirats en el funcionament del cervell humà. Consisteixen en un conjunt d'unitats (neurones artificials o nodes) connectades entre si per transmetre un seguit de senyals. Les dades d'entrada travessen la xarxa, on són sotmeses a operacions matemàtiques, abans d'arribar a la sortida.

Entenem el terme “neurona” com la unitat bàsica de processament dintre d’una xarxa neuronal. Com hem dit anteriorment, les neurones i les xarxes neuronals prenen el nom de la seva contrapart biològica degut al seu similar funcionament. Una neurona té connexions d’entrada a través de les quals reben estímuls externs, els valors d’entrada. Aleshores, tal i com qualsevol funció matemàtica, transforma aquests valors per acabar generant uns valors de sortida.

Els càlculs que realitza la neurona són força senzills, ja que es tracta, bàsicament, d’una suma ponderada.

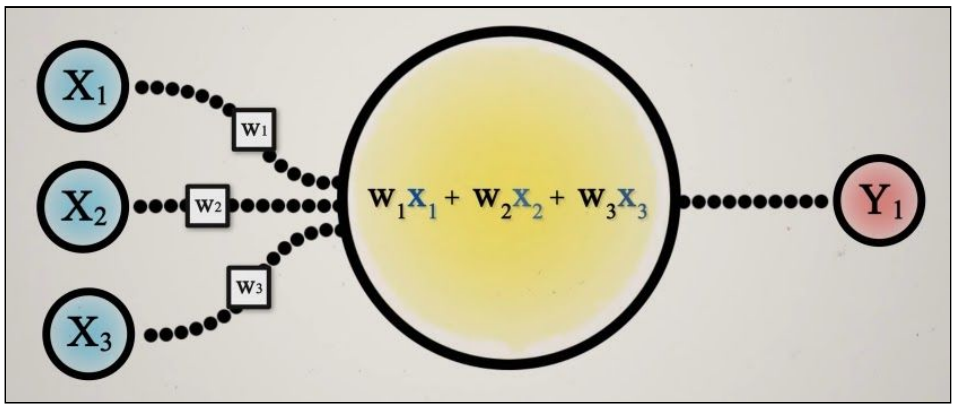


Figura 2.8. Funcionament d’una neurona artificial

Els valors d’entrada (X_1 ,X_2 ,X_3) es multipliquen pels pesos de cada branca (w_1, w_2, w_3, respectivament), que serveixen per determinar amb quina intensitat afecta cada variable d’entrada al resultat final, al valor de sortida (Y_1). A més, existeix un valor anomenat *bias* (biaix en català), que actua com a terme independent per dotar la neurona d’una major llibertat algebraica. Finalment, trobem un altre component, la funció d’activació, una funció per la qual passem el resultat obtingut a la suma ponderada.

Una neurona en si mateixa ja pot ser considerada com a xarxa, ja que ens permet codificar certa informació. El problema, però, és que aquesta informació és massa simple i per això es combinen les neurones formant xarxes.

Aquesta combinació permet aprendre coneixement cada vegada més complex de forma jerarquitzada com més capes afegim. És el que es coneix com ***Deep Learning*** (aprenentatge profund, en anglès). En les primeres capes es podrien aprendre conceptes com què és un retrovisor, una roda o un seient, per acabar distingint entre cotxe, camió, moto, etc.

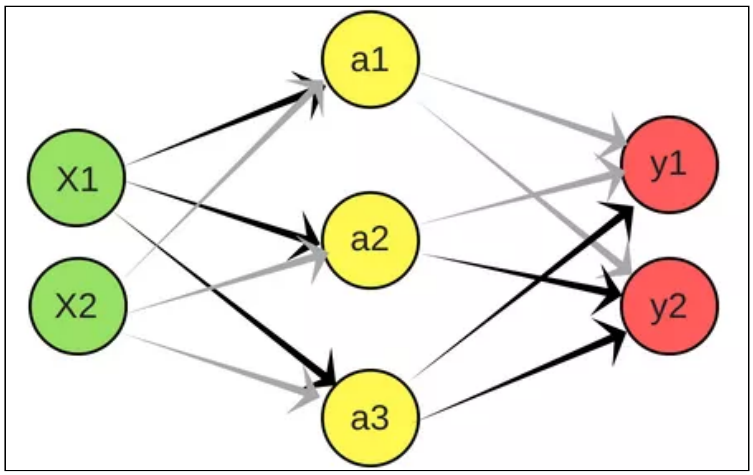


Figura 2.9. Exemple de xarxa neuronal simple

Un cop tenim la xarxa, però, com aconseguim que aprengui de forma automàtica (entenent aprendre com ajustar els pesos entre les diferents neurones per tal d'aconseguir el resultat esperat)?
Aquesta va ser la pregunta que es van estar fent els investigadors durant molt de temps i que va portar les xarxes neuronals a convertir-se en el que són avui.

Al començament, aquest procés es realitzava per força bruta: a base de moltes iteracions s'anava veient quina importància tenia cada neurona en el resultat final i, per tant, en l'error produit. D'aquesta forma s'anava ajustant la xarxa fins aconseguir el resultat obtingut, així com la minimització de l'error. El problema d'aquesta manera de treballar era que es requeria una gran capacitat computacional per poder donar lloc a les molt nombroses iteracions que permetien ajustar la xarxa. A més, la tendencia era augmentar cada cop el nombre de capes utilitzat i, com a conseqüència, el nombre de connexions entre nodes que ajustar. La impossibilitat de millorar l'eficiència d'aquests sistemes va provocar que es deixés d'investigar durant molt de temps, fins que va arribar un algoritme que va revolucionar completament el camp de la intel·ligència artificial: l'algoritme de ***Backpropagation***.[5]

Backpropagation (propagació cap enrere, en anglès) és un algoritme que, com diu el seu nom, propaga l'error de la xarxa cap enrere. L'error obtingut en la sortida és dividit en les neurones de la capa anterior segons el valor del pes assignat i això es realitza successivament en totes les capes. Aquest algoritme permet ajustar la xarxa per minimitzar l'error de forma força simple i requerint de molt menys temps i recursos que els mètodes anteriorment utilitzats.

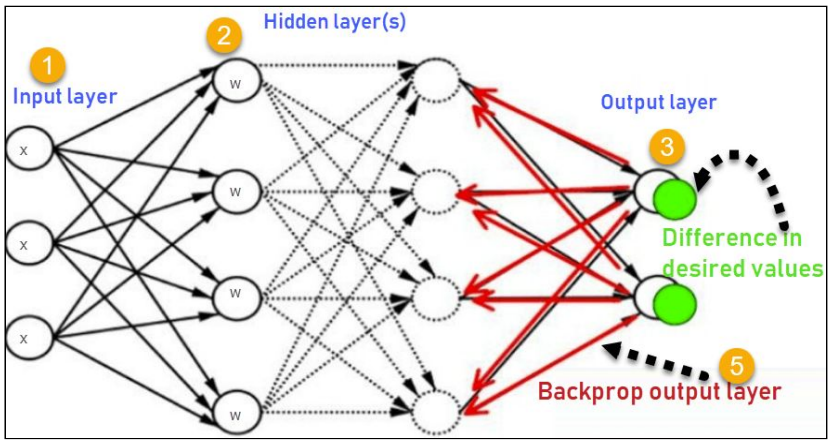


Figura 2.10. Funcionament de *Backpropagation*.

Capítol 3. Desenvolupament

3.1. Arduino IDE

Per dur a terme aquest projecte, utilitzem l'Arduino IDE, l'entorn de programació d'Arduino, que ens permet programar des del nostre ordinador a través d'un cable microUSB una sèrie de plaques i mòduls de forma senzilla i clara, en C++, un dels llenguatges de programació més flexibles, ja que es pot adaptar a tot tipus de situacions.

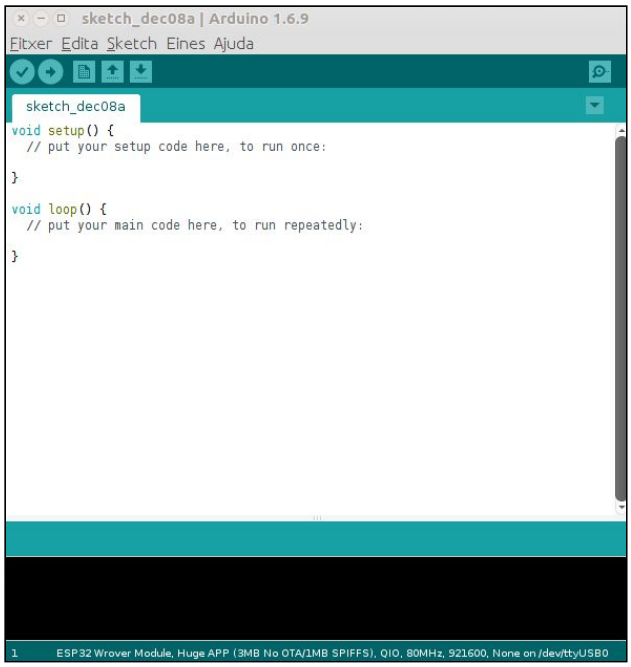


Figura 3.1. IDE Arduino

En el desenvolupament del projecte, però, no només hem utilitzat el *software* bàsic d'Arduino, sinó que, a més, hem instal·lat dos "pedaços" que ens han permès treballar amb una sèrie d'eines que, d'altra forma, no tindríem. Aquests pedaços són l'ESP8266 i l'ESP32. [Annex A]

3.2. Components electrònics

3.2.1. Node MCU

És una de les plaques principals del projecte, ja que és molt útil a l'hora de treballar amb motors. Està basada en l'ESP8266. La versió concreta que utilitzem és la NodeMCU 0.9.

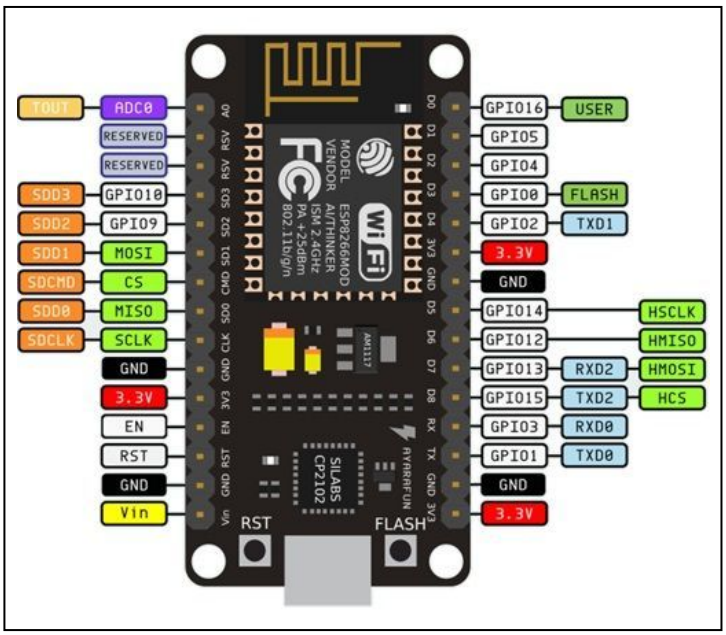


Figura 3.2. NodeMCU pinout

3.2.2. Motor Shield i Motors DC

Per tal de connectar la placa (NodeMCU) amb els motors, utilitzem un motor shield, dissenyat i optimitzat per aconseguir una comunicació fàcil entre els dos sistemes i poder moure els motors de forma independent.

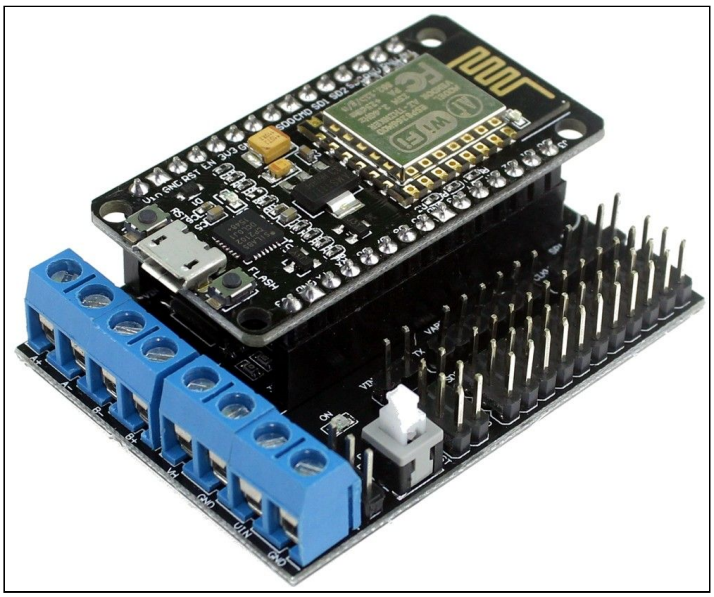


Figura 3.3. NodeMCU motor shield

Contarem, a més amb 2 motors DC (*Direct Current*, en anglès Corrent Continu) funcionant a 5V, connectats a dues potes de la placa NodeMCU cadascun i a GND. Les potes són D1, D2, D3 i D4, que equivalen a les potes GPIO5, GPIO4, GPIO0 i GPIO2 d'Arduino, respectivament.

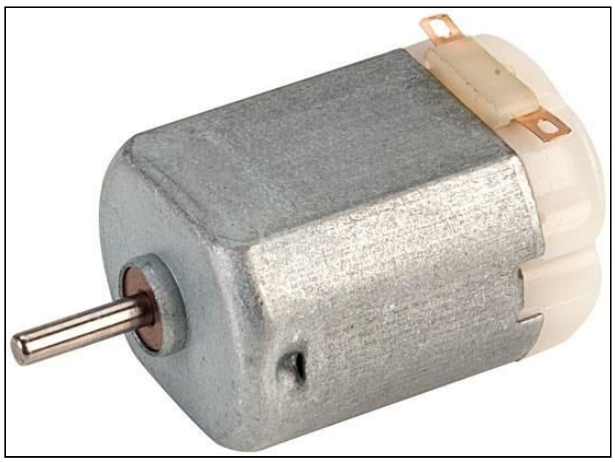


Figura 3.4. Arduino motor dc

Podem comprovar el funcionament dels motors amb l'exemple següent:

```
//Exemple motor

 //Declarem les potes del motor
int motor1 = 5;
int motor2 = 0;

void setup() {
```

```
 //Declarem les dues potes com a sortida

    pinMode(motor1, OUTPUT);
    pinMode(motor2, OUTPUT);

}

void loop () {

 //Movem el motor endavant

    analogWrite(motor1, 1023);    //Indiquem la velocitat del motor
    digitalWrite(motor2, 1);    //Indiquem la direcció del motor

    delay(100);    //Delay en ms

 //Movem endarrere

    analogWrite(motor1, 1023);
    digitalWrite(motor2, 0);

    delay(100);
}
```

3.2.3. Servo

Un altre component que utilitzarem és un servo, en el nostre cas el model *Modelcraft RS-2*.

Figura 3.5. Servo *Modelcraft RS-2*.

Connectarem el servo a la pota D8 de la placa NodeMCU, que equival a la pota GPIO15 d'Arduino.

Utilitzem aquest codi per comprovar-ne el funcionament:

```
//Exemple servo
```

```
//Incloem la llibreria Servo.h que ens permetrà treballar més fàcilment amb el servo [Annex B]

#include <Servo.h>

Servo myservo;  // Creem un objecte servo per crontrolar el servo

void setup() {

 myservo.attach(15);  // Fixem l'objecte servo a la pota D8

}

void loop () {

 int pos;

  for (pos = 0; pos <= 180; pos += 1) {      // Movem el servo des de 0º a 180º en passes d'1º

    myservo.write(pos);              // Dir al servo d'anar on indica la variable "pos"
    delay(15);                       // Espera 15 ms a que el servo arribi a la posició

  }

// Fem el mateix però en sentit contrari
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(15);
  }
}
```

3.2.4. Sensor Ultrasons

També contarem amb un sensor d'ultrasons, concretament un *HC-SR04*.

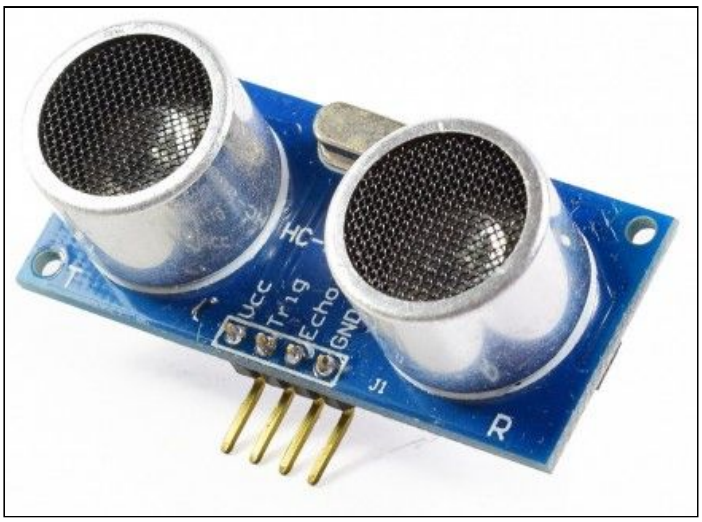


Figura 3.6. Sensor d'ultrasons *HC-SR04*.

Connectarem l'*echoPin* (el que rep el so) a D5 i el *trigPin* (emet el so) a D6. Equivalen a GPIO12 i GPIO14, respectivament.

Un cop més, comprovem el seu correcte funcionament amb el codi de prova següent:

```
//Exemple sensor ultrasons

 //Declarem les potes del sensor
#define trigPin D6
#define echoPin D5

void setup() {

 //Declarem les dues potes i iniciem el port sèrie
    Serial.begin (115200);
      pinMode(trigPin, OUTPUT);
      pinMode(echoPin, INPUT);

}
void loop () {

 long duracion, distancia ;
          digitalWrite(trigPin, LOW);       // Ens assegurem que el trigger està LOW
          delayMicroseconds(2);

          digitalWrite(trigPin, HIGH);      // Activem el pols de sortida
          delayMicroseconds(10);             // Esperem 10µs. El pols segueix actiu aquest temps
          digitalWrite(trigPin, LOW);       // Tallem el pols i esperem l'echo

          duracion = pulseIn(echoPin, HIGH) ;    // Mesurem el temps que triga el so en arribar al
sensor
          distancia = duracion / 2 / 29.1  ;   // Convertim el temps en cm*

          Serial.println(String(distancia) + " cm.") ; Escrivim el resultat al port sèrie
          delay(100);

}
```

*El temps es divideix entre dos ja que el so ha de fer el camí d'anada i tornada del sensor i, també és divideix per 29,1 (µs/cm), relació obtinguda pel canvi d'unitats de la velocitat de so al buit.

Un cop enviat el programa, comprovem que els valors dels resultats s'envien al port sèrie i que aquests valors són coherents.

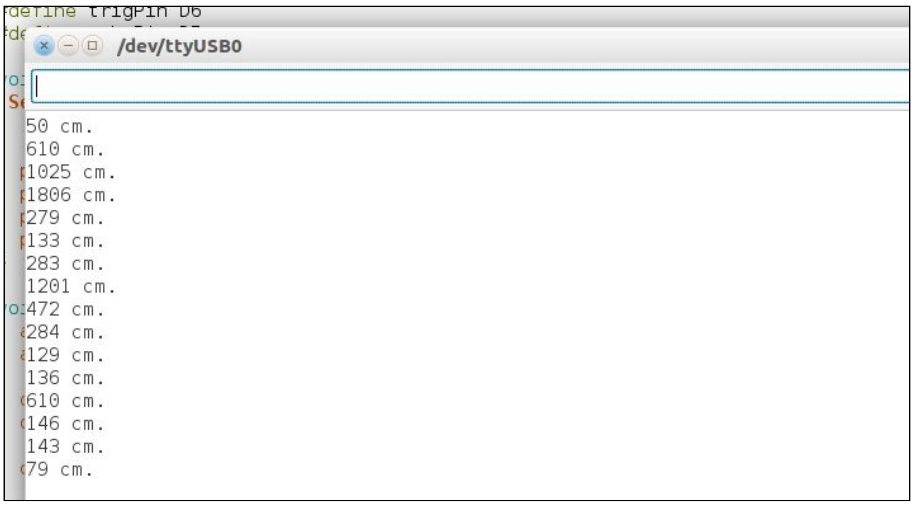


Figura 3.7. Imatge del port sèrie rebent dades del sensor d'ultrasons.

3.2.5. Esp32cam

A més de tots els component anteriorment anomenats, n'utilitzarem un de més, l'Esp32cam, una placa a la que incorporem una petita càmera. Per treballar amb ella utilitzarem el software ESP32.

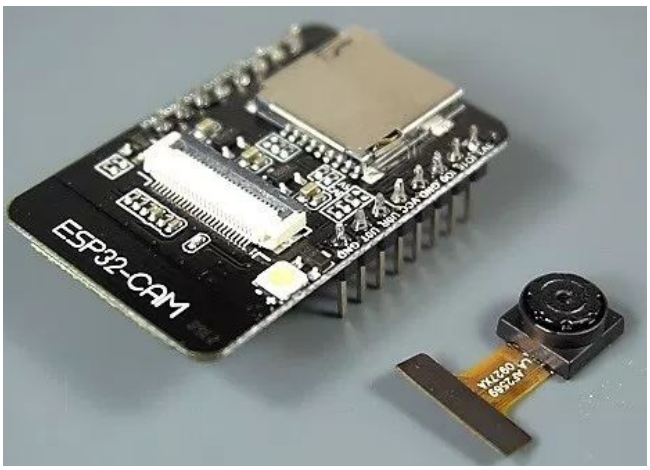


Figura 3.8. Mòdul ESP32-cam

A més, ja que aquesta placa no ve amb connector USB, l'haurem de programar mitjançant un programador FTDI. Per tant, per tal d'enviar un codi, primer haurem de connectar placa i programador seguint el següent esquema:

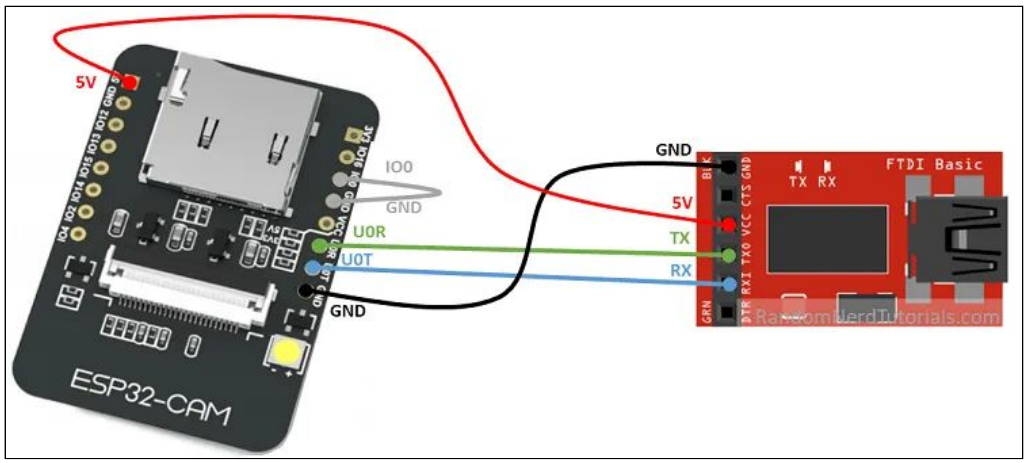


Figura 3.9. Connexió ESP32-Programador FTDI

Finalment, un cop enviat el codi, haurem de treure el cable IO0-GND i prémer el botó de Reset pel correcte funcionament del programa.

Veurem un exemple de funcionament més endavant.

3.2.6. Base mòbil

Com a base de tots els components, utilitzarem un vehicle de plàstic negre amb els dos motors connectats a una roda cadascun, alimentats per 4 piles 1,5V. Els altres components, però, seran alimentats per una bateria portàtil de 5V.

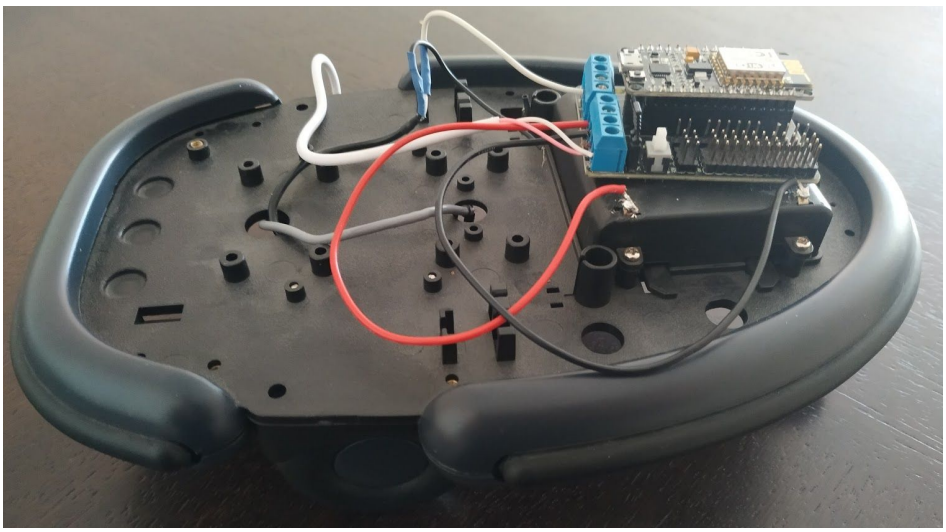


Figura 3.10. Base mòbil

3.3. Desenvolupament del projecte

Per començar a treballar en el projecte primer havíem de comprovar el correcte funcionament de tots els components electrònics, com hem explicat en l'apartat anterior. Un cop fet això, vam començar a posar en pràctica els coneixements adquirits, programant una xarxa neuronal. Tot i que no sabíem segur quin seria el projecte final, si que volíem que tingués a veure amb l'automoció i els cotxes autònoms.

Aleshores vam començar a desenvolupar (inspirant-nos en una publicació d'Internet) una xarxa neuronal capaç de conduir un petit robot d'Arduino evitant els obstacles que trobi pel camí, utilitzant els components que hem explicat anteriorment. Programarem la xarxa en Python. [Annex D]

Les dades d'entrada són:

- **Distància a l'obstacle:**
    - si és 0 no hi ha obstacles a la vista
    - si és 0,5 s'apropa a un obstacle
    - si és 1 es troba massa a prop d'un obstacle

- **Posició de l'obstacle:**
    - L'obstacle vist a l'esquerra serà -1
    - Vist a la dreta serà 1

Les sortides seran:

- **Girar:**
    - Dreta 1
    - Esquerra -1

- **Direcció:**
    - Avançar 1
    - Retrocedir -1

La velocitat del vehicle podria ser una sortida més que permetria anar desaccelerant com més t'apropessis a l'obstacle, però per simplicitat no la tindrem en compte (també es podrien afegir altres sensors per controlar altres variables).

Per tant, per entrenar la xarxa tindrem aquestes combinacions d'entrades i sortides:

Entrada: Sensor Distància	Entrada: Posició Obstacle	Sortida: Gir	Sortida: Direcció	Acció de la Sortida
0	0	0	1	Avançar
0	1	0	1	Avançar
0	-1	0	1	Avançar
0.5	1	-1	1	Gir a l'esquerra
0.5	-1	1	1	Gir a la dreta
0.5	0	0	1	Avançar
1	1	0	-1	Retrocedir
1	-1	0	-1	Retrocedir
1	0	0	-1	Retrocedir

Al iniciar la xarxa, els pesos entre les neurones seran completament a l'atzar, però conforme l'entrenament vagi avançant, mitjançant l'algoritme de backpropagation aquests pesos s'aniran actualitzant perquè l'error es vagi reduint i les sortides coincideixin amb les que hem indicat (estem tractant amb un algoritme supervisat). Al final, un cop executada la xarxa el resultat obtingut és aquest:

```
X: [0. 0.] y: [0 1] Network: [-6.80860029e-04  9.99956133e-01]
X: [0. 1.] y: [0 1] Network: [-4.90723326e-04  9.99942810e-01]
X: [ 0. -1.] y: [0 1] Network: [0.002269   0.99981716]
X: [0.5 1. ] y: [-1  1] Network: [-0.95261482  0.95387801]
X: [ 0.5 -1. ] y: [1 1] Network: [0.94415312 0.96644068]
X: [1. 1.] y: [ 0 -1] Network: [-0.0023972  -0.97191192]
X: [ 1. -1.] y: [ 0 -1] Network: [ 0.00781254 -0.98043774]
```

Figura 3.11. Output de la xarxa neuronal de prova

Com es pot veure, els resultats són molt bons, s'ajusten molt als valors que hem indicat.

Es pot veure el codi complet a l'annex [Annex E].

A continuació, hem d'aplicar la xarxa al robot d'arduino. Per fer-ho, utilitzarem un codi d'arduino on programarem la xarxa i, a més, haurem d'afegir els pesos de les connexions entre neurones de la xarxa que hem entrenat prèviament per no haver-la d'entrenar de nou cada cop que posem el robot en marxa. [Annex E]

Un cop fet això, però, vam voler continuar treballant per dur a terme un projecte més ambiciós. Després de barallar múltiples possibilitats, vam decidir centrar els nostres esforços en investigar i posar en pràctica un sistema relacionat amb la visió computacional, un sistema que pogués, per exemple, reconèixer semàfors i el seu estat per tal de controlar el cotxe segons quina senyal estigui "veient".

Per fer això vam decidir utilitzar la placa ESP32, com hem comentat anteriorment. Al tractar-se d'un mòdul amb càmera incorporada, petit, de molt fàcil ús i econòmic, va ser una bona elecció per la tasca a realitzar. A més, contava amb una sèrie d'exemples especialment dedicats a treballar amb intel·ligència artificial.

L'exemple que ens interessa en aquest cas s'anomena "Camera Web Server". Es tracta d'un programa que s'encarrega d'enviar una senyal de vídeo en directe mitjançant wifi a qualsevol aparell proper que es connecti a una determinada adreça. Un cop allà, podem seleccionar una opció per tal de reconèixer cares de les persones, les quals seran emmarcades en un quadrat. Una altra opció que ens ofereix el programa (la que més ens interessa) és poder reconèixer la cara d'un subjecte concret (després de prendre una sèrie de mostres) que passarà a ser anomenat com a *Subject X* (sent X un número a partir de 0 segons la quantitat de persones enregistrades). Qualsevol cara no memoritzada serà reconeguda com a *Intruder* (intrús). Això es fa evident en els marcs que envolten les cares: els *Subjects* el tindran de color verd mentre que els *Intruders* de color vermell.

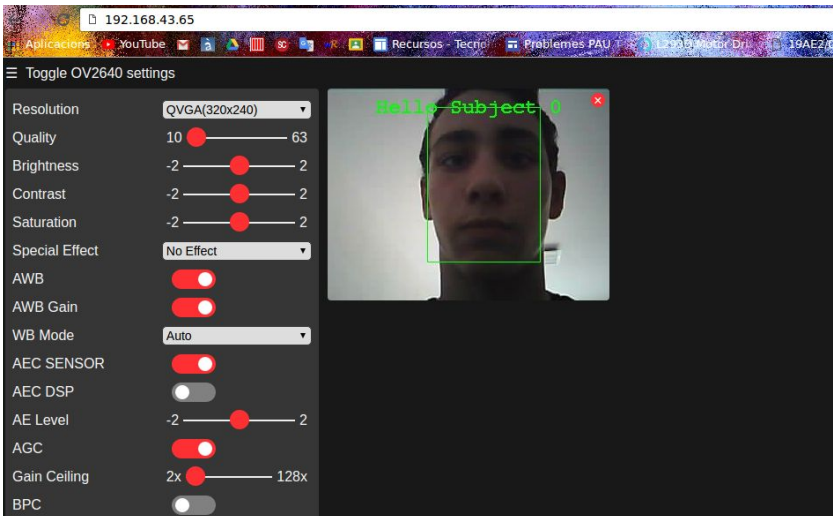


Figura 3.13. Reconeixement facial amb ESP32

A l'hora de programar aquest codi hem de tenir en compte vàries coses. La primera és enrecordar-nos d'escriure la nostra xarxa wifi, a la que l'ESP32 es connectarà i la seva contrasenya.

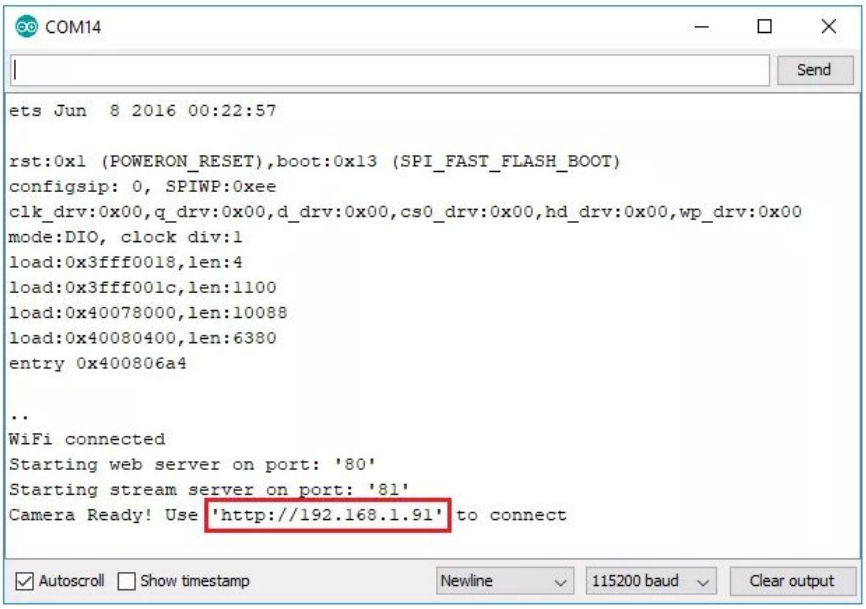
```
const char* ssid = "webcamtoni";
const char* password = "robotica";
```

També hem de seleccionar el model correcte d'entre tots els que hi ha, ja que seleccionar un o un altre afectarà després en el comportament del programa.

```
// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER
```

Finalment haurem d'escollir la placa amb la que treballarem i ajustar les opcions de compilació des del menú *Eines* de l'Arduino IDE.
La placa que haurem de seleccionar és: ESP32 Wrover Module
També hem d'ajustar el *Partition Scheme*: Huge APP (3MB No OTA / 1MB SPIFFS)

Un cop ajustat tot això, el programa funciona correctament i veurem la següent informació en el port sèrie.



A continuació ens connectarem a la IP senyalada i podrem fer ús del programa.
(Codi complet a [Annex E])
Un cop el programa estava en funcionament, vam començar a investigar per tal d'aconseguir que, amb el mateix programa o un de molt similar es poguesssin reconèixer senyals de trànsit i actuar tal com indiquen. Aquí però, ens vam trobar un mur molt difícil d'escalar amb el coneixement i els recursos que posseíem. Per tal de portar a la pràctica aquesta idea, s'havia de crear una gran base de dades amb les senyals que volguèssim que el programa reconegués, etiquetar les imatges, crear una nova xarxa neuronal i entrenar-la. A més, aplicar-la a l'ESP32 i, finalment, connectar-la amb l'altra placa.

Un cop vam adonar-nos d'això, vam decidir ser creatius i buscar una solució que ens permetés crear el programa que volíem amb els recursos limitats dels que disposàvem. Així va néixer la idea desenvolupada en el projecte final. No sense abans haver estat treballant en un gran conjunt de coses que, al final, no van arribar enlloc (Explicades amb més detall a l'Annex [Annex B]).

3.4. Projecte final

La idea per al projecte final consisteix en utilitzar les xarxes neuronals amb les que hem treballat anteriorment per tal de crear un vehicle que pugui evitar obstacles i, en veure una senyal de trànsit, fer el que indiqui. El problema, com hem dit abans és la dificultat de realitzar aquesta tasca amb senyals de trànsit. Per tant, les substituïm per cares humanes.

Quan el robot detecti la cara d'una persona etiquetada com a *Subject 0*, aquest, per exemple, girarà a la dreta.

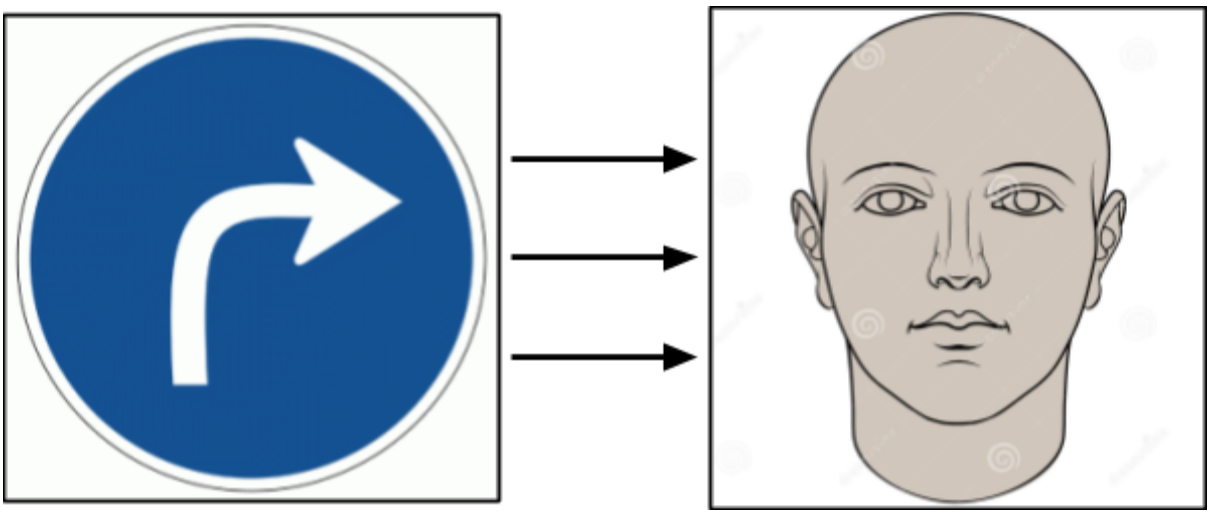


Figura 3.13. Cares com a senyals de trànsit

Els principals problemes que hem de solucionar per aconseguir que funcioni correctament són 2:

- Fer reaccionar els motors en detectar la cara corresponent
- Connectar les dues plaques per enviar aquesta informació (la càmera detecta la cara però és el NodeMCU qui ha d'actuar).

Per solucionar el primer problema, primer s'ha de llegir el codi molt a fons i entendre com funciona, al menys, una gran part del mateix (ja que al ser un codi força complex i d'una persona amb la seva forma de programar, és difícil comprendre'l en la seva totalitat).

La funció que modificarem serà *"run_face_recognition"*, que s'encarrega del reconeixement facial. En aquesta funció, estan definits diferents casos, segons el programa estigui enregistrant una nova cara o estigui detectant una que sigui, o no, coneguda. També tindrem en compte una variable anomenada *"matched_id"*, que anirà canviant (entre els diferents nombres naturals: 1,2,3...) segons el *Subject* que estigui detectant. Un cop sabem això, creem un apartat en el codi en que declarem el següent:

```
if (matched_id == 1) {
            //Gira a la dreta
          }
```

Quan la càmera detecti un determinat subjecte, el vehicle, per exemple, girarà a la dreta. Ara hem de transmetre aquesta informació al vehicle: hem de solucionar la segona gran dificultat.

Per fer-ho, declarem 2 noves variables que corresponen a dos pins, que declararem com a OUTPUT. Després, completem el codi d'abans fent que cada pin enviï al port sèrie una senyal de HIGH o LOW.

```
int P1 = 14;
int P2 = 15;

void setup (){
   pinMode(P1, OUTPUT);
   pinMode(P2, OUTPUT);

   Serial.begin(115200);
}

if (matched_id == 1) {
              digitalWrite(P1,HIGH);
             }

 if (matched_id == 2) {
              digitalWrite(P2,HIGH);
              }
```

Com hem vist, quan el programa reconegui el primer subjecte, enviarà una senyal de HIGH a P1, mentre que quan reconegui el segon, l'enviarà a P2.

A continuació, modificarem el codi que controlarà el vehicle, on ja tenim diverses funcions pròpies per moure'l (endavant, endarrere, dreta o esquerra). Repetim el procediment i declarem dues variables amb els pins, aquest cop com a INPUT, ja que estem rebent informació. El següent pas serà crear dues variables (V1 i V2) que llegeixin la informació rebuda, és a dir, la informació emesa per l'ESP32. Finalment, crearem un bucle *"if"* el cual, segons la combinació dels valors de V1 i V2, activarà una funció o una altra.

```
int P1 = D5;
int P2 = D6;

void setup() {
  pinMode(5, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(0, OUTPUT);
```

```
  pinMode(2, OUTPUT);

  pinMode(P1,INPUT);
  pinMode(P2,INPUT);

  Serial.begin(115200);
}

void moveForward(){
  analogWrite(5, 1023);
  analogWrite(4, 1023);

  digitalWrite(0, 1);
  digitalWrite(2, 1);

  delay(100);
  }

void moveBackwards(){
  analogWrite(5, 1023);
  analogWrite(4, 1023);

  digitalWrite(0, 0);
  digitalWrite(2, 0);

  delay(100);
  }


void turnRight (){
  analogWrite(5, 1023);
  analogWrite(4, 1023);

  digitalWrite(0, 1);
  digitalWrite(2, 0);

  delay(100);
  }

void turnLeft (){
  analogWrite(5, 1023);
  analogWrite(4, 1023);

  digitalWrite(0, 0);
  digitalWrite(2, 1);

  delay(100);
  }

void loop(){
```

```
   moveForward;

int V1 = digitalRead(P1);
int V2 = digitalRead(P2);

  if (V1 == LOW){
     turnRight;
     }

  if (V2 == LOW){
     turnLeft;
     }

}
```

En primer lloc, provem el codi utilitzant dos botons com a input, ja que el codi necessari és el mateix i això ens permetrà comprovar el seu funcionament. Quan no premem els botons, emeten una senyal HIGH, per això escrivim els bucles *If* per a quan la senyal rebuda és LOW (quan estem prement els botons).

Un cop això funciona, afegim també els codis del servo i el sensor d'ultrasons juntament amb unes petites instruccions de funcionament: quan la distància entre el vehicle i un obstacle sigui menor a 30 cm (en aquest cas), moure's cap endarrere. Per qualsevol altre cas, moure's endavant. A més, afegim una línia de codi darrere cada acció per poder veure en el port sèrie què detecten els sensors i què fa el vehicle.

Hem de vigilar en que els pins no es repeteixin, per això, canviem els pins del sensor d'ultrasons a D7 i D0.

```
#define trigPin D7
#define echoPin D0
#include <Servo.h>
Servo myservo;
int P1 = D5;
int P2 = D6;

void setup() {
      pinMode(trigPin, OUTPUT);
      pinMode(echoPin, INPUT);
  pinMode(5, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(0, OUTPUT);
  pinMode(2, OUTPUT);
```

```
  pinMode(P1,INPUT);
  pinMode(P2,INPUT);

  myservo.attach(15);

  Serial.begin(115200);
}

void moveForward(){
  analogWrite(5, 1023);
  analogWrite(4, 1023);

  digitalWrite(0, 1);
  digitalWrite(2, 1);

  delay(100);

  }

void moveBackwards(){
  analogWrite(5, 1023);
  analogWrite(4, 1023);

  digitalWrite(0, 0);
  digitalWrite(2, 0);

  delay(100);
  }

void turnRight(){
  analogWrite(5, 1023);
  analogWrite(4, 1023);

  digitalWrite(0, 1);
  digitalWrite(2, 0);

  delay(100);
  }

void turnLeft(){
  analogWrite(5, 1023);
  analogWrite(4, 1023);

  digitalWrite(0, 0);
  digitalWrite(2, 1);

  delay(100);
  }
```

```
void loop(){
   long duracion, distancia ;
          digitalWrite(trigPin, LOW);       // Nos aseguramos de que el trigger está
desactivado
          delayMicroseconds(2);             // Para asegurarnos de que el trigger esta LOW
          digitalWrite(trigPin, HIGH);      // Activamos el pulso de salida
          delayMicroseconds(10);            // Esperamos 10µs. El pulso sigue active este
tiempo
          digitalWrite(trigPin, LOW);       // Cortamos el pulso y a esperar el echo
          duracion = pulseIn(echoPin, HIGH) ;
          distancia = duracion / 2 / 29.1  ;
          Serial.println(String(distancia) + " cm.") ;
          delay(5);

   moveForward;

  if (distancia < 30){
   moveBackwards;
   Serial.println("ENDARRERE");
   }
   else {
     moveForward;
     Serial.println("ENDAVANT");
     }

int V1 = digitalRead(P1);
int V2 = digitalRead(P2);

   if (V1 == LOW){
     turnRight;
     Serial.println("Gir a la Dreta");
     }

   if (V2 == LOW){
     turnLeft;
     Serial.println("Gir a l'Esquerra");
     }

  int pos;
  for (pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);             // tell servo to go to position in variable 'pos'
    delay(15);     }                // waits 15ms for the servo to reach the position
  for (pos = 90; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);             // tell servo to go to position in variable 'pos'
    delay(15);       }              // waits 15ms for the servo to reach the position

       }
```

Aquest és l'aspecte del port sèrie quan posem el vehicle en funcionament:

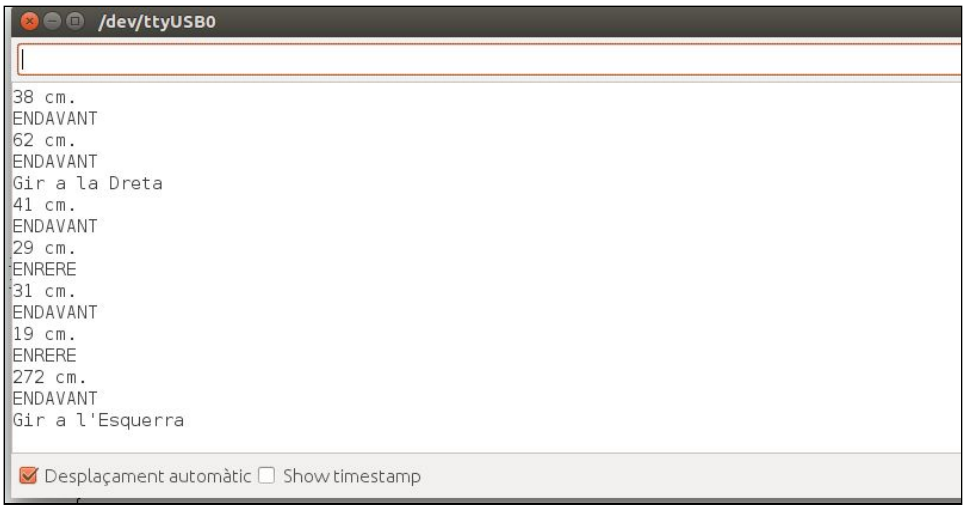


Figura 3.14. Comprovació de funcionament amb el port sèrie

Finalment, connectem la càmera (amb el codi ja carregat), als pins corresponents de la placa NodeMCU.

Aleshores, haurem creat un vehicle que s'anirà movent sense xocar amb els obstacles que trobi pel camí i, a més, podrà seguir les indicacions d'algunes "senyals" fetes amb imatges de cares humanes que haurà de memoritzar.

Capítol 4. Conclusions

4.1. Conclusió del projecte

Tot i la gran quantitat de treball que hi ha al darrere d'aquest projecte, hi ha encara problemes a solucionar i possibles millores a fer.

El principal problema és que el programa no pot distingir entre els diferents *Subjects* que detecta la càmera tot i haver manipulat el codi perquè ho pugui fer, per tant, només serà efectiu si li assignem una única ordre.

A partir d'aquí la resta són millores que, tot i optimitzar el funcionament del vehicle, no són indispensables per al mateix. Per exemple, memoritzar les cares que actuen com a senyals i guardar les imatges en una targeta SD per no haver-les de memoritzar de nou cada cop que

volem utilitzar el vehicle, cosa que, tot i estar força temps provant, no vam obtenir resultats favorables [Annex B].

A més, podríem intentar modificar la xarxa neuronal i el programa d'arduino del primer exemple per tal de combinar-lo amb el programa del projecte final.

Una altra possible millora seria el disseny d'un nou xassís pel vehicle que permetés acoplar tots els components utilitzats amb facilitat (motors, sensors, càmera, servo).

Una tercera millora, tot i que força més complicada que les anteriors, seria el programar una nova xarxa neuronal per tal que reconegués les senyals (ara sí) de trànsit.

Malgrat tot, hem aconseguit crear un vehicle que pot circular sense xocar amb els obstacles al seu camí i que pot seguir alguna indicació en forma de cara, un producte que, amb unes certes millores, és la base del que s'està investigant en conducció autònoma i que és força semblant als vehicles que trobarem en un futur pels carrers de totes les ciutats del món.

4.2. Valoració personal

Aquest ha estat un projecte molt dur, per la quantitat de treball i d'hores que he hagut de dedicar. No estic del tot satisfet amb el resultat, ja que m'hagués agradat poder aconseguir un producte molt més complet i millor acabat, però sí que valoro molt tot el que he après. Vull destacar, per sobre dels coneixements més tècnics que hem estat comentant al llarg de tot l'informe, les altres coses que he après amb aquest treball.

He après a enfrontar-me als treballs d'una altra manera, a saber que si m'espero fer alguna cosa en un temps determinat, trigaré molt més, i a estar preparat per això.

També he après a esperar-me problemes. M'explico: a l'hora de per exemple compilar un codi d'arduino o d'enviar-lo a la placa, l'única cosa que espero es veure el problema que hi ha per tal de solucionar-lo de la millor i més ràpida forma possible, no m'espero que tot surti bé.

D'altra banda he acomplert la majoria dels objectius que m'havia plantejat en un inici, a l'hora de començar el treball. He après molt tant sobre el tema del treball com de moltes

altres coses, he realitzat un projecte en el que intento aplicar tot allò que he après i, posteriorment, he explicat tot el procés en un bon informe.

Finalment, he adquirit habilitats per buscar informació, treballar pel meu compte, ... En definitiva, i de forma col·loquial, he après a "buscar-me la vida", habilitat que crec que em serà molt útil per al meu futur com a estudiant i com a ésser humà.

Malgrat tot el que m'ha aportat el projecte i les coses que considero que he fet bé, hi ha també detalls que hauria d'haver fet millor ja que haguessin ajudat a realitzar un millor projecte. El principal és la concreció del projecte, ja que vaig trigar molt en decidir concretament què volia fer, el que ha ocasionat que hagi dedicat temps a investigar o a treballar sobre coses que no he acabat utilitzant i hagi pogut dedicar-ne el suficient a coses realment importants.
També penso que en un inici tenia unes aspiracions massa altes sobre el projecte a realitzar i no m'adonava de la dificultat real d'un treball d'aquesta magnitud, cosa que va contribuir a trigar més en prendre la decisió definitiva.

Com a reflexió final, si hagués de definir aquest treball en una paraula possiblement seria "aprenentatge" (o "esforç") ja que puc afirmar que he après molt en general i que els errors que he comès, tot i molestar-me, també m'han servit per, un cop més, aprendre.

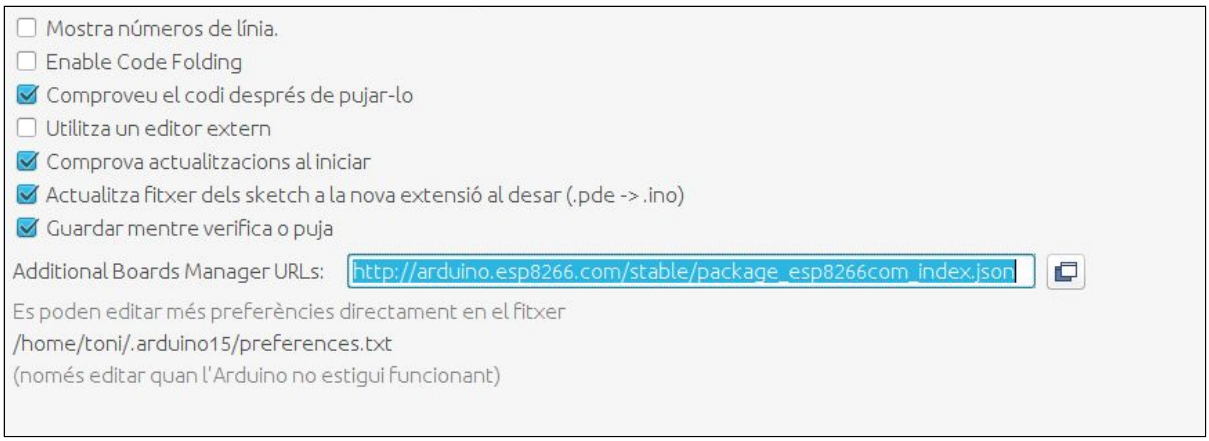
Capítol 5. Annexos i bibliografia

5.1. Annexos

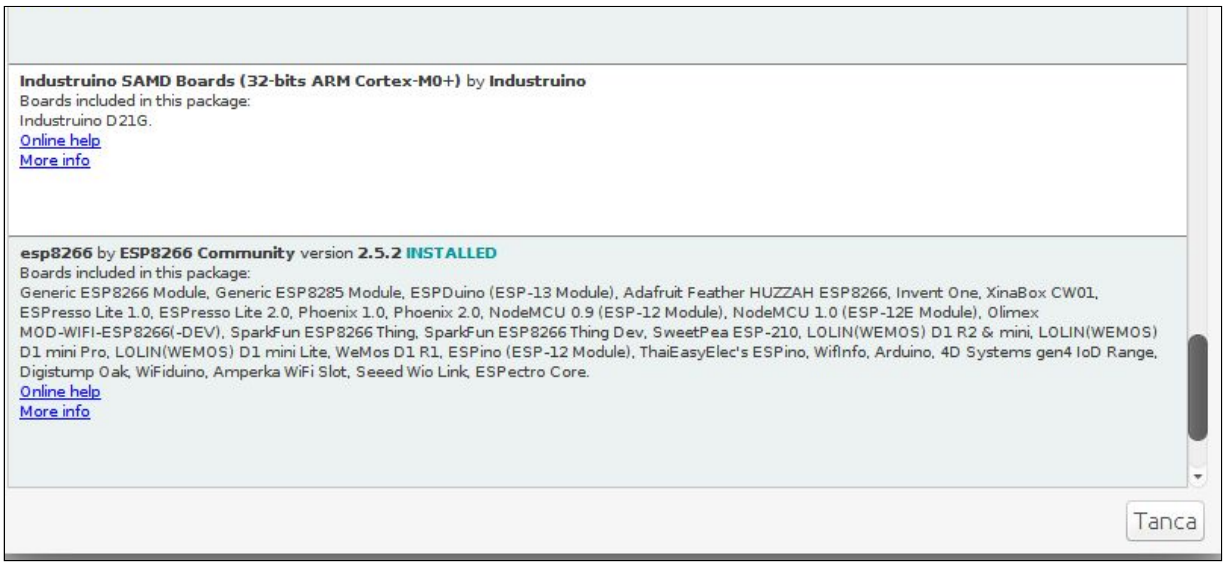
[A] Instal·lació de software especialitzat d'arduino

Per instal·lar l'ESP8266 hem d'utilitzar una versió d'Arduino que ens permeti afegir noves plaques.
Quan en tinguem una instal·lada, obrim l'IDE i anem a *Fitxer* + *Preferències*.
Allà escrivim la següent adreça a *Additional Boards Manager URLs*:

http://arduino.esp8266.com/stable/package_esp8266com_index.json



A continuació, anem a *Eines* + *Tarja* + *Gestor targes*, i seleccionem esp8266 per instal·lar-la.



Un cop instal·lada, ja podem fer-ne ús de tots els recursos que ens proporciona.

Instal·lar l'esp32 es fa de forma molt semblant. De fet, de forma idéntica, però hem de tenir molt més en compte la versió d'Arduino que utilitzem. Al començament, treballàvem amb la 1.6.9 i donava una sèrie d'errors que no sabíem identificar. Per aquest motiu, vam decidir descarregar i instal·lar una nova versió d'Arduino, la 1.8.10. Un cop fet això, tot va funcionar correctament.

El link que s'ha d'insertar per instal·lar l'esp32 és:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

[B] Proves fallides

Per dur a terme el projecte hi ha moltes idees sobre les quals hem treballat pero que al final, per diversos motius, no han arribat a funcionar.

- Connexió arduino amo-esclau: amb aquesta idea preteniem connectar les dues plaques mitjançant una connexió amo-esclau, en la qual la placa de la càmera estava subordinada a les ordres del seu amo, l'ESP8266. L'amo havia de demanar la informació a la càmera sobre si aquesta estava reconeixent alguna cara i, en cas afirmatiu, quin número era per, segons el cas, executar un o un altre comandament i moure el vehicle en conseqüència. Finalment van sorgir una sèrie de problemes que no sabíem solucionar i la idea va ser descartada.
Més informació a:
https://sites.google.com/a/iepegasoviana.cat/sensors-arduino/comunicacions/arduino-i2c-slave

- SoftwareSerial: un altra intent de realitzar aquesta connexió va ser mitjançant la llibreria SoftwareSerial, amb la que podíem comunicar les dues plaques a través d'un port sèrie digital. La idea no va arribar a bon port degut a certes incompatibilitats entre llibreries usades en els diferents programes.

- Guardar imatges a la SD: vam estar intentant guardar les imatges que la càmera de l'ESP32 reconeixia en una targeta SD. Primer vam estar provant diferents programes per, simplement guardar les imatges i més endavant vam voler combinar-lo amb el reconeixement facial, però a l'hora de la veritat no vam saber fer-ho, tot i investigar molt i tenir alguns resultats exitosos (vam poder guardar imatges a la sd però no que les llegís el programa).
Més informació a: https://randomnerdtutorials.com/esp32-cam-take-photo-save-microsd-card/
- Yolo: al començar a investigar el reconeixement d'imatges vam trobar YOLO (*you only look once*, en anglès, només mires un cop) un programa que permetia reconèixer tot tipus d'objectes en una imatge, un video o una grabació en directe. Vam dur a terme moltes proves exitoses amb el programa, però al final va ser descartat per la impossibilitat d'utilitzar-lo amb l'ESP32 i de, posteriorment fer actuar un vehicle segons el que detectés.
Més informació a: https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/

[C] Llibreries

A continuació podem trobar el codi de les llibreries utilitzades més importants:

- **Servo.h**

```
/*
  Servo.h - Interrupt driven Servo library for Arduino using 16 bit timers- Version 2
  Copyright (c) 2009 Michael Margolis.  All right reserved.
```

```
  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.
  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  Lesser General Public License for more details.
  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/

/*
  A servo is activated by creating an instance of the Servo class passing
  the desired pin to the attach() method.
  The servos are pulsed in the background using the value most recently
  written using the write() method.
  Note that analogWrite of PWM on pins associated with the timer are
  disabled when the first servo is attached.
  Timers are seized as needed in groups of 12 servos - 24 servos use two
  timers, 48 servos will use four.
  The sequence used to sieze timers is defined in timers.h
  The methods are:
    Servo - Class for manipulating servo motors connected to Arduino pins.
    attach(pin )  - Attaches a servo motor to an i/o pin.
    attach(pin, min, max  ) - Attaches to a pin setting min and max values in microseconds
    default min is 544, max is 2400

    write()     - Sets the servo angle in degrees.  (invalid angle that is valid as pulse in microseconds is treated as
microseconds)
    writeMicroseconds() - Sets the servo pulse width in microseconds
    read()      - Gets the last written servo pulse width as an angle between 0 and 180.
    readMicroseconds()   - Gets the last written servo pulse width in microseconds. (was read_us() in first release)
    attached()  - Returns true if there is a servo attached.
    detach()    - Stops an attached servos from pulsing its i/o pin.
 */

#ifndef Servo_h
#define Servo_h

#include <inttypes.h>

/*
 * Defines for 16 bit timers used with  Servo library
 *
 * If _useTimerX is defined then TimerX is a 16 bit timer on the current board
 * timer16_Sequence_t enumerates the sequence that the timers should be allocated
 * _Nbr_16timers indicates how many 16 bit timers are available.
 */

// Architecture specific include
#if defined(ARDUINO_ARCH_AVR)
#include "avr/ServoTimers.h"
#elif defined(ARDUINO_ARCH_SAM)
#include "sam/ServoTimers.h"
#elif defined(ARDUINO_ARCH_SAMD)
#include "samd/ServoTimers.h"
#elif defined(ARDUINO_ARCH_STM32F4)
#include "stm32f4/ServoTimers.h"
#elif defined(ARDUINO_ARCH_NRF52)
#include "nrf52/ServoTimers.h"
#elif defined(ARDUINO_ARCH_MEGAAVR)
#include "megaavr/ServoTimers.h"
#elif defined(ARDUINO_ARCH_MBED)
```

```
#include "mbed/ServoTimers.h"
#else
#error "This library only supports boards with an AVR, SAM, SAMD, NRF52 or STM32F4 processor."
#endif

#define Servo_VERSION           2     // software version of this library

#define MIN_PULSE_WIDTH       544     // the shortest pulse sent to a servo
#define MAX_PULSE_WIDTH      2400     // the longest pulse sent to a servo
#define DEFAULT_PULSE_WIDTH  1500     // default pulse width when servo is attached
#define REFRESH_INTERVAL    20000     // minumim time to refresh servos in microseconds

#define SERVOS_PER_TIMER       12     // the maximum number of servos controlled by one timer
#define MAX_SERVOS   (_Nbr_16timers  * SERVOS_PER_TIMER)

#define INVALID_SERVO         255     // flag indicating an invalid servo index

#if !defined(ARDUINO_ARCH_STM32F4)

typedef struct  {
  uint8_t nbr        :6 ;             // a pin number from 0 to 63
  uint8_t isActive   :1 ;             // true if this channel is enabled, pin not pulsed if false
} ServoPin_t   ;

typedef struct {
  ServoPin_t Pin;
  volatile unsigned int ticks;
} servo_t;

class Servo
{
public:
  Servo();
  uint8_t attach(int pin);           // attach the given pin to the next free channel, sets pinMode, returns channel
number or 0 if failure
  uint8_t attach(int pin, int min, int max); // as above but also sets min and max values for writes.
  void detach();
  void write(int value);             // if value is < 200 its treated as an angle, otherwise as pulse width in microseconds
  void writeMicroseconds(int value); // Write pulse width in microseconds
  int read();                        // returns current pulse width as an angle between 0 and 180 degrees
  int readMicroseconds();            // returns current pulse width in microseconds for this servo (was read_us() in
first release)
  bool attached();                   // return true if this servo is attached, otherwise false
private:
   uint8_t servoIndex;               // index into the channel data for this servo
   int8_t min;                       // minimum is this value times 4 added to MIN_PULSE_WIDTH
   int8_t max;                       // maximum is this value times 4 added to MAX_PULSE_WIDTH
};

#endif
#endif
```

- **esp_camera.h**

```
// Copyright 2015-2016 Espressif Systems (Shanghai) PTE LTD
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
```

```
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.
/*
 * Example Use
 *
    static camera_config_t camera_example_config = {
        .pin_pwdn       = PIN_PWDN,
        .pin_reset      = PIN_RESET,
        .pin_xclk       = PIN_XCLK,
        .pin_sscb_sda   = PIN_SIOD,
        .pin_sscb_scl   = PIN_SIOC,
        .pin_d7         = PIN_D7,
        .pin_d6         = PIN_D6,
        .pin_d5         = PIN_D5,
        .pin_d4         = PIN_D4,
        .pin_d3         = PIN_D3,
        .pin_d2         = PIN_D2,
        .pin_d1         = PIN_D1,
        .pin_d0         = PIN_D0,
        .pin_vsync      = PIN_VSYNC,
        .pin_href       = PIN_HREF,
        .pin_pclk       = PIN_PCLK,
        .xclk_freq_hz   = 20000000,
        .ledc_timer     = LEDC_TIMER_0,
        .ledc_channel   = LEDC_CHANNEL_0,
        .pixel_format   = PIXFORMAT_JPEG,
        .frame_size     = FRAMESIZE_SVGA,
        .jpeg_quality   = 10,
        .fb_count       = 2
    };
    esp_err_t camera_example_init(){
        return esp_camera_init(&camera_example_config);
    }
    esp_err_t camera_example_capture(){
        //capture a frame
        camera_fb_t * fb = esp_camera_fb_get();
        if (!fb) {
            ESP_LOGE(TAG, "Frame buffer could not be acquired");
            return ESP_FAIL;
        }
        //replace this with your own function
        display_image(fb->width, fb->height, fb->pixformat, fb->buf, fb->len);
        //return the frame buffer back to be reused
        esp_camera_fb_return(fb);
        return ESP_OK;
    }
*/

#pragma once

#include "esp_err.h"
#include "driver/ledc.h"
#include "sensor.h"

#ifdef __cplusplus
extern "C" {
#endif

/**
 * @brief Configuration structure for camera initialization
 */
typedef struct {
    int pin_pwdn;                /*!< GPIO pin for camera power down line */
```

```
    int pin_reset;          /*!< GPIO pin for camera reset line */
    int pin_xclk;           /*!< GPIO pin for camera XCLK line */
    int pin_sscb_sda;       /*!< GPIO pin for camera SDA line */
    int pin_sscb_scl;       /*!< GPIO pin for camera SCL line */
    int pin_d7;             /*!< GPIO pin for camera D7 line */
    int pin_d6;             /*!< GPIO pin for camera D6 line */
    int pin_d5;             /*!< GPIO pin for camera D5 line */
    int pin_d4;             /*!< GPIO pin for camera D4 line */
    int pin_d3;             /*!< GPIO pin for camera D3 line */
    int pin_d2;             /*!< GPIO pin for camera D2 line */
    int pin_d1;             /*!< GPIO pin for camera D1 line */
    int pin_d0;             /*!< GPIO pin for camera D0 line */
    int pin_vsync;          /*!< GPIO pin for camera VSYNC line */
    int pin_href;           /*!< GPIO pin for camera HREF line */
    int pin_pclk;           /*!< GPIO pin for camera PCLK line */

    int xclk_freq_hz;       /*!< Frequency of XCLK signal, in Hz. Either 20KHz or 10KHz for OV2640 double
FPS (Experimental) */

    ledc_timer_t ledc_timer;    /*!< LEDC timer to be used for generating XCLK  */
    ledc_channel_t ledc_channel;    /*!< LEDC channel to be used for generating XCLK  */

    pixformat_t pixel_format;   /*!< Format of the pixel data: PIXFORMAT_ +
YUV422|GRAYSCALE|RGB565|JPEG  */
    framesize_t frame_size;     /*!< Size of the output image: FRAMESIZE_ +
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA  */

    int jpeg_quality;       /*!< Quality of JPEG output. 0-63 lower means higher quality  */
    size_t fb_count;        /*!< Number of frame buffers to be allocated. If more than one, then each frame will
be acquired (double speed)  */
} camera_config_t;

/**
 * @brief Data structure of camera frame buffer
 */
typedef struct {
    uint8_t * buf;          /*!< Pointer to the pixel data */
    size_t len;             /*!< Length of the buffer in bytes */
    size_t width;           /*!< Width of the buffer in pixels */
    size_t height;          /*!< Height of the buffer in pixels */
    pixformat_t format;     /*!< Format of the pixel data */
} camera_fb_t;

#define ESP_ERR_CAMERA_BASE 0x20000
#define ESP_ERR_CAMERA_NOT_DETECTED         (ESP_ERR_CAMERA_BASE + 1)
#define ESP_ERR_CAMERA_FAILED_TO_SET_FRAME_SIZE (ESP_ERR_CAMERA_BASE + 2)
#define ESP_ERR_CAMERA_FAILED_TO_SET_OUT_FORMAT (ESP_ERR_CAMERA_BASE + 3)
#define ESP_ERR_CAMERA_NOT_SUPPORTED        (ESP_ERR_CAMERA_BASE + 4)

/**
 * @brief Initialize the camera driver
 *
 * @note call camera_probe before calling this function
 *
 * This function detects and configures camera over I2C interface,
 * allocates framebuffer and DMA buffers,
 * initializes parallel I2S input, and sets up DMA descriptors.
 *
 * Currently this function can only be called once and there is
 * no way to de-initialize this module.
 *
 * @param config  Camera configuration parameters
 *
 * @return ESP_OK on success
 */
```

```
esp_err_t esp_camera_init(const camera_config_t* config);

/**
 * @brief Deinitialize the camera driver
 *
 * @return
 *      - ESP_OK on success
 *      - ESP_ERR_INVALID_STATE if the driver hasn't been initialized yet
 */
esp_err_t esp_camera_deinit();

/**
 * @brief Obtain pointer to a frame buffer.
 *
 * @return pointer to the frame buffer
 */
camera_fb_t* esp_camera_fb_get();

/**
 * @brief Return the frame buffer to be reused again.
 *
 * @param fb    Pointer to the frame buffer
 */
void esp_camera_fb_return(camera_fb_t * fb);

/**
 * @brief Get a pointer to the image sensor control structure
 *
 * @return pointer to the sensor
 */
sensor_t * esp_camera_sensor_get();


#ifdef __cplusplus
}
#endif

#include "img_converters.h"
```

- **img_converters.h**

```
// Copyright 2015-2016 Espressif Systems (Shanghai) PTE LTD
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.
#ifndef _IMG_CONVERTERS_H_
#define _IMG_CONVERTERS_H_

#ifdef __cplusplus
extern "C" {
#endif

#include <stddef.h>
#include <stdint.h>
```

```
#include <stdbool.h>
#include "esp_camera.h"

typedef size_t (* jpg_out_cb)(void * arg, size_t index, const void* data, size_t len);

/**
 * @brief Convert image buffer to JPEG
 *
 * @param src       Source buffer in RGB565, RGB888, YUYV or GRAYSCALE format
 * @param src_len   Length in bytes of the source buffer
 * @param width     Width in pixels of the source image
 * @param height    Height in pixels of the source image
 * @param format    Format of the source image
 * @param quality   JPEG quality of the resulting image
 * @param cp        Callback to be called to write the bytes of the output JPEG
 * @param arg       Pointer to be passed to the callback
 *
 * @return true on success
 */
bool fmt2jpg_cb(uint8_t *src, size_t src_len, uint16_t width, uint16_t height, pixformat_t format, uint8_t quality,
jpg_out_cb cb, void * arg);

/**
 * @brief Convert camera frame buffer to JPEG
 *
 * @param fb        Source camera frame buffer
 * @param quality   JPEG quality of the resulting image
 * @param cp        Callback to be called to write the bytes of the output JPEG
 * @param arg       Pointer to be passed to the callback
 *
 * @return true on success
 */
bool frame2jpg_cb(camera_fb_t * fb, uint8_t quality, jpg_out_cb cb, void * arg);

/**
 * @brief Convert image buffer to JPEG buffer
 *
 * @param src       Source buffer in RGB565, RGB888, YUYV or GRAYSCALE format
 * @param src_len   Length in bytes of the source buffer
 * @param width     Width in pixels of the source image
 * @param height    Height in pixels of the source image
 * @param format    Format of the source image
 * @param quality   JPEG quality of the resulting image
 * @param out       Pointer to be populated with the address of the resulting buffer
 * @param out_len   Pointer to be populated with the length of the output buffer
 *
 * @return true on success
 */
bool fmt2jpg(uint8_t *src, size_t src_len, uint16_t width, uint16_t height, pixformat_t format, uint8_t quality,
uint8_t ** out, size_t * out_len);

/**
 * @brief Convert camera frame buffer to JPEG buffer
 *
 * @param fb        Source camera frame buffer
 * @param quality   JPEG quality of the resulting image
 * @param out       Pointer to be populated with the address of the resulting buffer
 * @param out_len   Pointer to be populated with the length of the output buffer
 *
 * @return true on success
 */
bool frame2jpg(camera_fb_t * fb, uint8_t quality, uint8_t ** out, size_t * out_len);

/**
 * @brief Convert image buffer to BMP buffer
```

```
 *
 * @param src       Source buffer in JPEG, RGB565, RGB888, YUYV or GRAYSCALE format
 * @param src_len   Length in bytes of the source buffer
 * @param width     Width in pixels of the source image
 * @param height    Height in pixels of the source image
 * @param format    Format of the source image
 * @param out       Pointer to be populated with the address of the resulting buffer
 * @param out_len   Pointer to be populated with the length of the output buffer
 *
 * @return true on success
 */
bool fmt2bmp(uint8_t *src, size_t src_len, uint16_t width, uint16_t height, pixformat_t format, uint8_t ** out, size_t
* out_len);

/**
 * @brief Convert camera frame buffer to BMP buffer
 *
 * @param fb        Source camera frame buffer
 * @param out       Pointer to be populated with the address of the resulting buffer
 * @param out_len   Pointer to be populated with the length of the output buffer
 *
 * @return true on success
 */
bool frame2bmp(camera_fb_t * fb, uint8_t ** out, size_t * out_len);

/**
 * @brief Convert image buffer to RGB888 buffer (used for face detection)
 *
 * @param src       Source buffer in JPEG, RGB565, RGB888, YUYV or GRAYSCALE format
 * @param src_len   Length in bytes of the source buffer
 * @param format    Format of the source image
 * @param rgb_buf   Pointer to the output buffer (width * height * 3)
 *
 * @return true on success
 */
bool fmt2rgb888(const uint8_t *src_buf, size_t src_len, pixformat_t format, uint8_t * rgb_buf);

#ifdef __cplusplus
}
#endif

#endif /* _IMG_CONVERTERS_H_ */
```

[D] Instal·lació d'un entorn de programació amb python

Python és un llenguatge de programació especialment útil per a treballar amb intel·ligència artificial. Per treballar amb ell el més còmode és instal·lar un entorn de programació que ens faciliti fer-ho.

Per instal·lar python podem descarregar-lo directament des de la pàgina web oficial o amb el comandament

```
sudo apt install python3.7
```

Un del multiples entorns de desenvolupament per a python és Atom, el qual podem descarregar seguint les instruccions especificades en aquest enllaç:
https://atom.io/packages/ide-python

[E] Codi

- **Codi Xarxa Neuronal Cotxe Arduino**

```
import numpy as np

def sigmoid(x):
    return 1.0/(1.0 + np.exp(-x))

def sigmoid_derivada(x):
    return sigmoid(x)*(1.0-sigmoid(x))

def tanh(x):
    return np.tanh(x)

def tanh_derivada(x):
    return 1.0 - x**2


class NeuralNetwork:

    def __init__(self, layers, activation='tanh'):
        if activation == 'sigmoid':
            self.activation = sigmoid
            self.activation_prime = sigmoid_derivada
        elif activation == 'tanh':
            self.activation = tanh
            self.activation_prime = tanh_derivada

        # inicializo los pesos
        self.weights = []
        self.deltas = []
        # capas = [2,3,2]
        # rando de pesos varia entre (-1,1)
        # asigno valores aleatorios a capa de entrada y capa oculta
        for i in range(1, len(layers) - 1):
            r = 2*np.random.random((layers[i-1] + 1, layers[i] + 1)) -1
            self.weights.append(r)
        # asigno aleatorios a capa de salida
        r = 2*np.random.random( (layers[i] + 1, layers[i+1])) - 1
        self.weights.append(r)

    def fit(self, X, y, learning_rate=0.2, epochs=100000):
        # Agrego columna de unos a las entradas X
        # Con esto agregamos la unidad de Bias a la capa de entrada
        ones = np.atleast_2d(np.ones(X.shape[0]))
        X = np.concatenate((ones.T, X), axis=1)

        for k in range(epochs):
            i = np.random.randint(X.shape[0])
            a = [X[i]]

            for l in range(len(self.weights)):
                    dot_value = np.dot(a[l], self.weights[l])
                    activation = self.activation(dot_value)
```

```
                a.append(activation)
            # Calculo la diferencia en la capa de salida y el valor obtenido
            error = y[i] - a[-1]
            deltas = [error * self.activation_prime(a[-1])]

            # Empezamos en el segundo layer hasta el ultimo
            # (Una capa anterior a la de salida)
            for l in range(len(a) - 2, 0, -1):
                deltas.append(deltas[-1].dot(self.weights[l].T)*self.activation_prime(a[l]))
            self.deltas.append(deltas)

            # invertir
            # [level3(output)->level2(hidden)]  => [level2(hidden)->level3(output)]
            deltas.reverse()

            # backpropagation
            # 1. Multiplcar los delta de salida con las activaciones de entrada
            #    para obtener el gradiente del peso.
            # 2. actualizo el peso restandole un porcentaje del gradiente
            for i in range(len(self.weights)):
                layer = np.atleast_2d(a[i])
                delta = np.atleast_2d(deltas[i])
                self.weights[i] += learning_rate * layer.T.dot(delta)

            if k % 10000 == 0: print('epochs:', k)

    def predict(self, x):
        ones = np.atleast_2d(np.ones(x.shape[0]))
        a = np.concatenate((np.ones(1).T, np.array(x)), axis=0)
        for l in range(0, len(self.weights)):
            a = self.activation(np.dot(a, self.weights[l]))
        return a

    def print_weights(self):
        print("LISTADO PESOS DE CONEXIONES")
        for i in range(len(self.weights)):
            print(self.weights[i])

    def get_deltas(self):
        return self.deltas

# funcion Coche Evita obstáculos
nn = NeuralNetwork([2,3,2],activation ='tanh')
X = np.array([[0, 0],   # sin obstaculos
              [0, 1],   # sin obstaculos
              [0, -1],  # sin obstaculos
              [0.5, 1], # obstaculo detectado a derecha
              [0.5,-1], # obstaculo a izq
              [1,1],    # demasiado cerca a derecha
              [1,-1]])  # demasiado cerca a izq

y = np.array([[0,1],    # avanzar
              [0,1],    # avanzar
              [0,1],    # avanzar
              [-1,1],   # giro izquierda
              [1,1],    # giro derecha
              [0,-1],   # retroceder
              [0,-1]])  # retroceder
nn.fit(X, y, learning_rate=0.03,epochs=15001)

index=0
for e in X:
    print("X:",e,"y:",y[index],"Network:",nn.predict(e))
    index=index+1
```

- **Codi cotxe arduino conduit amb xarxa neuronal**

```
#include <Servo.h>  //servo library
Servo myservo;      // create servo object to control servo

int Echo = A4;
int Trig = A5;
#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11

/*********************************************************************
  Network Configuration
 *********************************************************************/
const int InputNodes = 3; // incluye neurona de BIAS
const int HiddenNodes = 4; //incluye neurona de BIAS
const int OutputNodes = 4;
int i, j;
double Accum;
double Hidden[HiddenNodes];
double Output[OutputNodes];
float HiddenWeights[3][4] = {{1.8991509504079183, -0.4769472541445052, -0.6483690220539764,
-0.38609165249078925}, {-0.2818610915467527, 4.040695699457223, 3.2291858058243843,
-2.894301104732614}, {0.3340650864625773, -1.4016114422346901, 1.3580053902963762,
-0.981415976256285}};
float OutputWeights[4][4] = {{1.136072297461121, 1.54602394937381, 1.6194612259569254,
1.8819066696635067}, {-1.546966506764457, 1.3951930739494225, 0.19393826092602756,
0.30992504138547006}, {-0.7755982417649826, 0.9390808625728915, 2.0862510744685485,
-1.1229484266101883}, {-1.2357090352280826, 0.8583930286034466, 0.724702079881947,
0.9762852709700459}};
/*********************************************************************
  End Network Configuration
 *********************************************************************/

void stop() {
  digitalWrite(ENA, LOW); //Desactivamos los motores
  digitalWrite(ENB, LOW); //Desactivamos los motores
  Serial.println("Stop!");
}

//Medir distancia en Centimetros
int Distance_test() {
  digitalWrite(Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(20);
  digitalWrite(Trig, LOW);
  float Fdistance = pulseIn(Echo, HIGH);
  Fdistance= Fdistance / 58;
  return (int)Fdistance;
}

void setup() {
  myservo.attach(3);  // attach servo on pin 3 to servo object
  Serial.begin(9600);
  pinMode(Echo, INPUT);
  pinMode(Trig, OUTPUT);
```

```
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  stop();
  myservo.write(90);  //posicion inicial en el centro
  delay(500);
}

unsigned long previousMillis = 0;   // para medir ciclos de tiempo
const long interval = 25;           // intervalos cada x milisegundos
int grados_servo = 90;              // posicion del servo que mueve el sensor ultrasonico
bool clockwise = true;              // sentido de giro del servo
const long ANGULO_MIN = 30;
const long ANGULO_MAX = 150;
double ditanciaMaxima = 50.0;       // distancia de lejania desde la que empieza a actuar la NN
int incrementos = 9;                // incrementos por ciclo de posicion del servo
int accionEnCurso = 1;              // cantidad de ciclos ejecutando una accion
int multiplicador = 1000/interval;  // multiplica la cant de ciclos para dar tiempo a que el coche pueda girar
const int SPEED = 100;              // velocidad del coche de las 4 ruedas a la vez.

void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

  /*****************************************************************
    MANEJAR GIRO de SERVO
  *****************************************************************/
    if(grados_servo<=ANGULO_MIN || grados_servo>=ANGULO_MAX){
      clockwise=!clockwise; // cambio de sentido
      grados_servo = constrain(grados_servo, ANGULO_MIN, ANGULO_MAX);
    }
    if(clockwise)
      grados_servo=grados_servo+incrementos;
    else
      grados_servo=grados_servo-incrementos;

    if(accionEnCurso>0){
      accionEnCurso=accionEnCurso-1;
    }else{
  /*****************************************************************
    LLAMAMOS a la FUNCION DE CONDUCCION
  *****************************************************************/
       conducir();
    }
    myservo.write(grados_servo);
  }

}

//USA LA RED NEURONAL YA ENTRENADA
void conducir()
{
    double TestInput[] = {0, 0,0};
    double entrada1=0,entrada2=0;

  /*****************************************************************
    OBTENER DISTANCIA DEL SENSOR
  *****************************************************************/
    double distance = double(Distance_test());
    distance= double(constrain(distance, 0.0, ditanciaMaxima));
```

```
    entrada1= ((-2.0/ditanciaMaxima)*double(distance))+1.0; //uso una funcion lineal para obtener cercania
    accionEnCurso = ((entrada1 +1) * multiplicador)+1; // si esta muy cerca del obstaculo, necestia mas
tiempo de reaccion

  /*****************************************************************
    OBTENER DIRECCION SEGUN ANGULO DEL SERVO
  *****************************************************************/
    entrada2 = map(grados_servo, ANGULO_MIN, ANGULO_MAX, -100, 100);
    entrada2 = double(constrain(entrada2, -100.00, 100.00));

  /*****************************************************************
    LLAMAMOS A LA RED FEEDFORWARD CON LAS ENTRADAS
  *****************************************************************/
  Serial.print("Entrada1:");
  Serial.println(entrada1);
  Serial.print("Entrada2:");
  Serial.println(entrada2/100.0);

  TestInput[0] = 1.0;//BIAS UNIT
  TestInput[1] = entrada1;
  TestInput[2] = entrada2/100.0;

  InputToOutput(TestInput[0], TestInput[1], TestInput[2]); //INPUT to ANN to obtain OUTPUT

  int out1 = round(abs(Output[0]));
  int out2 = round(abs(Output[1]));
  int out3 = round(abs(Output[2]));
  int out4 = round(abs(Output[3]));
  Serial.print("Salida1:");
  Serial.println(out1);
  Serial.print("Salida2:");
  Serial.println(out2);
  Serial.println(Output[1]);
  Serial.print("Salida3:");
  Serial.println(out3);
  Serial.print("Salida4:");
  Serial.println(out4);

  /*****************************************************************
    IMPULSAR MOTORES CON LA SALIDA DE LA RED
  *****************************************************************/
  int carSpeed = SPEED; //hacia adelante o atras
  if((out1+out3)==2 || (out2+out4)==2){ // si es giro, necesita doble fuerza los motores
    carSpeed = SPEED * 2;
  }
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, out1 * HIGH);
  digitalWrite(IN2, out2 * HIGH);
  digitalWrite(IN3, out3 * HIGH);
  digitalWrite(IN4, out4 * HIGH);
}

void InputToOutput(double In1, double In2, double In3)
{
  double TestInput[] = {0, 0,0};
  TestInput[0] = In1;
  TestInput[1] = In2;
  TestInput[2] = In3;

  /*****************************************************************
    Calcular las activaciones en las capas ocultas
  *****************************************************************/

  for ( i = 0 ; i < HiddenNodes ; i++ ) {
```

```
    Accum = 0;//HiddenWeights[InputNodes][i] ;
    for ( j = 0 ; j < InputNodes ; j++ ) {
      Accum += TestInput[j] * HiddenWeights[j][i] ;
    }
    //Hidden[i] = 1.0 / (1.0 + exp(-Accum)) ; //Sigmoid
    Hidden[i] = tanh(Accum) ; //tanh
  }

  /******************************************************************
    Calcular activacion y error en la capa de Salida
  ******************************************************************/

  for ( i = 0 ; i < OutputNodes ; i++ ) {
    Accum = 0;//OutputWeights[HiddenNodes][i];
    for ( j = 0 ; j < HiddenNodes ; j++ ) {
        Accum += Hidden[j] * OutputWeights[j][i] ;
    }
    Output[i] = tanh(Accum) ;//tanh
  }

}
```

- **Codi Camera Web Server**

```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! Make sure that you have either selected ESP32 Wrover Module,
//            or another board which has PSRAM enabled
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

const char* ssid = "webcamtoni";
const char* password = "robotica";

void startCameraServer();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
```

```
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  //initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);//flip it back
    s->set_brightness(s, 1);//up the blightness just a bit
    s->set_saturation(s, -2);//lower the saturation
  }
  //drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // put your main code here, to run repeatedly:
```

```
  delay(10000);
}
```

```
// Copyright 2015-2016 Espressif Systems (Shanghai) PTE LTD
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.
#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"

#include "fb_gfx.h"
#include "fd_forward.h"
#include "fr_forward.h"

#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7

#define FACE_COLOR_WHITE  0x00FFFFFF
#define FACE_COLOR_BLACK  0x00000000
#define FACE_COLOR_RED    0x000000FF
#define FACE_COLOR_GREEN  0x0000FF00
#define FACE_COLOR_BLUE   0x00FF0000
#define FACE_COLOR_YELLOW (FACE_COLOR_RED | FACE_COLOR_GREEN)
#define FACE_COLOR_CYAN   (FACE_COLOR_BLUE | FACE_COLOR_GREEN)
#define FACE_COLOR_PURPLE (FACE_COLOR_BLUE | FACE_COLOR_RED)

typedef struct {
        size_t size; //number of values used for filtering
        size_t index; //current value index
        size_t count; //value count
        int sum;
        int * values; //array to be filled with values
} ra_filter_t;

typedef struct {
        httpd_req_t *req;
        size_t len;
} jpg_chunking_t;

#define PART_BOUNDARY "123456789000000000000987654321"
static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";

static ra_filter_t ra_filter;
httpd_handle_t stream_httpd = NULL;
httpd_handle_t camera_httpd = NULL;

static mtmn_config_t mtmn_config = {0};
static int8_t detection_enabled = 0;
```

```
static int8_t recognition_enabled = 0;
static int8_t is_enrolling = 0;
static face_id_list id_list = {0};

static ra_filter_t * ra_filter_init(ra_filter_t * filter, size_t sample_size){
    memset(filter, 0, sizeof(ra_filter_t));

    filter->values = (int *)malloc(sample_size * sizeof(int));
    if(!filter->values){
        return NULL;
    }
    memset(filter->values, 0, sample_size * sizeof(int));

    filter->size = sample_size;
    return filter;
}

static int ra_filter_run(ra_filter_t * filter, int value){
    if(!filter->values){
        return value;
    }
    filter->sum -= filter->values[filter->index];
    filter->values[filter->index] = value;
    filter->sum += filter->values[filter->index];
    filter->index++;
    filter->index = filter->index % filter->size;
    if (filter->count < filter->size) {
        filter->count++;
    }
    return filter->sum / filter->count;
}

static void rgb_print(dl_matrix3du_t *image_matrix, uint32_t color, const char * str){
    fb_data_t fb;
    fb.width = image_matrix->w;
    fb.height = image_matrix->h;
    fb.data = image_matrix->item;
    fb.bytes_per_pixel = 3;
    fb.format = FB_BGR888;
    fb_gfx_print(&fb, (fb.width - (strlen(str) * 14)) / 2, 10, color, str);
}

static int rgb_printf(dl_matrix3du_t *image_matrix, uint32_t color, const char *format, ...){
    char loc_buf[64];
    char * temp = loc_buf;
    int len;
    va_list arg;
    va_list copy;
    va_start(arg, format);
    va_copy(copy, arg);
    len = vsnprintf(loc_buf, sizeof(loc_buf), format, arg);
    va_end(copy);
    if(len >= sizeof(loc_buf)){
        temp = (char*)malloc(len+1);
        if(temp == NULL) {
            return 0;
        }
    }
    vsnprintf(temp, len+1, format, arg);
    va_end(arg);
    rgb_print(image_matrix, color, temp);
    if(len > 64){
        free(temp);
    }
    return len;
```

```
}

static void draw_face_boxes(dl_matrix3du_t *image_matrix, box_array_t *boxes, int face_id){
    int x, y, w, h, i;
    uint32_t color = FACE_COLOR_YELLOW;
    if(face_id < 0){
        color = FACE_COLOR_RED;
    } else if(face_id > 0){
        color = FACE_COLOR_GREEN;
    }
    fb_data_t fb;
    fb.width = image_matrix->w;
    fb.height = image_matrix->h;
    fb.data = image_matrix->item;
    fb.bytes_per_pixel = 3;
    fb.format = FB_BGR888;
    for (i = 0; i < boxes->len; i++){
        // rectangle box
        x = (int)boxes->box[i].box_p[0];
        y = (int)boxes->box[i].box_p[1];
        w = (int)boxes->box[i].box_p[2] - x + 1;
        h = (int)boxes->box[i].box_p[3] - y + 1;
        fb_gfx_drawFastHLine(&fb, x, y, w, color);
        fb_gfx_drawFastHLine(&fb, x, y+h-1, w, color);
        fb_gfx_drawFastVLine(&fb, x, y, h, color);
        fb_gfx_drawFastVLine(&fb, x+w-1, y, h, color);
#if 0
        // landmark
        int x0, y0, j;
        for (j = 0; j < 10; j+=2) {
            x0 = (int)boxes->landmark[i].landmark_p[j];
            y0 = (int)boxes->landmark[i].landmark_p[j+1];
            fb_gfx_fillRect(&fb, x0, y0, 3, 3, color);
        }
#endif
    }
}

static int run_face_recognition(dl_matrix3du_t *image_matrix, box_array_t *net_boxes){
    dl_matrix3du_t *aligned_face = NULL;
    int matched_id = 0;

    aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
    if(!aligned_face){
        Serial.println("Could not allocate face recognition buffer");
        return matched_id;
    }
    if (align_face(net_boxes, image_matrix, aligned_face) == ESP_OK){
        if (is_enrolling == 1){
            int8_t left_sample_face = enroll_face(&id_list, aligned_face);

            if(left_sample_face == (ENROLL_CONFIRM_TIMES - 1)){
                Serial.printf("Enrolling Face ID: %d\n", id_list.tail);
            }
            Serial.printf("Enrolling Face ID: %d sample %d\n", id_list.tail, ENROLL_CONFIRM_TIMES -
left_sample_face);
            rgb_printf(image_matrix, FACE_COLOR_CYAN, "ID[%u] Sample[%u]", id_list.tail,
ENROLL_CONFIRM_TIMES - left_sample_face);
            if (left_sample_face == 0){
                is_enrolling = 0;
                Serial.printf("Enrolled Face ID: %d\n", id_list.tail);
            }
        } else {
            matched_id = recognize_face(&id_list, aligned_face);
            if (matched_id >= 0) {
```

```
                Serial.printf("Match Face ID: %u\n", matched_id);
                rgb_printf(image_matrix, FACE_COLOR_GREEN, "Hello Subject %u", matched_id);
            } else {
                Serial.println("No Match Found");
                rgb_print(image_matrix, FACE_COLOR_RED, "Intruder Alert!");
                matched_id = -1;
            }
        }
    } else {
        Serial.println("Face Not Aligned");
        //rgb_print(image_matrix, FACE_COLOR_YELLOW, "Human Detected");
    }

    dl_matrix3du_free(aligned_face);
    return matched_id;
}

static size_t jpg_encode_stream(void * arg, size_t index, const void* data, size_t len){
    jpg_chunking_t *j = (jpg_chunking_t *)arg;
    if(!index){
        j->len = 0;
    }
    if(httpd_resp_send_chunk(j->req, (const char *)data, len) != ESP_OK){
        return 0;
    }
    j->len += len;
    return len;
}

static esp_err_t capture_handler(httpd_req_t *req){
    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    int64_t fr_start = esp_timer_get_time();

    fb = esp_camera_fb_get();
    if (!fb) {
        Serial.println("Camera capture failed");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    httpd_resp_set_type(req, "image/jpeg");
    httpd_resp_set_hdr(req, "Content-Disposition", "inline; filename=capture.jpg");
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");

    size_t out_len, out_width, out_height;
    uint8_t * out_buf;
    bool s;
    bool detected = false;
    int face_id = 0;
    if(!detection_enabled || fb->width > 400){
        size_t fb_len = 0;
        if(fb->format == PIXFORMAT_JPEG){
            fb_len = fb->len;
            res = httpd_resp_send(req, (const char *)fb->buf, fb->len);
        } else {
            jpg_chunking_t jchunk = {req, 0};
            res = frame2jpg_cb(fb, 80, jpg_encode_stream, &jchunk)?ESP_OK:ESP_FAIL;
            httpd_resp_send_chunk(req, NULL, 0);
            fb_len = jchunk.len;
        }
        esp_camera_fb_return(fb);
        int64_t fr_end = esp_timer_get_time();
        Serial.printf("JPG: %uB %ums\n", (uint32_t)(fb_len), (uint32_t)((fr_end - fr_start)/1000));
        return res;
```

```
    }

    dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);
    if (!image_matrix) {
        esp_camera_fb_return(fb);
        Serial.println("dl_matrix3du_alloc failed");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    out_buf = image_matrix->item;
    out_len = fb->width * fb->height * 3;
    out_width = fb->width;
    out_height = fb->height;

    s = fmt2rgb888(fb->buf, fb->len, fb->format, out_buf);
    esp_camera_fb_return(fb);
    if(!s){
        dl_matrix3du_free(image_matrix);
        Serial.println("to rgb888 failed");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    box_array_t *net_boxes = face_detect(image_matrix, &mtmn_config);

    if (net_boxes){
        detected = true;
        if(recognition_enabled){
            face_id = run_face_recognition(image_matrix, net_boxes);
        }
        draw_face_boxes(image_matrix, net_boxes, face_id);
        free(net_boxes->score);
        free(net_boxes->box);
        free(net_boxes->landmark);
        free(net_boxes);
    }

    jpg_chunking_t jchunk = {req, 0};
    s = fmt2jpg_cb(out_buf, out_len, out_width, out_height, PIXFORMAT_RGB888, 90, jpg_encode_stream,
&jchunk);
    dl_matrix3du_free(image_matrix);
    if(!s){
        Serial.println("JPEG compression failed");
        return ESP_FAIL;
    }

    int64_t fr_end = esp_timer_get_time();
    Serial.printf("FACE: %uB %ums %s%d\n", (uint32_t)(jchunk.len), (uint32_t)((fr_end - fr_start)/1000),
detected?"DETECTED ":"", face_id);
    return res;
}

static esp_err_t stream_handler(httpd_req_t *req){
    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    size_t _jpg_buf_len = 0;
    uint8_t * _jpg_buf = NULL;
    char * part_buf[64];
    dl_matrix3du_t *image_matrix = NULL;
    bool detected = false;
    int face_id = 0;
    int64_t fr_start = 0;
    int64_t fr_ready = 0;
    int64_t fr_face = 0;
```

```
int64_t fr_recognize = 0;
int64_t fr_encode = 0;

static int64_t last_frame = 0;
if(!last_frame) {
    last_frame = esp_timer_get_time();
}

res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
if(res != ESP_OK){
    return res;
}

httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");

while(true){
    detected = false;
    face_id = 0;
    fb = esp_camera_fb_get();
    if (!fb) {
        Serial.println("Camera capture failed");
        res = ESP_FAIL;
    } else {
        fr_start = esp_timer_get_time();
        fr_ready = fr_start;
        fr_face = fr_start;
        fr_encode = fr_start;
        fr_recognize = fr_start;
        if(!detection_enabled || fb->width > 400){
            if(fb->format != PIXFORMAT_JPEG){
                bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
                esp_camera_fb_return(fb);
                fb = NULL;
                if(!jpeg_converted){
                    Serial.println("JPEG compression failed");
                    res = ESP_FAIL;
                }
            } else {
                _jpg_buf_len = fb->len;
                _jpg_buf = fb->buf;
            }
        } else {

            image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);

            if (!image_matrix) {
                Serial.println("dl_matrix3du_alloc failed");
                res = ESP_FAIL;
            } else {
                if(!fmt2rgb888(fb->buf, fb->len, fb->format, image_matrix->item)){
                    Serial.println("fmt2rgb888 failed");
                    res = ESP_FAIL;
                } else {
                    fr_ready = esp_timer_get_time();
                    box_array_t *net_boxes = NULL;
                    if(detection_enabled){
                        net_boxes = face_detect(image_matrix, &mtmn_config);
                    }
                    fr_face = esp_timer_get_time();
                    fr_recognize = fr_face;
                    if (net_boxes || fb->format != PIXFORMAT_JPEG){
                        if(net_boxes){
                            detected = true;
                            if(recognition_enabled){
                                face_id = run_face_recognition(image_matrix, net_boxes);
```

```
                        }
                        fr_recognize = esp_timer_get_time();
                        draw_face_boxes(image_matrix, net_boxes, face_id);
                        free(net_boxes->score);
                        free(net_boxes->box);
                        free(net_boxes->landmark);
                        free(net_boxes);
                    }
                    if(!fmt2jpg(image_matrix->item, fb->width*fb->height*3, fb->width, fb->height,
PIXFORMAT_RGB888, 90, &_jpg_buf, &_jpg_buf_len)){
                        Serial.println("fmt2jpg failed");
                        res = ESP_FAIL;
                    }
                    esp_camera_fb_return(fb);
                    fb = NULL;
                } else {
                    _jpg_buf = fb->buf;
                    _jpg_buf_len = fb->len;
                }
                fr_encode = esp_timer_get_time();
            }
            dl_matrix3du_free(image_matrix);
        }
    }
    if(res == ESP_OK){
        size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
        res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
        res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){
        res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
    }
    if(fb){
        esp_camera_fb_return(fb);
        fb = NULL;
        _jpg_buf = NULL;
    } else if(_jpg_buf){
        free(_jpg_buf);
        _jpg_buf = NULL;
    }
    if(res != ESP_OK){
        break;
    }
    int64_t fr_end = esp_timer_get_time();

    int64_t ready_time = (fr_ready - fr_start)/1000;
    int64_t face_time = (fr_face - fr_ready)/1000;
    int64_t recognize_time = (fr_recognize - fr_face)/1000;
    int64_t encode_time = (fr_encode - fr_recognize)/1000;
    int64_t process_time = (fr_encode - fr_start)/1000;

    int64_t frame_time = fr_end - last_frame;
    last_frame = fr_end;
    frame_time /= 1000;
    uint32_t avg_frame_time = ra_filter_run(&ra_filter, frame_time);
    Serial.printf("MJPG: %uB %ums (%.1ffps), AVG: %ums (%.1ffps), %u+%u+%u+%u=%u %s%d\n",
        (uint32_t)(_jpg_buf_len),
        (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time,
        avg_frame_time, 1000.0 / avg_frame_time,
        (uint32_t)ready_time, (uint32_t)face_time, (uint32_t)recognize_time, (uint32_t)encode_time,
(uint32_t)process_time,
        (detected)?"DETECTED ":"", face_id
```

```
        );
    }

    last_frame = 0;
    return res;
}

static esp_err_t cmd_handler(httpd_req_t *req){
    char*  buf;
    size_t buf_len;
    char variable[32] = {0,};
    char value[32] = {0,};

    buf_len = httpd_req_get_url_query_len(req) + 1;
    if (buf_len > 1) {
        buf = (char*)malloc(buf_len);
        if(!buf){
            httpd_resp_send_500(req);
            return ESP_FAIL;
        }
        if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
            if (httpd_query_key_value(buf, "var", variable, sizeof(variable)) == ESP_OK &&
                httpd_query_key_value(buf, "val", value, sizeof(value)) == ESP_OK) {
            } else {
                free(buf);
                httpd_resp_send_404(req);
                return ESP_FAIL;
            }
        } else {
            free(buf);
            httpd_resp_send_404(req);
            return ESP_FAIL;
        }
        free(buf);
    } else {
        httpd_resp_send_404(req);
        return ESP_FAIL;
    }

    int val = atoi(value);
    sensor_t * s = esp_camera_sensor_get();
    int res = 0;

    if(!strcmp(variable, "framesize")) {
        if(s->pixformat == PIXFORMAT_JPEG) res = s->set_framesize(s, (framesize_t)val);
    }
    else if(!strcmp(variable, "quality")) res = s->set_quality(s, val);
    else if(!strcmp(variable, "contrast")) res = s->set_contrast(s, val);
    else if(!strcmp(variable, "brightness")) res = s->set_brightness(s, val);
    else if(!strcmp(variable, "saturation")) res = s->set_saturation(s, val);
    else if(!strcmp(variable, "gainceiling")) res = s->set_gainceiling(s, (gainceiling_t)val);
    else if(!strcmp(variable, "colorbar")) res = s->set_colorbar(s, val);
    else if(!strcmp(variable, "awb")) res = s->set_whitebal(s, val);
    else if(!strcmp(variable, "agc")) res = s->set_gain_ctrl(s, val);
    else if(!strcmp(variable, "aec")) res = s->set_exposure_ctrl(s, val);
    else if(!strcmp(variable, "hmirror")) res = s->set_hmirror(s, val);
    else if(!strcmp(variable, "vflip")) res = s->set_vflip(s, val);
    else if(!strcmp(variable, "awb_gain")) res = s->set_awb_gain(s, val);
    else if(!strcmp(variable, "agc_gain")) res = s->set_agc_gain(s, val);
    else if(!strcmp(variable, "aec_value")) res = s->set_aec_value(s, val);
    else if(!strcmp(variable, "aec2")) res = s->set_aec2(s, val);
    else if(!strcmp(variable, "dcw")) res = s->set_dcw(s, val);
    else if(!strcmp(variable, "bpc")) res = s->set_bpc(s, val);
    else if(!strcmp(variable, "wpc")) res = s->set_wpc(s, val);
    else if(!strcmp(variable, "raw_gma")) res = s->set_raw_gma(s, val);
```

```
    else if(!strcmp(variable, "lenc")) res = s->set_lenc(s, val);
    else if(!strcmp(variable, "special_effect")) res = s->set_special_effect(s, val);
    else if(!strcmp(variable, "wb_mode")) res = s->set_wb_mode(s, val);
    else if(!strcmp(variable, "ae_level")) res = s->set_ae_level(s, val);
    else if(!strcmp(variable, "face_detect")) {
        detection_enabled = val;
        if(!detection_enabled) {
            recognition_enabled = 0;
        }
    }
    else if(!strcmp(variable, "face_enroll")) is_enrolling = val;
    else if(!strcmp(variable, "face_recognize")) {
        recognition_enabled = val;
        if(recognition_enabled){
            detection_enabled = val;
        }
    }
    else {
        res = -1;
    }

    if(res){
        return httpd_resp_send_500(req);
    }

    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, NULL, 0);
}

static esp_err_t status_handler(httpd_req_t *req){
    static char json_response[1024];

    sensor_t * s = esp_camera_sensor_get();
    char * p = json_response;
    *p++ = '{';

    p+=sprintf(p, "\"framesize\":%u,", s->status.framesize);
    p+=sprintf(p, "\"quality\":%u,", s->status.quality);
    p+=sprintf(p, "\"brightness\":%d,", s->status.brightness);
    p+=sprintf(p, "\"contrast\":%d,", s->status.contrast);
    p+=sprintf(p, "\"saturation\":%d,", s->status.saturation);
    p+=sprintf(p, "\"sharpness\":%d,", s->status.sharpness);
    p+=sprintf(p, "\"special_effect\":%u,", s->status.special_effect);
    p+=sprintf(p, "\"wb_mode\":%u,", s->status.wb_mode);
    p+=sprintf(p, "\"awb\":%u,", s->status.awb);
    p+=sprintf(p, "\"awb_gain\":%u,", s->status.awb_gain);
    p+=sprintf(p, "\"aec\":%u,", s->status.aec);
    p+=sprintf(p, "\"aec2\":%u,", s->status.aec2);
    p+=sprintf(p, "\"ae_level\":%d,", s->status.ae_level);
    p+=sprintf(p, "\"aec_value\":%u,", s->status.aec_value);
    p+=sprintf(p, "\"agc\":%u,", s->status.agc);
    p+=sprintf(p, "\"agc_gain\":%u,", s->status.agc_gain);
    p+=sprintf(p, "\"gainceiling\":%u,", s->status.gainceiling);
    p+=sprintf(p, "\"bpc\":%u,", s->status.bpc);
    p+=sprintf(p, "\"wpc\":%u,", s->status.wpc);
    p+=sprintf(p, "\"raw_gma\":%u,", s->status.raw_gma);
    p+=sprintf(p, "\"lenc\":%u,", s->status.lenc);
    p+=sprintf(p, "\"vflip\":%u,", s->status.vflip);
    p+=sprintf(p, "\"hmirror\":%u,", s->status.hmirror);
    p+=sprintf(p, "\"dcw\":%u,", s->status.dcw);
    p+=sprintf(p, "\"colorbar\":%u,", s->status.colorbar);
    p+=sprintf(p, "\"face_detect\":%u,", detection_enabled);
    p+=sprintf(p, "\"face_enroll\":%u,", is_enrolling);
    p+=sprintf(p, "\"face_recognize\":%u", recognition_enabled);
    *p++ = '}';
```

```
    *p++ = 0;
    httpd_resp_set_type(req, "application/json");
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, json_response, strlen(json_response));
}

static esp_err_t index_handler(httpd_req_t *req){
    httpd_resp_set_type(req, "text/html");
    httpd_resp_set_hdr(req, "Content-Encoding", "gzip");
    sensor_t * s = esp_camera_sensor_get();
    if (s->id.PID == OV3660_PID) {
        return httpd_resp_send(req, (const char *)index_ov3660_html_gz, index_ov3660_html_gz_len);
    }
    return httpd_resp_send(req, (const char *)index_ov2640_html_gz, index_ov2640_html_gz_len);
}

void startCameraServer(){
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();

    httpd_uri_t index_uri = {
        .uri       = "/",
        .method    = HTTP_GET,
        .handler   = index_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t status_uri = {
        .uri       = "/status",
        .method    = HTTP_GET,
        .handler   = status_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t cmd_uri = {
        .uri       = "/control",
        .method    = HTTP_GET,
        .handler   = cmd_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t capture_uri = {
        .uri       = "/capture",
        .method    = HTTP_GET,
        .handler   = capture_handler,
        .user_ctx  = NULL
    };

   httpd_uri_t stream_uri = {
        .uri       = "/stream",
        .method    = HTTP_GET,
        .handler   = stream_handler,
        .user_ctx  = NULL
    };


    ra_filter_init(&ra_filter, 20);

    mtmn_config.type = FAST;
    mtmn_config.min_face = 80;
    mtmn_config.pyramid = 0.707;
    mtmn_config.pyramid_times = 4;
    mtmn_config.p_threshold.score = 0.6;
    mtmn_config.p_threshold.nms = 0.7;
    mtmn_config.p_threshold.candidate_number = 20;
    mtmn_config.r_threshold.score = 0.7;
```

```
    mtmn_config.r_threshold.nms = 0.7;
    mtmn_config.r_threshold.candidate_number = 10;
    mtmn_config.o_threshold.score = 0.7;
    mtmn_config.o_threshold.nms = 0.7;
    mtmn_config.o_threshold.candidate_number = 1;

    face_id_init(&id_list, FACE_ID_SAVE_NUMBER, ENROLL_CONFIRM_TIMES);

    Serial.printf("Starting web server on port: '%d'\n", config.server_port);
    if (httpd_start(&camera_httpd, &config) == ESP_OK) {
        httpd_register_uri_handler(camera_httpd, &index_uri);
        httpd_register_uri_handler(camera_httpd, &cmd_uri);
        httpd_register_uri_handler(camera_httpd, &status_uri);
        httpd_register_uri_handler(camera_httpd, &capture_uri);
    }

    config.server_port += 1;
    config.ctrl_port += 1;
    Serial.printf("Starting stream server on port: '%d'\n", config.server_port);
    if (httpd_start(&stream_httpd, &config) == ESP_OK) {
        httpd_register_uri_handler(stream_httpd, &stream_uri);
    }
}
```

5.2. Bibliografia

- Dot CSV: Canal de youtube sobre d'intel·ligència artificial.
  https://www.youtube.com/channel/UCy5znSnfMsDwaLlROnZ7Qbg

- Orts, Jordi (2019). *IoT amb D1 mini (ESP8266) i codi Arduino.*
  https://github.com/jorts64

- RoboIndia.com: tutorial per utilitzar i controlar els motors amb NodeMCU.
  https://roboindia.com/tutorials/nodemcu-motor-driver-pwm/

- RandomNerdTutorials: pàgina web amb explicacions i exemples sobre l'ESP32, ús de la càmera, problemes i com solucionar-los...
  https://randomnerdtutorials.com/
    - Tutorial càmera amb reconeixement facial
      https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/
    - Solucionar errors ESP32
      https://randomnerdtutorials.com/esp32-cam-troubleshooting-guide/
    - Guardar fotos a la sd
      https://randomnerdtutorials.com/esp32-cam-take-photo-save-microsd-card/

- AprendeMachineLearning: web amb articles sobre python, xarxes neuronals (teoria i exemples pràctics)...
  https://www.aprendemachinelearning.com/
    - Creació d'una xarxa neuronal per a un cotxe amb arduino
      https://www.aprendemachinelearning.com/crear-una-red-neuronal-en-python-desde-cero/
    - Programar un cotxe arduino amb una xarxa neuronal
      https://www.aprendemachinelearning.com/programa-un-coche-arduino-con-inteligencia-artificial/

- Viquipèdia:
    - Intel·ligència artificial
      https://en.wikipedia.org/wiki/Artificial_intelligence
    - Machine Learning
      https://en.wikipedia.org/wiki/Machine_learning
    - Natural Language Processing
      https://en.wikipedia.org/wiki/Natural_language_processing

5.3. Adreces d'interès

[1] Deep Blue vs Garri Kaspàrov. Article de Viquipèdia.
https://ca.wikipedia.org/wiki/Deep_Blue_contra_Garri_Kasp%C3%A0rov

[2] Pàgina oficial de Boston Dynamics. Spot, el "gos".
https://www.bostondynamics.com/spot

[3] Article sobre l'algoritme de N.L.P. GPT-2
https://www.theverge.com/2019/11/7/20953040/openai-text-generation-ai-gpt-2-full-model-release-1-5b-parameters

[4] Article sobre IA i videojocs d'Atari
https://news.developer.nvidia.com/artificial-intelligence-can-beat-humans-at-31-atari-games/

[5] Informació sobre Backpropagation i la seva història
http://neuralnetworksanddeeplearning.com/chap2.html