

PROYECTO FINAL DE CARRERA

SISTEMA PARA REDUCIR EL RIESGO DE ACCIDENTES EN INTERSECCIONES
(ARGOS[1] CAM)



ARGOS

Nombre: Ramon Angosto Artigues

Tema: Proyecto final

Grado: Ingeniería mecatrónica.

Profesor/a: Laura Dempere-Marco

27 octubre de 2021



FACULTAT
**DE CIÈNCIES
I TECNOLOGIA**
UVIC | UVIC-UCC

Ingeniería mecatrónica

Trabajo final de carrera

ÍNDICE DE CONTENIDO

Índice de contenido	3
Endex dellustraciones	4
Index de lasaulas T	5
1. Abstract	6
2. Introducció	8
3. Motivació	10
4. Objectivos	10
5. Estado del arte	11
6. Metodología	17
7. Resultados	36
8. Discusión	46
9. Conclusión	48
10. Bibliografía	49
11. Anexo	54

ENDEX DE ILUSTRACIONES

Ilustración 1: <i>Pipeline</i> del proyecto	9
Ilustración 2: Distancia entre el conductor y la parte delantera del coche	9
Ilustración 3: Lector con cable Obd II (Imagen obtenida de [25])	17
Ilustración 4: Raspberry piCamera V2 (Imagen obtenida de [27])	18
Ilustración 5: Posición de la cámara en el coche	18
Ilustración 6: Posición del coche para la captura de imágenes	19
Ilustración 7: Ejemplos de capturas de imágenes en diferentes esquinas	20
Ilustración 8: Representación de conexiones de componentes con el Jetson Nano. Los materiales utilizados para el proyecto son: dos cámaras Raspberry Pi v2, utilizadas para la captura de imágenes; una pantalla táctil para ver imágenes e interactuar con la interfaz de usuario; un lector OBD utilizado para la recogida de datos del vehículo; una batería portátil de la marca Xiaomi, con capacidad de alimentación de 5v 2.6A; para las conexiones GPIO, se utilizan cuatro LED y un <i>buzzer</i> (con los componentes adicionales necesarios)	22
Ilustración 9: Ejemplo K-NN obtenido de [32].	25
Ilustración 10: A) Imagen original B) NSamples con valor 10 C) NSamples con valor 3	26
Ilustración 11: Matrices que componen los <i>Kernels</i>	27
Ilustración 12: Ejemplo del efecto de <i>Opening</i> (Fuente: Referencia [33])	28
Ilustración 13: Ejemplo de función de <i>Closing</i> (Fuente: Referencia [33])	28
Ilustración 14: Ejemplo de la función <i>Dilate</i> (Fuente: Referencia [33])	29
Ilustración 15: Etapas del Background subtraction a) Imagen original b) Blur c) Algoritmo KNN d) <i>Openning</i> E) <i>Closing</i> F) <i>Dilate</i> G) Find Contours & Bounding Box	30
Ilustración 16: Representación figurativa de la capa <i>depthwise separable</i> (Fuente: Referencia [37])	31
Ilustración 17: Estructura de red MobileNetSSD-v2 (Fuente: Referencia [37])	31
Ilustración 18: Pantallas GUI. A) Pantalla de carga B) Pantalla mientras el vehículo está en movimiento C) Pantalla con una sola cámara D) Pantalla con ambas cámaras	33
Ilustración 19: Circuito electrónico generado con KiCad	34

Ilustración 20: Piezas de la carcasa de la pantalla, la batería y las placas. A) Cubierta de la carcasa de la placa B) Base de la carcasa para la placa C) y D) Tapa parte trasera carcasa pantalla E) Carcasa de la pantalla y la batería	35
Ilustración 21: Carcasa de cámara derecha.	35
Ilustración 22: Ejemplificación de la GUI con el coche parado y con el coche en movimiento	36
Ilustración 23: Visión de las cámaras desde el punto de vista del conductor	37
Ilustración 24: Comparación de máscaras de algoritmos de <i>Background Subtraction</i>	38
Ilustración 25: Ejemplos de <i>Backgroun Subtraction</i> aplicada	40
Ilustración 26: Ejemplos de advertencias luminosas en diferentes situaciones	45
Ilustración 27:Sistema de levas ARGOS montado en un coche	45

INDEX DE LASAULAS T

Tabla 1: Evaluación de los algoritmos de <i>Backgroud subtraction</i>	38
Tabla 2: Comparación de algoritmos de detección de objetos [35],[11],[22], [40]	41
Tabla 3: Velocidad de procesamiento del reconocimiento de objetos en diferentes situaciones	42
Tabla 4: Número de muestra por clase y valores TP, FP y FN	43
Tabla 5: Resultados de la evaluación del reconocimiento de objetos	44

1. ABSTRACT

1.1 CASTELLANO

En el siguiente Trabajo Fin de Grado se expone el proyecto realizado para el desarrollo de un dispositivo que tiene como objetivo aumentar la visibilidad de los vehículos en las intersecciones con el fin de evitar accidentes. En concreto, el trabajo explicará los métodos utilizados para identificar posibles fuentes de peligro, la integración del dispositivo en el vehículo y el método de notificación al conductor.

El proyecto cuenta con dos líneas metodológicas principales: 1) adquisición, procesamiento de imágenes y reconocimiento de objetos, y 2) diseño del dispositivo electrónico. Estas dos líneas de trabajo se integran en un dispositivo funcional que, una vez procesadas las imágenes, junto con el análisis del estado de movimiento del vehículo, emite avisos sonoros y luminosos indicando la presencia de vehículos o vehículos acercándose por los laterales.

Para ello, en primer lugar, se han adquirido una serie de vídeos que se han utilizado para la realización de las pruebas, previas a la integración en el vehículo. En cuanto a la programación, esta se ha llevado a cabo con un Jeston Nano y el lenguaje Python junto con una serie de librerías *open source*, entre las que destaca OpenCV. El diseño del dispositivo se llevó a cabo con el programa de diseño 3D, Fusion360.

Por lo tanto, el objetivo principal del proyecto es aumentar la visibilidad del conductor cuando está en una intersección tratando de girar. El dispositivo construido permite aumentar la visibilidad del conductor mediante el uso de dos cámaras ubicadas en la parte delantera del coche e indicará la presencia de objetos peligrosos.

Los resultados obtenidos son prometedores. Por un lado, mostrar la imagen de la parte frontal del coche ya representa una gran ventaja de visibilidad. Además, en los vídeos utilizados para realizar las pruebas, en días soleados o nublados y en las pruebas realizadas fuera del coche, se han obtenido resultados satisfactorios, ya que se avisa al conductor correctamente cuando se acerca un vehículo o peatón. Por otro lado, en condiciones adversas, como lluvia y niebla, se han observado algunos problemas de reconocimiento de vehículos y personas. Además, debido a la limitación de potencia al conectar la placa a una batería portátil para incorporarla al coche no ha tenido el funcionamiento esperado y no ha dado los avisos correctamente.

En definitiva, el proyecto ha mostrado resultados que sugieren que este dispositivo, con un poco más de desarrollo, podría incorporarse como otro sistema avanzado de asistencia a la conducción.

1.2 INGLÉS

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 hob 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. INTRODUCCIÓN

Este proyecto final describe el desarrollo y prototipado de un sistema para aumentar la visibilidad de los conductores en las intersecciones con el fin de reducir el número de accidentes que se producen en estas zonas.

La visibilidad reducida aumenta el riesgo de colisiones con vehículos que se acercan a la calle perpendicular a la que se está circulando. Sobre todo, si el coche tiene que entrar en la vía que se quiere incorporar para tener visibilidad. Este problema se da principalmente en zonas rurales donde las calles son más estrechas y en las salidas de los aparcamientos donde muchas veces el coche tiene que cruzar la acera para entrar en la calle, existiendo el riesgo de atropellar a un peatón.

Según la DGT el número de accidentes anuales en estas intersecciones es de aproximadamente 12.000 en carreteras interurbanas y 55.000 en vías urbanas, esta información pertenece al grupo 7 de la DGT, donde el número de accidentes se especifica según el tipo de vía [2].

El proyecto desarrolla un sistema que se llamará ARGOS CAM, refiriéndose a la mitología griega en Argos Panoptes [1], que era un guardián y sirviente de Hera que tenía muchos ojos, lo que lo hacía muy eficiente para la vigilancia porque siempre tenía uno abierto para vigilar. El logotipo del proyecto es una referencia a este mito.

ARGOS Cam consta de dos partes principales: una que se encarga de la recopilación de datos del vehículo (es decir, la velocidad) y una segunda parte que se encarga de capturar imágenes. Este segundo módulo -centrado en el procesamiento de imágenes- se encarga de ofrecer ayuda al conductor, a través del reconocimiento de imágenes, y está modulado por el primero, que se centra más en aspectos de la electrónica del vehículo. La estructura de ARGOS Cam se presenta en la Figura 1, que a su vez incluye la estructura del trabajo:

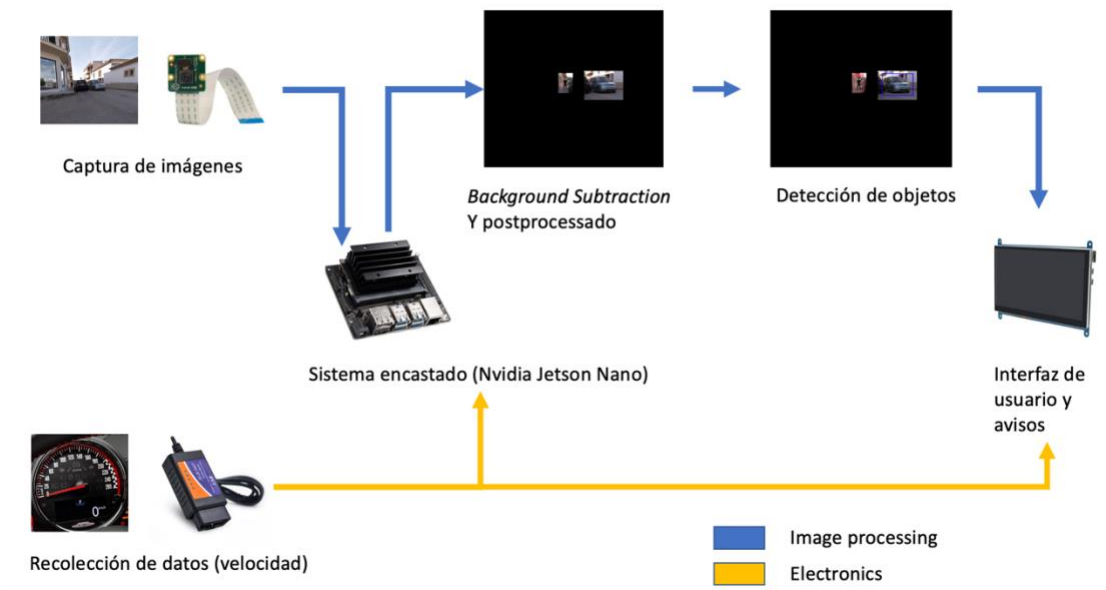


Ilustración 1 Pipeline del proyecto

En la figura 2 se puede ver una imagen que representa la ventaja del sistema para el conductor, si se colocan dos cámaras en la parte delantera del coche.

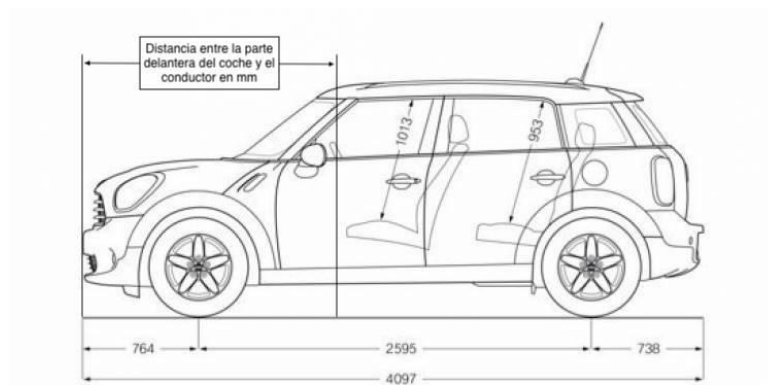


Ilustración 2: Distancia entre el conductor y la parte delantera del coche

Cabe decir que, además de la ayuda visual intrínseca que supondrán las cámaras, se utilizarán algoritmos de procesamiento de imágenes para avisar al conductor si aparece algún elemento al que se le tenga que prestar atención, como un coche que se acerque o un peatón que esté cruzando la calle.

3. MOTIVACIÓN

La idea del proyecto parte de la identificación de un problema que se ha observado conduciendo, y como ya se ha mencionado en la introducción, es difícil ver si otros vehículos se acercan por los laterales en las intersecciones. Es cierto que en algunas intersecciones hay espejos que pretenden solucionar estos problemas, pero no siempre están ahí y suelen estar en mal estado, lo que limita su uso.

Después de identificar el problema he llevado a cabo una investigación para ver el número de accidentes en estas intersecciones, y el alarmante número de 67.000 accidentes al año de este tipo se ha convertido en la otra razón que me ha motivado a llevar a cabo este proyecto, para tratar de reducir este número de accidentes y hacer la vida de las personas más fácil.

Además de intentar hacer la vida más sencilla a las personas, el hecho de trabajar con algoritmos *Deep Learnig* es un reto que me estimula y me permite aplicar los conocimientos adquiridos a lo largo de mis estudios en ingeniería mecatrónica, y me ofrece la oportunidad de profundizar en conocimientos que pueden ser útiles para un futuro lugar de trabajo en el entorno de programación.

4. OBJETIVOS

Como se menciona en la motivación de este proyecto, el objetivo general es reducir el número de accidentes que se producen en las intersecciones debido a la mala visibilidad.

En concreto, se quiere desarrollar un dispositivo que se pueda integrar en el coche, que permita la visualización de la calle a la que el conductor quiere incorporarse sin tener que sacar la parte delantera del coche y que se notifique al conductor si se acerca otro vehículo o hay un peatón cruzando o con intención de hacerlo.

Asimismo, el proyecto consta de dos objetivos específicos. En primer lugar, la recogida de imágenes y datos obtenidos del vehículo, junto con algoritmos de inteligencia artificial para ayudar al conductor en este tipo de vías. En segundo lugar, el diseño de una interfaz gráfica y un sistema de alerta para incorporar al vehículo.

Los objetivos examinados pueden ordenarse por orden de prioridad de la siguiente manera:

1. Reducir el número de accidentes en este tipo de carreteras como consecuencia de la mala visibilidad
2. Hacer uso de las cámaras y la información proporcionada por los sensores del coche, junto con algoritmos de inteligencia artificial para ayudar en la conducción.
3. Diseñar un dispositivo con sistema de alerta para su incorporación al vehículo

5. ESTADO DEL ARTE

En este punto, se discutirá el estado actual de la literatura que existe sobre el uso de dispositivos de asistencia a la conducción y cada uno de los puntos mencionados en la introducción que conforman el proyecto.

En primer lugar, los sistemas avanzados de asistencia al conductor, también conocidos como ADAS [3], existen en el sector de la automoción desde 1950, lo que hace que la conducción sea más segura. Algunos ejemplos de estos primeros usos fueron: sensores de exceso de velocidad y el sistema de frenado ABS. Actualmente estos sistemas están integrados en todos los vehículos, y la Comisión Europea propone hacer obligatoria la incorporación de más sistemas asistenciales para reducir el número de accidentes.

Hoy en día, han surgido muchos más de estos sistemas que ayudan a evitar accidentes. Además de la asistencia en carretera, también están relacionadas con la asistencia al conductor dentro de las ciudades. Los sistemas de frenado automático por proximidad, con sensores de ultrasonido y LIDARS (*Light Detection And Ranging*¹) y los sistemas de asistencia al aparcamiento -con cámaras y sensores de proximidad- son dos de los ejemplos más claros en este caso de uso. La empresa Nissan presentó en 2007 un concepto de coche que había incorporado 4 cámaras alrededor del coche que permitía simular una vista dron del mismo para asistir durante el aparcamiento, además de permitir seleccionar las cámaras que se veían en la pantalla [4]. También incorporó sistemas de ultrasonido para alertar a objetos cercanos.

Los coches autónomos trabajan principalmente con LIDARS, cámaras o la combinación de estas dos tecnologías (o similares)[5]. Esta combinación se conoce como sensor fusión y permite evitar algunos de los problemas que presenta cada tecnología.

¹ LIDAR: Sensor que permite determinar la distancia a un objeto o superficie mediante un láser pulsante. [16]

En este trabajo se apuesta por un enfoque similar al que adopto Nissan y al que utilizan los coches autónomos que utilizan cámaras y sistemas de inteligencia artificial.

5.1 RECOPIACIÓN DE DATOS DEL VEHÍCULO

En cuanto a la recogida de datos del vehículo, en 1989 se introdujo el puerto ODB I, a los coches americanos, que permitía el control del estado del motor a través de códigos de error. Y en 1996 se presenta el puerto OBD II, que además de permitir la visualización de códigos de error permite la lectura de datos del vehículo como velocidad, revoluciones y otros parámetros no relacionados con el motor. Desde 2003 este puerto es obligatorio en todos los vehículos. [6, 7]

5.2 CAPTURA DE IMÁGENES

Otro aspecto clave del sistema desarrollado es la captura de imágenes. En este sentido, existen dos tipos principales de conexiones, utilizadas en sistemas embebidos para el trabajo de conducción autónoma y para la integración en sistemas embebidos. Estos dos tipos son: MIPI CSI-2 y USB 3.0

Por un lado, la conexión CSI-2 MIPI es la más utilizada en sistemas embebidos y en telefonía móvil. Esto permite una transmisión de imágenes más rápida, 6 GB/s de ancho de banda, y más eficiente que el USB 3.0, ya que no ocupa parte de la CPU para funcionar. También tienden a tener un tamaño más pequeño tanto del cuerpo de la cámara como del cable. Por el contrario, presenta algunas desventajas como la necesidad de bibliotecas especiales y la dificultad de utilizar diferentes sensores en las cámaras [8].

Por otro lado, la conexión USB 3.0 alcanza hasta un ancho de banda de 5 Gb/s y solo necesita estar conectada para funcionar. Debido a que el tipo de conexión es más común, es más fácil encontrar un dispositivo compatible o cambiar un dispositivo roto o mejorar el sistema [8], [9].

Además de la conexión hay otros parámetros que influyen en la captura de imágenes, esto es si la cámara es monocromática o con un sensor de color, si está equipada con un disparador de sensor completo o rodante, si tiene autoenfoco o no, si está equipada con capacidades de visión nocturna, así como otros parámetros que no son relevantes para este proyecto [9].

Dado que este proyecto utilizará un sistema embebido, teniendo en cuenta lo que se ha comentado en este apartado, se ha optado por seleccionar una cámara con la conexión CSI-2 MIPI, sensor de color sin visión nocturna ni autoenfoco - ya que puede producirse un movimiento muy rápido delante de la cámara y el autoenfoco podría ser más lento que el movimiento que produce.

5.3 SISTEMA EMBEBIDO

Los sistemas embebidos son sistemas informáticos de propósito especial diseñados para realizar una o pocas tareas dedicadas, y que generalmente se limitan a la computación en tiempo real. Generalmente se integra como parte de un dispositivo completo, siendo la unidad de procesamiento de dispositivo completo. Estos sistemas integrados se encuentran en la mayoría de los dispositivos de consumo que se utilizan actualmente [10].

Hoy en día existe una amplia variedad de sistemas embebidos en el mercado, pero en la mayoría de los proyectos de prototipado a coste reducido la decisión se reduce a dos: la Jetson Nano [11] y la Raspberry [12]. Estas dos placas tienen la capacidad de ejecutar un sistema operativo y actuar como un equipo de pequeña capacidad.

Por un lado, la Raspberry fue creada en 2012 para enseñar programación y cómo interactuar con dispositivos electrónicos. Tiene una memoria RAM que oscila entre 1 GB y 8 GB, lo que permite ejecutar más o menos tareas paralelas, y una CPU de 4 núcleos. Por otro lado, la Jetson Nano es una placa diseñada por la empresa NVIDIA, diseñada para la ejecución de *machine learning* y programas de procesamiento de imágenes. Esta cuenta con dos versiones. Una de 2 GB de RAM y otra de 4 GB, la de 4 cuenta con la ventaja que tiene dos puertos CSI MIPI, para conectar dos cámaras en paralelo. También cuenta con una GPU, esto proporciona una gran capacidad para el cálculo paralelo de tareas complejas como el procesado de imagen y el *Machine Learning*.

Debido a que el proyecto requiere dos cámaras y se basa principalmente en el procesamiento de imágenes, se utilizará el sistema integrado Jetson Nano.

5.4 ALGORITMOS DE VISIÓN ARTIFICIAL PARA ASISTENCIA A LA CONDUCCIÓN EN INTERSECCIONES

En esta sección se repasan los principales algoritmos de visión por computadora que se han desarrollado para resolver las tareas que nos ocupan en este proyecto: *Background Subtraction* y detección de objetos.

5.4.1 BACKGROUND SUBTRACTION

Background Subtraction es una técnica que consiste en separar el fondo(*background*) de aquellos objetos que están en movimiento en un vídeo. Una de las funciones en las que más se utiliza es en la video vigilancia, ya que permite una detección más simplificada de los objetos que están en movimiento. El *Background Subtraction* consiste en una serie de etapas. Estas son: 1) la inicialización del fondo (donde normalmente se toma una primera imagen), 2) la inicialización del modelo (que se utilizará para la definición de lo que pertenece al fondo y lo que está en movimiento), 3) el mecanismo para actualizar el fondo, y 4) la clasificación de las imágenes que se introducen [13].

Tradicionalmente, los métodos más comúnmente utilizados para esta tarea son los *Gaussian Mixture Models (MoG)*[13], ya que estos métodos tienen en cuenta los cambios en la iluminación, el ruido de la imagen y los objetos de movimiento más lento.

Además de esto también existen otros modelos para la detección del fondo que han aparecido en los últimos años. Algunos de ellos se basan en algoritmos para segmentar el fondo a través de redes neuronales y segmentación semántica, que asigna cada píxel de la imagen a una clase [12, 13]. También hay investigadores que se han centrado en el uso de otras técnicas basadas en el *Machine Learning*, como *K-Nearest Neighbours*[13].

Antes de la decisión final de utilizar el algoritmo basado en *K-Nearest Neighbour (K-NN)*, se ha realizado una comparación de los diferentes algoritmos. En este trabajo, debido a las limitaciones en la extensión, no se muestra un análisis exhaustivo de todos los resultados obtenidos, pero sí las principales conclusiones extraídas de este estudio. Estas se presentan en la sección de discusión.

5.4.2 DETECCIÓN DE OBJETOS

La detección de objetos consiste en combinar dos tareas, clasificar un objeto que aparece en la imagen y localizarlo, es decir, decir en que posición se encuentran uno o más objetos en la imagen [16]. En el campo de la visión artificial, se han desarrollado diferentes métodos para llevar a cabo esta tarea, tanto desde la perspectiva de la visión por ordenador clásica como la basada en el *Deep Learning*. Dado que en este proyecto nos centraremos en este

segundo enfoque, repasamos en esta sección los principales métodos desarrollados en el campo de la detección de objetos.

Sin embargo, es necesario establecer primero algunos conceptos e introducir qué es el *Deep Learning*, así como introducir la noción de *Transfer Learning*. El *Deep Learning* consiste en el uso de estructuras organizadas que emulan el procesamiento de la información que tiene lugar en el sistema nervioso. Esta estructura tiene unidades de procesamiento de datos, llamadas neuronas conectadas entre sí a través de *sinapsis* caracterizadas por pesos que el algoritmo aprende a partir de los datos. El procesamiento de imágenes es un área que se ha beneficiado enormemente de estas técnicas de procesamiento de datos. La empresa NVIDIA ha contribuido sustancialmente con la creación de la biblioteca CUDNN, que permite el uso de la GPU del ordenador para acelerar este proceso [17]. Este método de aproximación al sistema nervioso permite imitarlo mediante la creación de áreas específicas para llevar a cabo tareas específicas, como la extracción de características de una imagen [18].

Por otro lado, el *Transfer Learning* consiste en utilizar los conocimientos adquiridos en tareas anteriores para entrenar una nueva tarea [19]. Esta es una forma similar de transmitir el conocimiento a la que tienen los humanos, en comparación con los métodos de aprendizaje tradicionales que se encuentran en los enfoques clásicos de *Machine Learning*. Por ejemplo, si una red neuronal ha sido entrenada en la primera tarea para reconocer plantas, y posteriormente queremos entrenar una red diferente para reconocer árboles, parte del conocimiento de la primera red se puede reutilizar para la segunda, siendo necesaria menos información para entrenar la segunda.

Para llevar a cabo esta tarea existen varios algoritmos de *Deep Learning*. Estos son algunos de ellos:

- **Modelos R-CNN:** el modelo fue introducido en 2014 por Ross Girshick, y se basa en redes neuronales convolucionales. La red se divide en tres etapas: propone una región de interés, las características de la misma se extraen y finalmente se clasifican [20].
- **Modelos Fast R-CNN:** se basan en el modelo anterior y fueron mejorados por el grupo de investigación de Microsoft, a diferencia del anterior, simplifica la etapa de propuesta de regiones de interés y combina las dos últimas etapas, permitiendo acelerar el proceso de reconocimiento [21].
- **YOLO (You only look once):** esta es una red que tiene como entrada una imagen, y da como resultado una *bounding box* y una clasificación. Esta técnica es menos



precisa que los modelos Fast R-CNN, pero ofrece una mayor velocidad de procesamiento. El funcionamiento de este algoritmo se basa en la división de la imagen, en cuadrículas, y se proponen diferentes *bounding boxes*, estos se determinarán posteriormente si pertenecen a una clase en función de la confianza que tengan asignada [22].

- **Métodos *Single Shot detector* (SSD):** Están diseñados para la detección de objetos en tiempo real. Son capaces de igualar la precisión de las redes Fast R-CNN, con una imagen de baja calidad. Logran aumentar la velocidad de procesamiento eliminando la propuesta de regiones y aumentar la precisión mediante el uso de la extracción de características a múltiples escalas. Este método consta de dos partes un extractor de características basado en una red neuronal y una serie de filtros convolucionales para la detección de objetos [23, 24].

En el contexto de este proyecto, la detección de objetos es la parte más importante, ya que se encarga de dar el aviso al conductor de lo que está sucediendo. En el apartado de metodología se explicará el modelo seleccionado.

6. METODOLOGÍA

6.1 RECOPIACIÓN DE DATOS DEL VEHÍCULO

El único dato recogido del vehículo será la velocidad. Esto se utiliza para encender y apagar la pantalla para que no distraiga al conductor mientras está en movimiento. Se han seleccionado 2 km/h como punto en el que se activa la pantalla. Es decir, la pantalla estará activa siempre y cuando el vehículo se mueva a una velocidad inferior a 2 km/h o esté detenido. El pequeño margen a cero se da para que la pantalla no se active y desactive cada vez que el vehículo avanza unos centímetros si se detiene detrás de otro coche esperando a que sea su turno para girar o avanzar.

Como ya se ha mencionado en el estado del arte, el método utilizado para la recopilación de datos será el puerto OBD II. Antes de la decisión de utilizar el puerto OBD, se han planteado otras alternativas, como el uso de GPS o la lectura directa del tacómetro a través de un cable. La primera opción ha sido descartada debido al desfase entre la lectura del GPS y la velocidad del coche, especialmente a bajas velocidades. El segundo método ha sido descartado para no dañar el coche que se utilizará para las pruebas, a pesar de ser el método más fiable.

El puerto OBD funciona de la siguiente manera: en la parte inferior del volante se suele encontrar el punto de conexión. Este recopila datos de todos los sensores del automóvil y los monitorea constantemente. Una vez conectado el lector, que puede ser Bluetooth, cable o WIFI, los datos del vehículo se transmiten al dispositivo utilizado para la lectura de datos. Este proyecto utilizará un lector con comunicación por cable USB, como el de la imagen inferior.



Ilustración 3: Lector con cable OBD II (Imagen obtenida de [25])

Los datos transmitidos desde el sensor están en forma de un código de 5 dígitos. El primer dígito indica de qué parte del coche se trata la lectura, el segundo la organización que ha

definido el código, el tercero especifica una función específica del coche y los dos últimos son para indicar errores [7].

Con el fin de simplificar la lectura de estos códigos, se utilizará la biblioteca Python-OBd[26], que ha incorporado funciones que permiten leer los códigos y proporcionar directamente sus valores. Una vez conectado el lector, se establece la conexión y se declara una función para monitorizar el valor deseado, en este caso la velocidad.

6.2 CAPTURA DE IMÁGENES

Cualquier sistema de visión artificial tiene una primera fase de adquisición de imágenes. Para esta tarea se utilizarán dos cámaras Raspberry piCameras V2 de 8 Megapíxeles. Estas se colocan en la parte delantera del coche, en la imagen 5 se puede ver el punto donde se colocan.

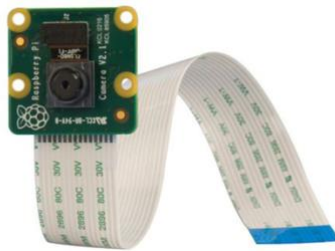


Ilustración 4: Raspberry piCamera V2 (Imagen obtenida de [27])



Ilustración 5: Posición de la cámara en el coche

Para obtener las imágenes de las cámaras, se utilizará la biblioteca OpenCV, de la que más adelante se darán más detalles, y el *framework* multimedia GStreamer que permite ajustar el formato de salida de la imagen, ajustar las dimensiones de la imagen capturada y la frecuencia de captura (framerate).

El *Framerate* es un parámetro muy importante en este proyecto ya que el sistema debe ser capaz de capturar coches en movimiento. En principio el límite de velocidad urbano es de 30 km/h, pero como margen de seguridad se pondrá que debe detectarse hasta 60 km/h (17,7 ms/s) en caso de que un vehículo vaya a una velocidad superior a la permitida.

Esta velocidad permite determinar el mínimo necesario para poder capturar coches a 60 km/h. De esta manera, si se hubieran tomado 18 imágenes por segundo, el coche se habría movido aproximadamente 1 metro. A medida que el *framerate* aumenta la distancia que el coche se desplaza cada *frame*, se ve reducida. En nuestro caso el *framerate* se establece en 60 FPS², con el fin de permitir un movimiento máximo del coche entre capturas de 30 cm.

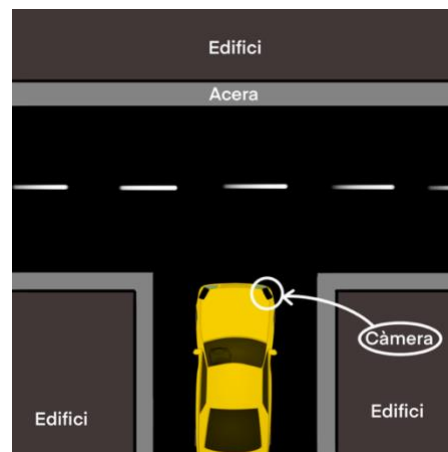
Por lo tanto, la cámara se ajusta con los siguientes parámetros:

Tamaño del fotograma de captura: 1280 x 720xels de píxel

Velocidad defotogramas: 60 FPS

Vale la pena anticipar que los 60 FPS no se terminarán dando ya que los algoritmos explicados a continuación harán que el proceso de captura sea más lento.

La ilustración 6 muestra una representación de la posición desde la que se capturan las imágenes en el proyecto. Como se puede ver, el conductor no invade la vía perpendicular.



Il·lustració 6 Posición del coche para la captura de imágenes

2 Fps: *Frames* por segundo

A continuació, se presenten una sèrie de imatges tomades de la posició indicada, i en el Annex 1, se poden encontrar m s exemples.



Il·lustraci n 7: Ejemplos de capturas de im genes en diferentes esquinas

6.2.1  TICA O PROTECCI N DE LOS DERECHOS DE IMAGEN

Dado que este proyecto captura im genes, es necesario aclarar que no se almacena ninguna de las im genes que el dispositivo utiliza para su procesamiento, de esta forma se mantiene la privacidad de las personas que puedan aparecer en el momento en que las c maras est n trabajando.

La  nica vez que se han almacenado las im genes ha sido cuando se prepar  un conjunto de datos de muestra para llevar a cabo las pruebas, y en estas, solo aparecen las personas que han dado su consentimiento para aparecer en los v deos.

6.3 SISTEMA EMBEBIDO

Como se mencionó anteriormente, el sistema integrado que se utilizará en el proyecto es una Jetson Nano B01 de la marca Nvidia. Esta placa es el lugar donde se dará todo el procesamiento de los datos. El sistema operativo que ha instalado es Jetpack 4.5, que se basa en Linux 18.0. El lenguaje de programación que se utilizará para realizar el procesamiento de datos y las alertas de controladores será Python 3.7.

El programa que controla el proyecto hace uso de las siguientes librerías:

- **Python-OBD:** Como se menciona en el apartado de la lectura de datos del vehículo esta biblioteca, permite la comunicación con el lector OBD, y permite obtener la velocidad del coche para poder utilizarla y regular si la pantalla está encendida o no.
- **OpenCV (con CUDA):** Es la librería que se encarga de todo el procesamiento de imagen y de la captura del vídeo. Es dentro de esta que se encuentran *Background Subtraction* que se explican en la siguiente sección. Además, esta biblioteca junto con la biblioteca *Tensorflow* es responsable de la detección de objetos. Debido a que el sistema integrado tiene una GPU que pertenece a Nvidia, puede beneficiarse de la biblioteca CUDA, que ofrece un procesamiento de imagen acelerado.
- **Tensorflow:** es una biblioteca de código abierto desarrollada por Google, que se utiliza para el entrenamiento y ejecución de algoritmos de *Deep Learning*.
- **Jetson inference:** la función de esta librería es la optimización de redes neuronales para que puedan ser utilizadas en el Jetson Nano.
- **PySimpleGUI:** Esta biblioteca se utiliza para crear la interfaz de usuario para que interactúe con la pantalla y pueda verse las cámaras.
- **Jetson.GPIO:** permite controlar los pines GPIO del Jetson, permitiendo activar los LED y avisos de sonido que se utilizarán para notificar al usuario de la presencia de un objeto reconocido.

El código escrito para el proyecto se incluye en los anexos, este se compone de dos *scripts*. El primero incluye una clase que se creó para administrar los algoritmos de procesamiento de imágenes. Incluye algoritmos de *Background Subtraction* y reconocimiento de imágenes. El segundo script, sirve para englobar todo el código, y permite la lectura del OBD. También administra la GUI y se comunica con el primer *script* para obtener los resultados del procesamiento de imágenes.

La siguiente figura muestra el esquema de conexiones, para entender cómo cada elemento se comunica con el Jetson Nano.

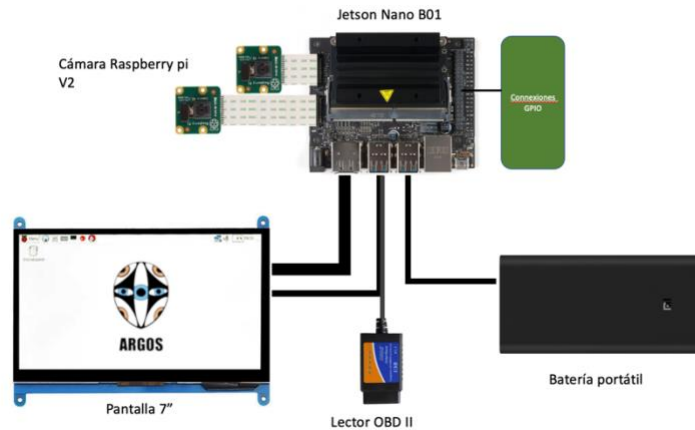


Ilustración 8: Representación de conexiones de componentes con el Jetson Nano. Los materiales utilizados para el proyecto son: dos cámaras Raspberry Pi v2, utilizadas para la captura de imágenes; una pantalla táctil para ver imágenes e interactuar con la interfaz de usuario; un lector OBD utilizado para la recogida de datos del vehículo; una batería portátil de la marca Xiaomi, con capacidad de alimentación de 5v 2.6A; para las conexiones GPIO, se utilizan cuatro LED y un buzzer (con los componentes adicionales necesarios)

Como se puede ver en la Figura 8, la placa se alimenta a través de una batería portátil. Esta no es la opción más recomendada y sería mejor alimentar la placa directamente con la batería del coche y un convertidor reductor de 12 V a 5 V, pero debido a un problema con la batería del coche utilizada para las pruebas, se ha elegido esta solución.

6.4 SUSTRACCIÓN DE FONDO

Las imágenes adquiridas por las cámaras frontales se procesan a través del algoritmo de *Background Subtraction* para aislar aquellos objetos que están en movimiento en la escena. A continuación se muestran los criterios utilizados para seleccionar el algoritmo con el que se trabajará y los detalles de cómo funciona.

6.4.1 SELECCIONAR ALGORITMO

Con el fin de seleccionar el algoritmo más adecuado para este proyecto, se han evaluado diferentes opciones de *Background Subtraction*. En particular, se han



considerado algoritmos: k-NN, MOG, MOG2 y GMG³. Para ello, estos algoritmos se han aplicado en una serie de vídeos, y se han guardado las máscaras resultantes. Estas se han comparado con el resultado esperado.

Estos resultados esperados se han obtenido mediante el etiquetado manual con la herramienta de diseño gráfico Photoshop. Con esto, se ha creado una máscara *ground truth* representativa, siguiendo la forma de etiquetado utilizada por el algoritmo, en otras palabras, se ha hecho un *bounding box* alrededor del objeto en movimiento como debería hacer el algoritmo. El color negro para el fondo, con una etiqueta 0, y el color blanco para el elemento móvil, con la etiqueta 1.

El algoritmo utilizado para esta evaluación se obtuvo a partir de las siguientes referencias [28, 29]. El rendimiento de cada uno de los algoritmos ha sido evaluado mediante una matriz de confusión, que representa las cuatro combinaciones posibles entre los valores reales y los obtenidos a partir de la máscara binaria generada por el algoritmo.

En nuestro caso, la clase de fondo pertenece al valor negativo, es decir, 0 y la clase en primer plano u objeto en movimiento, sería el valor positivo, es decir, 1.

La matriz de confusión se define a partir de los siguientes resultados de clasificación:

MATRIU DE CONFUSIÓ:

Verdadero positivo (PV): Representa el número de predicciones positivas, es decir, el número de píxeles que se han etiquetado con la clase 1 y que realmente pertenecen a esta clase.

Negativo verdadero (NV): representa el número de predicciones negativas, es decir, el número de píxeles que se han etiquetado como clase 0 y que realmente pertenecen a esa clase.

Falso positivo (FP): representa el número de muestras que se han clasificado como positivas (primer plano) pero que en realidad son negativas (fondo).

³ GMG: Algoritmo se basa en el artículo [42]. La base de su funcionamiento es la estimación de imagen y la aplicación por pixel de filtros bayesianos.



Falso negativo (FN): representa el número de muestras que se han clasificado como negativas (inferior) pero que son realmente positivas (primer plano).

MEDIDAS DE RENDIMIENTO DEL ALGORITMO:

De ello, se obtienen las siguientes medidas:

Precisión: Esta es la proporción de píxeles correctamente clasificada en comparación con todos los píxeles evaluados. Esta medida es muy sensible a la evaluación de conjuntos de datos en los que las clases no están equilibradas. La ecuación 1 muestra cómo se calcula.

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

Sensibilidad o relación de verdaderos positivos: Corresponde a la proporción de muestras clasificadas como verdaderos positivos (PV), con respecto a todos los positivos (PV + FP). La ecuación 2 se define como hacer esta medición. Esta medición se puede interpretar en qué tan bien el algoritmo detecta los objetos que están en movimiento.

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

Especificidad o relación de falsos positivos: Por otro lado, lo especificado es la proporción de muestras negativas clasificadas como positivas (FP) con respecto a todas aquellas que son realmente negativas (NV + FP) como se muestra en la ecuación 3. Este valor se interpreta como la capacidad de detectar correctamente cuál es el fondo de las imágenes obtenidas.

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

Valor F1: se trata de una medida que relaciona los valores de precisión y la sensibilidad. Esta medición es bastante útil cuando se tiene una gran desigualdad en los datos utilizados para la evaluación, y este es uno de los casos en los que es necesario utilizarlo, ya que la mayoría de los píxeles de las imágenes pertenecen al fondo. La siguiente ecuación muestra cómo se realizaría el cálculo.

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

En el apartado de resultados se presentarán los resultados obtenidos en este estudio. Estos resultados y sus implicaciones se analizan en la discusión.

6.4.2 KNN BACKGROUND SUBTRACTION

El algoritmo seleccionado se basa en el *Gaussian Mixture Model* (GMM), y el *K-nearest neighbour* (K-NN). Este se describe en profundidad en [30]. En términos generales, se presenta un método recursivo para actualizar los parámetros de GMM y actualizarlos para cada píxel. El algoritmo K-NN está incluido en la biblioteca OpenCv [31] y tiene una serie de parámetros que deben ajustarse para alcanzar el funcionamiento deseado del algoritmo. La inicialización del modelo se lleva a cabo con el número de *frames* asignados a la variable *NSamples*. Durante este período se define qué objetos pertenecen a la clase de fondo y qué objetos de la clase de objeto en movimiento comparando los píxeles de estos fotogramas. A continuación, el modelo se actualiza cada *N frames*, donde *N* es el número de *frames* que se han definido en el *parámetro History*.

K-NN es un algoritmo utilizado principalmente para problemas de clasificación predictiva. Este algoritmo no cuenta con una fase de entrenamiento específica, sino que durante el mismo entrenamiento se lleva a cabo una tarea de clasificación. Es un algoritmo no paramétrico, es decir, el entrenamiento se lleva a cabo sin información sobre los datos de entrada [32]. El principio de funcionamiento del algoritmo se basa en un sistema de votación llevado a cabo entre los *K* vecinos más cercanos y la similitud de características - con respecto a los datos de entrenamiento- para predecir la clasificación de los nuevos datos. La siguiente imagen presenta un ejemplo sencillo de cómo funciona el algoritmo. El punto negro es la nueva información que se está evaluando. Si $K=3$, tenemos que mirar los tres puntos que están más cerca. Dado que la mayoría de estos son rojos, el nuevo punto se asigna a la clase roja.

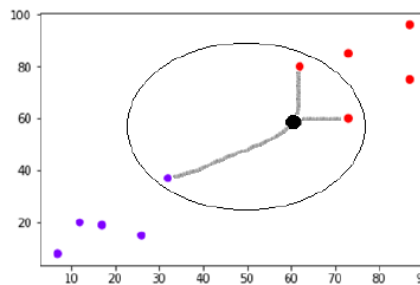


Ilustración 9: Ejemplo K-NN obtenido de [32].

Los parámetros necesarios para ajustar el funcionamiento del algoritmo son:

History: Este es el número de *frames* que afectarán a la modificación de los pesos asignados a la clase de fondo y al objeto que está en movimiento. Con este parámetro se puede ajustar la referencia sobre la que se comparará el nuevo *frame* para comprobar si esta ha cambiado o no. Este valor se ha establecido en un valor de 100. Si el valor es demasiado pequeño los cambios en la imagen tardan demasiado en aparecer reflejados, y si el valor es demasiado alto, la máscara cambiara con demasiada frecuencia provocando que el cambio mínimo en la iluminación modifique drásticamente la máscara. El *framerate* de la cámara afectará al valor de esta configuración, ya que cuanto mayor sea este parámetro, menos tiempo tardará en actualizarse.

Nsamples: Define el numero de muestras guardadas en memoria con las que cuenta el algoritmo *Background Subtraction* para comparar y asignar una nueva clase a los píxeles del nuevo *frame*. Si se cuenta con un número muy pequeño de muestras de, la segmentación del fondo tiene una gran cantidad de ruido como se puede ver en la imagen de abajo. Por otro lado, con el valor ajustado se consigue una máscara más uniforme. Para este proyecto se utiliza un valor de 10.



Ilustración 10: A) Imagen original B) NSamples con valor 10 C) NSamples con valor 3

KNNSamples: El parámetro K-NN Samples, es la variable K que siempre se define en los algoritmos de K-NN. Con esta se ajusta el número de muestras con las que se comparará cada píxel para decidir si pertenece a una clase u otra, se asignará la que tenga más votos. En el artículo original de este algoritmo, se recomienda utilizar un valor que esté entre 3 y 60. [30] A medida que el valor K aumenta la precisión con la que se detecta el fondo también aumenta porque el píxel actual se compara con un mayor nombre de píxeles cercanos para ser clasificados como fondo o como un objeto en movimiento, pero como efecto negativo de este aumento la velocidad de adquisición de la máscara se ve reducida. En este caso se utilizará un

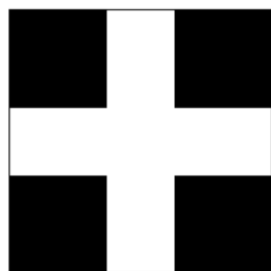
valor de 3, ya que un valor más alto hace que el algoritmo funcione demasiado lento.

Detectar sombras: Modificando el valor de esta variable se puede seleccionar si las sombras serán consideradas como fondo o como objetos en movimiento, dependiendo de si la variable está activa o no, se puede ajustar para descartar únicamente sombras muy marcadas o todas las que aparecen en el *frame*. Para este proyecto se detectarán todos los tipos de sombra porque cualquier pequeño cambio en la imagen se puede utilizar para obtener un reconocimiento más rápido del objeto.

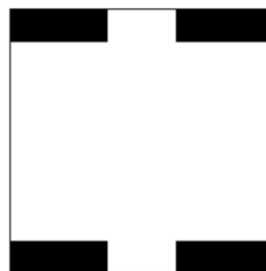
Una vez aplicado el algoritmo de *Background Subtraction*, se realizarán una serie de operaciones morfológicas sobre la imagen binaria obtenida con el fin de facilitar la detección de objetos.

6.4.3 TRANSFORMACIONES MORFOLÓGICAS

Las transformaciones morfológicas son operaciones basadas en la estructura geométrica de los objetos de la imagen. Estos tienen como entrada una imagen, generalmente binaria, y un elemento estructural o *kernel*. Usando la combinación de estos dos y la operación deseada, la máscara se modifica[33]. En este trabajo se utilizan dos *Kernels* diferentes, lo que permite su aplicación en diferentes casos. El primero tiene forma de cruz y un tamaño de 5x5 píxeles y el segundo tiene forma de elipse, con un tamaño de píxel de 7x7. La forma elíptica abarca más superficie, lo que permite que la máscara se expanda de manera más significativa.



Kernel Cruz



Kernel Elipse

Ilustración 11 Matrices que componen los *Kernels*

Las transformaciones morfológicas utilizadas se llevan a cabo en el siguiente orden:

OPENING

En primer lugar, se trata de llevar a cabo una **Opening**, esta operación es la combinación de dos operaciones, una erosión seguida de una dilatación de la máscara. Frecuentemente utilizado para eliminar el ruido en las máscaras, por lo que se ha utilizado en primer lugar. De esta manera, se eliminan pequeños elementos que pueden haber sido detectados erróneamente. Esta operación será la única que utilizará el *kernel* en forma de cruz para evitar que el resto de la máscara se vea demasiado modificado.



Ilustración 12: Ejemplo del efecto de *Opening* (Fuente: Referencia [33])

CLOSING

Esta es la operación contraria al *Opening*. En este caso su función es cerrar las pequeñas aberturas que pueden quedar en la máscara.



Ilustración 13 Ejemplo de función de *Closing* (Fuente: Referencia [33])

DILATE

Finalmente, utilizando la función **Dilate**, la máscara se expandirá para reducir la posibilidad de que alguna parte del objeto en movimiento quede sin enmascarar.

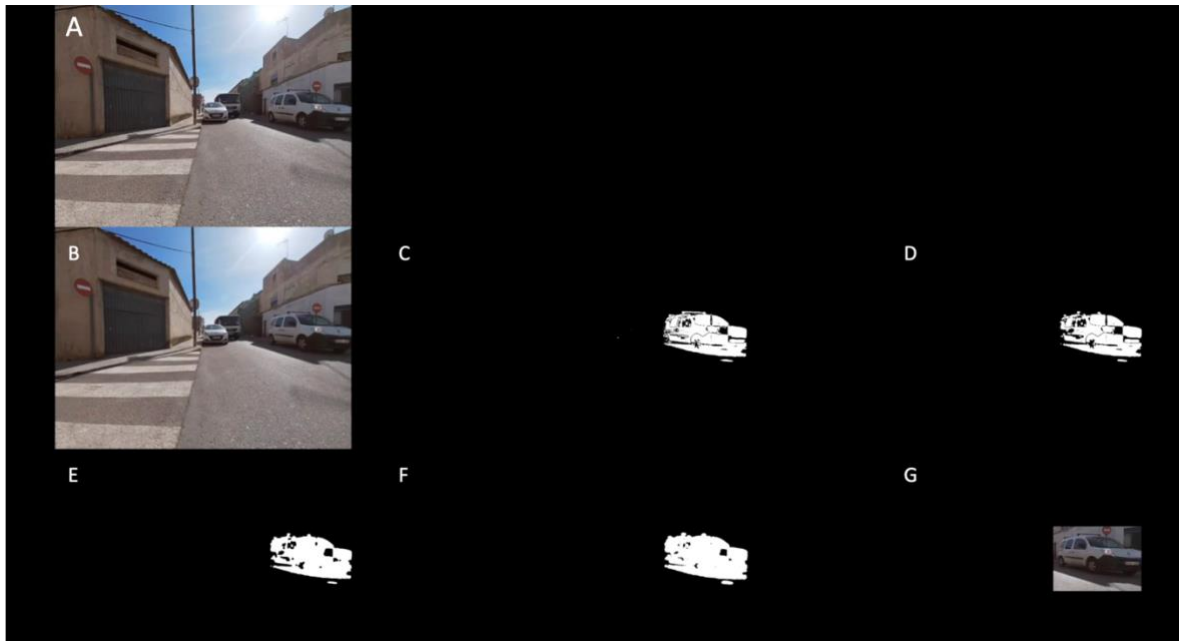


Ilustración 14: Ejemplo de la función *Dilate* (Fuente: Referencia [33])

6.4.4 SELECCIÓN DE CONTORNOS Y *BOUNDING BOX*

Para finalizar el *Background Subtraction*, se buscarán los contornos de la máscara obtenidos tras la transformación morfológica, ya que puede ser que en el procesado anterior pudieran quedar aberturas en la máscara. Los contornos obtenidos se utilizarán para generar *Bounding Box* - que encapsularán aquellos contornos que hayan pasado un filtro de tamaño- logrando el objetivo de ampliar las mascararas de los objetos en movimiento.

En la siguiente ilustración se presentan los diferentes pasos que sigue el algoritmo para aislar los elementos móviles de las imágenes. La imagen A, es la original. Entonces esta se suaviza (para quitar ruido), y se aplica el algoritmo de ***Background Subtraction***. En la imagen D se puede ver cómo los pequeños puntos del coche en la imagen C desaparecen cuando se ha aplicado **el *openig***. En las imágenes E y F se puede ver el efecto de aplicar el **closing** y **el *dilate***. Por último, el algoritmo de selección de contorno y *bounding box* crea un rectángulo alrededor del objeto móvil.



Il·lustració 15 del Background subtraction a) Imagen original b) Blur c) Algoritmo KNN d) Openning E) Closing F) Dilate G) Find Contours & Bounding Box

6.5 DETECCIÓN DE OBJETOS

A continuació, *del Background Subtraction*, se realitza la detecció de objectes. Gràcies al pas anterior no es reconecran els objectes estàtics, com cotxes aparcats. Això evita que el conductor sigui notificat erròneament, el qual ha de estar alerta a aquells objectes que puguin representar un perill (per exemple, un cotxe que s'acera o un peatón creuant la carretera).

Per realitzar aquesta tasca, s'utilitzarà la xarxa Mobilenet-SSD v2 pre-entrenada amb el data set COCO (Common Object in Context) [34], podent reconecre fins a 90 objectes diferents i el fons [35]. Aquesta xarxa es compon de dos components principals: MobileNet [36], una xarxa neuronal profunda amb una arquitectura eficient dissenyada per ser utilitzada en dispositius amb menor capacitat informàtica com telèfons i sistemes embebidos, i una xarxa SSD (SingleShotMultibox Detector) [23], que permet la detecció simultània de múltiples objectes en una imatge.

MOBILENET

Es una xarxa neuronal convolucional que funciona com un classificador. La característica que el diferencia del rest - i li permet treballar de manera més eficient en sistemes de menor rendiment- és el ús d'un nou tipus de capes

convolucionales llamadas *Depthwise Separable Covolutions*. Estos se basan en una capa convolucional normal, de la dimensión deseada, pero en este caso se suman dos capas convolucionales 1x1, una delante y la otra al final. La primera tiene la función de expandir los tensores y la última que actúa como cuello de botella reduciendo el número de tensores. En otras palabras, es como si los datos recogidos en la imagen se descomprimieran y una vez procesados el resultado se comprimiera [32,33].

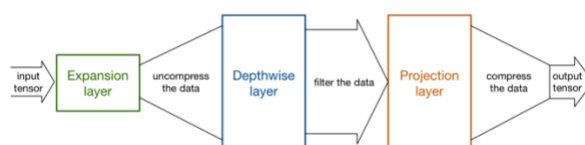


Ilustración 16: Representación figurativa de la capa depthwise separable (Fuente: Referencia [37])

SSD

Realiza la clasificación y la localización en una única pasada por la red y, al mismo tiempo, predice la clase y el *bounding box* a medida que procesa la imagen. La imagen 17 muestra la arquitectura que tiene. En esta se puede apreciar que Mobilenet queda integrada en la primera parte y funciona como un extractor de características para SSD. Esto es posible gracias a la flexibilidad del SSD que permite cambiar la arquitectura básica que utiliza [37].

Esta red genera un número fijo de *bounding boxes* y una puntuación para la presencia de instancias de objetos de una clase dentro de estos *bounding boxes*. Por último, finaliza con un paso de *Non-maximum supression* para agrupar los *bounding boxes* con un alto grado de superposición. [23]

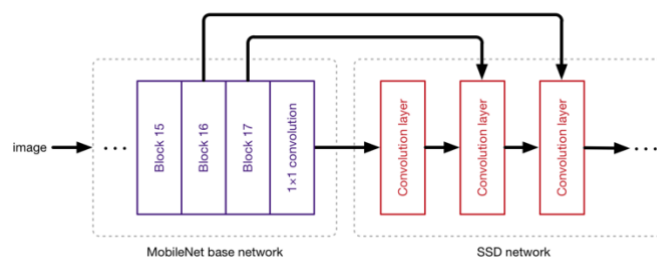


Ilustración 17: Estructura de red MobileNetSSD-v2 (Fuente: Referencia [37])

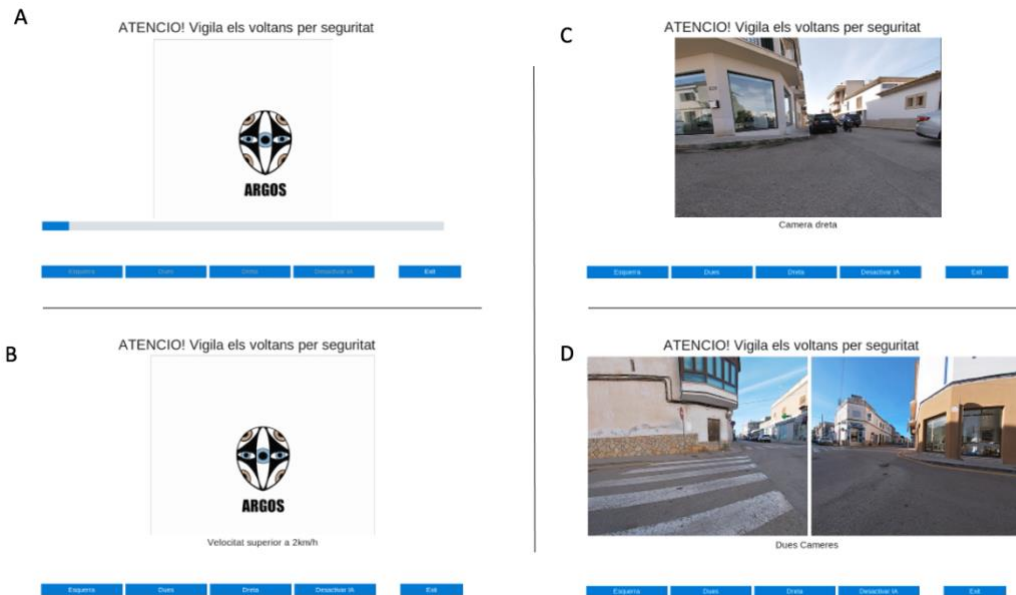
Una vez definido el algoritmo de reconocimiento de objetos utilizado, se explicará cómo se procesan ahora los datos obtenidos en el mismo. Los objetos que se reconocen, es decir, las clases permitidas, són: vehículos (bicicletas, autobuses, coches, motocicletas) y personas. En el caso de que un objeto no esté en las clases mencionadas anteriormente, no se detectaría. Más adelante, en la sección de discusión, se discutirá qué pasaría si apareciera un objeto que no está en la lista de reconocidos.

Cuando se reconoce cualquier objeto de los objetos establecidos, por primera vez desde que el coche se detiene en una intersección, sonará un ruido para avisar al conductor. Este no volverá a sonar hasta que se deje de detectar alguna clase mientras el vehículo siga parado o una vez que el automóvil se mueva de nuevo y se detenga en una nueva intersección.

Además del aviso sonoro, el dispositivo cuenta con avisos luminosos para identificar si lo que se ha detectado es un peatón o un vehículo y de qué lado se encuentra. Este permanecerá activo mientras se detecte una de las clases validas. En la siguiente sección podrás ver con más detalle cómo verás la salida que recibirá el conductor.

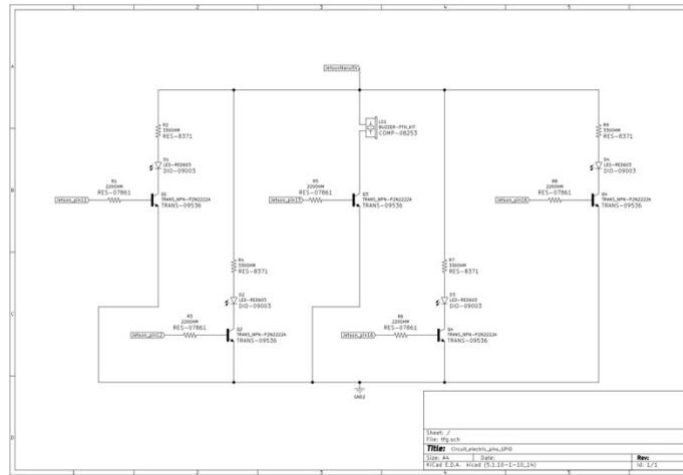
6.6 INTERFAZ DE USUARIO Y ALERTAS

Como se ha mencionado anteriormente, para la creación de la interfaz de usuario se ha usado PySimpleGUI. El usuario podrá interactuar con la GUI mientras el coche se mueve a una velocidad inferior a 2 km/h o este parado, y podrá decidir cuál de las cámaras se ve. Además de esto, se ha incorporado un botón que permite desactivar el reconocimiento de objetos, en caso de que el conductor solo desee tener activas las cámaras y no los avisos. A continuación se muestran las diferentes pantallas que verá el usuario:



Il·lustració 18 Pantalles GUI. A) Pantalla de carga B) Pantalla mientras el vehículo está en movimiento C) Pantalla con una sola cámara D) Pantalla con ambas cámaras

Los avisos luminosos y sonoros están integrados en el circuito electrónico que se muestra en la Figura 19. Estos se activarán cuando se reconozca cualquiera de los objetos relevantes. Hay cuatro señales luminosas, 2 a cada lado. De esta manera, el conductor puede saber rápidamente en qué lado se ha detectado el objeto. Las señales luminosas se dividen en dos categorías: personas y vehículos (coches, motocicletas, bicicletas y camiones).



Il·lustració 19: Circuit electrònic generat amb KiCad

Para encapsular el proyecto, se ha diseñado una carcasa para la pantalla, la batería, el sistema empotrado y la placa electrónica, y otra diferente para cada cámara. Esta tarea se llevó a cabo utilizando el programa CAD Fusion360. También se han generado los planos de diseño de las diferentes piezas.

En las imágenes 20 y 21, se muestra en las diferentes partes del diseño final. En el Anexo 4 se pueden encontrar los planos de diseño de cada pieza.

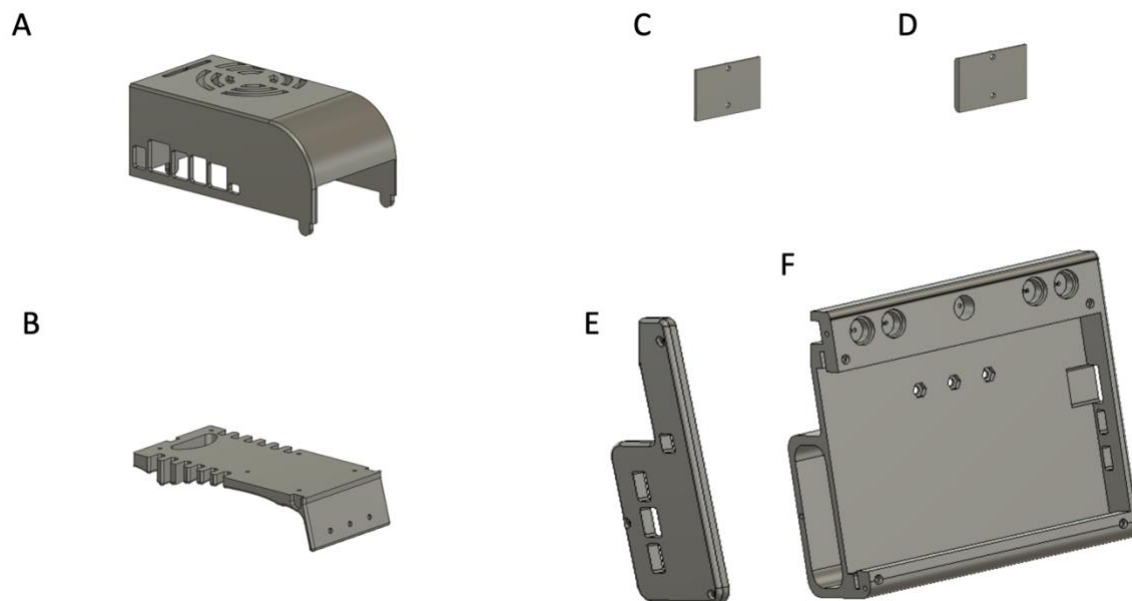


Ilustración 20: Piezas de la carcasa de la pantalla, la batería y las placas. A) Cubierta de la carcasa de la placa B) Base de la carcasa para la placa C) y D) Tapa parte trasera carcasa pantalla E) Carcasa de la pantalla y la batería

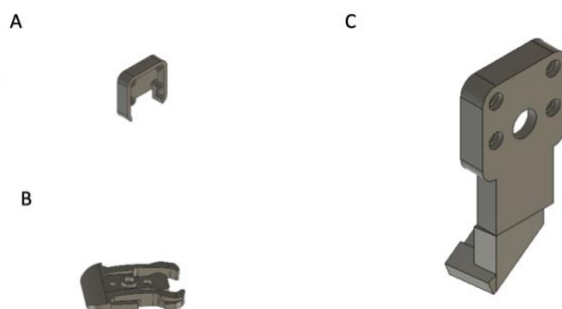


Ilustración 21: Carcasa de cámara derecha.

A) Parte trasera carcasa B) Clip para engancharse al coche C) Parte delantera carcasa

7. RESULTADOS

Con el fin de evaluar los resultados que implican el procesamiento de imágenes, se han grabado 12 videos en diferentes esquinas, con las cámaras colocadas en su posición, con el fin de crear un conjunto de datos para la validación de los algoritmos propuestos. De estos, 5 videos han sido seleccionados al azar de los cuales se han recopilado 4000 imágenes (*frames*). De las 4000 imágenes, 300 de ellas fueron seleccionadas al azar.

7.1 RECOPIACIÓN DE DATOS DEL VEHÍCULO

La recopilación de los datos se lleva a cabo con bastante precisión. Esto se debe a que la señal enviada al tacómetro es la misma que la leída por el lector OBD. En la imagen 22, se puede ver cómo se ve la pantalla cuando el coche va a una velocidad de más de 2 km/h.

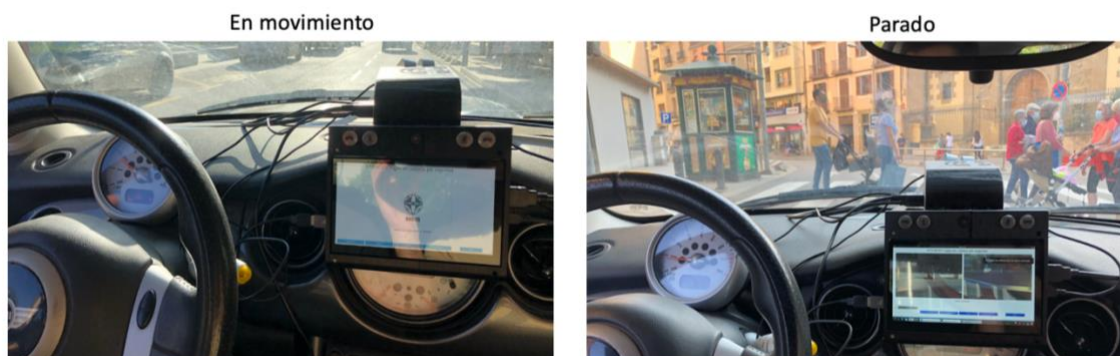


Ilustración 22: Ejemplificación de la GUI con el coche parado y con el coche en movimiento

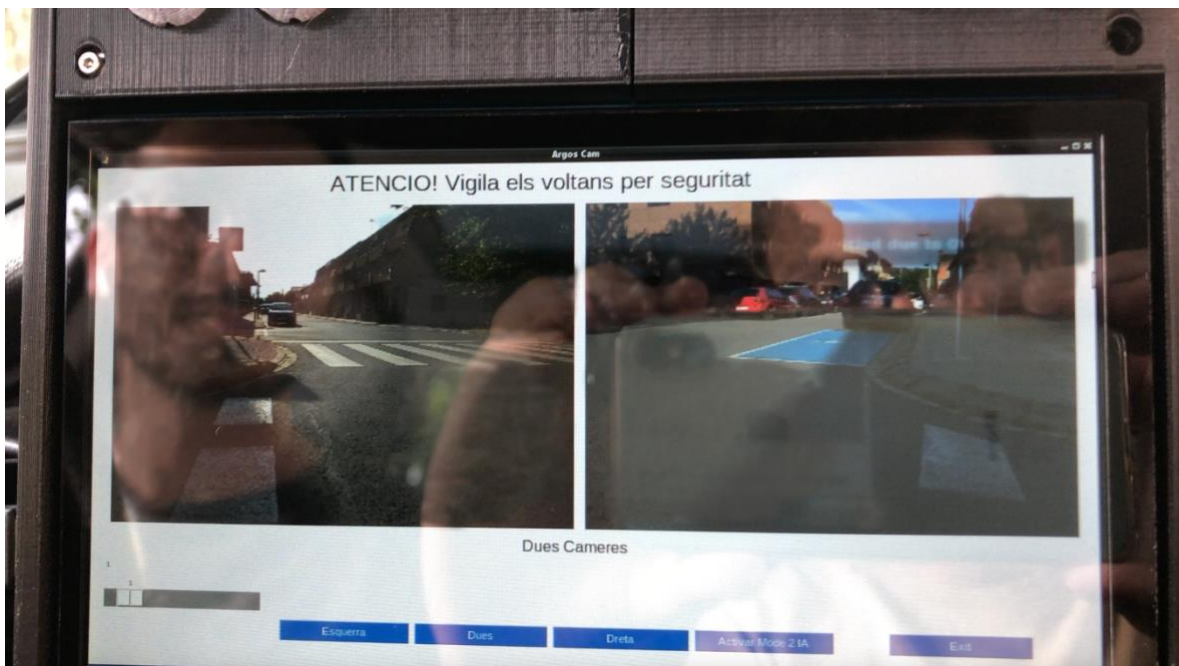
La conexión entre la placa no se ha podido establecer debido a un error de la placa adquirida para el proyecto. Pero para probar el funcionamiento del lector, se ha utilizado un ordenador con este se ha comprobado que el funcionamiento de este era adecuado y se cumplía la función de recoger la velocidad del coche y activar y desactivar la pantalla.

Cabe señalar que hay un pequeño retraso, de 4 segundos, debido a la comunicación en serie entre el ordenador y el lector. Esto no es de suma importancia y realmente no afecta significativamente el resultado final del proyecto. La consecuencia principal es la GUI tarda este tiempo más en cambiar.

7.2 VISIBILIDAD MEJORADA

Una vez revisadas las diferentes imágenes de la fecha fijada y los vídeos se puede afirmar que el posicionamiento de las cámaras permite conseguir el objetivo propuesto de reducir la distancia a la que el coche invade la calle, mejorando así la visibilidad del conductor y reduciendo el riesgo de accidente.

En la siguiente imagen, puede ver un ejemplo desde el punto de vista del controlador de cómo se vería la imagen en el dispositivo.



Il·lustració 23: Visión de las cámaras desde el punto de vista del conductor

7.3 SUSTRACCIÓN DE FONDO

En esta sección se presentarán los resultados de la selección del algoritmo de fondo y algunos ejemplos de las imágenes obtenidas después de que se haya aplicado la *substracción de fondo*.

Para seleccionar y evaluar el algoritmo **Background Subtraction**, se ha utilizado el conjunto de imágenes mencionadas al principio de la sección. Con el fin de comparar los resultados obtenidos con los esperados, se ha obtenido la *ground truth* utilizando Photoshop, tal y

como se indica en el apartado de metodología. La siguiente ilustración presenta un ejemplo de las máscaras obtenidas en comparación con la *ground truth*.

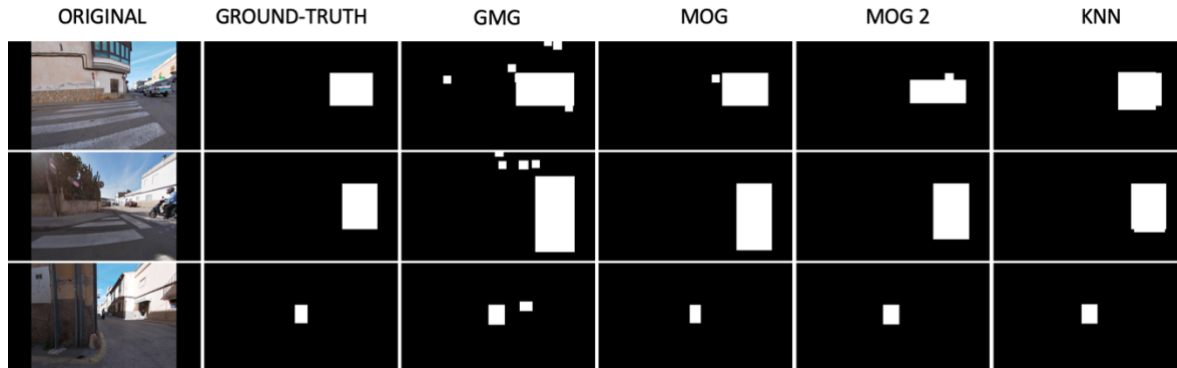


Ilustración 24 algoritmos de *Background Subtraction*

7.3.1 SELECCIONAR ALGORITMO

En la Tabla 1 se presentan los resultados obtenidos de la evaluación de los diferentes algoritmos de *Background Subtraction*. Para la evaluación de cada uno de los algoritmos, las máscaras -segmentadas manualmente y descritas anteriormente - se han utilizado como *ground truth*. La tabla presenta las mediciones utilizadas para la evaluación que se han presentado a la metodología: **Precisión** (proporción de píxeles clasificada correctamente), **Sensibilidad** (Verdaderos positivos con respecto a todos los positivos), **Especificidad** (Número de muestras clasificadas como negativas con respecto a todas las muestras), **F-1** (relación entre sensibilidad y precisión) y tiempo de procesamiento (tiempo que se tarda en procesar cada *fotograma*)

Tabla 1 los algoritmos de *Background subtraction*

Algoritmo BS	Precisión	Sensibilidad	Especificidad	Valor F-1	Tiempo(s) de procesamiento
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

En la tabla 1 se muestra que todos los algoritmos tienen un valor de precisión alta, pero como se mencionó anteriormente, este valor puede verse afectado por datos no marcados. Teniendo esto en cuenta, es necesario evaluar el resto de medidas.

Al ser un proyecto por el que lo más importante es que los positivos se detecten correctamente, es decir, objetos que estén en movimiento, la sensibilidad es un parámetro que hay que tener en cuenta. Esta necesidad se debe a que un error menor podría acabar provocando un accidente, por ejemplo, en el caso de que no se detectara un coche que se aproximase por el lateral y -como consecuencia de no recibir ninguna notificación- se podría producir una colisión. En este sentido, el costo de hacer un error de tipo FN vs FP es mucho mayor.

De acuerdo con esto, los algoritmos MOG y GMG pueden ser descartados, ya que no detectan estos objetos correctamente. El algoritmo GMG tiene una especificidad muy alta pero en la aplicación de uso de este proyecto no es tan importante. De hecho, si nos fijamos en el valor F es el más bajo de todos los algoritmos.

Finalmente, entre el MOG2 y el KNN, se ha determinado que KNN es el algoritmo que mejor funciona de todos los que han sido probados, teniendo en cuenta que el valor de F-1 es mayor que el resto además de tener la mayor especificidad. Los valores obtenidos en esta evaluación coinciden con los obtenidos por otros estudios realizados [38],[13].

7.3.2 K-NN ANTECEDENTES SUBTRACTION Y POSTPROCESSING

La evaluación de este algoritmo ha tenido lugar en el proceso de selección descrito en el apartado anterior. Esto se debe a que, en todos los casos anteriores, las imágenes utilizadas para la evaluación habían pasado por todos los pasos del procesamiento desde que se obtenía la imagen en el *bounding box*.

En cualquier caso, se presenta a continuación una breve evaluación cualitativa del proceso. Las siguientes imágenes presentan diferentes situaciones a las que se ha aplicado el algoritmo de *Background Subtraction*. En algunas de las imágenes presentadas se puede ver cómo se aíslan los elementos estáticos de las imágenes, como los coches aparcados y otros elementos que no necesitan ser reconocidos. En otros casos se ve cómo se detectan elementos que están en movimiento, pero no deberían ser detectados, como pueden ser reflejos y sombras.

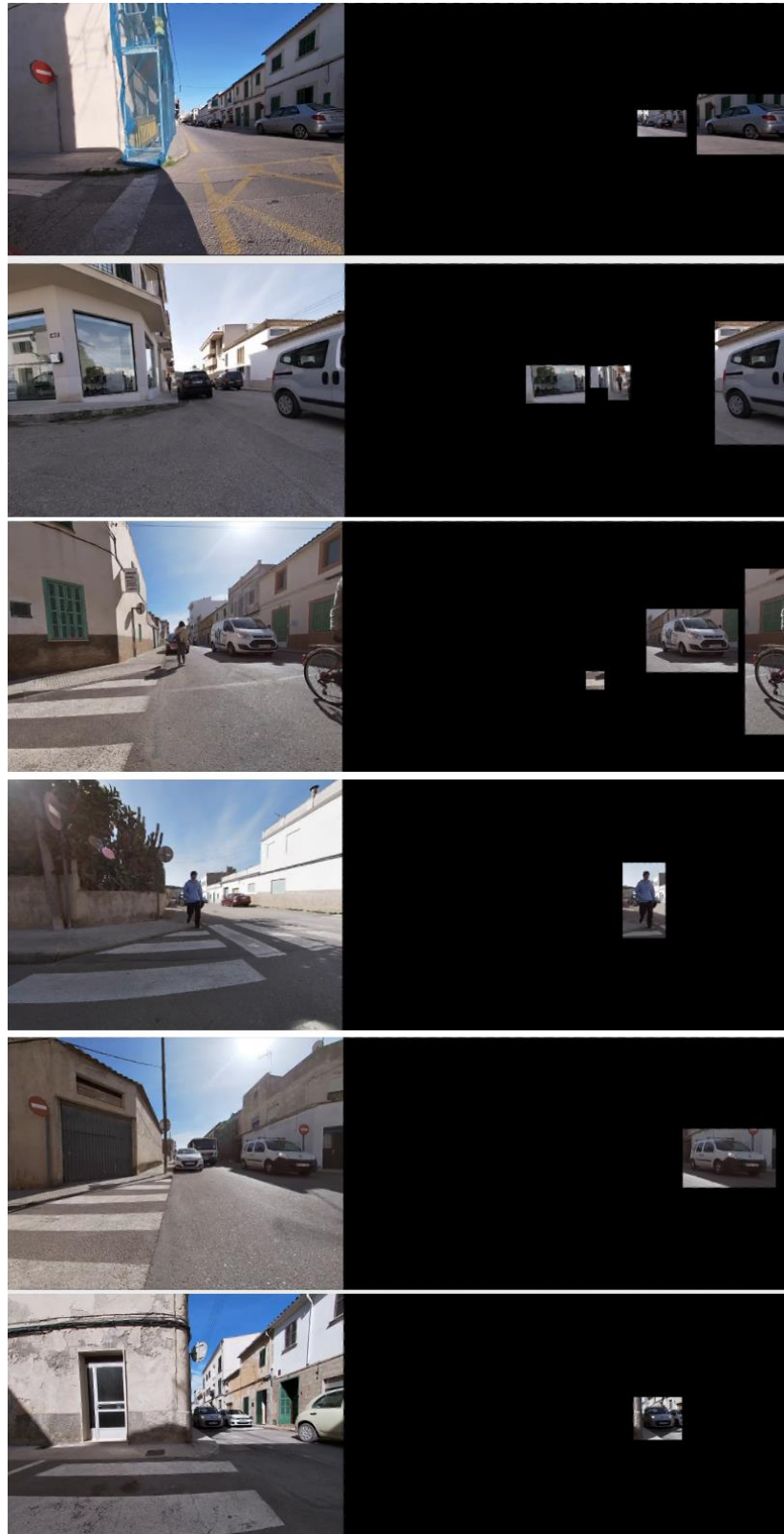


Ilustración 25: Ejemplos de *Background Subtraction* aplicada

7.4 DETECCIÓN DE OBJETOS

7.4.1 SELECCIONAR ALGORITMO

En esta sección se explicará por qué se ha seleccionado este método de detección de objetos, además de las ventajas que conlleva sobre otros. Antes de la decisión de utilizar la red Mobilenet -SSD v2, se han realizado pruebas con el detector de objetos YOLO [22]: estas dos redes son las más utilizadas por los sistemas embebidos. Como se mencionó anteriormente en la recopilación de datos de vehículos, para detectar vehículos que se mueven a una velocidad de 60 km/h, la adquisición de imágenes debe tener lugar a una velocidad de más de 16 FPS. Teniendo esto en cuenta, el algoritmo YOLOv3 debe ser descartado, ya que solo permitiría reconocer objetos que se mueven a 18 km/h. En la sala 2, se comparan las velocidades de ejecución (en FPS) de los algoritmos y mAP de cada algoritmo. ⁴

[39]

Tabla 2 Comparación de algoritmos de detección de objetos [35],[11],[22], [40]

Algoritmo	Velocidad de ejecución (FPS)	mAP	Dataset
YOLOv3	5	0.7423	VOC2007 [41]
		0,606	MS-COCO [34]
Tiiny YOLO	25	0,331	MS-COCO
MobileNet-SSD v2	39	0,727	VOC0712
		0,68	MS-COCO

Otro factor muy importante es la correcta detección de objetos, en este caso vehículos y personas. La detección errónea de estos podría causar un accidente. Teniendo en cuenta los datos de la tabla, y comparando las dos redes restantes, se determina que el mejor algoritmo para este caso de uso es la red MobileNet-SSD v2, ya que presenta una mejor precisión que tinyYOLO.

⁴ **mAP:** Se trata de la *significar promedio precisión (mAP)*, una medida ampliamente utilizada para evaluación algoritmos de reconocimiento de objetos. El *promedio precisión (AP)* se calcula mediante la búsqueda la zona bajo la curva de **exactitud** y **recordar**. El mAP es el promedio AP de todos Clases.

7.4.2 DETECCIÓN DE OBJETOS CON MOBILENET-SSD V2

Los resultados del procesamiento obtenidos para la detección de objetos se han visto afectados por diferentes elementos.

En primer lugar, la velocidad de ejecución (en FPS) obtenida con la Jetson Nano se reduce en comparación con las obtenidas en el ordenador, debido a la capacidad de procesamiento más limitada. El uso de una GUI aumenta esta disminución en la velocidad de ejecución de 15 FPS a 10 FPS. Esto sucede como resultado de que la GUI se queda esperando la entrada de un evento, lo que provoca un pequeño retraso y por lo tanto una reducción en la velocidad. En la Tabla 3, se pueden observar las diferentes velocidades de ejecución en las diferentes situaciones. Otro factor que provoca la reducción de velocidad es el uso de una batería portátil como fuente de alimentación. De esta forma, hace que el algoritmo deje de ser funcional, debido a la alta pérdida de fotogramas. Tiene tal efecto que incluso la detección de personas es imposible. Pero como se menciona en la sección de metodología, solo se ha utilizado este método de alimentación, debido a la limitación del vehículo utilizado para las pruebas. En el caso de que se incorporara a un vehículo se haría con la potencia de 10W.

Tabla 3 Velocidad de procesamiento del reconocimiento de objetos en diferentes situaciones

Dispositivos					
	Macbook pro, 16 GB de Ram y procesador i5	Jetson Nano (fuente de alimentación de 10W) con GUI, conectar a la corriente	Jetson Nano (fuente de alimentación de 10W) sin GUI, conectar a la corriente	Jetson Nano (fuente de alimentación de 5W) conectar a la corriente	Jetson Nano (fuente de alimentación de 5W) atrapado en el banco de energía
FPS	20	10	15	6	2

La evaluación de la detección de objetos se llevó a cabo con el conjunto de 300 imágenes de la data set. Estas han sido inspeccionadas visualmente y se ha establecido si había o no alguno de los objetos a detectar. Cada clase se ha etiquetado como un problema binario.

Para el criterio de evaluación, se han considerado tres casos, lo que permitirá el cálculo de la **precisión** y el **recall**. El primer caso sería que el resultado del algoritmo y el valorado visualmente coincidiesen, esto sería PV. En el caso de que el resultado del algoritmo indique que el objeto detectado pertenece a la clase equivocada será FP y finalmente si el objeto está presente en la imagen pero no ha sido detectado se etiquetará como FN. Cabe mencionar que hay dos clases que no se pudieron evaluar debido a la falta de datos. La Tabla 4 muestra el número de muestras para cada clase dentro de la data set de 300 imágenes, y el número de PV, FP y FN para cada una de ellas. La tabla muestra que la mayoría de las imágenes pertenecen a las clases de persona y coche.

Tabla 4 Número de muestra por clase y valores TP, FP y FN

	Número de muestras con objeto presente	PV	FP	FN
coche	128	78	6	50
motocicleta	24	8	0	16
persona	122	80	10	42
bicicleta	0	0	0	0
autobús	0	0	6	0

Con los valores de PV, FP y FN, se ha creado la siguiente tabla donde se pueden ver los valores de **precisión** y **recall** obtenidos por cada clase.

Tabla 5 Resultados de la evaluación del reconocimiento de objetos

	precisión	recall
coche	0.92	0.68
motocicleta	1	0.33
persona	0.93	0.65
bicicleta	No hay suficientes imágenes	
autobús	No hay suficientes imágenes	

En la tabla anterior se puede ver que el reconocimiento se lleva a cabo con mucha precisión. Esto indica que, de todas las detecciones que se han realizado, una gran parte ha sido correcta. Por otro lado, el *recall* indica que de todas las detecciones que el modelo debería haber tenido, ha habido un gran número que no se han detectado. Tras analizar los resultados y revisar las imágenes, se ha podido detectar que el principal problema en los casos en los que no se ha detectado el objeto, se debe a la distancia que se encuentra con respecto a la cámara. El algoritmo comienza a tener problemas con la detección cuando el objeto está a unos 7 metros de la cámara, esto puede ser un problema y requeriría una evaluación más profunda para ver si significa que el sistema no es efectivo.

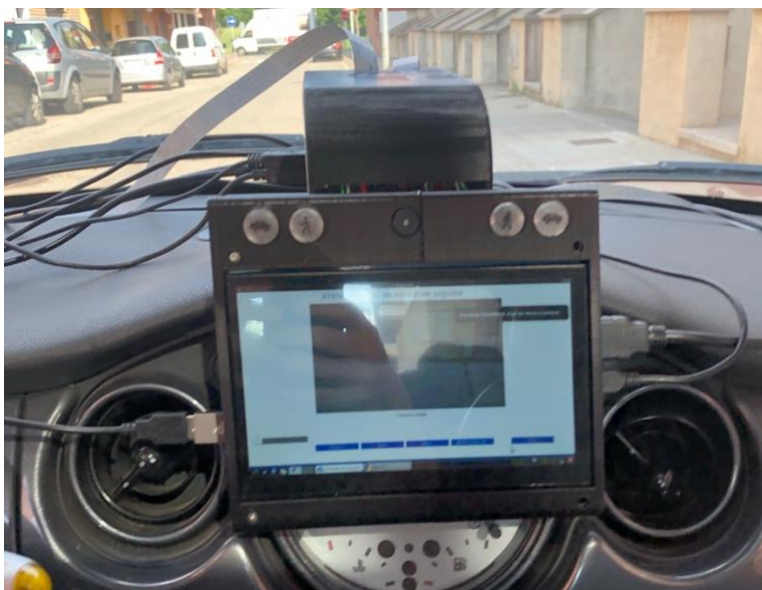
7.5 INTERFAZ DE USUARIO Y ALERTAS

Una vez analizados los datos, a continuación, se muestran una serie de imágenes donde se puede ver cómo se activan las advertencias luminosas cuando se detecta un objeto reconocido. Hay que recordar que sólo se activa en el lado en que se ha detectado el objeto. El primer ejemplo muestra cómo, cuando se detecta a una persona, el LED se activa con el símbolo de persona. En la siguiente imagen se pueden ver los dos LED activos simultáneamente. Y por último en la última imagen se puede ver cómo un LED está activo a cada lado de la pantalla.



Il·lustració 26: Ejemplos de advertencias luminosas en diferentes situaciones

Por último, la imagen 27 muestra el dispositivo montado en el coche. Este se ha unido al salpicadero del coche con Velcro.



Il·lustració 27 Sistema de levas ARGOS montado en un coche

8. DISCUSIÓN

La diferencia de distancia que el coche sale de la calle supone una mayor seguridad vial porque no está invadiendo la carretera a la que el conductor quiere incorporarse, lo que reduce el riesgo de colisión. La calidad de las imágenes mostradas por la pantalla, aunque no es de alta calidad, es suficiente para que se puedan reconocer los objetos que se pretendía: vehículos y personas. Sin embargo, en condiciones adversas, como fuertes lluvias, niebla o una zona mal iluminada durante la noche o si la cámara se encuentra a contraluz, pueden significar que las imágenes capturadas queden inutilizadas y el objeto no se detecta correctamente. Estas adversidades podrían superarse, aunque ya sería necesario un estudio más completo para encontrar un método adecuado.

En relación con el módulo *Background Subtraction*, el algoritmo resultante ha presentado resultados que cumplen su función de aislar objetos estáticos como coches aparcados. El hecho de enmascarar sujetos estáticos también puede conducir a problemas en algunos casos donde una persona permanece estática. En la imagen 24, se puede ver un ejemplo. Si esto ocurre, también se enmascarará y no será detectado por el algoritmo de detección de objetos. En otros casos, también se detectan reflejos, ya que estos se traducen en movimiento en la imagen. Estos no producen un efecto negativo, solo aumentan las áreas detectadas.

Para resolver el problema, de las personas que permanecen estáticas, habría que intentar modificar el algoritmo, por ejemplo, combinándolo con otro detector de objetos para reconocer solo a las personas o hacer uso de un algoritmo de *Semantic Segmentation* como los que se presentan en el estado del arte.

En cuanto al reconocimiento de imágenes, se ha demostrado que es posible lograr la detección con FPS adecuados y con alta precisión permitiendo detectar objetos correctamente, siempre y cuando funcione con una potencia de 10W y esté alimentado a 5V 4A. El principal problema es que esta detección no se puede llevar a cabo con la placa seleccionada si se utiliza la interfaz gráfica, ya que reduce los fps a la mitad, pasando a un valor inferior al deseado. Además, los bajos resultados de *recall* obtenidos y los casos de FN observados durante el etiquetado son una indicación de cuánto debe mejorarse la red neuronal para que pueda reconocer objetos a una mayor distancia.

Otro punto por destacar es que, si un objeto que no está en las clases de objetos mencionadas no sería reconocido, y esto podría causar una situación peligrosa. Un ejemplo podría ser: si se colocaba una pelota en medio de la calle no se detectaría, pero en el primer momento que apareciese una persona en detrás de esta el algoritmo la detectaría. Otro ejemplo podría ser un animal que se pusiese en medio de la calle, a menos que sea seguido por un objeto reconocido no se detectaría y el conductor no sería notificado, lo



que podría terminar causando un accidente. Dados estos ejemplos, puede ser apropiado ampliar el número de clases detectadas mediante la introducción de objetos comunes que pueden estar en medio de la carretera.

9. CONCLUSIÓN

En conclusión, con la realización de este proyecto, se ha demostrado que es posible fabricar un dispositivo, que hace uso del procesamiento de imágenes, que permite mejorar la visibilidad en las esquinas y de esta manera hacer posible la reducción de este tipo de accidentes.

Este dispositivo podría incorporarse a los sistemas que ya están integrados en la mayoría de los coches en la actualidad, como pantallas, cámaras y otros sensores que facilitarían su construcción. En el caso ideal esto podría convertirse en un nuevo ADAS que incorpore los coches.

La parte más problemática de todo el proyecto ha sido la correcta configuración del Jetson Nano, por el hecho de estar poco familiarizado con el sistema operativo Linux y la instalación de paquetes en él, además de las incompatibilidades que se han encontrado con las bibliotecas con las que había trabajado anteriormente. Otro punto que ha causado problemas ha sido la drástica reducción de FPS observada al pasar el código a la placa, debido a la limitada capacidad de multiprocesamiento que ha presentado y la limitación de la potencia de procesamiento, debido a la fuente de alimentación del power bank.

Los tres objetivos propuestos al comienzo del proyecto se han cumplido en su mayor parte. Lo único que no se ha podido cumplir plenamente, ha sido el de la integración completa del sistema en el coche, pero sabiendo el origen del problema, este es sencillo de resolver, alimentando la placa de una manera más eficiente.

En definitiva, para que el sistema se incorpore a un coche como parte de un nuevo sistema ADAS, tendría que pasar por una serie de mejoras. A continuación, se presentan posibles futuras mejoras que podrían aplicarse al proyecto para que este pueda ser un nuevo sistema de asistencia al conductor.

9.1 FUTUROS MILLORES

9.1.1 RECOPIACIÓN DE DATOS DEL VEHÍCULO

En el caso de la integración en un vehículo, la lectura de la velocidad se haría internamente, haciéndola más precisa.

9.1.2 CAPTURA DE IMÁGENES

Por ejemplo, para los dos primeros se podría intentar utilizar la tecnología de *sensor fusion*, utilizada por algunos coches autónomos, que se ha discutido en la sección de estado del arte. La solución sería combinar la lectura de las cámaras y los sensores de distancia.

El segundo problema podría ser tratar de resolver cambiando el tipo de cámaras utilizadas por las cámaras de visión nocturna. Esto ya significaría la adición de sensores infrarrojos para permitir la visibilidad.

9.1.3 SISTEMA EMBEBIDO

El uso de una versión superior del Jetson Nano, como la Jetson Nano Xavier NX, permite una mayor velocidad de procesamiento, mejorando el rendimiento del sistema en todos los escenarios.

La mejora que sería más importante hacer sería la conexión de la placa a una fuente que tuviera el voltaje y las corrientes adecuadas. La conexión directa a la batería del coche con un convertidor reductor de 12 a 5 V sería una de las mejores opciones.

9.1.4 DETECCIÓN DE OBJETOS

Para mejorar este apartado, se podrían utilizar técnicas de *Transfer learning*, para reentrenar la red neuronal utilizada, aplicadas a la situación presentada por el proyecto, en la que la cámara se encuentra en la esquina. Además, también podrías entrenar con imágenes en situaciones adversas como las presentadas en el apartado de resultados, para ver si consigues una mejora sin tener que cambiar el hardware.

10. BIBLIOGRAFÍA



- [1]"El grimorio de bestias: Argos." <https://grimoriodebestias.blogspot.com/2014/04/argos.html> (consultado el 11 de mayo de 2021).
- [2] DGT, "Tablas Estadísticas 2019", 2019. Consultado: 15 de mayo de 2021. [En línea]. <https://www.dgt.es/es/seguridad-vial/estadisticas-e-indicadores/accidentes-30dias/tablas-estadisticas/2019/>.
- [3]"ADAS: Pasado, presente y futuro | Vehicle Service Pros." <https://www.vehicleservicepros.com/service-repair/diagnostics-and-drivability/article/21198482/adas-past-present-and-future> (consultado el 11 de mayo de 2021).
- [4]"¿Qué son las cámaras de visión envolvente de automóviles y por qué son mejores de lo que deberían ser? - ExtremeTech." <https://www.extremetech.com/extreme/186160-what-are-surround-view-cameras-and-why-are-they-better-than-they-need-to-be> (consultado el 11 de mayo de 2021).
- [5]"LIDAR vs Cámara — ¿Cuál es el mejor para los coches autónomos? | por Vincent Tabora | 0xMachina | Medium." <https://medium.com/0xmachina/lidar-vs-camera-which-is-the-best-for-self-driving-cars-9335b684f8d> (consultado el 11 de mayo de 2021).
- [6]"Taller de Automóviles – Historia del OBD." <https://ibtaller.com/historia-del-obd/> (consultado el 11 de mayo de 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/> (consultado el 11 de mayo de 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> (consultado el 12 de mayo de 2021).
- [9]"Guía para principiantes: elija los módulos de cámara adecuados para su kit de desarrollo raspberry pi o jetson nano." <https://www.arducam.com/choose-camera-modules-raspberry-pi-jetson-nano-guide/> (consultado el 12 de mayo de 2021).
- [10] D. Reifs, "Sistemas embebidos".
- [11]"Jetson Nano: Deep Learning Inference Benchmarks | NVIDIA Developer." <https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks> (consultado el 21 de abril de 2021).
- [12]"Buy a Raspberry Pi 4 Model B – Raspberry Pi." <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/> (consultado el 27 de mayo de 2021).
- [13] T. Trnovszký, P. Sýkora y R. Hudec, "Comparison of Background Subtraction Methods on Near Infra-Red Spectrum Video Sequences", en *Procedia Engineering*, enero de 2017, vol. 192, pp. 887–892, doi: 10.1016/j.proeng.2017.06.15.
- [14] D. Zeng, X. Chen, M. Zhu, M. Goesele y 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> (consultado el 13 de mayo de 2021).
- [16]"¿Qué es la tecnología LiDAR?" <https://blog.generationrobots.com/en/what-is-lidar-technology/> (consultado el 14 de febrero de 2021).
- [17]"Deep Learning: what is it go to be a technology clave in the future of artificial intelligence." <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> (consultado el 13 de mayo de 2021).
- [18] K. L. Masita, A. N. Hasan y T. Shongwe, "Deep learning in object detection: A review", agosto de 2020, doi: 10.1109/icABCD49160.2020.9183866.
- [19] D. Sarkar and Towards Data Science, "A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning | por 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> (consultado el 13 de mayo de 2021).
- [20] R. Girshick, J. Donahue, T. Darrell y J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)". Consultado: 13 de mayo de 2021. [En línea]. Disponible: <http://www.cs.berkeley.edu/~rbg/rcnn>.
- [21] S. Ren, K. He, R. Girshick y J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". Consultado: 13 de mayo de 2021. [En línea]. <http://image-net.org/challenges/LSVRC/2015/results>.
- [22]"YOLO: Detección de objetos en tiempo real." <https://pjreddie.com/darknet/yolo/> (consultado el 21 de abril de 2021).
- [23] W. Liu *et al.* , "SSD: Single Shot MultiBox Detector." Consultado: 21 de abril de 2021. [En línea]. <https://github.com/weiliu89/caffe/tree/ssd>.
- [24]"Detección de objetos SSD: Detector multibox de disparo único para el procesamiento en tiempo real | por Jonathan Hui | Medium." <https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06> (consultado el 13 de mayo de 2021).
- [25]"Elm327-interfaz Usb Cable V1.5 Para Diagnostic Coche,Compatible Con Todos Los Protocolos Obd2 Para Windows Elm 327,Scanner 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> (consultado el 27 de mayo de 2021).
- [26]"Getting Started - python-OBD." <https://python-obd.readthedocs.io/en/latest/> (consultado el 14 de mayo de 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/> (consultado el 17 de mayo de 2021).



- [28]"opencv_contrib/evaluación.py amo · opencv/opencv_contrib · GitHub."
https://github.com/opencv/opencv_contrib/blob/master/modules/bgsegm/sample_s/evaluation.py (consultado el 18 de abril de 2021).
- [29]"Background Subtraction with OpenCV and BGS Libraries | Learn OpenCV."
<https://learnopencv.com/background-subtraction-with-opencv-and-bgs-libraries/>
(consultado el 18 de abril de 2021).
- [30] Z. Zivkovic y 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, mayo de 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
(consultado el 18 de abril de 2021).
- [32]"Algoritmo KNN - Finding Nearest Neighbors - Tutorialspoint."
https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm (consultado el 14 de mayo de 2021).
- [33]"OpenCV: Morphological Transformations."
https://docs.opencv.org/4.5.2/d9/d61/tutorial_py_morphological_ops.html
(consultado el 18 de abril de 2021).
- [34]"COCO - Objetos comunes en contexto." <https://cocodataset.org/#home> (consultado el 21 de abril de 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> (consultado el 21 de abril de 2021).
- [36] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications."
- [37]"MobileNet versión 2." <https://machinethink.net/blog/mobilenet-v2/> (consultado el 21 de abril de 2021).
- [38]"[OpenCV Current Combat] 43 Use OpenCV for background segmentation - Programmer Sought." <https://www.programmersought.com/article/10826881459/>
(consultado el 18 de abril de 2021).
- [39]"mAP (precisión media media) podría confundirte! | por Shivy Yohanandan | Towards Data Science." <https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2> (consultado el 31 de mayo de 2021).
- [40]"GitHub - taehoonlee/tensornets: Definiciones de red de alto nivel con pesos previamente entrenados en TensorFlow." <https://github.com/taehoonlee/tensornets>
(consultado el 21 de abril de 2021).
- [41]"The PASCAL Visual Object Classes Homepage."
<http://host.robots.ox.ac.uk/pascal/VOC/> (consultado el 21 de abril de 2021).



- [42] M. Grundmann, V. Kwatra, and I. Essa, "Self-directed video stabilization with robust L1 optimal camera paths," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition*, n.º 1, págs. 225–232, 2011, doi: 10.1109/CVPR.2011.5995525.

11. ANEXO

Anexo 1: Más ejemplos de esquinas donde se puede aplicar el proyecto.....	55
Anexo 2:Clase de proceso de imágenes.....	65
Anexo 3: Código principal.....	72
Anexo 4:Planes para el diseño 3D del proyecto	73

11.1 ESQUINAS



Anexo 1: Más ejemplos de esquinas donde se puede aplicar el proyecto

11.2 CÓDIGO HECHO CON PYTHON

Como se menciona en la metodología, el código consta de dos partes. La primera es una clase creada para controlar los algoritmos de cámara y la segunda se encarga de gestionar la GUI, leer el OBD e integrar las cámaras en el sistema.

11.2.1 CLASE DE PROCESAMIENTO DE IMÁGENES

```
#la parte de la código de ejecución de las cámaras en paralelo se basa en el código siguiente
#la part del codi d'execució en paral·lel 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
```

```
#Importación de librerías
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 para definir la camara
class vStream:
    #inicialització classe, entrada parametres
    def __init__(self, red, width, height, placa, pin_led_persona, pin_led_coche, pin_buzzer):
        #GPIO#
        #definició parametres necessaris per als pins GPIO
        self.placa = placa
        self.pin_led_persona = pin_led_persona
```




```
self.pin_led_coche = pin_led_coche
self.pin_buzzer=pin_buzzer
#definición de los pines como output y inicialización apagados
self.placa.setup(self.pin_led_coche, 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. margen_frames = 5 #numero de frames antes de cambiar GPIO

#Parametros cámara#
#definición tamaño del frame capturado
self.width = width
self.height = height
self.capture = None
# Ulrimo frame cogido por la camara
self.frame = None
self.grabbed = False
self.running = True

#Rconocimiento de objetos#
self.net = red
self.thr = 0.5 # threshold, 0.5 valor por defecto
# definición de los nombres de las clases que se reconoceran
self.clases = ['background', 'truck', 'bicycle', 'bus','car','motorbike','person']
self.classNames = ['person']
self.classVehicle = ['bicycle', 'truck', 'bus', 'car', 'motorbike']
#Parametros del control de GPIO durante reconocimiento

self.led_persona_On = False
self.led_vehicle_On = False
self.detectado_persona = False
self.detectado_vehicle = False
self.contador_detectados_persona = 0 # cuenta veces que se detecta una persona
self.contador_no_detectados_persona = 0
self.contador_detectados_vehicle = 0 # cuenta veces que se detecta una vehiculo
self.contador_no_detectados_vehicle = 0

#Background Subtraction#
self.aumento_cuadro = 15 #valor que aumenta el bounding box
self.color = (255, 255, 255)
#creación background subtraction y inicialicació parametros
self.fgbg = cv2.createBackgroundSubtractorKNN()
self.fgbg.setHistory(100)
self.fgbg.setNSamples(10)
self.fgbg.setkNNSamples(3)

#definicio kernels usados
self.kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (5, 5))
self.kernel1 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))

#Thread#
```



```
#parametros para el control del thread
self.read_thread = None
self.read_lock = threading.Lock()
self.running = False
```

```
#función para inicializar las camaras
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 hay un error al intentar abrir la cámara
        self.capture = None
        print("Imposible abrir camera")
        print("Pipeline: " + gstreamer_pipeline_string)
        return
```

```
#Función para inicializar frames y llama a función para adquirir nuevos frames
def start(self):
    if self.running:
        print("Ya esta capturando video")
        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
```

```
#Funcion para parar el thread
def stop(self):
    self.running = False
    self.read_thread.join()
```

```
#Funcion para actualizar la cámara y coger nuevos frames, llamando al 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: #comprovación de errors
            print("falla captura cámara", err)
```

```
except RuntimeError:  
    print("No se puede leer la cámara")
```

#Función para obtener framers sin ningún tipo de procesamiento, mantienen parados gpio

```
def getFrame(self):  
    print("frame")  
    self.placa.output(self.pin_led_coche, 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
```

#Función para aplicar el background subtraction

```
def Background_Subtraction(self):  
    try:  
        self.blur= cv2.GaussianBlur(self.frame2, (15, 15), 0) # Difumina l'imagen  
        self.fgmask = self.fgbg.apply(self.blur)# Calcu la mascara de fondo  
        _, self.binary = cv2.threshold(self.fgmask, 50, 255, cv2.THRESH_BINARY) # Asegura que la  
mascara sea binaria  
        self.open = cv2.morphologyEx(self.binary, cv2.MORPH_OPEN, self.kernel) # reduce el ruido de  
la imagen  
  
        self.close= cv2.morphologyEx(self.open, cv2.MORPH_CLOSE, self.kernel1) # hace que la  
mascara quede mas limpia  
  
        self.dilate = cv2.morphologyEx(self.close, cv2.MORPH_DILATE, self.kernel1) # expande la  
mascara
```

#primera detección del contorno y única en caso de que solo haya un contorno

```
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.aumento_cuadro),  
                                (int(self.boundRect[self.i][1]) - self.aumento_cuadro),  
                                ((int(self.boundRect[self.i][0] + self.boundRect[self.i][2]) + self.aumento_cuadro),  
                                (int(self.boundRect[self.i][1] + self.boundRect[self.i][3]) + self.aumento_cuadro),  
self.color, cv2.FILLED)  
  
self.draw = cv2.cvtColor(self.drawing, cv2.COLOR_BGR2GRAY)
```



```
# segunda detección del contorno, combina cuadros en contacto
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 con el área 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)

#filtro segunda detección contornos
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.aumento_cuadro),
(int(self.boundRect[self.h][1])) - self.aumento_cuadro), \
            ((int(self.boundRect[self.h][0] + self.boundRect[self.h][2]) + self.aumento_cuadro),
(int(self.boundRect[self.h][1] + self.boundRect[self.h][3])) + self.aumento_cuadro),
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: #comprueba errores
    print("error background Subtraction ", fallo)

#función para el reconocimiento de objetos con background subtraction aplicado
def reconocimiento_con_bg(self):
    try:
        #aplica background Subtraction
        self.res_box=self.Background_Subtraction()
        self.backgroud_resized= cv2.resize(self.res_box, (300, 300)) #reescal la imateg
        self.copia_frame = self.frame2.copy()
        self.backgroud_resized = cv2.cvtColor(self.backgroud_resized, cv2.COLOR_BGR2RGBA)

        # aplica la función para transformar la imagen en formato 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) #funcion reconocimiento optimizada
para cada
    self.detectado_persona = False
    self.detectado_vehicle = False
    for self.detect in self.detections: #loop que recorre todass las detecciones obtenidas
        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 por clase y threshold
        print(self.item)

# Control GPIO
# Si uno de los leds no esta ctivo y el contador es menor de 5 pitara,y encenderá la luz
if self.item in self.classVehicle:
    self.detectado_vehicle=True
    if (not self.led_vehicle_On) and self.contador_no_detectados_vehicle >
self.margen_frames: # si el led no esta encendido
        #enciende led correspondiente a la clase
        print("suen buzzer")
        self.placa.output(self.pin_buzzer, self.placa.HIGH)
        self.led_vehicle_On = True
        self.contador_no_detectados_vehicle = 0
        self.placa.output(self.pin_led_coche, self.placa.HIGH)
        print("enciende led vehiculo")

if self.item in self.classNames:
    self.detectado_persona = True
    if (not self.led_persona_On) and self.contador_no_detectados_persona >
self.margen_frames: # si el led no esta encendido

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

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

if self.detectado_persona: #aumenta numero detectados
    self.contador_detectados_persona += 1
    if self.contador_detectados_persona > 2:
        self.placa.output(self.pin_buzzer, self.placa.LOW)
        print("detectado", self.contador_detectados_persona)

else: #aumenta el numero de detetctados
```



```
self.contador_no_detectados_persona += 1
print("no detectado", self.contador_no_detectados_persona)
if self.contador_no_detectados_persona>2:
    self.placa.output(self.pin_buzzer, self.placa.LOW)

if self.detectado_vehicle: #aumenta numero de detectados
    self.contador_detectados_vehicle += 1
    if self.contador_detectados_vehicle > 2:
        self.placa.output(self.pin_buzzer, self.placa.LOW)
    print("detectado", self.contador_detectados_vehicle)

else: #aumenta el numero de no detectados
    self.contador_no_detectados += 1
    print("no detectado", self.contador_no_detectados_vehicle)
    if self.contador_no_detectados>5:
        self.placa.output(self.pin_buzzer, self.placa.LOW)

#if not self.detectado: # si el led ON y no detectado
if (not self.detectado_vehicle) and self.contador_detectados_vehicle > self.margen_frames:
    if self.led_vehicle_On:
        print("led vehiculo apagado")
        self.placa.output(self.pin_led_coche, self.placa.LOW)
        self.led_vehicle_On = False
        self.contador_detectados_vehicle = 0

if (not self.detectado_persona) and self.contador_detectados_persona > self.margen_frames:
    if self.led_persona_On:
        print("led persona apagado")
        self.placa.output(self.pin_led_persona, self.placa.LOW)
        self.led_persona_On = False
        self.contador_detectados_persona = 0

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

#Reconocimiento sin background subtraction
def reconocimiento_sin_bg(self):
    try:
        # coge el frame directamente de la camara
        self.background_resized = cv2.resize(self.frame2, (300, 300))
        self.copia_frame = self.frame2.copy()
        self.background_resized = cv2.cvtColor(self.background_resized, cv2.COLOR_BGR2RGBA)
        self.background_resized = utils.cudaFromNumpy(self.background_resized)
        self.cols = self.background_resized.shape[1]
        self.rows = self.background_resized.shape[0]

        #aplica reconocimiento con cuda
        self.detections = self.net.Detect(self.background_resized)
```



```
self.detectado_persona = False
self.detectado_vehicle = False
for self.detect in self.detections: #loop que recorre todas las detecciones obtenidas
    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 por clase y threshold
        print(self.item)

# Control GPIO
# Si uno de los leds no esta ctivo y el contador es menor de 5 pitara,y encenderá la luz
if self.item in self.classVehicle:
    self.detectado_vehicle=True
    if (not self.led_vehicle_On) and self.contador_no_detectados_vehicle >
self.margen_frames: # si el led no esta encendido
        #enciende led correspondiente a la clase
        print("suen buzzer")
        self.placa.output(self.pin_buzzer, self.placa.HIGH)
        self.led_vehicle_On = True
        self.contador_no_detectados_vehicle = 0
        self.placa.output(self.pin_led_coche, self.placa.HIGH)
        print("enciende led vehiculo")

if self.item in self.classNames:
    self.detectado_persona = True
    if (not self.led_persona_On) and self.contador_no_detectados_persona >
self.margen_frames: # si el led no esta encendido

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

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

if self.detectado_persona: #aumenta numero detectados
    self.contador_detectados_persona += 1
    if self.contador_detectados_persona > 2:
        self.placa.output(self.pin_buzzer, self.placa.LOW)
        print("detectado", self.contador_detectados_persona)

else: #aumenta el numero de no detetctados
    self.contador_no_detectados_persona += 1
    print("no detectado", self.contador_no_detectados_persona)
    if self.contador_no_detectados_persona>2:
```



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

if self.detectado_vehicle: #aumenta numero de detectados
    self.contador_detectados_vehicle += 1
    if self.contador_detectados_vehicle > 2:
        self.placa.output(self.pin_buzzer, self.placa.LOW)
        print("detectado", self.contador_detectados_vehicle)

else: #aumenta el numero de no detectados
    self.contador_no_detectados += 1
    print("no detectado", self.contador_no_detectados_vehicle)
    if self.contador_no_detectados>5:
        self.placa.output(self.pin_buzzer, self.placa.LOW)

#if not self.detectado: # si el led ON y no detectado
if (not self.detectado_vehicle) and self.contador_detectados_vehicle > self.margen_frames:
    if self.led_vehicle_On:
        print("led vehiculo apagado")
        self.placa.output(self.pin_led_coche, self.placa.LOW)
        self.led_vehicle_On = False
        self.contador_detectados_vehicle = 0

if (not self.detectado_persona) and self.contador_detectados_persona > self.margen_frames:
    if self.led_persona_On:
        print("led persona apagado")
        self.placa.output(self.pin_led_persona, self.placa.LOW)
        self.led_persona_On = False
        self.contador_detectados_persona = 0

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

#funcion para definir parámetros de la cámara 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,  
)  
)
```

Anexo 2 Clase de proceso de imágenes

11.2.2 CÓDIGO PRINCIPAL

```
#Se important las librerias  
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 OBDSStatus  
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()

# definicion de la variable global velocidad para poder acceder a ella
vel = 0
velocidad_alta=True
sale =True

#funcion para poder realizar la conexión con el obd, mediante thread
def connect(n):
    n = "bien"
    rapido = True
    ports = obd.scan_serial() # escanea puertos usb, y devuelve los disponibles

    print(ports) #mostra els ports trobats

    #establece una conexión asíncrona para poder leer los valores del coche
    connexion = obd.Async(ports[0],baudrate=38400, protocol="3", fast=rapido, timeout=30)
    while len(connexion.supported_commands) < 10: #intenta connectar fins que s'obtenen més de
10 commandaments funcionals
        try:
            connexion.stop()
            time.sleep(5)# para el thread 5 segundos para intenatr que se connecte sin problemas

            connexion = obd.Async(ports[0], 38400, "3", fast=rapido, timeout=30)

        except Exception as error:
            print(error)

    print("acaba")
    if OBDStatus.CAR_CONNECTED: #confirma la conexión con el coche
        print("conectado")
    return connexion

#funcion para obtener el valor de velocidad por el puerto serie
def new_value1(velocidad):
    #definicion de variables globales para poder usarlas en todo el programa
    global velocidad_alta
    global sale
    global vel
    vel = velocidad.value.magnitude # asigna valor a la variable velocidad
    print(vel)
    #condicional de regulación de la gui
    if vel < 2:
        velocidad_alta = False

    else:
        velocidad_alta = True
```

#Funcion para regular el tiempo de la barra de carga

```
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()): #comprueba si el thread de conexión ha terminado
        window['-THREAD-'].click()
        break
    print("threading1", progress)

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

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

    # ----- creación layouts de las pantallas-----
    gif = "parche.png" #imagen inicial de la pantalla
    altura = 400
    ancho_1 = 650
    ancho_2 = 490
    #Layout de carga 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 cámara izquierda
    layout2 = [
        [sg.Image(filename=gif, key='image1', pad=(200, 0), size=(ancho_1, altura))],
        [sg.Text('Camera izquierda', size=(40, 1), justification='center', font='Helvetica 20')]]

    # layout dos camaras
    layout3 = [[sg.Image(filename=gif, key='image2', size=(ancho_2, altura)),
               sg.Image(filename=gif, key='image3', size=(ancho_2, altura))],
               [sg.Text('Dos Camaras', size=(40, 1), justification='center', font='Helvetica 20')]]

    # layout camara derecha
    layout4 = [
        [sg.Image(filename=gif, key='image4', pad=(200, 0), size=(ancho_1, altura))],
        [sg.Text('Camera derecha', size=(40, 1), justification='center', font='Helvetica 20')]]

    layout_movimiento = [
        [sg.Image(filename=gif, background_color='white', key='-IMAGE2-', pad=(200, 0), size=(ancho_1,
        altura))],
```



```
[sg.Text('Velocidad superior a 2km/h', size=(40, 1), justification='center', font='Helvetica 20')]]

#layout de los botones
layout5 = [[sg.Button('Izquierda', size=(20, 1), pad=(2, 2), font='Helvetica 14', visible=True,
disabled=True),
    # cambiar per icono
    sg.Button('Dos', size=(20, 1), pad=(2, 2), font='Helvetica 14', visible=True, disabled=True),
    # cambiar per icono
    sg.Button('Derecha', 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 que combina los anteriores, permite activar y desactivar los demás layouts
layout = [[sg.Text('ATENCIÓN! Vigila los alrededores por seguridad, 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_movimiento, 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-
')]]

#definición de la ventana con dimensiones determinadas
window = sg.Window('Argos Cam', layout, resizable=False, size=(1024, 600)).Finalize()

timeout = thread = None

acelerar_barra_progreso = False
timeout = None
t = 50 #segundos máximos de tiempo de iniciación
c = "a" #argumento usado para la entrega de parámetros en cola

#threads para la pantalla de carga
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()

#elemento para cargar la barra desde el punto que se queda el thread
time_ = 0
```



```
#loop de la pantalla de carga
while True: # Event Loop
    event, values = window.read(timeout=10)
    if event in (None, 'Exit', 'Cancel') or time_ > 100: #Si el tiempo se acaba, o se sale del thread se
cierra el loop
        break

    if event == '-SEC-': #click interno del boto invisible para aumentar lka velocidad de carga
        #la, barra accelera una vez hecha la connexionn
        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 el thread ha terminado, devuelve el objeto conexión
if event == '-THREAD-':
    #Cierra los threads
    thread.join(timeout=0)
    connexion = que.get()
    #connexion.watch(obd.commands.SPEED, callback=new_value1)
    #connexion.start()
    thread1.join(timeout=0)
    print('Thread finished', thread.is_alive(), thread1.is_alive())
    acelerar_barra_progreso = True
    window['-SEC-'].click()
    # break

#define las dimensiones de las ventanas según el método de pantalla seleccionada
dispW = 640
dispH = 480
altura_ = 400
ancho_1_ = 650
ancho_2_ = 490

#define los pines usados en la placa
Jetson = GPIO
Jetson.setmode(GPIO.BCM)
pins = [18, 17, 7, 27, 10]e
Jetson.setup(pins, GPIO.OUT, initial=GPIO.LOW)

#inicia la red neuronal usada en el programa
red = inference.detectNet("ssd-mobilenet-v2", threshold=0.5)

#Se crean los objetos Vstream con los parámetros necesarios para la captura de frames

camara1 = vStream(red, dispW, dispH, Jetson, 18, 17, 7)
camara1.open(gstreamer_pipeline(flip_method=2))
```

```
camara1.start()
camara2 = vStream(red, dispW, dispH, Jetson, 27, 10, 7)
camara2.open(gstreamer_pipeline(sensor_id=1, flip_method=2))
camara2.start()

# empieza a adquirir el valor de la velocidad de vehiculo
connexion.watch(obd.commands.SPEED, callback=new_value1)
connexion.start()
window['f-COL1-'].update(visible=False)
layout_visible = 3
window['f-COL{layout_visible}-'].update(visible=True)

# Hace visibles los botones para cambiar entre las camaras
window['Izquierda'].update(disabled=False)
window['Dos'].update(disabled=False)
window['Derecha'].update(disabled=False)
window['-IA-'].update(disabled=False)

# tamaño de redimensión de la camara
dim_1 = (ancho_1_, altura_)
dim_2 = (ancho_2_, altura_)
old_slider = 10
layout_movimiento = True

#variable para controlar algoritmo activo
Inteligente = 2 # 0 desactivado, 1 sin BG, 2, con 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 se clicca exit o se cierra la ventana el programa acaba        break

    elif event == 'Izquierda': # Si se clicca el botón, se enseñara únicamente esta cámara

        #desactiva layout actual
        window['f-COL{layout_visible}-'].update(visible=False)
        layout_visible = 2
        window['f-COL{layout_visible}-'].update(visible=True)

    elif event == 'Dos':
        window['f-COL{layout_visible}-'].update(visible=False)
        layout_visible = 3
        window['f-COL{layout_visible}-'].update(visible=True)

    elif event == 'Derecha':

        window['f-COL{layout_visible}-'].update(visible=False)
        layout_visible = 4
        window['f-COL{layout_visible}-'].update(visible=True)
```

```
elif event == '-IA-':
    # Seleccióna el algoritmo de captura de imágenes y actualiza el texto del boton

    if Inteligente == 0:
        window['-IA-'].update('Activar Modo 2 IA')
        Inteligente = 1

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

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

if velocidad_alta:
    # Modifica el layout visible en función de la variable velocidad, se hace de forma
    automatica
    window[f'-COL{layout_visible}-'].update(visible=False)
    window['-COL5-'].update(visible=True)
    layout_movimiento = True

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

if layout_movimiento == False: # si el coche va a una velocidad inferior a 2km/h o esta parado
if layout_visible == 2: # enseña cámara izquierda
    if Inteligente == 2:
        reconocimiento, frame = camara1.reconocimiento_con_bg()

    elif Inteligente == 1:
        reconocimiento, frame = camara1.reconocimiento_sin_bg()
    else:
        frame = camara1.getFrame()

    # codificación imagen para pysimple gui
    imgbytes = cv2.imencode('.png', cv2.resize(frame, dim_1))[1].tobytes()
    window['image1'].update(data=imgbytes)

elif layout_visible == 4:
    if Inteligente == 2:
        reconocimiento1, frame1 = camara2.reconocimiento_con_bg()

    elif Inteligente == 1:
        reconocimiento1, frame1 = camara2.reconocimiento_sin_bg()
```



```
else:
    frame1 = camara2.getFrame()

    imgbytes1 = cv2.imencode('.png', cv2.resize(frame1, dim_1))[1].tobytes() # ditto
    window['image4'].update(data=imgbytes1)

else:
    if Inteligente == 2:
        reconocimiento, frame = camara1.reconocimiento_con_bg()
        reconocimiento1, frame1 = camara2.reconocimiento_con_bg()

    elif Inteligente == 1:
        reconocimiento, frame = camara1.reconocimiento_sin_bg()
        reconocimiento1, frame1 = camara2.reconocimiento_sin_bg()

    else:
        frame = camara1.getFrame()
        frame1 = camara2.getFrame()

    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)

#limpia los objetos usados para liberar memoria
window.close()
camara1.stop()
connexion.close()
camara1.capture.release()
camara2.stop()
camara2.capture.release()
GPIO.cleanup()

#funcio principal
if __name__ == '__main__':

    main ()
```

Anexo 3: Código principal

11.3 DISEÑAR PLANES

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

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)	

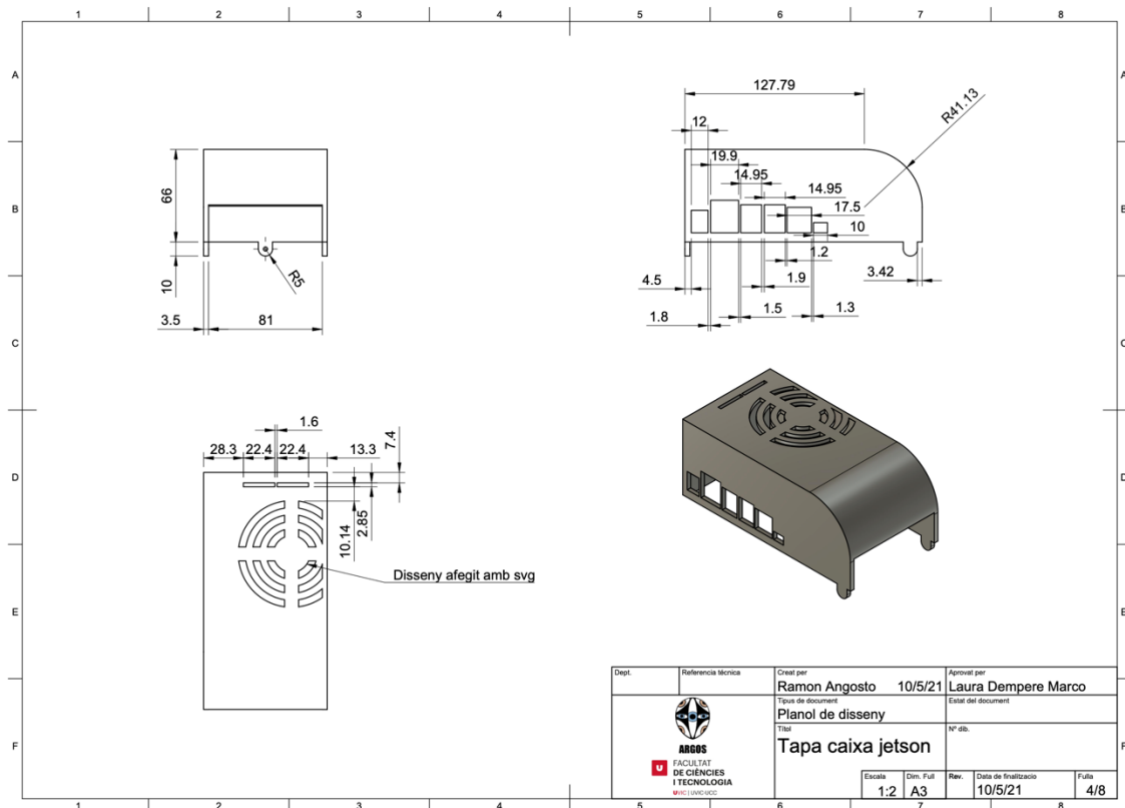
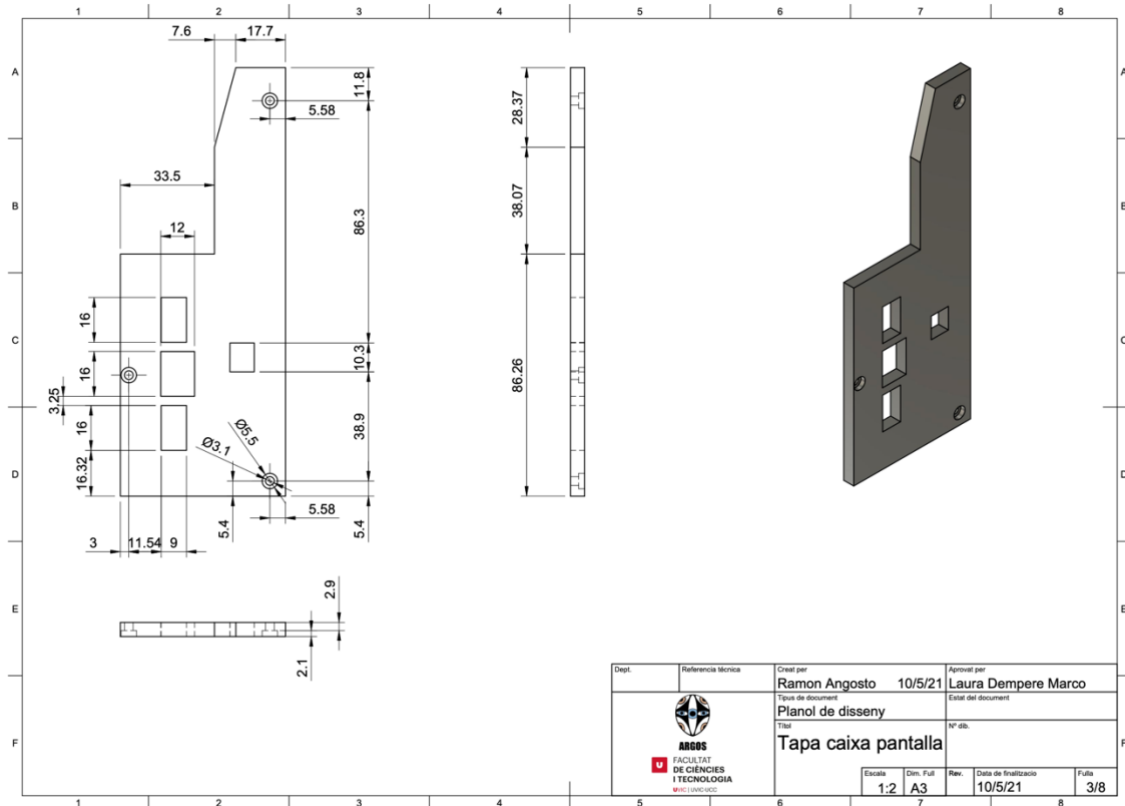
Taula de parts

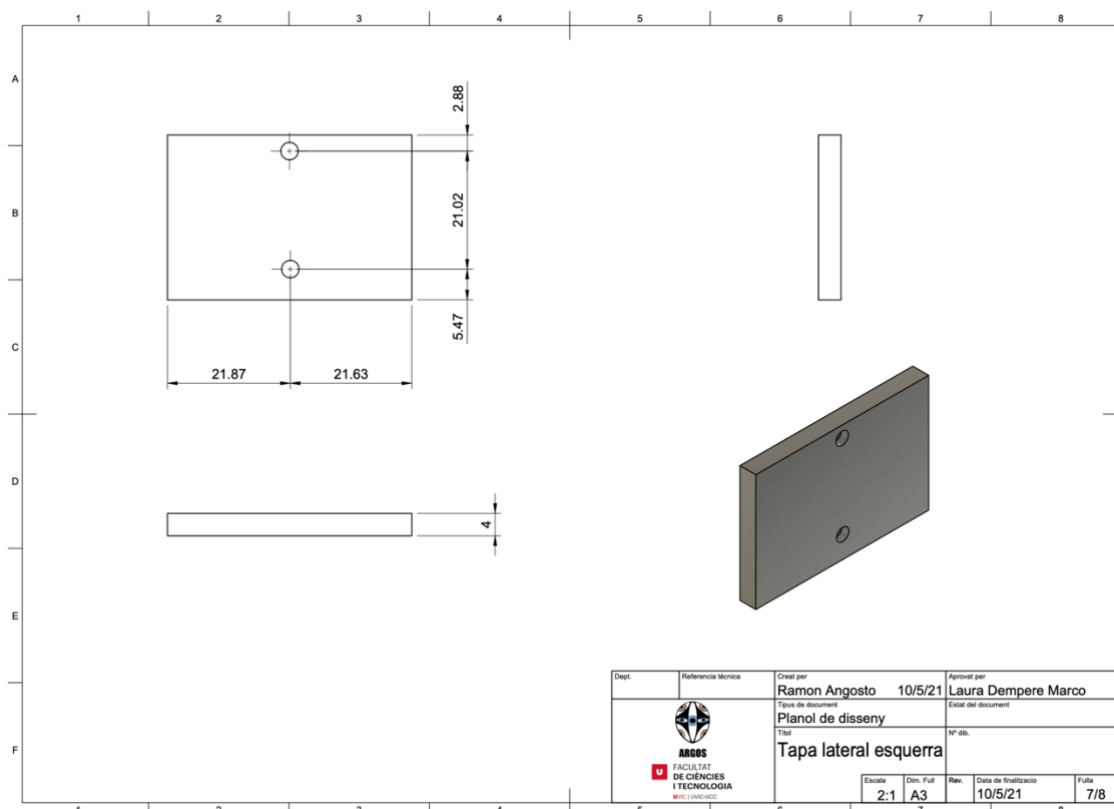
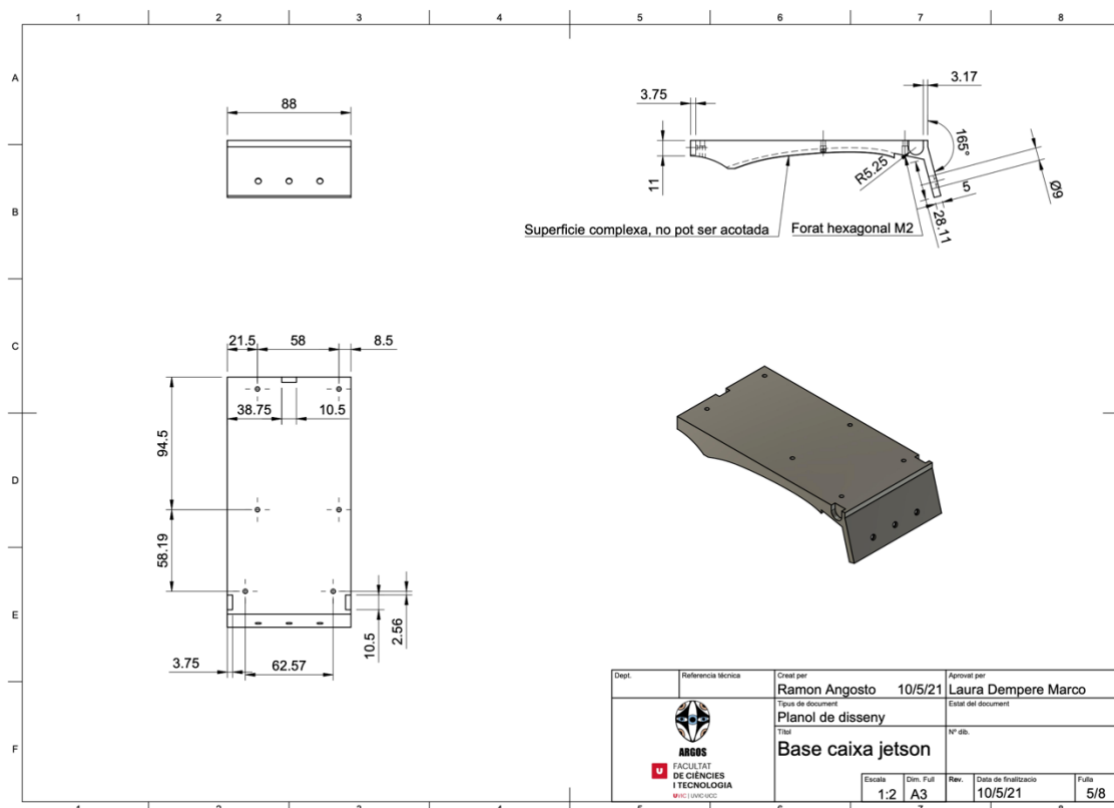
Dept.	Referencia tècnica	Disseñat per	Revisat per	Estad del document
		Ramon Angosto	10/5/21	Laura Dempere Marco
		Tipus de document		
		Ensamblatge		
		Títol		
		Encapsulat interior cobre		
		Nº dib.		
		Escala		
		1:5		
		Dim. Full		
		A3		
		Rev.		
		Data de finalització		
		10/5/21		
		Fulls		
		1/8		

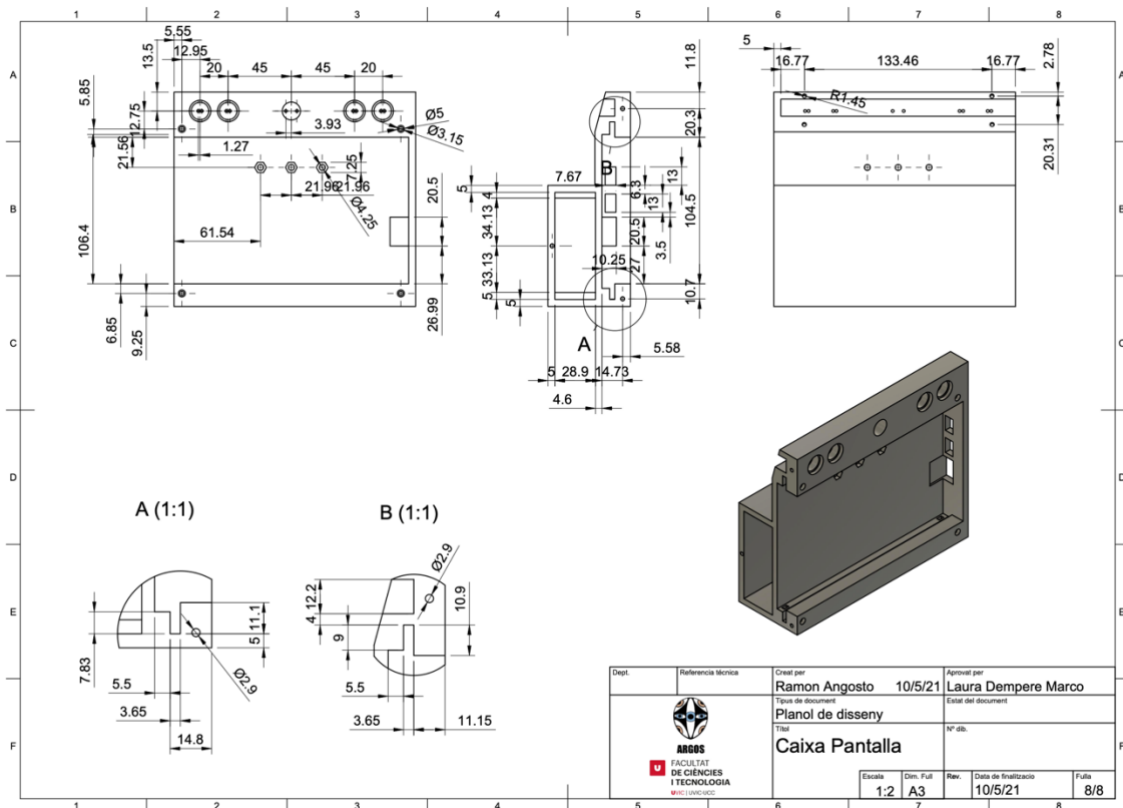
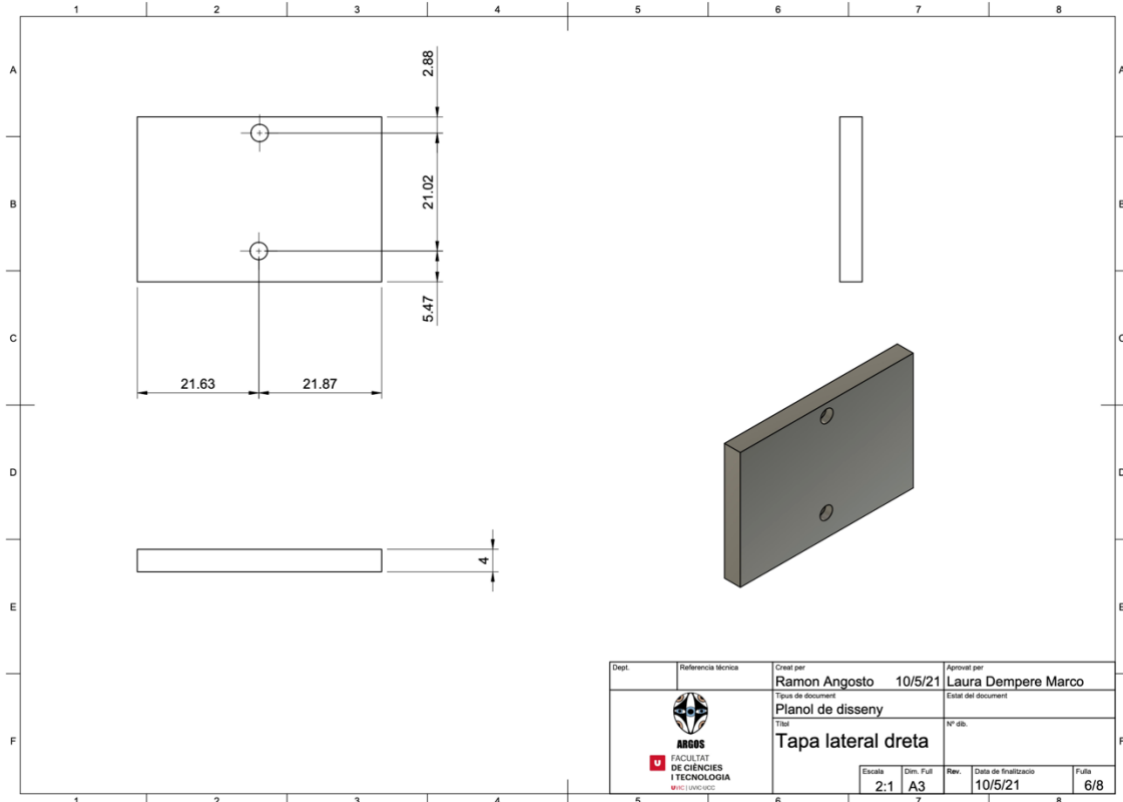
Planol de disseny

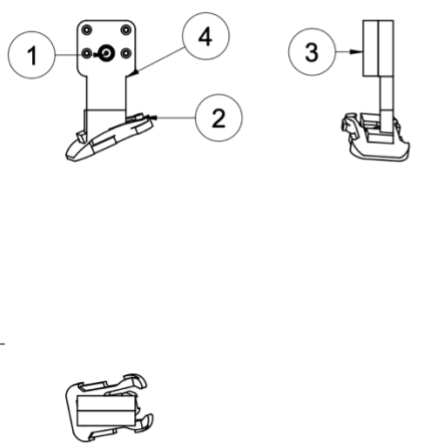
Spacer

Dept.	Referencia tècnica	Disseñat per	Revisat per	Estad del document
		Ramon Angosto	10/5/21	Laura Dempere Marco
		Tipus de document		
		Planol de disseny		
		Títol		
		Spacer		
		Nº dib.		
		Escala		
		10:1		
		Dim. Full		
		A3		
		Rev.		
		Data de finalització		
		10/5/21		
		Fulls		
		2/8		






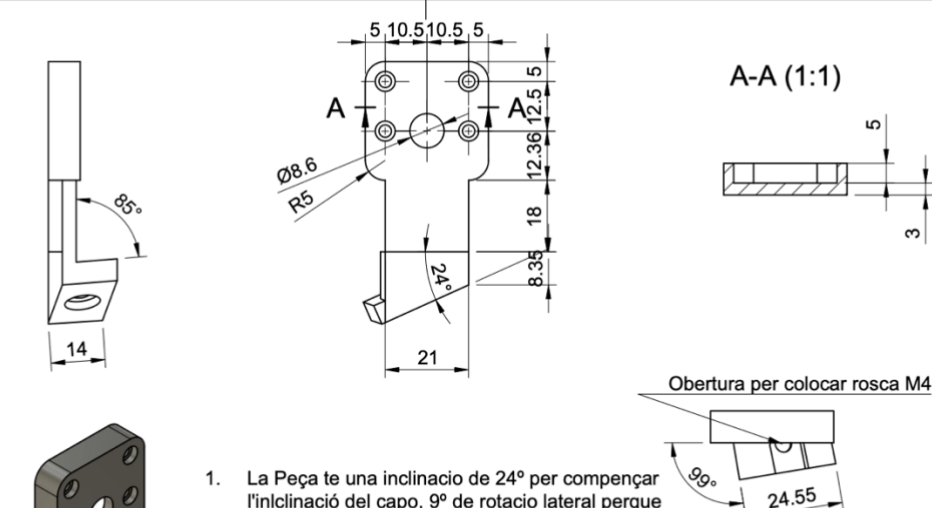





10	2	Cargol M4x10	
9	2	Arandela M4	
8	2	Rosca M4	
7	4	Cargol M2x16	
6	4	Arandela M2	
5	4	Rosca M2	
4	2	Part posterior carcasa	PLA
3	2	Part frontal carcasa	PLA
2	2	Clip Gopro (Internet)	PLA
1	2	Raspi Cam v2 (Internet)	
Numero	Quantitat	Nom part	Material

Lista de parts

Dept.	Referència tècnica	Creat per	Aprovat per	
 FACULTAT DE CIÈNCIES I TECNOLOGIA UVIC UVIC-UCC		Tipus de document Ensamblatge	Estat del document Acabat	
		Títol Càmera TFG	Nº Dib.	
		Escala 1:2	Dim. full A4	Rev.
		Data acabat 11/5/21	Fulla	



1. La Peça te una inclinació de 24° per començar l'inclinació del cap, 9° de rotació lateral perquè apunti una mica cap endavant i 5 perquè no apunti al terra.

Dept.	Referència tècnica	Creat per	Aprovat per
 FACULTAT DE CIÈNCIES I TECNOLOGIA UVIC UVIC-UCC		Ramon Angosto 11/5/21	Laura Dempere Marco
		Tipus de document Planol de disseny	Estat del document Acabat
		Títol Part frontal càmera	Nº Dib.
		Escala 1:1	Dim. full A4
	Data acabat 11/5/21	Fulla 2/3	

