TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Enginyeria Tècnica Industrial: Especialitat Electrònica Industrial

IMPLEMENTACIÓ DE RECONEIXEMENT D'IMATGE A UN ROBOT

Memòria

ARNAU INSERTE POYATOS
PONENT: JOSEP LOPEZ XARBAU

PRIMAVERA 2021

TecnoCampus
Mataró-Maresme

Agraïments

A l'Ester, per ser-hi sempre al meu costat i donar-me forces en els moments complicats.

A la meva família, per aguantar-me tots aquests anys i donar-me suport.

Resum

El present projecte tracta del desenvolupament d'una aplicació de reconeixement d'imatge mitjançant l'ús d'un model de xarxa neuronal entrenat fent servir tècniques de Transfer Learning. El model ha de ser capaç de detectar diferents tipus de peces per tal de fer una posterior classificació mitjançant l'ús d'un robot industrial.

La seva realització consisteix en el disseny i construcció d'una estructura que funcionarà com a sistema d'il·luminació i suport per a l'electrònica necessària, la programació del microcontrolador encarregat de dur a terme el reconeixement, la programació del robot i la programació d'un PLC que controla una cinta transportadora.

Resumen

El presente proyecto trata el desarrollo de una aplicación de reconocimiento de imagen mediante el uso de un modelo de red neuronal entrenado utilizando técnicas de Transfer Learning. El modelo debe ser capaz de detectar diferentes tipos de piezas para hacer una posterior clasificación mediante el uso de un robot industrial.

Su realización consiste en el diseño y construcción de una estructura que funcionará como sistema de iluminación y soporte para la electrónica necesaria, la programación del microcontrolador encargado de llevar a cabo el reconocimiento, la programación del robot y la programación de un PLC que controla una cinta transportadora.

Abstract

This project deals with the development of an image recognition application through the use of a trained neural network model using Transfer Learning techniques. The model must be able to detect different types of parts to make a subsequent classification through the use of an industrial robot.

Its realization consists of the design and construction of a structure that will function as a lighting system and support for the necessary electronics, the programming of the microcontroller in charge of carrying out the recognition, the programming of the robot and the programming of a PLC that controls a conveyor belt.

Índex.

Índex de figures. V
Índex de taules. VII
Glossari de termes. IX
1. Objectius 1
1.1. Propòsit. 1
1.2. Finalitat. 1
1.3. Objecte. 1
1.4. Abast. 1
2. Perspectiva de gènere. 3
3. Antecedents i necessitats d'informació 5
3.1. Robòtica. 5
3.1.1. Definició de robòtica. 5
3.1.2. Lleis de la robòtica. 5
3.1.3. Camps d'aplicació. 6
3.2. Robòtica Industrial. 7
3.2.1. Elements d'un robot. 8
3.2.2. Paràmetres dels robots. 12
3.3. Visió artificial. 12
3.4. Normativa. 14
4. Objectius i especificacions tècniques. 15
5. Sistema d'il·luminació. 17
5.1. Digitalització. 17
5.1.1. Exemple pràctic. 19
5.2. Disseny del sistema d'il·luminació. 20
6. Reconeixement d'imatge. 23
6.1 Intel·ligència artificial. 23

6.2. Machine Learning. 24
6.3. Xarxes neuronals i Deep Learning. 25
6.4. Xarxes neuronal convolucionals (CNN). 26
6.4.1 Operació de convolució. 27
6.4.2 Operació de padding. 29
6.4.3 Operació de pooling. 30
6.4.4 Operació de convolució per a imatges RGB 30
6.4.5 Funcions d´activació. 31
6.4.6 Arquitectura general d'una CNN. 33
6.5. Transfer Learning. 34
7. Implementació. 37
7.1. Hardware. 37
7.1.1. Placa de desenvolupament Jetson Nano. 37
7.1.2. Robot ABB IBR 120. 37
7.1.3. PLC Rockwell CompactLogix. 38
7.2. Preparació de l'entorn de programació. 38
7.3. Creació de la base de dades. 41
7.4. Entrenament de la xarxa neuronal. 45
7.5. Programació. 47
7.6. Algorisme del PLC. 51
7.7. Algorisme del robot. 53
7.8. Resultats. 55
8. Planificació. 59
8.1. Tasques i prelacions. 59
8.2. Descripció de les tasques. 60
8.3. Diagrama de Gantt. 63
9. Impacte mediambiental. 65

10. Conclusions finals, dificultats i futures línies de treball. 67
10.1. Dificultats. 67
10.2. Futures línies de treball. 69
10.3. Conclusions. 69
11. Referències. 71

Índex de figures.

Figura 3.1: Instal·lacions de robots al 2019. 7
Figura 3.2: Robot cartesià. 9
Figura 3.3: Robot cilíndric. 9
Figura 3.4: Robot SCARA. 10
Figura 3.5: Robot angular. 10
Figura 5.1: Composició d'un píxel d'un sensor monocromàtic. 17
Figura 5.2: Sensor CMOS. 18
Figura 5.3: Matriu de Bayer i píxels cromàtics. 18
Figura 5.4: Imatge original (esquerra) i imatge amb resolució reduïda(dreta). 19
Figura 5.5: Matriu B (blau) de la icona. 19
Figura 5.6: Estructura del sistema d'il·luminació. 20
Figura 5.7: Càmera i LEDs del sistema d'il·luminació. 21
Figura 6.1: Estructura d'una xarxa neuronal. 26
Figura 6.2: Flattening d'una imatge 2x2. 27
Figura 6.3: Operació de convolució. 28
Figura 6.4: Operació de padding. 29
Figura 6.5: Operació de MAX Pooling 2x2 amb un desplaçament de 2. 30
Figura 6.6: Funció d'activació sigmoide. 32
Figura 6.7: Funció d'activació tangencial hiperbòlica. 32
Figura 6.8: Funció d'activació ReLU. 33
Figura 6.9: Arquitectura d'una CNN. 33
Figura 7.1: Terminal. 39
Figura 7.2: Instal·lador de Jetson Inference. 41
Figura 7.3: Arxiu labels.txt. 42
Figura 7.4: Aplicació Camera-capture. 43
Figura 7.5: Etiquetatge utilitzant Camera-capture. 44
Figura 7.6: Etiquetatge d'una classe. 44
Figura 7.7: Instrucció d'entrenament. 45
Figura 7.8: Carpeta del nou model. 46
Figura 7.9: Importació de les llibreries. 47
Figura 7.10: Importació del model entrenat. 47

Figura 7.11: Configuració de la detecció. 48
Figura 7.12: Condicionament de sortides. 49
Figura 7.13: Comprovació de detecció 50
Figura 7.14: Grafcet del PLC. 51
Figura 7.15: Algorisme del robot. 53
Figura 7.16: Carro amb la fusta de subjecció de la peça. 55
Figura 7.17: Detecció d'una peça blava 56
Figura 7.18: Aproximació del robot a la peça. 56
Figura 7.19: Resultat final de la classificació. 57
Figura 8.1: Diagrama de Gantt. 63

Índex de taules.

Taula 7.1: Sortides associades a la classe..... 49
Taula 9.1: Resum de les taules d'impacte mediambiental..... 65
Taula 9.2: Resum de les taules d'impacte mediambiental..... 65

Glossari de termes.

AGI	Artificial General Intelligence
AGV	Automated Guided Vehicle
ANI	Artificial Narrow Intelligence
API	Application Programming Interface
AQU	Agència per a la Qualitat del Sistema Universitari de Catalunya
ASI	Artificial Super Intelligence
CMOS	Complementary metal-oxide semiconductor
CPU	Central Processing Unit
CSI	Camera Serial Interface
FPS	Frames Per Second
GPIO	General Purpose Input/Output
GPU	Graphics Processing Unit
IA	Intel·ligència artificial
RAM	Random Access Memory
TCP/IP	Transmission Control Protocol/Internet Protocol

1. Objectius.

1.1. Propòsit.

Proporcionar a un robot la capacitat de reconèixer objectes mitjançant visió artificial.

1.2. Finalitat.

Incrementar les prestacions d'un robot per proporcionar una eina d'ensenyament a la universitat amb una tecnologia enfocada a la indústria 4.0.

1.3. Objecte.

Disseny i construcció d'un sistema de visió artificial el qual permetrà a un robot tenir la capacitat de reconeixement d'imatge. La solució desenvolupada implica una millora de les característiques del robot amb un cost inferior al que suposaria l'adquisició d'un equip nou amb reconeixement d'imatge integrat.

1.4. Abast.

Aquest projecte inclou la programació d'una xarxa neuronal, del software del sistema de visió artificial i del robot antropomòrfic disponible a la universitat Tecnocampus per tal de dur a terme una aplicació de classificació. També es realitzarà la comunicació entre el microcontrolador, el robot i el sistema de visió.

Es dissenyarà i es construirà un sistema d'il·luminació que permeti tenir una intensitat lumínica òptima en el moment de la captura d'imatges per tal de garantir un bon funcionament en qualsevol condició de treball.

El sistema de reconeixement d'imatge serà capaç d'identificar diferents estils de peces. Posteriorment, el robot haurà de fer una classificació segons el tipus de peça detectada. Referent a la comunicació, el microcontrolador anirà connectat a través d'un cable Ethernet al sistema de control del robot. La transferència de dades es durà a terme mitjançant el protocol de comunicació TCP/IP.

2. Perspectiva de gènere.

El present projecte s'ha dut a terme tenint en compte el marc general per a la incorporació de la perspectiva de gènere en la docència universitària de l'Agència per a la Qualitat del Sistema Universitari de Catalunya (AQU).

Degut a la naturalesa del projecte, s'ha considerat neutral davant la perspectiva de gènere, sent igualment vàlid per a ser utilitzat tant per homes com per a dones.

En els camps d'automatització, és molt freqüent que ambdós gèneres siguin afectats per la mateixa dificultat d'ús, sent la major part de l'estudi programació. Referent a l'assemblatge, els components utilitzats són de mida reduïda i per tant no requereixen unes aptituds físiques desenvolupades.

En el present treball es vol garantir que cap pràctica resultant d'utilitzar aquest projecte tingui efectes diferenciats entre homes i dones, donant importància a no diferenciació entre gèneres.

3. Antecedents i necessitats d'informació.

3.1. Robòtica.

3.1.1. Definició de robòtica.

Abans d'entrar en profunditat, s'ha de tenir clar què és un robot i què s'entén per robòtica.

Segons l'Institut d'Estudis Catalans, un robot és una màquina que pot realitzar automàticament una sèrie de tasques que normalment duen a terme persones [1]. Realment, no existeix una única definició de robot, per exemple segons Definicion.de, un robot és una màquina programable que pot manipular objectes i pot fer operacions que abans només podien ser realitzades per humans [2].

En aquest treball es prendrà com a definició de robot: màquina programable que permet dur a terme tasques de manera autònoma que normalment serien realitzades per humans.

Així doncs, la robòtica és la ciència que estudia el disseny i el desenvolupament de robots.

La primera persona a utilitzar el terme "robot" va ser el txec Karel Čapek, a la seva obra R.U.R (Robots Universales Rossum) el 1920, però no va ser ell l'inventor de la paraula sinó que va ser el seu germà Josef. La idea inicial de Karel era utilitzar la paraula *laboři* (del llatí *labor*, treball) però finalment el seu germà li va proposar *roboti*, provinent de la paraula *robota* i que significa "treballar durament" [3].

3.1.2. Lleis de la robòtica.

L'any 1942, Isaac Asimov va establir les anomenades lleis de la robòtica en la seva obra *Jo, robot*, que tenen com a objectiu protegir la humanitat. Segons Asimov, les tres lleis que s'han de complir obligatòriament són:

1. Un robot no pot fer mal a cap ésser humà ni pot permetre que un ésser humà prengui mal per la seva inacció.
2. Un robot ha d'obeir les ordres dels éssers humans, excepte si entren en conflicte amb la primera llei.

3. Un robot ha de protegir la seva existència sempre que no entri en conflicte amb la primera i la segona llei.

Tot i que aquestes lleis són pensades des del punt de vista de la ciència-ficció, cada cop existeixen màquines amb un grau d'intel·ligència superior, i com es veurà en futurs apartats, la intel·ligència artificial (IA) avança de manera notable fins al punt que existeixen IA capaces de pensar de manera autònoma, i per tant, en un futur no molt llunyà s'hauran de tenir en compte.

3.1.3. Camps d'aplicació.

En el camp de la robòtica existeixen moltes àrees d'aplicació. La principal és la industrial, però hi ha camps com el de la medicina o el de serveis que guanyen importància a mesura que la tecnologia avança.

Robòtica mèdica.

En la robòtica destinada al sector de la medicina es poden diferenciar dos grans grups: robòtica destinada als pacients i robòtica per a l'assistència als metges.

Dins de la robòtica destinada als pacients es pot fer una divisió segons assistència permanent o de rehabilitació. Per assistència permanent s'entén pròtesis, robots d'assistència o electroestimulació. En canvi, els robots de rehabilitació són utilitzats per afavorir la recuperació dels pacients.

Pel que fa a robots destinats a l'assistència dels metges, existeixen els robots quirúrgics. Aquest tipus de robots són empleats per a operacions on és necessària una precisió molt elevada [4].

Robòtica de servei.

Per robot de servei s'entén un robot que ajuda a les persones mitjançant la realització de treballs repetitius, perillosos, bruts així com poden ser les tasques de la llar. En aquesta àrea trobem robots aspiradors o de manteniment de piscines. També podem trobar robots de servei en el sector de la logística com per exemple els robots tipus AGV (Automated guided vehicle), vehicles capaços de transportar càrrega de manera autònoma [5][6].

Robòtica industrial.

Un robot industrial és una màquina electrònica, multifuncional, reprogramable, capaç de moure càrregues, matèries, peces, eines o dispositius especials, a través de moviments i trajectòries variables, programades per acomplir diferents tasques [7].

Aquest projecte té com a objectiu centrar-se a desenvolupar una aplicació industrial, així doncs el robot a utilitzar estarà dins de l'àrea de la robòtica industrial.

3.2. Robòtica Industrial.

Cada cop són més els robots instal·lats a les fàbriques. A Espanya hi ha uns 17 robots per cada mil treballadors [8]. És una dada significativa per veure la importància que tenen els robots dins de la indústria.

Segons IFR, només l'any 2019, Xina va instal·lar 140.500 robots nous mentre que Espanya va instal·lar 3.800.

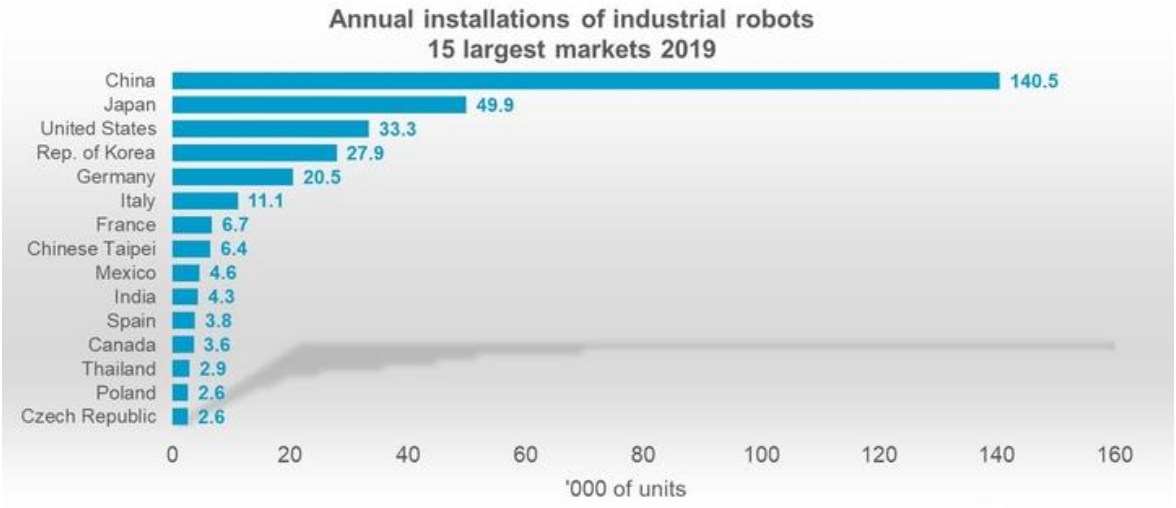

Figura 3.1: Instal·lacions de robots al 2019.
Font: ifr.org

Tal com hem dit a l'apartat anterior, els robots industrials són màquines que repliquen moviments programats. Es poden utilitzar per a minimitzar el perill per a un ésser humà (per exemple en un espai nociu), proporcionar més força o precisió, o quan es vol tenir una producció continuada.

Moltes vegades es té una imatge que els robots són més ràpids que els humans a l'hora de produir, però la realitat és que acostumen a ser més lents. L'avantatge que tenen els robots és la capacitat de repetibilitat i la gran precisió durant llargs períodes de temps. Com a resultat s'obtenen productes d'una gran qualitat. La capacitat de poder produir de manera continuada sense necessitat de descans també augmenta de manera significativa la productivitat [9].

3.2.1. Elements d'un robot.

Un robot està format per diferents elements que es poden modificar segons les necessitats de cada aplicació. Aquests elements són:

1. Estructura mecànica
2. Sensors
3. Sistema de control
4. Actuadors
5. Element terminal

3.2.1.1. Estructura mecànica.

Existeixen diversos tipus d'estructures mecàniques per a un robot. El tipus de configuració determinarà quins tipus de moviments el robot serà capaç de fer i la facilitat per fer-los. A continuació s'exposen els quatre tipus d'arquitectura més clàssics.

Robot cartesià.

Aquest tipus de robot té tres articulacions de tipus lineal corresponents als eixos X, Y i Z amb la qual cosa proporciona un espai de treball en forma de cub. Gràcies a treballar amb tres eixos de moviment lineal es pot utilitzar un sistema de coordenades cartesianes, fent que aquest tipus de configuració sigui la més senzilla de controlar. Actualment el seu us està molt estès gràcies a la seva facilitat de programació i en la seva senzillesa en la construcció. És molt utilitzat en aplicacions d'assemblatge, màquines-eina, soldadura d'arc o por a desplaçar càrrega [10].

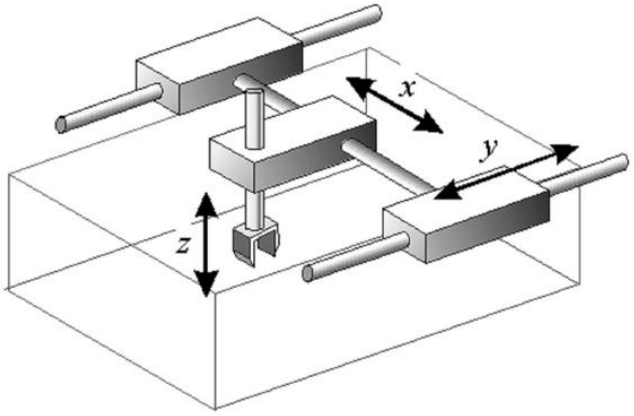

Figura 3.2: Robot cartesià.
Font: http://www.udesantiagovirtual.cl

Robot cilíndric.

En aquesta configuració tenim que la primera articulació és del tipus rotacional, el que produeix un moviment angular respecte a la base. La segona i tercera articulació són prismàtiques amb el qual es guanya alçada i profunditat. El robot cilíndric utilitza coordenades angulars, és a dir, angle β, alçada Y i radi Z. És utilitzat en màquines-eina, soldadura per punt i assemblatge [10].

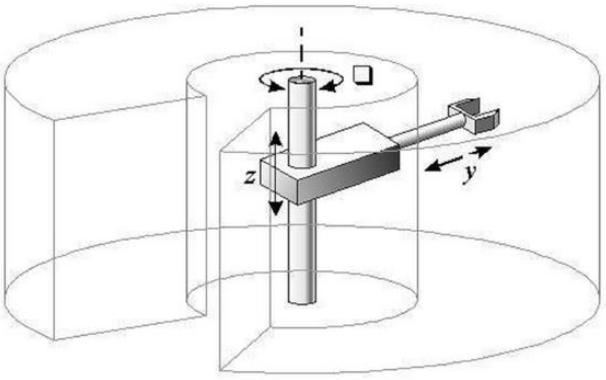

Figura 3.3: Robot cilíndric.
Font: http://www.udesantiagovirtual.cl

Robot SCARA.

El robot SCARA és característic per tenir les dues primeres articulacions angulars i la tercera lineal. Gràcies a la seva configuració és molt ràpid i precís a part de ser econòmic. L'inconvenient és que només té accés a zones de treball perpendiculars al seu eix vertical.

És molt utilitzat en processos d'assemblatge de components elèctrics i en treballs similars [11].

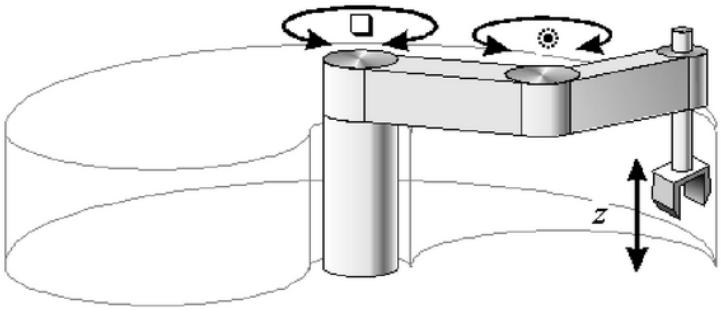

Figura 3.4: Robot SCARA.
Font: http://www.udesantiagovirtual.cl

Robot angular o antropomòrfic.

El robot angular rep el nom d'antropomòrfic per la seva semblança amb un braç humà. Totes les seves articulacions són de tipus angular, el que permet al robot poder accedir a una àmplia zona al voltant d'ell. És un robot molt polivalent i popular dins de la indústria, utilitzat en operacions de soldadura, pintura i assemblatge [10].

Figura 3.5: Robot angular.
Font: http://platea.pntic.mec.es

3.2.1.2. Sensors.

Els sensors són els encarregats de recollir la informació de l'entorn i enviar-la al sistema de control per al seu processament.

Els sensors es poden classificar en dos grups:

- Sensors externs: serveixen per adquirir dades de l'entorn del robot com pot ser el nivell d'il·luminació, la temperatura o en el nostre cas, la detecció d'objectes.
- Sensors interns: són els encarregats de controlar el funcionament del robot com per exemple la posició de les articulacions, mitjançant encoder, la velocitat o el parell.

3.2.1.3. Sistema de control.

Els sistemes de control són els encarregats d'analitzar la informació provinent dels sensors, processar aquesta informació, prendre decisions segons la programació i donar ordre als actuadors.

3.2.1.4. Actuadors.

Són els elements encarregats de proporcionar el moviment a les articulacions del robot segons l'ordre rebuda del sistema de control. Es poden classificar en tres grups:

- Neumàtics: Utilitzen aire com a font d'energia. Són actuadors molt ràpids encara que la seva precisió és molt limitada, per la qual cosa acostumen a ser utilitzats en moviments on només es requereixen dues posicions sense necessitat de regulació de velocitat.
- Hidràulics: A diferència dels neumàtics, els actuadors hidràulics poden tenir un control precís de la velocitat i una millor precisió en la posició. Tenen una capacitat elevada de càrrega i una velocitat considerable.
- Elèctrics: Són els actuadors més utilitzats gràcies a la seva precisió i facilitat de control. Són ideals en robots on no és necessària una gran potència ni velocitat, oferint exactitud i repetibilitat [12].

3.2.1.5. Element terminal.

L'element terminal és un mecanisme especial que s'acobla a l'extrem del robot i li permet executar una tasca específica. Per a una correcta elecció de l'element terminal cal fer un estudi previ de les necessitats per a la nostra aplicació. Existeixen infinitat d'elements terminals, però els més importants són:

- Terminals de subjecció a pressió: Formada majoritàriament per pinces que permeten el transport d'elements. És el tipus de terminal més utilitzat per robots manipuladors. Existeixen infinitat de pinces segons les especificacions requerides pel procés i poden ser accionades mitjançant pneumàtica, hidràulica o electricitat.
- Terminal de subjecció per enganxada: Disposa de garres amb elements de retenció específics per a una càrrega en concret per tal de no efectuar pressió sobre ells. S'utilitza per a càrregues on és necessari un control sobre la pressió.
- Terminals de subjecció d'eines: Permeten la subjecció d'eines específiques com poden ser pistoles de pintura o soldadors. Permeten un canvi d'eina de manera senzilla per a processos on és necessari utilitzar més d'un tipus d'eina.
- Terminals de subjecció per contacte: Aquests tipus de terminals estan destinats a la subjecció de materials llisos com pot ser vidre o cartó. El tipus de subjecció pot ser utilitzant ventoses, imants o fins i tot adhesiu [13].

3.2.2. Paràmetres dels robots.

Graus de llibertat.

S'anomena grau de llibertat al nombre de moviments que una articulació permet fer. El nombre total de graus de llibertat que tindrà el robot serà la suma dels graus de llibertat de totes les articulacions del robot.

Cal un mínim de 6 graus de llibertat en un robot per a poder accedir a qualsevol lloc de l'espai i en qualsevol orientació de l'element terminal.

Repetibilitat.

Es tracta de la desviació que pateix un robot en dur a terme una mateixa tasca dues vegades seguides en condicions idèntiques. Tenir una repetibilitat molt baixa significa que el robot serà capaç de reproduir els moviments d'una manera més exacta i per tant es tracta d'una característica desitjable.

3.3. Visió artificial.

En l'actualitat, les solucions més utilitzades per a afegir visió artificial o reconeixement d'imatge a un robot són els sistemes de visió artificial industrials.

Es tracta d'un sistema generalment compost per una càmera i una unitat de processament. Segons la complexitat del sistema es pot classificar en tres tipus de dispositius.

Sensor de visió artificial.

S'anomena sensor de visió als sistemes de reconeixement d'imatge amb prestacions més restringides. Aquests dispositius limiten la seva funció a fer una avaluació de la imatge amb un resultat de correcte o incorrecte, oferint així una única sortida. La baixa potència de càlcul de la que disposen fa que només siguin capaços de fer comparacions amb una imatge establerta de formes o canvis en la il·luminació/color.

Aquests dispositius permeten aplicacions de control de qualitat poc precisa (com per exemple avaluar si una ampolla porta etiqueta o porta tap).

L'avantatge d'aquesta solució és la senzillesa i rapidesa d'implementació a causa de les seves carències a un preu considerablement reduït.

Sistemes de visió integrats.

Es tracta d'una tecnologia més avançada que la dels sensors de visió. Aquests sistemes tenen una major potència de càlcul a més d'una millor qualitat d'imatge. Aquest tipus de dispositius permeten el reconeixement de caràcters, la mesura de distàncies, detecció de formes complexes, entre altres funcions.

Els sistemes de visió integrats acostumen a tenir el sistema de control separat de la càmera, donant flexibilitat a l'hora de ser instal·lat, ja que les dimensions de la càmera es redueixen, ja que no han d'incorporar el sistema de processament.

També permeten una configuració d'entrades i sortides més àmplia a més de tenir un port de comunicacions per a poder ser integrat en sistemes complexos com per exemple estacions robotitzades.

Aquesta solució és utilitzada en processos de control de qualitat automàtica, permetent fer un anàlisi de tots els objectes que es produeixen. També són utilitzats per a poder controlar sistemes automatitzats, podent fer una distinció d'objectes per al seu posterior processat.

Sistemes de visió avançats.

Són els sistemes de visió més sofisticats dins de l'àmbit industrial de la visió artificial. Es tracta d'un sistema semblant als de visió integrats amb la diferència d'un hardware més sofisticats i complet.

Gràcies a una millor potència de càlcul respecte als sistemes integrats, permeten fer les aplicacions més complexes. També ofereixen processament de dades i funcions de software específiques per a cada aplicació.

Els sistemes de visió avançats són solucions altament complexes i difícils d'implementar, per la qual cosa acostumen a ser implementats en la fabricació de la màquina i ser programat a mida per als processos als quals van destinats per tant la seva implementació en processos industrials existents resulta molt costós [14].

3.4. Normativa.

- Directiva 2006/42/CE relativa a les màquines.
- EN ISO 10218-1:2011 Robots i dispositius robòtics. Requeriment de seguretat per a robots industrials. Part 1: Robots.
- EN ISO 10218-2:2011 Robots i dispositius robòtics. Requeriment de seguretat per a robots industrials. Part 2: Sistemes robòtics i integració.
- UNE-EN 61010-1:2011 Requeriments de seguretat d'equips electrònics de mesura, control i ús en laboratoris. Part 1: Requeriments generals.
- EN IEC 61326-1:2019 Material elèctric per a mesura, control i ús en laboratori. Requeriments de compatibilitat electromagnètica. Part 1: Requeriments generals.

4. Objectius i especificacions tècniques.

L'objectiu d'aquest projecte és la implementació de reconeixement d'imatge a un robot existent a l'Escola Superior Politècnica de Mataró. El sistema de reconeixement d'imatge ha de ser capaç de distingir diferents tipus de peces, comunicar-se amb el robot i aquest fer una classificació segons el tipus de peça detectada.

A continuació s'exposen les especificacions tècniques del projecte:

Sistema de reconeixement d'imatge:

- Microcontrolador Jetson nano.
    - Processador Cortex A-57 a 1,43 GHz.
    - Processador gràfic Maxwell de 128 nuclis.
    - Memòria RAM de 4 GB LPDDR4-3200.
    - Port Gigabit Ethernet.
    - Port CSI per a càmera.
- Càmera de visió diürna.
    - Resolució de 3280 x 2464 píxels.
    - Freqüència màxima de 30 fps.
    - Dimensió de 23,86 x 25 x 9mm.
    - Connector CSI.
- Plataforma de desenvolupament de codi obert.
    - Jetson Inference.
    - Python.

Robot:

- Robot ABB IRB 120.
    - 6 graus de llibertat
    - Radi d'operació de 580mm.
    - Repetibilitat de 0,01mm.
- Unitat de control IRC5.
- Programació amb consola de comandament i editor de text.

Sistema d'il·luminació:

- Estructura
    - Material: Fusta aglomerada.
    - Dimensions: 300 x 200 x 200 mm.
- Focus LED.
    - Potència elèctrica: 6W.
    - Potència lumínica: 510 lúmens.
- Espai preparat per a fixar la càmera.

5. Sistema d'il·luminació.

Per tal d'implementar un sistema de reconeixement d'imatge, s'ha d'assegurar que la càmera captura amb la millor qualitat possible i sempre amb la mateixa intensitat de colors. Més endavant, a l'apartat de reconeixement d'imatge, es fa una explicació més profunda del funcionament del processament de dades del sistema de reconeixement d'imatge, però per poder entendre la importància d'incorporar un equip d'il·luminació és necessari conèixer els conceptes bàsics de la digitalització d'imatge.

5.1. Digitalització.

El terme digitalització fa referència al procés de transformar una informació en format analògic a un format digital. Aquest procés resulta essencial per a poder tractar dades en un sistema digital.

En aquest cas, la digitalització es du a terme en el sensor d'imatge (càmera). El dispositiu emprat utilitza un sensor del tipus CMOS (Semiconductor complementari d'òxid metàl·lic).

Els raigs de llum arriben a una petita lent que dirigeix els fotons cap a un fotodíode. En aquest punt es genera un corrent degut a l'efecte fotoelèctric del fotodíode. Aquest conjunt és anomenat píxel [15].

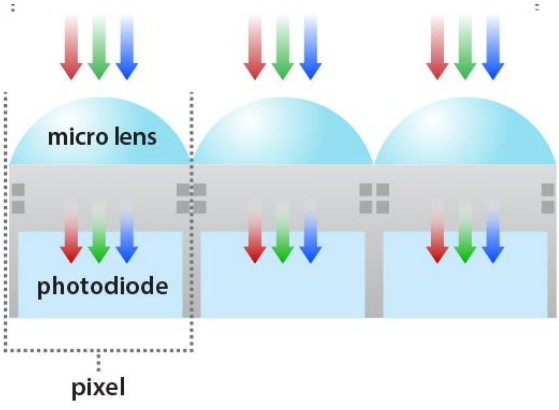

Figura 5.1: Composició d'un píxel d'un sensor monocromàtic.
Font: https://thinklucid.com

El corrent generat per cada píxel és amplificat de manera individual i posteriorment connectat a un convertidor analògic-digital que el tradueix en una codificació digital. D'aquesta manera podem tractar aquest senyal de sortida.

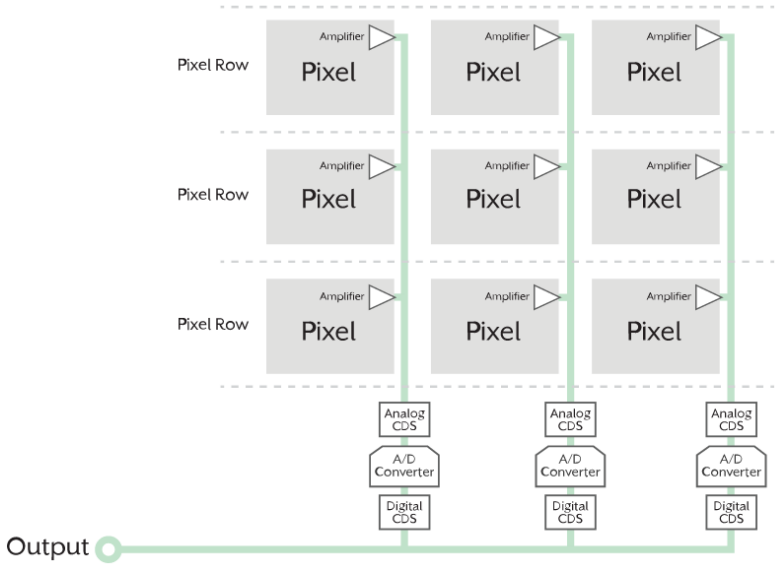

Figura 5.2: Sensor CMOS.
Font: https://thinklucid.com

Per a càmeres cromàtiques s'utilitza el mateix sistema amb la diferència que, en aquest cas, s'afegeix un filtre del mateix color que es vol deixar passar cap al fotodíode. Existeixen diferents patrons per a definir la disposició dels píxels de colors, però la més utilitzada és el patró de Bayer. Com es pot veure a la figura 4.3, aquesta matriu utilitza un 50% de píxels verds, un 25% de píxels blaus i el 25% restant de píxels vermells. D'aquesta manera, podem separar la informació en tres matrius corresponent a la intensitat de color vermell, verd i blau de cada píxel (matrius RGB) [15].

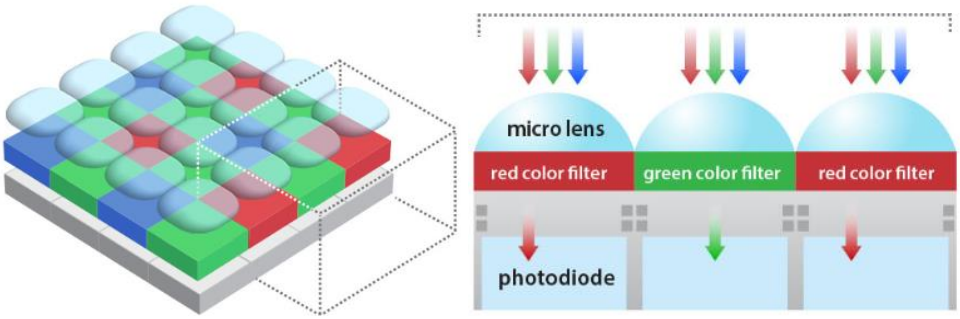

Figura 5.3: Matriu de Bayer i píxels cromàtics.
Font: https://thinklucid.com

5.1.1. Exemple pràctic.

Una forma de veure el funcionament és utilitzant l'eina MATLAB amb una imatge. En aquest cas, s'utilitzarà una imatge amb resolució reduïda de 16x16 píxels d'una icona d'una càmera per tal que les matrius resultants tinguin una mida més continguda.

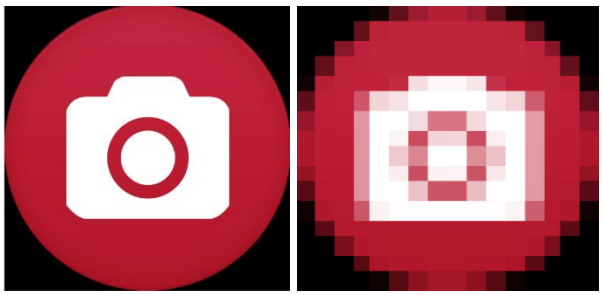

Figura 5.4: Imatge original (esquerra) i imatge amb resolució reduïda(dreta).
Font: Freeiconspng.com.

En el moment d'obrir l'arxiu es generen tres matrius corresponent als tres colors (RGB). Si es pren com a exemple la matriu B (blau) s'obté el següent resultat on es pot veure la silueta de la icona:

0	0	0	0	5	28	46	55	55	46	28	5	0	0	0	0
0	0	0	28	56	60	61	62	62	61	60	56	29	0	0	0
0	0	38	58	61	61	61	61	61	61	61	61	58	38	0	0
0	27	56	60	60	60	60	60	60	60	60	60	60	56	28	0
5	52	58	58	58	65	223	247	247	231	72	58	58	58	52	5
24	54	56	104	215	228	255	255	255	255	231	217	110	56	54	25
39	53	54	160	255	255	223	131	130	220	255	255	166	54	53	39
45	51	51	159	255	245	94	216	218	95	242	255	165	51	51	45
43	50	50	158	255	206	146	255	255	152	199	255	165	50	50	43
36	48	49	158	255	238	96	242	244	98	234	255	164	49	48	36
20	46	47	157	255	255	199	101	100	195	255	255	164	47	46	21
3	43	46	103	242	252	252	252	252	252	252	244	109	46	43	4
0	21	44	45	45	45	45	45	45	45	45	45	45	44	21	0
0	0	27	43	44	44	44	44	44	44	44	44	43	28	0	0
0	0	0	19	41	43	43	43	43	43	43	41	20	0	0	0
0	0	0	0	3	18	31	38	38	31	18	3	0	0	0	0

Figura 5.5: Matriu B (blau) de la icona.
Font: Elaboració pròpia.

Com es pot observar, la matriu consta de 16 columnes i 16 files corresponent als 16x16 píxels de la imatge. Els valors de cada píxel es troben en format de 8 bits, és per això que tenen una valoració de 0 a 255 (2^8 = 256 valors possibles) sent 0 la falta total del color corresponent a la matriu i 255 l'aparició pura del mateix color. Si les tres matrius RGB coincideixen en 0, apareix el color negre, mentre que si coincideixen en 255 indica color blanc.

Aquest procés de quantificació és l'anomenat digitalització d'imatge.

5.2. Disseny del sistema d'il·luminació.

És molt important disposar d'un bon sistema d'il·luminació per tal de disminuir la variació d'incidència de llum entre dues imatges. Tal com s'ha vist a l'apartat anterior, dues imatges d'un mateix objecte amb una il·luminació diferent poden donar matrius RGB amb valors molt diferents entre si. Aquest efecte, tal com s'explica a l'apartat 5, produeix dificultats a l'hora d'utilitzar una xarxa neuronal.

És per aquest motiu que s'ha dissenyat una estructura de fusta pensada per acollir tant el microprocessador com la càmera i alhora permet reduir la incidència de llum de l'ambient. A més, porta incorporada llum LED blanca que subministra la quantitat necessària de llum per tenir una bona qualitat d'imatge.

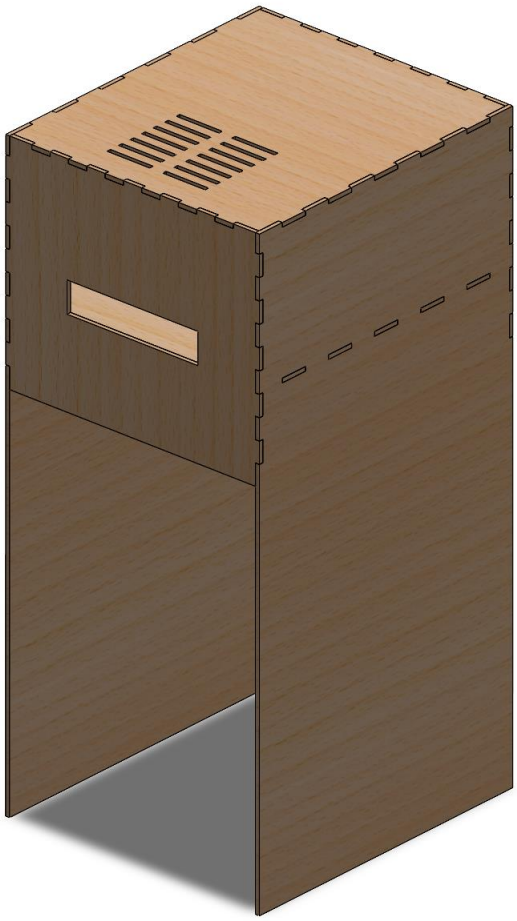

Figura 5.6: Estructura del sistema d'il·luminació.
Font: Elaboració pròpia.

Com es pot veure a la figura anterior, l'electrònica anirà allotjada en el compartiment superior, on disposa d'uns orificis per facilitat la refrigeració dels components. Té una alçada suficient per poder instal·lar un ventilador de suport a la Jetson Nano i d'aquesta manera assegurar el seu correcte funcionament i allargar al vida útil. També disposa d'un petit forat pensat per fixar la càmera.

Per la part inferior de la base del compartiment de l'electrònica, s'ha instal·lat una tira LED que ofereix 510 lúmens. Amb això s'aconsegueix una il·luminació directa sobre la peça, millorant considerablement la qualitat d'imatge i reduint els efectes lumínics exteriors.

Figura 5.7: Càmera i LEDs del sistema d'il·luminació.
Font: Elaboració pròpia.

6. Reconeixement d'imatge.

6.1 Intel·ligència artificial.

Es podria definir Intel·ligència artificial (IA) com la combinació d'algoritmes amb el propòsit de donar a les màquines la capacitat de desenvolupar tasques intel·lectuals normalment realitzades per als humans, com pot ser l'aprenentatge, el raonament i l'autocorrecció [16].

En l'actualitat podem fer una classificació en tres grans tipus d'intel·ligència artificial segons les seves capacitats.

El primer grup es denomina Intel·ligència Artificial Dèbil o en anglès *Artificial Narrow Intelligence* (ANI). Com el seu nom indica, és el tipus d'intel·ligència artificial amb unes característiques més reduïdes, però alhora l'únic tipus d'intel·ligència que s'ha aconseguit desenvolupar. El seu objectiu està enfocat a la realització d'una única tasca la qual millora de manera automàtica.

Exemples d'ANI podrien ser la conducció autònoma, reconeixement facial o fins i tot els assistents com poden ser Alexa o Siri.

El segon grup correspon a les Intel·ligències Artificials Generals, *Artificial General Intelligence* (AGI). Les AGI són un concepte d'intel·ligència (ja que encara no s'ha aconseguit desenvolupar) que replica el comportament intel·lectual de les persones, amb l'habilitat d'aprendre i poder solucionar qualsevol mena de problema. Les AGI poden pensar, comprendre i actuar d'una manera gairebé indistingible de com ho faria una persona en la mateixa situació.

El problema que els investigadors es troben a l'hora desenvolupar les IA radica en la falta de potència de càlcul. El supercomputador Fujitsu-build K, un dels més ràpids del món, va trigar 40 minuts a simular un únic segon d'activitat neuronal humana.

El tercer i últim grup és la Superintel·ligència Artificial, *Artificial Super Intelligence* (ASI).

Es tracta d'una IA hipotètica que no només replica el funcionament intel·lectual humà, sinó que supera la intel·ligència humana. Aquest tipus d'intel·ligència seria capaç de desenvolupar tecnologies o solucionar problemes matemàtics que els humans no som capaços de fer. Existeixen teories on es preveu que les ASI tindrien unes capacitats intel·lectuals tan desenvolupades que podrien tornar-se en contra de les persones. Totes les hipòtesis i teories de les ASI són merament especulatives, ja que en l'actualitat no s'ha aconseguit desenvolupar una IA general i molt menys una Superintel·ligència [17].

6.2. Machine Learning.

Machine Learning es tradueix com «Aprenentatge Automàtic». Segons Jordi Torres, professor de la UPC, es podria definir com el subcamp de la intel·ligència artificial que proporciona als ordinadors la capacitat d'aprendre sense ser explícitament programats, és a dir, sense que el programador indiqui els passos que s'han de seguir per dur a terme la tasca, sinó que ho fa automàticament [16].

Es podria dir que el Machine Learning consisteix a desenvolupar un algoritme de predicció per a cada problema i cas d'ús particular. Aquests algoritmes aprenen de les dades que obtenen amb la finalitat de trobar patrons o tendències i d'aquesta manera poder predir o classificar els elements.

Segons el tipus d'aprenentatge, podem diferenciar tres categories:

Aprenentatge supervisat.

L'aprenentatge supervisat és quan les dades que s'utilitzen per a l'entrenament de la intel·ligència contenen la solució desitjada, anomenada etiqueta. En aquest cas, l'aprenentatge es basa a buscar un model o funció que relacioni una entrada a una sortida. En un escenari òptim i un cop entrenada la intel·ligència, serà capaç de donar una sortida correcta per a valors d'entrada no coneguts anteriorment. A mesura que el model va fent prediccions, aquest va modificant els patrons per tal d'aprendre i millorar la resposta.

Aprenentatge no supervisat.

En aquest cas, només s'indica un conjunt de patrons d'entrada, però mai la sortida desitjada. Utilitzant un sistema de ponderacions, la intel·ligència proporcionarà una sortida. Segons la

sortida obtinguda s'ajustarà la ponderació per obtenir una resposta correcta. A mesura que la intel·ligència ha après, la qualitat de la sortida serà més elevada, ja que l'ajust podrà ser més precís.

Aprenentatge per reforç.

Aquest aprenentatge està dissenyat perquè sigui la mateixa intel·ligència la que determini les relacions i les accions a fer per a aconseguir una sortida favorable mitjançant prova-error. La intel·ligència aprèn gràcies a un sistema de recompenses i penalitzacions que consisteix a indicar si la solució obtinguda és correcta o incorrecta, però sense introduir la solució de manera explícita [16].

6.3. Xarxes neuronals i Deep Learning.

Un exemple d'algorismes de Machine Learning són les anomenades xarxes neuronals artificials. El nom d´aquests algorismes és degut a la semblança de funcionament amb les capes neuronals d'un cervell humà. La seva funció és la d'aprendre els patrons de dades que provenen de diferents fonts com poden ser imatges, so, text, etc.

Una de les característiques més importants de les xarxes neuronals és la capacitat d'abstracció, la qual permet aïllar les propietats d'un objecte per a analitzar-les de manera independent. Amb aquesta característica el que s'aconsegueix és que el sistema tingui un aprenentatge d'una alta complexitat, a vegades difícil de programar de manera explícita.

Una xarxa neuronal consisteix en un conjunt de neurones agrupades en capes connectades entre si. Es poden diferenciar tres tipus diferents de capes: capa d'entrada, amb neurones representant les unitats mesurades; capes ocultes, encarregades de tractar la informació d'entrada mitjançant operacions matemàtiques; capa de sortida, encarregades de donar un o més resultats.

Cada neurona de cada capa es troba connectada amb un conjunt de neurones de la següent capa. La informació que és rebuda a la capa d'entrada es transmet a la primera capa oculta. Aquesta informació és multiplicada per una ponderació depenent de la seva importància. Aquesta ponderació es du a terme a cada connexió entre neurones i d'aquesta manera a la sortida només arriba la informació realment rellevant i a més amb un pes determinat per a poder escollir un o més valors de sortida de manera correcta.

En el moment de crear una xarxa neuronal nova, aquestes ponderacions es defineixen de manera aleatòria, i per tant els valors de sortida seran incorrectes. A mesura que la xarxa dóna valors incorrectes, les ponderacions es van ajustant automàticament. Aquest procés és l'anomenat entrenament de la xarxa i és la manera amb la qual es du a terme l'aprenentatge.

Prenent com a referència l'aprenentatge supervisat vist anteriorment, l'algorisme comparará el valor de sortida amb el valor de l'etiqueta corresponent. Si els dos valors no coincideixen, reajustarà els pesos de les connexions abans de continuar amb la següent iteració.

Perquè una aplicació neuronal rebi el nom de Deep Learning, la seva xarxa ha de tenir un mínim d'una capa oculta [16, 18].

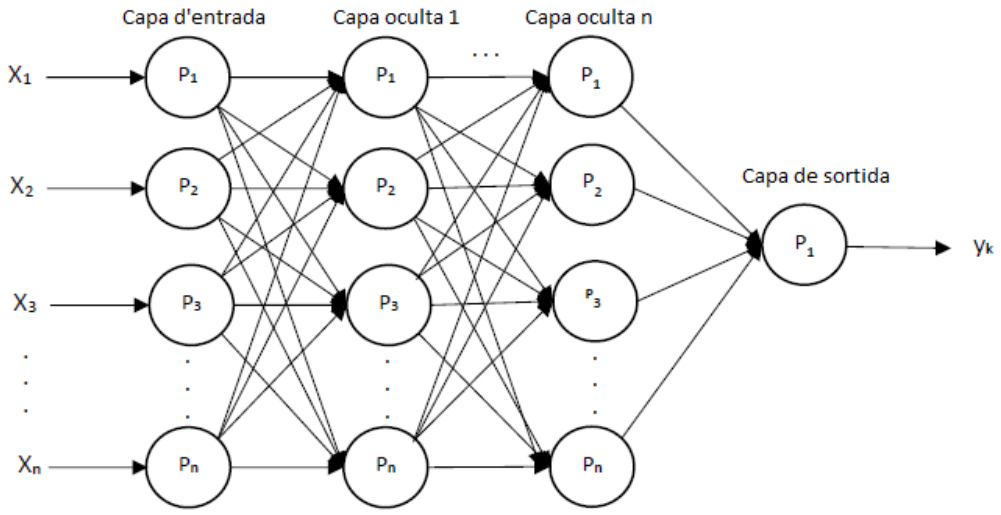

Figura 6.1: Estructura d'una xarxa neuronal.
Font: https://scielo.conicyt.cl

6.4. Xarxes neuronal convolucionals (CNN).

Una xarxa convolucional (Convolutional Neural Network) és un tipus de xarxa molt concret utilitzat en aplicacions de reconeixement d'imatge gràcies a la seva eficàcia en aquest apartat. La característica principal d'aquest tipus de xarxes és que treballen amb matrius bidimensionals. D'aquesta manera es pot treballar molt més eficientment amb les matrius de les imatges vistes a l'apartat 4.1.

Un dels problemes que generen treballar amb imatges és la dimensió que tenen les capes d'entrada. Per a una imatge a color de mida reduïda com pot ser 32x32 píxels, es representarà

en tres matrius de 32x32, és a dir 32x32x3. Això dóna una quantitat de 3072 valors. Per la manera en què treballa una xarxa neuronal, aquests 3072 valors es registren en un únic vector d'entrada. Aquesta operació rep el nom de flattening [16].

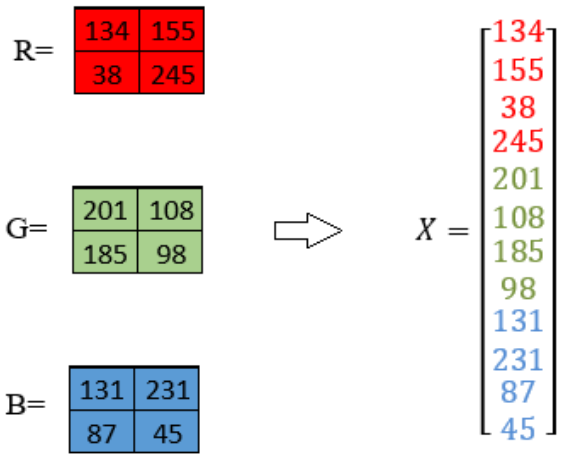

Figura 6.2: Flattening d'una imatge 2x2.
Font: Elaboració pròpia.

Com es pot veure a la imatge anterior, *X* correspon a l'entrada del sistema. Tot i ser una imatge 2x2, obtenim 12 valors a capa d'entrada de la xarxa neuronal. Si fos el cas d'una imatge amb una resolució d'1 Megapíxel (1290x960), el vector d'entrada constaria de gairebé 3,7 milions de valors.

Davant la inviabilitat de processar aquesta quantitat de dades, es du a terme l'operació de convolució, operació que dóna nom a la xarxa neuronal degut a la seva importància.

6.4.1 Operació de convolució.

El propòsit de la convolució és detectar formes o característiques de les imatges com poden ser arestes, línies, zones de colors, etc. Aquesta és una característica molt important, ja que redueix el nombre de valors introduïts a la xarxa considerablement. En comptes de comparar píxel a píxel, es compara les formes i colors obtinguts de la convolució. Això presenta dos avantatges: la conseqüent reducció d'elements d'entrada i la capacitat de poder comparar aquests elements a qualsevol part de la imatge.

Aquesta és una propietat molt interessant perquè una vegada apresa una característica en un punt concret de la imatge, la pot reconèixer després a qualsevol part d'aquesta. En canvi,

una xarxa neuronal comuna ha d'aprendre el patró novament si aquest apareix en una nova localització de la imatge.

Una altra característica és la capacitat de les capes convolucionals d'aprendre jerarquies espacials en els patrons, és a dir, una primera capa pot aprendre arestes i una segona capa identificar patrons compostos dels elements de la primera capa. D'aquesta manera es poden extreure patrons molt complexos. Aquesta característica dóna la possibilitat d'identificar conceptes visuals cada cop més abstractes i difícils de detectar.

La manera en la qual la convolució es du a terme és aplicant un filtre a la matriu de la imatge. Si es pren com a exemple la figura 5.3, es presenta una imatge d'un canal (gris) de 6x6 en la qual es pot deduir que hi ha una aresta. Si s'aplica un filtre específic per a la detecció d'arestes verticals com és el filtre 3x3 que es representa a la figura, dóna com a resultat una matriu on apareix dibuixat en blanc l'aresta.

L'aplicació del filtre es fa de la següent manera: es superposa el filtre a la part superior esquerra de la matriu de la imatge. Seguidament es multiplica cada píxel amb la seva corresponent casella del filtre i després se sumen tots els valors, és a dir: 10 x 1 + 10 x 0 + 10 * (-1) + 10 x 1 + [...] + 10 x (-1) = 0. A la següent iteració, es desplaça el filtre un píxel a la dreta, que donarà com a resultat el número 30 de la primera fila, segona columna.

Es pot observar com s'ha reduït les dimensions de la matriu resultant. Això és degut al fet que si a una matriu de 6x6 se li aplica un filtre 3x3, aquest filtre només es podrà desplaçar 3 posicions a la dreta des del vèrtex superior esquerra, i el mateix cap a la part inferior.

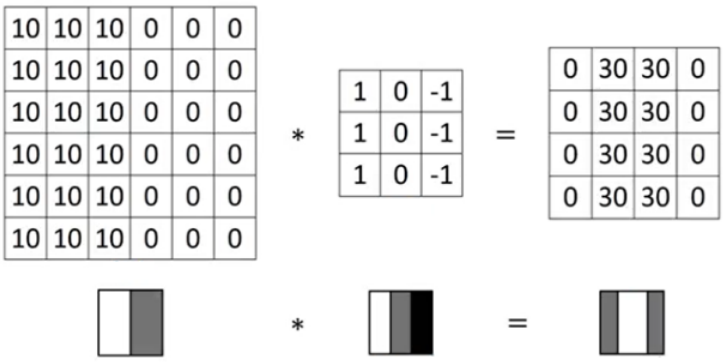

Figura 6.3: Operació de convolució.
Font: deeplearning.ai

6.4.2 Operació de padding.

Com s'ha explicat, l'operació de convolució permet tenir vàries capes amb diferents filtres per detectar conjunts de formes més complexes, però també s'ha deduït que a l'aplicar un filtre, la matriu resultant tenia unes dimensions més reduïdes que les de la matriu original. Per tant, es limita el nombre de capes de convolució que es poden aplicar a una imatge [19].

A més, també apareix un segon problema que és la pèrdua d'informació a les voreres. Un píxel a un vèrtex només es veurà reflectit en l'aplicació del filtre un cop, ja que a la pròxima iteració, el filtre s'haurà mogut un píxel, deixant el vèrtex fora. En canvi, un píxel a la part central de la imatge participarà 9 vegades en els càlculs resultants si es tracta com a l'exemple anterior d'un filtre 3x3. Per evitar aquest encongiment de la matriu i la pèrdua d'informació als extrems de la imatge, s'utilitza la tècnica del padding.

Aquesta operació consisteix a afegir una vorera a tota la matriu, el que significa que la imatge original passarà d'una mida de 6x6 a 8x8. A conseqüència d'aquest augment, es minimitzen els dos efectes negatius generats per la convolució:

Per una part, si s'aplica un filtre 3x3 a una matriu 8x8, la matriu resultant serà de 6x6. D'aquesta manera, es manté la dimensió original de la matriu, permetent aplicar més capes d'aquesta operació sense encongir la imatge resultant.

Per una altra, si s'analitza el moviment del filtre, es pot veure com el mateix vèrtex superior esquerra que abans només participava en una iteració durant el procés de convolució, ara participarà quatre vegades, sent més rellevant en el resultat final.

0	0	0	0	0	0	0	0
0	10	10	10	0	0	0	0
0	10	10	10	0	0	0	0
0	10	10	10	0	0	0	0
0	10	10	10	0	0	0	0
0	10	10	10	0	0	0	0
0	10	10	10	0	0	0	0
0	0	0	0	0	0	0	0

Figura 6.4: Operació de padding.
Font: Elaboració pròpia.

6.4.3 Operació de pooling.

Com s'ha vist en apartats anteriors, les matrius de les representacions de les imatges poden ser de mides considerablement grans. Amb l'objectiu d'accelerar els càlculs a les CNN, sovint s'utilitzen capes d'agrupament per a reduir la mida de les capes. Aquesta operació és coneguda com a *Pooling.*

El pooling consisteix a fer agrupacions depenent de la dimensió del grup, per exemple 2x2. Si la matriu original té una superfície de 20x20, aplicant el pooling de 2x2 amb un salt de 2 unitats donaria com a resultat una matriu 10x10 amb les característiques més importants de la matriu original.

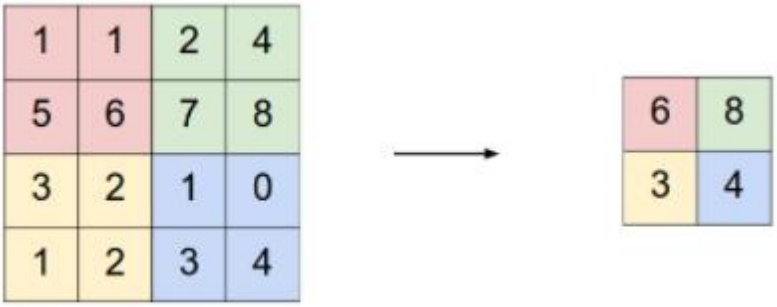

Figura 6.5: Operació de MAX Pooling 2x2 amb un desplaçament de 2.
Font: https://cs231n.github.io

L'operació més comuna per a una capa de *pooling* és l'operació MAX. Es tracta d'agafar el número més elevat dins del volum desitjat. També és possible fer la mitjana dels valors, però acostuma a tenir uns resultats pitjors.

També és aconsellable utilitzar agrupaments 2x2, ja que per a valors superiors el procés resulta massa destructiu i es perden moltes característiques de la imatge original.

6.4.4 Operació de convolució per a imatges RGB

Els apartats anteriors s'han exemplificat amb imatges en escala de grisos, per tant d'un únic canal. Una imatge RGB, tal com s'ha comentat anteriorment, disposa de tres canals separats corresponent a cada color. En aquest cas, continuant l'exemple de convolució d'una imatge 6x6, tindria una dimensió de 6x6x3. El filtre, que tenia una dimensió de 3x3 ara serà de 3x3x3, és a dir, un filtre per a cada canal.

La manera d'operar de la convolució serà idèntica a la forma que ho fa una imatge d'un canal, amb la diferència que cada canal disposa del seu propi filtre. El resultat en canvi continua sent 4x4x1 (en el cas de no utilitzar padding) en comptes de 4x4x3. La manera d'aconseguir la sortida és sumant el valor de les 27 xifres corresponent a les 9 xifres de cada canal.

L'operació de convolució també permet utilitzar diferents filtres al mateix temps. Cada filtre 3x3x3 produirà una sortida diferent, fent que la dimensió de la sortida canviï de 4x4x1 a 4x4x2 si es tracta de dos filtres diferents. L'avantatge d'utilitzar filtres diferents és la possibilitat de detectar diferents característiques a la imatge, ja siguin línies verticals, horitzontals, etc.

Com es pot observar, a mesura que augmenten les capes d'una xarxa neuronal convolucional, la dimensió de les imatges es va modificant [19].

6.4.5 Funcions d´activació.

A les xarxes neuronals, les funcions d´activació defineixen les sortides dels nodes depenent de l'entrada o conjunts d'entrades.

Com s'ha vist, les entrades d'un node es multipliquen per els pesos corresponents al node i se sumen. Aquest valor resultat és conegut com a suma d'activació del node. La suma d'activació es processarà mitjançant una funció d'activació per determinar la sortida del node.

Es podria entendre com un circuit on segons el valor d'entrada, activa o desactiva la sortida. Gràcies a la no-linealitat de les funcions d'activació, s'aconsegueix un augment en l'eficiència de l'aprenentatge per a problemes no trivials.

Existeixen diferents tipus de funcions d'activació que s'utilitzen a les xarxes neuronals:

Funció d'activació sigmoide.

La funció d'activació sigmoide o logística és una funció molt utilitzada tradicionalment. L'entrada de la funció és transformada en una sortida entre 0 i 1. Per a entrades molt superiors o inferiors a 0 i 1, la sortida obtindrà la valoració mínima o màxima respectivament degut a la seva forma en S centrada en una sortida de 0,5 per a una entrada de 0 [20].

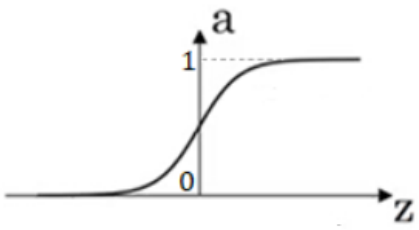

Figura 6.6: Funció d'activació sigmoide.
Font: www.deeplearing.ai

Funció d'activació tangencial hiperbòlica.

La funció d'activació tangencial hiperbòlica (tanh) és una funció similar a la sigmoide, però en aquest cas es troba centrada a 0. D'aquesta manera s'aconsegueix tenir valors a la sortida entre -1 i 1. Generalment les funcions tanh tenen uns millors resultats que les sigmoides degut a aquest increment en els valors de sortida i la seva ponderació negativa.

Com es pot observar, tant les funcions sigmoides com tangencials tenen problemes per a valors allunyats de 0 on la seva sortida queda saturada. Aquest és un problema molt freqüent, ja que els valors d'entrada acostumen a ser distants de 0.

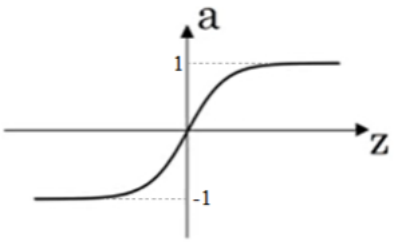

Figura 6.7: Funció d'activació tangencial hiperbòlica.
Font: www.deeplearing.ai

Funció d'activació ReL.

La funció d'activació lineal rectificada (en anglès Rectified Linear function) actua com una funció semi lineal. Per a valors superiors a 0, la sortida es comportarà de manera lineal sent la sortida igual a l'entrada. Per a valors inferiors o iguals a 0, la funció perd la linealitat donant una sortida de 0. Gràcies a la linealitat de la funció en la part positiva, les xarxes neuronals incrementen l'eficiència en l'aprenentatge, oferint xarxes molt més profundes, mentre que la no linealitat que presenta per a valors negatius i per a 0, permet produir relacions complexes en les dades que es vol aprendre.

Els nodes que implementen una funció d'activació lineal rectificada reben el nom de ReLU (Rectified Linear Activation Unit). Aquestes unitats ReLU van suposar una revolució en el desenvolupament de les xarxes neuronals, permetent produir xarxes molt més profundes que amb funcions clàssiques com són la sigmoide i la tanh [20].

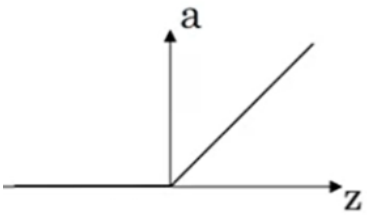

Figura 6.8: Funció d'activació ReLU.
Font: www.deeplearing.ai

6.4.6 Arquitectura general d'una CNN.

Un cop conegudes les operacions més utilitzades a l'hora de crear una xarxa neuronal convolucional, es pot entendre d'una manera més clara el funcionament d'aquesta xarxa.

Si es pren com a exemple la figura 5.9, es pot veure com una xarxa neuronal convolucional es pot dividir en dues parts ben diferenciades: el procés d'aprenentatge de característiques i el procés de classificació [21].

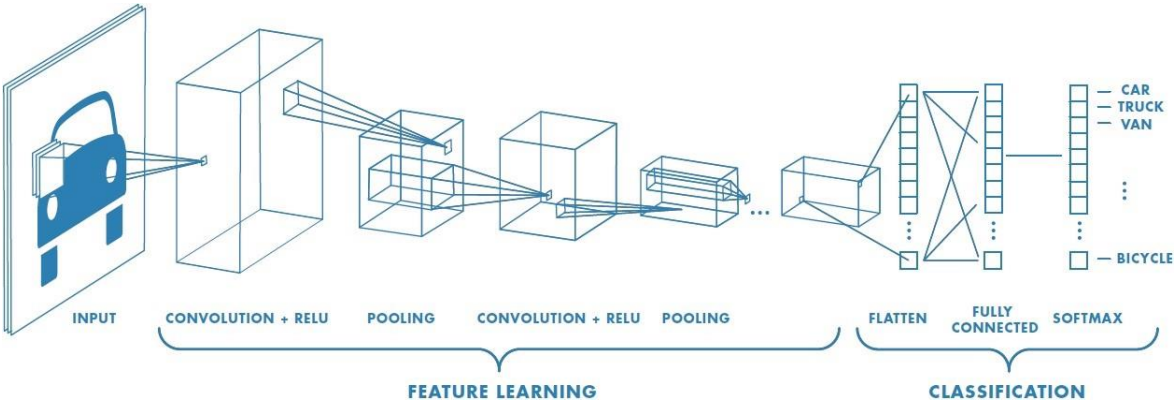

Figura 6.9: Arquitectura d'una CNN.
Font: medium.com

Com es pot observar, a l'entrada hi ha la imatge original capturada per la càmera en format de tres matrius RGB.

Primerament, aquestes matrius són tractades mitjançant l'operació de convolució per tal d'extreure característiques rellevants. Utilitzant una ReLU s'aconsegueix eliminar els valors menors o iguals a zero, és a dir, se suprimeix l'aparició de negre i així només es mantenen els elements que tenen uns valors positius.

Un cop feta l'operació de convolució i el filtratge amb la ReLU, es du a terme una reducció de la dimensió de les matrius utilitzant MAX pooling per tal d'eliminar les característiques menys rellevants.

Aquest procés es pot repetir fins a aconseguir un nombre de característiques prou elevades amb unes dimensions reduïdes. Tot aquest procediment es coneix com a procés d'aprenentatge de característiques, ja que passa de les tres matrius originals RGB a moltes matrius de mida molt reduïda corresponent als trets més diferencials.

Un cop obtingudes les matrius característiques de la imatge, es processaran en una xarxa neuronal simple per a procedir a la seva classificació.

Amb l'operació de flattening s'aconsegueix obtenir un vector amb tots els valors de l'operació anterior. Cada valor d'aquest vector es correspondrà amb una entrada de la xarxa neuronal.

La xarxa neuronal serà l'encarregada de processar aquests valors d'entrada i segons el seu entrenament i arquitectura interna donarà un resultat per a cada classe possible.

Per últim s'utilitza una funció anomenada *softmax*. Aquesta funció realitza un càlcul de la probabilitat en funció del resultat de totes les sortides. D'aquesta manera es pot oferir una única sortida que serà la corresponent a la probabilitat més elevada.

Un cop enteses les diferents operacions que es duen a terme en una CNN, cal assenyalar que la diferència característica d'aquestes xarxes és el processament de dades previ al procés de classificació.

6.5. Transfer Learning.

Tal com s'ha pogut comprovar en els apartats anteriors, els models actuals de Deep Learning tenen una quantitat molt elevada de paràmetres i dades. Per ficar un exemple, la xarxa GoogLeNet està formada per uns 6,8 milions de paràmetres dividits en 22 capes de profunditat [22]. Davant la inviabilitat que suposa la creació d'una xarxa neuronal des de zero, el més comú a l'hora de voler implementar una aplicació d'aquestes característiques és fer ús del Transfer Learning.

Transfer Learning (o transferència d'aprenentatge) és una tècnica amb la qual s'utilitza una xarxa neuronal funcional i es torna a entrenar per a un tipus d´aplicació semblant al qual va ser dissenyada. D'aquesta manera s'aconsegueix un estalvi de temps important i els resultats poden continuar sent molt positius.

La manera d'utilitzar Transfer Learning és "congelant" les capes de la xarxa neuronal més genèriques, com per exemple la detecció de característiques de la imatge. Si una xarxa estava entrenada per a reconèixer objectes, és molt probable que les capes dissenyades per a detectar formes funcionin en una altra aplicació de detecció.

Per altra banda, les capes que s'acostumen a canviar són les capes finals de la xarxa, encarregades de fer la classificació. Si una xarxa estava entrenada per a poder detectar 2000 objectes diferents, s'ha de modificar perquè només pugui classificar les classes d'objectes desitjats.

7. Implementació.

7.1. Hardware.

7.1.1. Placa de desenvolupament Jetson Nano.

Com a microcontrolador del sistema de reconeixement d'imatge s'ha decidit utilitzar una Jetson Nano 4 GB. Aquesta placa de desenvolupament ha estat dissenyada per NVIDIA amb l'objectiu d'oferir un entorn de programació de xarxes neuronals amb un preu reduït.

L'avantatge que ofereix respecte a altres plaques de desenvolupament com per exemple la Raspberry Pi 4, és la utilització d'una GPU Maxwell de 128 nuclis en comptes de fer ús d'una CPU. Un processador gràfic és més apropiat en aplicacions de Deep Learning degut a l'arquitectura que utilitza. En aquest cas, la GPU de la Jetson Nano disposa de 128 nuclis mentre que la CPU només disposa de quatre. Si més no, aquests quatre nuclis són més potents pel que fa al càlcul, la diferència de nuclis i la disponibilitat d'una memòria dedicada per a la GPU fan que la potència computacional sigui molt superior.

Com a analogia per a poder entendre d'una manera més clara la diferència entre GPU i CPU es podria dir que la CPU és un cotxe esportiu de grans prestacions, mentre que la GPU és un camió molt gran. La CPU podrà dur a terme poques accions a gran velocitat mentre que la GPU ho farà més lent, però ho processarà tot alhora.

7.1.2. Robot ABB IBR 120.

El centre d'estudis universitaris Tecnocampus disposa d'un robot antropomòrfic de 6 eixos al laboratori 4. Aquest robot és utilitzat pels estudiants de l'assignatura Robòtica, aprenent a programar utilitzant RobotStudio i el seu llenguatge de programació RAPID. Es tracta d'una bona eina d'aprenentatge que aproxima a un entorn laboral real, ja que és una eina industrial semblant a qualsevol altre robot que es pugui trobar en una fàbrica.

El robot incorpora una consola de programació on es pot controlar sense la necessitat d'un sistema informàtic. A més, permet portar el robot a les posicions desitjades per a gravar les coordenades manualment. D'aquesta manera el programador es pot assegurar del correcte posicionament del robot, a diferència de la programació sobre un entorn virtual que les

posicions seran aproximades degut a la dificultat de saber les coordenades exactes visualment.

Com a característiques tècniques importants, es tracta d'un robot amb un radi d'acció de 850 mm, repetibilitat de 0,01 mm i càrrega màxima de 3,5 Kg. Es tracta d'un robot industrial de mida mitjana amb unes bones característiques tècniques, vàlid per a infinitat d'aplicacions industrials.

7.1.3. PLC Rockwell CompactLogix.

Per a dur el control de la cinta transportadora i els polsadors de control, s'utilitzarà un PLC de la marca Rockwell. Es tracta d'un CompactLogix disponible al laboratori 4 del centre Tecnocampus. Els estudiants d'aquest centre utilitzen aquest PLC conjuntament amb el robot ABB per realitzar pràctiques a l'assignatura de robòtica. Incorpora 3 mòduls d'entrades digitals, 2 mòduls de sortides digitals i 1 mòdul de sortides de relé.

La programació s'ha fet utilitzant el programari de Rockwell RSLogix5000 disponible al Tecnocampus. El tipus de programació és estil Ladder, és a dir, estructurada en segments amb contactes i sortides.

La programació d'estil Ladder és molt utilitzada per a la programació de PLC fins i tot d´altres fabricants com Siemens o Omron, la qual cosa permet un ràpid aprenentatge dels altres sistemes.

7.2. Preparació de l'entorn de programació.

La Jetson Nano utilitza Ubuntu, una distribució de Linux. Es tracta d'un sistema operatiu molt utilitzat en plaques de desenvolupament gràcies al fet de ser de codi obert. Per tal de dur a terme la instal·lació, cal seguir la guia oficial d'Nvidia disponible al seu web.

Un cop instal·lat Ubuntu, és necessari fer una actualització de les eines bàsiques. Per fer-ho cal obrir primerament el terminal amb la combinació de tecles ctrl + alt + T. El terminal és una eina utilitzada per a controlar qualsevol característica del sistema operatiu mitjançant l'ús de comandaments. L'avantatge d'utilitzar el terminal és la rapidesa amb la qual es poden dur a terme tasques que mitjançant la manera convencional es trigaria molt més temps, a canvi, requereix un aprenentatge previ i conèixer les ordes necessàries.

Amb el terminal obert, s'ha d'utilitzar la comanda: sudo apt-get update. Aquesta ordre realitzarà una actualització de totes les eines bàsiques que es troben instal·lades.

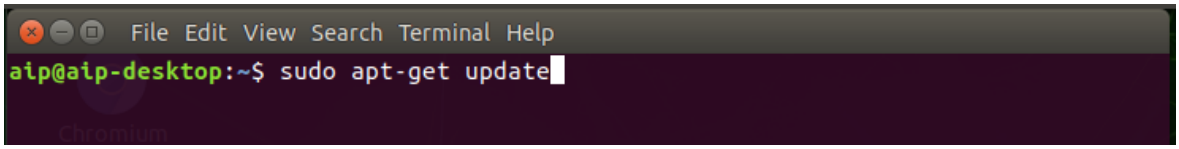

Figura 7.1: Terminal.
Font: Elaboració pròpia.

A continuació, és convenient instal·lar diferents utilitats i programes necessaris per editar arxius i utilitzar Python. Les ordres utilitzades són:

- `sudo apt-get install git cmake`
- `sudo apt-get install libatlas-base-dev gfortran`
- `sudo apt-get install libhdf5-serial-dev hdf5-tools`
- `sudo apt-get install python3-dev`

El següent pas a seguir és fer la instal·lació de *pip*. Es tracta d'un software que simplifica la descàrrega de programari i llibreries escrites en Python. La manera de fer-ho és:

- `wget https://bootstrap.pypa.io/get-pip.py`
- `sudo python3 get-pip.py`
- `rm get-pip.py`

Un altre software important a l'hora de programar, és l'editor de codi. En aquest cas s'ha decidit utilitzar Code-OSS, ja que es tracta d'un programari lliure i incorpora totes les funcions necessàries.

La seva instal·lació es pot fer amb les següents instruccions al terminal:

- `sudo apt-get install curl`
- `curl -L https://github.com/toolboc/vscode/releases/download/1.32.3/code-oss_1.32.3-arm64.deb -o code-oss_1.32.3-arm64.deb`
- `sudo dpkg -I code-oss_1.32.3-arm64.deb`

A continuació s'instal·larà PyLogix, una llibreria de Python la qual permet dur a terme una comunicació mitjançant TCP/IP amb un PLC CompactLogix. Amb PyLogix es poden

llegir i escriure directament sobre les variables del PLC d'una manera senzilla i sense haver de modificar el programa al controlador. Un altre cop es farà servir el terminal per dur a terme la instal·lació:

- `pip install pylogix`

Per últim, s'ha d'instal·lar Jetson Inference, un paquet d'utilitats creat per Nvidia amb l'objectiu d'utilitzar el mètode d'inferència. Per inferència s'entén, dins del camp del Machine Learning, com el procés d'utilitzar una xarxa neuronal entrenada amb moltes classes de dades perquè sigui capaç de reconèixer un petit conjunt d'objectes. D'aquesta manera s'aprofiten tots els paràmetres anteriors i el temps necessari per a crear la xarxa es redueix considerablement. Aquest mètode és un tipus de Transfer Learning, i per tant és recomanable que la xarxa neuronal que s'utilitzi tingui una funció semblant a la que es vol aconseguir.

Per tal d'instal·lar Jetson Inference, primer cal instal·lar numpy, una llibreria per a fer operacions matemàtiques:

- `sudo apt-get install git cmake libpython3-dev python3-numpy`

Un cop instal·lat numpy, es podrà dur a terme la instal·lació de Jetson Inference amb les següents instruccions:

- `git clone --recursive https://github.com/dusty-nv/jetson-inference`
- `cmake`

A continuació apareix un instal·lador on es demana que se seleccioni quins models es volen descarregar:

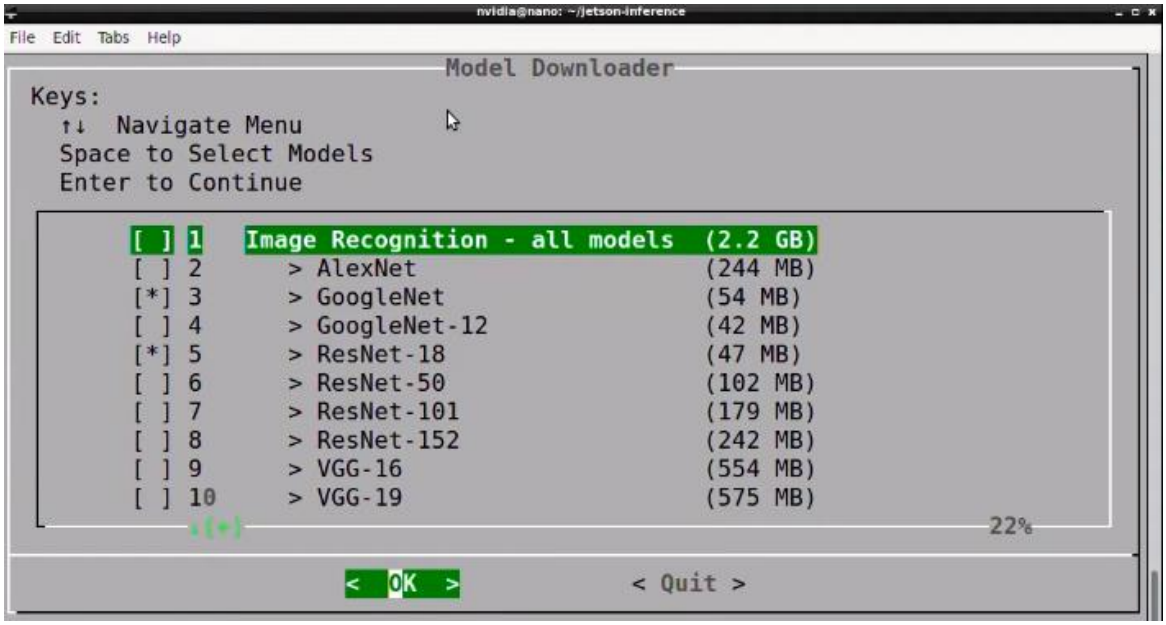

Figura 7.2: Instal·lador de Jetson Inference.
Font: Elaboració pròpia.

S'aconsella seleccionar ResNet-18, ja que serà la que s'utilitzarà més endavant per a fer la detecció de peces LEGO, però és interessant instal·lar GoogleNet i AlexNet per si es volen fer altres tipus de deteccions més endavant. Un cop seleccionats els models desitjats, continuar amb el botó OK.

El procés d'instal·lació pot trigar uns 30-35 minuts depenent de la connexió a internet.

Per a finalitzar, cal descarregar les utilitats necessàries:

- `make -j$(nproc)`
- `sudo make install`
- `sudo ldconfig`
- `sudo apt-get install v4l-utils`

7.3. Creació de la base de dades.

Un pas essencial en el procés d'entrenament d'una xarxa neuronal és l'obtenció d'imatges per a crear la base de dades. Aquest conjunt d'imatges serà l'encarregat de determinar les característiques de cada classe específica. Per aquest motiu, és molt important que la presa d'imatges sigui realitzada a l'entorn de treball o en unes condicions el més semblant possibles, si no, és molt probable que a l'hora d'implementar el sistema de visió no es dugui a terme una correcta detecció, ja que el model entrenat haurà après uns trets característics de les formes diferents als desitjats.

Abans de crear la base de dades, és important crear un arxiu de text anomenat *labels* amb el nom de les classes que es volen crear. Aquest arxiu serà utilitzat per les properes utilitats per a saber quins grups es volen identificar. Una manera de crear un arxiu de text en Ubuntu és utilitzant Gedit de la següent manera:

- `gedit labels.txt`

D'aquesta manera s'obrirà un editor de text amb el nom labels, on s'ha d'introduir les classes desitjades separades en diferents files tal com s'indica a la següent figura:

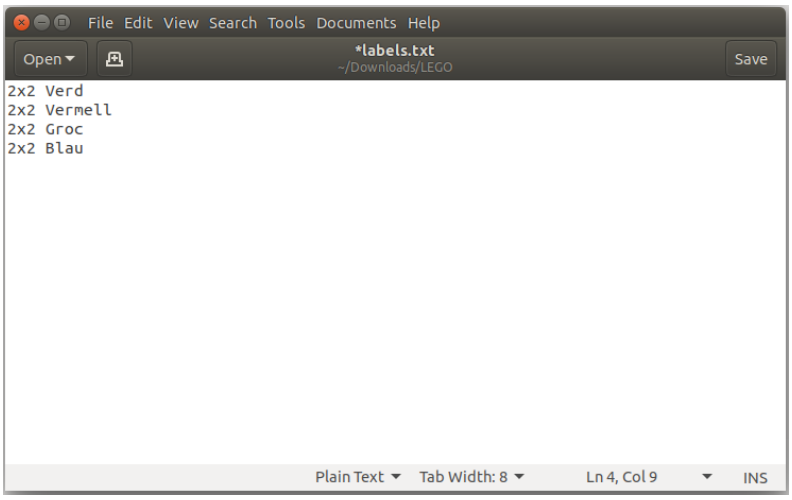

Figura 7.3: Arxiu labels.txt.
Font: Elaboració pròpia.

És molt important que el nom de l'arxiu sigui exactament labels.txt, ja que els programes posteriors estan programats per a obrir aquest nom en concret i si no es troba donarà error i no es podrà continuar amb el procediment.

Jetson Inference incorpora una eina anomenada *camera-capture* la qual permet agafar fotografies per al seu posterior etiquetatge. L'etiquetatge ajuda a la xarxa a entendre quin objecte hi ha a la imatge i a on es troba. D'aquesta manera s'optimitza l'entrenament i s'obtenen uns millors resultats.

Per poder accedir a l'eina *camera-capture*, primerament s'ha d'accedir al directori mitjançant l'ordre:

- `cd Downloads/jetson-inference/tools/`

Un cop al directori, amb la següent instrucció es pot executar el programa:

- `camera-capture --width=800 --height=600`

D'aquesta manera l'aplicació s'executarà utilitzant la càmera instal·lada al port CSI amb una resolució de 800x600 píxels.

En el moment d'iniciar, es donarà l'opció d'escollir quin tipus d'entrenament es vol fer: detecció d'objectes o classificació. Classificació intentarà donar sempre una sortida segons els percentatges de semblança, mentre que detecció només activarà una sortida en el moment de reconèixer un objecte entrenat. En el moment de seleccionar detecció, apareixerà la pantalla principal on cal complimentar la direcció on es vol desar informació generada amb les fotografies i també es demanarà la ubicació de l'arxiu labels.txt generat anteriorment.

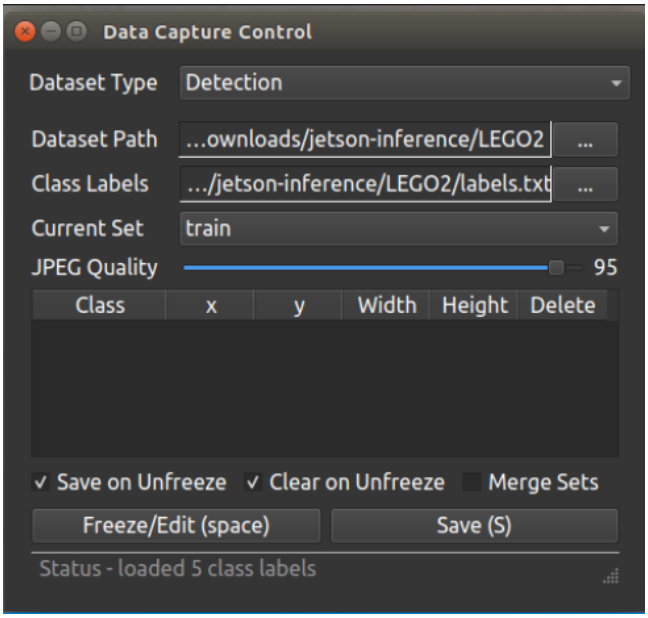

Figura 7.4: Aplicació Camera-capture.
Font: Elaboració pròpia.

A continuació apareixerà una finestra mostrant la càmera connectada a la Jetson nano. Per a generar una imatge d'entrenament, s'ha de congelar la imatge amb el botó Freeze/Edit de la figura 6.4. Un cop pausada la imatge es donarà l'opció de dibuixar un requadre per a indicar que hi ha un objecte que es vol detectar.

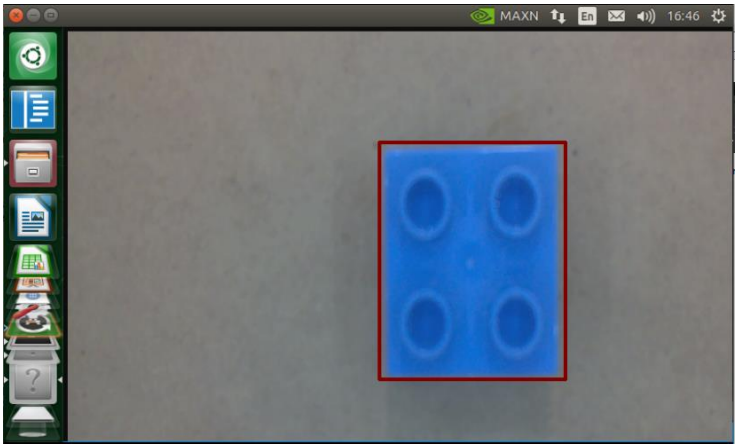

Figura 7.5: Etiquetatge utilitzant Camera-capture.
Font: Elaboració pròpia.

Un cop creat el requadre, a la finestra principal apareixerà una nova línia indicant un tipus de classe amb una posició a la imatge i les dimensions. És molt important canviar la classe a la correcta, en aquest cas, es tractava d'una peça 2x2 de color blava.

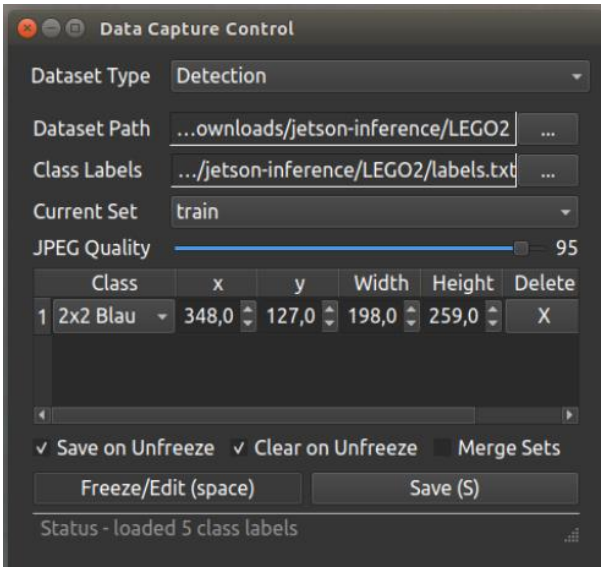

Figura 7.6: Etiquetatge d'una classe.
Font: Elaboració pròpia.

Per a aquest projecte, s'han agafat 30 fotografies per a cada classe, fent un total de 120 només d'entrenament. Aquest número resulta molt inferior al recomanable per a una aplicació de reconeixement d'imatge, però en aquest cas és suficient per a tenir un bon rendiment, ja que es preveu que la peça sigui reconeguda sempre al mateix punt. D'aquesta manera el model no ha d'estar entrenat per a ser capaç de fer deteccions des de diferents perspectives. Un

model més complex pot estar format per unes 500 – 1000 imatges per a cada classe per a obtenir un resultat adequat.

Un cop obtingudes totes les fotografies, s'ha de tornar a agafar imatges però en aquest cas per a realitzar la validació. El mètode serà el mateix que amb les imatges d'entrenament però canviant l´opció *Current Set* a *val.* Aquestes fotografies seran utilitzades pel model per a comprovar els resultats obtinguts i d'aquesta manera ajustar els paràmetres.

Per a un funcionament correcte, és recomanable tenir un 20% del total de fotografies d'entrenament com a validació, és a dir, en el cas del present projecte hi ha 30 imatges d'entrenament i 6 de validació.

7.4. Entrenament de la xarxa neuronal.

Una vegada generada la base de dades, el següent pas és entrenar un nou model utilitzant les eines disponibles de Jetson-Inference.

Primerament, cal accedir al directori de l'eina d'entrenament mitjançant la comanda:

- `cd Downloads/Jetson-inference/python/training/detection/ssd/`

Des de la carpeta ssd, es pot utilitzar l'aplicació d'entrenament *train.py* amb la seguent instrucció:

- ```
  python3 train_ssd.py --dataset-type=voc
  --data=/home/jetson/Downloads/jetson-inference/LEGO2
  --model-dir=/home/jetson/Downloads/jetson-inference/LEGO
  --batch-size=4 --epochs=20
  ```

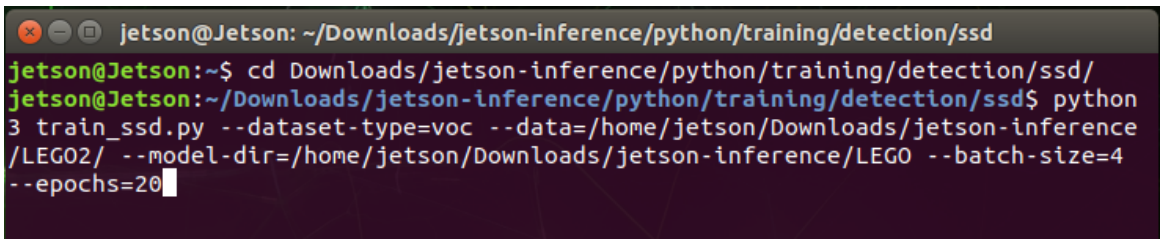

Figura 7.7: Instrucció d'entrenament.
Font: Elaboració pròpia.
  ```

Aquesta instrucció permet modificar els paràmetres per a personalitzar l'entrenament de la xarxa.

- Dataset-type: Especifica en quin format es troba la base de dades, en aquest cas és del tipus PASCAL-VOC.
- Data: Directori on es troba la base de dades creada.
- Model-dir: Direcció on es desarà el model entrenat.
- Batch-size: Número d'imatges que es processaran alhora. Aquesta opció influeix en el consum de memòria RAM, per tant no és recomanable un número superior a 4.
- Epochs: Número de vegades que es processarà la base de dades completa, és a dir, el nombre d'iteracions que es farà. En el cas del model entrenat en aquest projecte, ha sigut de 20 epochs, mentre que el recomanable per aplicacions més complexes és d'unes 80 o 100 iteracions, amb l'inconvenient d'un augment considerable del temps d'entrenament.

Un cop finalitzat l'entrenament, s'haurà generat una carpeta amb informació de cada epoch.

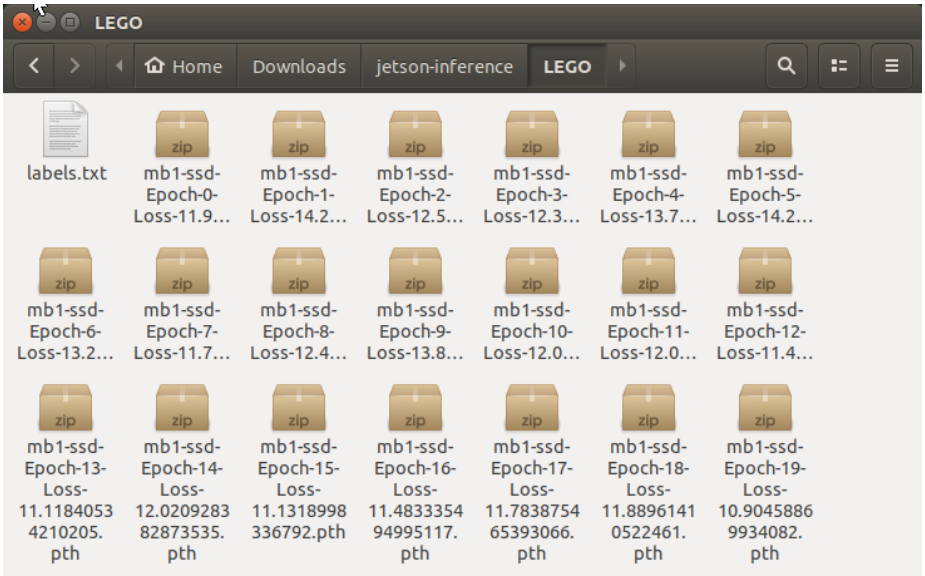

Figura 7.8: Carpeta del nou model.
Font: Elaboració pròpia.

Per a poder utilitzar el model, cal fer una conversió a un format reconegut per a la llibreria utilitzada en Python. Aquest pas s'ha dut a terme utilitzant l'eina ONNX (Open Neural Network Exchange). Es tracta d'una eina que transforma les xarxes a un format obert de representació de xarxes neuronals.

Per a fer la transformació s'ha utilitzat la següent instrucció:

- `py3 onnx_export.py --model-dir=/home/jetson/Downloads/jetson-inference/LEGO`

En acabar la transformació, s'haurà creat un nou arxiu amb el nom ssd-mobilenet.onnx.

7.5. Programació.

Tal com s'ha comentat anteriorment, el programa de detecció d'objectes estarà en llenguatge Python.

El primer que s'ha de fer en un programa d'aquestes característiques és importar les llibreries necessàries:

- jetson.inference: llibreria destinada a utilitzar xarxes neuronals.
- jetson.utils: llibreria amb utilitats específiques per a programar sobre la Jetson nano.
- time: llibreria per a poder utilitzar temporitzadors i introduir esperes al programa.
- pylogix: llibreria per a poder escriure i llegir variables d'un PLC Rockwell CompactLogix. Cal especificar l'adreça IP del PLC, tal com s'indica a la figura 7.9.

```
1  import jetson.inference
2  import jetson.utils
3  import time
4  from pylogix import PLC
5  comm = PLC()
6  comm.IPAddress = '192.168.100.50' #direccio del PLC
```

Figura 7.9: Importació de les llibreries.
Font: Elaboració pròpia.

A continuació, es pot importar el model creat anteriorment de la següent manera:

```
7  net = jetson.inference.detectNet(argv=["--model=/home/aip/Downloads/LEGO/ssd-mobilenet.onnx",
8   "--labels=/home/aip/Downloads/LEGO/labels.txt", "--input-blob=input_0", "--output-cvg=scores",
9   "--output-bbox=boxes","--threshold=0.8" ]) #configuracio del model de xarxa neuronal
```

Figura 7.10: Importació del model entrenat.
Font: Elaboració pròpia.

Aquesta instrucció consta dels següents paràmetres:

- model: indica la direcció on es troba desat el model.

- labels: direcció on es troba l'arxiu labels.txt que conté les diferents classes.
- input-blob: entrada del blob (Binary Large Object o Objecte Binari Gran), referent a les imatges. En aquest cas la càmera es troba a la input_0, el port 0 de la Jetson nano.
- output-cvg: text de sortida quan es realitza una detecció. "*Scores*" mostrarà quina puntuació ha tingut l'objecte en una valoració del 0 a l'1.
- output-bbox: genera una figura per a indicar la posició de la detecció. "*boxes*" generarà un requadre al voltant de la detecció.
- threshold: valoració mínima per a considerar que una detecció és vàlida. En aquest projecte s'ha considerat un 80% de valoració. Aquesta decisió s´ha pres observant les valoracions obtingudes en pràctiques realitzades de la detecció. Amb una valoració inferior a 75% s'obtenien deteccions incorrectes de manera constant.

Un cop importat el model, s'especifica la utilització de la càmera connectada al port 0 de la Jetson nano amb una resolució de 1280 x 720 píxels. Encara que la càmera és capaç d'una resolució més elevada, la proposada aconsegueix uns resultats molt positius amb un rendiment elevat.

A continuació es configura l'aplicació encarregada de mostrar els resultats de la càmera, concretament glDisplay, inclosa a la llibreria de jetson.utils. Per evitar fer la configuració anterior, la resta de programa anirà en un bucle while amb condició de tenir oberta la finestra on s'observaran els resultats de la càmera. La finestra en qüestió romandrà oberta la resta de programa i per tant no es sortirà en cap cas del bucle amb l'única excepció de tancar manualment la finestra, la qual cosa provocarà la finalització del programa.

```
10  camera = jetson.utils.gstCamera(1280,720,'0') #Configuracio de la camera
11  display = jetson.utils.glDisplay() #Aplicacio per veure la camera
12  while display.IsOpen(): #Mentre s'estigui executant el programa:
13          img, width, height = camera.CaptureRGBA() #Processa una imatge del video i agafa el parametres
14          detections = net.Detect(img, width, height) #Escaneja la imatge en busca d'objectes
15          display.RenderOnce(img, width, height) #Mostra la imatge
16          display.SetTitle("Object Detection | Network {:.0f} FPS".format(1000.0 / net.GetNetworkTime())) #Titol de la finestra
```

Figura 7.11: Configuració de la detecció.
Font: Elaboració pròpia.

La primera instrucció dins del bucle és agafar una imatge de la càmera. La funció camera.CaptureRGBA retorna les matrius RGB generades per la càmera, a més, indica les dimensions d'aquestes. A continuació es pot dur a terme el processament d'aquestes matrius per tal de dur a terme la detecció utilitzant el model importat.

La funció net.Detect automatitza el procés de detecció, utilitzant les matrius RGB com a entrada a la xarxa neuronal i produint una sortida en cas d'haver-hi una valoració positiva per a qualsevol classe.

A continuació es renderitza la imatge per tal de mostrar-la per pantalla amb la funció RenderOnce. Per tal de donar-li un títol a la finestra, s´ha utilitzat SetTitle i s'ha inclòs les imatges per segon de la càmera com a indicador del rendiment del sistema. Obtenir uns FPS elevats indica un funcionament fluid de la placa de desenvolupament. En cas d'obtenir un número inferior de 15, voldrà dir que el processament de la imatge en el model resulta molt complex i per tant es podria reduir la resolució de la càmera per minimitzar les entrades a la xarxa neuronal i guanyar velocitat de processament.

El model utilitzat permet la detecció de més d'una classe alhora, encara que sigui innecessari en el present projecte, ja que només es mourà una peça cap al sistema de visió per cada cicle de treball. Les deteccions es troben emmagatzemades a la variable detections. Mitjançant un bucle for, es modificaran les sortides del PLC segons la classe detectada, activant la sortida corresponent i desactivant la resta de sortides.

Classe	Sortida (PLC)
2 (blava)	Local:3:O.Data.12
3 (vermella)	Local:3:O.Data.14
4 (groga)	Local:3:O.Data.13
5 (verda)	Local:3:O.Data.11

Taula 7.1: Sortides associades a la classe.

```
17
18          for i in detections: #Bucle for per a cada deteccio
19
20                  if i.ClassID == 5: #Si la peca es blava, activar sortida del PLC 6-11 i desactivar la resta
21                      sortida = comm.Write('Local:6:O.Data.11',True)
22                      sortida = comm.Write('Local:6:O.Data.12',False)
23                      sortida = comm.Write('Local:6:O.Data.13',False)
24                      sortida = comm.Write('Local:6:O.Data.14',False)
25                  if i.ClassID == 2: #Si la peca es verda, activar sortida del PLC 6-12 i desactivar la resta
26                      sortida = comm.Write('Local:6:O.Data.11',False)
27                      sortida = comm.Write('Local:6:O.Data.12',True)
28                      sortida = comm.Write('Local:6:O.Data.13',False)
29                      sortida = comm.Write('Local:6:O.Data.14',False)
30                  if i.ClassID == 4: #Si la peca es groga, activar sortida del PLC 6-13 i desactivar la resta
31                      sortida = comm.Write('Local:6:O.Data.11',False)
32                      sortida = comm.Write('Local:6:O.Data.12',False)
33                      sortida = comm.Write('Local:6:O.Data.13',True)
34                      sortida = comm.Write('Local:6:O.Data.14',False)
35                  if i.ClassID == 3: #Si la peca es vermella, activar sortida del PLC 6-14 i desactivar la resta
36                      sortida = comm.Write('Local:6:O.Data.11',False)
37                      sortida = comm.Write('Local:6:O.Data.12',False)
38                      sortida = comm.Write('Local:6:O.Data.13',False)
39                      sortida = comm.Write('Local:6:O.Data.14',True)
```

Figura 7.12: Condicionament de sortides.
Font: Elaboració pròpia.

Per últim, s'ha inclòs dues instruccions més per poder veure al terminal quina classe ha sigut detectada i a quin número correspon. També s'ha utilitzat una espera d'un segon per a cada iteració, ja que no és necessari una velocitat de resposta molt elevada.

```
41              print(i.ClassID) #escriu el numero de la classe
42              print("class_desc:", net.GetClassDesc(i.ClassID)) # a quina classe correspon
43              time.sleep(1) #espera d'1 segon
```

Figura 7.13: Comprovació de detecció
Font: Elaboració pròpia.

7.6. Algorisme del PLC.

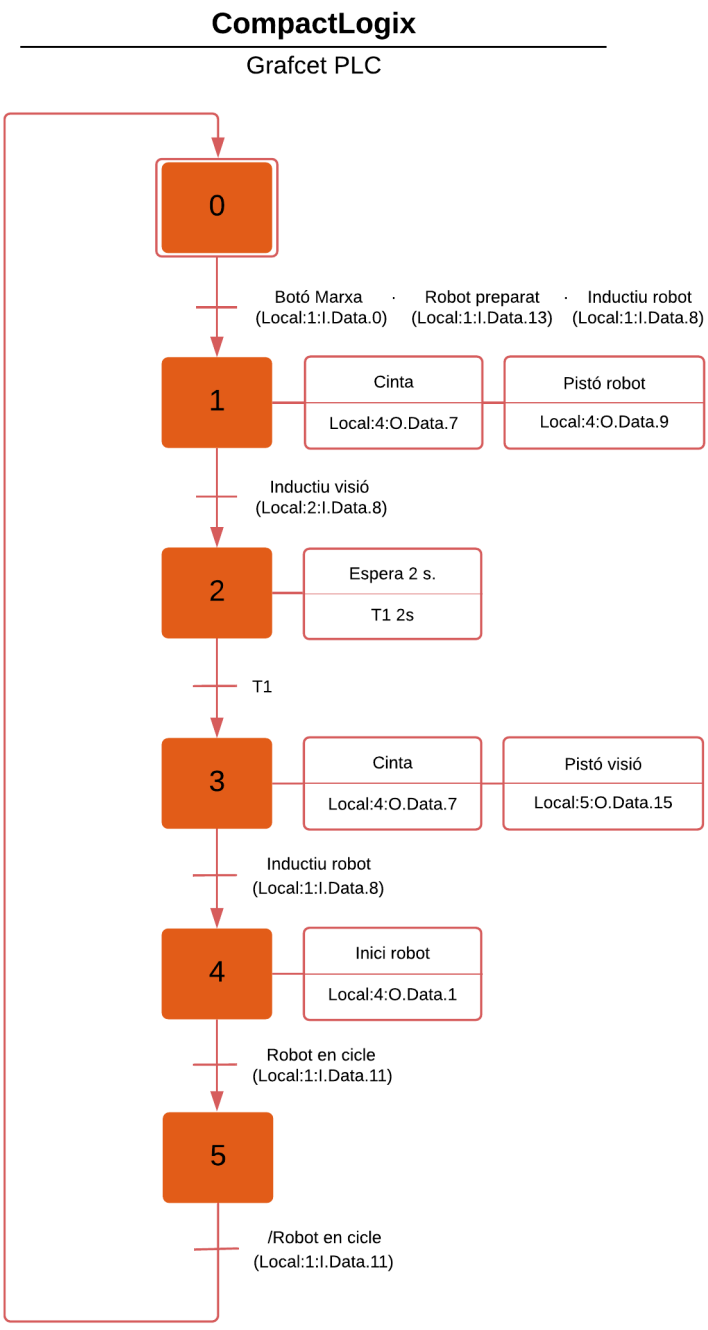

Figura 7.14: Grafcet del PLC.
Font: Elaboració pròpia.

Per a fer el control de la cinta que mourà les peces entre les dues estacions, s'ha dissenyat un petit programa amb un PLC CompactLogix de la marca Rockwell Automation.

En rebre tensió, el PLC s'inicialitzarà a l'etapa 0, una etapa d'espera on romandrà fins que l'operari premi el botó físic de marxa disponible al quadre elèctric. En aquest punt, l'operari ha d'introduir una peça dins del forat en forma de quadre del carro de la cinta. Un cop accionat el botó, si el robot es troba a la posició de repòs i si el carro és a l'estació del robot, es farà el pas a la primera etapa.

A la primera etapa, s'activarà el motor de la cinta per tal de fer el transport de la peça a l'estació on es troba el sistema de visió. Les dues estacions disposen d'uns pistons que actuen a mode de fre perquè el carro quedi ben posicionat. Aquests pistons es troben comandats per dues electrovàlvules. Per tal de permetre el moviment cap a l'estació de visió, s'accionarà l'electrovàlvula per retreure el pistó de l'estació del robot. La cinta actuarà fins que el carro arribi al sensor inductiu de l'estació de visió, on s'activarà la segona etapa.

A la segona etapa es farà una espera de dos segons per donar temps al sistema de visió a capturar una bona imatge sense moviments, d'aquesta manera es redueixen els possibles errors de reconeixement degut a imatges borroses. A més, també es dóna temps a què la Jetson Nano activi unes sortides del PLC segons la peça detectada mitjançant la comunicació TCP/IP.

Després dels dos segons d'espera, s'activarà la tercera etapa posant en funcionament el motor de la cinta i activant l'electrovàlvula del pistó de l'estació de visió. Quan el carro arriba a l´estació del robot, el sensor inductiu activarà la quarta etapa on s'atura la cinta. En aquesta etapa s'envia un senyal al robot perquè pugui començar el cicle de classificació.

En aquest punt ha de rebre un senyal del robot que indiqui que aquest ha començat el cicle de classificació. Un cop rebut el senyal, desactivarà la sortida d'inici del robot per indicar que ho ha rebut (handshake). Quan el robot ha acabat de fer el cicle, desactivarà l'entrada 1q del mòdul q indicant que ha acabat el cicle. En aquest moment es tornarà a l'etapa 0, preparat per tornar a començar.

7.7. Algorisme del robot.

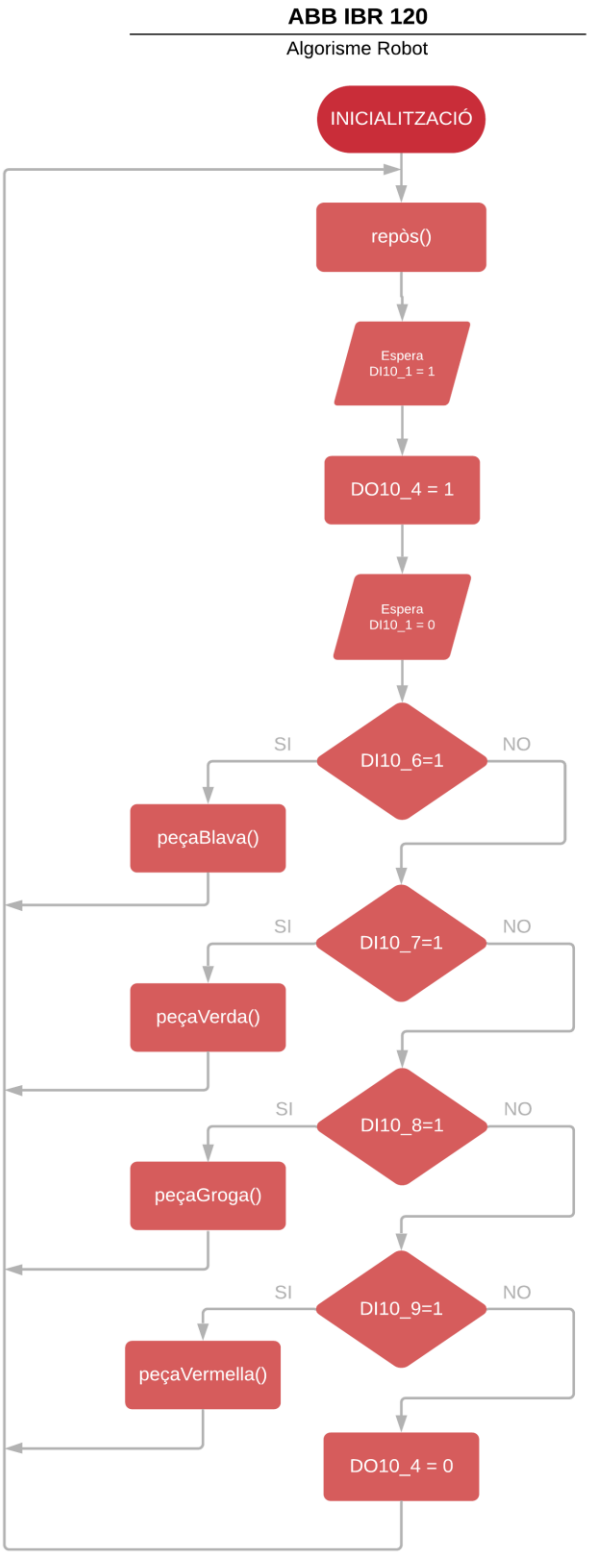

Figura 7.15: Algorisme del robot.
Font: Elaboració pròpia.

El robot ABB disponible al laboratori 4 del centre d'estudis universitaris Tecnocampus és el model IBR 120. Aquest robot és programable mitjançant el software RobotStudio de la mateixa companyia o també a través de la consola de control incorporada. Originalment es volia programar utilitzant el software, però degut a problemes de llicències s'ha decidit fer la programació utilitzant la consola. La principal diferència entre les dues maneres de programar és que utilitzant RobotStudio es pot simular el sistema en un entorn virtual mentre que amb la consola s'ha d'anar més amb compte, ja que les proves es fan directament sobre el robot real.

En el moment de rebre tensió, el robot efectua un procés d'inicialització intern on comprova l'estat del sistema i engega els subsistemes necessaris. Un cop inicialitzat, el robot esperarà a rebre senyal de que s'ha polsat el botó de rearmament. En prémer el botó, s'iniciarà la funció *repòs*. Aquesta funció porta al robot a una posició segura d'espera i obre la pinça. En aquest punt el robot es troba a l'espera del senyal del PLC que li dóna permís per començar el cicle. Un cop rebut el senyal a l'entrada 10_1, el robot respondrà al PLC amb un altre senyal a la sortida 10_4, indicant que inicia el cicle de classificació. Com a seguretat, el PLC tornarà a baixar el senyal per indicar que s'ha rebut correctament el senyal d'inici. Aquesta tècnica coneguda com a *handshake*, és utilitzada per evitar que el robot comenci el cicle sense que el PLC en tingui constància, com per exemple, si es trenca el cable del senyal. D'aquesta manera s'evita que el carro es mogui mentre el robot es troba treballant.

Un cop iniciat el cicle, el robot analitza quina entrada es troba activa per a saber quina peça hi ha al carro. Si l'entrada 10_6 es troba a 1, vol dir que la peça és blava. Si és l'entrada 10_7, la peça serà verda, la 10_8 indica peça groga i per últim, la 10_9 indica vermella. Depenent del color de la peça, es durà a terme la funció corresponent a cada color. En aquestes funcions, el robot avançarà fins a una posició propera al carro per fer una primera aproximació a una velocitat elevada i seguidament, acabarà d'agafar la peça tancant la pinça. A continuació, el robot anirà fins a un punt proper a la posició destinada al color corresponent a una velocitat més elevada per posteriorment, fer una aproximació més lenta a la zona objectiu. En aquest punt, obrirà la pinça deixant la peça i tornarà a executar la funció *repòs*, on activarà la sortida 10_1, a l'espera d'una altra peça.

7.8. Resultats.

Després d'implementar les tres programacions als seus respectius dispositius (Robot, PLC i Jetson nano), s'han dut a terme un conjunt de proves per tal de validar el correcte funcionament del conjunt.

Per a una correcta subjecció de la peça s'ha mecanitzat una fusta amb la forma del carro i un forat quadrat de la mida de les peces. D'aquesta manera s'assegura que la peça queda fixada en una posició i es redueixen els problemes a l'hora de fer el reconeixement i també en el moment d'actuar el robot.

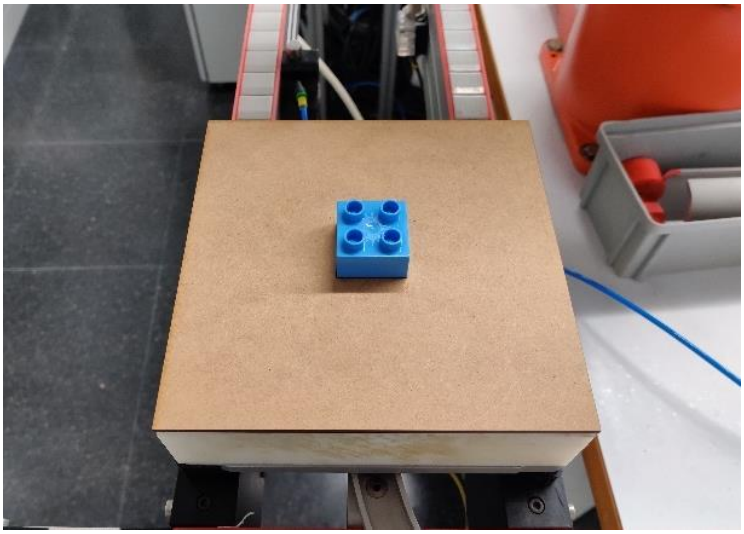

Figura 7.16: Carro amb la fusta de subjecció de la peça.
Font: Elaboració pròpia.

Un cop introduïda la peça al carro es comprova que realment, tot i prémer el botó de marxa, el carro no es mou si no es troba en posició i el robot emetent el senyal de confirmació de repòs.

En arribar al sistema de detecció, el carro efectua correctament una parada de dos segons fins a continuar la marxa. En aquesta parada, es pot comprovar mitjançant una pantalla connectada a la Jetson Nano, com la detecció ha sigut correcta i ha estat indicada al terminal de l'entorn de desenvolupament. També es comprova que les sortides digitals del mòdul 4 del PLC es modifiquen en funció de la classe detectada de manera correcta.

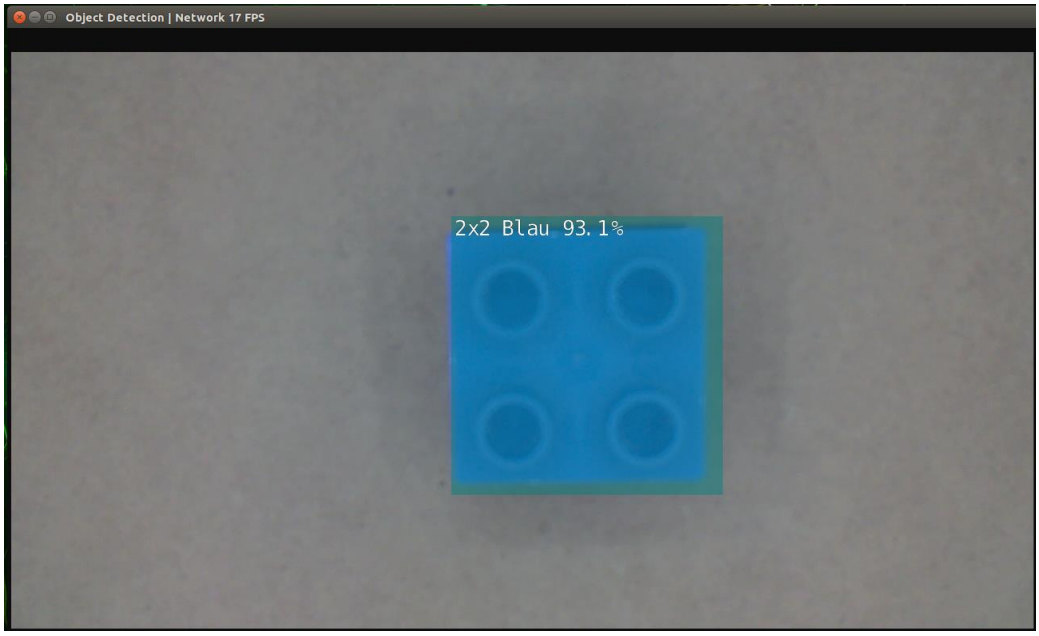

Figura 7.17: Detecció d'una peça blava
Font: Elaboració pròpia.

Quan el carro torna a posicionar-se a l'estació del robot es du a terme de manera satisfactòria el handshake, fent que el robot iniciï el moviment d'aproximació cap a la peça.

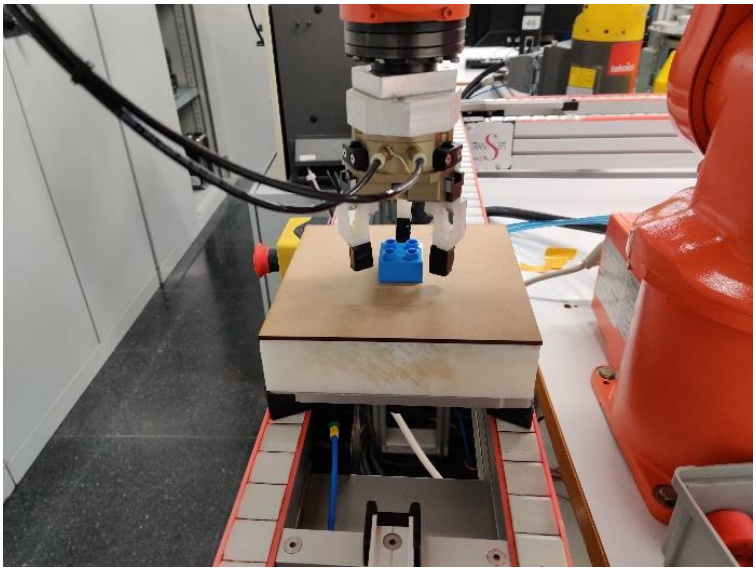

Figura 7.18: Aproximació del robot a la peça.
Font: Elaboració pròpia.

Tal com es pot observar a la següent imatge, el robot ha sigut capaç de posicionar les peces de manera separada, el qual indica una correcta classificació segons el tipus de peça.

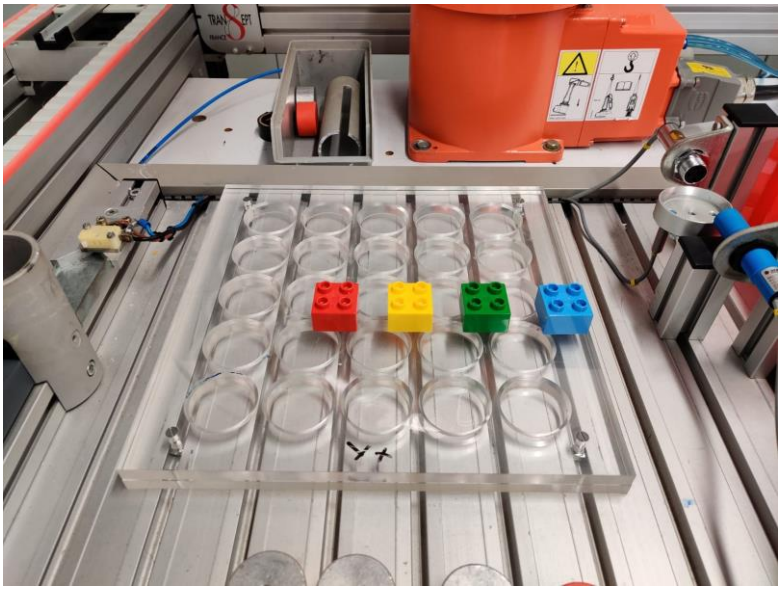

Figura 7.19: Resultat final de la classificació.
Font: Elaboració pròpia.

8. Planificació.

8.1. Tasques i prelacions.

Codi	Tasca	Durada (hores)	Prelacions
1.1	Avantprojecte	70	-
1.2	Recerca d'informació inicial	20	1.1
1.3	Recerca de materials	8	-
1.4	Disseny del sistema d'il·luminació	40	-
1.5	Disseny del sistema de visió	8	-
1.6	Comanda de materials	4	1.3;1.4;1.5
1.7	Construcció del sistema d'il·luminació	10	1.6
1.8	Creació d'una base de dades	20	1.7
1.9	Programació de la xarxa neuronal	80	1.8
1.10	Entrenament de la xarxa neuronal	40	1.9
1.11	Compilació de la xarxa neuronal en la Jetson Nano	20	1.10
1.12	Muntatge dels components electrònics	12	1.6
1.13	Generació del codi de la càmera	40	1.12
1.14	Programació del Robot	24	-
1.15	Connexió Jetson – Robot - PLC	20	1.14
1.16	Implementació	4	1.11;1.13;1.15
1.17	Proves de funcionament	20	1.16
1.18	Elaboració de la documentació	90	-
1.19	Lliurament de la documentació	2	1.18
1.20	Elaboració de la presentació	48	1.16;1.18
1.21	Defensa del projecte	2	1.20

Tabla 8.1: Tasques i prelacions.
Font: Elaboració pròpia.

8.2. Descripció de les tasques.

1.1. Avantprojecte.

Realització de l'avantprojecte, inclou recerca d'informació necessària i redacció de la documentació. També hi ha inclòs el pressupost del projecte.

1.2. Recerca d'informació inicial.

Recerca de la informació necessària per poder començar el projecte.

1.3. Recerca de materials.

Cerca de diferents materials necessaris per a l'elaboració del projecte i de diferents fabricants per tenir un coneixement de les possibilitats en el disseny dels diferents sistemes i veure les possibilitats.

1.4. Disseny del sistema d'il·luminació.

Disseny estructural de suport i del sistema d'il·luminació. Elaboració dels plànols necessaris i selecció de materials per a dur a terme la seva construcció.

1.5. Disseny del sistema de visió.

Disseny del sistema de visió tenint en compte les especificacions tècniques necessàries i el material disponible.

1.6. Comanda de materials.

Compra dels materials necessaris en el projecte segons els dissenys establerts i la recerca feta a tasca 1.3.

1.7. Construcció del sistema d'il·luminació.

Preparació de tots els components necessaris segons els plànols i especificacions definides a la tasca A. La preparació de les peces es farà utilitzant una talladora làser disponible al Tecnocampus. L'assemblatge de les peces es farà mitjançant cola blanca i cargols en els elements necessaris.

1.8. Creació d'una base de dades.

Per a la correcta programació i entrenament de la xarxa neuronal serà necessari disposar d'una base de dades, concretament d'imatges per tal de tenir un correcte reconeixement de l'objecte desitjat. En aquesta tasca es prendran fotografies utilitzant el sistema d'il·luminació dissenyat a la tasca precursora per tal que els resultats siguin el més semblant a un cas real i no hi hagi canvis en la lluminositat ni el to de la imatge.

1.9. Programació de la xarxa neuronal.

En aquesta tasca s'ha dut a terme la programació de la xarxa neuronal utilitzant Jetson Inference i altres utilitats descrites en el projecte..

1.10. Entrenament de la xarxa neuronal.

Un cop programada la xarxa, s'ha fet l'entrenament utilitzant la base de dades generada anteriorment. Aquest entrenament fa possible la identificació de diferents peces segons unes característiques definides en el projecte.

1.11. Compilació de la xarxa neuronal en la Jetson Nano.

Càrrega del programa i configuració del controlador (Jetson Nano) per fer les comprovacions necessàries de funcionament i fer la depuració necessària en un entorn real.

1.12. Muntatge dels components electrònics.

Muntatge de tots els components electrònics que formen part tant del sistema d'il·luminació com del sistema de control (Càmera, Jetson Nano i llum LED).

1.13. Generació del codi de la càmera.

En aquesta tasca s'ha programat i comprovat el correcte funcionament de la càmera. Per evitar possibles errors, s'ha utilitzat un programa de proves que només habilita els senyals de sortida en detectar una peça sense actuar sobre el PLC. També s'ha dissenyat el programa final, però no ha sigut provat durant aquesta tasca ja que la resta de components no es trobaven disponibles.

1.14. Programació del Robot.

Utilització de la consola de control per programar el robot ABB IRB 120, tenint en compte les interaccions possibles amb la Jetson Nano i el PLC per dur a terme el procés de classificació. La generació de codi s'ha dut a terme utilitzant l'editor de text Notepad ++.

1.15. Connexió Jetson Nano – Robot - PLC.

En aquest punt es farà la configuració per tal de dur a terme una comunicació entre la Jetson Nano, el robot i el PLC mitjançant el protocol TCP/IP. Per a fer aquesta connexió caldrà una connexió Ethernet entre els tres dispositius.

1.16. Assemblatge.

Assemblatge final del conjunt i comprovació que tots els subsistemes estan ben muntats i a la seva posició.

1.17. Proves de funcionament.

Prova final de funcionament amb tots els components del projecte. Depuració del codi del Robot, Càmera i la Jetson Nano per tal de deixar el sistema en perfecte funcionament. Comprovació amb peces reals sobre l'entorn objectiu del projecte.

1.18. Elaboració de la documentació.

Durant el transcurs del projecte s'ha anat documentant tot el procés en una memòria a part de la realització dels estudis necessaris sobre la seva viabilitat.

1.19. Lliurament de la documentació.

El lliurament de la memòria final el dia 17/06/2021. En aquest lliurament, es fa entrega de tota la documentació que compon aquest projecte, tant memòria com estudi econòmic, plànols i annexos.

1.20. Elaboració de la presentació.

Desenvolupament d'una presentació per tal de defensar el projecte davant d'un tribunal. Cal fer el disseny i una demostració del prototip desenvolupat en el treball.

1.21. Defensa del projecte.

Defensa del projecte mitjançant l'explicació del procés d'elaboració del treball realitzat i una demostració en directe dels resultats obtinguts.

8.3. Diagrama de Gantt.

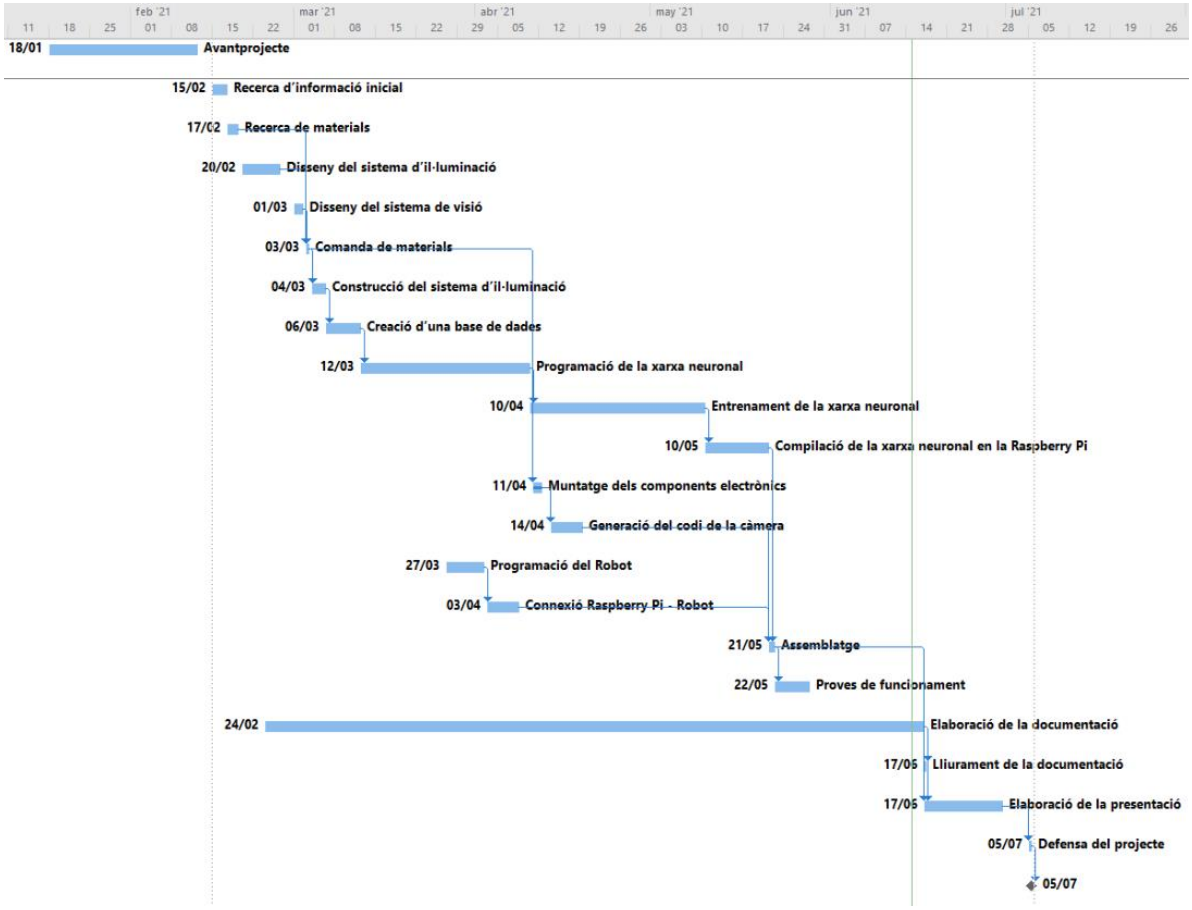

Figura 8.1: Diagrama de Gantt.
Font: Elaboració pròpia.

Com es pot veure, el projecte té com a data d'inici el dia 15 de febrer, just després de l'entrega de l'avantprojecte i la data de finalització és el dia 8 de juny. Tot i que la data d'entrega és el dia 17 s'ha deixat un temps per a possibles imprevistos i demores sovint comuns en aquest tipus de projectes. Un cop s'ha acabat d'elaborar el projecte, es comença a definir la presentació, amb la defensa final del projecte el dia 5 de juliol.

9. Impacte mediambiental.

Per tal d'identificar l'impacte mediambiental que causa el projecte, s'han realitzat unes llistes de control disponibles als annexos. A continuació es mostraran unes taules que representen un resum confeccionat a partir de les llistes de control.

Accions		**Conclusions**
Fase de construcció	Acústiques	Sorolls produïts per la utilització de maquines i eines. Impacte lleu.
	Visuals	Impacte nul.
	Residus	Impacte lleu.
Fase de funcionament	Acústiques	Impacte nul.
	Visuals	Espai ocupat pel sistema de reconeixement, robot i sistema de control. Impacte lleu.
	Residus	Impacte nul.
Final de la vida útil	Acústiques	Impacte nul.
	Visuals	Impacte nul.
	Residus	Components electrònics i metalls com per exemple l'estructura del robot i els motors. Impacte lleu.

Taula 9.1: Resum de les taules d'impacte mediambiental.

Factor ambiental		**Impacte**
Medi natural	Atmosfera	Impacte nul.
	Sòl	Impacte nul.
	Aigua	Impacte nul.
	Flora	Impacte nul.
	Fauna	Impacte nul.
Medi socioeconòmic	Usos del territori	Impacte nul.
	Cultural	Impacte nul.
	Infraestructura	Impacte nul.
	Humans	Impacte nul.
	Economia i població	Impacte nul.

Taula 9.2: Resum de les taules d'impacte mediambiental.

10. Conclusions finals, dificultats i futures línies de treball.

10.1. Dificultats.

El present projecte presenta una sèrie de dificultats ja que tracta àrees no estudiades durant el grau en enginyeria electrònica i automàtica, però alhora són d'importància degut a la transformació industrial cap a màquines més intel·ligents i autònomes. El projecte tracta d'una manera amigable el funcionament d´una xarxa neuronal i com a partir de programes senzills es poden dur a terme aplicacions que anys enrere presentarien unes dificultats importants. D'aquesta manera, l'objectiu d'implementar reconeixement d'imatge a un robot és una eina per aconseguir entendre mitjançant la pràctica i l'aplicació de la teoria, el funcionament d'aquesta tecnologia.

Durant la realització del projecte s'han patit una sèrie d'entrebancs que han afectat al desenvolupament d'aquest. A continuació es farà una enumeració de les dificultats més importants patides en la realització del projecte:

1. **Informació referent a la Jetson Nano.**

La placa de desenvolupament Jetson Nano està dissenyada per a córrer intel·ligències artificials, però no es tracta d'un hardware molt popular, com podria ser una Raspberry Pi. La cerca d'informació es torna difícil si l'objectiu del projecte és entrenar una xarxa neuronal i no utilitzar una disponible. Finalment es va arribar a trobar la infirmació necessària per a dur a terme la configuració inicial i l'entrenament del model, tot i patir un retard important en l'elaboració de la memòria.

2. **RobotStudio.**

De manera inicial, s'havia pensat en programar el robot ABB utilitzant el software disponible als laboratoris del centre, ja que és la forma de programació que s'utilitza a l'assignatura de Robòtica. Aquesta manera es pot comprovar mitjançant una simulació que els moviments i la programació estiguin correctament.

A causa d'un problema de llicències de RobotStudio, no va ser possible utilitzar el software i es va haver de fer tota la programació utilitzant la consola integrada del robot. Aquest canvi ha suposat un aprenentatge del funcionament de la consola i per tant la necessitat de disposar del robot més temps del previst. Tot i aquest inconvenient, s'ha après a programar un robot d'una manera alternativa i sense necessitat d'un software, reduint el pressupost del projecte a causa de no necessitar llicències. A més, s'ha trobat que la programació mitjançant consola és més semblant a la programació d'altres fabricants de robot com podria ser Fanuc o Kuka, i per tant, és un coneixement que es podria aplicar en futurs projectes.

3. Ubuntu.

Ubuntu és un sistema operatiu amb moltes possibilitats que ofereix programari de codi obert. És molt utilitzat en aplicacions de recerca o servidors a causa de tenir una comunitat disposada a oferir millores del sistema. El problema d'utilitzar un sistema operatiu desconegut és el temps d'aprenentatge necessari per a poder utilitzar les seves funcionalitats i també conèixer les aplicacions necessàries per a dur a terme determinats processos. Gràcies al fet que disposa d'una comunitat molt extensa, no és difícil trobar guies i documentació per aprendre de manera més ràpida.

4. PyLogix.

PyLogix és una llibreria de Python que permet la comunicació mitjançant TCP/IP amb un PLC CompactLogix. Aquesta eina ha sigut essencial a l'hora d'implementar la comunicació amb el robot, facilitant l'escriptura de senyals de sortida en el PLC i el robot.

Aquesta llibreria pot presentar incompatibilitats a l'hora de ser instal·lada, com la primera vegada que es va utilitzar. El compilador de Python no reconeixia les instruccions de la llibreria tot i haver sigut instal·lada correctament. L'única manera mitjançant la qual es va aconseguir solucionar aquest problema va ser fent una instal·lació des de zero tal com s'indica al capítol ¨Preparació de l'entorn de programació¨.

5. Comunicacions

Un altre entrebanc important en la realització del projecte va ser en el moment d'intentar comunicar el PLC amb el robot. Un cop configurat PyLogix es va provar que es poguessin pilotar les entrades i sortides del PLC. Aquesta comunicació es va poder efectuar sense més

problemes que els esmentats anteriorment. El problema va sorgir en intentar activar les sortides del PLC que estaven cablejades directament al robot, que el robot no reconeixia cap entrada en estat encès, la qual cosa indicava un error en la comunicació entre PLC-Robot.

El problema residia en què la documentació sobre les connexions disponible estava desactualitzada i havia canviat el mòdul de sortides del PLC que comunicava amb el robot. Inicialment, aquest mòdul era el número 6, mentre que el mòdul cablejar era el 4. També va passar el mateix amb el mòdul d'entrades, canviant del 3 a l'1.

Un cop feta la modificació pertinent als programes, es va poder provar el sistema sense més complicacions.

10.2. Futures línies de treball.

El present projecte presenta moltes vies de millora, sent les possibilitats gairebé infinites. Com es tracta d'equips independents i programables, es podria afegir més equips dins del sistema com per exemple un segon robot o una pantalla per controlar un SCADA. Com la llibreria PyLogix permet l'escriptura i lectura de variables, es podria seleccionar en una pantalla quina peça es vol reconèixer.

El model de xarxa neuronal entrenat també pot ser modificat amb peces totalment diferents amb un entrenament relativament curt si es disposa de la base de dades creada. Per exemple es podria utilitzar com a substitució de l'actual classificació de peces rodones mitjançant sensors i fer-ho utilitzant una tecnologia més actual.

10.3. Conclusions.

Aquest projecte és una mostra dels avenços dins del camp de l'industria 4.0. Les intel·ligències artificials són cada cop més presents en el món industrial i ajuden a millorar diferents processos, com en aquest cas, la classificació o inspecció visual. La creació d'una xarxa neuronal nova és un procés llarg i difícil, però per a la majoria d'aplicacions no és necessari i és suficient amb el reentrenament d'un model creat. Utilitzant tècniques de Transefer Learning i amb l'ajuda d'un microcontrolador, s'ha pogut crear un sistema funcional de classificació de peces, aprenent el funcionament de les xarxes neuronals i com es poden integrar en sistemes convencionals com poden ser robots i PLCs.

11. Referències.

[1] https://dlc.iec.cat/Results?DecEntradaText=robot, Diccionari de l'Institut d'Estudis Catalans (DIEC2), Gener 2021.

[2] https://definicion.de/robot, *Definición de robot,* Julián Pérez Porto y María Merino. Gener 2021.

[3] https://es.wikipedia.org/wiki/Robot, *Robot.* Gener 2021.

[4] https://iertec.com/aplicaciones-de-la-robotica-al-campo-de-la-medicina, *Aplicaciones de la robotica al campo de la medicina.* Gener 2021.

[5] https://es.wikipedia.org/wiki/Robot_de_servicio, *Robot de Servicio,* Wikipèdia. Gener 2021.

[6] https://www.ulmahandling.com/es/intralogistica-automatizada/sistema-agv-lgv, *Automated Guided Vehicle,* ULMA. Gener 2021.

[7] Josep López Xarbau, *Introducció als robots*. Octubre 2019.

[8] https://www.expansion.com/economia-digital/innovacion, Expansión. Gener 2021.

[9] https://www.avansis.es/industria-4-0/robotica-industrial, Avansis. Gener 2021.

[10] http://www.udesantiagovirtual.cl, *Estructura de los robots*. Universidad de Santiago de Chile. Gener del 2021.

[11] www.itca.edu.sv, *Diseño y construcción de un prototipo de robot con tres grados de libertad para posicionamiento de objetos*, ITCA-FAPADE. Gener 2018.

[12] clr.es, *Actuadores eléctricos en robòtica*. CLR. Gener de 2021.

[13] uniovi.es, *Elementos terminales.* ISA ingeniería. Gener 2021.

[14] edimar.com/vision-artificial-en-la-industria/, *Visión Artificial en la industria.* Edimar. Febrer 2021.

[15] artediez.es, *Digitalización de imágenes.* Març 2021.

[16] Jordi Torres, *Python Deep Learning.* Marcombo. 2020.

[17] codebots.com/artificial-intelligence, *What aret he 3 types of AI? A guide to narrow, general and super artificial intelligence,* Codebots. Febrer 2021.

[18] scielo.conicyt.cl, *Redes neuronales artificales.* Febrer 2021.

[19] vincentblog.xyz, *Redes neuronales convolucionales,* Vicente R. Novembre 2018.

[20] www.deeplearing.ai, *Activation funcions in Deep Learning.* Maig 2021.

[21] mèdium.com, *A brief guide to CNN.* Maig 2021.

[22] towardsdatascience.com, *Deep Learning: GoogLeNet Explained.* Maig 2021.