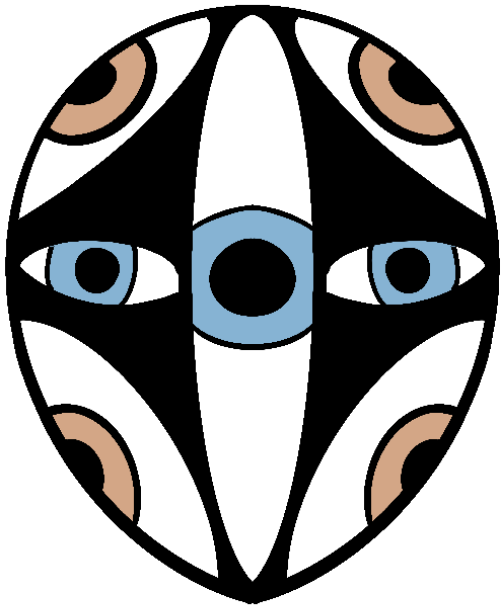
TREBALL FINAL DE CARRERA

SISTEMA PER REDUIR EL RISC D'ACCIDENTS A LES CANTONADES (ARGOS[1] CAM)



ARGOS

Nom: Ramon Angosto Artigues

Assignatura: Treball de final de grau

Grau en enginyeria mecatrònica.

Professor/a: Laura Dempere-Marco

6 juny de 2021

ÍNDEX DE CONTINGUT

ÍNDEX D'IL·LUSTRACIONS

ÍNDEX DE TAULES

1. ABSTRACT

1.1 CATALÀ

En el següent Treball de Final de Grau, s'exposa el projecte dut a terme per al desenvolupament d'un dispositiu que té per objectiu augmentar la visibilitat dels vehicles a les cantonades per tal d'evitar accidents. Concretament al treball s'explicaran els mètodes que s'usen per a la identificació de possibles fonts de perill, la integració al vehicle del dispositiu i el mètode de notificació al conductor.

El projecte té dues línies metodològiques principals: 1) adquisició, processament d'imatges i reconeixement d'objectes, i 2) disseny del dispositiu electrònic. Aquestes dues línies de treball s'integren en un dispositiu funcional que un cop processades les imatges, juntament amb l'anàlisi de l'estat de moviment del vehicle, emet avisos sonors i lluminosos que indiquen la presència de vianants o vehicles que s'aproximen pels laterals.

Per dur a terme això, en primer lloc, s'han adquirit una sèrie de vídeos que han estat usats per a la realització de les proves, de forma prèvia a la integració al vehicle. Quant a la programació, aquesta s'ha dut a terme amb una Jeston Nano i el llenguatge Python junt amb una sèrie de llibreries *open source,* entre les quals destaca OpenCV. El disseny del dispositiu s'ha dut a terme amb el programa de disseny 3D Fusion360.

Així doncs, l'objectiu principal del projecte és augmentar la visibilitat del conductor quan es troba a una cantonada intentant girar. El dispositiu construït permet incrementar la visibilitat del conductor mitjançant l'ús de dues càmeres situades a la part del davant del cotxe i indicarà la presència d'objectes perillosos.

Els resultats obtinguts són prometedors. Per una banda, mostrar la imatge de la part frontal del cotxe ja suposa un gran avantatge de visibilitat. A més, als vídeos usats per realitzar les proves, en dies solejats o ennuvolats i les proves realitzades fora del cotxe, s'han obtingut resultats satisfactoris ja que s'avisava correctament al conductor quan un vehicle o vianant s'estava aproximant. En canvi, en condicions adverses, com pluja i boira, s'han observat alguns problemes de reconeixement dels vehicles i persones. També, a causa de la limitació de potencia al connectar la placa a una bateria portàtil per incorporar-la al cotxe aquesta no ha tingut el funcionament esperat i no ha donat els avisos correctament.

En definitiva, el projecte ha mostrat resultats que fan pensar que aquest dispositiu, amb una mica més de desenvolupament, podria ser incorporat com un sistema més d'assistència a la conducció.

1.2 ANGLÈS

In the following Final Degree Project, the project carried out for the development of a device is exposed that aims to increase the visibility of vehicles in the corners in order to avoid accidents. Specifically, the work will explain the methods used to identify possible hazard sources, the integration into the device's vehicle and the driver notification method.

The project has two main methodological lines: 1) acquisition, image processing and object recognition, and 2) design of the electronic device. These two lines of work are integrated into a functional device that, together with the analysis of the vehicle's state of movement, emits sound and luminous warnings indicating the presence of pedestrians or vehicles approaching the sides.

To carry out this, in the first place, a series of videos have been acquired that have been used for the realization of the tests, prior to the integration into the vehicle. As for the programming, this has been carried out with a Jeston Nano and the Python language together with a series of *open-source libraries,* among which OpenCV stands out. The device's design was carried out with the Fusion360 3D design program.

Therefore, the main objective of the project is to increase the visibility of the driver when he is in a corner trying to turn. The built device allows to increase the visibility of the driver by using two cameras located in the front of the car and will indicate the presence of dangerous objects.

The results obtained are promising. On the one hand, showing the image of the front of the car already represents a great advantage of visibility. In addition, in the videos used to carry out the tests, on sunny or cloudy days and the tests carried out outside the car, satisfactory results have been obtained since the driver was correctly notified when a vehicle or pedestrian was approaching. On the other hand, in adverse conditions, such as rain and fog, some problems of recognition of vehicles and people have been observed. Also, due to the power limitation when connecting the board to a portable battery to incorporate it into the car it has not had the expected operation and has not given the warnings correctly.

In short, the project has shown results that suggest that this device, with a little more development, could be incorporated as another driving assistance system.

2. INTRODUCCIÓ

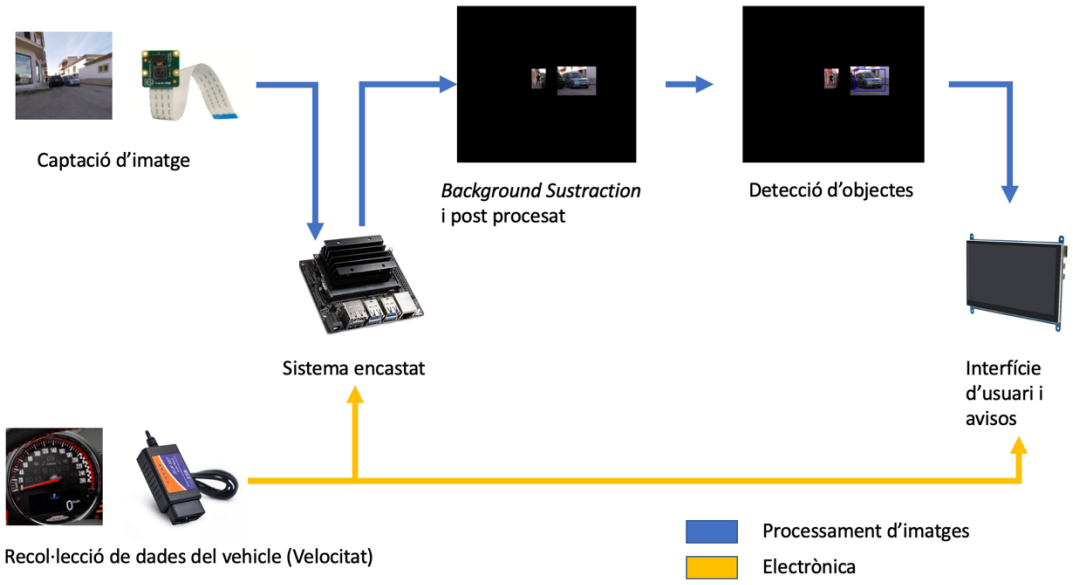
En aquest projecte de final de carrera es descriu el desenvolupament i prototipat d'un sistema per augmentar la visibilitat dels conductors a les cantonades per tal de reduir el nombre d'accidents que es produeixen en aquestes zones.

La visibilitat reduïda augmenta el risc de col·lisió amb vehicles que s'aproximin pel carrer perpendicular al que s'està circulant. Sobretot si el cotxe ha d'entrar a la via que es vol incorporar per tenir visibilitat. Aquest problema es dona sobretot a zones rurals on els carrers són més estrets i a les sortides dels pàrquings on molts de cops el cotxe ha de travessar l'acera per incorporar-se al carrer, havent-hi el risc d'atropellar algun vianant.

Segons la DGT el nombre d'accidents anuals en aquestes interseccions és d'aproximadament 12.000 en vies interurbanes i de 55.000 en vies urbanes, aquesta informació pertany al grup 7 de la DGT, on s'especifiquen el nombre d'accidents segons el tipus de via [2].

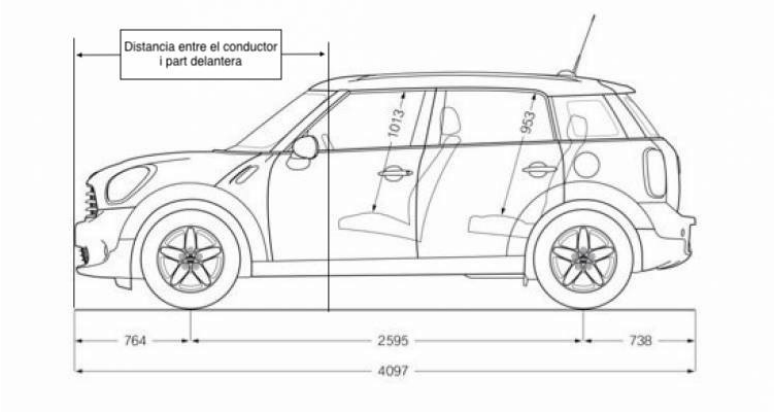
En el projecte es desenvolupa un sistema que serà anomenat ARGOS CAM, fent referència a la mitologia grega a Argos Panoptes [1], que era un guardià i servent d'Hera que tenia molts ulls, la qual cosa el feia molt eficient per la vigilància perquè sempre en tenia algun d'obert per vigilar. El logo del projecte fa referència a aquest mite.

ARGOS Cam consta de dues parts principals: una que s'encarrega de la recollida de dades del vehicle (i.e. velocitat) i una segona part que s'encarrega de la captació d'imatges. Aquest segon mòdul –centrat en el processat d'imatges- és l'encarregat d'oferir l'ajuda al conductor, mitjançant reconeixement d'imatge, i es troba modulat pel primer, que s'enfoca més en aspectes de l'electrònica de vehicles. L'estructura d'ARGOS Cam es presenta en la figura 1, que al seu torn recull la pròpia estructura del treball :



Il·lustració 1: *Pipeline* del projecte

A la següent figura 2 es pot veure una imatge que representa l'avantatge que suposa el sistema per al conductor, si es col·loquen dues càmeres a la part davantera del cotxe.



Il·lustració 2: Distància entre el conductor i la part davantera del cotxe

Val a dir que, a més de l'ajuda visual intrínseca que suposaran les càmeres, aquestes usaran algoritmes de processament d'imatge per poder notificar al conductor si apareix algun element al qual és necessari que presti atenció, com podria ser un cotxe que s'està aproximant o un vianant que està travessant el carrer.

3. MOTIVACIÓ

La idea del projecte surt de la identificació d'un problema que he pogut veure conduint, i com ja s'ha comentat a la introducció, és la dificultat per veure si s'aproximen altres vehicles a les cantonades. És cert que en algunes cantonades hi han miralls que pretenen solucionar aquests problemes, però no hi són sempre i sovint estan en mal estat, la qual cosa en limita el seu ús.

Després d'identificar el problema he realitzat una recerca per veure el nombre d'accidents en aquestes interseccions, i l'alarmant nombre de 67.000 accidents anuals d'aquest tipus ha passat a ser l'altre motiu que m'ha motivat a dur a terme aquest projecte, per intentar reduir aquest nombre d'accidents i fer la vida de les persones més fàcil.

A més d'intentar fer la vida més senzilla a les persones, el fet de treballar amb algoritmes de *Deep Learnig* suposa un repte que m'estimula i em permet aplicar els coneixements adquirits al llarg dels estudis de mecatrònica i m'ofereix una oportunitat per aprofundir en coneixements que poden ser útils per un futur lloc de treball en l'entorn de la programació.

4. OBJECTIUS

Com s'ha comentat a la motivació d'aquest projecte, l'objectiu general és la reducció del nombre d'accidents que es produeixen a les cantonades a causa de la mala visibilitat.

Concretament es vol fer un dispositiu que es pugui integrar al cotxe, que permeti la visualització del carrer al qual el conductor es vol incorporar sense haver de treure la part davantera del cotxe i que es notifiqui al conductor si s'aproxima un altre vehicle o hi ha un vianant creuant o amb intenció de fer-ho.

Així mateix el projecte consta de dos objectius específics. En primer lloc la recol·lecció d'imatges i dades obtingudes del vehicle, junt amb algoritmes d'intel·ligència artificial per assistir al conductor en aquests tipus de vies. En segon lloc el disseny d'una interfície gràfica i un sistema d'alertes per incorporar al vehicle.

Els objectius comentats es poden ordenar per orde de prioritat de la següent forma:

1. Reduir el nombre d'accidents en aquest tipus de vies a conseqüència de la mala visibilitat

2. Fer ús de càmeres i informació proporcionada pels sensors del cotxe, juntament amb algoritmes d'intel·ligència artificial per assistir en la conducció.
3. Dissenyar un dispositiu amb sistema d'alertes per a la incorporació al vehicle

5. ESTAT DE L'ART

En aquest punt es tractarà l'estat actual de la literatura que existeix sobre l'ús de dispositius per a l'assistència en conducció i cada un dels punts esmentats a la introducció que conformen el projecte.

D'entrada els sistemes d'assistència avançada al conductor, també coneguts com a ADAS [3], existeixen al sector automobilístic des del 1950, fent que la conducció sigui més segura. Alguns exemples d'aquests primers usos varen ser: sensors d'excés de velocitat i el sistema de frenada ABS. Actualment aquests sistemes es troben integrats a tots els vehicles, i la Comissió Europea planteja fer obligatòria la incorporació de més sistemes d'assistència per reduir el nombre d'accidents.

En l'actualitat han sorgit molts més d'aquests sistemes que ajuden a evitar accidents. Aquests, a més de l'assistència en carretera, també es troben relacionats amb l'ajuda al conductor dins de les ciutats. Els sistemes de frenada automàtica per proximitat, amb sensors d'ultrasons i LIDARS (*Light Detection And Ranging*[1]), i els sistemes d'ajuda a l'aparcament -amb càmeres i sensors de proximitat- són dos dels exemples més clars en aquest cas d'ús. La companyia Nissan l'any 2007 va presentar un concepte de cotxe que tenia incorporades 4 càmeres al voltant del cotxe que permetien simular una vista de dron del mateix per assistir a l'aparcament, a més de permetre seleccionar les càmeres que es veien a la pantalla [4]. També incorporava sistemes d'ultrasons per alertar d'objectes propers.

Els cotxes autònoms, principalment funcionen amb LIDARS, càmeres o la combinació d'aquestes dues tecnologies (o similars) [5]. Aquesta combinació es coneix com a *sensor fusion* i permet evitar alguns dels problemes que presenta cada tecnologia.

En aquest treball s'ha optat per una aproximació similar a la que va prendre Nissan i la que usen els cotxes autònoms que fan servir càmeres i sistemes d'intel·ligència artificial.

[1] LIDAR: Sensor que permet determinar la distancia a un objecte o superfície mitjançant un làser polsant. [16]

5.1 RECOL·LECCIÓ DADES DEL VEHICLE

Pel que fa a la recol·lecció de dades del vehicle, l'any 1989 es va introduir el port ODB I, als automòbils americans, que permetia controlar l'estat del motor mitjançant uns codis d'error. El 1996 es presenta el port OBD II que a més de permetre la visualització de codis d'errors permet la lectura de dades del vehicle com poden ser la velocitat, les revolucions i altres paràmetres no relacionats amb el motor. Des del 2003 aquest port és obligatori a tots els vehicles. [6, 7]

5.2 CAPTURA D'IMATGES

Un altre aspecte cabdal del sistema desenvolupat és la captura d'imatges. En aquest sentit, existeixen dos tipus principals de connexions, usats en sistemes encastats per treballs de conducció autònoma i per a la integració en sistemes encastats. Aquestes dos tipus són: MIPI CSI-2 i USB 3.0

Per un costat la connexió MIPI CSI-2 és la més usada en sistemes encastats i en telefonia mòbil. Aquesta permet una transmissió d'imatges més ràpida, 6 GB/s d'ampla de banda, i més eficient que l'USB 3.0, pel fet que no ocupa part de la CPU per funcionar. A més solen tenir una mida més reduïda tant el cos de la càmera com el cable. Per contra, presenta alguns desavantatges com són la necessitat de llibreries especials i la dificultat per usar diferents sensors a les càmeres [8].

Per l'altre costat, la connexió USB 3.0 arriba fins a un ampla de banda de 5 Gb/s i únicament s'ha de connectar per a funcionar. Com que el tipus de connexió és més comú és més fàcil trobar un dispositiu compatible o canviar un dispositiu trencat o millorar el sistema [8], [9].

A més de la connexió hi ha altres paràmetres que influeixen en la captura d'imatges, aquest són si la càmera és monocromàtica o amb un sensor a color, si està equipada amb un disparador de sensor complet o rodant, si té autoenfocament o no, si està equipada amb capacitats de visió nocturna, així com altres paràmetres que no són rellevants per aquest projecte [9].

Atès que en aquest projecte s'usarà un sistema encastat, tot tenint en compte allò que s'ha comentat en aquest apartat, s'ha optat per seleccionar una càmera amb la connexió MIPI CSI-2, sensor a color sense visió nocturna ni autoenfocament- ja que es pot produir un

moviment molt ràpid davant de la càmera i l'autoenfocament podria ser més lent que el moviment que produït.

5.3 SISTEMA ENCASTAT

Els sistemes encastats són sistemes informàtics de propòsit especial dissenyats per dur a terme una o poques tasques dedicades, i que normalment es troben limitades a la computació en temps real. Generalment s'incrusta com a part d'un dispositiu complet, sent la unitat de processament del dispositiu complet. Aquests sistemes encastats es troben a la majoria de dispositius de consum que s'usen actualmnent [10].

Avui en dia al mercat existeix una gran varietat de sistemes encastats, però a la majoria dels projectes de prototipat a cost reduït la decisió queda reduïda a dues: la Jetson Nano [11] i la Raspberry [12]. Aquestes dues plaques tenen la capacitat de fer anar un sistema operatiu i actuar com a un ordinador de petites capacitats.

Per una banda, la Raspberry va ser creada el 2012 per a ensenyar programació i com interactuar amb dispositius electrònics. Té una memòria RAM que oscil·la entre 1 GB i 8 GB, permetent l'execució de més o menys tasques paral·leles, i una CPU de 4 nuclis. Per l'altre banda, la Jetson Nano és una placa dissenyada per la companyia NVIDIA, pensada per a l'execució de programes de *Machine Learning* i processament d'imatge. Aquesta compta amb dues versions. Una de 2 GB de RAM i una de 4 GB, la de 4 tenint l'avantatge que compta amb dos ports MIPI CSI, per a connectar dues càmeres en paral·lel. A més a més té una GPU, aquesta aporta una gran capacitat per al computat paral·lel de tasques complexes com són el Processament d'Imatge i el *Machine Learning*.

A raó de què el projecte requereix dues càmeres i principalment es basa en el processament d'imatge es farà ús del sistema encastat Jetson Nano.

5.4 ALGORITMES DE VISIÓ PER COMPUTADOR PER L'ASSISTÈNCIA A LA CONDUCCIÓ EN CANTONADES

En aquest apartat es revisen els algoritmes principals de visió per computador que s'han desenvolupat per resoldre les tasques que ens ocupen en aquest projecte: *Background subtraction* i detecció d'objectes.

5.4.1 BACKGROUND SUBTRACTION

El *Background Substraction* és una tècnica que consisteix en la separació del fons (*background*) d'aquells objectes que es troben en moviment en un vídeo. Una de les funcions en què és més usada és en la videovigilància, pel fet que permet una detecció més simplificada dels objectes que es troben en moviment. El *Background Substraction* consta d'una sèrie d'etapes. Aquestes són: 1) la inicialització del fons (on en sol agafar una primera imatge), 2) la inicialització del model (que s'usarà per a la definició del que pertany al fons i el que es troba en moviment), 3) el mecanisme per actualitzar el fons, i 4) la classificació de les imatges que es van introduint [13].

Tradicionalment, els mètodes més usats per aquesta tasca són els basats en *Gaussian Mixture Models (MoG)* [13], ja que aquests mètodes tenen en compte canvis a la il·luminació, soroll de la imatge i objectes amb un desplaçament més lent.

A més d'aquest també existeixen altres models per a la detecció del fons que han aparegut als darrers anys. Alguns d'aquests es basen en algoritmes per segmentar el fons mitjançant xarxes neuronals i segmentació semàntica, la qual assigna cada píxel de la imatge a una classe [12, 13]. També hi ha investigadors que s'han centrat en l'ús d'altres tècniques basades en el Machine Learning com és el *K-Nearest Neighbours* [13].

Prèviament a la decisió final d'usar l'algoritme basat en K-Nearest Neighbour (K-NN), s'ha realitzat una comparació dels diferents algoritmes. En aquest treball, per limitacions en l'extensió no es mostra una anàlisi exhaustiva de tots els resultats obtinguts però si que es presenten en l'apartat de discussió les principals conclusions extretes d'aquest estudi.

5.4.2 DETECCIÓ D'OBJECTES

La detecció d'objectes consisteix en la combinació de dues tasques, la classificació d'un objecte que apareix a la imatge i la localització d'aquest, en altres paraules, dir en quina posició es troba un o més objectes a la imatge [16]. En l'àmbit de la visió per computador s'han desenvolupat distints mètodes per realitzar aquesta tasca, tant des de la vessant de la visió per computador clàssica com la basada en *Deep Learning*. Atesos que en aquest projecte ens centrarem en aquesta segona aproximació, revisem en aquesta secció els principals mètodes desenvolupats en l'àmbit de la detecció d'objectes.

Cal, però, primer fixar alguns conceptes i introduir què és el *Deep Learning*, així com introduir la noció de *Transfer learning*. El *Deep Learning* consisteix en l'ús d'estructures organitzades

que emulen el processament d'informació que té lloc en el sistema nerviós. Aquesta estructura compta amb unitats de processament de dades, anomenades *neurones* connectades entre elles a través de *sinapsis* caracteritzades per pesos que l'algoritme aprèn a partir de les dades. El processament d'imatges és una àrea que ha estat molt beneficiada per aquestes tècniques de processament de dades. La companyia NVIDIA ha contribuït de forma substancial amb la creació de la llibreria CUDNN, que permet usar la GPU dels ordinadors per accelerar aquest procés [17]. Aquest mètode d'aproximació al sistema nerviós permet imitar-lo creant àrees específiques per a dur a terme tasques concretes, com pot ser l'extracció de característiques d'una imatge [18].

Per un altre costat el *Transfer Learning* consisteix a usar els coneixements adquirits en tasques prèvies per a entrenar una nova tasca [19]. Aquest és una forma més semblant a la manera de transmetre coneixement que tenim els humans, en comparació als mètodes tradicionals d'aprenentatge que es troben en les aproximacions clàssiques de *Machine Learning*. Per exemple, si a una primera tasca s'ha entrenat una xarxa neuronal per a reconèixer plantes, i més endavant es vol entrenar una xarxa diferent per a reconèixer arbres, part del coneixement de la primera xarxa es podrà reutilitzar per a la segona, sent necessària menys informació per a entrenar la segona.

Per dur a terme aquesta tasca existeixen diversos algoritmes de *Deep Learning*. En revisem a continuació alguns d'ells:

- **Models R-CNN** : el model va ser presentat al 2014 per Ross Girshick, i es basa en xarxes neuronals convolucionals. La xarxa es divideix en tres etapes: proposa una regió d'interès, s'extreuen les característiques d'aquesta i finalment es classifica [20].

- **Models Fast R-CNN**: es basen en el model previ i varen ser millorats pel grup de recerca de Microsoft, a diferència de l'anterior, se simplifica l'etapa de proposta de regions d'interès i combina les dues darreres etapes, permeten accelerar el procés de reconeixement [21].

- **YOLO (You only look once)**: es tracta d'una xarxa que té com a entrada una imatge, i dóna com a resultat una *bounding box* i una classificació. Aquest tècnica és menys precisa que els models Fast R-CNN, però ofereix una major velocitat de processament. El funcionament d'aquest algoritme es basa en la divisió de la imatge, en quadrícules, i es proposen diferents *bounding boxes*, aquestes posteriorment es determinaran si pertanyen a una classe depenent de la confiança que tenen assignades [22].

- **Mètodes *Single Shot detector* (SSD**): Estan dissenyats per a la detecció d'objectes en temps real. Són capaços d'igualar la precisió de les xarxes Fast R-CNN, amb imatge de baixa qualitat. Aconsegueixen augmentar la velocitat de processat eliminant la proposta de regions i incrementen la precisió fent ús de l'extracció de característiques a múltiples escales. Aquest mètode es compon de dues parts un extractor de característiques basat en una xarxa neuronal i una sèrie de filtres convolucionals per a la detecció d'objectes [23], [24].

En el context d'aquest projecte, la detecció d'objectes és la part més important ja que és l'encarregada de donar l'avís al conductor de què està passant. En la secció de Metodologia s'explicarà el model que s'ha seleccionat.

6. METODOLOGIA

6.1 RECOL·LECCIÓ DADES DEL VEHICLE

L'única dada que es recull del vehicle serà la velocitat. Aquesta serveix per activar i desactivar la pantalla perquè aquesta no distregui al conductor mentre està en moviment. S'ha seleccionat 2 km/h com el punt al qual s'activa la pantalla. En altres paraules, la pantalla estarà activa sempre que el vehicle es desplaci a una velocitat inferior a 2 km/h o estigui parat. El petit marge fins a zero es dóna perquè la pantalla no s'estigui activant i desactivant cada cop que el vehicle avanci uns centímetres si està parat al darrere d'un altre cotxe esperant que sigui el seu torn.

Com ja s'ha mencionat a l'estat de l'art, el mètode que s'usarà per a la recol·lecció de dades serà el port OBD II. Prèviament a la decisió d'usar el port OBD s'han plantejat altres alternatives, com són l'ús de GPS o la lectura directa del tacòmetre a través d'un cable. La primera opció ha estat descartada pel fet que existeix un retard elevat entre la lectura del GPS i la velocitat del cotxe, sobretot en velocitats baixes. El segon mètode s'ha descartat per no fer malbé el cotxe que s'usarà per fer les proves, malgrat ser el mètode més fiable.

El port OBD funciona de la següent manera: a la part inferior del volant se sol trobar el punt de connexió. Aquest recull les dades de tots els sensors de cotxe i els monitoritza constantment. Un cop es connecta el lector, que pot ser Bluetooth, per cable o amb WIFI, es transmeten les dades del vehicle al dispositiu usat per la lectura de dades. En aquest projecte s'usarà un lector amb comunicació per cable USB, com el de la imatge inferior.

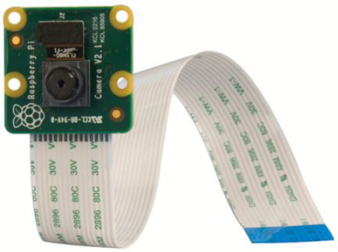
Il·lustració 3: Lector OBD II amb cable (Imatge obtinguda de [25]**)**

Les dades que es transmeten del sensor es troben en forma de codi de 5 dígits. El primer dígit indica de quina part del cotxe es tracta la lectura, el segon l'organització que ha definit el codi, el tercer especifica una funció concreta del cotxe i els dos darrers són per indicar errors [7].

Per tal de simplificar la lectura d'aquests codis s'usarà la llibreria Python-OBD [26], aquesta du incorporades funcions que permeten llegir els codis i proporcionen directament els valors d'aquests. Un cop connectat el lector, s'estableix la connexió i es declara una funció per monitoritzar el valor desitjat, en aquest cas la velocitat.

6.2 CAPTURA D'IMATGES

Qualsevol sistema de visió per computador té una primera fase d'adquisició de la imatge . Per aquesta tasca s'usaran dues càmeres Raspberry piCameras V2 de 8 Megapixels. Aquestes van col·locades a la part frontal del cotxe, a la imatge 5 es pot veure el punt on van col·locades.



Il·lustració 4: Raspberry piCamera V2 (Imatge obtinguda de [27]**)**

Il·lustració 5: Posició de la càmera al cotxe

Per obtenir les imatges de les càmeres es farà ús de la llibreria OpenCV, de la que es donaran més detalls més endavant, i el *framework* multimèdia GStreamer que permet ajustar el format de sortida de la imatge, ajustar les dimensions de la imatge capturada i la freqüència de captura (*framerate)*.

El *framerate* és un paràmetre molt important en aquest projecte ja que el sistema ha de ser capaç de captar els cotxes en moviment. En principi el límit de velocitat urbà és de 30 km/h, però com a marge de seguretat es posarà que s'ha de detectar fins a 60 km/h (17.7 ms/s) en cas que algun vehicle vagi a una velocitat superior a la permesa.

Aquesta velocitat permet determinar el mínim *framerate* necessari per poder captar cotxes a 60 km/h. D'aquesta manera, si es prenen 18 imatges per segon, el cotxe s'hauria desplaçat aproximadament 1 metre entre cada imatge. A mesura que s'augmenta el *framerate* la distància que es desplaça el cotxe per cada *frame* es redueix. En el nostre cas s'estableix el *framerate* a 60 FPS[2], per tal de permetre un desplaçament màxim del cotxe entre captures de 30 cm.

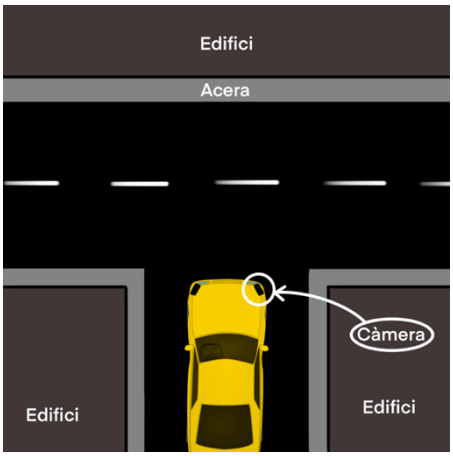
Per tant, la càmera queda ajustada amb els següents paràmetres:

Mida del marc de captura: 1280 x 720 píxels

Framerate: 60 FPS

Val anticipar que els 60 FPS no s'acabaran donant ja que els algoritmes que s'expliquen més endavant faran que el procés de captura vagi més lent.

A la il·lustració 6 es mostra una representació de la posició des de la que es capturen les imatges pel projecte. Tal i com es pot apreciar, el conductor no envaeix la via.



Il·lustració 6: Posició del cotxe per la captura d 'imatges

2 FPS: *Frames* per segon

A continuació, es presenten una sèrie d'imatges preses des de la posició indicada, i a l'annex 1, es poden trobar més exemples.

Il·lustració 7: Exemples de captures d'imatges a diferents cantonades

6.2.1 ÈTICA O PROTECCIÓ DRETS D'IMATGE

Atès que en aquest projecte es capturen imatges, és necessari clarificar que cap de les imatges que usa el dispositiu pel processat és emmagatzemada, d'aquesta manera es manté la privacitat de les persones que puguin aparèixer en el moment que les càmeres estan funcionant.

L'únic moment en el qual s'han emmagatzemat imatges ha estat en el moment en què es preparava un conjunt de dades de mostra per poder realitzar les proves, i en aquests, únicament apareixen persones que han donat el seu consentiment per aparèixer als vídeos.

6.3 SISTEMA ENCASTAT

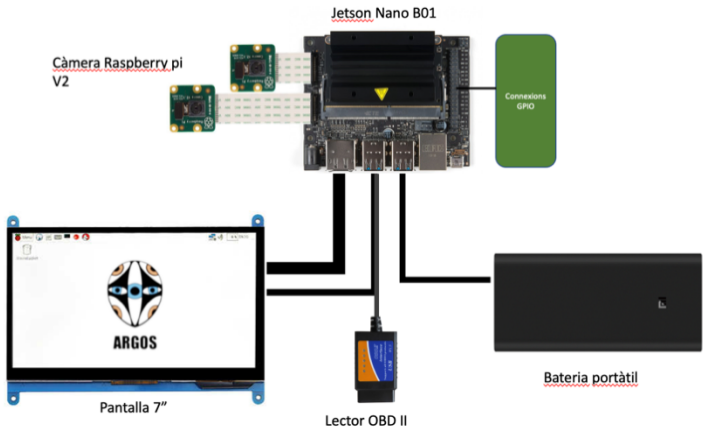
Tal i com s'ha indicat anteriorment, el sistema encastat que s'usarà en el projecte és una Jetson Nano B01 de la marca Nvidia. Aquesta placa és el lloc on es donarà tot el processament de les dades. El sistema operatiu que té instal·lat és el Jetpack 4.5, que està basat en Linux 18.0. El llenguatge de programació que s'usarà per dur a terme el processament de les dades i els avisos al conductor serà Python 3.7.

El programa que controla el projecte fa ús de les següents llibreries:

- **Python-OBD**: Com s'ha comentat a l'apartat de la lectura de dades del vehicle aquesta llibreria, permet la comunicació amb el lector OBD, i permet obtenir la velocitat del cotxe per poder usar-la i regular si la pantalla està encesa o no.
- **OpenCV (amb CUDA)**: És la llibreria que s'encarrega de tot el processament d'imatge i de la captura del vídeo. És dins d'aquesta que es troben els algoritmes de *Background Substraction* que s'expliquen al següent apartat. A més, aquesta llibreria juntament amb la llibreria *Tensorflow* és l'encarregada de la detecció d'objectes. Com que el sistema encastat compta amb una GPU que pertany a Nvidia es pot beneficiar de la llibreria CUDA, que ofereix un processament d'imatge accelerat.
- **Tensorflow**: es una llibreria de codi obert desenvolupada per Google, que s'usa per a l'entrenament i execució d'algoritmes de *Deep Learning*.
- ***Jetson inference*:** la funció d'aquesta llibreria és l'optimització de xarxes neuronals per tal que es puguin fer servir a la Jetson Nano.
- **PySimpleGUI**: aquesta llibreria és usada per crear la interfície d'usuari perquè aquest interactuï amb la pantalla i pugui veure les càmeres.
- **Jetson.GPIO**: permet controlar els pins GPIO de la Jetson, permetent activar els LEDs i avisos sonors que s'usaran per notificar a l'usuari de la presència d'algun objecte reconegut.

El codi escrit per al projecte es troba inclòs als annexos, aquest està format per dos *scripts*. El primer, inclou una classe que s'ha creat per gestionar els algoritmes de processament d'imatges. Inclou els algoritmes de *background substraction* i de reconeixement d'imatge. El segon script, serveix per englobar tot el codi, i permet la lectura del OBD.També gestiona la GUI i es comunica amb el primer *script* per obtenir els resultats dels processament d'imatge.

En la següent figura es mostra l'esquema de connexions, per entendre com es comunica cada element amb la Jeston Nano.



Il·lustració 8: Representació de les connexions dels components amb la Jetson Nano. Els materials usats per al projecte són: dues càmeres Raspberry Pi v2, usades per a la captura d'imatges; una pantalla tàctil per a la visualització d'imatges i interactuar amb la interfície d'usuari; un lector OBD usat per la recol·lecció de dades del vehicle; una bateria portàtil de la marca Xiaomi, amb capacitat per alimentar a 5v 2.6A; per a les connexions GPIO, s'usen quatre LEDS i un *buzzer* (amb els components addicionals necessaris)

Com es pot veure a la Figura 8, la placa és alimentada a través d'una bateria portàtil. Aquesta no és l'opció més recomanable i seria millor alimentar la placa directament amb la bateria del cotxe i un convertidor reductor de 12 V a 5V però a causa d'un problema amb la bateria del cotxe usat per les proves s'ha optat per aquesta solució.

6.4 BACKGROUND SUBTRACTION

Les imatges adquirides per les càmeres frontals són processades a través de l'algoritme de *Background Subtraction* per tal d'aïllar aquells objectes que es troben en moviment en l'escena. A continuació s'expliquen els criteris emprats per seleccionar l'algoritme amb què es treballarà i es detalla com funciona aquest.

6.4.1 SELECCIÓ DE L'ALGORITME

Per tal de seleccionar l'algoritme més adequat per aquest projecte s'han avaluat diferents opcions de *Background Substraction*. En particular, s'han considerat els

algoritmes: k-NN, MOG, MOG2 i GMG[3]. Per fer-ho s'han aplicat aquests algoritmes sobre una sèrie de vídeos, i s'han guardat les màscares resultants. Aquestes han estat comparades enfront del resultat esperat.

Aquests resultats esperats s'han obtingut fent un etiquetat de forma manual amb l'eina de disseny gràfic Photoshop. Amb aquesta s'ha creat una màscara representativa del *ground truth*, seguint la forma d'etiquetar usada per l'algoritme, dit d'un altre manera, s'ha fet un *bounding box*, al voltant de l'objecte que es troba en moviment tal com l'hauria de fer l'algoritme. El color negre pel fons, amb etiqueta 0, i el color blanc per l'element mòbil, amb etiqueta 1.

L'algoritme usat per aquesta valoració s'ha obtingut a partir de les següents referències [28], [29]. El rendiment de cadascú dels algoritmes ha estat avaluat mitjançant una matriu de confusió, que representa les quatre possibles combinacions entre els valors reals i els obtinguts a partir de la màscara binària generada per l'algoritme.

En el nostre cas, la classe fons, pertany al valor negatiu, és a dir, 0 i la classe en primer pla o objecte en moviment, seria el valor positiu, és a dir, 1.

La matriu de confusió es defineix a partir dels següents resultats de la classificació:

MATRIU DE CONFUSIÓ:

Positiu vertader (PV): Representa el nombre de prediccions positives, és a dir, nombre de píxels que han estat etiquetats amb la classe 1 i que realment pertanyen a aquesta classe.

Negatiu vertader (NV): Representa el nombre de prediccions negatives, és a dir, nombre de píxels que han estat etiquetats amb la classe 0 i que realment pertanyen a aquesta classe.

Fals positiu (FP): Representa el nombre de mostres que s'han classificat com a positives (primer pla) però en realitat són de negatives (fons).

[3] GMG: Algoritme està basat en l'article [42]. La base del seu funcionament es l'estimació d'imatge i l'aplicació per píxel de filtres Bayesians

Fals negatiu (FN): Representa el nombre de mostres que s'han classificat com a negatives (fons) però en realitat són de positives (primer pla).

MESURES DE RENDIMENT DE L'ALGORITME:

A partir d'aquest s'obtenen les següents mesures:

Precisió: És la ràtio de píxels classificats correctament en comparació a tots els píxels avaluats. Aquesta mesura és molt sensible per avaluar conjunts de dades en què les classes no estiguin balancejades. A l'equació 1 es pot veure com es calcula.

$$Precisió = \frac{PV+NV}{PV+NV+FP+FN} \qquad \text{(Eq. 1)}$$

Sensitivitat o rati de positius vertaders: Correspon a la proporció de mostres classificades com a positius vertaders (PV), respecte a tots els positius (PV + FP). A l'equació 2 es troba definit com fer aquesta mesura. Aquesta mesura pot ser interpretada en com de bé l'algoritme detecta els objectes que es troben en moviment.

$$Sensitivitat = \frac{PV}{PV+FP} \qquad \text{(Eq. 2)}$$

Especificitat o rati de fals positiu: En canvi l'especificat és la proporció de mostres negatives classificades com a positives (FP) respecte a totes les que realment són negatives (NV + FP) com es mostra a l'equació 3. Aquest valor s'interpreta com la capacitat de detectar correctament el que és fons de les imatges obtingudes.

$$Especificitat = \frac{FP}{NV+FP} \qquad \text{(Eq. 3)}$$

Valor F1: Es tracta d'una mesura que relaciona els valors de precisió i la sensitivitat. Aquesta mesura és bastant útil quan es té una gran desigualtat a les dades usades per a l'avaluació, i aquest és un dels casos en què és necessari usar-la, pel fet que la majoria dels píxels a les imatges pertanyen al fons. A l'equació següent es presenta com es faria el càlcul.

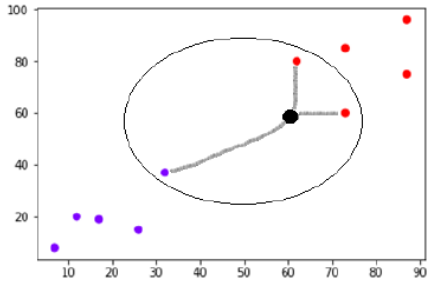
$$F1 = 2 * \frac{(Sensitivitat * Precisió)}{(Sensitivitat + Precisió)} \qquad \text{(Eq. 4)}$$

A l'apartat de resultats es presentaran els resultats obtinguts en aquest estudi. En la discussió s'analitzen aquests resultats i les seves implicacions.

6.4.2 KNN BACKGROUND SUBSTRACTION

L'algoritme seleccionat es basa en el *Gaussian Mixture Model* (GMM), i es tracta del *K-nearest neighbour (K-NN)*. Aquest es descriu en profunditat en [30]. A grans trets, es presenta un mètode recursiu per anar actualitzant els paràmetres del GMM, i actualitzar-los per a cada píxel. L'algoritme K-NN es troba inclòs a la llibreria OpenCv [31] i compta amb una sèrie de paràmetres que s'han d'ajustar per arribar al funcionament desitjat de l'algoritme. La inicialització del model es du a terme amb el nombre de *frames* assignats a la variable *NSamples*. Durant aquest període es defineix quins objectes pertanyen a la classe fons i quins a la classe d'objecte en moviment mitjançant la comparació dels píxels d'aquests frames. Llavors el model es va actualitzant cada *N* de *frames*, on N és el número *frames* que s'hagin definint al parametre *History*.

El *K-NN* és un algoritme principalment usat per problemes de classificació predictiva. Aquest algoritme no té una fase d'entrenament específica sinó que durant el mateix entrenament es du a terme una tasca de classificació. És un algoritme no paramètric, és a dir, que l'entrenament es du a terme sense informació de les dades entrades [32]. El principi de funcionament de l'algoritme es basa en un sistema de votació dut a terme entre els *K* veïns més propers i en la similitud de característiques -respecte a les dades d'entrenament- per predir la classificació de les noves dades. La següent imatge presenta un exemple senzill del funcionament de l'algoritme. El punt negre és la nova dada que s'està avaluant. Si K=3, hem de mirar els tres punts que estan més a prop. Com que la majoria d'aquests són vermells, el punt nou s'assigna a la classe vermella.



Il·lustració 9: Exemple K-NN obtingut de [32].

Els paràmetres necessaris per ajustar el funcionament de l'algoritme són:

History: Es tracta del número de *frames* que afectarà la modificació dels pessos assignats a la classe fons i objecte que està en moviment . Mitjançant aquest paràmetre es pot ajustar la referència sobre la qual es compararà el nou *frame* per comprovar si aquest ha canviat o no. Aquest valor s'ha establert a un valor de 100. Si el valor és massa petit els canvis en la imatge triguen massa temps a aparèixer reflectits, i si el valor és massa elevat, la màscara canvia en massa freqüència provocant que el mínim canvi en la il·luminació modifiqui la màscara de manera dràstica. El *framerate* de la càmera afectarà al valor d'aquest paràmetre, ja que, com major sigui aquest menys temps es trigarà a actualitzar.

Nsamples: Defineix el nombre de mostres guardades en memòria amb les quals compta l'algoritme de *Background Subtraction* per comparar i assignar una nova classe als píxels del nou *frame*. Si es compta amb un número de mostres molt reduïdes, la segmentació del fons té molt de soroll com es pot observar a la imatge inferior. En canvi amb el valor ajustat s'aconsegueix una màscara més uniforme. Per aquest projecte s'usa un valor de 10.

Il·lustració 10: A) Imatge Original B) NSamples amb valor 10 C) NSamples amb valor 3

KNNSamples: El paràmetre K-NN Samples, és la variable K que es troba sempre definida als algoritmes de K-NN. Amb aquest s'ajusta el nombre de mostres amb les que es compararà cada píxel per decidir si pertany a una classe o a l'altre, la que tingui més vots serà l'assignada. A l'article original d'aquest algoritme, es recomana usar un valor que es trobi entre 3 i 60. [30] A mesura que s'augmenta el valor de K la precisió amb la qual es detecta el fons també augmenta perquè el píxel actual es compara amb un major nombre de píxels propers per a ser classificat com a fons o com a objecte en moviment, però com a efecte negatiu d'aquest augment trobem que es redueix la velocitat a la qual s'obté la màscara. En aquest cas s'usarà un valor de 3, ja que un valor superior provoca que l'algoritme funcioni massa lent.

Detect shadows: Modificant el valor d'aquesta variable es pot seleccionar si es consideraran les ombres com a fons o com a objectes en moviment, depenent de si la variable està activa o no, aquesta es pot ajustar per descartar només ombres molt marcades o totes les que apareguin al *frame*. Per aquest projecte es detectaran tots els tipus d'ombra pel fet que qualsevol petit canvi en la imatge pot sevir per obtenir un reconeixement més ràpid de l'objecte.

Un cop aplicat l'algoritme de *Background Subtraction* es realitzaran un seguit d'operacions morfològiques sobre la imatge binària obtinguda per tal de facilitar la detecció d'objectes.

6.4.3 TRANSFORMACIONS MORFOLÒGIQUES

Les transformacions morfològiques són operacions basades en l'estructura geomètrica dels objectes de la imatge. Aquestes tenen com entrada una imatge, normalment binària, i un element estructural o *kernel*. Fent ús de la combinació d'aquests dos i de l'operació desitjada, la màscara es veu modificada [33]. En aquest treball es fa ús de dos *kernels* diferents, permetent aplicar-los en casos diferents. El primer té forma de creu i una mida de 5x5 píxels i el segon té forma d'el·lipse, amb una mida de 7x7 píxels. La forma el·líptica engloba més superfície, la qual cosa permet que la màscara s'amplïi de manera més significativa.

Il·lustració 11: Matrius que conformen els *kernels*

Les transformacions morfològiques usades es duen a terme en el següent ordre:

OPENING:

En primer lloc és durà a terme un ***Opening***, aquesta operació és la combinació de dues operacions, una erosió seguida d'una dilatació de la màscara. Freqüentment és usada per eliminar soroll a les màscares, per això s'ha usat en primer lloc. D'aquesta manera s'eliminen petits elements que poden haver estat detectats erròniament. Aquesta operació serà l'única que usarà el *kernel* amb forma de creu per tal d'evitar que la resta de la màscara es vegi massa modificada.

Il·lustració 12: Exemple de l'efecte de *Opening* (Font: Referència [33]**)**

CLOSING

Es tracta de l'operació contrària a l'*Opening*. En aquest cas la seva funció és tancar les petites obertures que poden quedar a la màscara.

Il·lustració 13: Exemple de la funció *Closing* (Font: Referència [33]**)**

DILATE

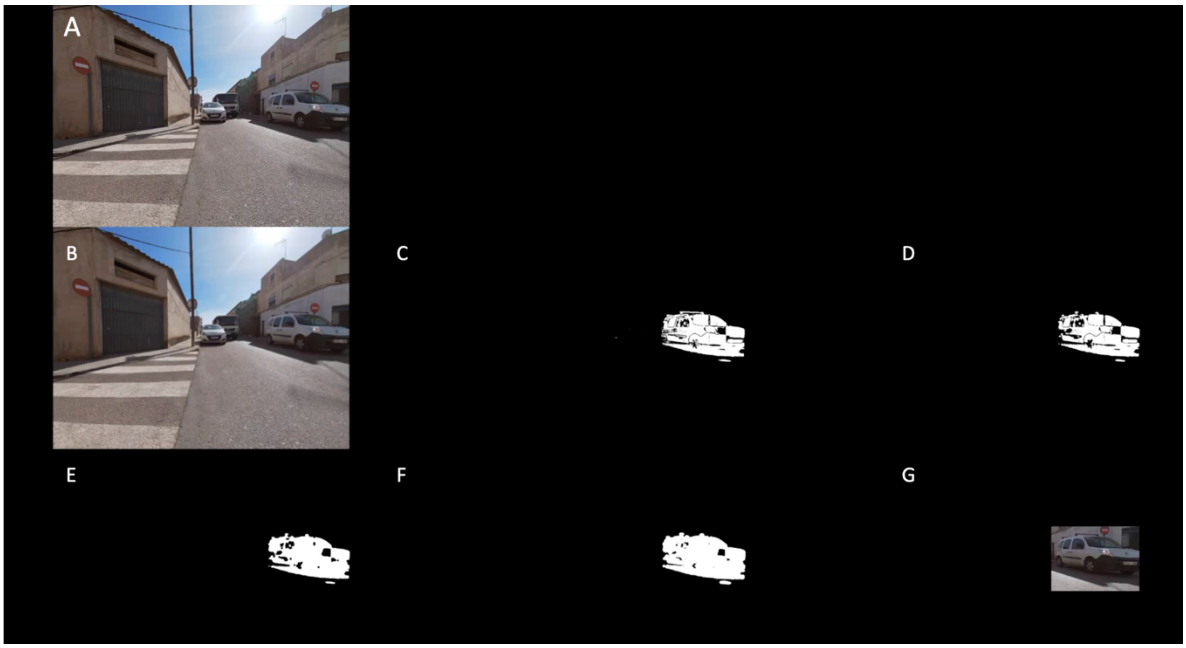
Finalment fent ús de la funció **Dilate** s'expandirà la màscara per tal de reduir la possibilitat que alguna part de l'objecte en moviment quedi emmascarat.

Il·lustració 14: Exemple de la funció *Dilate* (Font: Referència [33]**)**

6.4.4 SELECCIÓ DE CONTORNS I *BOUNDING BOX*

Per acabar el *Background Substraction*, se cercaran els contorns de la màscara obtinguda després de la transformació morfològica atès que pot ser que al processat anterior podrien quedar obertures a la màscara, la qual cosa Els contorns que s'obtinguin seran usats per a generar unes ***bounding boxes*** -que encapsularan aquells contorns que hagin passat un filtre de mida- aconseguint l'objectiu d'ampliar les màscares dels objectes que estan en moviment.

La següent il·lustració presenta els diferents passos que va seguint l'algoritme per aïllar els elements mòbils de les imatges. La imatge A, és l'original. Després aquesta es suavitza (per tal d'eliminar soroll), i se li aplica l'algoritme de ***background substraction***. A la imatge D es pot veure com els petits puntets que hi ha al cotxe de la imatge C desapareixen quan s'ha aplicat **l'*opennig*.** A les imatges E i F es veu l'efecte d'aplicar el ***closing*** i el ***dilate***. Finalment es veu que l'algoritme de selecció de contorns i ***bounding box***, crea un rectangle al voltant de l'objecte mòbil.



Il·lustració 15: Etapes del Background substraction. a) Imatge original b) Blur c) Algoritme KNN d) Opening E) Closing F) Dilate G) Find Contours & Bounding Box

6.5 DETECCIÓ D'OBJECTES

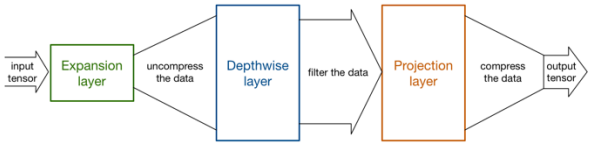
A continuació del *Background Substraction* es realitza la detecció d'objectes. Gràcies al pas anterior els objectes estàtics, com són els cotxes aparcats, no es reconeixeran. D'aquesta manera s'evita que es notifiqui erròniament al conductor, qui ha d'estar alerta d'aquells objecte que poden suposar un perill(e.g. un cotxe aproximant-se o un vianant travessant el carrer).

Per fer aquesta tasca s'usarà la xarxa Mobilenet-SSD v2 preentrenada amb el data set COCO(Comon Object in Context) [34], sent capaç de reconèixer fins a 90 objectes diferents i el fons [35]. Aquesta està composta per dos components principals: MobileNet [36], una xarxa neuronal profunda amb una arquitectura eficient pensada per poder ser usada en dispositius amb menor capacitat de computació com poden ser telèfons i sistemes encastats, i una xarxa SSD (*Single Shot Multibox Detector*) [23], que permet la detecció de simultània de múltiples objectes en una imatge.

MOBILENET

Es tracta d'una xarxa neuronal convolucional que funciona com a classificador. La característica que la diferencia de la resta -i permet que aquesta pugui treballar de manera més eficient en sistemes de menor prestacions- és l'ús d'un nou tipus de

capes convolucionals anomenades *Depthwise Separable Convolutions*. Aquestes estan basades en una capa convolucional normal, de la dimensió desitjada, però en aquest cas s'afegeixen dues capes convolucionals 1x1, una al davant i l'altre al darrere. La primera té la funció d'expandir els tensors i la darrera que actua com un coll d'ampolla reduint el nombre de tensors. Dit d'una altra manera, és com si es descomprimissin les dades recollides a la imatge i un cop processades es comprimís el resultat [32,33].

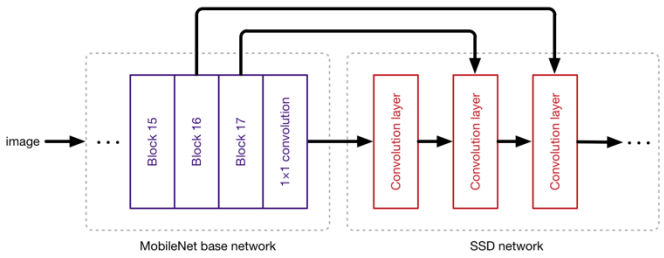


Il·lustració 16: Representació figurativa de capa *depthwise* separable (Font: Referència [37]**)**

SSD

Realitza la classificació i la localització en una sola passada de la xarxa, i simultàniament prediu la classe i les *bounding box* a mesura que processa la imatge. A la imatge 17 es mostra l'arquitectura que té. En aquesta es pot apreciar que Mobilenet queda integrat a la primera part i funciona com a extractor de característiques per a SSD. Això és possible gràcies a la flexibilitat del SSD que permet canviar l'arquitectura bàsica que usa [37].

Aquesta xarxa produeix un nombre fix de *bounding boxes* i una puntuació per la presència d'instàncies d'objectes d'una classe dins aquestes *bounding boxes*. Finalment acaba amb un pas de *Non-maximum supression* per agrupar les *bounding boxes* amb un alt grau de solapament. [23]



Il·lustració 17: Estructura de la xarxa MobileNetSSD-v2 (Font: Referència [37]**)**

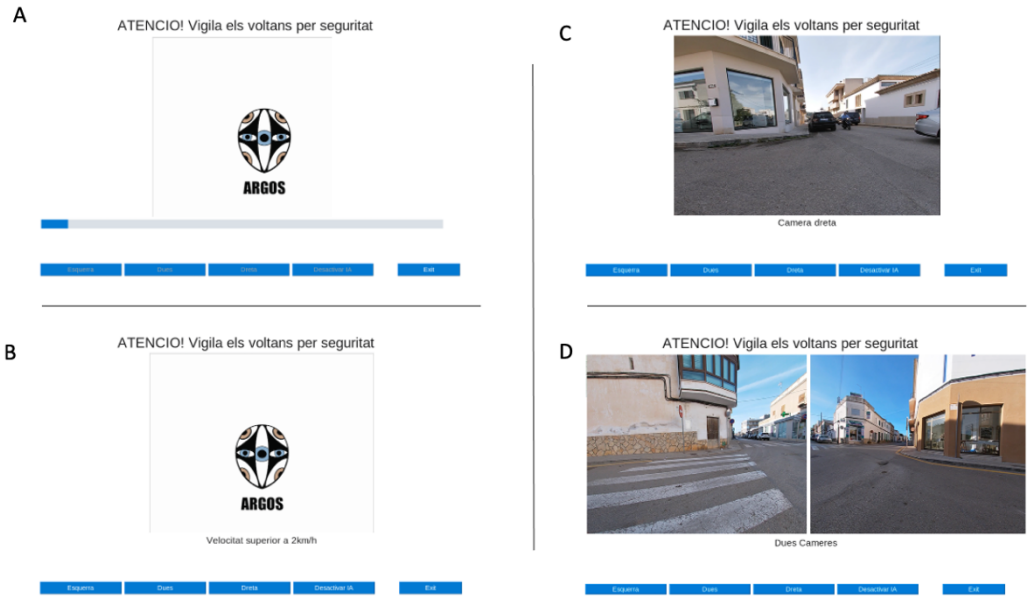
Un cop definit l'algoritme de reconeixement d'objectes que s'utilitza, expliquem ara de quina manera es tracten les dades que s'obtenen d'aquest. Els objectes que es reconeixen, i.e. les classes permeses, són: vehicles (bicis, busos, cotxes, motos) i persones. En cas que un objecte no estigui a les classes mencionades, aquest no es detectaria. Per exemple, en el cas que una pilota rodés al mig del carrer no es detectaria, però si un nin la seguís per agafar-la aquest sí que seria detectat, ja que, la classe persona sí que es detecta. Més endavant, a l'apartat de discussió, es comentarà què passaria en cas que aparegués un objecte que no està a la llista de reconeguts.

Quan es reconegui algun objecte dels objectes establerts, per primer cop d'ença que el cotxe es para a una cantonada, sonarà un soroll per avisar al conductor. Aquest no tornarà a sonar fins que no es deixi de detectar alguna classe mentre que el vehicle segueixi parat o un cop que el cotxe es torni a moure i es pari a una nova cantonada.

A més de l'avís sonor el dispositiu compta amb uns avisos lluminosos per identificar si el que s'ha detectat és un vianant o un vehicle i a quin costat d'aquest es troba. Aquests es mantindran actius en tot moment que es detecti una de les classes vàlides. A l'apartat següent es podrà veure amb més detall com es veurà la sortida que rebrà el conductor.

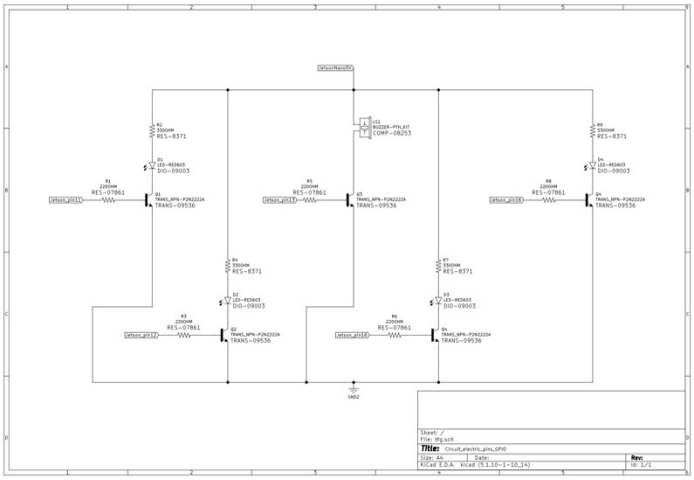
6.6 INTERFÍCIE D'USUARI I AVISOS

Com s'ha comentat anteriorment, per la creació de la interfície d'usuari es fa servir PySimpleGUI. L'usuari podrà interactuar amb la GUI mentre que el cotxe es desplaci a una velocitat inferior a 2 km/h o estigui parat, i podrà decidir quina de les càmeres es veu. A més d'això s'ha incorporat un boto que permet desactivar el reconeixement d'objectes, en cas que el conductor únicament desitgi tenir actives les càmeres i no els avisos. A continuació es presenten les diferents pantalles que veurà l'usuari:



Il·lustració 18: Pantalles de la GUI. A) Pantalla de carrega B) Pantalla mentre el vehicle està en moviment C) Pantalla amb una sola càmera D) Pantalla amb les dues càmeres

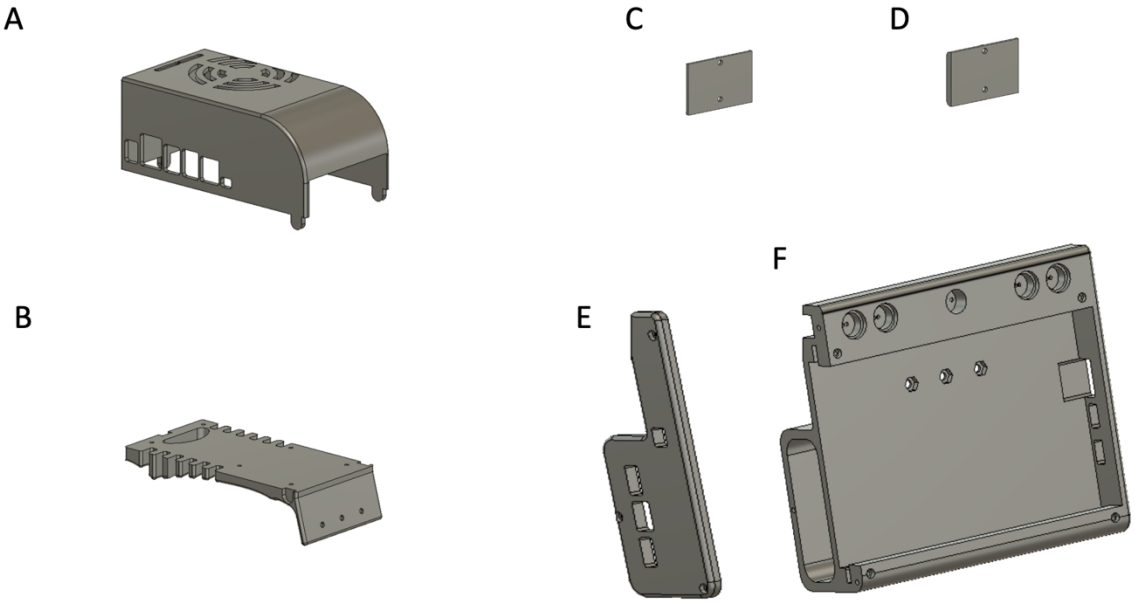
Els avisos lluminosos i sonors estan integrats en el circuit electrònic que es mostra en la Figura 19. Aquests s'activaran quan es reconegui algun dels objectes rellevants a les càmeres. Hi ha quatre senyals lluminosos, 2 a cada banda. D'aquesta manera el conductor pot saber ràpidament en quin costat s'ha detectat l'objecte/s. Els senyals lluminosos es divideixen en dues categories: persones i vehicles (cotxes, motocicletes, bicicletes i camions).



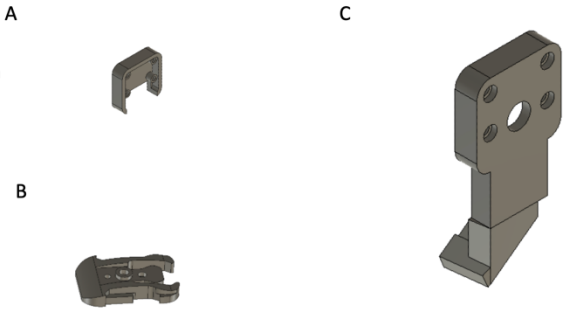
Il·lustració 19: Circuit electrònic generat amb KiCad

Per encapsular el projecte, s'ha dissenyat una carcassa per la pantalla, la bateria, el sistema encastat i la placa d'electrònica, i una altra diferent per cada càmera. Aquesta carcassa també inclou el mètode de subjecció al cotxe. Aquesta tasca s'ha dut a terme fent ús del programa de CAD Fusion360. També s'han generat els plànols de disseny de les diferents parts.

A les imatges 20 i 21, es mostren les diferents parts del disseny final. A l'annex 4 es poden trobar els plànols de disseny de cada peça.



Il·lustració 20: Peces de la carcassa de pantalla, bateria i plaques. A) Tapa carcassa placa B) Base carcassa placa C) i D) Tapes part posterior carcassa pantalla E) Carcassa pantalla i bateria



Il·lustració 21: Carcassa de la càmera dreta.

A) Part posterior carcassa. B) Clip per enganxar al cotxe C) Part davantera carcassa

7. RESULTATS

A fi de poder avaluar els resultats que impliquin processament d'imatge, s'han gravat 12 vídeos a diferents cantonades, amb les càmeres col·locades a la seva posició, per tal de crear un conjunt de dades per la validació dels algoritmes proposats. D'aquests, s'han seleccionat 5 vídeos de forma aleatòria dels quals s'han recollit 4000 imatges (*frames)*. De les 4000 imatges, se n'han tornat a seleccionar 300 de manera aleatòria.

7.1 RECOL·LECCIÓ DADES DEL VEHICLE

La recol·lecció de les dades es du a terme de forma força precisa. Això es deu al fet que el senyal que s'envia al tacòmetre és el mateix que llegeix el lector OBD. A la imatge 22, es pot veure com queda la pantalla quan el cotxe va a una velocitat superior a 2 km/h.

Il·lustració 22: Exemplificació de la GUI amb el cotxe parat, i amb el cotxe en moviment

La connexió entre la placa no ha pogut ser establerta a causa d'un error de la placa adquirida per el projecte. Però per provar el funcionament del lector s'ha fet ús d'un ordinador amb aquest s'ha provat que el funcionament d'aquest era adequat i es complia la funció de recol·lectar la velocitat del cotxe i activar i desactivar la pantalla.

Cal comentar que existeix un petit retard, de 4 segons, a causa de la comunicació sèrie entre la l'ordinador i el lector. Aquest no té una importància cabdal i realment no afecta de forma significativa el resultat final del projecte. La principal conseqüència és que la GUI tarda aquest temps de més a canviar.

7.2 MILLORA DE LA VISIBILITAT

Un cop revisades les diferents imatges del data set i els vídeos es pot afirmar que el posicionament de les càmeres permet aconseguir l'objectiu proposat de reduir la distància que el cotxe envaeix el carrer, millorant així la visibilitat del conductor i reduint el risc d'accident.

A la següent imatge, es pot veure un exemple des del punt de vista del conductor de com es veuria la imatge al dispositiu.

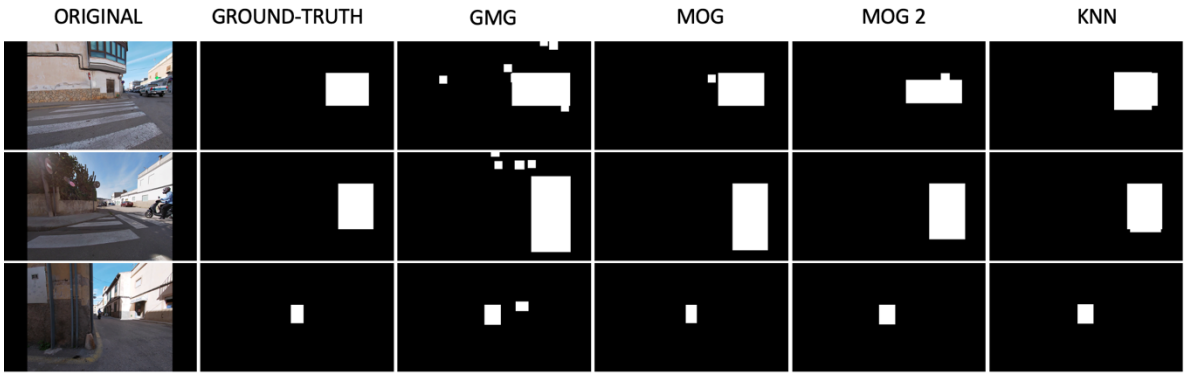
Il·lustració 23: Visió de les càmeres des del punt de vista del conductor

7.3 BACKGROUND SUBTRACTION

En aquest apartat es presentaran els resultats de la selecció de l'algoritme de fons i alguns exemples de les imatges obtingudes després de que s'hagi aplicat el *Background Substraction*.

Amb l'objectiu de seleccionar i avaluar l'algoritme de ***Background subtraction***, s'ha fet ús del conjunt d'imatges mencionat al principi de l'apartat . Per poder comparar els resultats obtinguts amb els esperats, s'ha obtingut el *ground-truth* fent ús de Photoshop, tal com s'ha

indicat a l'apartat de metodologia. La següent il·lustració presenta un exemple de les mascares obtingudes comparades amb el *ground-truth*.



Il·lustració 24: Comparació de les mascares dels algoritmes de *background substraction*

7.3.1 SELECCIÓ DE L'ALGORITME

A la taula 1 es presenten els resultats obtinguts de l'avaluació dels diferents algoritmes de *Background Subtraction*. Per l'avaluació de cadascun dels algoritmes s'han utilitzat les màscares -segmentades manualment i descrites amb anterioritat- com a *ground thruth*. La taula presenta les mesures usades per a l'avaluació que s'han presentat a la metodologia: **Precisió** (rati píxels classificats correctament), **Sensitivitat** (Positius vertaders respecte tots els positius), **Especificitat** (Número de mostres classificades com a negatives respecte de totes les mostres), **F-1** (relació entre sensitivitat i precisió) i temps de processat (temps que es triga a processar cada *frame*)

Taula 1: Avaluació algoritmes de Backgroud Substraction

Algoritme BS	Precisió	Sensitivitat	Especificitat	Valor F-1	Temps de processat (s)
KNN	0,983	0,883	0,922	0,853	0,0015
MOG	0,913	0,599	0,736	0,543	0,003
MOG2	0,973	0,785	0,711	0,648	0,002
GMG	0,950	0,375	0,931	0,474	0,0025

A la taula 1 es veu que tots els algoritmes tenen un valor elevat de precisió, però com s'ha comentat abans, aquest valor es pot veure afectat per les dades no balancejades. Tenint en compte això és necessari avaluar la resta de mesures.

Al ser un projecte pel qual allò més important és que es detectin correctament els positius, és a dir, els objectes que es trobin en moviment, la sensitivitat és un paràmetre que s'ha de tenir en compte. Aquesta necessitat es deu al fet que una detecció errònia podria acabar provocant un accident, per exemple, en el cas que no es detectés un cotxe que vingués pel costat i -com a conseqüència de no rebre cap notificació, es podruís una col·lisió. En aquest sentit, el cost de cometre un error de tipus FN *vs* un del tipus FP és molt major.

D'acord amb això es pot descartar l'algoritme MOG i el GMG, ja que no detecten correctament aquests objectes. L'algoritme GMG té una especificitat molt elevada però a l'aplicació d'ús d'aquest projecte no és tan importat. De fet,si es mira el valor-F és el més baix de tots els algoritmes.

Finalment entre el MOG2 i el KNN, s'ha determinat que el KNN és l'algoritme que té un millor funcionament de tots els que s'han provat, tenint en compte que el valor F-1 és més elevat que la resta a més de tenir l'especificitat més gran. Els valors obtinguts en aquesta avaluació concorden amb els obtinguts per altres estudis realitzats [38],[13].

7.3.2 K-NN *BACKGROUND SUBSTRACTION* I POSTPROCESSAT

L'avaluació d'aquest algoritme ha tingut lloc en el propi procés de selecció descrit en l'apartat anterior. Això es deu al fet que, en tots els casos anteriors, les imatges usades per a l'avaluació havien passat per totes les passes del processat des de l'obtenció de la imatge al *bounding box*.

En qualsevol cas, presentem a continuació una breu valoració qualitativa dels procés. A les següents imatges es presenten diferents situacions a les quals s'ha aplicat l'algoritme de *Background Subtraction*. En algunes de les imatges presentades es pot veure com s'aïllen els elements estàtics de les imatges, com són els cotxes aparcats i altres elements que no interessa que es reconeguin. En altres casos es veu com es detecten elements que estan en moviment i potser no haurien de ser detectats, com ara reflexos i ombres.

Il·lustració 25: Exemples del *Background Subtraction* aplicat

7.4 DETECCIÓ D'OBJECTES

7.4.1 SELECCIÓ DE L'ALGORITME

En aquest apartat es donarà una explicació al perquè s'ha seleccionat aquest mètode de detecció d'objectes, a més dels avantatges que aquest suposa sobre els altres. Prèviament a la decisió d'usar la xarxa Mobilenet-SSD v2 s'han fet proves amb el detector d'objectes YOLO (*You Only Look Once*) [22]: aquestes dues xarxes són les més usades per sistemes encastats. Com ja s'ha comentat abans a la recol·lecció de dades del vehicle, per tal de poder detectar vehicles que es desplacin a una velocitat 60 km/h, l'adquisició d'imatges ha de tenir lloc a un ritme superior als 16 FPS. Tenint això en compte, l'algoritme YOLOv3 ha de ser descartat, ja que només permetria reconèixer objectes que es desplacessin a 18 km/h. A la taula 2, es comparen les velocitats d'execució (en FPS) dels algoritmes i les mAP[4] de cada algoritme.

[39]

Taula 2: Comparació resultats algoritmes detecció d'objectes [35],[11],[22], [40]

Algoritme	Velocitat d'execució (FPS)	mAp	Dataset
YOLOv3	5	0.7423 0,606	VOC2007 [41] MS-COCO [34]
Tiny YOLO	25	0,331	MS-COCO
MobileNet-SSD v2	39	0,727 0,68	VOC0712 MS-COCO

Un altre factor molt important és la correcta detecció dels objectes, en aquest cas vehicles i persones. La detecció errònia d'aquests podria provocar un accident. Tenint en compte les dades de la taula, i comparant les dues xarxes restants, es determina que el millor algoritme per aquest cas d'ús és la xarxa MobileNet-SSD v2, atès que presenta una millor precisió que el TinyYOLO.

[4] **mAP:** Es tracta de la *mean Average Precision(mAP)*, una mesura molt usada per a l'avaluació d'algoritmes de reconeixement d'objectes. *L'Average Precision (AP)* es calcula trobant l'àrea sota la corba de **precisió** i ***recall***. La mAP és la mitjana de les AP de totes les classes.

7.4.2 DETECCIÓ D'OBJECTES AMB MOBILENET-SSD V2

Els resultats de processament obtinguts per a la detecció d'objectes s'han vist afectats per diferents elements.

En primer lloc la velocitat d'execució (en FPS) obtinguts a la Jetson Nano es veuen reduïts en comparació als obtinguts a l'ordinador, a causa de la capacitat més limitada de processament. El fet d'usar una GUI incrementa aquesta disminució en la velocitat d'execució passant de 15 FPS a 10 FPS. Això passa a conseqüència de què la GUI es queda esperant l'entrada d'un esdeveniment (*event*), fet que provoca un petit retard i per tant una reducció de velocitat. A la Taula 3, es poden observar les diferent velocitats d'execució en les diferents situacions. Un altre factor que provoca la reducció de velocitat és a conseqüència d'usar una bateria portàtil com a font d'alimentació. Aquest fet fa que l'algoritme deixi de ser funcional, a causa de l'elevada pèrdua de *frames*. Té tal efecte que inclús la detecció de persones és impossible. Però com s'ha comentat a l'apartat de metodologia únicament s'ha usat aquets mètode d'alimentació degut a la limitació del vehicle usat per les proves. En el cas que fos incorporat a un vehicle es faria amb l'alimentació de 10W.

Taula 3: Velocitat de processament del reconeixement d'objectes en diferents situacions

	Dispositius				
	Macbook pro, 16 GB Ram i processador i5	Jetson Nano (alimentació 10W) amb GUI, endollat a la corrent	Jetson Nano (alimentació 10W) sense GUI, endollat a la corrent	Jetson Nano (alimentació 5W) endollat a la corrent	Jetson Nano (alimentació 5W) enxufat al power bank
FPS	20	10	15	6	2

L'avaluació de la detecció d'objectes s'ha dut a terme amb el conjunt de 300 imatges del datat set. Aquestes han estat inspeccionades visualment i s'ha establert si hi havia algun dels objectes a detectar o no. Cada classe s'ha etiquetat com si es tractés d'un problema binari. Per al criteri d'avaluació s'han considerat tres casos, que permetran el càlcul de la **precisió** i el ***recall***. El primer cas seria que el resultat de l'algoritme i el valorat visualment coincideixin, aquest seria PV. En cas que el resultat de l'algoritme indiqui que l'objecte

detectat pertany a una classe errònia serà FP i finalment si l'objecte està present a la imatge però no ha estat detectat s'etiquetarà com a FN. Cal esmentar que hi ha dues classes que no han pogut ser avaluades per la falta de dades. A la Taula 4 es mostra el número de mostres per cada classe que hi ha dins del data set de 300 imatges, i el nombre de PV, FP i FN per cada una d'elles. A la taula s'observa que la majoria d'imatges pertanyen a les classes persona i cotxe.

Taula 4: Numero de mostres per classe, i valors de TP, FP i FN

	Nombre de mostres amb l'objecte present	PV	FP	FN
Cotxe	128	78	6	50
Motocicleta	24	8	0	16
Persona	122	80	10	42
Bicicleta	0	0	0	0
Bus	0	0	6	0

Amb els valors de PV, FP i FN, s'ha creat la següent taula on es poden veure els valors de precisió i ***recall*** obtinguts per cada classe.

Taula 5: Resultats avaluació reconeixement d'objectes

	Precision	*Recall*
Cotxe	0.92	0.68
Motocicleta	1	0.33
Persona	0.93	0.65
Bicicleta		No es tenen suficients imatges
Bus		No es tenen suficients imatges

A la taula anterior es pot observar que el reconeixement es du a terme de manera molt precisa. Això indica que, de totes les deteccions que s'han fet, una gran part han estat correctes. En canvi, el *recall* ens indica que de totes les deteccions que hauria d'haver tingut el model, n'hi ha hagut un gran nombre que no s'han detectat. Després d'analitzar els resultats i revisar les imatges s'ha pogut detectar que el principal problema en els casos en què no s'ha detectat l'objecte és a causa de la distància que es troba respecte a la càmera. L'algoritme comença a tenir problemes amb la detecció quan l'objecte es troba al voltant de 7 metres de la càmera, això pot ser un problema i requeriria una avaluació més profunda per veure si suposa que el sistema no sigui eficaç.

7.5 INTERFÍCIE D'USUARI I AVISOS

Un cop analitzades les dades, a continuació es presenten una sèrie d'imatges on es veu com els avisos lluminosos s'activen quan es detecta algun objecte reconegut. Cal recordar que únicament s'activa del costat que s'ha detectat l'objecte. Al primer exemple es veu com, al detectar-se una persona, s'activa el LED amb el símbol de persona. A la següent imatge es poden veure els dos LEDs actius simultàniament. I finalment a la darrera imatge es pot veure com està actiu un LED a cada costat de la pantalla.

Il·lustració 26: Exemples dels avisos lluminosos en diferents situacions

Finalment, a la imatge 27 es presenta el dispositiu muntat al cotxe. Aquest s'ha enganxat al quadre de comandaments del cotxe amb velcro.

Il·lustració 27: Sistema ARGOS Cam muntat al cotxe

8. DISCUSSIÓ

La diferència de distància que el cotxe surt al carrer suposa una major seguretat vial pel fet que no s'està envaint la via a la que es vol incorporar el conductor, fet que redueix el risc de col·lisió. La qualitat de les imatges mostrades per pantalla, tot i que no alta, és suficient perquè es puguin reconèixer els objectes que es pretenia: vehicles i persones. Ara bé, en condicions adverses, com pot ser una pluja forta, boira o una zona mal il·luminada durant el vespre o a que la càmera quedi a contra llum, poden suposar que les imatges captades siguin inservibles i l'objecte no sigui detectat correctament. Aquestes adversitats podrien ser superades, tot i que ja es requeriria un estudi més complet per poder trobar un mètode adequat.

En relació al mòdul de *Background Subtraction*, l'algoritme resultant ha presentat un resultats que compleixen amb la seva funció d'aïllar els objectes estàtics com són els cotxes aparcats. Però el fet d'emmascarar els subjectes estàtics també pot portar problemes en alguns casos en què una persona es quedi estàtica, e.g. Il·lustració 24, s'ha pogut veure un exemple. Si es dóna el cas aquesta també serà emmascarada i no serà detectada per l'algoritme de detecció d'objectes. En altres casos també es detecten els reflexes, ja que, aquests es tradueixen en moviment en la imatge. Val a dir que aquests no produeixen un efecte negatiu, únicament augmenten les zones detectades.

Per poder solucionar el problema de les persones que es queden estàtiques, s'hauria d'intentar modificar l'algoritme, per exemple, combinant-lo amb algun altre detector d'objectes per reconèixer únicament persones o fer us d'un algoritme de *Semantic Segmentation* com els que s'ha presentat a l'estat de l'art.

Respecte del reconeixement d'imatge, s'ha demostrat que és possible aconseguir una detecció amb uns FPS adequats i amb una precisió elevada permetent detectar els objectes de forma correcta, sempre i quan funcioni amb una potencia de 10W i sigui alimentada a 5V 4A. El problema principal és que aquesta detecció no es pot dur a terme amb la placa seleccionada si es fa ús de la interfície gràfica, ja que aquesta redueix els FPS a la meitat passant a un valor inferior al que es desitjava. A més a més, els baixos resultats de *recall* obtinguts i els casos de FN observats durant l'etiquetatge, són un indicatiu de què s'hauria de millorar la xarxa neuronal per tal que pugui reconèixer els objectes a una major distància.

Un altre punt a destacar és que en cas que un objecte que no estigui dins les classes d'objectes mencionades, aquest no seria reconegut. Això podria provocar una situació perillosa. Un exemple podria ser: si una pilota es posés en mig del carrer aquesta no seria detectada, però al primer moment en què aparegués una persona al darrere aquesta sí que apareixeria. Un altre exemple, podria ser un animal que es creués en mig del carrer, aquest a no ser que anés seguit d'un objecte reconegut no es detectaria i no s'avisaria al conductor, fet que podria acabar provocant un accident. Vist aquets exemples es deixa com a treball futur ampliar el nombre de classes detectades introduint objectes comuns que poden estar al mig de la carretera.

9. CONCLUSIÓ

En conclusió amb la realització d'aquest projecte s'ha demostrat que és possible la fabricació d'un dispositiu que fa ús de processament d'imatge, que permeti millorar la visibilitat a les cantonades i d'aquesta manera fent possible la reducció d'aquests tipus d'accidents.

Aquest dispositiu podria ser incorporat als sistemes que ja es troben integrats a la majoria dels cotxes d'avui en dia, com pantalles, càmeres i altres sensors que facilitarien la construcció d'aquest. En el cas ideal aquest podria passar a ser un nou ADAS que incorporessin els cotxes.

La part més problemàtica de tot el projecte ha estat la correcte configuració de la Jetson Nano, pel fet d'estar poc familiaritzat amb el sistema operatiu Linux i la instal·lació de paquets en aquest, a més de les incompatibilitats que s'han trobat amb les llibreries amb les que s'havia treballat prèviament. Un altre punt que ha donat problemes ha estat la dràstica reducció de FPS observada en passar el codi a la placa, a causa de la capacitat limitada de multiprocessament que ha presentat i la limitació de la potència de processament – com a conseqüència de l'alimentació amb el *power bank*.

Els tres objectius proposats al principi del projecte han estat completats en la major part. L'únic que no ha pogut ser complert del tot ha estat el de la integració completa del sistema al cotxe, però conegut la font del problema, és senzill de solucionar, alimentant la placa d'una manera més eficient.

En resum, perquè el sistema pogués ser incorporat en un cotxe com a part d'un nou sistema ADAS, aquest hauria de passar per una sèrie de millores. A continuació, es presenten unes possibles futures millores que es podrien aplicar al projecte perquè aquest pugui ser un nou sistema d'assistència al conductor.

9.1 FUTURES MILLORES

9.1.1 RECOL·LECCIÓ DADES DEL VEHICLE

En el cas de la integració dins un vehicle, la lectura de la velocitat es faria de forma interna, fent que aquesta fos més precisa.

9.1.2 CAPTURA D'IMATGES

En primer lloc, per tal de superar el problema de les condicions meteorològiques adverses, es podria intentar usar la tecnologia, *sensor fusion*. Aquesta es usada per alguns cotxes autònoms, com ja s'ha comentat a l'apartat d'estat de l'art. La solució consistiria a combinar la lectura de les càmeres i sensors de distància.

El segon problema a solucionar, seria la dificultat de les càmeres per capturar imatges en zones mal il·luminades, i que es poguessin reconèixer els objectes. Aquest problema es podria intentar solucionar canviant el tipus de càmeres usades per càmeres de visió nocturna. Aquestes ja suposarien l'addició de sensors infrarojos per permetre la visibilitat.

9.1.3 SISTEMA ENCASTAT

La utilització d'una versió superior de la Jetson Nano, com podria ser la Jetson Nano Xavier NX, permetria una major velocitat de processament, millorant així el rendiment del sistema en tots els escenaris.

La millora que seria més important realitzar seria la connexió de la placa a una font que tingués el voltatge i corrents adequats. La connexió directa a la bateria del cotxe amb un convertidor reductor de 12 a 5 V seria una de les millors opcions.

9.1.4 DETECCIÓ D'OBJECTES

Per a la millora d'aquesta secció es podria fer ús de tècniques de *Transfer Learning*, per al reentrenament de la xarxa neuronal usada, aplicades a la situació presentada el projecte, en què la càmera està situada a la cantonada. A més a més, també es podria entrenar amb imatges en situacions adverses com les que s'han presentat a l'apartat de resultats, per veure si s'obté una millora sense necessitat de canviar el hardware.

10. BIBLIOGRAFIA

[1] "El grimorio de bestias: Argos." https://grimoriodebestias.blogspot.com/2014/04/argos.html (accessed May 11, 2021).

[2] DGT, "Tablas estadísticas 2019," 2019. Accessed: May 15, 2021. [Online]. Available: https://www.dgt.es/es/seguridad-vial/estadisticas-e-indicadores/accidentes-30dias/tablas-estadisticas/2019/.

[3] "ADAS: Past, present and future | Vehicle Service Pros." https://www.vehicleservicepros.com/service-repair/diagnostics-and-drivability/article/21198482/adas-past-present-and-future (accessed May 11, 2021).

[4] "What are car surround view cameras, and why are they better than they need to be? - ExtremeTech." https://www.extremetech.com/extreme/186160-what-are-surround-view-cameras-and-why-are-they-better-than-they-need-to-be (accessed May 11, 2021).

[5] "LIDAR vs. Camera — Which Is The Best for Self-Driving Cars? | by Vincent Tabora | 0xMachina | Medium." https://medium.com/0xmachina/lidar-vs-camera-which-is-the-best-for-self-driving-cars-9335b684f8d (accessed May 11, 2021).

[6] "Taller del automovil – Historia del OBD." https://ibtaller.com/historia-del-obd/ (accessed May 11, 2021).

[7] "Sistema OBD de un coche: todo lo que debes saber - canalMOTOR." https://www.motor.mapfre.es/consejos-practicos/consejos-de-mantenimiento/como-funciona-el-sistema-obd/ (accessed May 11, 2021).

[8] "Comparing the Different Interface Standards for Embedded Vision," *14/11/18*. https://www.automate.org/blogs/comparing-the-different-interface-standards-for-embedded-vision (accessed May 12, 2021).

[9] "Beginner's Guide: Choose the Right Camera Modules for Your Raspberry Pi or Jetson Nano Dev Kit." https://www.arducam.com/choose-camera-modules-raspberry-pi-jetson-nano-guide/ (accessed May 12, 2021).

[10] D. Reifs, "Sistemes Encastats."

[11] "Jetson Nano: Deep Learning Inference Benchmarks | NVIDIA Developer." https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks (accessed Apr. 21, 2021).

[12] "Buy a Raspberry Pi 4 Model B – Raspberry Pi." https://www.raspberrypi.org/products/raspberry-pi-4-model-b/ (accessed May 27, 2021).

[13] T. Trnovszký, P. Sýkora, and R. Hudec, "Comparison of Background Subtraction Methods on Near Infra-Red Spectrum Video Sequences," in *Procedia Engineering*, Jan. 2017, vol. 192, pp. 887–892, doi: 10.1016/j.proeng.2017.06.153.

[14] D. Zeng, X. Chen, M. Zhu, M. Goesele, and A. Kuijper, "Background Subtraction with Real-time Semantic Segmentation."

[15] "Semantic Segmentation - MATLAB & Simulink." https://www.mathworks.com/solutions/image-video-processing/semantic-segmentation.html (accessed May 13, 2021).

[16] "What is LiDAR technology?" https://blog.generationrobots.com/en/what-is-lidar-technology/ (accessed Feb. 14, 2021).

[17] "Deep Learning: qué es y por qué va a ser una tecnología clave en el futuro de la inteligencia artificial." https://www.xataka.com/robotica-e-ia/deep-learning-que-es-y-por-que-va-a-ser-una-tecnologia-clave-en-el-futuro-de-la-inteligencia-artificial (accessed May 13, 2021).

[18] K. L. Masita, A. N. Hasan, and T. Shongwe, "Deep learning in object detection: A review," Aug. 2020, doi: 10.1109/icABCD49160.2020.9183866.

[19] D. Sarkar and Towards Data Sience, "A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning | by Dipanjan (DJ) Sarkar | Towards Data Science," *14/11/18*. https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a (accessed May 13, 2021).

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)." Accessed: May 13, 2021. [Online]. Available: http://www.cs.berkeley.edu/˜rbg/rcnn.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." Accessed: May 13, 2021. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/results.

[22] "YOLO: Real-Time Object Detection." https://pjreddie.com/darknet/yolo/ (accessed Apr. 21, 2021).

[23] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector." Accessed: Apr. 21, 2021. [Online]. Available: https://github.com/weiliu89/caffe/tree/ssd.

[24] "SSD object detection: Single Shot MultiBox Detector for real-time processing | by Jonathan Hui | Medium." https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06 (accessed May 13, 2021).

[25] "Elm327-interfaz De Cable Usb V1.5 Para Diagnóstico De Coche,Compatible Con Todos Los Protocolos Obd2 Para Windows Elm 327,Escáner Obd Usb - Buy Elm327 Usb V1.5,Usb Cable Interface,Obd Scanner Product on Alibaba.com." https://spanish.alibaba.com/product-detail/elm327-usb-v1-5-car-diagnostic-usb-cable-interface-supports-all-obd2-protocols-for-windows-elm-327-usb-obd-scanner-697370759.html?spm=a2700.galleryofferlist.0.0.73c03ca2WshEaa (accessed May 27, 2021).

[26] "Getting Started - python-OBD." https://python-obd.readthedocs.io/en/latest/ (accessed May 14, 2021).

[27] "Raspberry Pi Camera Module V2.1 | Módulo de cámara Raspberry Pi, interfaz CSI-2, resolución 3280 x 2464 píxeles, 30fps | RS Components." https://es.rs-online.com/web/p/camaras-para-raspberry-pi/9132664/ (accessed May 17, 2021).

[28] "opencv_contrib/evaluation.py at master · opencv/opencv_contrib · GitHub." https://github.com/opencv/opencv_contrib/blob/master/modules/bgsegm/samples/evaluation.py (accessed Apr. 18, 2021).

[29] "Background Subtraction with OpenCV and BGS Libraries | Learn OpenCV." https://learnopencv.com/background-subtraction-with-opencv-and-bgs-libraries/ (accessed Apr. 18, 2021).

[30] Z. Zivkovic and F. Van Der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780, May 2006, doi: 10.1016/j.patrec.2005.11.005.

[31] "OpenCV: cv::BackgroundSubtractorKNN Class Reference." https://docs.opencv.org/4.5.2/db/d88/classcv_1_1BackgroundSubtractorKNN.html (accessed Apr. 18, 2021).

[32] "KNN Algorithm - Finding Nearest Neighbors - Tutorialspoint." https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm (accessed May 14, 2021).

[33] "OpenCV: Morphological Transformations." https://docs.opencv.org/4.5.2/d9/d61/tutorial_py_morphological_ops.html (accessed Apr. 18, 2021).

[34] "COCO - Common Objects in Context." https://cocodataset.org/#home (accessed Apr. 21, 2021).

[35] "GitHub - chuanqi305/MobileNet-SSD: Caffe implementation of Google MobileNet SSD detection network, with pretrained weights on VOC0712 and mAP=0.727." https://github.com/chuanqi305/MobileNet-SSD (accessed Apr. 21, 2021).

[36] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications."

[37] "MobileNet version 2." https://machinethink.net/blog/mobilenet-v2/ (accessed Apr. 21, 2021).

[38] "[OpenCV Actual Combat] 43 Use OpenCV for background segmentation - Programmer Sought." https://www.programmersought.com/article/10826881459/ (accessed Apr. 18, 2021).

[39] "mAP (mean Average Precision) might confuse you! | by Shivy Yohanandan | Towards Data Science." https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2 (accessed May 31, 2021).

[40] "GitHub - taehoonlee/tensornets: High level network definitions with pre-trained weights in TensorFlow." https://github.com/taehoonlee/tensornets (accessed Apr. 21, 2021).

[41] "The PASCAL Visual Object Classes Homepage." http://host.robots.ox.ac.uk/pascal/VOC/ (accessed Apr. 21, 2021).

[42] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust L1 optimal camera paths," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, no. 1, pp. 225–232, 2011, doi: 10.1109/CVPR.2011.5995525.

11. ANNEX

11.1 CANTONADES

Annex 1: Més exemples de cantonades on es pot aplicar el projecte

11.2 CODI FET AMB PYTHON

Com s'ha comentat a la metodologia el codi consta de dues parts. La primera es una classe creada per controlar els algoritmes de les càmeres i la segona es l'encarregada de gestionar la GUI, la lectura del OBD i la integració de les càmeres al sistema.

11.2.1 CLASSE PROCESSAT D'IMATGE

```
#la part del codi d'execució en paralel de les càmeres esta basada en el codi següent
# MIT License
# Copyright (c) 2019,2020 JetsonHacks
# See license
# A very simple code snippet
# Using two  CSI cameras (such as the Raspberry Pi Version 2) connected to a
# NVIDIA Jetson Nano Developer Kit (Rev B01) using OpenCV
# Drivers for the camera and OpenCV are included in the base image in JetPack 4.3+

# This script will open a window and place the camera stream from each camera in a window
# arranged horizontally.
# The camera streams are each read in their own thread, as when done sequentially there
# is a noticeable lag
# For better performance, the next step would be to experiment with having the window display
# in a separate thread


#Importació de llibreries
from threading import Thread
import threading
import cv2 as cv, cv2
import time
import numpy as np
import os
import RPi.GPIO as GPIO
import jetson.inference as inference
import jetson.utils as utils



#crea clase per definir la camara
class vStream:
   #inicialitzacio clase, entrar paramteres variables
   def __init__(self, xarxa, width, height,placa ,pin_led_persona,pin_led_cotxe, pin_buzzer):
#parametres de la camara (font, ample i alçada)

      #GPIO#
      #definicio parametres necessaris per els pins GPIO
```

```
self.placa = placa
self.pin_led_persona = pin_led_persona
self.pin_led_cotxe = pin_led_cotxe
self.pin_buzzer=pin_buzzer
#definició dels pins com a output i es deixen apagats
self.placa.setup(self.pin_led_cotxe, self.placa.OUT, initial=self.placa.LOW)
self.placa.setup(self.pin_led_persona, self.placa.OUT, initial=self.placa.LOW)
self.placa.setup(self.pin_buzzer, self.placa.OUT, initial=self.placa.LOW)
self. marge_frames = 5  #numero de frames abans de cambiar estat GPIO

#Parametres càmera#
#definició tamany del frame capturat
self.width = width
self.height = height
self.capture = None
# Darrer frame agafat imatge
self.frame = None
self.grabbed = False
self.running = True

#Rexoneixemnet d'objectes#
self.net = xarxa
self.thr = 0.5  # threshold, 0.5 valor per defecte
# definició dels noms de les classes que es reconeixeran
self.clases = ['background', 'truck', 'bicycle', 'bus','car','motorbike','person']
self.classNames = ['person']
self.classVehicle = ['bicycle', 'truck', 'bus', 'car', 'motorbike']
#Parametres de control de l'estat dels GPIO durant el reconeixement
self.led_persona_On = False
self.led_vehicle_On = False
self.detectat_persona = False
self.detectat_vehicle = False
self.contador_detectats_persona = 0  # conta els cops que es detecten persones
self.contador_no_detectats_persona = 0
self.contador_detectats_vehicle = 0  # conta els cops que es detecten vehicles
self.contador_no_detectats_vehicle = 0


#Background substraction#
self.aument_quadre = 15 #valor que s'augmenta el bounding box obtingut
self.color = (255, 255, 255)
#creació backgroun substraction i ajust de parametres
self.fgbg = cv2.createBackgroundSubtractorKNN()
self.fgbg.setHistory(100)
self.fgbg.setNSamples(10)
self.fgbg.setkNNSamples(3)

#definicio dels kernels usats
self.kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (5, 5))
```

```
    self.kernel1 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))

    #Thread#
    #parametres per el control del thread
    self.read_thread = None
    self.read_lock = threading.Lock()
    self.running = False


#Funico per inicialitzar la càmera
def open(self, gstreamer_pipeline_string):
    try:
        # el controlador de vídeo es gstramer
        self.capture = cv2.VideoCapture(gstreamer_pipeline_string, cv2.CAP_GSTREAMER)
        self.grabbed, self.frame = self.capture.read()

    except RuntimeError: #avisa si hi ha un error intentant obrir la càmera
        self.capture = None
        print("Imposible obrir camera")
        print("Pipeline: " + gstreamer_pipeline_string)
        return


#Funcio per iniciar el thread, crida la funcio que agafa nous frames
def start(self):
    if self.running:
        print("Ja s'esta capturant vídeo")
        return None

    if self.capture != None:
        self.running = True
        self.read_thread = threading.Thread(target=self.updateCamera)
        self.read_thread.daemon = True
        self.read_thread.start()
    return self

#funcio per aturar el thread, necessaria per fer neteja quan es tanca el programa
def stop(self):
    self.running = False
    self.read_thread.join()

#Funció per actualitzar la càmera i agafar nous frames, es cridada per el thread
def updateCamera(self):
    while self.running:
        try:
            grabbed, frame = self.capture.read()
            with self.read_lock:
                self.grabbed = grabbed
                self.frame = frame
```

```
            try:
                self.frame2 = cv2.resize(self.frame, (300, 300))  # redimensiona el frame

            except Exception as err: #comprovació d'errors
                print("falla captura camara", err)

        except RuntimeError:
            print("No s'ha pogut llegir la càmera")


    # Funció per obtenir el frame sense cap tipus de processament, manté parats els GPIO
    def getFrame(self):
        print("frame")
        self.placa.output(self.pin_led_cotxe, self.placa.LOW)
        self.placa.output(self.pin_led_persona, self.placa.LOW)
        self.placa.output(self.pin_buzzer, self.placa.LOW)
        return self.frame2

    #Funcio per aplicar el background susbtraction
    def Background_substraction(self):
        try:
            self.blur= cv2.GaussianBlur(self.frame2, (15, 15), 0) # Difumina l'imatge
            self.fgmask = self.fgbg.apply(self.blur)# Calcual la mascara de fons
            _, self.binary = cv2.threshold(self.fgmask, 50, 255, cv2.THRESH_BINARY)  # Asegura que la
mascara sigui binaria
            self.open = cv2.morphologyEx(self.binary, cv2.MORPH_OPEN, self.kernel)  # redueix el soroll de
la imatge
            self.close= cv2.morphologyEx(self.open, cv2.MORPH_CLOSE, self.kernel1)  # fa que la mascara
quedi mes neta, tanca forats
            self.dilate = cv2.morphologyEx(self.close, cv2.MORPH_DILATE, self.kernel1)  # expandeix la
mascara

            #primera detecció de contorns i unica en cas de que hi hagi un sol contorn
            self.contours, self.hierarchy = cv.findContours(self.dilate, cv.RETR_TREE,
cv.CHAIN_APPROX_SIMPLE)
            self.contours_poly = [None] * len(self.contours)
            self.boundRect = [None] * len(self.contours)
            for self.i, self.c in enumerate(self.contours):
                self.contours_poly[self.i] = cv.approxPolyDP(self.c, 3, True)
                self.boundRect[self.i] = cv.boundingRect(self.contours_poly[self.i])

            self.drawing = np.zeros((self.frame2.shape[0], self.frame2.shape[1], 3), dtype=np.uint8)

            for self.i in range(len(self.contours)):
                cv.rectangle(self.drawing, ((int(self.boundRect[self.i][0]) - self.aument_quadre),
                                    (int(self.boundRect[self.i][1])) - self.aument_quadre),
                                ((int(self.boundRect[self.i][0] + self.boundRect[self.i][2]) + self.aument_quadre),
                                (int(self.boundRect[self.i][1] + self.boundRect[self.i][3])) + self.aument_quadre),
self.color, cv2.FILLED)
```

```
        self.draw = cv2.cvtColor(self.drawing, cv2.COLOR_BGR2GRAY)

        # segona detecció de contorns, serveix per unificar la mascara en cas de que hi hagi alguns
requadres en contacte
        self.contours_, self.hierarchy_ = cv.findContours(self.draw, cv.RETR_TREE,
cv.CHAIN_APPROX_SIMPLE)
        self.contours_poly = [None] * len(self.contours_)
        self.boundRect = [None] * len(self.contours_)
        self.tamany = 20 #filtra els contorns oer la mida
        self.pasa_filtro = []  # lista de contornos que superan el area deseada
        for self.j, self.co in enumerate(self.contours_):
            self.area = cv2.contourArea(self.co)
            if self.area > self.tamany:
                self.pasa_filtro.append(self.j)
                self.contours_poly[self.j] = cv.approxPolyDP(self.co, 3, True)
                self.boundRect[self.j] = cv.boundingRect(self.contours_poly[self.j])

        self.drawing_ = np.zeros((self.frame2.shape[0], self.frame2.shape[1], 3), dtype=np.uint8)

        #filtre per veure si s'aplica la segona detecció de contorns
        if len(self.contours_) > 0:
            for self.h in self.pasa_filtro:
                cv.drawContours(self.drawing_, self.contours_poly, self.h, self.color)
                cv.rectangle(self.drawing_,
                        ((int(self.boundRect[self.h][0]) - self.aument_quadre),
(int(self.boundRect[self.h][1])) - self.aument_quadre), \
                        ((int(self.boundRect[self.h][0] + self.boundRect[self.h][2]) + self.aument_quadre),
                        (int(self.boundRect[self.h][1] + self.boundRect[self.h][3])) + self.aument_quadre),
self.color, cv2.FILLED)

            self.res_box = cv2.bitwise_and(self.frame2, self.frame2,
                            mask=cv2.cvtColor(self.drawing_, cv2.COLOR_BGR2GRAY))
        else:
            self.res_box = cv2.bitwise_and(self.frame2, self.frame2,
                            mask=cv2.cvtColor(self.drawing, cv2.COLOR_BGR2GRAY))

        return self.res_box

    except Exception as fallo: #comprova errors
        print("error background substraction ", fallo)


#funcio per el reconeixement d'objectes amb el background substraction aplicat
def reconocimiento_con_bg(self):
    try:
        #aplica background substraction
        self.res_box=self.Background_substraction()
        self.backgroud_resized= cv2.resize(self.res_box, (300, 300)) #reescalat de la imateg
```

```
        self.copia_frame = self.frame2.copy()
        self.backgroud_resized = cv2.cvtColor(self.backgroud_resized, cv2.COLOR_BGR2RGBA)

        # aplicació de la funcio per transformar imatge al format cuda
        self.backgroud_resized = utils.cudaFromNumpy(self.backgroud_resized)
        self.cols = self.backgroud_resized.shape[1]
        self.rows = self.backgroud_resized.shape[0]
        self.detections = self.net.Detect(self.backgroud_resized) #funcio de reconeixement
optimitzada amb cuda

        self.detectat_persona = False
        self.detectat_vehicle = False
        for self.detect in self.detections: #loop que recorre totes les deteccions obtingudes
            self.ID = self.detect.ClassID
            self.item = self.net.GetClassDesc(self.ID)

            self.confidence=self.detect.Confidence

            if self.confidence > self.thr and (self.item in self.clases): #filtra per classe i per el threshold
definit
                print(self.item)


                # Control dels GPIO
                # Si un dels leds no esta actiu i el contador de detectats es menor de 5 pitara i
engegadra el llum
                if self.item in self.classVehicle:
                    self.detectat_vehicle=True
                    if (not self.led_vehicle_On) and self.contador_no_detectats_vehicle >
self.marge_frames:  # si el led no es troba ences
                        #engega el led corresponent a la classe trovada
                        print("suen buzzer")
                        self.placa.output(self.pin_buzzer, self.placa.HIGH)
                        self.led_vehicle_On = True
                        self.contador_no_detectats_vehicle = 0
                        self.placa.output(self.pin_led_cotxe, self.placa.HIGH)
                        print("engega led vehicle")

                if self.item in self.classNames:
                    self.detectat_persona = True
                    if (not self.led_persona_On) and self.contador_no_detectats_persona >
self.marge_frames:  # si el led no es troba ences

                        self.led_persona_On = True
                        self.contador_no_detectats_persona = 0
                        self.placa.output(self.pin_led_persona, self.placa.HIGH)
                        print("engega led persona")

                    #self.placa.output(self.pin_buzzer, self.placa.LOW)
```

```
        if self.detectat_persona:  #augmenta el nombre de detectats
            self.contador_detectats_persona += 1
            if self.contador_detectats_persona > 2:
                self.placa.output(self.pin_buzzer, self.placa.LOW)
            print("detectat", self.contador_detectats_persona)

        else: #augmenta el nombre de no detetcats
            self.contador_no_detectats_persona += 1
            print("no  detectat", self.contador_no_detectats_persona)
            if self.contador_no_detectats_persona>2:
                self.placa.output(self.pin_buzzer, self.placa.LOW)

        if self.detectat_vehicle:  #augmenta el nombre de detectats
            self.contador_detectats_vehicle += 1
            if self.contador_detectats_vehicle > 2:
                self.placa.output(self.pin_buzzer, self.placa.LOW)
            print("detectat", self.contador_detectats_vehicle)

        else: #augmenta el nombre de no detetcats
            self.contador_no_detectats += 1
            print("no  detectat", self.contador_no_detectats_vehicle)
            if self.contador_no_detectats>5:
                self.placa.output(self.pin_buzzer, self.placa.LOW)

        #if not self.detectat:  # si el led es troba on y no es detecta
        if (not self.detectat_vehicle) and self.contador_detectats_vehicle > self.marge_frames:
            if self.led_vehicle_On:
                print("led vehicle apagat")
                self.placa.output(self.pin_led_cotxe, self.placa.LOW)
                self.led_vehicle_On = False
                self.contador_detectats_vehicle = 0

        if (not self.detectat_persona) and self.contador_detectats_persona > self.marge_frames:
            if self.led_persona_On:
                print("led persona apagat")
                self.placa.output(self.pin_led_persona, self.placa.LOW)
                self.led_persona_On = False
                self.contador_detectats_persona = 0


        return self.copia_frame, self.frame2  #retorna el frame original
    except Exception as error_rec: #control d'errors
        print("Error reconocimiento", error_rec)


#funcio per dur a terme el reconeixement sense background substraction
```

```
  def reconocimiento_sin_bg(self):
    try:
      # en lloc d'agafar la sortida del background agafa el frame de la càmera
      self.backgroud_resized = cv2.resize(self.frame2, (300, 300))
      self.copia_frame = self.frame2.copy()
      self.backgroud_resized = cv2.cvtColor(self.backgroud_resized, cv2.COLOR_BGR2RGBA)
      self.backgroud_resized = utils.cudaFromNumpy(self.backgroud_resized)
      self.cols = self.backgroud_resized.shape[1]
      self.rows = self.backgroud_resized.shape[0]

      #aplica reconeixement d'objectes amb cuda
      self.detections = self.net.Detect(self.backgroud_resized)

      self.detectat_persona = False
      self.detectat_vehicle = False
      for self.detect in self.detections:  # loop que recorre totes les deteccions obtingudes
        self.ID = self.detect.ClassID
        self.item = self.net.GetClassDesc(self.ID)

        self.confidence = self.detect.Confidence

        if self.confidence > self.thr and (
            self.item in self.clases):  # filtra per classe i per el threshold definit
          print(self.item)

          # Control dels GPIO
          # Si un dels leds no esta actiu i el contador de detectats es menor de 5 pitara i
engegadra el llum
          if self.item in self.classVehicle:
            self.detectat_vehicle = True
            if (
            not self.led_vehicle_On) and self.contador_no_detectats_vehicle > self.marge_frames:
# si el led no es troba ences
              # engega el led corresponent a la classe trovada
              print("suen buzzer")
              self.placa.output(self.pin_buzzer, self.placa.HIGH)
              self.led_vehicle_On = True
              self.contador_no_detectats_vehicle = 0
              self.placa.output(self.pin_led_cotxe, self.placa.HIGH)
              print("engega led vehicle")

          if self.item in self.classNames:
            self.detectat_persona = True
            if (
            not self.led_persona_On) and self.contador_no_detectats_persona >
self.marge_frames:  # si el led no es troba ences

              self.led_persona_On = True
              self.contador_no_detectats_persona = 0
```

```
                self.placa.output(self.pin_led_persona, self.placa.HIGH)
                print("engega led persona")

            # self.placa.output(self.pin_buzzer, self.placa.LOW)

    if self.detectat_persona:  # augmenta el nombre de detectats
        self.contador_detectats_persona += 1
        if self.contador_detectats_persona > 2:
            self.placa.output(self.pin_buzzer, self.placa.LOW)
        print("detectat", self.contador_detectats_persona)

    else:  # augmenta el nombre de no detetcats
        self.contador_no_detectats_persona += 1
        print("no  detectat", self.contador_no_detectats_persona)
        if self.contador_no_detectats_persona > 2:
            self.placa.output(self.pin_buzzer, self.placa.LOW)

    if self.detectat_vehicle:  # augmenta el nombre de detectats
        self.contador_detectats_vehicle += 1
        if self.contador_detectats_vehicle > 2:
            self.placa.output(self.pin_buzzer, self.placa.LOW)
        print("detectat", self.contador_detectats_vehicle)

    else:  # augmenta el nombre de no detetcats
        self.contador_no_detectats += 1
        print("no  detectat", self.contador_no_detectats_vehicle)
        if self.contador_no_detectats > 5:
            self.placa.output(self.pin_buzzer, self.placa.LOW)

    # if not self.detectat:  # si el led es troba on y no es detecta
    if (not self.detectat_vehicle) and self.contador_detectats_vehicle > self.marge_frames:
        if self.led_vehicle_On:
            print("led vehicle apagat")
            self.placa.output(self.pin_led_cotxe, self.placa.LOW)
            self.led_vehicle_On = False
            self.contador_detectats_vehicle = 0

    if (not self.detectat_persona) and self.contador_detectats_persona > self.marge_frames:
        if self.led_persona_On:
            print("led persona apagat")
            self.placa.output(self.pin_led_persona, self.placa.LOW)
            self.led_persona_On = False
            self.contador_detectats_persona = 0

    return self.copia_frame, self.frame2
except Exception as error_rec:
    print("Error reconocimiento", error_rec)
```

```
#funcio per definir els parametres de la càmera raspberry a la Jetson Nano
def gstreamer_pipeline(
    sensor_id=0,
    sensor_mode=3,
    capture_width=1280,
    capture_height=720,
    display_width=640,
    display_height=480,
    framerate=30,
    flip_method=0,
):
    return (
        "nvarguscamerasrc sensor-id=%d sensor-mode=%d ! "
        "video/x-raw(memory:NVMM), "
        "width=(int)%d, height=(int)%d, "
        "format=(string)NV12, framerate=(fraction)%d/1 ! "
        "nvvidconv flip-method=%d ! "
        "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
        "videoconvert ! "
        "video/x-raw, format=(string)BGR ! appsink"
        % (
            sensor_id,
            sensor_mode,
            capture_width,
            capture_height,
            framerate,
            flip_method,
            display_width,
            display_height,
        )
    )
```

Annex 2: Classe processament d'imatge

11.2.2 CODI PRINCIPAL

```
#S'importen les llibreries
import PySimpleGUI as sg
from camaras_sincronizadas_reconocimiento_modos import vStream, gstreamer_pipeline
import numpy as np
import cv2 as cv, cv2
import time
from obd.protocols import ECU
from obd import OBDStatus
import obd
from queue import Queue
import threading
import Jetson.GPIO as GPIO
import jetson.inference as inference
import jetson.utils as utils


obd.commands.TIME_SINCE_DTC_CLEARED.ecu = ECU.ALL
que = Queue()

# definicio de la velocidad per poder ser usada com a global
vel = 0
velocitat_alta=True
surt =True

#funcio per poder realitzar la connexio

#Funcio per connectar el lector OBD a la placa mitjançant un thread
def connect(n):
    n = "bien"
    rapid = True
    ports = obd.scan_serial()  # escanetja els ports usb, i torna els disponibles
    print(ports)  #mostra els ports trobats

    #estableix una connexio assincrona per poder anar llegint valors del cotxe
    connexio = obd.Async(ports[0],baudrate=38400, protocol="3", fast=rapid, timeout=30)
    while len(connexio.supported_commands) < 10: #intenta connectar fins que s'obtenen més de 10
commandaments funcionals
        try:
            connexio.stop()
            time.sleep(5)# para el thread 5 segons per intentar que es connecti sense problemes
            connexio = obd.Async(ports[0], 38400, "3", fast=rapid, timeout=30)
            #connexio = obd.Async("/dev/tty.usbserial-145220", 38400, "3", fast=rapid, timeout=30)

        except Exception as error:
            print(error)

    print("acaba")
```

```
    if OBDStatus.CAR_CONNECTED: #comfirma la connexio amb el cotxe
        print("conectat")
    return connexio


#funcio per obtenir el valor de la velocitat per el port serie
def new_value1(velocitat):
    #definició de variables globals per poder usar a tot el programa
    global velocitat_alta
    global surt
    global vel
    vel = velocitat.value.magnitude  # assigna valor a la velocitat
    print(vel)
    #condicional per poder regular la GUI
    if vel < 6:
        velocitat_alta = False

    else:
        velocitat_alta = True

# Funcio que actua com a contador de temps per a la barra de carrega
def tiempo(second, window, thread):
    progress = 0
    for i in range(int(second * 10)):
        time.sleep(1)  # sleep for a while
        progress += 100 / (second * 10)
        window['-SEC-'].click()

        if not (thread.is_alive()): #comprova si el thread de carrega ha acabat o no
            window['-THREAD-'].click()
            break
        print("threading1", progress)

    window['-THREAD-'].click()
    print("acabado_2") #confirma que acaba el thread


def main():
    sg.theme('Reddit')

    # ---------- Es creen els layouts que conformen les pantalles----------
    gif = "parche.png" #imatge de la pantalla de carrega
    altura = 400
    ample_1 = 650
    ample_2 = 490
    #Layout de carrega de la pantalla
    layout1 = [[sg.Image(filename=gif, size=(altura, altura), background_color='white', key='-IMAGE-')],
               [sg.ProgressBar(30, orientation='h', size=(100, 20), key='progbar')],
               ]
```

```
    # layout amb la camara esquerra
    layout2 = [
        [sg.Image(filename=gif, key='image1', pad=(200, 0), size=(ample_1, altura))],
        [sg.Text('Camera esquerra', size=(40, 1), justification='center', font='Helvetica 20')]]

    # layout amb dues cameres
    layout3 = [[sg.Image(filename=gif, key='image2', size=(ample_2, altura)),
                sg.Image(filename=gif, key='image3', size=(ample_2, altura))],
               [sg.Text('Dues Cameres', size=(40, 1), justification='center', font='Helvetica 20')]
               ]

    # layout camara dreta
    layout4 = [
        [sg.Image(filename=gif, key='image4', pad=(200, 0), size=(ample_1, altura))],
        [sg.Text('Camera dreta', size=(40, 1), justification='center', font='Helvetica 20')]]

    layout_moviment = [
        [sg.Image(filename=gif, background_color='white', key='-IMAGE2-', pad=(200, 0), size=(ample_1,
altura))],
        [sg.Text('Velocitat superior a 2km/h', size=(40, 1), justification='center', font='Helvetica 20')]]

    #layout dels botons
    layout5 = [[sg.Button('Esquerra', size=(20, 1), pad=(2, 2), font='Helvetica 14', visible=True,
disabled=True),
                # cambiar per icono
                sg.Button('Dues', size=(20, 1), pad=(2, 2), font='Helvetica 14', visible=True, disabled=True),
                # cambiar per icono
                sg.Button('Dreta', size=(20, 1), pad=(2, 2), font='Helvetica 14', visible=True, disabled=True),
                sg.Button(button_text='Desactivar IA', size=(20, 1), pad=(2, 2), font='Helvetica 14',
visible=True,
                          disabled=True, key='-IA-'),
                sg.Button('Exit', size=(20, 1), pad=(50, 2), font='Helvetica 14', visible=True)]]

    # layout principal, combina els anteriors i permet activar i desactivar-los
    layout = [[sg.Text('ATENCIO! Vigila els voltans per seguritat', justification="c", font="Helvetica 30",
                       pad=(125, 0), size=(40, 1), key='text_superior')],
              [sg.Column(layout1, justification="t", element_justification='center', key='-COL1-'),
               sg.Column(layout2, justification="t", element_justification='center', visible=False, key='-COL2-
'),
               sg.Column(layout3, justification="t", element_justification='center', visible=False, key='-COL3-
'),
               sg.Column(layout4, justification="t", element_justification='center', visible=False, key='-COL4-
'),
               sg.Column(layout_moviment, justification="t", element_justification='center', visible=False,
                         key='-COL5-')],
              [sg.Button("-SEC-",visible=False), sg.Button('-THREAD-',visible=False)],
              [sg.Column(layout5, justification="r", element_justification='center', visible=True, key='-COL6-
')]]
```

```
    #definicio de la finestra amb les dimensions correctes
    window = sg.Window('Argos Cam', layout, resizable=False, size=(1024, 600)).Finalize()

    timeout = thread = None

    acelerar_barra_progreso = False
    timeout = None
    t = 50 #segons maxims de temps d'iniciacio
    c = "a" #argument necesari perque funcioni l'entrega de parametre en coa

    #treads per a la pantalla de carrega
    thread = threading.Thread(target=lambda q, arg1: q.put(connect(arg1)), args=(que, c),
daemon=True)
    thread1 = threading.Thread(target=tiempo, args=(t, window, thread), daemon=True)
    thread.start()
    thread1.start()


    #element per a poder carregar la barra des del punt que es queda el thread
    time_ = 0
    #loop de la pantalla de carrega
    while True:  # Event Loop
        event, values = window.read(timeout=10)
        if event in (None, 'Exit', 'Cancel') or time_ > 100: #si el temps acaba o es surt es tanca el loop
            break

        if event == '-SEC-': #click intern del boto invisible per augmentar la barra de progres
            #un cop s'ha connectat la barra de progres accelera
            if acelerar_barra_progreso:
                time_ = time_ + 1
                window['progbar'].update_bar(time_, 100)
                time.sleep(.1)
                window['-SEC-'].click()
            else:
                time_ = time_ + 1
            window['progbar'].update_bar(time_, 100)

        # indica que ha acabat el thread, obte la variable connexió
        if event == '-THREAD-':
            #tanca els threads
            thread.join(timeout=0)
            connexio = que.get()
            #connexio.watch(obd.commands.SPEED, callback=new_value1)
            #connexio.start()
            thread1.join(timeout=0)
            print('Thread finished', thread.is_alive(), thread1.is_alive())
            acelerar_barra_progreso = True #aumenta velocitat de carrega de la barra
            window['-SEC-'].click()
```

```
# break

#defineix les dimensions de les càmeres a les finestres depenent del mode selecciat
dispW = 640
dispH = 480
altura_ = 400
ample_1_ = 650
ample_2_ = 490

#defineix els pins de GPIO usats
Jetson = GPIO
Jetson.setmode(GPIO.BCM)
pins = [18, 17, 7, 27, 10]
# set pin as an output pin with optional initial state of HIGH
Jetson.setup(pins, GPIO.OUT, initial=GPIO.LOW)

#inicialitza la xarxa neuronal que s'entra a les càmares
xarxa = inference.detectNet("ssd-mobilenet-v2", threshold=0.5)

#es creen els objectes Vstream amb els parametres necessaris i s'inicia la captura de frames
camara1 = vStream(xarxa, dispW, dispH, Jetson, 18, 17, 7)
camara1.open(gstreamer_pipeline(flip_method=2))
camara1.start()
camara2 = vStream(xarxa, dispW, dispH, Jetson, 27, 10, 7)
camara2.open(gstreamer_pipeline(sensor_id=1, flip_method=2))
camara2.start()

# comença a captar el valor de la velocitat
connexio.watch(obd.commands.SPEED, callback=new_value1)
connexio.start()

# comença a captar el valor de la velocitat
window[f'-COL1-'].update(visible=False)
layout_visible = 3
window[f'-COL{layout_visible}-'].update(visible=True)

# Fa visibles els botons per canviar la camara que es veu
window['Esquerra'].update(disabled=False)
window['Dues'].update(disabled=False)
window['Dreta'].update(disabled=False)
window['-IA-'].update(disabled=False)

# tamany de redimensio de camara
dim_1 = (ample_1_, altura_)
dim_2 = (ample_2_, altura_)
old_slider = 10
layout_moviment = True

#variable per controlar l'algoritme actiu
```

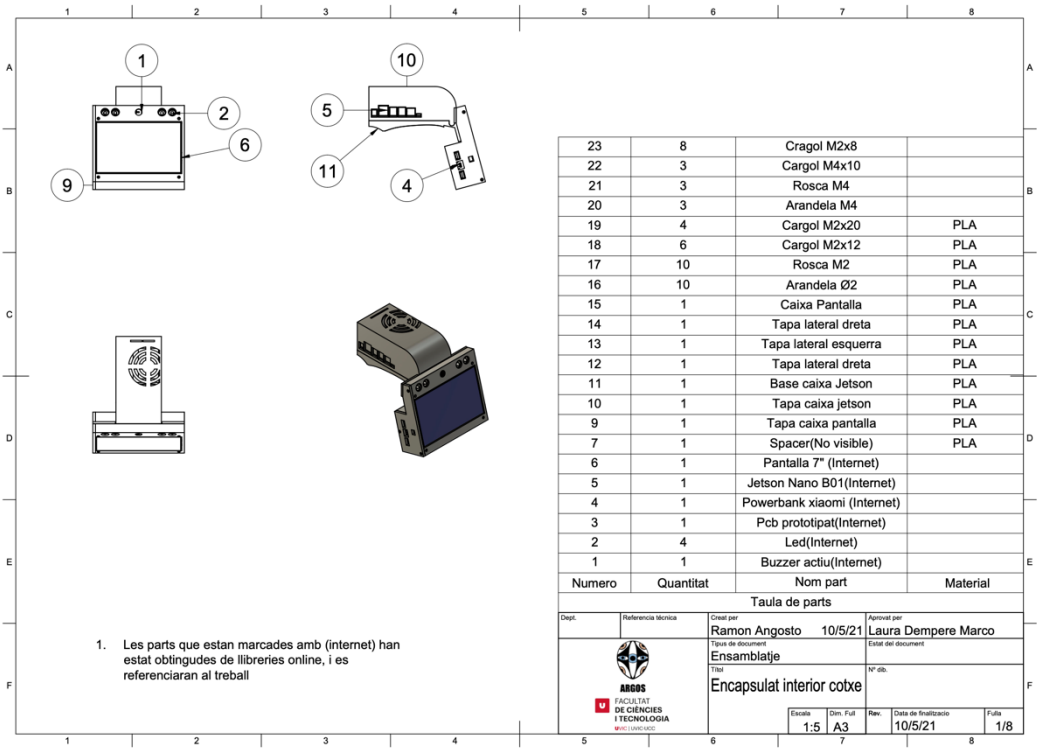
```
    Inteligent = 2  # 0 desactivada, 1 sense BG, 2 amb BG

    # loop principal
    while True:
        event, values = window.read(timeout=20)
        #event, values = window.read()
        if event == 'Exit' or event == sg.WIN_CLOSED or cv2.waitKey(1) == ord(
                'q'):  # si es clica exit o es tanca la finestra s'acaba el programa
            break

        elif event == 'Esquerra':  # es clica el boto de camara esquerra, es capturaran els frames
d'aquesta camara
            #desactiva layout actual i activa el pertinent
            window[f'-COL{layout_visible}-'].update(visible=False)
            layout_visible = 2
            window[f'-COL{layout_visible}-'].update(visible=True)

        elif event == 'Dues':  # es capturen les dues camares
            # desactiva layout actual i activa el pertinent
            window[f'-COL{layout_visible}-'].update(visible=False)
            layout_visible = 3
            window[f'-COL{layout_visible}-'].update(visible=True)

        elif event == 'Dreta':  # es clica el boto de camara dreta, es capturaran els frames d'aquesta
camara
            # desactiva layout actual i activa el pertinent
            window[f'-COL{layout_visible}-'].update(visible=False)
            layout_visible = 4
            window[f'-COL{layout_visible}-'].update(visible=True)

        elif event == '-IA-':
            # selecciona l'algoritme que s'usa per la captura d'imatges i actualitza el text del boto
            if Inteligent == 0:
                window['-IA-'].update('Activar Mode 2 IA')
                Inteligent = 1

            elif Inteligent == 1:
                window['-IA-'].update('Desactivar IA')
                Inteligent = 2

            else:
                window['-IA-'].update('Activar Mode 1 IA')
                Inteligent = 0



        if velocitat_alta:
            # modifica el layout visble en funcio de la variable global velocitat alta
            # Es fa automatic, no es requereix que el conductor faci click a cap boto
```

```
            window[f'-COL{layout_visible}-'].update(visible=False)
            window['-COL5-'].update(visible=True)
            layout_moviment = True

        else:
            window['-COL5-'].update(visible=False)
            window[f'-COL{layout_visible}-'].update(visible=True)
            layout_moviment = False

        if layout_moviment == False:  # si el cotxe esta per sota de 2 km/h es capturen imatges i es
mostren
            if layout_visible == 2:  # mostra la càmera esquerra
                if Inteligent == 2:
                    reconeixement, frame = camara1.reconocimiento_con_bg()

                elif Inteligent == 1:
                    reconeixement, frame = camara1.reconocimiento_sin_bg()
                else:
                    frame = camara1.getFrame()

                # codificació de l'imatge per ser usada a pysimplegui
                imgbytes = cv2.imencode('.png', cv2.resize(frame, dim_1))[1].tobytes()
                window['image1'].update(data=imgbytes)

            elif layout_visible == 4:  # mostra la càmera dreta
                if Inteligent == 2:
                    reconeixement1, frame1 = camara2.reconocimiento_con_bg()

                elif Inteligent == 1:
                    reconeixement1, frame1 = camara2.reconocimiento_sin_bg()

                else:
                    frame1 = camara2.getFrame()

                # codificació de l'imatge per ser usada a pysimplegui
                imgbytes1 = cv2.imencode('.png', cv2.resize(frame1, dim_1))[1].tobytes()  # ditto
                window['image4'].update(data=imgbytes1)

            else: # mostra les dues càmeres
                if Inteligent == 2:
                    reconeixement, frame = camara1.reconocimiento_con_bg()
                    reconeixement1, frame1 = camara2.reconocimiento_con_bg()

                elif Inteligent == 1:
                    reconeixement, frame = camara1.reconocimiento_sin_bg()
                    reconeixement1, frame1 = camara2.reconocimiento_sin_bg()

                else:
                    frame = camara1.getFrame()
```

```
            frame1 = camara2.getFrame()

            # codificació de l'imatge per ser usada a pysimplegui
            imgbytes = cv2.imencode('.png', cv2.resize(frame, dim_2))[1].tobytes()
            imgbytes1 = cv2.imencode('.png', cv2.resize(frame1, dim_2))[1].tobytes()
            window['image2'].update(data=imgbytes)
            window['image3'].update(data=imgbytes1)

    #neteja dels objectes usats per alliberar-los
    window.close()
    camara1.stop()
    connexio.close()
    camara1.capture.release()
    camara2.stop()
    camara2.capture.release()
    GPIO.cleanup()

#funcio principal
if __name__ == '__main__':
```

Annex 3: Codi principal

11.3 PLANOLS DE DISSENY

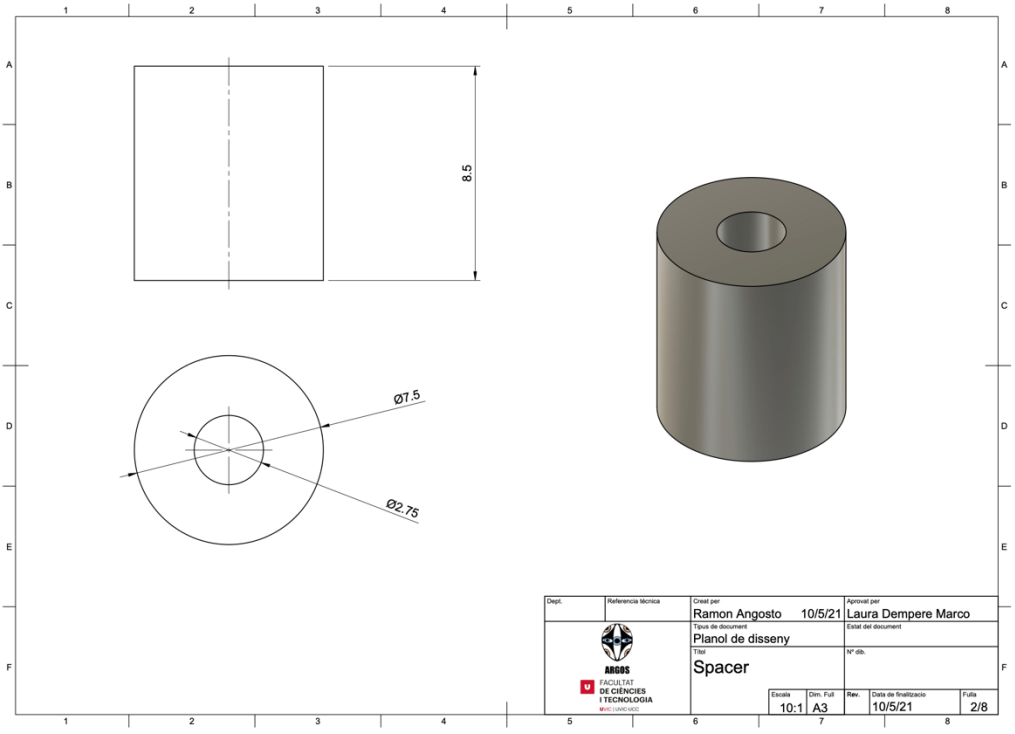


Numero	Quantitat	Nom part	Material
23	8	Cragol M2x8	
22	3	Cargol M4x10	
21	3	Rosca M4	
20	3	Arandela M4	
19	4	Cargol M2x20	PLA
18	6	Cargol M2x12	PLA
17	10	Rosca M2	PLA
16	10	Arandela Ø2	PLA
15	1	Caixa Pantalla	PLA
14	1	Tapa lateral dreta	PLA
13	1	Tapa lateral esquerra	PLA
12	1	Tapa lateral dreta	PLA
11	1	Base caixa Jetson	PLA
10	1	Tapa caixa jetson	PLA
9	1	Tapa caixa pantalla	PLA
7	1	Spacer(No visible)	PLA
6	1	Pantalla 7" (Internet)	
5	1	Jetson Nano B01(Internet)	
4	1	Powerbank xiaomi (Internet)	
3	1	Pcb prototipat(Internet)	
2	4	Led(Internet)	
1	1	Buzzer actiu(Internet)	

Taula de parts

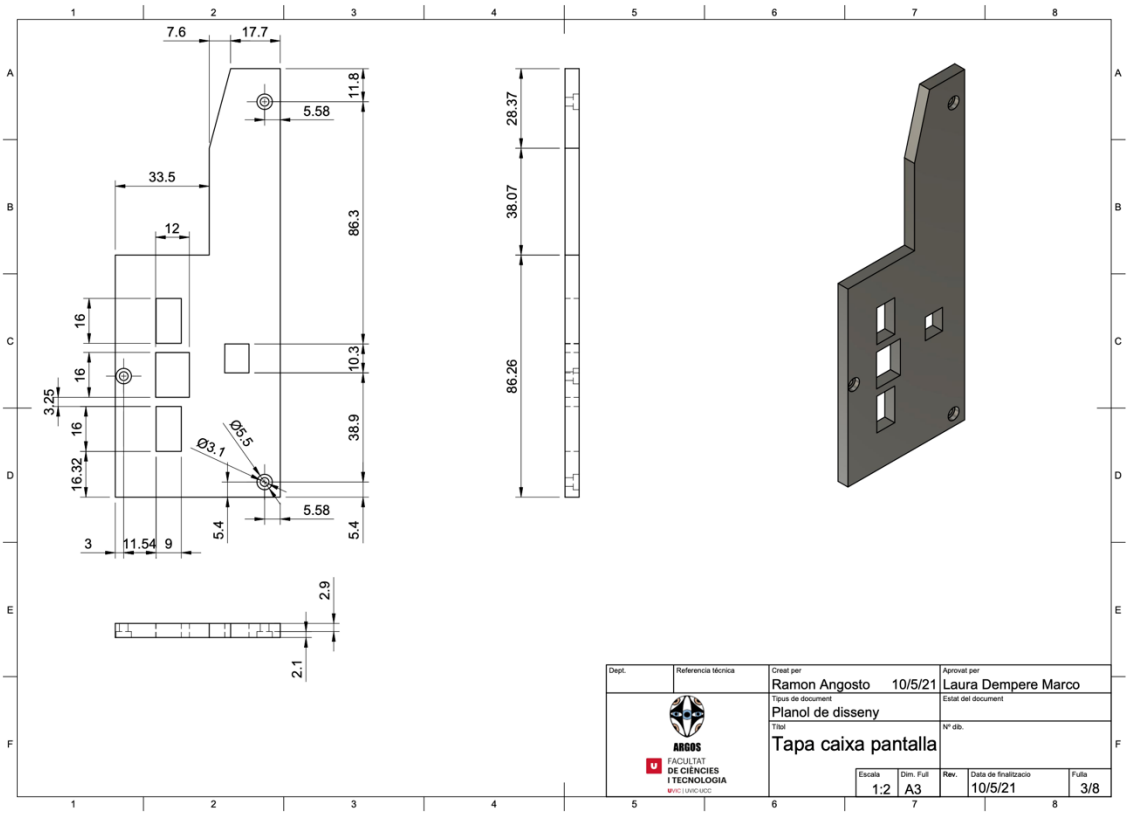
1. Les parts que estan marcades amb (internet) han estat obtingudes de llibreries online, i es referenciaran al treball

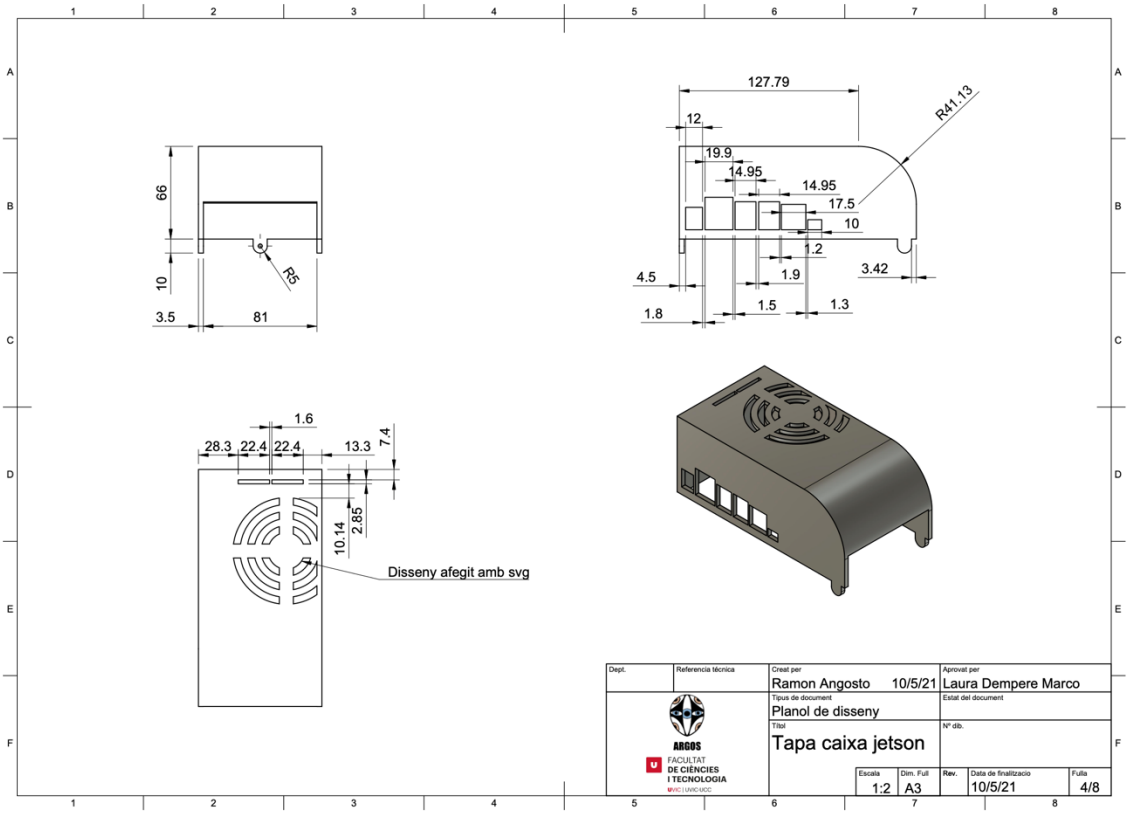
Dept.	Referencia técnica	Creat per: Ramon Angosto 10/5/21	Aprovat per: Laura Dempere Marco
		Tipus de document: Ensamblatje	Estat del document
ARGOS FACULTAT DE CIÈNCIES I TECNOLOGIA UVIC | UVIC·UCC		Títol: Encapsulat interior cotxe	Nº dib.
		Escala: 1:5 / Dim. Full: A3	Rev. / Data de finalització: 10/5/21 / Fulla: 1/8

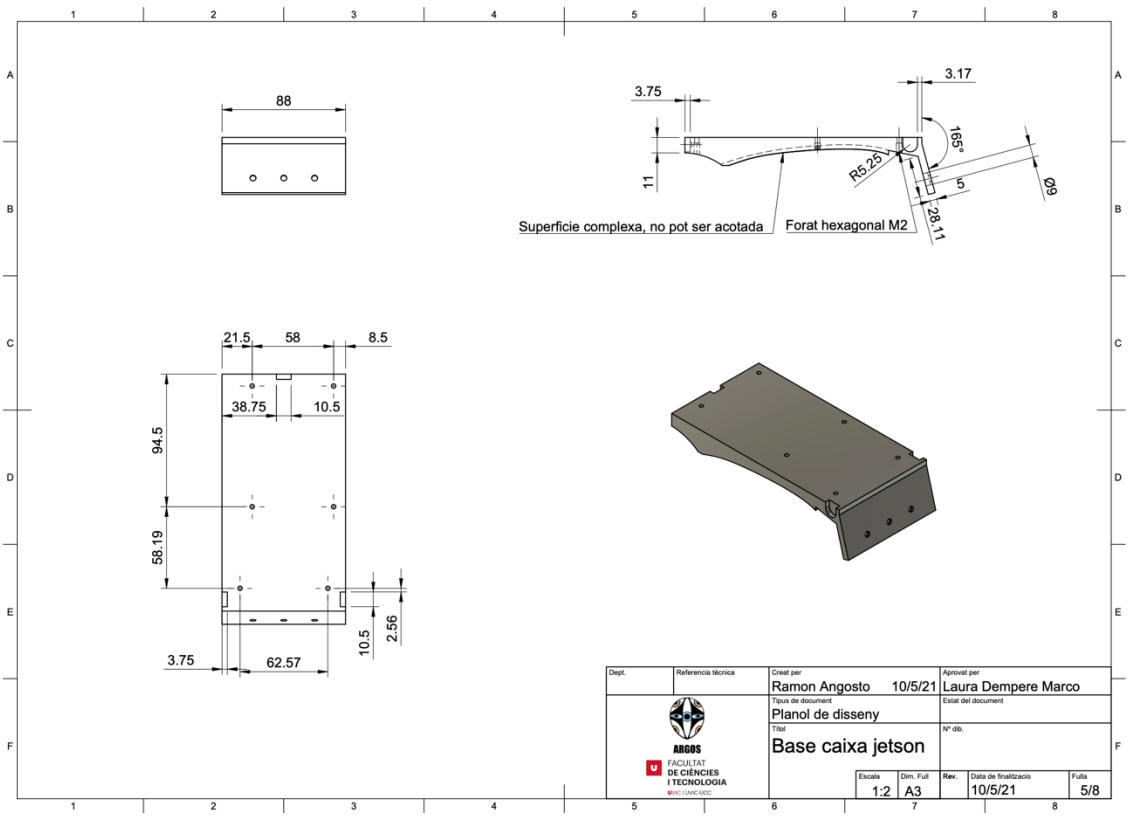


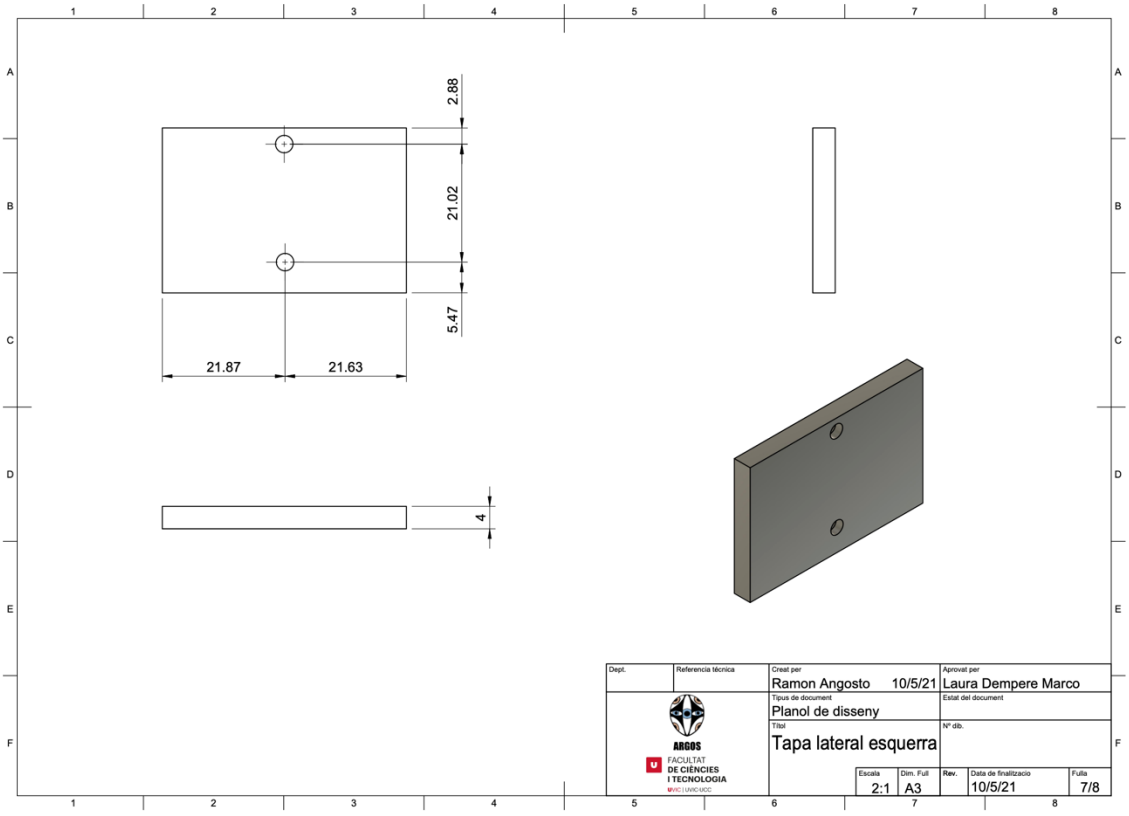
Dept.	Referencia técnica	Creat per: Ramon Angosto 10/5/21	Aprovat per: Laura Dempere Marco
		Tipus de document: Planol de disseny	Estat del document
ARGOS FACULTAT DE CIÈNCIES I TECNOLOGIA UVIC | UVIC·UCC		Títol: Spacer	Nº dib.
		Escala: 10:1 / Dim. Full: A3	Rev. / Data de finalització: 10/5/21 / Fulla: 2/8

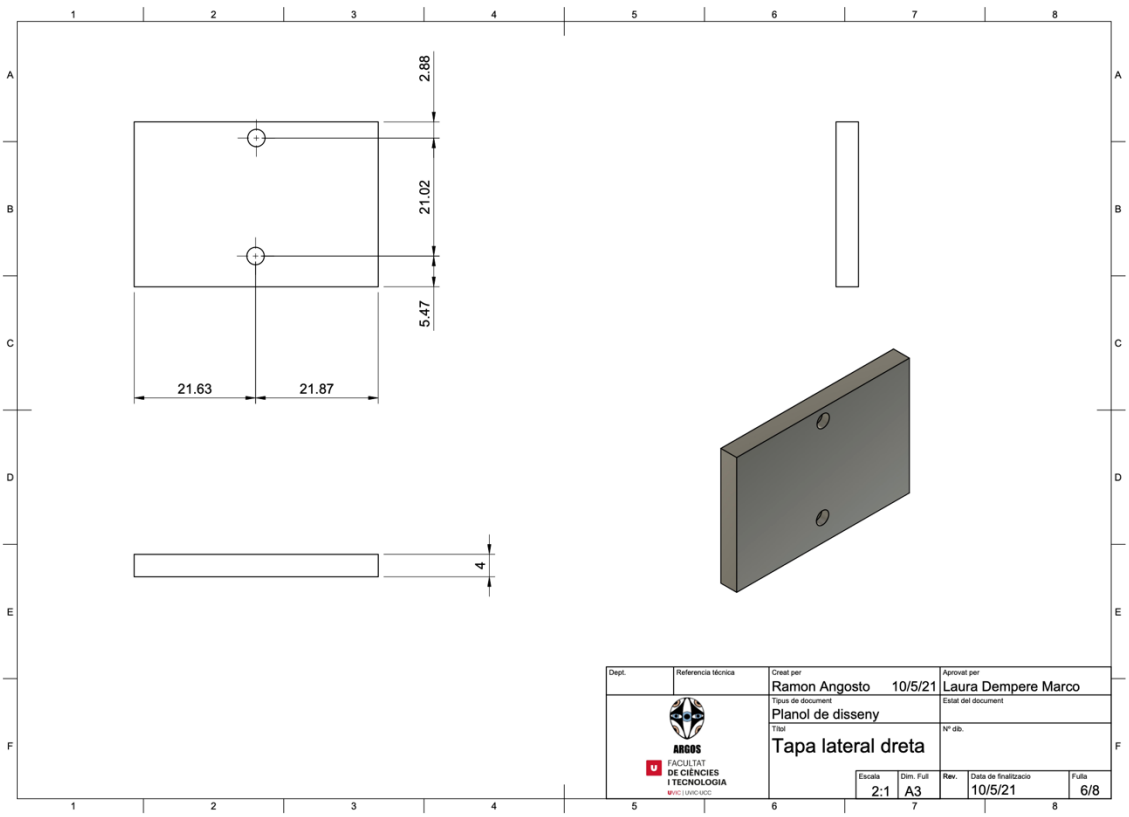
Annex 4: Plànols del disseny 3D del projecte

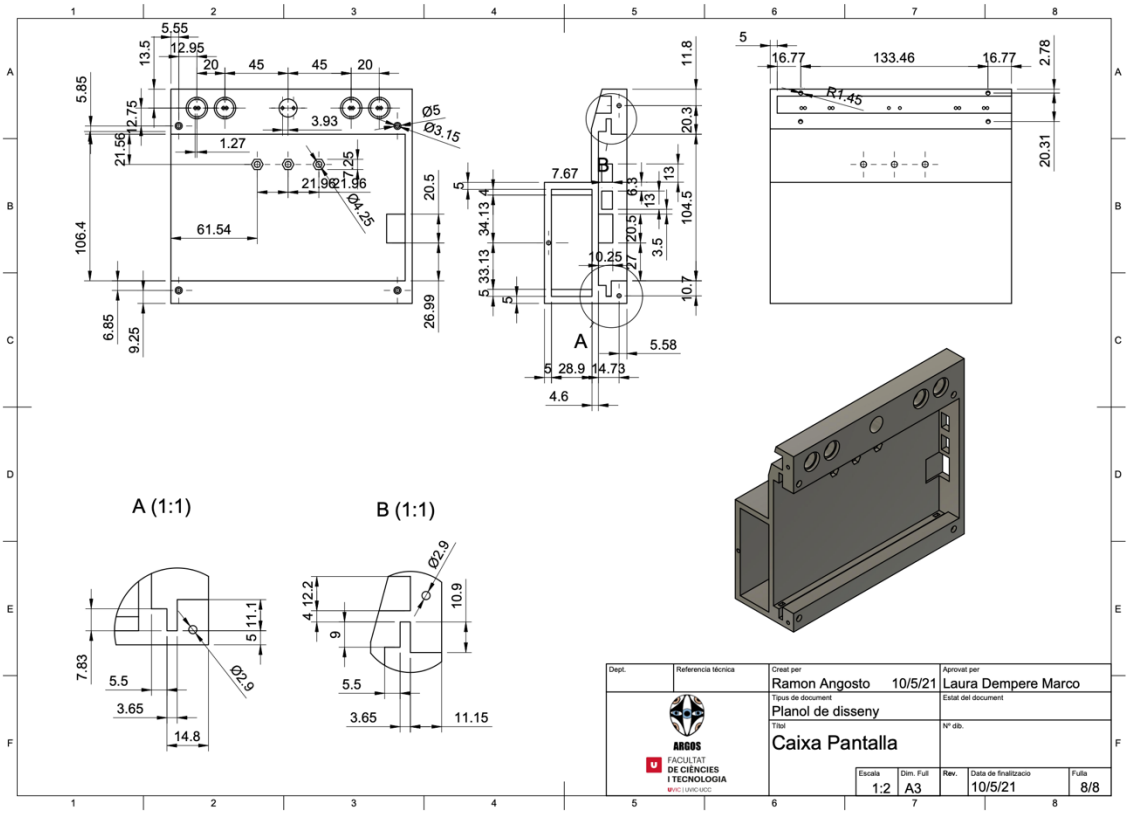

7.6
17.7
11.8
5.58
33.5
12
86.3
16
16
10.3
3.25
16
38.9
Ø5.5
Ø3.1
16.32
5.58
5.4
5.4
3
11.54
9
28.37
38.07
86.26
2.9
2.1
Dept.
Referència tècnica
Creat per
Ramon Angosto 10/5/21
Aprovat per
Laura Dempere Marco
Tipus de document
Planol de disseny
Estat del document
Títol
Tapa caixa pantalla
Nº dib.
Escala 1:2
Dim. Full A3
Rev.
Data de finalització 10/5/21
Fulla 3/8
ARGOS
FACULTAT DE CIÈNCIES I TECNOLOGIA
UVIC | UVIC·UCC

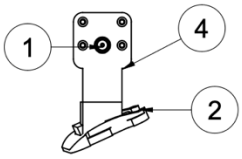

66
10
R5
3.5
81
127.79
12
R41.13
19.9
14.95
14.95
17.5
10
1.2
3.42
4.5
1.9
1.8
1.5
1.3
1.6
28.3
22.4
22.4
13.3
7.4
10.14
2.85
Disseny afegit amb svg
Dept.
Referència tècnica
Creat per
Ramon Angosto 10/5/21
Aprovat per
Laura Dempere Marco
Tipus de document
Planol de disseny
Estat del document
Títol
Tapa caixa jetson
Nº dib.
Escala 1:2
Dim. Full A3
Rev.
Data de finalització 10/5/21
Fulla 4/8
ARGOS
FACULTAT DE CIÈNCIES I TECNOLOGIA
UVIC | UVIC·UCC

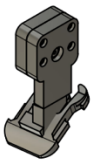

88
3.75
3.17
165°
R5.25
11
Ø9
28.11
Superficie complexa, no pot ser acotada
Forat hexagonal M2
21.5
58
8.5
38.75
10.5
94.5
58.19
10.5
2.56
3.75
62.57
Dept.
Referència tècnica
Creat per
Ramon Angosto 10/5/21
Aprovat per
Laura Dempere Marco
Tipus de document
Planol de disseny
Estat del document
Títol
Base caixa jetson
Nº dib.
Escala 1:2
Dim. Full A3
Rev.
Data de finalització 10/5/21
Fulla 5/8


2.88
21.02
5.47
21.87
21.63
4
Dept.
Referència tècnica
Creat per
Ramon Angosto 10/5/21
Aprovat per
Laura Dempere Marco
Tipus de document
Planol de disseny
Estat del document
Títol
Tapa lateral esquerra
Nº dib.
Escala 2:1
Dim. Full A3
Rev.
Data de finalització 10/5/21
Fulla 7/8


2.88
21.02
5.47
21.63
21.87
4
Dept.
Referència tècnica
Creat per
Ramon Angosto 10/5/21
Aprovat per
Laura Dempere Marco
Tipus de document
Planol de disseny
Estat del document
Títol
Tapa lateral dreta
Nº dib.
Escala 2:1
Dim. Full A3
Rev.
Data de finalització 10/5/21
Fulla 6/8


A (1:1)
B (1:1)
Dept.
Referència tècnica
Creat per
Ramon Angosto 10/5/21
Aprovat per
Laura Dempere Marco
Tipus de document
Planol de disseny
Estat del document
Títol
Caixa Pantalla
Nº dib.
Escala 1:2
Dim. Full A3
Rev.
Data de finalització 10/5/21
Fulla 8/8





Numero	Quantitat	Nom part	Material
10	2	Cargol M4x10	
9	2	Arandela M4	
8	2	Rosca M4	
7	4	Cargol M2x16	
6	4	Arandela M2	
5	4	Rosca M2	
4	2	Part posterior carcasa	PLA
3	2	Part frontal carcasa	PLA
2	2	Clip Gopro (Internet)	PLA
1	2	Raspi Cam v2 (Internet)	

Llista de parts

Dept.	Referència tècnica	Creat per	Aprovat per

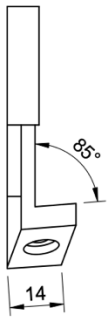
FACULTAT DE CIÈNCIES I TECNOLOGIA
UVIC | UVIC·UCC

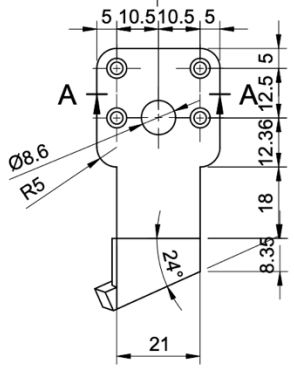
Tipus de document	Estat del document
Ensamblatje	Acabat

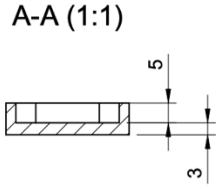
Títol: Càmera TFG

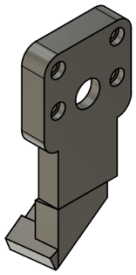
Nº Dib.:

Escala	Dim. full	Rev.	Data acabat	Fulla
1:2	A4		11/5/21	

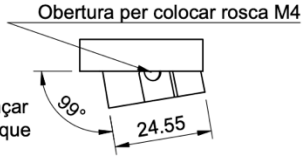








1. La Peça te una inclinacio de 24º per compençar l'inlclinació del capo, 9º de rotacio lateral perque apunti una mica cap endavant i 5 perque no apunti al terra.



Dept.	Referència tècnica	Creat per	Aprovat per
		Ramon Angosto 11/5/21	Laura Dempere Marco

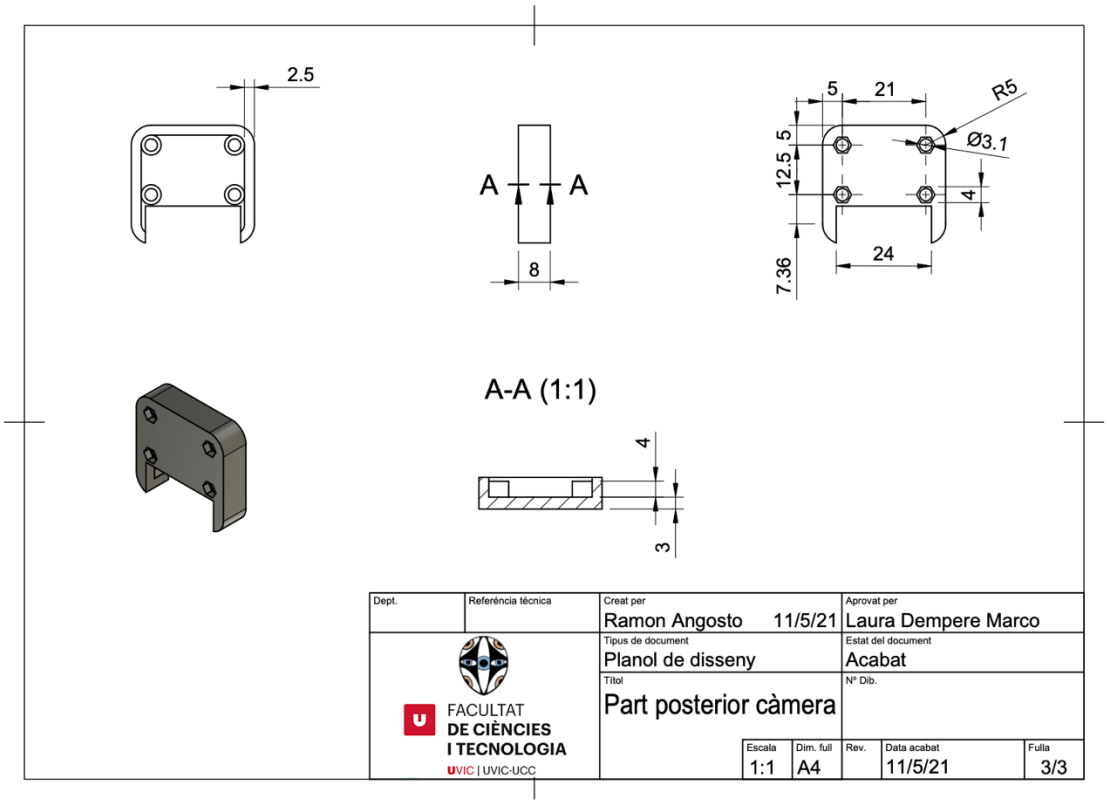
FACULTAT DE CIÈNCIES I TECNOLOGIA
UVIC | UVIC·UCC

Tipus de document	Estat del document
Planol de disseny	Acabat

Títol: Part frontal càmera

Nº Dib.:

Escala	Dim. full	Rev.	Data acabat	Fulla
1:1	A4		11/5/21	2/3



Dept.	Referència tècnica	Creat per		Aprovat per
		Ramon Angosto	11/5/21	Laura Dempere Marco
FACULTAT DE CIÈNCIES I TECNOLOGIA UVIC | UVIC·UCC		Tipus de document: Planol de disseny		Estat del document: Acabat
		Títol: Part posterior càmera		Nº Dib.

Escala	Dim. full	Rev.	Data acabat	Fulla
1:1	A4		11/5/21	3/3