Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de un Sistema Embebido Low Cost de Visión Artificial Apoyado por Sensores para el Control de Conformidad Industrial

Development of a Low Cost Embedded Machine Vision System Supported by Sensors for Industrial Compliance Control

Autor

Jorge Lasarte Aneiros

Director

Javier Esteban Escaño

Escuela Universitaria Politécnica La Almunia

Octubre 2024

Agradecimientos

El presente trabajo no habría sido posible sin el apoyo y la compañía de las personas que han estado a mi lado en este largo camino.

Gracias a mi madre, por su amor incondicional y su fortaleza, siempre presente y apoyándome en cada paso que he dado; no sería lo que soy hoy, sin tu figura.

A mi padre, que está atravesando un momento difícil en su lucha contra una enfermedad, por enseñarme con su ejemplo lo que realmente significa la valentía, el esfuerzo y luchar por lo que uno ama de verdad en la vida.

A mi hermano, cuya protección y respaldo incondicional han sido fundamentales a lo largo de este camino, brindándome confianza incluso en los momentos más inciertos.

Resumen

El trabajo de fin de grado descrito en este documento consiste en el desarrollo e implementación de un Sistema Embebido de Bajo Coste de Visión Artificial para el Control Industrial de Conformidades en una línea de producción. El proyecto se ha llevado a cabo en la planta industrial de Stellantis en Figueruelas, y más específicamente al final de la línea dos en la etapa conocida como TRIM. En esta etapa, se instalan muchos de los componentes que tienen relación con la comodidad, la apariencia y el funcionamiento del interior del automóvil.

El sistema se divide en tres partes esenciales. La primera parte consiste en el uso de dos cámaras industriales conectadas mediante Ethernet y comunicadas utilizando el protocolo GenICam sobre GigE Vision para capturar información referida a algunas de las piezas y componentes que lleva el automóvil. Esto se lleva a cabo tomando imágenes a una hoja de montaje situada en la parte inferior del capo del coche que contiene dicha información y procesándola con un software de Reconocimiento Óptico de Caracteres (OCR).

En la segunda parte, la Jetson Xavier NX (Plataforma de computación embebida), que actúa como "cerebro" del sistema, controla este proceso junto con otro adicional, donde se obtienen imágenes de la parte lateral, superior trasera y superior delantera del automóvil mediante cámaras FPD LINK situadas sobre unos perfiles de aluminio a ambos lados de la línea. Además de estas, existen otras cámaras Blackfly S distribuidas a lo largo de estos perfiles para tomar imágenes tanto de la parte frontal como trasera del automóvil, también a ambos lados de la línea. Añadimos un sensor laser fotoreflectivo que actúa a modo de disparo o trigger para tomar todas estas imágenes mediante software. Estas imágenes son posteriormente analizadas por la red neuronal Yolov5 para determinar qué componentes se han ensamblado en el vehículo.

Finalmente, la información obtenida a través de la red neuronal y del OCR se contrasta para identificar si hay discrepancias entre las piezas presentes en el vehículo y los que realmente deberían estar. Si se detecta alguna discrepancia, se notifica a los operarios mostrando el fallo del vehículo en un televisor situado en la línea de producción. Además, el sistema cuenta con un servidor donde los supervisores del puesto pueden consultar todas las imágenes e información obtenida por el sistema.

Este sistema permite la detección temprana de errores en la instalación de piezas en una de las etapas de ensamblaje del vehículo, reduciendo considerablemente los costos de reparación. Además, su diseño está pensado para facilitar su implementación en otras líneas de producción dentro de la planta. Hasta la fecha de presentación de este trabajo, el sistema ha analizado más de 300.000 coches.

Abstract

The final degree project described in this document consists of the development and implementation of a Low-Cost Embedded System for Computer Vision in Industrial Compliance Control on a production line. The project has been carried out at Stellantis' industrial plant in Figueruelas, specifically at the end of line two during the stage known as TRIM. In this stage, many of the components related to the comfort, appearance, and functioning of the car's interior are installed.

The system is divided into three essential parts. The first part involves the use of two industrial cameras connected via Ethernet using GenICam over GigE Vision protocol to capture information related to some of the parts and components installed in the car. This is done by taking images of an assembly sheet located under the car's hood, which contains this information, and processing it using Optical Character Recognition (OCR) software.

In the second part, the Jetson Xavier NX (embedded computing platform), which acts as the "brain" of the system, controls this process along with an additional one, where images are obtained from the car's side, rear upper, and front upper parts using FPD LINK cameras positioned on aluminum profiles on both sides of the line. In addition to this, other Blackfly S cameras are distributed along these profiles to take images of both the front and rear parts of the car, also on both sides of the line. A laser photo-reflective sensor is added to act as a trigger for capturing these images via software. These images are then analyzed by the Yolov5 neural network to determine which components have been assembled on the vehicle.

Finally, the information obtained through the neural network and OCR is cross-checked to identify any discrepancies between the parts present in the vehicle and those that should actually be there. If any discrepancies are detected, the operators are notified by displaying the fault on a screen located on the production line. Additionally, the system includes a server where the workstation supervisors can consult all the images and information gathered by the system.

This system enables early detection of errors in part installation during one of the vehicle assembly stages, significantly reducing repair costs. Moreover, its design is aimed at facilitating its implementation in other production lines within the plant. As of the date of this presentation, the system has analyzed more than 300.000 cars.

Palabras clave

Sistema embebido; visión artificial; control industrial de conformidades, Reconocimiento Óptico de Caracteres (OCR); redes neuronales; Yolov5; ensamblaje de vehículos; reducción de costes; sensor láser foto-reflectivo.

Key words

Embedded system; computer vision; industrial compliance control; Optical Character Recognition (OCR); neural networks; Yolov5; vehicle assembly; cost reduction; laser photo-reflective sensor.

Escuela Universitaria Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

MEMORIA

Desarrollo de un Sistema Embebido Low Cost de Visión Artificial Apoyado por Sensores para el Control de Conformidad Industrial

Development of a Low-Cost Embedded Machine Vision System Supported by Sensors for Industrial Compliance Control

Autor:	Jorge Lasarte Aneiros
Director empresa:	Ignacio Dalda Rivas
Director universidad:	Javier Esteban Escaño
Fecha:	10 2024

Página intencionadamente en blanco.

INDICE DE CONTENIDO BREVE

INDICE DE CONTENIDO

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

1. INTRODUCCIÓN

1.1. PRE-INTRODUCCIÓN

El año 2023 marca un punto de inflexión en la evolución de la Inteligencia Artificial (IA), consolidándose como una herramienta esencial en múltiples sectores, especialmente en la industrial moderna. La visión artificial, en particular, ha demostrado ser de un valor incalculable en el control de calidad, permitiendo automatizar inspecciones, reducir tiempos y costes, mejorar la fiabilidad y minimizar errores humanos.

En el sector automovilístico, uno de los mercados más competitivos a nivel global, la adopción de tecnologías avanzadas es crucial para mantener la competitividad. La inversión en innovaciones como la IA no solo responde a esta necesidad, sino que también impulsa la eficiencia y la sostenibilidad en los procesos de producción.

Este Trabajo de Fin de Grado surge de una necesidad detectada en la planta de Stellantis en Figueruelas, donde la correcta verificación de componentes durante el ensamblaje en una de sus líneas de producción representa un desafío clave. Tradicionalmente, los controles de conformidad se realizan en etapas posteriores del proceso productivo, lo que incrementa significativamente los costos de corrección cuando se detectan errores.

Para resolver este problema, se propone un sistema embebido de visión artificial diseñado para detectar discrepancias en tiempo real durante el ensamblaje. Este enfoque permite una identificación temprana de errores, reduciendo costes y optimizando los procesos. El sistema no solo está concebido para abordar esta problemática en particular, sino que su diseño pretende ser modular y adaptable para facilitar su implementación en otras áreas de la planta.

1.2. METODOLOGÍA

La metodología empleada para desarrollar este proyecto ha sido la PDCA (*Plan-Do-Check-Act*). Comienza con la etapa de planificación, en la cual se definen los requisitos y se asignan los recursos necesarios. A continuación, en la fase hacer, se implementa el plan propuesto. Seguidamente, en la fase verificar, se examinan los resultados y se contrastan con los objetivos preestablecidos. Dependiendo de estos hallazgos, en la etapa actuar, se implementan acciones correctivas para resolver problemas y promover alguna mejora.

Esta secuencia de pasos se aplica en cada fase del proyecto, lo que permite un plan de acción dinámico y la corrección temprana de posibles errores.

1.3. ORGANIZACIÓN DEL DOCUMENTO

El trabajo presentado se estructura en cuatro apartados principales, seguidos de uno final con las conclusiones del proyecto.

El presente y primer capítulo está dedicado a explicar la problemática a la que enfrentamos en el proyecto, la justificación de la solución encontrada, los objetivos planteados y finalmente se exponen los requisitos del sistema a desarrollar, tanto funcionales como no funcionales. El segundo apartado habla acerca del estado del arte de los sistemas de control de conformidad industrial en la planta de Stellantis en Figueruelas. En la tercera parte se describe detalladamente la sección donde se va a implementar el sistema, el diseño tanto a nivel hardware como software. Finalmente, en quinto apartado, se exponen las conclusiones y las futuras implementaciones y mejoras del sistema.

Además, junto a este documento, se incluyen varios anexos que proporcionan información adicional sobre diferentes aspectos del trabajo realizado, incluyendo un glosario, y planos.

1.4. PLANTEAMIENTO DEL PROBLEMA

En el proceso de producción de vehículos, la correcta instalación de componentes es esencial para garantizar tanto la funcionalidad como la calidad del producto final. Sin embargo, en la planta de Stellantis en Figueruelas, se enfrenta un desafío en la etapa TRIM de ensamblaje: la verificación de conformidad de los componentes instalados en los vehículos se realiza en una fase tardía del proceso, cuando el costo de corregir el error es considerablemente más alto.

En la actualidad, no existe un sistema automatizado que permita comprobar, en tiempo real, si los componentes ensamblados en un vehículo coinciden con las especificaciones del pedido. Este control depende de procesos manuales o inspecciones tardías en otras áreas de la planta que no solo son propensas a errores humanos, sino que también generan retrasos y aumentan los costes de reparación y almacenamiento de los vehículos.

Dada la escala de producción de la planta, que supera los 1700 vehículos diarios en la mayoría de días a pleno funcionamiento, cualquier error no detectado en la etapa inicial puede multiplicar los costos asociados a reparaciones y afectar negativamente a la eficiencia operativa. Por ejemplo, el montaje incorrecto de un componente interior puede requerir el posterior desmontaje parcial del vehículo, un proceso que no solo incrementa los costes operativos, sino que también retrasa el flujo de producción. Además, los errores no detectados podrían derivar en problemas de calidad que afecten a la percepción del cliente y generen devoluciones o reclamaciones.

Ante estas circunstancias, resulta evidente la necesidad de un mecanismo más eficiente y confiable para la verificación de conformidad de los componentes en las etapas tempranas del proceso productivo, reduciendo así los costes de reparación y mejorando la calidad del producto final.

1.5. JUSTIFICACIÓN

La creciente complejidad de los procesos de producción en la industria automovilística, junto con la alta personalización de los vehículos según las especificaciones del cliente, ha incrementado significativamente la necesidad de controles de calidad más precisos y automatizados. En este contexto, la implementación de un sistema embebido de visión artificial se justifica como una herramienta clave para optimizar la producción y minimizar los costos asociados a errores de ensamblaje.

Este proyecto propone una solución que permite identificar errores de manera temprana, impactando directamente en la eficiencia operativa y reduciendo costes de reparación en más del 50%. Además, el proyecto responde a las tendencias actuales de la industria 4.0, donde la integración de tecnologías avanzadas como la visión artificial y la inteligencia artificial no solo optimizan procesos, sino que también establecen estándares más altos de calidad y adaptabilidad en la fabricación.

Por último, hay que destacar el diseño modular y escalable del sistema propuesto como punto fuerte, que asegura su utilidad más allá del problema puntual abordado en este trabajo, ya que puede adaptarse a otras líneas de producción y procesos dentro de la planta, fomentando la innovación tecnológica y la sostenibilidad operativa en Stellantis.

1.6. OBJETIVOS

1. Desarrollar un sistema embebido de visión artificial para el control de conformidad industrial en la línea de producción de Stellantis, con el objetivo de detectar errores de ensamblaje de piezas en tiempo real
2. Implementar un sistema eficiente y de bajo costo, utilizando cámaras industriales y plataformas como la Jetson Xavier NX, para realizar inspecciones automáticas en cada vehículo
3. Incorporar la red neuronal Yolov5 para identificar y clasificar los componentes montados en el vehículo, asegurando la precisión en la detección de errores
4. Mejorar la calidad del ensamblaje al permitir la detección temprana de fallos y su corrección antes de que se conviertan en costosos problemas

5. Facilitar la integración del sistema en otras líneas de producción y etapas de ensamblaje
6. Asegurar la fiabilidad y disponibilidad del sistema, con tiempos de respuesta rápidos y sin interrupciones en la producción

1.7. REQUISITOS FUNCIONALES

1. El sistema debe ser capaz de tomar una imagen a la hoja de montaje de cada vehículo.
2. El sistema debe ser capaz de leer de la hoja de montaje el código de identificación del vehículo y las piezas específicas que se deben de montar en él.
3. El sistema debe ser capaz de tomar fotos de las partes del vehículo.
4. El sistema debe ser capaz de identificar que piezas lleva montado un vehículo a partir de la imagen de este.
5. El sistema debe ser capaz de detectar si un vehículo lleva montadas las piezas que le corresponden.
6. El sistema debe ser capaz de avisar al operario responsable si un vehículo tiene montadas piezas que no le corresponden.
7. El sistema debe contar con una interfaz que muestre al supervisor del puesto y a los operarios los resultados de los últimos 10 vehículos ensamblados.
8. El sistema debe almacenar las diferentes fotos de las partes del vehículo durante al menos, 15 días, para que el supervisor pueda consultarlas.
9. El sistema debe ser capaz de tener una nueva comprobación funcional en un plazo inferior a 15 días.

1.8. REQUISITOS NO FUNCIONALES

1. El sistema debe ser *offline*.
2. El sistema debe ser eficiente energéticamente.
3. La interfaz con la que interactúan los operarios y supervisores de la línea debe ser intuitiva y fácil de usar.
4. El sistema debe estar disponible todo el tiempo.
5. El sistema no debe interferir en el trabajo de los operarios.

6. El sistema debe tener al menos un 90% de precisión.
7. El sistema debe ser modular, permitiendo la incorporación de nuevos sensores o cámaras sin modificaciones importantes.
8. El sistema debe ser fácilmente integrable con otros sistemas de gestión de la planta y bases de datos.
9. El sistema debe ser escalable, permitiendo su expansión y adaptación a nuevas líneas de producción sin problemas.
10. El sistema debe ser seguro, con protecciones ante accesos no autorizados y cumplimiento de normativas de protección de datos
11. El sistema debe ser de fácil mantenimiento, permitiendo actualizaciones y reparaciones rápidas sin interrumpir la producción.

2. ESTADO DEL ARTE

En esta sección se describen de forma introductoria los principales sistemas de detección de defectos que utilizan tecnologías de visión artificial. Se describen tanto las soluciones actualmente instaladas en diferentes áreas de la planta de Stellantis Zaragoza como ejemplos en otras industrias.

En el ámbito de la detección de no conformidades mediante visión artificial, es posible distinguir entre dos enfoques principales: los sistemas comerciales, desarrollados por proveedores especializados y los sistemas desarrollados internamente por la propia empresa.

Los sistemas comerciales suelen implicar una mayor inversión inicial, pero tienen la ventaja de una implantación más rápida, ya que el proveedor se encarga de todo el proceso: desde la elección del hardware hasta la programación del software y el entrenamiento del modelo de visión artificial. Sin embargo, este tipo de soluciones suelen presentar una dependencia tecnológica significativa del proveedor, lo que limita su flexibilidad. Por ejemplo, puede ser complejo adaptar estos sistemas ante cambios en el modelo del vehículo, la incorporación de nuevas referencias o variantes del color.

Por otro lado, las instalaciones internas o *in house* son más económicas, pero requieren que el personal de la empresa adquiera el conocimiento necesario para programar, instalar y entrenar la red neuronal, además de seleccionar y comprar el hardware. Estas instalaciones son más versátiles en cuanto a actualizaciones. Además, una vez que se tiene el programa base bien definido y entrenado el proceso de añadir más controles es sencillo, también se puede replicar y modularizar para otras instalaciones de forma sencilla. En otras palabras, las horas invertidas por el personal para adquirir los conocimientos para programar estas instalaciones serán útiles para las siguientes instalaciones a replicar, por lo que el coste de estas horas se dividirá entre las distintas instalaciones de la planta o incluso de otras plantas del grupo que deseen instalar este sistema.

En la planta de montaje final de Stellantis Zaragoza encontramos diversas instalaciones de detección de defectos por visión artificial. Entre ellas, se encuentran las del módulo de montaje de puertas (Figuras 2.1 y 2.2), detección de bajos o *underbody* (Figura 2.2) y CVT o *conformity check* (Figura 2.3) en las líneas 1 y 2. Estas tres instalaciones detectan defectos relacionados con la no conformidad de las piezas montadas, verificando si están montadas y si corresponden o no.

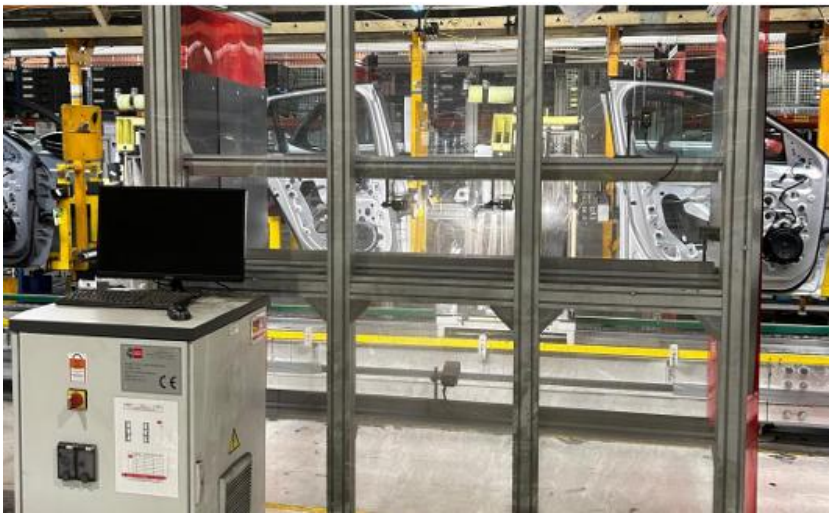


Figura 2. 1: Estación de visión artificial para detección de defectos en módulo de puertas

Fuente: Elaboración Propia

Figura 2. 2: Detección de fallos realizada por la estación de visión artificial del módulo de puertas

Fuente: Elaboración Propia

También existen otros tipos de instalaciones que detectan distintos defectos. En la planta de Stellantis Zaragoza existen sistemas de detección de defectos superficiales o mutilaciones en el estado de la pintura y la chapa del vehículo. Además, existen sistemas que detectan los denominados *gap and flush*, que comprueban la uniformidad de las distancias entre paneles (*gap*) , por ejemplo, la distancia entre aleta y puerta; y la alineación en el mismo plano (*flush*), por ejemplo, que el capo y las aletas laterales estén al mismo nivel y no sobresalgan. Este tipo de instalaciones son mucho más complejas, requieren de material más caro y de un software más específico.

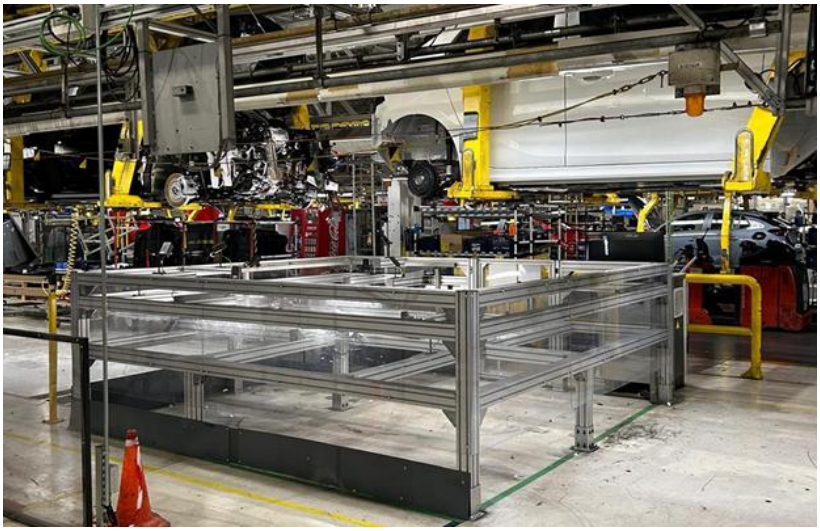


Figura 2. 3: Estación de visión artificial para detección de defectos en módulo de underbody

Fuente: Elaboración Propia

(VITRONIC, 2023) Además de la industria del automóvil, podemos encontrar estos sistemas en otros sectores. En la industria alimentaria y farmacéutica, ayudan a garantizar la seguridad alimentaria. En la industria textil, detectan problemas en tejidos y patrones difíciles de percibir con visión humana.

En general estos sistemas de detección mediante visión artificial requieren una fuerte inversión y son rentables solamente en condiciones muy específicas.

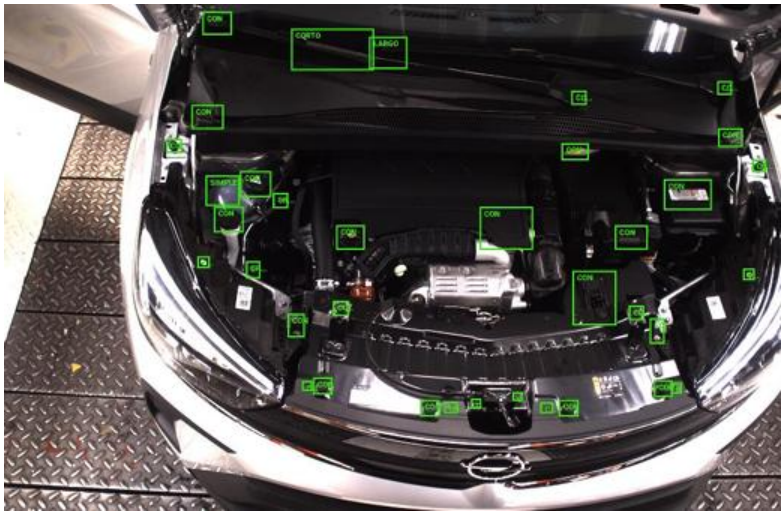


Figura 2. 4: Detección de fallos realizada por la estación de visión artificial de conformity check

Fuente: Elaboración Propia

Como ingeniero, y teniendo en cuenta los requisitos específicos de una estación de visión artificial orientada a la detección de no conformidades en entorno industrial, es fundamental valorar de forma critica las decisiones relativas al hardware de la instalación. Si bien en el presente proyecto se ha trabajado con ciertos equipos ya disponibles en planta – como las cámaras FLIR – una instalación ideal debería contemplar desde su diseño un conjunto de elementos mas robustos y escalables.

Por ejemplo, la incorporación de concentradores industriales tipo IFM o SIG300, que permiten una integración mas estructurada y segura de sensores digitales y analógicos, así como una comunicación eficiente con Profinet o EtherCAT. Del mismo modo, el uso de tarjetas de red industriales con mayor ancho de banda y capacidad de aislamiento eléctrico permite gestionar múltiples flujos de imagen en tiempo real sin riesgo de cuellos de botella de transferencia de datos.

En cuanto al procesamiento, la elección de plataformas de computación mas potentes como NVIDIA Jetson AGX Orin o estaciones de trabajo industriales con GPU dedicadas garantizaría tiempos de inferencia más rápidos y permitiría escalar el sistema para inspeccionar más elementos por vehículo o utilizar modelos de redes neuronales más complejos. Además, se valoraría muy positivamente una arquitectura más modular basada en PLC industriales o edge devices especializados, que faciliten tanto la depuración como la replicabilidad del sistema.

3. MARCO TEÓRICO

3.1. VISIÓN ARTIFICIAL

Según (López Pérez, 2021) la visión artificial es un campo en desarrollo dentro de la inteligencia artificial, el cual se ocupa de dotar a los ordenadores de la capacidad de extraer información de imágenes o videos. La manera de proceder de los algoritmos de este campo busca asimilarse a la visión humana.

La detección de objetos, el fundamento principal de la visión artificial consiste en la identificación de estas personas, animales o cosas, así como su posición dentro de una imagen o video. Hay distintos métodos aplicables para conseguir este objetivo, siendo uno de ellos el entrenamiento previo de una red neuronal para detectar dichos objetos. Existen dos aproximaciones a este problema, la segmentación semántica y la segmentación de instancias. La primera asocia cada píxel a la imagen de una clase (objeto), mientras que la segunda identifica distintos elementos de la misma clase por separado.

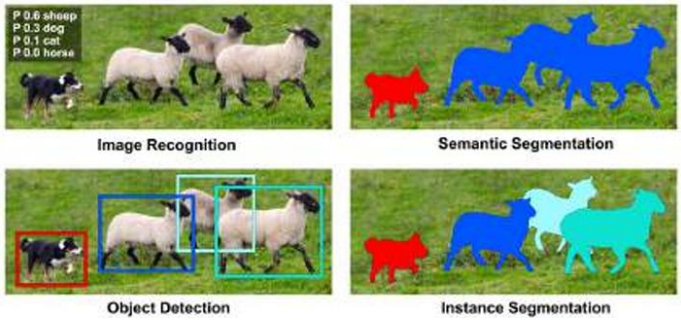


Figura 3. 1: Ejemplo de segmentación semántica y de instancias para una misma imagen

Fuente: (López Pérez, 2021)

3.2. INTELIGENCIA ARTIFICIAL

Inteligencia artificial, machine learning y deep learning son tres conceptos que a menudo son confundidos o aplicados de manera incorrecta. Antes de continuar es necesario diferenciarlos y explicar qué abarca cada uno de ellos.

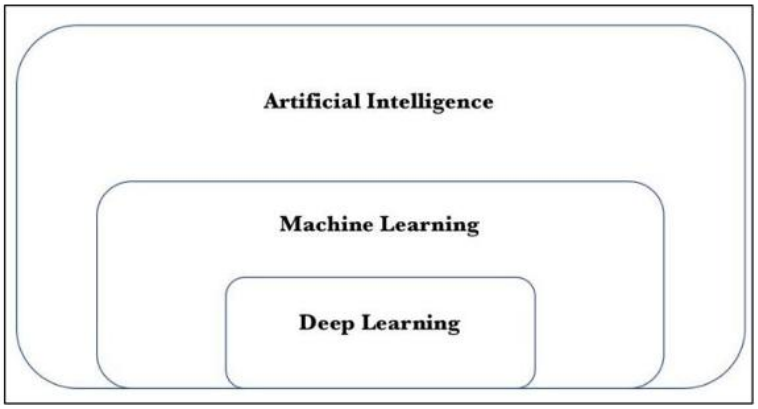


Figura 3. 2: Inteligencia Artificial engloba el Machine Learning, y éste al Deep Learning

Fuente: Martin Romero (2022)

En este apartado, se dará una pequeña introducción para señalar las diferencias entre la inteligencia artificial y el machine learning. El deep learning está íntimamente relacionado con la forma de agruparse de las redes neuronales artificiales y su estructura en forma de capas, así que se tratará más en profundidad en el siguiente apartado.

3.3. FUNDAMENTOS DEL APRENDIZAJE AUTOMÁTICO (ML)

Machine Learning (ML) se puede entender como la capacidad de una máquina de encontrar un patrón o función adecuada para una aplicación determinada.

En (Murphy, 2012) define Machine Learning como algoritmo que puede detectar de forma automática patrones en el conjunto de datos de entrada y utilizar dichos patrones en nuevos datos para predecir nuevos resultados o tomar algún tipo de decisión según la aplicación.

En (Zhou, 2021) define el aprendizaje automático como una disciplina que se compromete a estudiar cómo usar los algoritmos a través de medios computacionales para mejorar el rendimiento del propio sistema.

Existen muchos tipos de ML, aunque los más utilizados en los modelos actuales son los que se detallarán a continuación. En la Figura 3.3 se puede observar un diagrama resumen de estos tipos de aprendizaje, sus algoritmos y sus principales funciones.

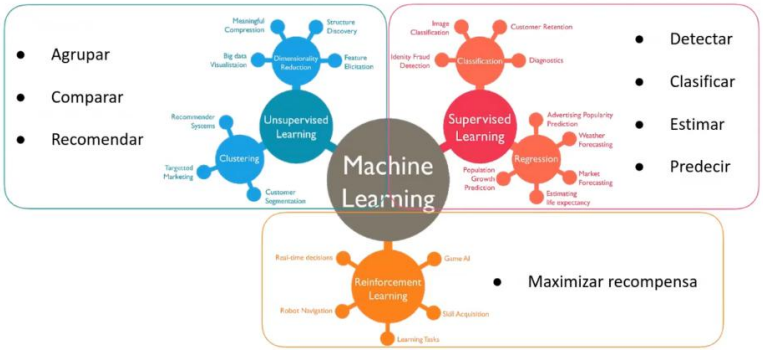


Figura 3. 3: Los tres grandes grupos de machine learning

Fuente: Navarro(2024)

3.3.1. Aprendizaje supervisado

(Murphy, 2012) En el ML predictivo o supervisado, el objetivo es determinar la relación entre los datos de entrada (x) con los datos de salida (y), dado un conjunto de datos (x,y) etiquetados. Se conocen tanto las variables predictoras, como las variables objetivas.

$$D = \{(x_i, y_i)\}_{i=1}^{N} \qquad \text{ecuación 1}$$

Donde D es el conjunto de datos de entrenamiento y N es el número de individuos, o el número de datos.

En la configuración más simple, cada entrada de entrenamiento x_i es un vector de número de D-dimensional, que representaría la característica, atributo o covariable. La variable de salida o respuestas, al igual que las variables de entrada, puede ser cualquier estructura compleja, sin embargo, la mayoría de las veces las variables de salida son escalares. En el caso de que la respuesta que se obtiene es una variable nominal, el problema es de clasificación o patrones de reconocimiento; en el caso de que sea un escalar, se trata de un problema de regresión.

3.3.2. Aprendizaje no supervisado

También conocido como aprendizaje descriptivo, donde el objetivo es encontrar patrones interesantes en los datos, a veces conocido como el "descubrimiento de conocimientos" (*Knowledge discovery*).

$$D = \{(x_i)\}_{i=1}^{N}$$ ecuación 2

3.3.3. Aprendizaje reforzado

(Martin Romero, 2022) Se presenta a la máquina un entorno desconocido y ella misma debe aprender qué acción debe realizar a través de la prueba y el error, habiendo un mecanismo que recompense o penalice las conductas que se acerquen o alejen a la solución deseada. Este método de aprendizaje se suele emplear en combinación con otros tipos de entrenamiento, ya que puede simular muchos escenarios del mundo real.

3.4. Redes Neuronales Artificiales (ANNs)

3.4.1. Inspiración biológica

(Durán Suárez, 2017) El modelo simplificado y abstracto de las redes neuronales artificiales surge de intentar imitar el comportamiento de las neuronas que se encuentran en el cerebro.

En esencia, el cerebro está formado por un numero ingente de neuronas (aproximadamente 10^{11}) conectadas entre sí creando la red neuronal. Las neuronas tienen tres partes principales: las dendritas, el cuerpo y el axón. Las dendritas son fibras nerviosas que transportan los impulsos eléctricos al cuerpo de la neurona. El cuerpo de la neurona se encarga de reconocer y trabajar con las señales que le llegan. Y, por último, el axón es una única fibra nerviosa que comunica el cuerpo de la neurona con las demás. El punto de contacto entre el axón y la dendrita de otra neurona se denomina sinapsis. La disposición de las neuronas y las fuerzas de cada sinapsis individualmente, determinada por reacciones químicas, establecen el funcionamiento del cerebro. Podemos observas esta estructura de neuronas en la Figura 3.4.

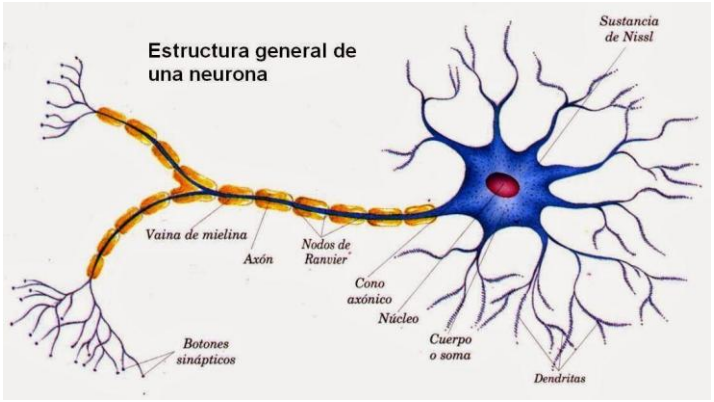


Figura 3. 4: Estructura general de una neurona biológica

Fuente: Durán Suárez (2017)

Cuando se habla de Red Neuronal Artificial (por sus siglas en ingles ANNs) en ML se refiere al modelo computacional inspirado en la estructura y el funcionamiento de un cerebro humano. La RN está formada por "neuronas", que funcionan como células nerviosas del cerebro, que se encarga de procesar y transmitir información en 0 y 1.

(Duan, 2018) En 1958, el informático Frank Rossenblatt propuso la primera regla de aprendizaje del perceptrón basada en el modelo M-P (McCulloch-Pitts neuron, MCP). Este perceptrón constituye la unidad básica de una red neuronal y lo explicaremos más en detalle para entender la base del funcionamiento de estas.

3.4.2. Modelos de correlación lineal

Para entender el funcionamiento del perceptrón simple, la unidad más básica en una red neuronal y su capacidad para resolver problemas de clasificación primero es necesario comprender los fundamentos matemáticos que subyacen en su estructura. El perceptrón, como muchas otras técnicas en ML, se basa en el concepto de modelos lineales.

Estos modelos constituyen la base matemática para realizar predicciones al combinar variables de entrada de manera ponderada, lo cual es clave en las operaciones internas de una red neuronal.

(Goodfellow et al., n.d.) Un modelo básico busca capturar la relación entre un conjunto de variables de entrada (x) y una salida predicha (y), mediante una fórmula de combinación lineal descrita a continuación:

$$y = \sum_{i=0}^{n} w[i] * x[i] + b \qquad \textit{ecuación 3}$$

Donde:

- y: la salida estimada o valor predicho
- x[i]: las variables de entrada (también llamadas características o *inputs*) para i=1,…,n
- w[i]: los pesos que ponderan cada entrada según su importancia en el modelo
- b: el sesgo (bias), que ajusta el valor de salida independientemente de las entradas
- n: el número de características o variables de entrada

El modelo lineal pondera cada entrada (x[i]) según su peso (w[i]) y suma un término adicional (b). Esto puede interpretarse como una forma de asignar importancia o influencia relativa a cada característica en la predicción del resultado.

3.4.3. El perceptrón simple

(De Nova Guerrero, 2021) Una forma de comprender el funcionamiento de una sola neurona en una red neuronal es a partir del perceptrón simple, que representa una implementación inicial de los modelos lineales en ANNs. Un perceptrón es un algoritmo basado en el funcionamiento del cerebro humano, teniendo como objetivo aprender y tomar decisiones por sí mismo.

Como se ha comentado, el perceptrón es la unidad básica de una red neuronal y uno de los modelos más básicos que podemos encontrar y consta de cuatro partes: unos datos de entrada representados como x_i, unos pesos asociados a cada entrada w_i, un sesgo o bias b, y una función de activación f(x). El funcionamiento de este modelo, representado en la Figura 3.5 consiste en multiplicar cada dato de entrada por su peso asociado y sumar todos los términos junto con el sesgo, dando lugar a un valor que se pasa a través de la función de activación para obtener la salida del perceptrón *o*, como se expresa a continuación.

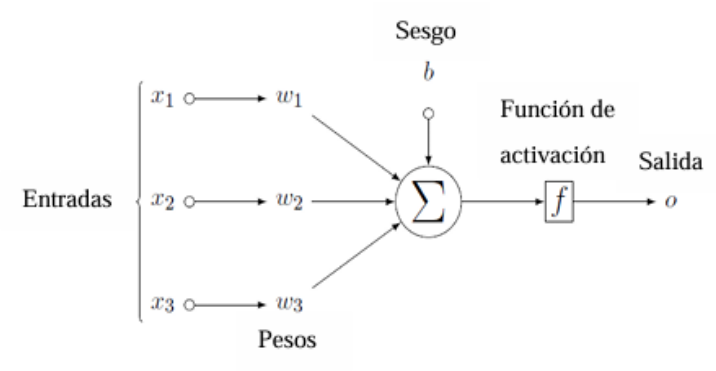


Figura 3. 5: Esquema del funcionamiento de un perceptrón

Fuente: De Nova Guerrero (2021)

Según (Goodfellow et al., n.d.) el perceptrón simple, además de realizar una combinación lineal de las entradas y sus respectivos pesos, requiere de un mecanismo que traduzca ese resultado en una salida interpretable. Este mecanismo es proporcionado por la función de activación (f(z)), que se aplica al resultado intermedio (z) calculado como sigue a continuación:

$$z = b + \sum_{\forall i} w_i x_i \qquad \textit{ecuación 4}$$

El objetivo de la función de activación es determinar la salida del perceptrón (*o*), basada en si el valor de *z* supera cierto umbral. En el caso del perceptrón simple, esta función de activación suele ser la función escalón, que genera una salida binaria 0 o 1 dependiendo del signo del valor de *z*:

$$o = f(z) = \begin{cases} 1 \; si \; z \geq 0 \\ 0 \; si \; z < 0 \end{cases} \qquad \textit{ecuación 5}$$

La función escalón permite al perceptrón tomar decisiones binarias, es decir, clasificar las entradas en dos categorías separadas por una frontera de decisión lineal.

3.4.4. El perceptrón multicapa MLP

Para desempeñar tareas más complejas se hace uso de lo que se conoce como el perceptrón multicapa o MLP (por sus siglas en inglés *Multi Layer Perceptron*).

Según (Duan, 2018) en el modelo MLP, el algoritmo agrega capas ocultas (*Hidden Layers*) en el proceso, las variables de entrada se combinan mediante pesos y se aplica una función de activación (h) sobre las propias combinaciones de variables. Finalmente, usa los resultados calculados de la capa oculta para generar el resultado final, como se puede apreciar en la Figura 3.6.

(De Nova Guerrero, 2021) En cada capa, cada neurona funciona como un perceptrón simple, siendo su entrada el resultado de las neuronas de la capa anterior. Cuando todas o la mayoría de las neuronas de una capa están conectadas a las neuronas de capas de capas anteriores o posteriores, se dice que es una capa completamente conectada o fully connected. En la Figura 3.2-5 se muestra un ejemplo de este tipo de red con 5 capas.

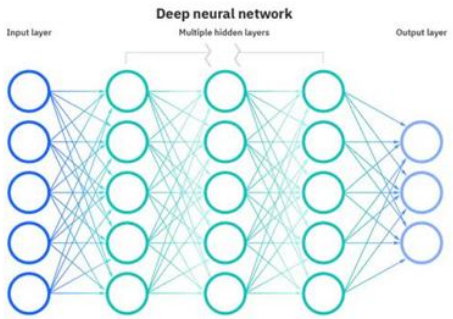


Figura 3. 6: Ejemplo de una MLP de 5 capas

Fuente: De Nova Guerrero (2021)

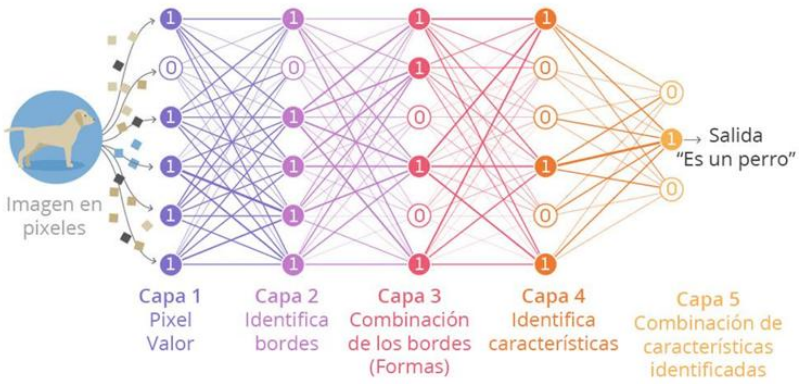


Figura 3. 7: Esquema funcional de las capas de una red neuronal para la identificación de un perro

Fuente: Raimon Blanes (2022)

3.4.4.1. Funciones de activación

Según (Martin Romero, 2022) para propagar la salida de una neurona hacia capas posteriores se hace uso de las denominadas funciones de activación. Las funciones de activación sirven para introducir no-linealidades en el modelo de la red. La Figura 3.8 muestra algunas de las funciones de activación, aunque se van a explicar en las siguientes páginas con un poco más de detalle las más utilizadas según (De Nova Guerrero, 2021): *ReLU, sigmoid* y *softmax*.

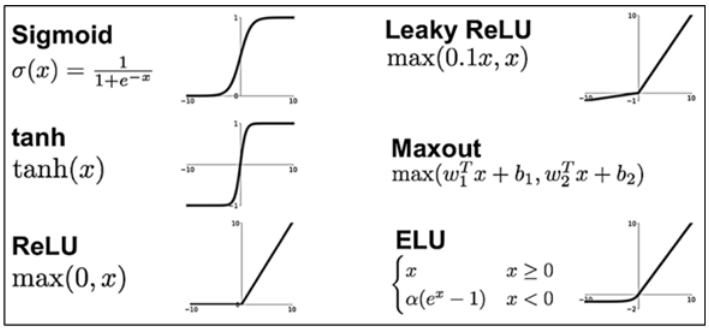


Figura 3. 8: Funciones de activación más comunes en machine learning

Fuente: Martin Romero (2022)

La función ReLU (*Rectified Linear Units*), representada en la figura 3.9-a, consiste en una función en la que la salida toma valor cero si la entrada es menor que cero. En caso contrario, la salida toma el valor de la entrada. Esta función es la más utilizada en capas intermedias de la red neuronal y se define como se muestra a continuación, siendo z el valor de la entrada a la función de activación.

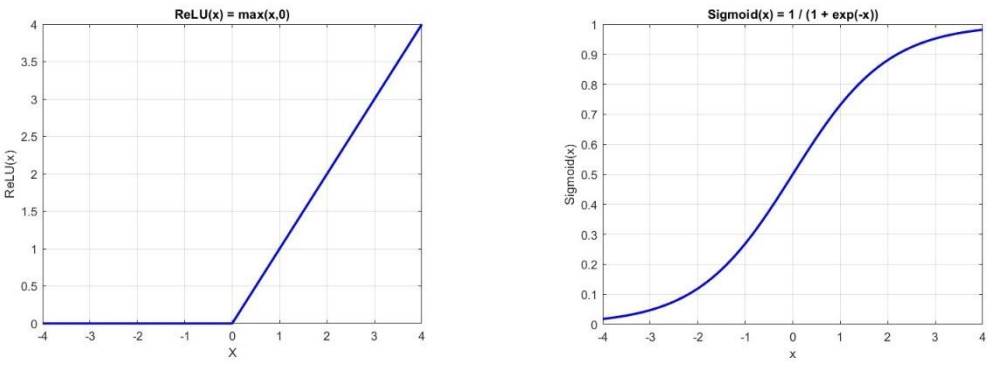
$$ReLU(z) = \max(0, z) \qquad \textit{ecuación 6}$$

La función sigmoid, representada en la figura 3.9-b, consiste en una función cuya salida está comprendida en el intervalo [0,1], por lo que resulta especialmente útil en la capa final de una red neuronal para tareas de clasificación multi-etiqueta en las que las clases son no excluyentes, por lo que la suma de probabilidades de todas las clases no tiene por qué sumar 1. En estos modelos la salida de la red neuronal consiste en un vector de dimensión 1xn, siendo *n* el número de clases posibles a clasificar, por lo que, con esta función se puede obtener una probabilidad entre 0 y 1 para cada una de las posibles clases. La función se define como se muestra a continuación, de nuevo siendo *z* el valor de la entrada de la función de activación:

$$sigmoid(z) = \frac{1}{1 + e^{-z}} \qquad \textit{ecuación 7}$$

La función softmax es similar a la función sigmoid, ya que está comprendida en el intervalo [0,1], por lo que también es muy utilizada en labores de clasificación. Sin embargo, a diferencia de la función anterior, esta se utiliza en problemas de clasificación multiclase en los que las diferentes clases son mutuamente excluyentes. Es decir, la suma de las probabilidades de todas las clases da como resultado 1. El cálculo de la función sigue a continuación, donde z_j es el valor de la entrada de la clase para la que se calcula el softmax y K el número total de clases.

$$softmax(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}} \; para \; j = 1, \ldots, K \qquad \textit{ecuación 8}$$



(a) *(b)*

Figura 3. 9: (a) Función ReLU, (b) Función sigmoid

Fuente: Elaboración propia con MATLAB

En (Martin Romero, 2022) se nos detalla que no todas las capas de la red tienen que usar la misma función de activación, existen arquitecturas en las que es conveniente que la capa de entrada o salida posea una función de activación no-lineal (sigmoide, tangente hiperbólica, etc.) frente a una función lineal a tramos para las capas ocultas (ReLU).

Dependiendo del tipo de problema que se aborde, algunas funciones de activación serán más apropiadas que otras. Por ejemplo, la función sigmoide puede ser muy útil para predicción de probabilidades; mientras que la función ReLU es muy usada en las capas ocultas de las redes convolucionales, para el tratamiento de imágenes, como es nuestro caso.

3.4.4.2. Algoritmo: retropropagacion (*Backpropagation*)

Para que una ANNs sea capaz de aprender de los datos y mejorar su predicciones, es necesario un mecanismo que permita ajustar los pesos de las conexiones entre las neuronas de manera eficiente.

Según (De Nova Guerrero, 2021) el proceso de aprendizaje de las redes MLP consiste en ajustar los pesos y sesgos de cada perceptrón, o neurona, con el objetivo de minimizar una función de coste. El proceso de minimización de esta función y la actualización de los parámetros de la red se realiza mediante la propagación de errores o backpropagation. El propósito de la retropropagacion es minimizar el error cometido por el modelo, es decir, reducir la diferencia entre la salida generada por la red y el valor deseado. Esto se logra ajustando los pesos mediante un proceso iterativo que aprovecha el cálculo del gradiente de la función de perdida (*Loss Function*). Este gradiente indica como deben modificarse esos pesos para reducir el error.

En (Duan, 2018) se explica que el algoritmo de retropropagacion funciona propagando el error hacia atrás a través de la red, comenzando por la capa de salida y retrocediendo hasta las capas ocultas. Para cada capa, se calcula la contribución relativa de cada neurona a la función de coste, y se utilizan esas contribuciones para ajustar los pesos de las conexiones.

Este proceso se basa en la regla de la cadena de las derivadas, que permite obtener la estimación del gradiente del vector peso (w). En el contexto de la retropropagacion, la regla de la cadena se utiliza para calcular la contribución de cada nodo a la función de coste, en función de sus entradas y de los pesos de las conexiones que la conectan con las capas siguientes. De nuevo, el objetivo de todo esto es estimar el vector de pesos para optimizar la función de pérdida o error.

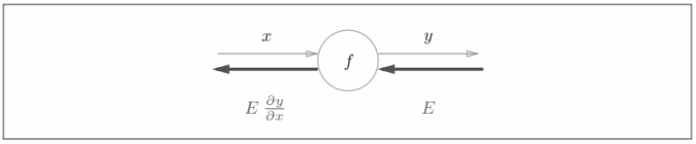


Figura 3. 10: BackPropagation

Fuente: Duan (2018)

Como se observa en la Figura 3.2-9, la secuencia de cálculo para la propagación hacia atrás es multiplicar la señal E por la derivada local del nodo y luego pasar el resultado al siguiente nodo.

3.5. Fundamentos Del Aprendizaje Profundo (DL)

Se denomina aprendizaje profundo a las redes neuronales multicapa o MLP. Según (Zhou, 2021), cuanto mayor es el número de parámetros de un modelo, mayor es su complejidad y capacidad para aprender, lo que le permite realizar tareas más complejas. Sin embargo, este aumento de complejidad también puede hacer que el proceso de entrenamiento sea más ineficiente y que el modelo se sobreajuste (*overfitting*), es decir, aprenda demasiado bien los datos de entrenamiento, pero no generalice correctamente a nuevos datos.

Por lo que, se puede entender el Deep Learning como un conjunto de algoritmos de aprendizaje automático (ML) que mediante el mecanismo de aplicar capas y usar la salida de la capa anterior procesada como entrada, conecta la entrada inicial con el objetivo de salida; en otras palabras, transforma los atributos de bajo nivel en alto nivel para resolver problemas complejos de un modo sencillo.

3.6. Redes Neuronales Convolucionales (CNNs)

En (Martin Romero, 2022) se nos relata que las redes neuronales convolucionales (CNNs, de sus siglas en inglés *Convolutional Neuronal Networks*) son un tipo concreto de red neuronal en deep learning que ya estaba presente en la década de los 90, pero que han ganado protagonismo en el campo de visión por computación en los últimos años debido al buen desempeño que poseen para tareas de reconocimiento de imágenes.

Siguen el mismo funcionamiento que las redes neuronales ya explicadas, utilizando pesos, sesgos y funciones de activación. Una de las diferencias es que la entrada de una CNN es o bien una imagen o bien una representación matricial de datos de entrada. La forma en que una CNN es capaz de identificar formas como caras, animales u objetos, es incrementando el nivel de abstracción que la red es capaz de realizar en cada capa, pasando por patrones y características más simples a más complejos.

La arquitectura básica de una CNN según (Duan, 2018) está estructurada en 3 capas:

- Capa de convolución
- Capa de agrupación (Pooling)
- Capa de salida (Fully Connected)

La capa de convolución se encarga de extraer características de la imagen, mientras que la capa de agrupación se encarga de reducir la dimensionalidad de la imagen. Tras los procesamientos en las capas anteriores, obtenemos una matriz bidimensional reducida, que conserva las

características de la imagen, y permite que la capa de salida pueda computar los valores producidos por las funciones de activación y clasificarlos.

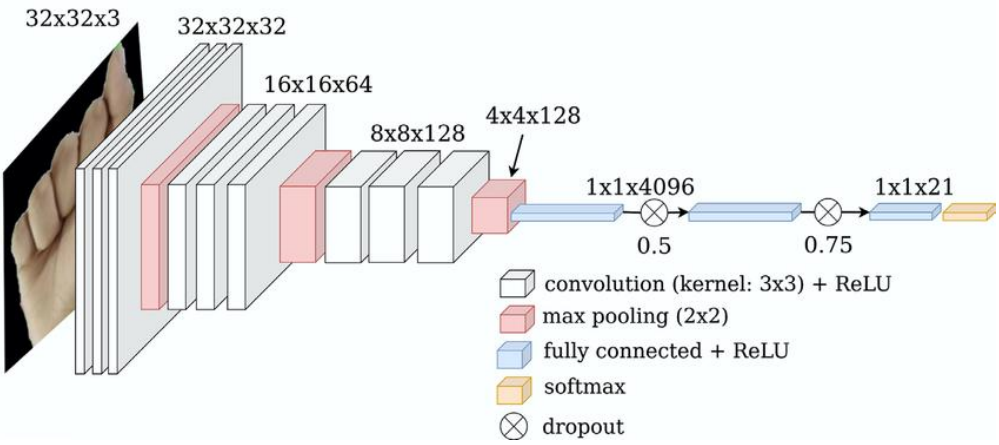


Figura 3. 11: Arquitectura de CNN

Fuente: Suat Rojas et al. (2021)

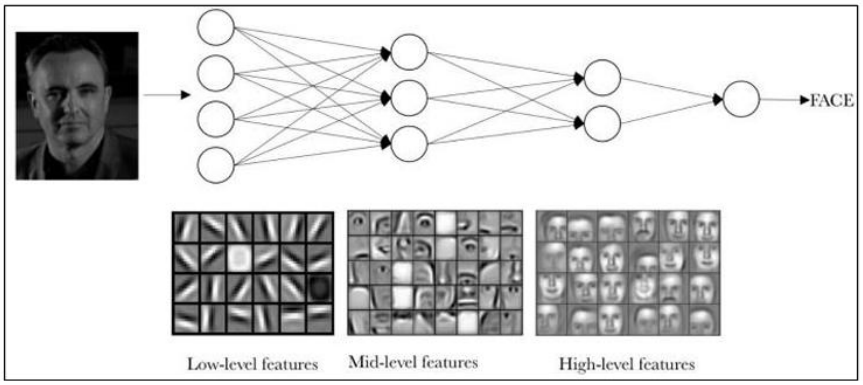


Figura 3. 12: Visualización intuitiva de cómo funciona una red neuronal convolucional

Fuente: Martin Romero (2022)

De nuevo (Martin Romero, 2022) nos dice que, sin embargo, otra diferencia fundamental respecto a las anteriores es que las redes convolucionales no pueden implementarse de la misma forma que una red neuronal artificial de capas completamente conectadas. Para una entrada de una imagen de tamaño reducido de, por ejemplo, 150x150 píxeles, x3 canales RGB, se da que las neuronas de la capa de entrada deberían trabajar con 67.500 parámetros, una cifra demasiado elevada que conllevaría un inmenso coste computacional. Para abordar esta problemática se introducen dos operaciones: convolución y pooling.

3.6.1. Convolución

La principal diferencia entre una capa completamente conectada y una capa convolucional (una capa a la que se aplica la operación de convolución), es que la capa completamente conectada aprende patrones globales para todo su input, mientras que la capa convolucional aprende patrones locales en pequeñas ventanas del input total.

La capa convolucional es capaz de detectar características como bordes, líneas, gradientes de color, entre otros; pudiendo extrapolar la detección de esas características a otros puntos cualquiera de la imagen. De esta forma, las primeras capas aprenden patrones simples como bordes, mientras que capas posteriores aprenden patrones más complejos compuestos por patrones y características aprendidas en capas anteriores; de esta forma, la red convolucional es capaz de aprender a reconocer conceptos de mayor complejidad y abstracción.

Por ejemplo: una imagen de dimensiones 28x28 en escala de grises se introduce como input a una capa. La siguiente capa tras la capa de entrada estará conectada convolucionalmente, esto quiere decir, las neuronas de esta capa oculta (intermedia) tendrán como entrada una ventana de 5x5 pixeles de la capa anterior. Esta ventana de 5x5 va deslizándose de izquierda a derecha y de arriba hacia abajo recorriendo la imagen completa. Por cada posición de la ventana, hay una neurona de la siguiente capa oculta que se encargará de procesar la información que le llegue de ese subconjunto de 5x5 pixeles.

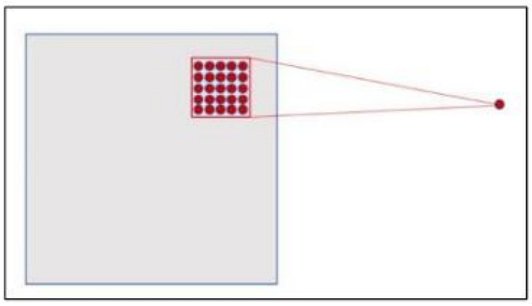


Figura 3. 13: Proceso de convolución

Fuente: Martin Romero (2022)

Usando una ventana de 5x5 que se mueve sólo una posición en cada desplazamiento, en una capa de 28x28 píxeles, se obtiene una capa oculta de 24x24 neuronas. El tamaño de la ventana o el número de posiciones que se mueven en cada desplazamiento (llamado *stride*) son parámetros que se pueden configurar, dando lugar a capas convolucionales más reducidas incluso.

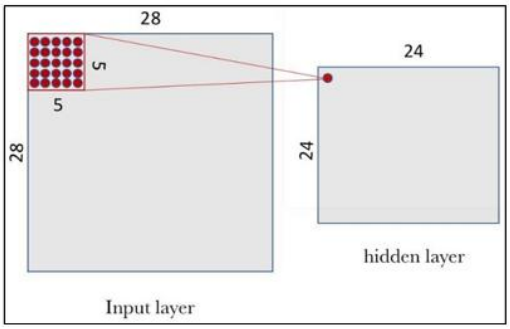


Figura 3. 14: Resultado del proceso de convolución

Fuente: Martin Romero (2022)

3.6.2. Pooling

Aparte de las capas convolucionales, las CNNs acompañan esas capas con capas de pooling. Las capas de pooling, descrito de forma muy simple, simplifican y condensan la información que se les introduce como entrada. Esto permite reducir el número de neuronas y parámetros necesarios en capas posteriores, simplificando el modelo de la red neuronal.

Para realizar la operación de pooling a una capa se define el tamaño de una ventana o kernel, y se divide dicha capa en porciones del tamaño de la ventana. Cada porción se puede condensar de diversas maneras, una de las más comunes se conoce como Max Pooling, que toma la máxima intensidad del píxel o valor neuronal en dicha porción.

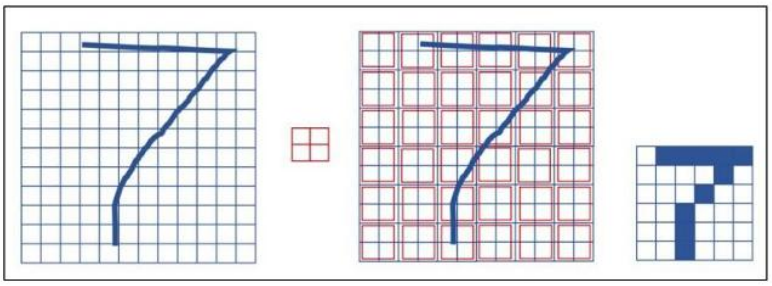


Figura 3. 15: Ejemplo de Max Pooling para un kernel 2x2

Fuente: Martin Romero (2022)

También existe el Min Pooling, análogo a lo anterior, pero tomando la mínima intensidad, o el Average Pooling, que hace una media ponderada de toda las intensidades de la porción.

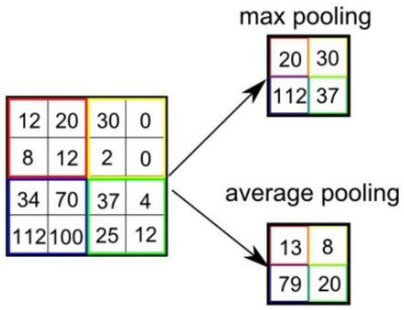


Figura 3. 16: Ejemplo de una capa Max Pooling y una capa Average Pooling

Fuente: De Nova Guerrero (2021)

Cuanto mayor sea la ventana o kernel empleado, más pequeña será la capa resultante, pudiéndose perder información relevante en el proceso. Sin embargo, si se escoge un tamaño adecuado, se puede reducir el número de neuronas y parámetros del modelo de la red neuronal sin que esto afecte a su desempeño, aliviando la carga computacional.

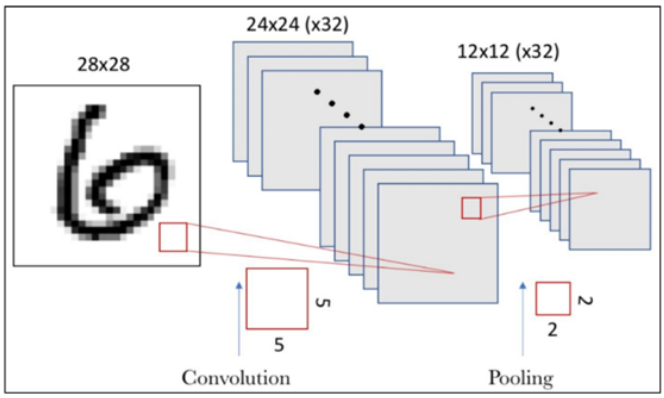


Figura 3. 17: Disminución progresiva del tamaño de las capas como resultado de aplicar procesos de convolución y pooling

Fuente: Martin Romero (2022)

Para el caso del ejemplo del apartado anterior; partiendo de una imagen de 28x28 pixeles se pasa a unas capas convolucionales de 24x24 neuronas tras una convolución usando una ventana de tamaño 5x5; y, posteriormente, aplicando un kernel de 2x2 se pasa a unas capas de pooling de 12x12. Este proceso puede repetirse varias veces, intercalando capas convolucionales y capas de pooling. Por último, la capa convolucional o de pooling precede a una capa altamente conectada, con una función de activación que adecúa al problema que debe resolver el modelo de red.

El problema, según (López Perez, 2021) radica en que al realizar detección de objetos en imágenes es necesario tener información espacial de estos. Una red convolucional estándar tiene una salida de tamaño fijado, mientras que para nuestro problema esta salida es variable, debido a que el número de apariciones del objeto en cuestión en cuestión a detectar no está fijado. Una solución para este problema sería buscar los objetos en determinadas regiones de la imagen aplicando en cada una

la red CNN, pero esto sería computacionalmente costoso debido a la gran cantidad de estas. Por ello, resulta necesario para la detección de objetos algún método para determinar en qué subregiones conviene buscar dichos objetos (aplicar la red convolucional).

Esto lleva a la principal división de los algoritmos de detección, dando lugar a dos grupos diferenciados según la manera de abordar dicho problema: la detección en Dos Etapas (Two Stages), o en Una Etapa (One Stage).

Se comentará por encima la primera aproximación de Dos Etapas, y se explicará con más detalle el algoritmo de Una Etapa, pues en la realización de este proyecto se ha utilizado una red neuronal CNN basada en este algoritmo, YOLO.

3.7. Algoritmos De Detección De Objetos

3.7.1. Algoritmos de detección de objetos en Dos Etapas (Two Stage)

Para solucionar el problema del gran número de regiones se busca dividir la imagen en zonas susceptibles de presentar objetos a detectar. Para identificar dichas regiones existen distintas técnicas. Una de las más usadas es el método de búsqueda selectiva, el cual combina la búsqueda exhaustiva y la segmentación, seleccionando aproximadamente 2000 regiones con forma rectangular basándose en las propiedades similares de estas.

Entre los diferentes métodos que existen en este algoritmo y su evolución, los más importantes son:

- R-CNN
- Fast R-CNN
- Faster R-CNN
- Mask R-CNN

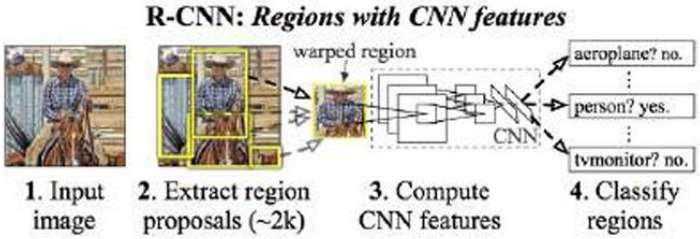


Figura 3. 18: Proceso de segmentación y convolución en el algoritmo R-CNN

Fuente: (López Pérez, 2021)

Sin embargo, este método requiere de un tiempo de computación alto al tener que realizar la convolución de las cerca de 2000 regiones propuestas.

3.7.2. Algoritmos de detección de Una Etapa (One Stage)

Los algoritmos de detección basados en dos etapas implican la necesidad de entrenar y optimizar dos componentes independientes, lo cual conlleva un mayor tiempo de procesado y dificulta el proceso de optimización, aunque por regla general los resultados muestran una precisión mayor. Una alternativa potencialmente más rápida es realizar directamente la detección de objetos sin la etapa previa de proposición de regiones.

Los principales ejemplos para este enfoque son: RetinaNet, que principalmente introduce las propiedades de Feature Piramidal Network (FPN) permitiendo así la detección con precisión a distintas escalas; y YOLO, el cual es el objeto de este proyecto y explicamos más en detalle en el siguiente apartado.

3.8. YOLO

(Silva Guzmán, 2020) YOLO (por sus siglas en inglés, *You Only Look Once*) es un sistema que utiliza una red neuronal convolucional para la detección de objetos en tiempo real. El sistema aplica una única red neuronal a la imagen completa, razón por la cual es muy rápido. Esta red divida la imagen en regiones y predice múltiples cajas delimitadores (*bounding box*), y la probabilidad de detección de las clases del entrenamiento para cada caja delimitadora. Por último, utiliza un método de supresión de no-máximos para eliminar múltiples detecciones del mismo objeto. Como resultado, el sistema imprime los objetos que detectó, su confidencia y el tiempo de ejecución.

En la Figura 3.19 se muestra una ejecución del sistema sobre una imagen de entrada.

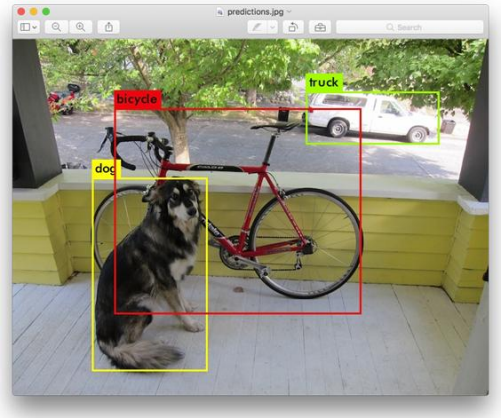


Figura 3. 19: Detección de objetos con YOLO

Fuente: (Silva Guzmán, 2020)

En (Pérez-Aguilar et al., 2024) se nos explica que en comparación a otros algoritmos como SSD o Faster R-CNN, esta red se emplea para la detección y reconocimiento en tiempo real de múltiples objetos. YOLO aborda la detección de objetos como un problema de regresión y proporciona probabilidades asociadas a cada clase detectada en una única ejecución del algoritmo.

Las ventajas claves de YOLO incluyen:

i. rapidez, posibilitando la detección en tiempo real

ii. precisión destacada debido a su baja tasa de errores

iii. notable capacidad de aprendizaje

YOLOv5, la versión de la red utilizada en este proyecto representa la quinta generación del algoritmo YOLO, siendo un detector en una sola etapa y una de las opciones más viables si se desea realizar la detección de objetos en tiempo real (FPS).

La Figura 3.20 muestra la arquitectura de YOLOv5, en la que se aprecian las capas personalizadas del algoritmo, compuestas principalmente por convoluciones y maxpooling; además en la figura 3.21 también podemos apreciar su pseudocódigo.

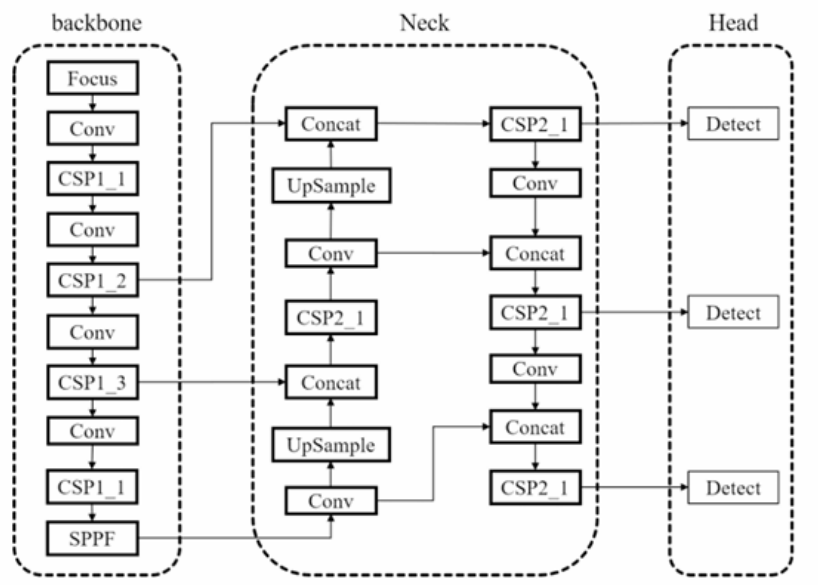


Figura 3. 20: Arquitectura de YOLOv5

Fuente: (Pérez-Aguilar et al., 2024)

Pseudocode - YOLOv5
Input: Number class; Class name; 1. Load images and pre-process data 2. Define the model architecture 3. - Backbone network (e.g., CSPNet, GAMMAttention, SPPFCSPC) 4. - Neck network (e.g., YOLOv5Neck) 5. - Detection head (e.g., YOLOv5Head) 6. - Loss function (e.g., Focal Loss) 7. Train the model 8. - Compute loss on mini-batch of images 9. - Compute gradients and update weights using optimizer (e.g., Adam) 10. Prediction 11. - Apply non-maximum suppression to remove overlapping predictions 12. - Output final detection results (bounding boxes, class probabilities, confidence scores)

Figura 3. 21: Pseudocódigo de YOLOv5

Fuente: (Pérez-Aguilar et al., 2024)

Matemáticamente (véase la ecuación tal), el modelo predice las coordenadas b_x, b_y, b_w, b_h de las cajas delimitadoras y la confianza *C*, de que hay un objeto en cada celda de una cuadrícula *SxS*, donde cada celda se encarga de detectar los objetos presentes en su interior.

Estas predicciones se obtienen aplicando una función sigmoide (σ) a la salida de una red neuronal, donde *W* representa los pesos de la red neuronal, $f(x)$ la entrada, y *b* el sesgo. El modelo realiza estas predicciones para *B* cajas, lo que le permite detectar múltiples objetos de una sola pasada, haciéndolo eficiente para la detección en tiempo real. YOLOv5 emplea principalmente tres técnicas: (i) utilización de bloques residuales, (ii) regresión de bounding box y (iii) intersección sobre unión (IOU). La combinación de estas tres técnicas genera resultados como los ilustrados en la figura 3.22.

$$B * \left(b_x, b_y, b_w, b_n, C\right) = \sigma(W * f(x) + b) \qquad \textit{ecuación 9}$$

Figura 3. 22: Combinación de técnicas en YOLOv5

(Pérez-Aguilar et al., 2024)

4. DESARROLLO

4.1. HARDWARE DE LA ESTACIÓN

En este apartado se describen los elementos de hardware utilizados en el sistema, explicando su funcionamiento y la razón de su elección. La mayoría de los componentes han sido reciclados de otros proyectos, por lo que su elección se basa principalmente en disponibilidad y coste, además de su capacidad para cumplir con los requisitos del sistema.

4.1.1. PC y periféricos

Para el procesamiento y gestión del sistema, el PC actúa como cerebro del sistema, procesando la información capturada por las cámaras y sensores. En este tipo de proyectos, su función principal es ejecutar los algoritmos de reconocimiento de imágenes, como los basados en redes neuronales, que analizan las imágenes en tiempo real para identificar defectos o patrones específicos, como conectores mal colocados.

Además, el PC gestiona la entrada de datos desde los sensores de presencia que disparan las cámaras en el momento adecuado, y coordina la clasificación de los resultados. También almacena las imágenes y los datos obtenidos en un disco duro de 2.5 TB, permitiendo así su posterior análisis o consulta.

Figura 4. 1: Ordenador y periféricos

Fuente: Elaboración Propia

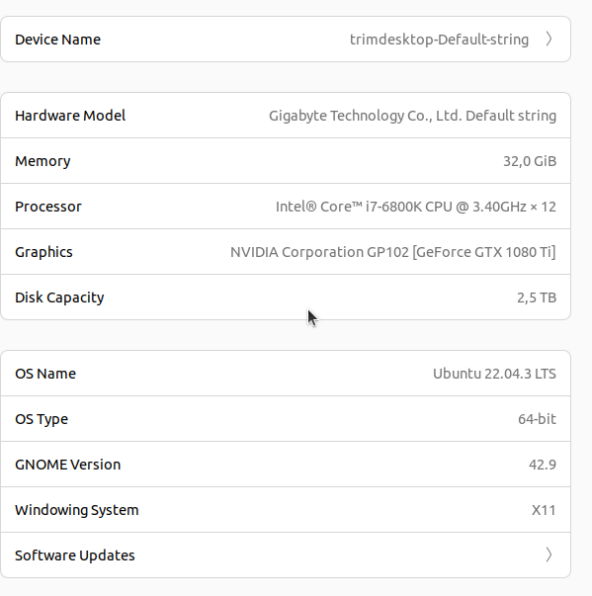


Figura 4. 2: Características del PC de la estación de visión

Fuente: Elaboración propia

La elección de este apartado para el proyecto ha sido un ordenador a base de piezas funcionales que ya habían sido utilizadas anteriormente en otras instalaciones de la empresa, concretamente en la denominada EagleEye de la instalación de pinturas, pero habían quedado obsoletas y las sustituyeron por otras mejores, por tanto, su coste ha sido cero.

4.1.2. Placa base GA-X99-Gaming 5P

La placa base es el componente principal de un ordenador que conecta y coordina todos los demás elementos, como el procesador, la memoria RAM, la GPU y los dispositivos de almacenamiento o entrada y salida. Actúa como un circuito central que permite la comunicación entre los componentes, proporcionando energía y soporte físico. Su diseño determina la compatibilidad del sistema y opciones de expansión disponibles.

Figura 4. 3: Placa base GA-X99 del PC empleada

Fuente: PCComponentes

4.1.3. Procesador Intel CORE I7-6800K

Componente principal para el funcionamiento del PC que ejecuta las instrucciones por medio de operaciones aritmeticológicas. Gestiona y coordina todas las tareas relacionadas con el procesamiento de la información.

Figura 4. 4: Procesador Intel CORE I7-6800K empleado

Fuente: PCComponentes

4.1.4. Tarjeta gráfica NVIDIA GeForce GT1080 Ti

Las tarjetas gráficas o GPU (del inglés Graphics Processing Unit o Unidad de Procesamiento Gráfico), son componentes de hardware que permiten gestionar la captura y procesamiento de imágenes, descargando esta tarea de la CPU para un rendimiento más eficiente. Gracias a su arquitectura de procesamiento de cálculos en paralelo, las GPUs son especialmente adecuadas para tareas de visión por computación y deep learning, ya que permite el procesamiento eficiente de imágenes por redes neuronales, acelerando la inferencia en estas aplicaciones de visión.

Figura 4. 5: Gráfica NVIDIA GeForce GT1080 Ti empleada

Fuente: PCComponentes

4.1.5. Televisor LG 43UR78

Televisor LG 43UR78 que nos proporcionó mantenimiento de su almacén.

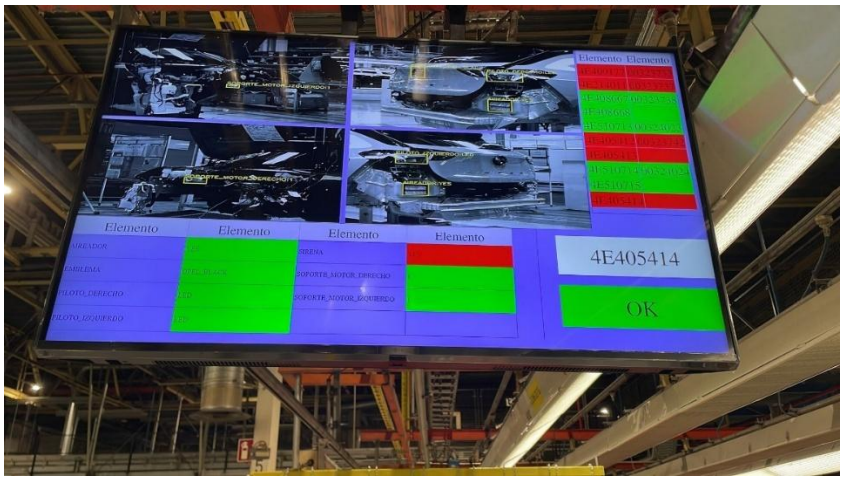


Figura 4. 6: Televisor para inspección visual de los operarios

Fuente: Elaboración propia

4.1.6. Baliza

La baliza es una SIEMENS 8WD4-420-5AD, compuesta por un módulo LED naranja y un módulo buzzer para emitir tanto señales visuales como sonoras. Se utiliza típicamente para alertas y notificación en entornos industriales. Proviene del almacén y se solicita al personal de mantenimiento con un coste asociado.

Figura 4. 7: Módulos LED y buzzer respectivamente de baliza SIEMENS 8WD4-420-5AD

Fuente: TEM

Este módulo SIEMENS se puede combinar de diferentes formas, en nuestro caso solo utilizamos el color naranja y la parte sonora. Ambos módulos comparten la alimentación de 24V y el GND (tierra). El LED y el buzzer se pueden controlan mediante señales separadas, pues la baliza

viene con varios terminales, en nuestro caso: uno para +24V, otro para GND y un cable para cada señal de activación con un total de 4 cables.

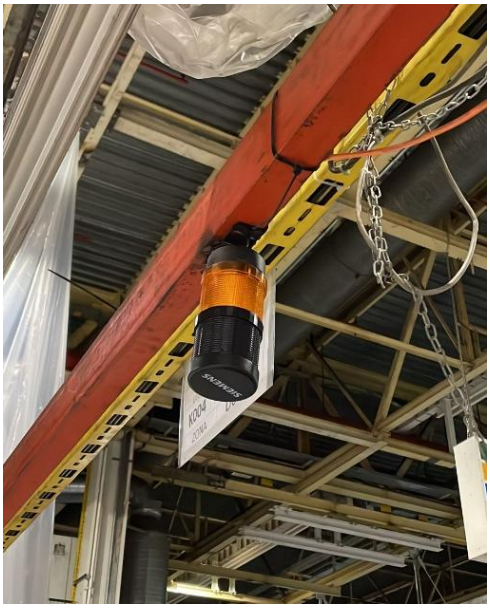


Figura 4. 8: Situación en la línea de la baliza SIEMENS 8WD4-420-5AD

Fuente: Elaboración propia

4.1.7. Placa de expansión con NVIDIA Jetson SoMs

En este proyecto se utilizó la placa de expansión 6 FPD-Link III con Jetson SoMs de Imagin Source, diseñada para trabajar con módulos NVIDIA Jetson, concretamente nosotros la utilizamos con la Jetson Xavier NX. Esta placa permite conectar hasta seis cámaras industriales mediante el protocolo FPD-Link III, lo que la hace ideal para aplicaciones de visión artificial que requieren múltiples entradas de video en tiempo real pues permiten distancias mucho mayores que el CSI o MIPI que veníamos utilizando hasta ahora.

La placa se alimenta a través de un conector de barril (2.5/5.5 mm), que acepta una entrada de voltaje de 12V a 36V DC con una potencia máxima de 60W. Esta fuente de alimentación suministra energía tanto al módulo Jetson como a las cámaras conectadas. Además, la placa también cuenta con una terminal de tornillo que ofrece una alternativa para la conexión de la fuente de alimentación, lo que proporciona flexibilidad en el diseño del sistema.

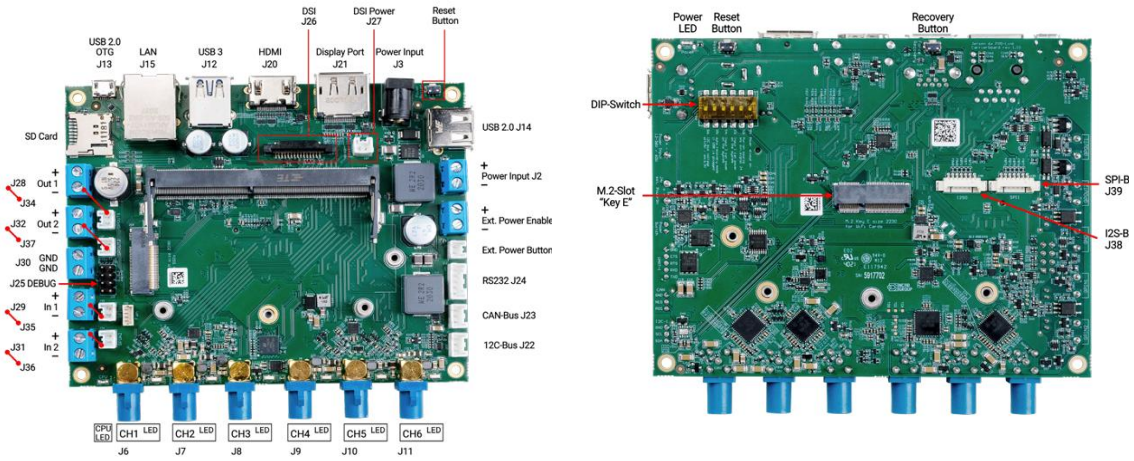


Figura 4. 9: Imagen de la documentación oficial de la 6 Channel Carrier Board – FPD-Link III

Fuente: (The Imaging Source, 2024)

Específicamente, la Jetson Xavier NX de NVIDIA es un módulo de alto rendimiento para sistemas embebidos, equipado con una GPU con arquitectura Volta de 384 núcleos CUDA y 48 núcleos Tensor, un procesador ARM de seis núcleos y 8 GB de RAM. Está pensada para ejecutar tareas de inteligencia artificial, visión por computador y modelos de deep learning directamente, sin depender de servidores externos.

Figura 4. 10: Imagen de la página web de NVIDIA de la Jetson Xavier NX

Fuente: (NVIDIA, 2024)

La elección de esta plataforma vino motivada por la necesidad de contar con un sistema embebido capaz de operar con cámaras situadas a más de 10 metros de distancia y se encontró un pack que ofrecía todo lo necesario integrado: placa, módulo Xavier NX y cámaras DCM36 compatibles en Imagin Source. Además, se buscaba dar un salto respecto a la Jetson Nano, ampliando la capacidad de procesamiento y conectividad para manejar múltiples entradas de video en tiempo real.

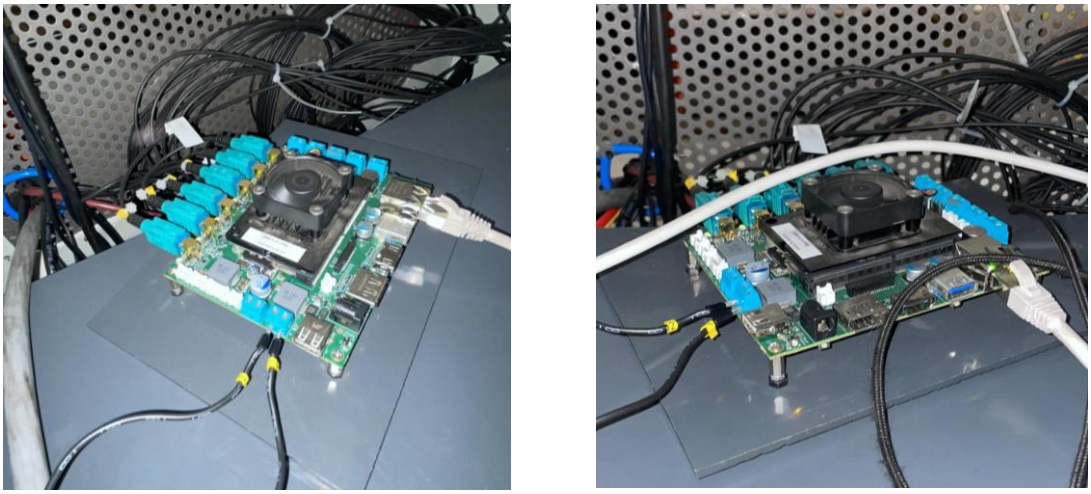


Figura 4. 11: 6-Channel FPD-Link III combinada con la placa Xavier NX de NVIDIA en el armario de la estación de visión

Fuente: Elaboración propia

No obstante, durante la implementación surgieron problemas importantes. La Jetson NX solo era compatible con versiones antiguas del sistema Jetpack (como la 4.6), que utilizaba Python 3.6, ya descontinuado. Esto dificultó la instalación de librerías actuales y la integración de modelos como YOLO, forzando el uso de versiones antiguas o poco mantenidas. También hubo un desfase temporal entre la adquisición del hardware y su integración efectiva, lo que complico la transición desde el entorno de desarrollo basado en Jetson Nano a la Xavier.

4.1.8. Cámaras BFLY-PGE-20E4M-CS

Estas cámaras, ahora de FLIR, antiguamente pertenecientes a PointGrey, ofrecen una resolución de 2.0 megapíxeles con un sensor Sony IMX249 (CMOS global shutter), ideal para capturar imágenes nítidas y sin distorsión en aplicaciones de visión artificial. Su tasa de fotogramas de 50 FPS permite un rendimiento optimo en tareas que requieren captura en tiempo real. Además, es una cámara monocromática, lo que maximiza la sensibilidad en condiciones de baja iluminación.

Con objetivo de mantener un enfoque de instalación low cost y minimizar costes, elegimos estas cámaras porque pudimos reutilizarlas de la instalación previa de visión artificial de la planta llamada EagleEye en la nave de pinturas.

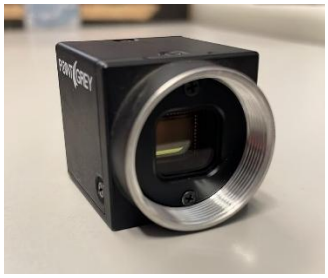


Figura 4. 12: Cámaras BlackFly S 2MP de FLIR (Pointgrey)

Fuente: Elaboración propia

Uno de los puntos fuertes de estas cámaras industriales es su compatibilidad con el protocolo GenICam (Generic Interface for Cameras), un estándar internacional desarrollado por el consorcio EMVA (European Machine Vision Association).

GenICam define una forma unificada y estandarizada de acceder a las funciones internas de cualquier cámara industrial, independiente del fabricante o del tipo de interfaz (GigE, USB3 Vision, Camera Link, etc.). Gracias a esto, se simplifica enormemente el desarrollo del software de adquisición de imágenes, ya que se puede interactuar con la cámara mediante descripciones genéricas en formato XML sin necesidad de programar funciones específicas del fabricante.

Highlighted	
Número de pieza	BFLY-PGE-20E4M-CS
Resolución	1600 × 1200
Megapíxeles	2
Velocidad de fotogramas máxima estándar	50
Modelo de sensor	Teledyne e2v EV76C570
Tamaño de píxel	4,5 µ m
Tipo de sensor	CMOS
Formato de sensor	1/1,8"
Tipo de obturador	Obturador global
Espectro	Mono
Filtro óptico	Ninguna
Soporte de lente	CS- mount (adaptador 5 m m C- mount no incluido)
Interfaz de datos	Gigabit Ethernet 1 Gbps

Figura 4. 13: Especificaciones de la cámara

Fuente: (Teledyne Vision Solutions, 2024)

4.1.9. Cámaras DFM 36CX290-ML VL

Cámara industrial de alta resolución equipada con un sensor CMOS que ofrece una resolución de 2.1 megapíxeles. Utiliza la interfaz FPD-Link III para una transmisión de datos eficiente y de baja latencia, ideal para aplicaciones de visión.

Captura imágenes en color con una profundidad de 12 bits y es capaz de alcanzar hasta 67 FPS a resolución completa.

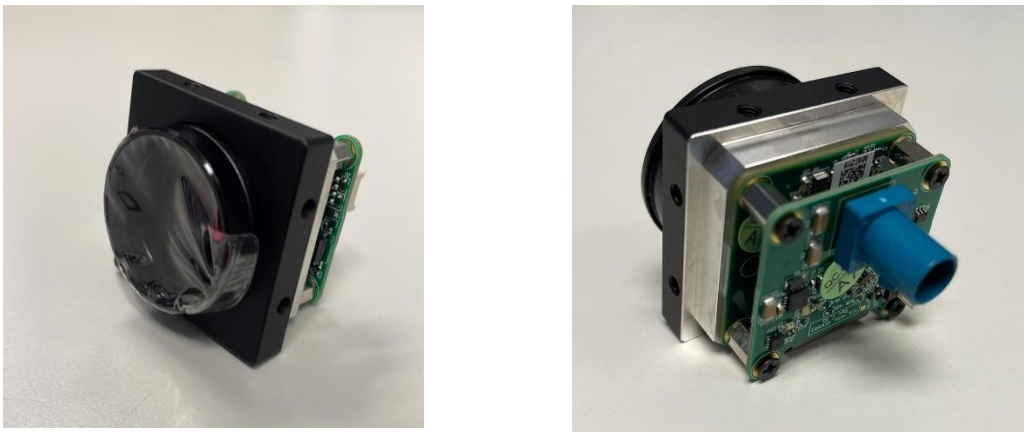


Figura 4. 14: Cámaras 36CX290-ML de Imagin Source

Fuente: Elaboración propia

Figura 4. 15: Empaquetado de las cámaras

Fuente: Elaboración propia

La elección de la cámara DFM 36CX290-ML se basa en su compatibilidad con la Jetson Xavier NX y la interfaz FPD-Link III, ideal para transmitir datos de manera estable en entornos industriales y aportando más fiabilidad al sistema. Su alto rendimiento con un sensor CMS de 2.1 MP y captura a

67n FPS la hace adecuada para inspección en líneas de producción rápidas. Estas cámaras fueron compradas a Imagin Source.

General	
Dynamic Range	12 bit
Resolution	1920x1080
Frame Rate at Full Resolution	60
Pixel Formats	12-Bit Bayer (RG)
Optical Interface	
Sensor Type	Sony IMX290LQR-C
Shutter Type	Rolling
Sensor Format	1/2.8 inch
Pixel Size	2.9 µm
Electrical Interface	
Interface	FPD-Link III via FAKRA connector
Supply voltage	10-27V
Current consumption	approx 80 mA @ 18 VDC
Mechanical Data	
Dimensions	H: 30 mm, W: 30 mm, L: 27.5 mm
Mass	12 g
Adjustments	
Shutter	15 µs to 1 s
Gain	0 dB to 72 dB

Figura 4. 16: Características de las cámaras

Fuente: (The Imaging Source, 2024c)

4.1.10. Objetivos FUJINON

Se compraron varios objetivos Fujinon para las cámaras DFM 36CX290-ML conectadas a la Xavier en función de las distancias de trabajo y las zonas focalizadas necesarias para las tareas de inspección y mutilaciones del vehículo.

Modelo	Distancia focal (mm)	Distancia de trabajo (cm)	Zona focalizada	Características destacadas
FJN-DF6HA-1S	6	50 – 60	40 x 30	Gran angular, ideal para distancias cortas
FJN-HF16HA-1S	16	120	30 x 20	Campo medio, buena relación entre detalle y cobertura
FJN-HF25HA-1S	25	120 (ajustado a 160 mm)	30 x 20	Teleobjetivo, mayor alcance para inspecciones lejanas

Tabla 1: Comparación objetivos FUJINON

a) *b)* *c)*

Figura 4. 17: Objetivos de la serie DF y HF calificados por distancia focal a) 6mm, b) 16 mm, c) 25 mm
Fuente: Elaboración propia

Figura 4. 18: Ejemplo de objetivo FJN-HF16HA-1S de distancia focal 16 mm

Fuente: Elaboración propia

Actualmente, tras varias pruebas de campo en condiciones reales, se comprobó que el modelo FJN-DF6HA-1S ofrecía un rendimiento optimo por su amplio ángulo de visión y su buena adaptación a distancia de trabajo medias en torno a 50-60 cm, que coincidían con la separación habitual entre las cámaras y el lateral de los vehículos. Estos objetivos también fueron comprados a Imagin Source.

Longitud focal [mm]	6
Pasos F	F1.2-F16
Ángulo de visión	57,3° x 43,8° (1/2")
Distancia de funcionamiento*5 (mm)	∞-100
Manejo del enfoque	Manual
Manejo del diafragma	Manual
Rosca para filtro [mm]	M27 x 0,5
Montura	Montura C
Peso (aprox.) [g]	50
Tamaño del sensor (estándar)*6	1/2" (4,5 µm)
Tamaño del sensor (máx.)*7	1/2" (4,5 µm)
Distorsión TV [%]	-1,84
Dimensión [mm]	Φ29,5 × 36,7

Figura 4. 19: Características objetivo FUJINON DF6HA-1S de 6 mm

Fuente: (FUJIFILM, 2024)

4.1.11. Objetivo TAMRON HF25HA-1S 1:1.4 16 mm 25.5 Ø

Objetivo utilizado para las cámaras Blackfly S de PointGrey. Es un objetivo de 16 mm con una apertura de f/1.4 y diámetro de 25.5 mm, ideal para aplicaciones de visión artificial en entornos de baja iluminación. Su apertura amplia permite una excelente captura de luz, lo que garantiza imágenes claras y detalladas, mientras que su distancia focal de 16 mm proporciona un campo de visión más que adecuado para nuestras tareas de inspección y monitoreo de precisión.

Figura 4. 20: Objetivos TAMRON HF25HA-1S

Fuente: Elaboración Propia

De nuevo, manteniendo el enfoque de instalación low cost reutilizamos estos objetivos al igual que las cámaras de PointGrey de la instalación de visión de pinturas, pues estos objetivos venían ya instalados en las propias cámaras.

4.1.12. Sensor SICK WL12L-2B530

Se incorporó el sensor fotoeléctrico SICK WL12L-2B530 al sistema como solución de detección y disparo para la captura de imágenes, gracias a su disponibilidad en el propio almacén de mantenimiento de la planta. Esta elección no solo optimizó los tiempos al aprovechar material ya disponible, sino que también se basó en criterios técnicos que lo hacen adecuado para un entorno industrial.

Decidimos utilizar sensores fotoeléctricos en lugar de utilizar cámaras como disparadores de hardware, pues ofrecen mayor robustez en el disparo, son más fiables, más económicos y consumen muchos menos recursos del sistema. Utilizar cámaras para esta tarea (otra opción que se planteó) implicaría un mayor consumo de procesamiento, aumento de costes y una reducción de la fiabilidad del sistema.

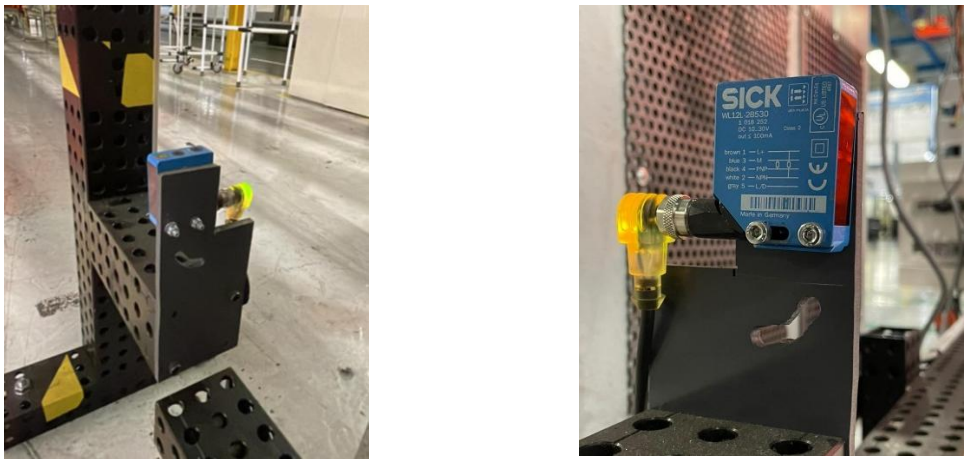


Figura 4. 21: Sensor SICK WL12L-2B530 situado en la parte derecha, encargado de disparar "trigger" las fotografías del vehículo mediante contadores

Fuente: Elaboración Propia

Ambos sensores nos fueron proporcionados por el almacén de mantenimiento y para su posterior rembolso por parte del departamento.

Figura 4. 22: Sensor SICK WL12L-2B530 situado en la parte izquierda, encargado de disparar para iniciar el ciclo de captura con la entrada de un coche a la estación

Fuente: Elaboración Propia

A nivel técnico, el WL12L-2B530 en concreto es un sensor óptico reflexivo con un rango de detección de hasta 7 metros (con reflector), respuesta rápida (tiempo de conmutación de 0,5 ms) y alta resistencia a condiciones adversas, gracias a su carcasa metálica IP67. Esto lo hace ideal para actuar como trigger para asegurar la toma de imágenes justo cuando el vehículo pasa por la posición deseada.

Principio funcional	Barrera fotoeléctrica réflex
Detalle del principio de funcionamiento	Reflector sin distancia mínima (autocolimación/óptica coaxial)
Alcance de detección máx.	0 m ... 18 m [1]
Filtros de polarización	Sí
Haz emitido	
Fuente de luz	Láser [2]
Tipo de luz	Luz roja visible
Tamaño del spot (distancia)	Ø 0,8 mm (600 mm)
Datos característicos del láser	
Referencia normativa	EN 60825-1:2014, IEC 60825-1:2007
Clase de láser	2 [3]
Datos característicos del LED	
Longitud de onda	650 nm
Ajuste	Ninguno
Características especiales	Distancia de conmutación: 600 mm, fija, sobre reflector P41F Enfoque a 240 mm, fijo Reserva de 4,5 sobre reflector P41F
Aplicaciones especiales	Detección de objetos pequeños, Detección de objetos a alta velocidad

Figura 4. 23: Características sensor fotoeléctrico SICK WL12L-2B530

Fuente: (SICK AG, 2024)

4.1.13. Switch D-Link 24 puertos DGS-1024D

Este switch, reciclado también de la instalación de visión de pinturas, cuenta con 24 puertos PoE (de sus siglas en inglés, Power Over Ethernet), lo que nos permite alimentar dispositivos como cámaras IP directamente a través del cable Ethernet, reduciendo la necesidad de fuentes de alimentación adicionales y simplificando el cableado del sistema.

Con soporte para velocidades Gigabit en todos los puertos, el swich garantiza una transferencia de datos rápida y estable, un aspecto crítico para manejar las imágenes capturadas por las cámaras y otros dispositivos conectados al sistema. Además, al ser compatible con el estándar IEE 803.3az Energy Efficient Ethernet, este siwtch consume menos energía cuando el tráfico de datos es bajo, lo que lo hace ideal en términos de sostenibilidad.

Figura 4. 24: Switch DGS-1024D 24 puertos de la marca D-Link

Fuente: (D-Link, 2023)

En la estación, se encuentra situado dentro del armario, haciendo de intermediario entre nuestro PC industrial y las cámaras conectadas mediante Ethernet. Fue reciclado también de la instalación de pinturas del EagleEye.

Figura 4. 25: Switch D-Link DGS-1024D 24 puertos en la estación

Fuente: Elaboración propia

4.1.14. Cables Ethernet RJ45

Se utilizaron cables Ethernet RJ45 CAT6 de 15 metros comprados de AliExpress para la conexión de las cámaras industriales Blackfly S de PointGrey al switch, ya que estas están equipadas con conectores RJ45 como única interfaz de red. Su uso garantiza una comunicación rápida y estable, siendo además la opción más estandarizada para este tipo de dispositivos.

Figura 4. 26: Cables Ethernet RJ45 CAT6 de la marca deleyCON comprado por AliExpress

Fuente: (AliExpress, 2024)

4.1.15. Cables FPD-Link III

Se emplearon cables compatibles con la interfaz FPD-Link III para la conexión directa de las cámaras DFM 36CX290-ML a la placa de expansión de la Xavier con Jetson SoM, ya que esta plataforma cuenta con entradas diseñadas específicamente para este tipo de transmisión. Estos cables permiten enviar vídeo, alimentación y señales de control a través de una sola conexión, reduciendo el cableado y mejorando la robustez del sistema. También fueron comprados a Imagin Source.

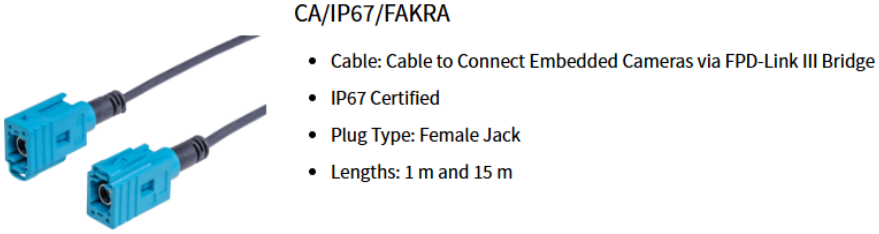


Figura 4. 27: Cables FAKRA para conectar cámaras embebidas en la placa vía FPD-Link III

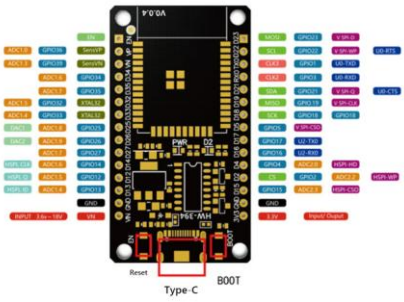
Fuente: (The Imaging Source, 2024)

Figura 4. 28: Ejemplo de conexión de las cámaras mediante estos cables vía FPD-Link III

Fuente: (The Imaging Source, 2024)

4.1.16. Módulo ESP32-WROOM-32

El módulo ESP32-WROOM-32 es un microcontrolador desarrollado por Espressif Systems, basado en el chip ESP32-D0WDQ6. Este SoC (por sus siglas en inglés, *System on Chip*) integra un microprocesador de doble núcleo Xtensa LX6 de 32 bits, capaz de operar hasta 240 MHz, y combina conectividad Wi-Fi 802.11 b/g/n y Bluetooth 4.2 (BLE y BR/EDR), lo que lo convierte en una solución versátil y barata para aplicaciones de IoT, automatización industrial a pequeña escala o sistemas embebidos de control y comunicación.



(a)

(b)

Figura 4. 29: a) Pin-Out del ESP32-WROOM-32 ; b) Modelo ESP32-WROOM-32

Fuente: (AliExpress, 2024)

Este componente se compró en AliExpress y se seleccionó principalmente por su bajo coste, su facilidad de programación mediante un entorno conocido Arduino IDE y su capacidad para actuar como puente y gestor de señales entre sensores y el PC o una red inalámbrica. En concreto, lo utilizamos para:

1. Acondicionar señales provenientes de sensores, adaptándolas para ser procesadas por el software principal

2. Transmitir señales digitales hacia una baliza luminosa, activándola en función del análisis de la red neuronal

4.1.17. Armario

Para alojar y proteger los equipos informáticos, se ha reutilizado un armario recuperado de materiales desechos por el taller de mantenimiento. Esto nos permite mantener el hardware protegido sin necesidad de adquirir ninguna estructura y seguir con la filosofía low cost.

Figura 4. 30: Armario encargado de proteger los componentes del hardware del sistema

Fuente: Elaboración propia

4.1.18. Soporte impreso en 3D para cámaras DFM 36CX290-ML

Carcasa diseñada en Inventor y fabricada mediante impresión 3D en PLA.

Diseñada a medida para sujetar las cámaras MIPI de forma precisa y segura, con geometría adaptada a sus dimensiones. La impresión en 3D nos la realizaron los compañeros del taller de mejora continua con sus propias impresoras.

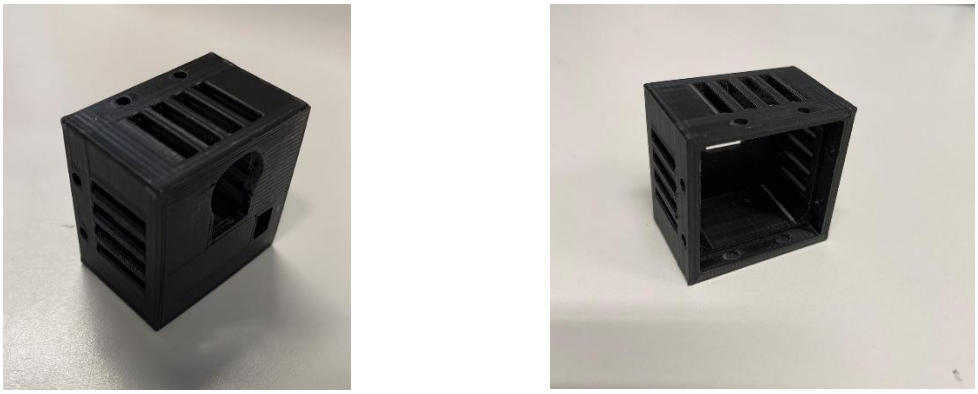


Figura 4. 31: Soportes para cámaras MIPI fabricados en PLA

Fuente: Elaboración propia

Los planos detallados de este componente se incluyen en el apartado Anexos A: Planos De Soporte Para Cámaras MIPI.

4.1.19. Placas de PVC para anclaje de cámaras a los perfiles

Placas recortadas de láminas de PVC rígido, con agujeros específicos para permitir el atornillado de las cámaras y los sensores a la estructura. Sirven de punto de apoyo entre las cámaras y los sensores para aportar estabilidad.

Todas las piezas cortadas de PVC fueron recicladas de los retales de los compañeros del taller de mejora continua.

Figura 4. 32: Placas de PVC para aportar rigidez al apoyo de las cámaras en la estructura

Fuente: Elaboración Propia

Los planos detallados de este componente se incluyen en el apartado Anexos B: Placas De PVC Para Anclaje De Cámaras.

En este contexto, también podemos encontrar los planos de las placas de PVC que fabricamos, también a partir de retales del taller de nuestros compañeros de mejora continua para anclar los sensores a las estructuras de la estación. Estos planos se encuentran en el apartado Anexos C: Placas De PVC Para Anclaje De Sensores.

4.1.20. Soporte articulado ballhead KJ-7 para cámaras BFS de Pointgrey

Soporte metálico con rótula que permite orientar las cámaras Blackfly S en múltiples direcciones. Atornillado a la placa de PVC, facilita ajustes finos en el ángulo de visión de la cámara.

Fue reciclado de las instalaciones de visión de la zona de pinturas, junto con las propias cámaras.

Figura 4. 33: Soporte articulado "ballhead" KJ-7

Fuente: Elaboración propia

El soporte dispone en su base de un agujero roscado pasante de 3/8" BSW (British Standard Withworth). Debido a esto, fue necesario realizar agujeros pasantes de 10 mm de diámetro en las placas de PVC que conforman la base de montaje de cada cámara Blackfly S, permitiendonos asi el anclaje del soporte KJ-7.

Figura 4. 34: Base con agujero roscado 3/8" BSW del soporte

Fuente: Elaboración Propia

4.1.21. Adaptador de trípode para Cámaras BFS de Pointgrey

Pieza mecánica estándar que permite acoplar las cámaras Blackfly S al soporte “ballhead” KJ-7. Asegura compatibilidad y una fijación firme del cuerpo de cámaras al soporte articulado.

De nuevo, fue reciclado de las instalaciones de visión de la zona de pinturas, junto con las cámaras y los soportes “ballhead” KJ-7. Venían ya atornillados a las cámaras.

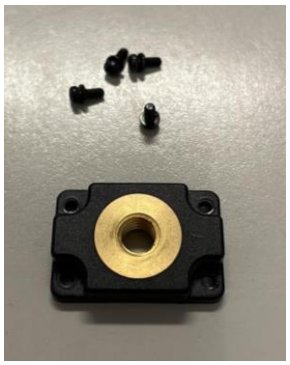


Figura 4. 35: Adaptador de trípode para las cámaras BlackFly S que permiten su conexión a otros soportes Fuente: Elaboración Propia

Disponen de un tamaño de tuerca 1/4”-20 donde 1/4” se refiere al diámetro del tornillo en pulgadas, equivalente a 6.35 mm de diámetro, y el 20 nos indica el paso de la rosca, lo que significa que hay 20 hilos por pulgada en la rosca del tornillo. Encaja perfectamente con el tornillo roscado superior del soporte “ballhead” de KJ-7 pues utiliza el estándar para conexiones de cámaras que es, precisamente, 1/4".20.

Para verlo de forma más clara y resumida del anclaje de las cámaras a la estructura, se han realizado unos pequeños diagramas:

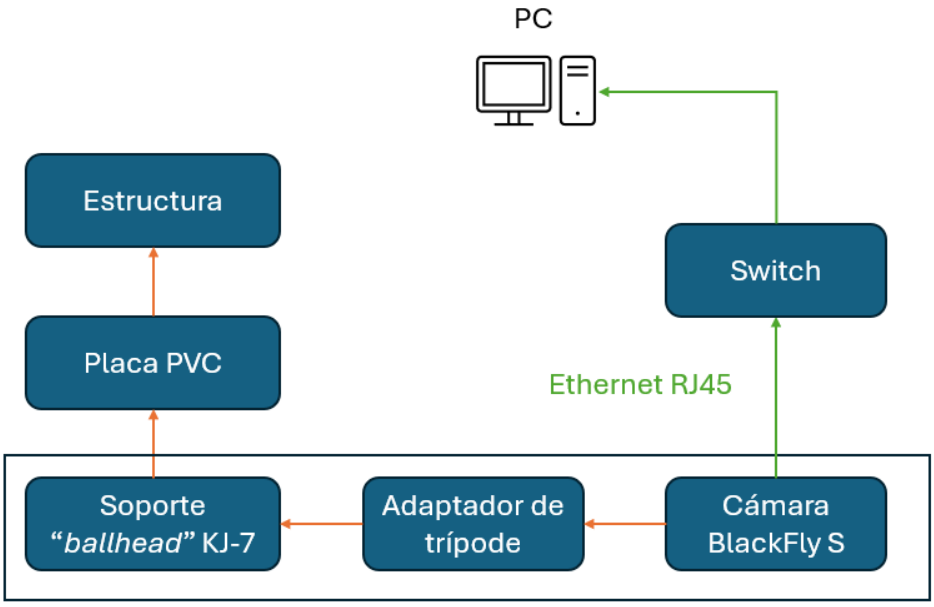


Figura 4. 36: Diagrama conexión mecánica y física cámaras BlackFly S de Pointgrey

Fuente: Elaboración Propia con PowerPoint

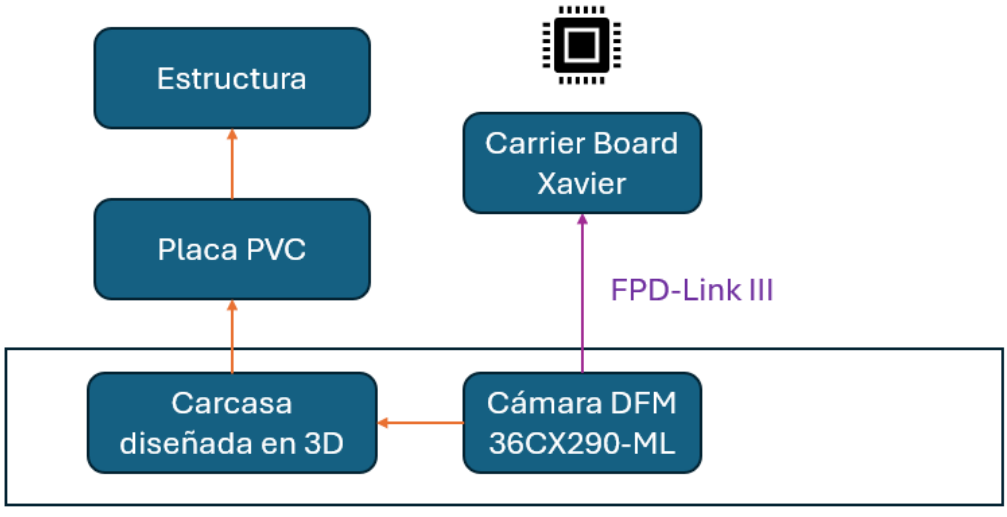


Figura 4. 37: Diagrama conexión mecánica y física cámaras DFM de Imagin Source

Fuente: Elaboración Propia con PowerPoint

4.1.22. Perfiles de aluminio

La estructura mecánica utilizada en la estación se basa en perfiles que fueron reciclados y reutilizados para cumplir con la filosofía de bajo costo y sostenibilidad del diseño. Estos perfiles fueron originalmente recuperados del taller de metrología de la planta de STELLANTIS, donde se utilizaban para sujetar piezas dimensionales y realizar mediciones de precisión. Para lograrlo, disponían de pinzas y uniones especificas diseñadas para sujetar piezas con alta precisión.

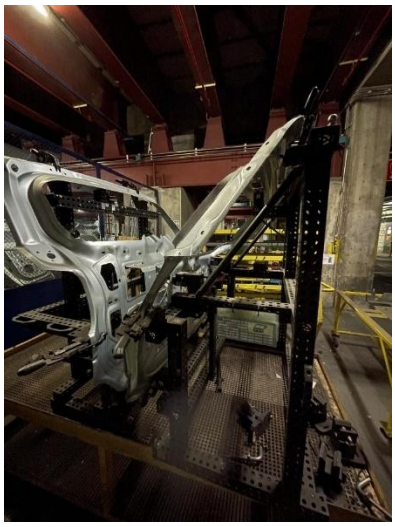


Figura 4. 38: Piezas de tolerancias dimensionales procedentes del laboratorio de metrología, almacenadas en el sótano de prensas de la planta

Fuente: Elaboración Propia

Los perfiles y elementos de unión utilizados en la estructura del sistema provienen del fabricante PALETTI, donde se pueden encontrar las medidas de la mayoría de los perfiles utilizados, otros, más antiguos, son de otras empresas diferentes. PALETTI es una empresa reconocida internacionalmente por su especialización en sistema de fijación y equipos de medición de precisión.

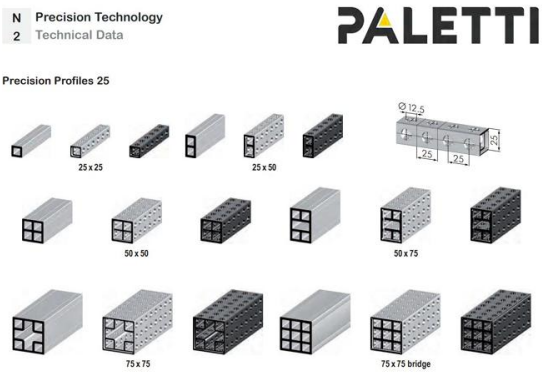


Figura 4. 39: Diferentes tipos de perfil del catálogo de PALETTI

Fuente: (Paletti Profilsysteme GmbH & Co. KG, 2023)

Las estructuras están construidas con aluminio extruido (aleación AlMg4,5Mn), sometido a anodizado E6/EV1 (en acabado natural o en negro) con una dureza superficial de 250-350 HV, lo que garantiza una alta resistencia mecánica y térmica, así como una buena durabilidad en entornos industriales, lo que la convierte en un material ideal para nuestras estaciones.

Como parte del proceso de diseño y adaptación de estos perfiles a la nueva instalación, desmontamos cuidadosamente las pinzas y uniones (tornillos, pasadores roscados, pasadores pasantes, tuercas, arandelas…) originales en el sótano de prensas de la planta, dejando los perfiles libres para su reutilización en nuestras estaciones de visión.

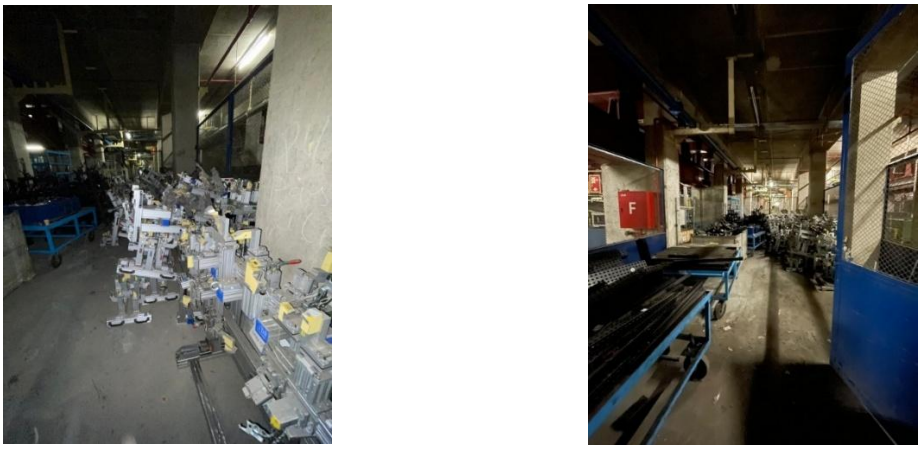
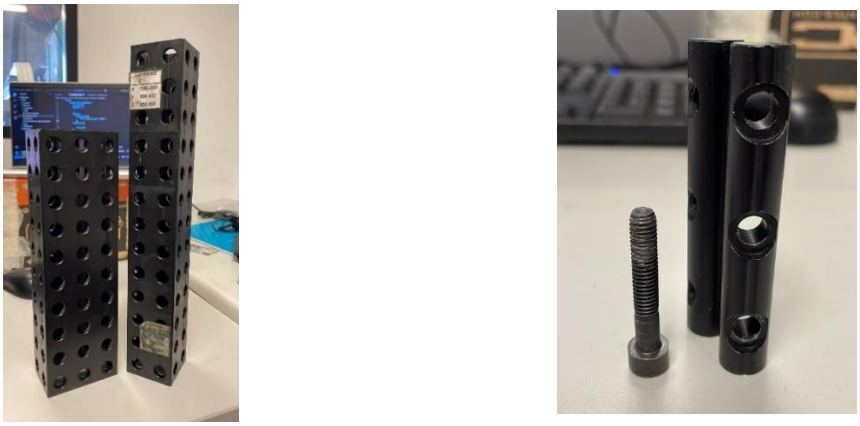


Figura 4. 40: imágenes del operativo de desmontaje de los perfiles ubicados en el sótano de prensas de la planta, para su posterior reutilización en las estaciones de visión

Fuente: Elaboración Propia

Este enfoque permitió aprovechar materiales ya disponibles en la planta al igual que con las cámaras de las instalaciones de pinturas.



(a) *(b)*

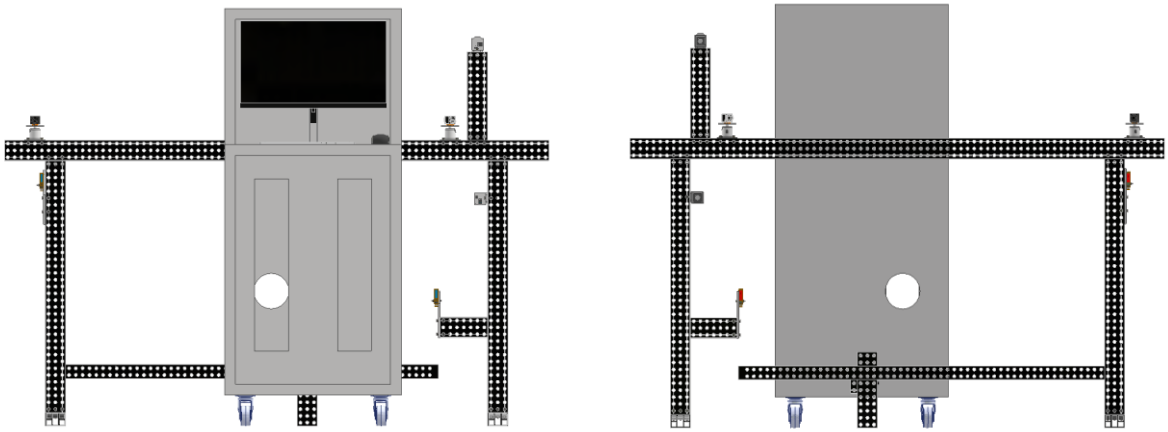
Figura 4. 41: a) Perfiles b) Elementos de unión de los perfiles: tornillos de cabeza allen DIN 912 M6, y pasadores roscados y pasantes

Fuente: Elaboración Propia

4.2. Diseño Estructural De La Estación

4.2.1. Vista general 3D

Se incluye un modelo 3D de la parte derecha de la estación realizado en Autodesk Inventor, donde se representa la disposición física de los perfiles de aluminio, soportes, cámaras y sensores (la parte izquierda es simétrica). Este modelo fue utilizado como guía para el montaje físico, facilitando la organización de los elementos antes de su instalación real.



(a) (b)

Figura 4. 42: a) Vista de la parte frontal derecha de la estación de visión b) Vista de la parte posterior derecha de la estación de visión

Fuente: Elaboración Propia en 3D con Autodesk Inventor

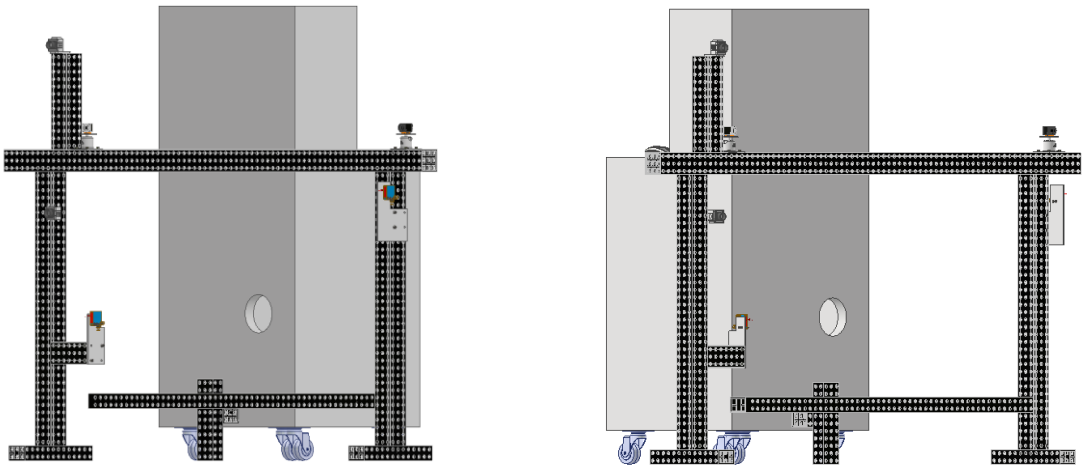
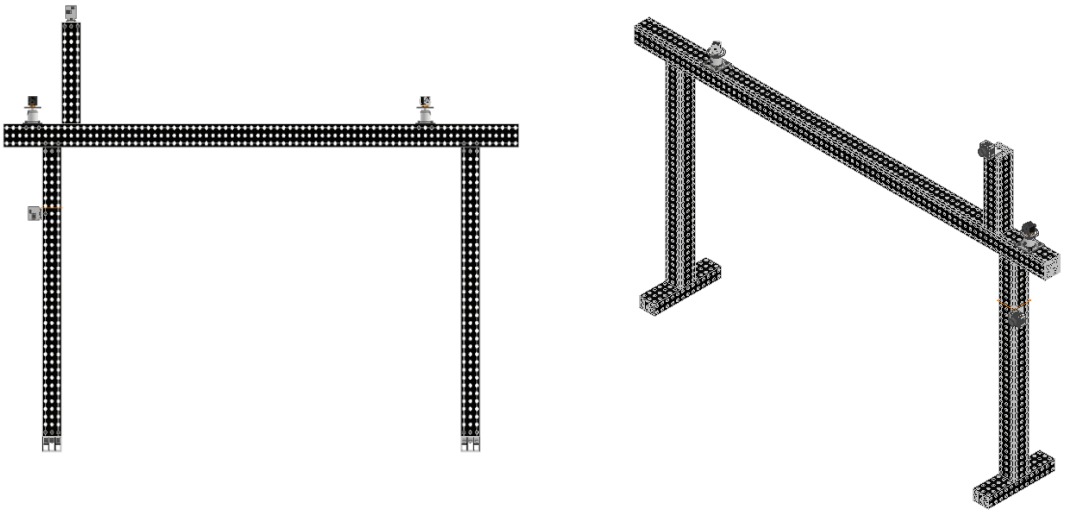


Figura 4. 43: Dos vistas complementarias de la parte semilateral derecha de la estación de visión

Fuente: Elaboración Propia en 3D con Autodesk Inventor



(a) (b)

Figura 4. 44: a) Vista de la parte frontal izquierda de la estación de visión b) Vista isométrica de la parte izquierda de la estación de visión

Fuente: Elaboración Propia en 3D con Autodesk Inventor

Las medidas de los perfiles utilizados se pueden observar en los planos de la parte del documento referida a Anexos D: Estación De Visión Artificial.

4.2.2. Unión de los perfiles y estructura

La unión de los perfiles se realiza mediante una combinación de dos tipos de pasadores, uno pasante y otro roscado, que trabajan en conjunto para formar una conexión estable. Estos elementos se insertan en los agujeros alineados de los perfiles, permitiendo la conexión entre ambos, formando bloques. En el perfil superior se utiliza uno pasante, mientras que en el inferior se emplea uno roscado.

Una vez posicionados, se fija el conjunto utilizando tornillos Allen M6, acompañados de una arandela para distribuir uniformemente las fuerzas y asegurar que la unión permanezca firme bajo condiciones de trabajo dinámicas.

Figura 4. 45: Unión atornillada con DIN 912 M6 que permite fijar los perfiles entre sí

Fuente: (Paletti Profilsysteme GmbH & Co. KG, 2023)

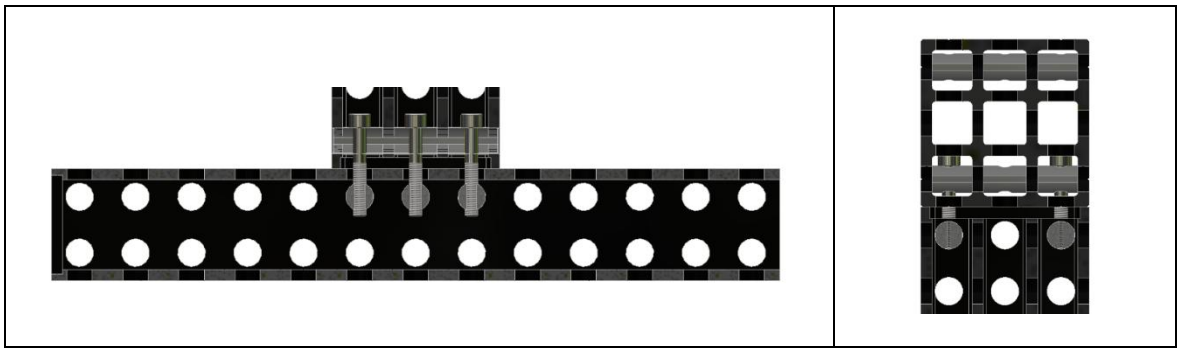


Figura 4. 46: Media vista del interior de un perfil, simulada en 3D para visualización de la unión atornillada de los perfiles

Fuente: Elaboración Propia en 3D con Autodesk Inventor

4.2.3. Ubicación de los componentes en la estación

En esta sección se muestran imágenes reales de la instalación donde se aprecia la ubicación final de los distintos componentes del sistema en el final de la línea 2 de TRIM de Stellantis.

Figura 4. 47: Parte posterior derecha de la estación

Fuente: Elaboración Propia

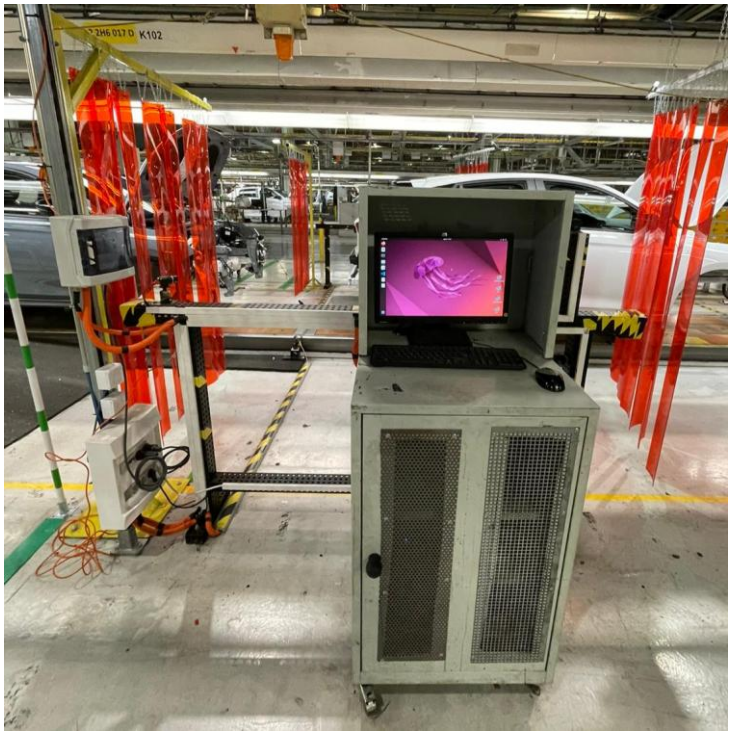


Figura 4. 48: Parte frontal derecha de la estación

Fuente: Elaboración Propia

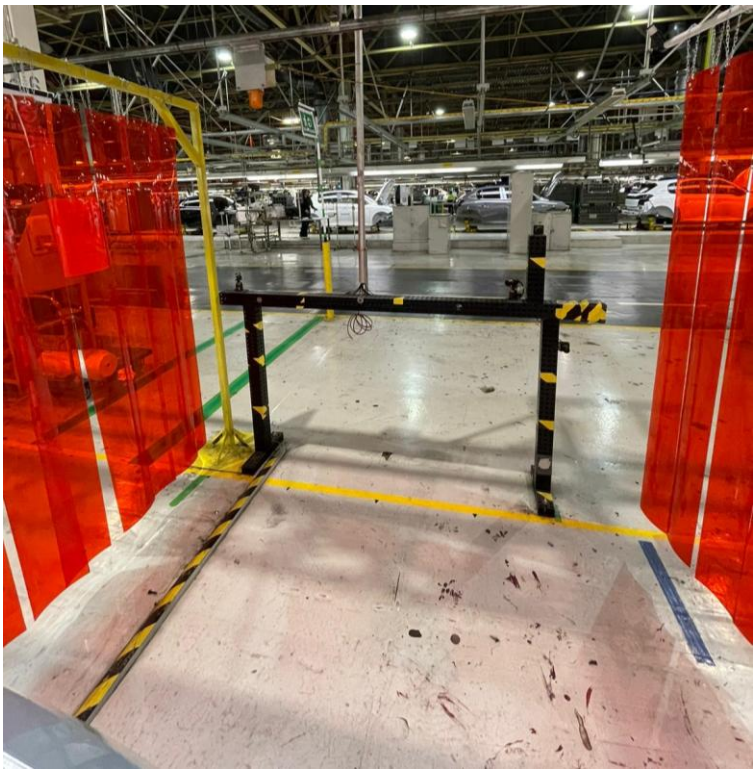


Figura 4. 49: Parte posterior izquierda de la estación

Fuente: Elaboración Propia

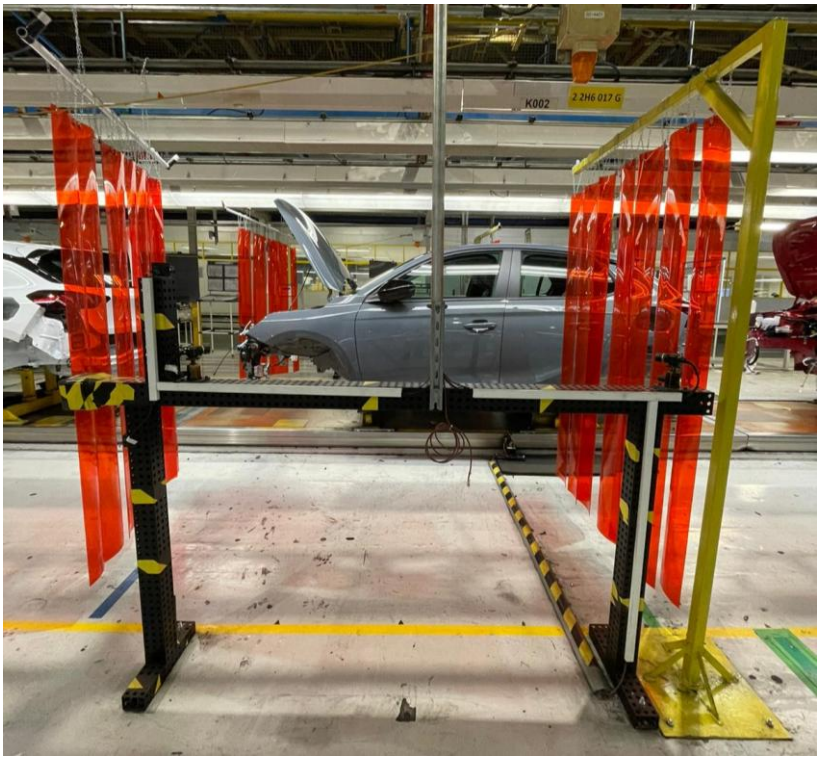


Figura 4. 50: Parte frontal izquierda de la estación

Fuente: Elaboración Propia

Figura 4. 51: Situación cámara frontal izquierda

Fuente: Elaboración Propia

Figura 4. 52: Situación cámara frontal derecha

Fuente: Elaboración Propia

Figura 4. 53: Situación cámara trasera izquierda

Fuente: Elaboración Propia

Figura 4. 54: Situación cámara trasera derecha

Fuente: Elaboración Propia

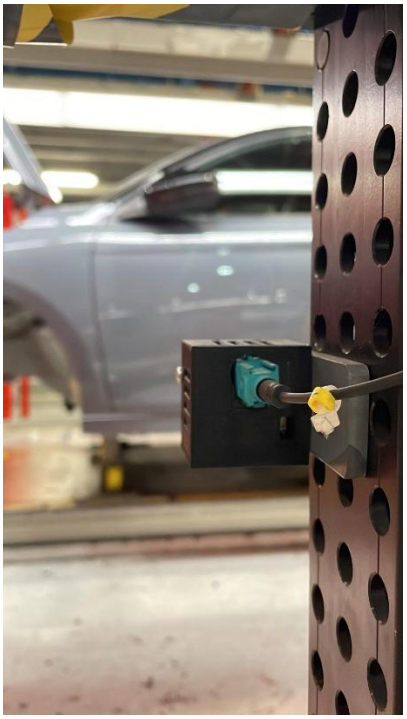


Figura 4. 55: Situación cámara lateral izquierda inferior

Fuente: Elaboración Propia

Figura 4. 56: Situación cámara lateral derecha inferior

Fuente: Elaboración Propia

Figura 4. 57: Situación cámara lateral izquierda superior

Fuente: Elaboración Propia

Figura 4. 58: Situación lateral derecha superior

Fuente: Elaboración Propia

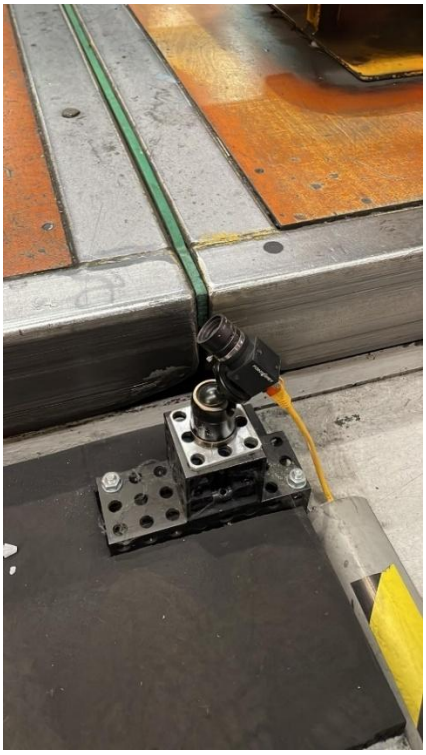


Figura 4. 59: Situación cámara OCR derecha

Fuente: Elaboración Propia

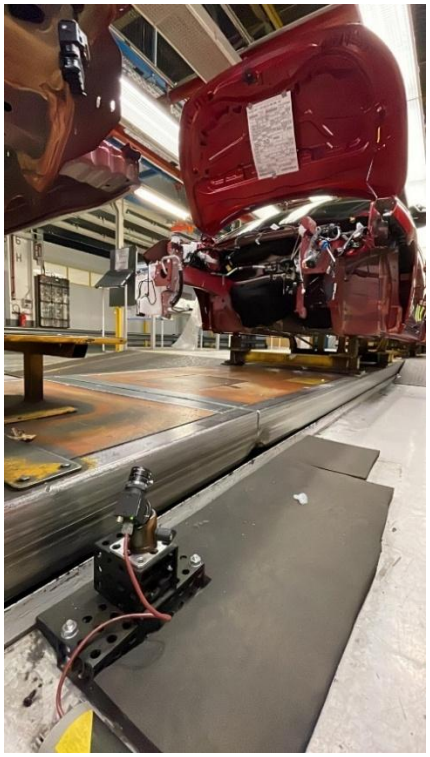


Figura 4. 60: Situación cámara OCR izquierda

Fuente: Elaboración Propia

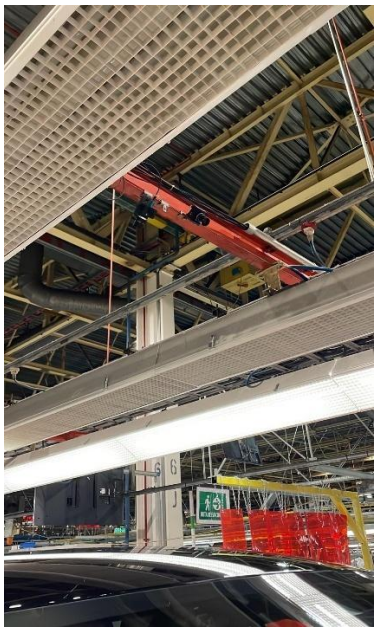


Figura 4. 61: Situación cámaras superiores

Fuente: Elaboración Propia

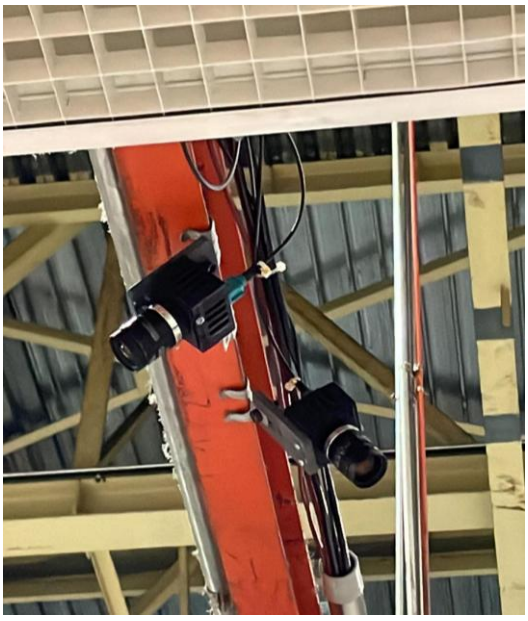


Figura 4. 62: Cámaras superiores

Fuente: Elaboración Propia

4.2.4. Layout de la estación

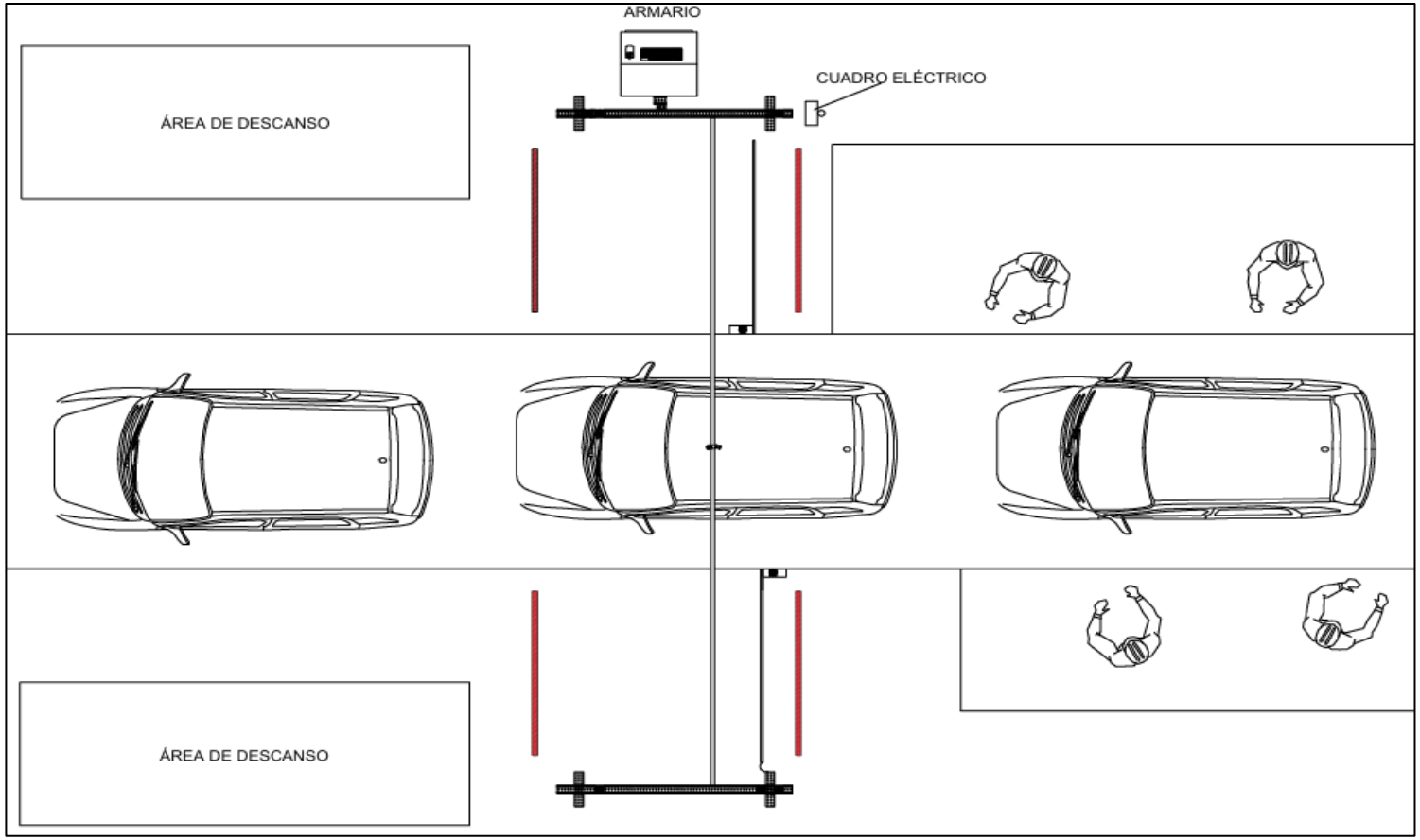


Figura 4. 63: Layout de la estación

Fuente: Elaboración Propia con AutoCAD

4.3. DISEÑO ELÉCTRICO DE LA ESTACIÓN

La instalación eléctrica de la estación de visión artificial se ha diseñado con el objetivo de garantizar la seguridad eléctrica, la compatibilidad de señales y la correcta comunicación entre sensores, actuadores y la unidad de procesamiento (ESP32).

Figura 4. 64: Cuadro eléctrico de la estación de visión artificial

Fuente: Elaboración Propia

Alimentación general

El sistema se alimenta a partir de una línea de 230 V AC procedente de la instalación principal. Esta tensión entra al cuadro eléctrico y es protegida mediante un interruptor automático bipolar (PIA) de 2A. A continuación, se alimenta una fuente de alimentación Siemens LOGO!Power, la cual convierte la tensión alterna de red en una salida continua de 24 V DC. Esta tensión es utilizada para alimentar diversos dispositivos de la estación, incluyendo los sensores conectados, una baliza y una placa de circuito impreso (PCB) fabricada en la oficina.

Figura 4. 65: Alimentación

Fuente: Elaboración Propia

Conversión y aislamiento de señales

La PCB hace un papel clave en la adaptación de niveles lógicos y en el aislamiento galvánico entre los elementos del sistema. Esta placa, cuyo circuito detallado se encuentra en la sección Anexos E: Plano Circuito PCB y que podemos observar en la figura 4. 63 contiene:

- Resistencias de 100, 200 y 51100 ohmios
- 4 diodos de protección 1N4007
- 4 condensadores de 10 nF
- 1 circuito integrado de optoacopladores modelo TLP521-4
- Diversos conectores de terminales fijo de 2 y 3 pines

Figura 4. 66: Placa de circuito impreso (PCB) fabricada

Fuente: Elaboración Propia

El objetivo principal de esta PCB es permitir la conversión segura de señales de 24 V DC provenientes de los sensores industriales conectados, a señales de 3,3 V compatibles con el microcontrolador ESP32, que es el encargado de procesar la lógica y detección y control. Esta conversión se realiza a través de optoacopladores, que permiten transferir la señal eléctrica de entrada mediante un haz de luz, proporcionando un aislamiento eléctrico entre la etapa de potencia (24 V) y la etapa de control (3,3 V).

Comunicación con el ESP32 y generación de alarma

Las señales provenientes de los sensores (por ejemplo, la detección de una etapa del coche para disparo de imágenes) se introducen en la PCB, donde son procesadas y convertidas a 3,3 V para ser interpretadas por el ESP32. Este microcontrolador se comunica con nuestro PC, donde se ejecuta el software de la estación.

En caso de detección de una condición de fallo (NOK), el software genera una salida digital desde el ESP32. Esta señal pasa nuevamente por la PCB, donde se convierte de 3,3 V a 24 V para, posteriormente, activar una baliza acústico-luminosa que alerta visual y sonoramente al operario sobre la posible incidencia.

Este diseño permite integrar sensores industriales de 24 V con sistemas de control basados en microcontroladores que trabajan a bajo voltaje, manteniendo al mismo tiempo seguridad eléctrica gracias al aislamiento proporcionado por los optoacopladores.

Se pueden consultar los detalles del plano eléctrico de la instalación en la sección Anexos F: Plano Eléctrico De La Estación. Así como los detalles del plano sinóptico de red en la sección Anexos G: Plano Sinóptico De Red De La Estación.

4.4. SOFTWARE DE LA ESTACIÓN

El software de la estación es el encargado de gestionar todo el proceso de control de calidad automatizada en la línea de producción. Su función principal es capturar imágenes en tiempo real, extraer información relevante y analizarla para detectar posibles discrepancias en la instalación de componentes.

4.4.1. Arquitectura del software

El software de la estación, desarrollado en Python, sigue un flujo estructurado que permite capturar, analizar y verificar la información de cada vehículo. Este sistema opera sobre una plataforma basada en Linux, específicamente Ubuntu 22.04 LTS (Long Term Support), lo que garantiza estabilidad, flexibilidad y facilidad de integración con los dispositivos industriales utilizados.

Todo comienza con la captura de la imagen de la hoja de montaje, ubicada bajo el capó del vehículo. Para estas fotos y para todas las siguientes, se utilizan cámaras industriales compatibles con el estándar GenICam, lo que permite un control flexible y unificado sobre parámetros como exposición o ganancia.

Gracias a este estándar, es posible integrar fácilmente bibliotecas como Spinnaker (de FLIR) o su equivalente en Python, PySpin, facilitando la configuración y captura de imágenes desde el software sin depende de herramientas propietarias. Las cámaras se activan mediante una señal digital enviada por un sensor laser retroreflectivo, que actúa como sistema de disparo sincronizado: al detectar el paso del vehículo, este sensor genera una señal que se envía a un microcontrolador ESP32, el cual lanza la orden de captura hacia el sistema de visión artificial.

A partir de estas imágenes, se aplica un proceso de Reconocimiento Óptico de Caracteres (OCR), el cual extrae información relevante como la lista de componentes que el vehículo debería llevar instalados (referencias). Uno de los datos más importantes obtenidos es el NOF (Número de Orden de Fabricación), que permite asociar todas las imágenes al vehículo correspondiente y garantizar su trazabilidad. Además, el NOF se almacena en archivos de texto junto con la fecha y la hora, creando un registro histórico de los vehículos inspeccionados y del rendimiento del OCR, para su posterior consulta por parte de supervisores de la línea y para nosotros en un momento dado.

Una vez identificado el NOF y realizada la extracción de datos, el sistema procede a capturar imágenes del vehículo desde diferentes ángulos, (frontal, trasero, lateral). Estas imágenes se almacenan en distintas ubicaciones:

1. En el escritorio del sistema hay un symbolic link de la carpeta de imágenes del disco duro, para consulta de supervisores y coordinadores
2. En una carpeta interna de nuestra estructura del programa, donde serán utilizadas más adelantes para el análisis con la red neuronal

El procesamiento de estas imágenes se lleva a cabo con la red neuronal YOLOv5, una red neuronal especializada en la detección de objetos. Esta analiza cada fotografía y genera un output con los componentes que ha identificado en la imagen del vehículo, junto con su ubicación en la imagen. Posteriormente, los datos extraídos del OCR, organizados por imagen, se comparan con los resultados obtenidos por YOLOv5, permitiendo verificar si el vehículo cuenta con todos los componentes que debería llevar. Esta comparación actualmente se realiza mediante el uso de diccionarios en Python y archivos JSON, lo que facilita la gestión de la información.

Al finalizar la inspección, el sistema genera un resultado indicando si la verificación ha sido exitosa (OK), si se han encontrado discrepancias en algún elemento (NOK) o si no ha sido posible obtener una lectura correcta del OCR (MISS). Estos resultados, junto con las imágenes analizadas, se muestran en un televisor situado en la línea de producción a modo de interfaz GUI, donde los operarios pueden visualizar en tiempo real el estado de cada vehículo y tomar las medidas necesarias.

En caso de detectarse un fallo (NOK), el ESP32 envía una señal a una baliza SIEMENS que activa una alerta visual y sonora, permitiendo a los operarios actuar de inmediato.

Finalmente, todos los resultados y las imágenes se almacenan en un archivo de texto para futuras consultas, análisis y estadísticas de la estación.

4.4.2. Diagramas del software

4.4.2.1. Diagrama de casos de uso

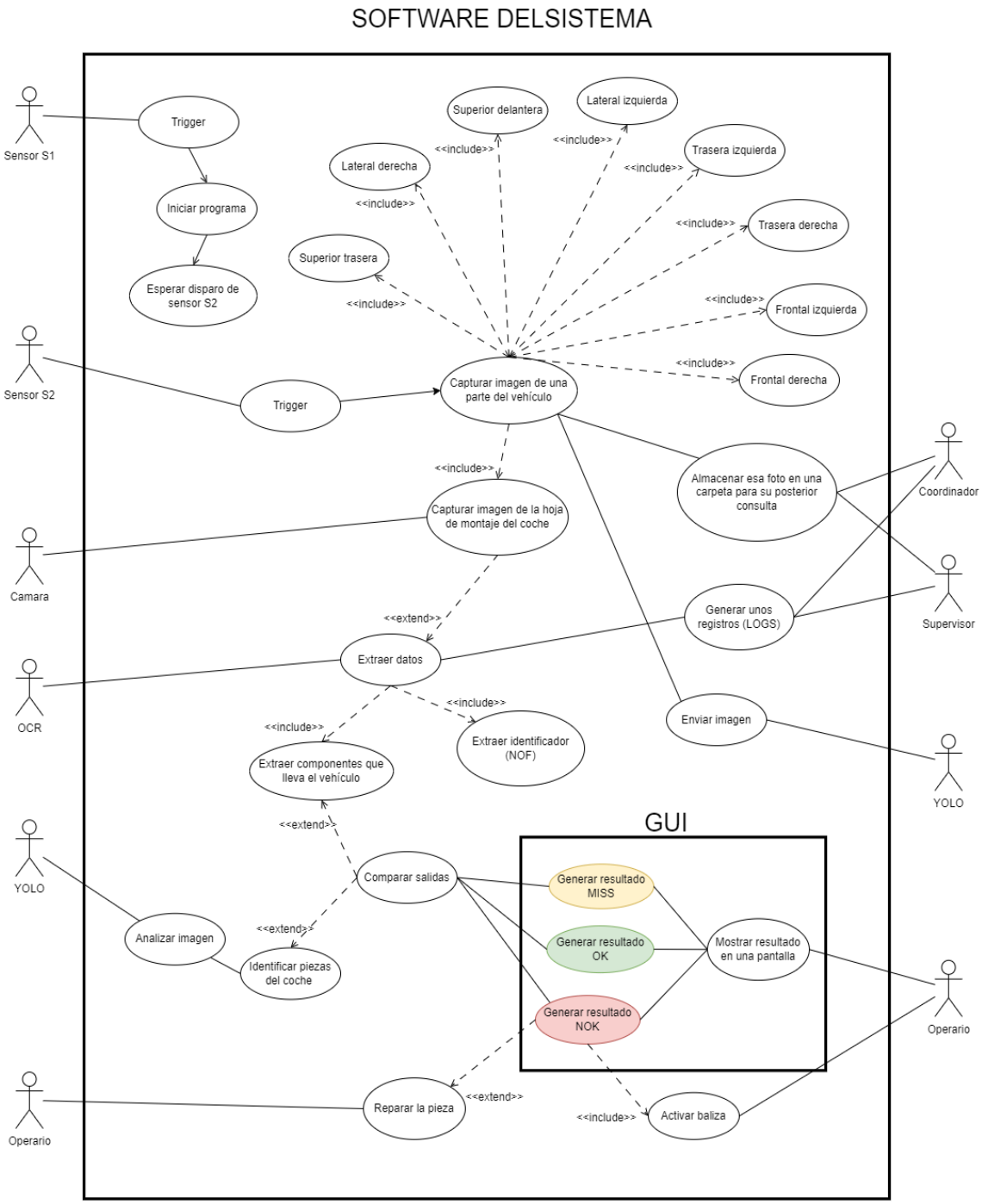


Figura 4. 67: Diagrama de casos de uso de la estación

Fuente: Elaboración Propia con Draw.io

4.4.2.2. Diagrama de actividades

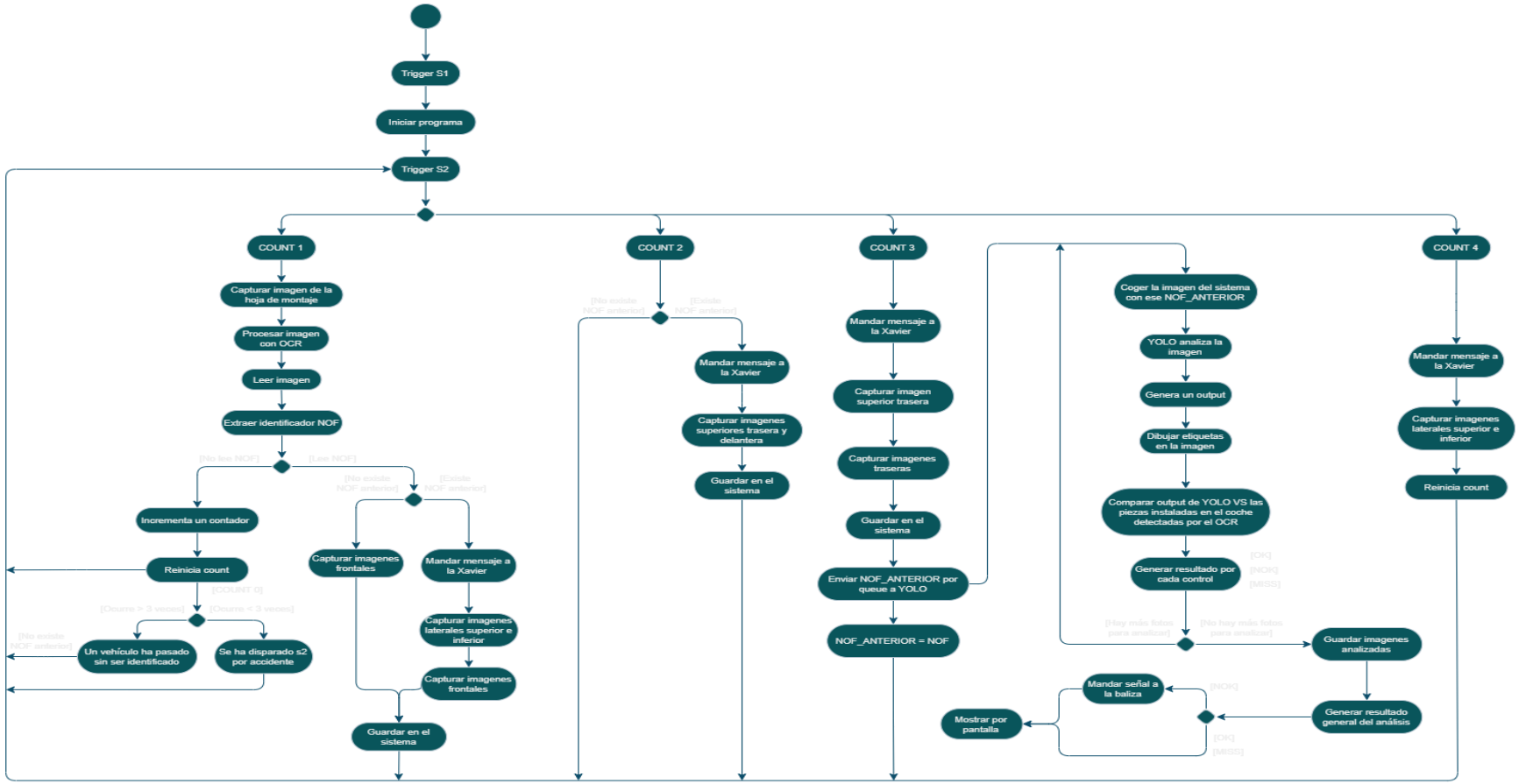


Figura 4. 68: Diagrama de procesos de la estación

Fuente: Elaboración Propia con Draw.io

El proceso de toma de imágenes de la estación puede resultar algo confuso debido a que capturamos imágenes de dos coches diferentes durante un ciclo completo de captura. Por ello, mediante unas viñetas realizadas en Draw.io se introducen para dar más visibilidad general del proceso de captura. Además, se añaden las fotos que las cámaras captura a lo largo de todo el proceso de captura de imágenes de la estación para ver el mapa completo.

4.4.2.3. Proceso de captura de imágenes en la línea

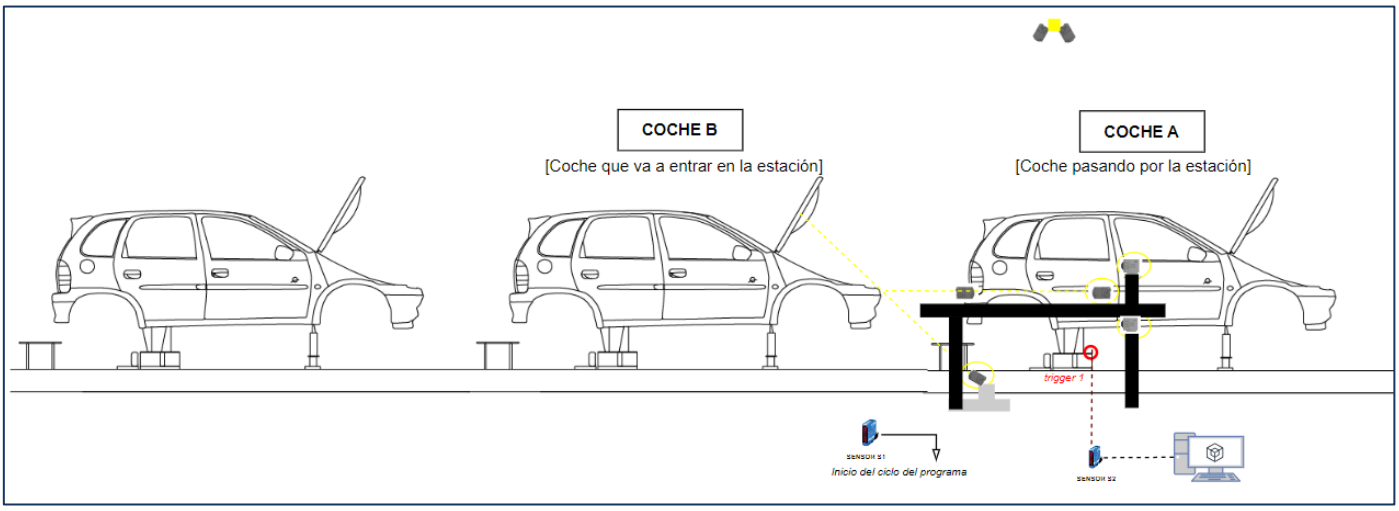


Figura 4. 69: Representación de la toma de imágenes en trigger 1

Fuente: Elaboración Propia con Draw.io

En este Trigger1 las cámaras a pie de línea se encargan de capturar la hoja de montaje del coche que está a punto de entrar en la estación (**coche B**). En el programa de Python encargado de manejar las cámaras cargamos el OCR para que detecte las piezas que el vehículo lleva instaladas, el modelo del coche y el identificador (NOF) del coche.

Leyenda

Modelo

Identificador

Código pieza instalada

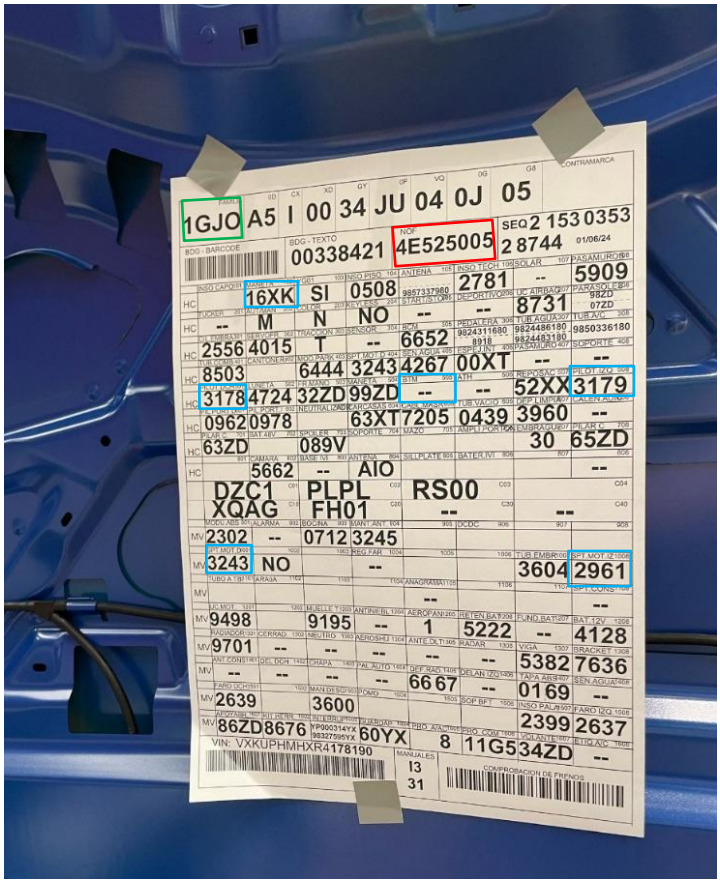


Figura 4. 70: Hoja de montaje de un vehículo

Fuente: Elaboración Propia

COUNT 1

NOF ANTERIOR CONOCIDO

NOF ANTERIOR DESCONOCIDO

Figura 4. 71: Foto lateral superior dcha. del coche

Fuente: Elaboración Propia

Figura 4. 72: Foto frontal izda. del coche

Fuente: Elaboración Propia

Figura 4. 73: Foto lateral inferior dcha. del coche

Fuente: Elaboración Propia

Figura 4. 74: Foto frontal dcha. del coche

Fuente: Elaboración Propia

Figura 4. 75: Foto lateral superior izda. del coche

Fuente: Elaboración Propia

Figura 4. 76: Foto lateral inferior izda. del coche

Fuente: Elaboración Propia

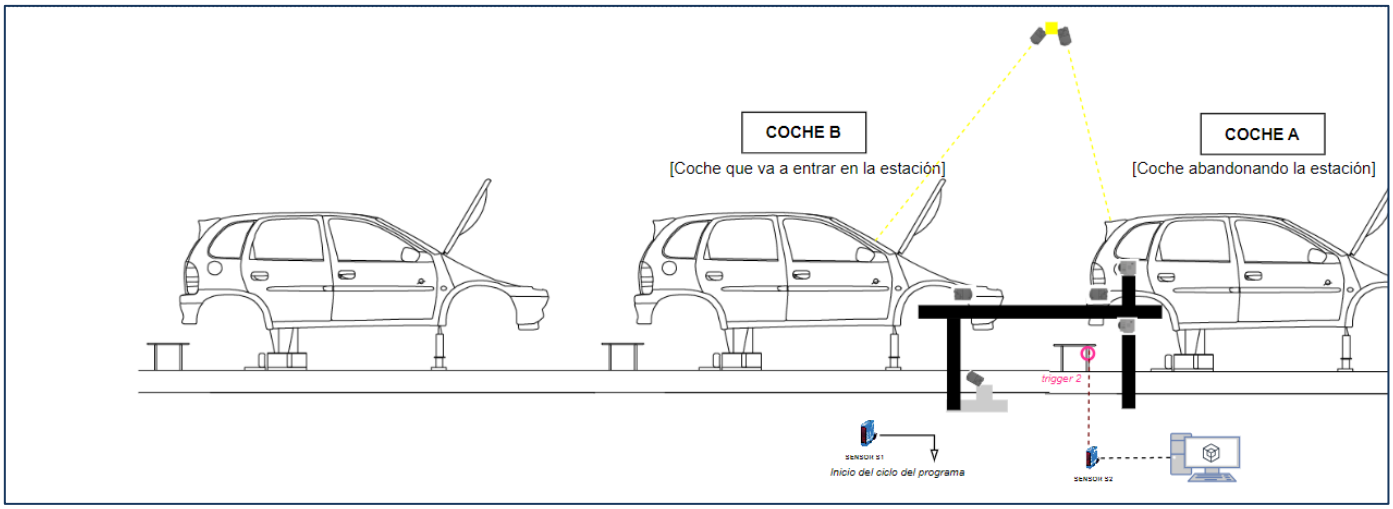


Figura 4. 77: Representación de la toma de imágenes en trigger 2

Fuente: Elaboración Propia con Draw.io

COUNT 2

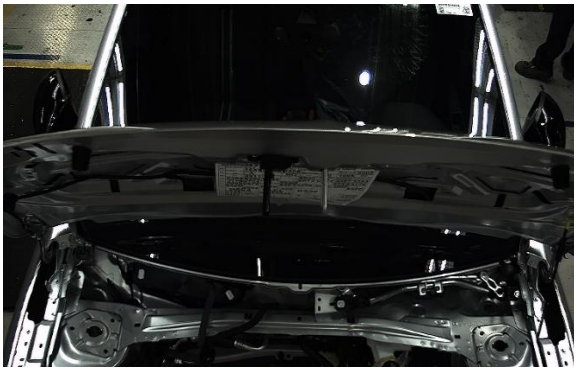


Figura 4. 78: Foto superior delantera

Fuente: Elaboración Propia

Figura 4. 79: Foto superior trasera

Fuente: Elaboración Propia

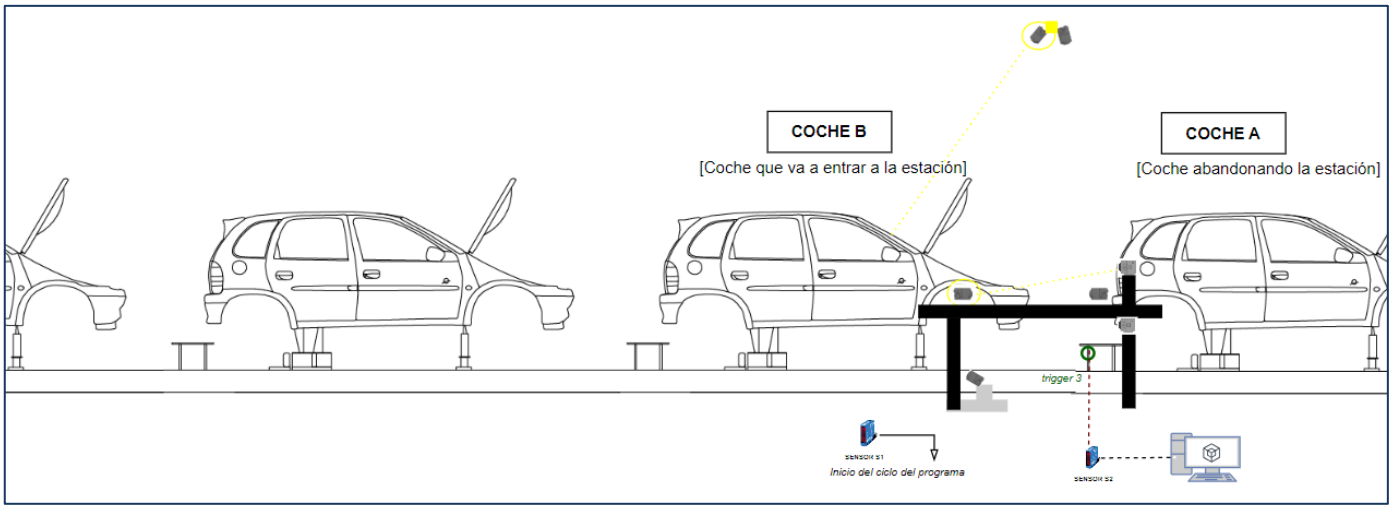


Figura 4. 80: Representación de la toma de imágenes en trigger 3

Fuente: Elaboración Propia

COUNT 3

Figura 4. 81: Foto trasera dcha.

Fuente: Elaboración Propia

Figura 4. 82: Foto luneta superior

Fuente: Elaboración Propia

Figura 4. 83: Foto trasera izda.

Fuente: Elaboración Propia

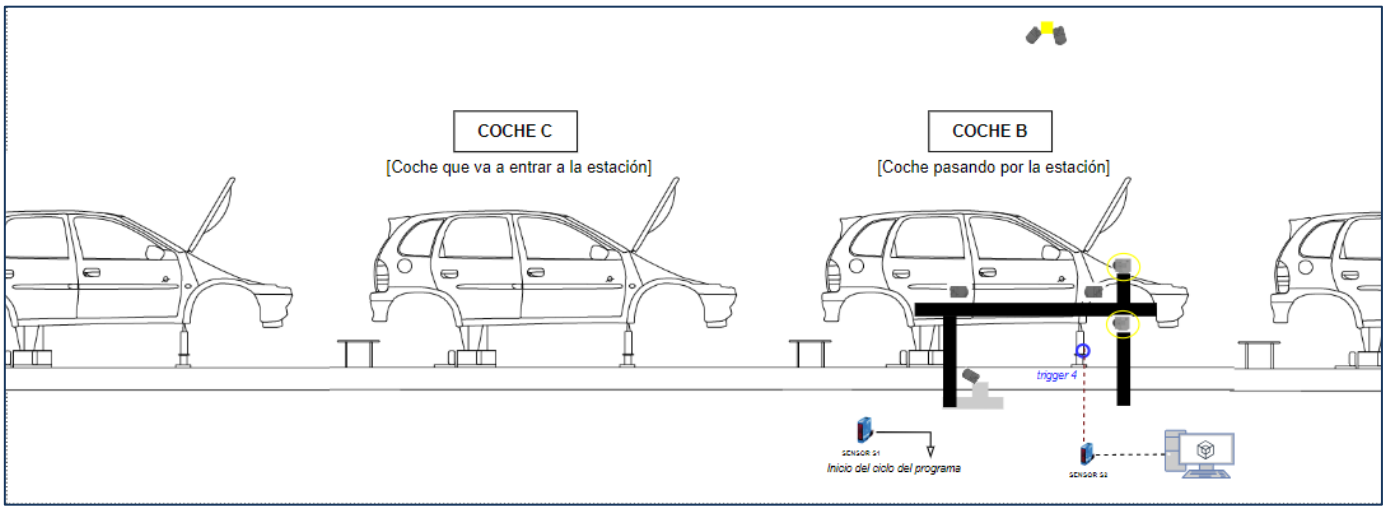


Figura 4. 84: Representación de la toma de imágenes en trigger 4

Fuente: Elaboración Propia con Draw.io

COUNT 4

Figura 4. 85: Foto lateral superior izda.

Fuente: Elaboración Propia

Figura 4. 86: Foto lateral superior dcha.

Fuente: Elaboración Propia

Figura 4. 87: Foto lateral inferior izda.

Fuente: Elaboración Propia

Figura 4. 88: Foto lateral inferior dcha.

Fuente: Elaboración Propia

4.4.3. Tecnologías y librerías utilizadas en la estación

Package	Version
gitdb	4.0.11
GitPython	3.1.41
imageio	2.33.1
imgaug	0.4.0
matplotlib	3.8.2
mpmath	1.3.0
networkx	3.2.1
numpy	1.26.3
nvidia-cublas-cu12	12.1.3.1
nvidia-cuda-cupti-cu12	12.1.105
nvidia-cuda-nvrtc-cu12	12.1.105
nvidia-cuda-runtime-cu12	12.1.105
nvidia-cudnn-cu12	8.9.2.26
nvidia-cufft-cu12	11.0.2.54
nvidia-curand-cu12	10.3.2.106
nvidia-cusolver-cu12	11.4.5.107
nvidia-cusparse-cu12	12.1.0.106
nvidia-nccl-cu12	2.18.1
nvidia-nvjitlink-cu12	12.3.101
nvidia-nvtx-cu12	12.1.105
opencv-contrib-python	4.6.0.66
opencv-python	4.6.0.66
openpyxl	3.1.2
packaging	23.2
paddleocr	2.7.0.3
paddlepaddle-gpu	2.6.0
pandas	2.2.0
pillow	10.2.0
pip	22.0.2
pyparsing	3.1.1
PyQt5	5.15.10
PyQt5-Qt5	5.15.2
PyQt5-sip	12.13.0
pyserial	3.5
python-dateutil	2.8.2
python-docx	1.1.0
PyYAML	6.0.1
spinnaker-python	4.0.0.116
scikit-image	0.22.0
termcolor	2.4.0
thop	0.1.1.post2209072238
tifffile	2023.12.9
torch	2.1.2
torchvision	0.16.2
ultralytics	8.1.4

4.4.4. Descripción de la instalación del software

Para garantizar un acceso eficiente y organizado a los datos y a los programas, utilizamos una estructura de carpetas bien definida. Esta organización facilita la búsqueda de imágenes, la gestión de almacenamiento y el acceso a los archivos del programa más importantes.

```
UBUNTU 22.04 LTS
├── "bin"  # Ejecutables del sistema
├── "boot"  # Archivos de arranque del sistema
├── "dev"  # Archivos de dispositivos
├── "etc"  # Archivos de configuración del sistema
├── "home"
|   ├── "user"
|   |   ├── "Desktop"
|   |   |   ├── "LOGS"                    # Almacena los registros de actividad del programa
|   |   |   |   ├── "Logs_OCR.txt"        # Registros de lectura OCR de hojas de montaje
|   |   |   |   ├── "Logs_NOF.txt"        # Identificadores NOF con fecha y hora
|   |   |   |   ├── "Resultados_red.txt"  # Resultados finales de control por vehículo
|   |   |   ├── "IMG"                     # Consulta de carpetas con fotos de cualquier vehículo por fecha y hora en los últimos 4 días
|   |   |   ├── "Historico_Fotos"         # Consulta de carpetas con fotos de cualquier vehículo por fecha y hora sin límite de fecha
|   |   ├── "Documents"
|   |   |   ├── "Torre"
|   |   |   |   ├── "IMG"
|   |   |   |   |   ├── "Analizadas"               # Imágenes analizadas por la red en tiempo de ejecución
|   |   |   |   |   ├── "Hojas de montaje"         # Hojas de montaje de los últimos 4 días
|   |   |   |   |   ├── "NOK"                      # Contiene carpetas por coche con resultado "NOK"
|   |   |   |   |   ├── "SinAnalizar"              # Imágenes capturadas en tiempo real
|   |   |   |   ├── "Programs"                     # Programas principales en Python y librerías
|   |   |   |   ├── "Resources"                    # Excepciones, configuraciones y JSONs
|   |   |   |   ├── "Scripts"                      # Comandos Linux para terminal
|   |   |   |   ├── "mmultiprocessing_trim_v2.py"  # Punto de entrada del programa
|   |   |   |   ├── "nohup.out"                    # Salida de ejecución con nohup (programa en segundo plano)
|   |   |   |   ├── "NOF_dch.png"                  # Imagen de hoja de montaje en tiempo de ejecución
|   |   |   |   ├── "NOF_izq.png"                  # Imagen de hoja de montaje en tiempo de ejecución
|   |   |   ├── "trim_Xavier_v2.py"               # Archivo Python fuera de la carpeta "Torre" para manejar la JETSON Xavier
|   ├── "Downloads"
|   ├── "Music"
|   ├── "Pictures"
|   ├── "Videos"
```

Figura 4. 89: Estructura de carpetas de la estación

Fuente: Elaboración Propia con un archivo .txt

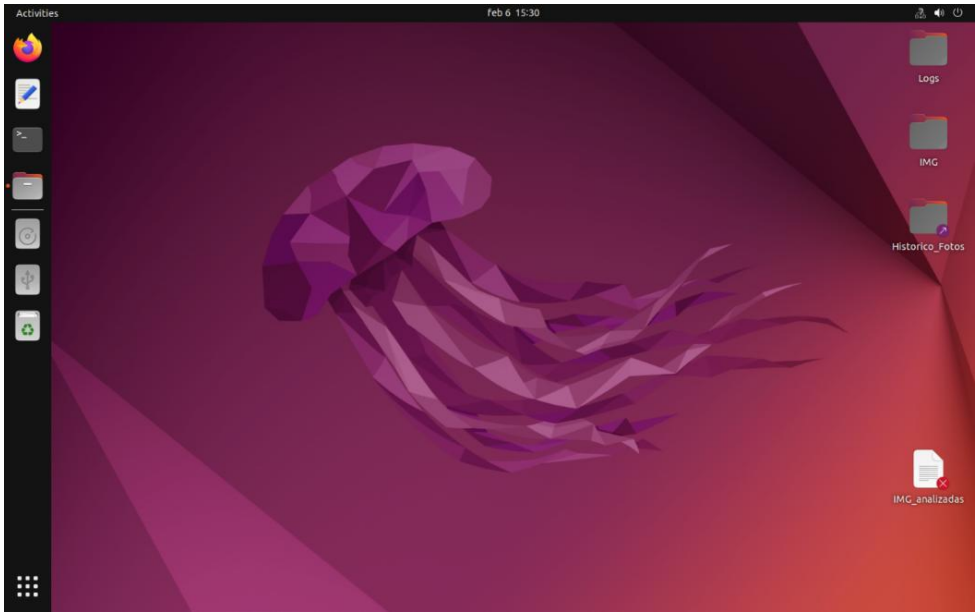


Figura 4. 90: Escritorio del PC de la estación en sistema operativo Ubuntu 22.04

Fuente: Elaboración Propia

Se han añadido capturas de pantalla de las carpetas de las imágenes para mostrar la estructura del contenido de estas carpetas más importantes, y alguna pequeña explicación importante.

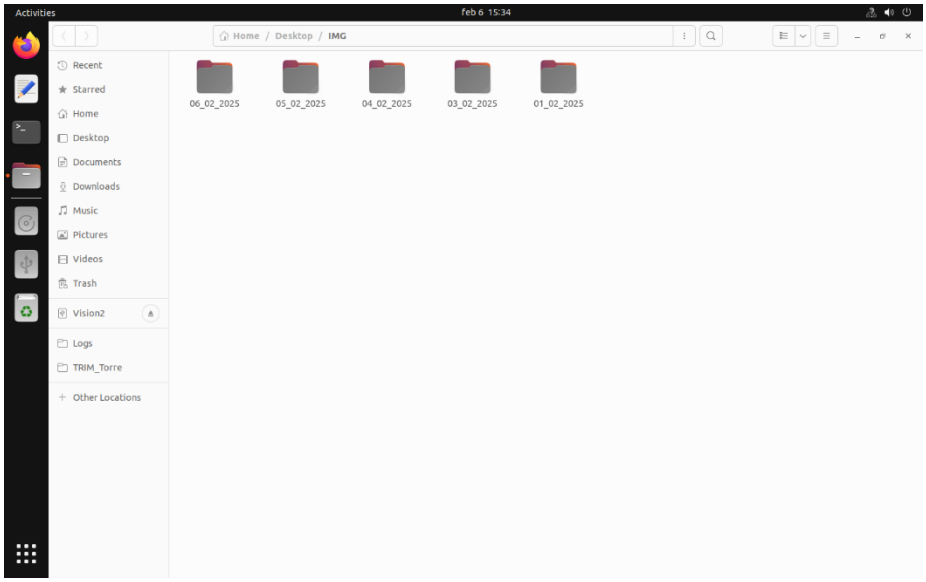


Figura 4. 91: Organización de las carpetas del escritorio, accesibles para consulta de los supervisores de la línea

Fuente: Elaboración Propia

Figura 4. 92: imágenes de todos los vehículos que pasan por la estación, contenidas en las carpetas del escritorio, para consulta de los supervisores de la línea

Fuente: Elaboración Propia

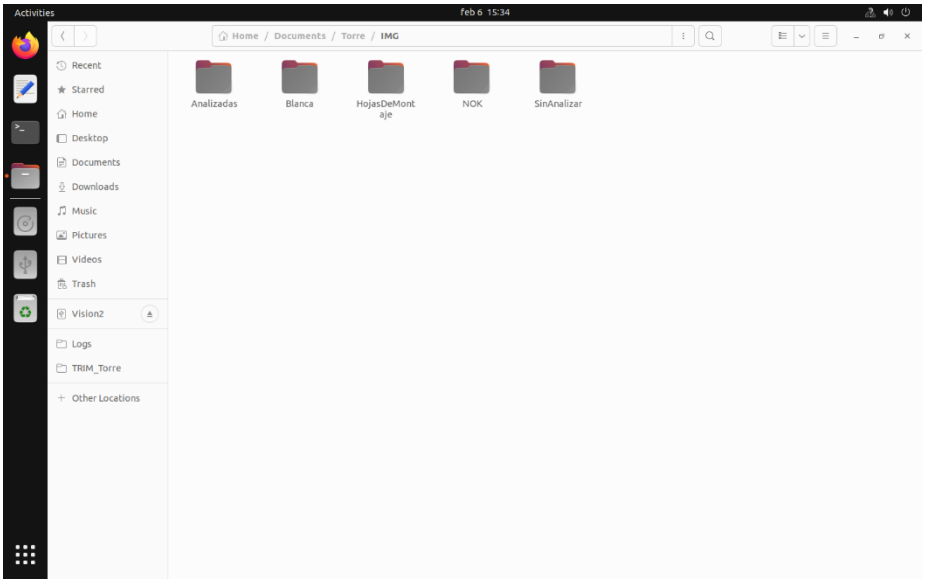


Figura 4. 93: Carpetas para almacenar imágenes que luego la red analizará

Fuente: Elaboración Propia

El programa accede a la carpeta de "SinAnalizar", coge las fotos del coche correspondiente una a una, pasan por YOLO para ser analizadas y finalmente terminan en la carpeta "Analizadas" o en una carpeta independiente dentro de "NOK" si el resultado del análisis no ha tenido éxito y ha detectado un fallo.

4.4.5. Gestión de permisos y usuarios

Durante la implementación del software en la estación, nos dimos cuenta de un problema relacionado con el acceso y la manipulación de archivos dentro del sistema. Tanto los operarios como los supervisores de línea podían interactuar libremente con el ordenador de la estación, lo que generaba riesgos en la seguridad del software.

Los operarios y supervisores podían editar o eliminar archivos esenciales del programa, incluidos scripts en Python (.py), así como fotografías y configuraciones del sistema. Esto, por error, o a propósito, podía causar fallos inesperados y comprometer el funcionamiento de la estación. También tenían acceso sin restricciones a directorios sensibles pues no existía un control diferenciado de permisos, lo que permitía que cualquier usuario manipulara archivos internos del software de la estación.

Ante esta problemática, decidimos implementar una gestión de usuarios con distintos niveles de acceso en el sistema, creando dos tipos de cuentas:

1. Usuario de supervisión y operarios ("supts12")

a. Permisos restringidos: solo lectura y ejecución
b. No pueden modificar, eliminar ni sobrescribir archivos de programa
c. Acceso limitado a ciertas carpetas, permitiéndoles únicamente visualizar imágenes y consultar registros (logs)
d. No pueden ejecutar comandos de administrador (sudo) ni instalar software

2. Usuario administrador ("trimdesktop")
   a. Cuenta con permiso de sudo, lo que le otorga control total sobre el sistema
   b. Puede editar y ejecutar archivos de programa, modificar configuraciones y realizar mantenimiento del software

Esta configuración nos permitió proteger los archivos críticos del sistema, asegurando que supervisores y operarios de la línea solo pudieran acceder a la información necesaria.

4.5. WORKSTATION

La Workstation es el equipo donde se procesan las imágenes capturadas en la estación de visión de la línea de producción. Su función principal es:

1. Etiquetado de un conjunto de datos o dataset, que en nuestro caso son imágenes del vehículo
2. Entrenamiento de modelos de inteligencia artificial con YOLOv5 Ultralytics
3. Validación de los resultados obtenidos

A diferencia de la estación de visión, que captura y analiza las imágenes en tiempo real, la Workstation se centra en mejorar la precisión del sistema mediante el desarrollo y ajuste del modelo de detección de defectos. Su objetivo principal es optimizar el rendimiento del modelo de YOLO.

4.5.1. Proceso de pre-etiquetado

Antes de la implementar la automatización del pre-etiquetado que se explicará a continuación, el proceso de etiquetado era completamente manual. Esto implicaba tomar cada imagen capturada, cargarla en Label Studio, y etiquetarlas a mano una a una, lo cual era un proceso largo y propenso a errores cuando había muchos controles por etiquetar en la imagen.

Con la integración de YOLO para las pre-anotaciones automáticas, optimizamos enormemente este flujo del trabajo para, únicamente, tener que realizar una revisión superficial de las imágenes.

A continuación, se explica cómo funciona y como fueron los pasos para automatizar el proceso actual de etiquetado.

4.5.1.1. Configurar Docker con YOLO

Para empezar, tuvimos que configurar un entorno basado en Docker que ejecutara el modelo YOLO y procesara las predicciones de manera eficiente.

Inicialmente creamos un contenedor Docker con la imagen de YOLO de Ultralytics, cargando un modelo preentrenado llamado yolov5nu.pt. Nos aseguramos de que el contenedor estuviera correctamente configurado para exponer una API, la cual recibiría las imágenes y devolvería las predicciones en forma de cajas de delimitación y etiquetas.

Además, para mejorar el rendimiento del proceso de inferencia, investigamos la posibilidad de ejecutar YOLO sobre GPU, lo cual requirió modificar el archivo docker-compose.yml y realizar cambios en la configuración del entorno de Docker para habilitar el soporte de aceleración por hardware con NVIDIA.

4.5.1.2. Integración de YOLO con Label Studio

Una vez tuvimos el contenedor de YOLO funcionando, el siguiente paso fue integrarlo con Label Studio, la herramienta de anotación utilizada en la Workstation. Para ello, configuramos Label Studio para que utilizara la API del contenedor como su ML Backend (modelo de Machine Learning externo).

Específicamente, en la configuración de Label Studio, establecimos el puerto de comunicación con YOLO (9090 según lo definido en docker-compose.yml) y verificamos que las etiquetas utilizadas en el modelo coincidieran con las etiquetas definidas en nuestro proyecto. Esta configuración permitió que, cada vez que se subía una imagen a Label Studio, esta fuera enviada automáticamente a YOLO para recibir una pre-anotación basada en la detección del modelo.

Para asegurar una conexión estable y fluida, también tuvimos que configurar correctamente la API Key de Label Studio y verificar que la conexión entre ambos sistemas se realizara sin errores.

4.5.1.3. Procedimiento para Generar Pre-Anotaciones en la Workstation

Una vez configurada la integración entre YOLO y Label Studio, el siguiente paso es seguir el procedimiento para generar las pre-anotaciones de manera automática en la Workstation cuando queremos entrenar un modelo de red neuronal. Para ello, seguimos siempre la siguiente secuencia de pasos:

1. Inicio del sistema
    - Encender la Workstation
    - Abrir Docker Desktop para gestionar los contenedores
2. Configuración del contenedor YOLO
    - Acceder a la ruta donde se encuentra el backend del modelo YOLO en Label Studio, que en nuestro caso es C:\Users\ALDLWS\label-studio-ml-backend\label-studio-ml\examples\yolo
    - Verificar que en el archivo docker-compose.yml se está utilizando el modelo correcto de red neuronal que queremos utilizar (pesos adecuados)
    - (opc.) Si es necesario, cambiar la ruta de la red neuronal para asegurarse de que apunta al archivo de pesos correctos (best.pt)
    - Ejecutar el siguiente comando en la terminal de Docker Desktop para iniciar el contenedor con YOLO como backend de Label Studio:

      ```
      docker compose up
      ```

      Esto lanza el servicio que realizará la pre-anotación automática de las imágenes en Label Studio.
3. Inicio de Label Studio
    - En la terminal, activar el entorno virtual donde tenemos instalado Label Studio con Conda mediante el comando:

      ```
      conda activate labelstudio
      ```

    - Iniciar Label Studio mediante el comando:

      ```
      label-studio
      ```

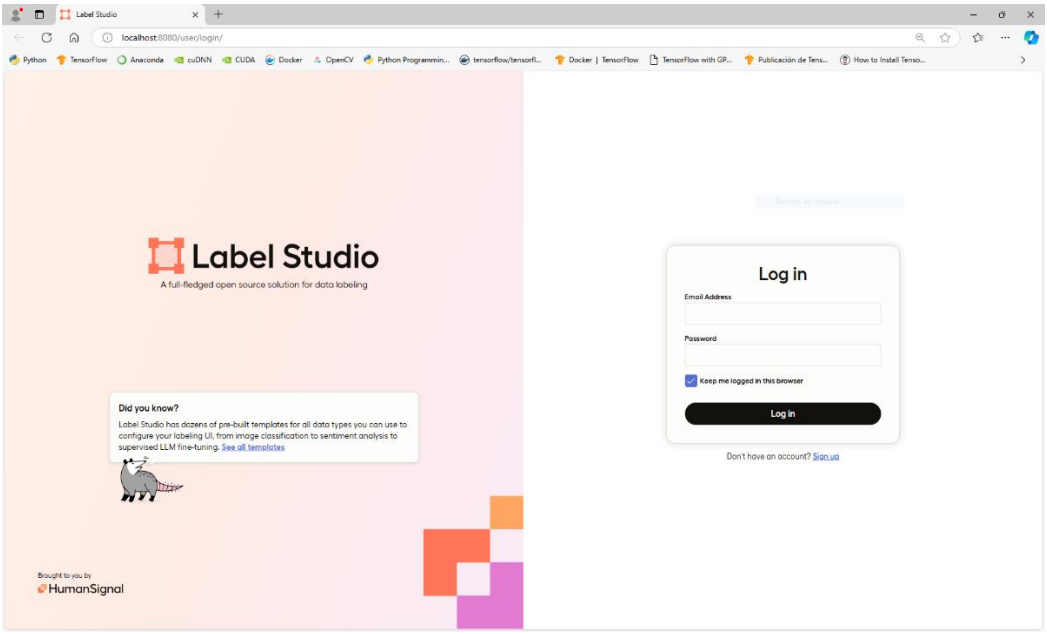


Figura 4. 94: Página de inicio de Label Studio

Fuente: Elaboración Propia

- Ingresar usuario y contraseña
- Acceder a nuestro proyecto, en este caso TRIM

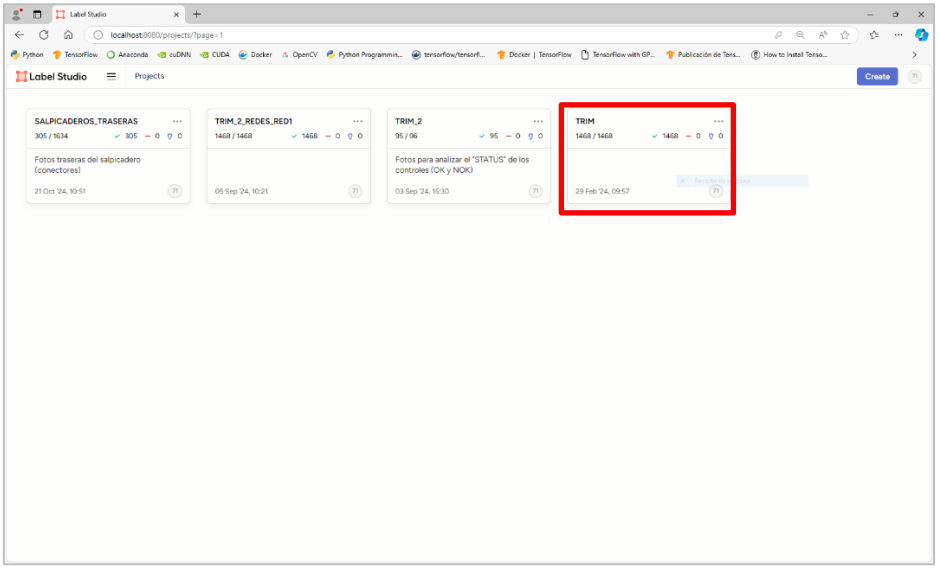


Figura 4. 95: Gestión de los proyectos con Label Studio

Fuente: Elaboración Propia

- Verificar que Label Studio se ha conectado correctamente con el backend de YOLO, se puede comprobar en **Settings > Models** dentro de la interfaz de Label Studio

4. Carga y sincronización de imágenes
    - Copiar las imágenes que queremos pre-anotar en la ruta de Cloud Storage del proyecto en Label Studio, en nuestro caso, esas imágenes se almacenan en una carpeta predefinida en Desktop/prueba_yolo_etiquetado/yolov5-master/data/TRIM
    - Dentro de Label Studio, ir a **Project > Settings > Cloud Storage** y hacer click en **Sync Storage** para sincronizar las imágenes

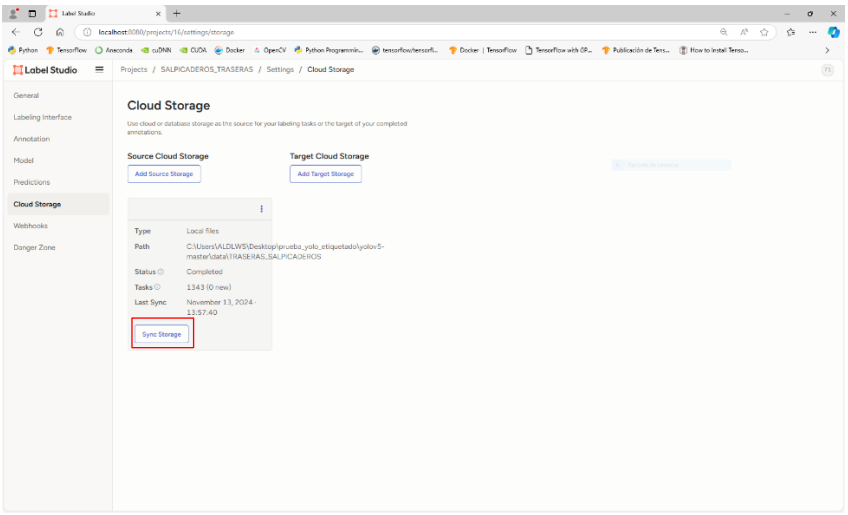


Figura 4. 96: Sincronización de imágenes para posterior etiquetado en Label Studio

Fuente: Elaboración Propia

- Filtrar las nuevas imágenes para visualizar únicamente las recién añadidas

5. Generación de Pre-Anotaciones
    - Al abrir una imagen en Label Studio, las pre-anotaciones aparecerán automáticamente basadas en las predicciones de modelo YOLO que hemos configurado en Docker
    - Revisar las pre-anotaciones y, en caso necesario, corregir etiquetas o ajustar las cajas de delimitación manualmente

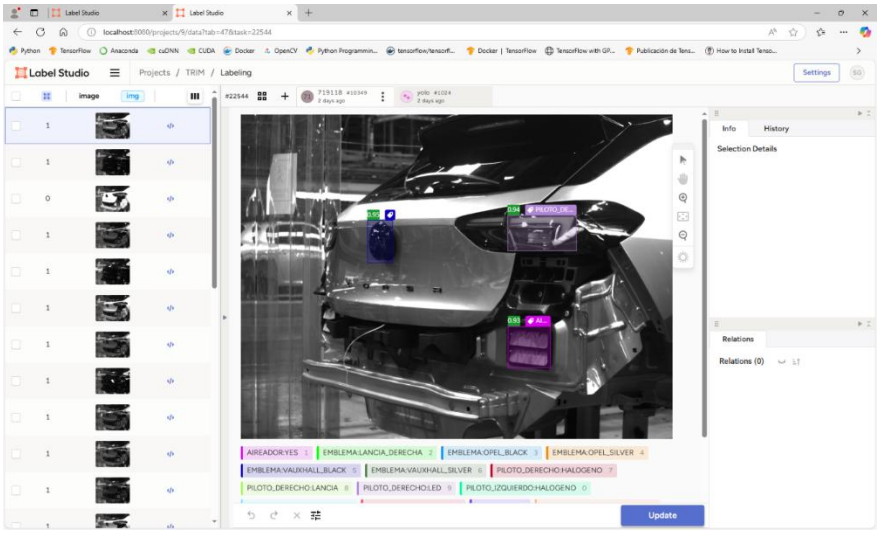


Figura 4. 97: Etiquetado de imágenes con Label Studio

Fuente: Elaboración Propia

4.5.2. Entrenamiento del modelo

4.5.2.1. Exportación del Dataset desde Label Studio

Una vez que las imágenes han sido etiquetadas en Label Studio, el siguiente paso es exportar los datos en un formato compatible con YOLO para poder utilizarlos en el entrenamiento del modelo. Para ello, seguimos el siguiente proceso:

1. En Label Studio, hacemos click en **"Export"** y seleccionamos el formato YOLO

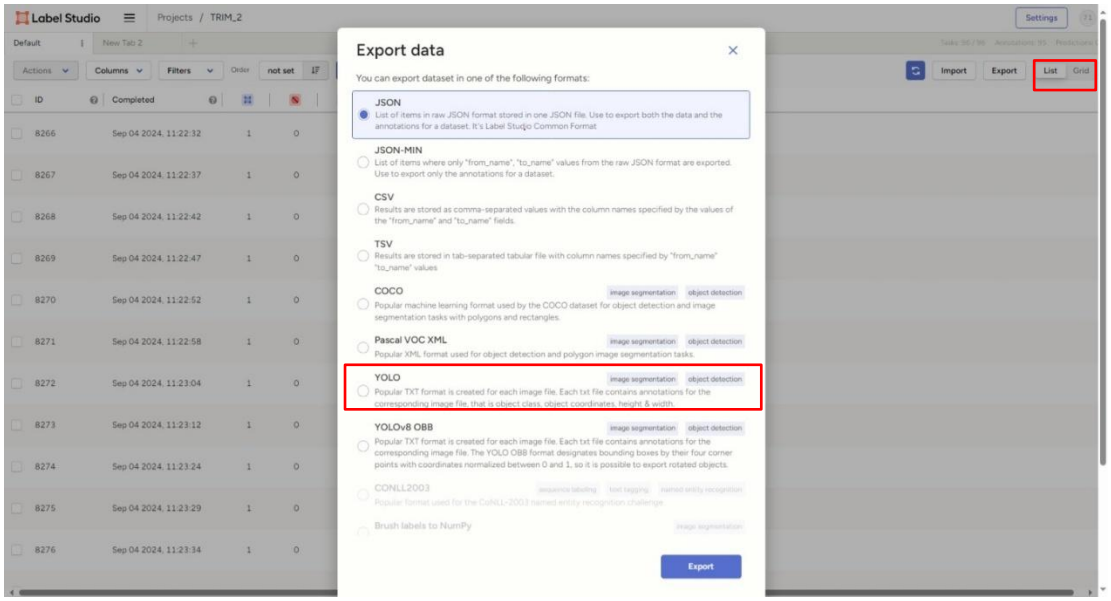


Figura 4. 98: Exportación del Dataset

Fuente: Elaboración Propia

2. Esto genera un archivo **ZIP** que contiene el dataset con la siguiente estructura:
   a. Labels: Archivos .txt que tienen el mismo nombre que la imagen correspondiente, cada archivo de etiquetas contiene:
      i. Coordenadas de las cajas de delimitación (bounding boxes)
      ii. La clase del objeto (representada por un número entero)
   b. Images: Carpeta con las imágenes originales utilizadas en el etiquetado
   c. notes.json: Archivo con metadatos adicionales
   d. class.txt: Contiene las clases del dataset
3. Extraemos este ZIP en una carpeta predefinida donde almacenamos los datasets de cada proyecto, en nuestro caso esa ruta es C:\Users\ALDLWS\label-studio-ml-backend\label-studio-ml\examples\yolo\data\entrenamiento\DATASETS\TRIM

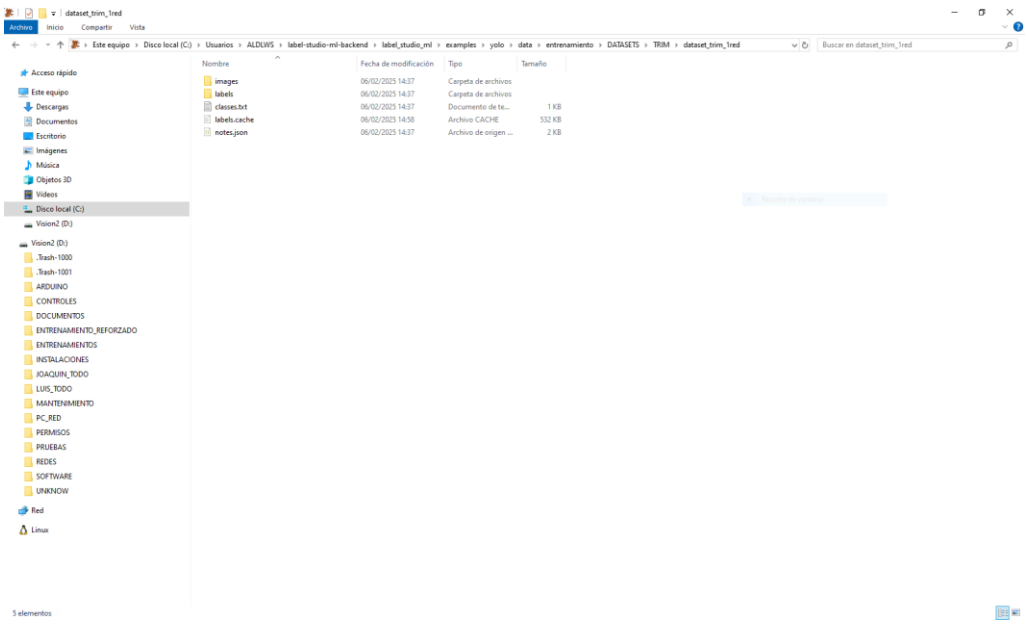


Figura 4. 99: Contenido de un Dataset exportado

Fuente: Elaboración Propia

4.5.2.2. Configuración del entorno de entrenamiento (Ultralytics)

Inicialmente, utilizábamos YOLOv5 de GitHub para entrenar nuestros modelos de detección de objetos. Sin embargo, al integrar la pre-anotacion automática en nuestro flujo de trabajo con Label Studio y Docker, nos encontramos un problema de compatibilidad.

Figura 4. 100: YOLOv5 de GitHub

Fuente: Elaboración Propia

El backend de Machine Learning que configuramos en Docker para realizas las pre-anotaciones automáticas utiliza la librería Ultralytics, que ha desarrollado su propia versión de YOLOv5 conocida como YOLOv5u.

Esto generaba conflicto porque los pesos generados con YOLOv5 (GitHub) no eran directamente compatibles con el entorno de Ultralytics en Docker y los modelos entrenamientos con YOLOv5 tradicional no podían ser utilizados para la pre-anotación en Label Studio. En definitiva, queríamos hacer pre-anotaciones automáticas con YOLO en docker así que tomamos la decisión de abandonar el entrenamiento con YOLOv5 de GitHub y migrar a YOLOv5u de Ultralytics.

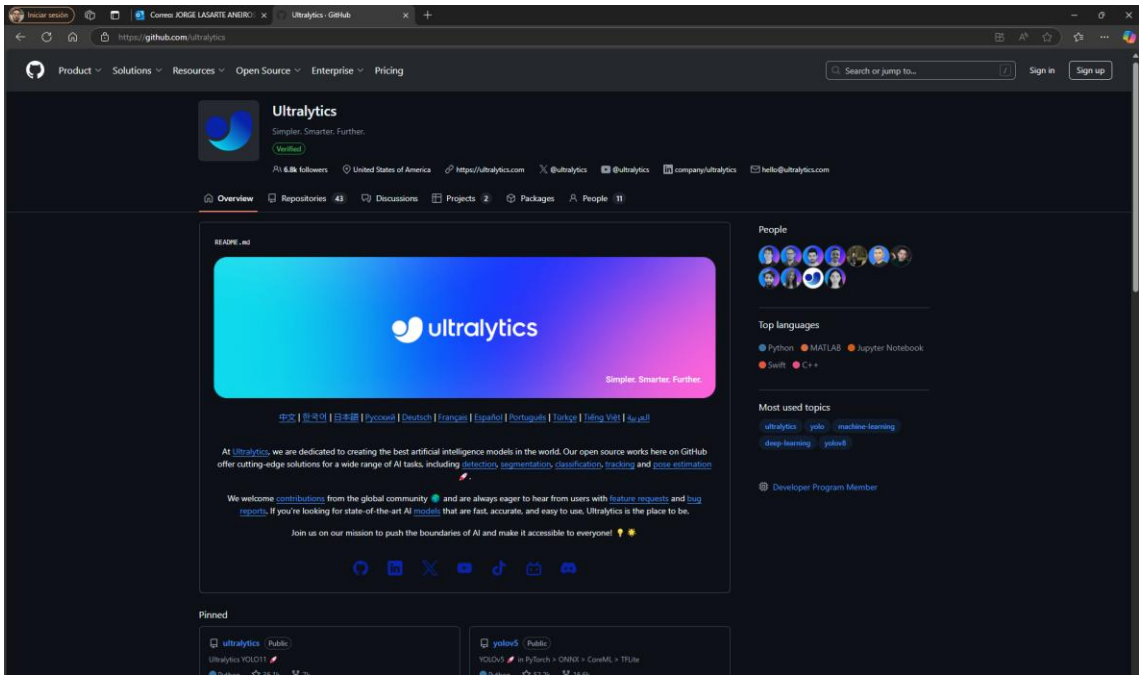


Figura 4. 101: YOLOv5u de Ultralytics

Fuente: Elaboración Propia

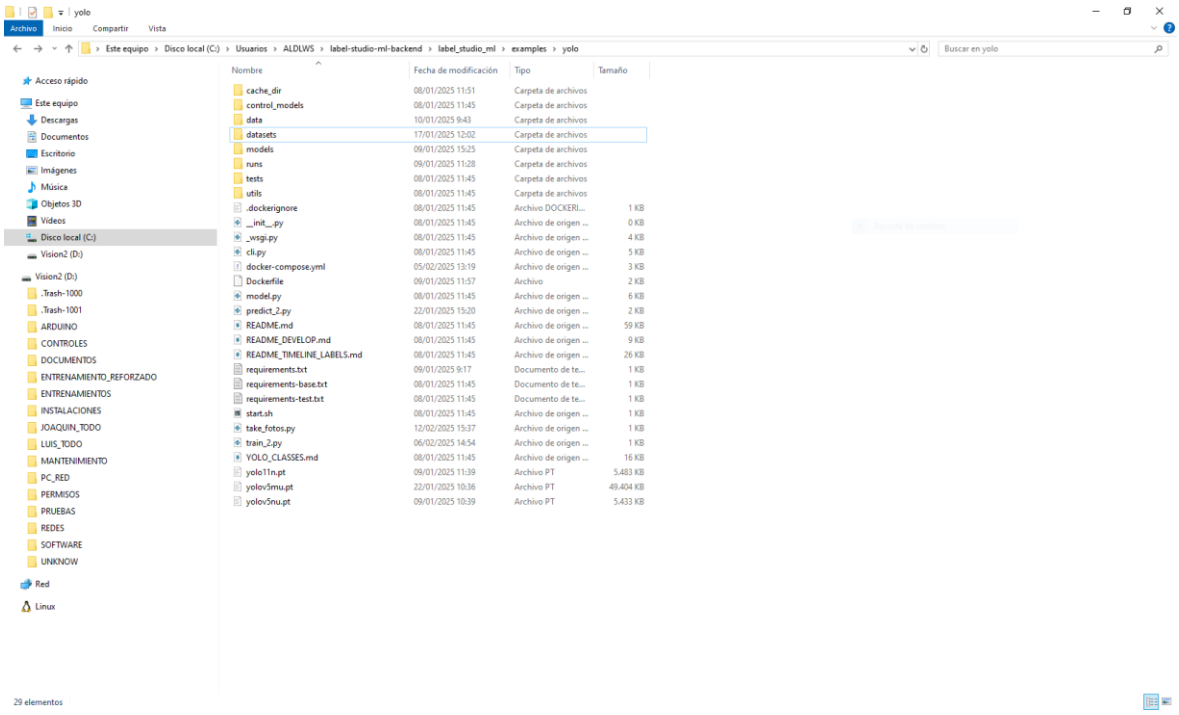


Figura 4. 102: Docker con YOLOv5u de Ultralytics

Fuente: Elaboración Propia

Para ello, creamos un entorno virtual específico de Ultralytics con las librerías necesarias para realizar el entrenamiento, que activamos antes de entrenar.

Figura 4. 103: Entorno virtual para trabajar con YOLOv5u Ultralytics

Fuente: Elaboración Propia

4.5.2.3. Configuración del archivo de entrenamiento (train_2.py)

Antes de iniciar el entrenamiento, es necesario modificar el archivo train_2.py, en nuestro caso ubicado en C:\Users\ALDLWS\label-studio-ml-backend\label-studio-ml\examples\yolo\train_2.py. Las modificaciones clave en este archivo son:

1. **Definir la ruta del dataset en el archivo YAML (Yet Another Markup Language)**

    a. Ubicado en nuestro caso en C:\Users\ALDLWS\label-studio-ml-backend\label-studio-ml\examples\yolo\data\entrenamiento\YAMLS\TRIM , se utiliza para configurar el dataset. Este archivo le indica al modelo:

        i. Dónde están las imágenes de entrenamiento y validación

        ii. Cuántas clases hay en el dataset y cuáles son

    b. Dentro del YAML, se especifica la ubicación de las imágenes y las etiquetas

```
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
# COCO128 dataset https://www.kaggle.com/ultralytics/coco128 (first 128 images from COCO train2017) by Ultralytics
# Example usage: python train.py --data coco128.yaml
# parent
# ├── yolov5
# └── datasets
#     └── coco128  ← downloads here (7 MB)

# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
path: C:\Users\ALDLWS\label-studio-ml-backend\label_studio_ml\examples\yolo\data\entrenamiento\DATASETS\TRIM\dataset_trim_1red  # dataset root dir

train: images  # train images (relative to 'path') 128 images
val:   images # val images (relative to 'path') 128 images
test:  # test images (optional)

# Classes
names:
  0: AIREADOR:YES
  1: EMBLEMA:LANCIA_DERECHA
  2: EMBLEMA:OPEL_BLACK
  3: EMBLEMA:OPEL_SILVER
  4: EMBLEMA:VAUXHALL_BLACK
  5: EMBLEMA:VAUXHALL_SILVER
  6: PILOTO_DERECHO:HALOGENO
  7: PILOTO_DERECHO:LANCIA
  8: PILOTO_DERECHO:LED
  9: PILOTO_IZQUIERDO:HALOGENO
  10: PILOTO_IZQUIERDO:LANCIA
  11: PILOTO_IZQUIERDO:LED
  12: SIRENA:YES
  13: SOPORTE_MOTOR_DERECHO:1
  14: SOPORTE_MOTOR_DERECHO:2
  15: SOPORTE_MOTOR_DERECHO:3
  16: SOPORTE_MOTOR_DERECHO:4
  17: SOPORTE_MOTOR_DERECHO:5
  18: SOPORTE_MOTOR_IZQUIERDO:1
  19: SOPORTE_MOTOR_IZQUIERDO:2
  20: SOPORTE_MOTOR_IZQUIERDO:3
  21: SOPORTE_MOTOR_DERECHO:6
  22: SOPORTE_MOTOR_DERECHO:7
  23: SOPORTE_MOTOR_IZQUIERDO:4
```

RUTA DATASET

Figura 4. 104: Archivo YAML para configurar el Dataset

Fuente: Elaboración Propia

    c. Si se añaden nuevas etiquetas en Label Studio, es obligatorio actualizar este YAML con las nuevas etiquetas

2. **Especificar el modelo de inicio del entrenamiento y la ruta del YAML configurado en el paso anterior:**

    a. Se puede entrenar desde cero o cargar pesos preentrenados

    b. Para entrenar desde una versión preentrenada, en train_2.py se configura:

```python
from ultralytics import YOLO
import os

if __name__ == '__main__':
    # Comprobar directorio de trabajo
    print("Current working directory:", os.getcwd())
    os.chdir("C:/Users/ALDLWS/label-studio-ml-backend/label_studio_ml/examples/yolo")

    # Verificar existencia del archivo YAML
    yaml_path = "C:/Users/ALDLWS/label-studio-ml-backend/label_studio_ml/examples/yolo/data/entrenamiento/YAMLS/TRIM.yaml"
    if not os.path.exists(yaml_path):
        raise FileNotFoundError(f"Archivo YAML no encontrado: {yaml_path}")

    # Cargar el modelo
    model = YOLO("C:/Users/ALDLWS/label-studio-ml-backend/label_studio_ml/examples/yolo/runs/detect/trim/weights/best.pt")

    # Entrenar el modelo
    train_results = model.train(
        data=yaml_path,  # Ruta absoluta al YAML
        epochs=5,       # Número de épocas
        imgsz=640,       # Tamaño de las imágenes
        device=0,        # GPU o CPU
    )

    # Evaluar el modelo
    metrics = model.val()
```

RUTA YAML

MODELO DE ENTRENAMIENTO

Figura 4. 105: Archivo train_2.py

Fuente: Elaboración Propia

c. También se puede reentrenar usando pesos de un modelo previamente entrenado

3. Configurar parámetros de entrenamiento:

```python
# Entrenar el modelo
train_results = model.train(
    data=yaml_path,  # Ruta absoluta al YAML
    epochs=5,       # Número de épocas
    imgsz=640,       # Tamaño de las imágenes
    device=0,        # GPU o CPU
)
```

Figura 4. 106: Configuración de los parámetros de entrenamiento del modelo, configurables en el archivo train_2.py

Fuente: Elaboración Propia

a. Batch Size (batch):

i. Define cuantas imágenes se procesan simultáneamente

ii. Un valor de 32 significa que el modelo procesará 32 imágenes a la vez antes de actualizar sus pesos

iii. Por ejemplo, si hay 1500 imágenes y el batch es de 32, cada época recorrerá 1500/32 = 47 lotes de imágenes

b. Épocas (epochs):

   i. Define cuántas veces el modelo recorrerá todo el dataset

   ii. Si configuramos 50 épocas, el modelo verá cada imagen del dataset 50 veces

c. Backpropagation y Ajuste de Pesos:

   i. Después de cada batch, el modelo calcula el loss (error) y ajusta los pesos mediante backpropagation

   ii. A lo largo de las epochs, el modelo optimiza sus pesos para reducir el error en la detección

4. **Ejecución del entrenamiento**

a. Una vez el entorno está activado y hemos ajustado los parámetros que queremos para nuestro modelo, ejecutamos el entrenamiento

```
(ultralytics_train) C:\Users\ALDLWS\label-studio-ml-backend\label_studio_ml\examples\yolo>python train_2.py
```

```
(ultralytics_train) C:\Users\ALDLWS\label-studio-ml-backend\label_studio_ml\examples\yolo>python train_2.py
Current working directory: C:\Users\ALDLWS\label-studio-ml-backend\label_studio_ml\examples\yolo
Ultralytics 8.3.58 🚀 Python-3.10.16 torch-2.3.0+cu121 CUDA:0 (NVIDIA GeForce RTX 3080 Ti, 12288MiB)
engine\trainer: task=detect, mode=train, model=C:/Users/ALDLWS/label-studio-ml-backend/label_studio_ml/examples/yolo/runs/detect/trim/weights/best.pt, data=C:/Users/ALDLWS/label-studio-ml-backend/label_studio_ml/examples/yolo/data/entrenamiento/YAMLS/TRIM.yaml, epochs=50, time=None, patience=100, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=0, workers=8, project=None, name=train26, exist_ok=False, pretrained=True, optimizer=auto, verbose=True, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.0, profile=False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True, split=val, save_json=False, save_hybrid=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None, vid_stride=1, stream_buffer=False, visualize=False, augment=False, agnostic_nms=False, classes=None, retina_masks=False, embed=None, show=False, save_frames=False, save_txt=False, save_conf=False, save_crop=False, show_labels=True, show_conf=True, show_boxes=True, line_width=None, format=torchscript, keras=False, optimize=False, int8=False, dynamic=False, simplify=True, opset=None, workspace=None, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.0, bgr=0.0, mosaic=1.0, mixup=0.0, copy_paste=0.0, copy_paste_mode=flip, auto_augment=randaugment, erasing=0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml, save_dir=runs\detect\train26

                   from  n    params  module                                       arguments
  0                  -1  1      5280  ultralytics.nn.modules.conv.Conv             [3, 48, 6, 2, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv             [48, 96, 3, 2]
  2                  -1  2     65280  ultralytics.nn.modules.block.C3              [96, 96, 2]
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv             [96, 192, 3, 2]
  4                  -1  4    444672  ultralytics.nn.modules.block.C3              [192, 192, 4]
  5                  -1  1    664320  ultralytics.nn.modules.conv.Conv             [192, 384, 3, 2]
  6                  -1  6   2512896  ultralytics.nn.modules.block.C3              [384, 384, 6]
  7                  -1  1   2655744  ultralytics.nn.modules.conv.Conv             [384, 768, 3, 2]
  8                  -1  2   4134912  ultralytics.nn.modules.block.C3              [768, 768, 2]
  9                  -1  1   1476864  ultralytics.nn.modules.block.SPPF            [768, 768, 5]
 10                  -1  1    295680  ultralytics.nn.modules.conv.Conv             [768, 384, 1, 1]
 11                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 12             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 13                  -1  2   1182720  ultralytics.nn.modules.block.C3              [768, 384, 2, False]
 14                  -1  1     74112  ultralytics.nn.modules.conv.Conv             [384, 192, 1, 1]
 15                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 16             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 17                  -1  2    296448  ultralytics.nn.modules.block.C3              [384, 192, 2, False]
 18                  -1  1    332160  ultralytics.nn.modules.conv.Conv             [192, 192, 3, 2]
 19            [-1, 14]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 20                  -1  2   1035264  ultralytics.nn.modules.block.C3              [384, 384, 2, False]
 21                  -1  1   1327872  ultralytics.nn.modules.conv.Conv             [384, 384, 3, 2]
 22            [-1, 10]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 23                  -1  2   4134912  ultralytics.nn.modules.block.C3              [768, 768, 2, False]
 24        [17, 20, 23]  1   4231960  ultralytics.nn.modules.head.Detect           [24, [192, 384, 768]]
YOLOv5m summary: 339 layers, 25,079,032 parameters, 25,079,016 gradients, 64.4 GFLOPs

Transferred 559/559 items from pretrained weights
Freezing layer 'model.24.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
AMP: checks passed ✅
train: Scanning C:\Users\ALDLWS\label-studio-ml-backend\label_studio_ml\examples\yolo\data\entrenamiento\DATASETS\TRIM\dataset_trim_1red\labels.cache... 1503 images, 2 backgrounds, 0 corrupt: 100%|
```

Figura 4. 107: Comienzo del entrenamiento

Fuente: Elaboración Propia

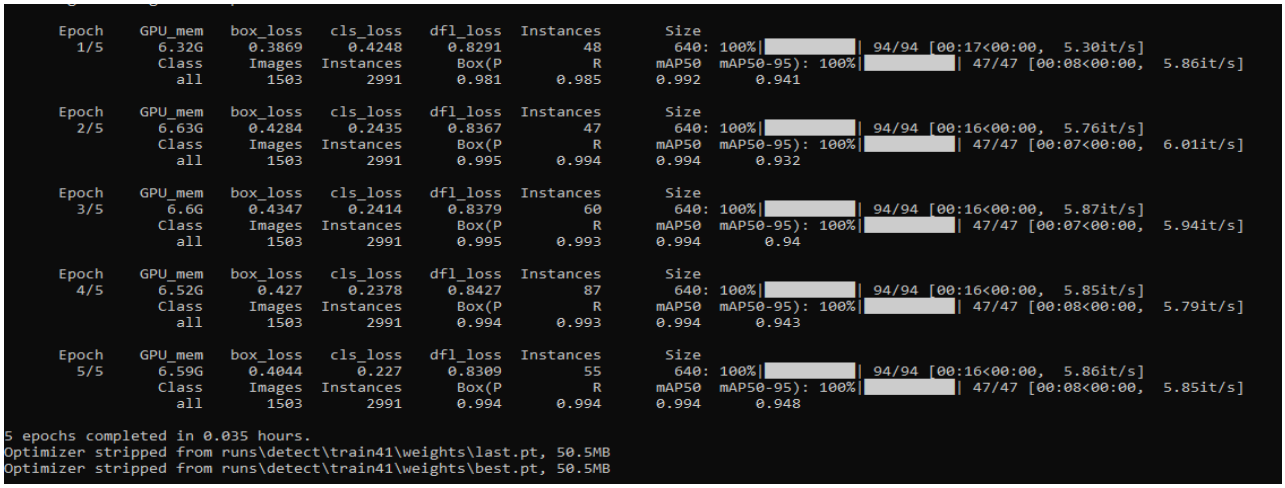


Figura 4. 108: Entrenamiento del modelo con YOLOv5u de Ultralytics

Fuente: Elaboración Propia

4.5.3. Evaluación de resultados del modelo de YOLO

Una vez el modelo ha sido entrenado, es fundamental analizar los resultados obtenidos para determinar su rendimiento y detectar posibles mejoras. En nuestro caso, estos resultados se almacenan en la carpeta **"trainX"**, ubicada en C:\Users\ALDL\LABEL-STUDIO-ML-BACKEND\label_studio_ml\examples\yolo\runs\detects.

Aquí encontramos diversas métricas y gráficos que nos ayudan a interpretar la calidad del modelo. En los siguientes apartados se van a explicar las más importantes, y en el apartado final se hará una pequeña conclusión de los posibles problemas de un modelo mal entrenado y sus soluciones.

4.5.3.1. results.png

- Contiene gráficos de perdida y métricas de precisión/recall a lo largo de las épocas
- Se espera que las pérdidas bajen y que precisión/recall/mAP aumenten

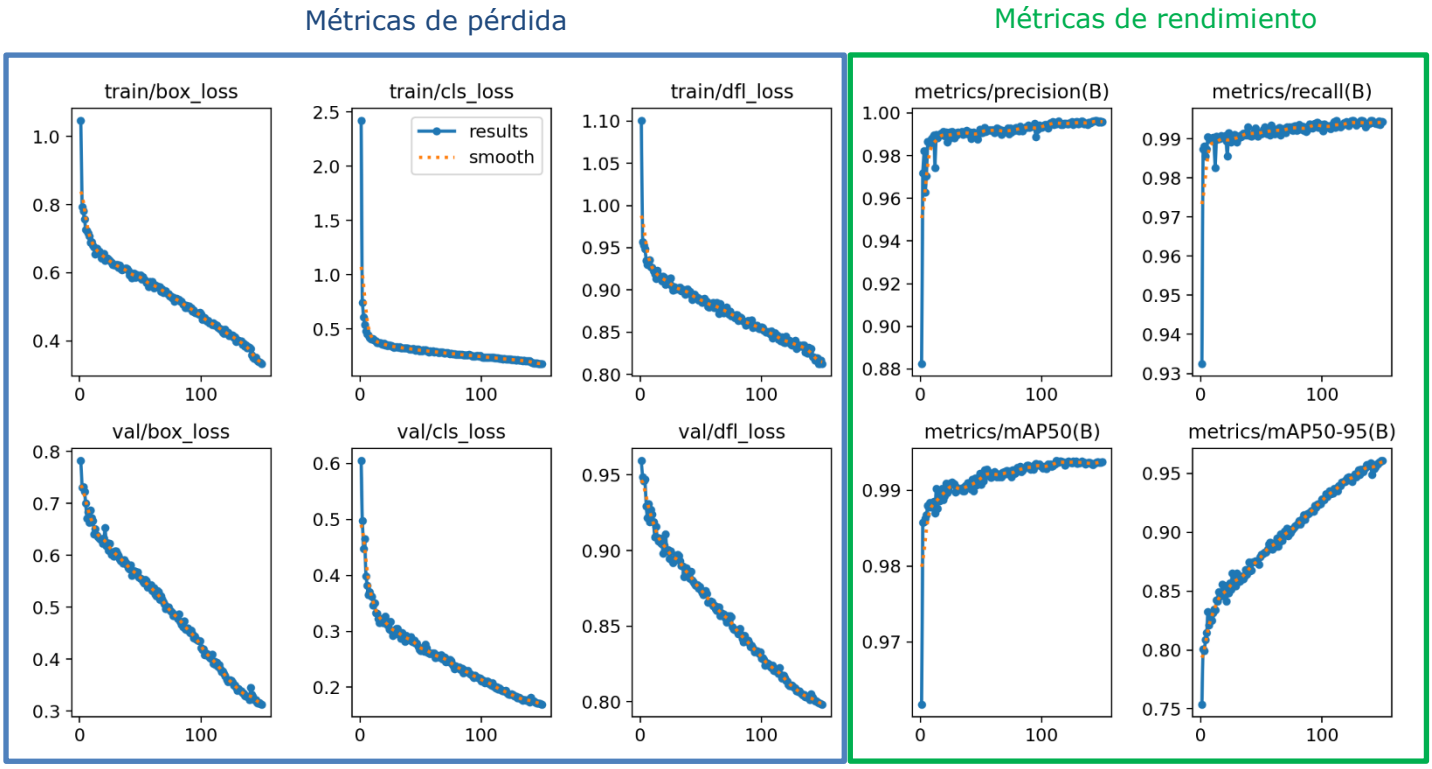


Figura 4. 109: Gráficos contenidos en results.png después del entrenamiento

Fuente: Elaboración Propia

Aquí tenemos que agrupar las métricas más relevantes en dos grandes categorías:

1. Métricas de pérdida (Loss)
2. Métricas de rendimiento (Precisión, Recall, mAP)

Métricas de pérdida (Loss Functions)

Las funciones de pérdida indican qué tan bien está aprendiendo el modelo durante el entrenamiento y la validación, matemáticamente, intentan minimizar la función de error. Un valor alto significa que el modelo se está equivocando mucho, mientras que un valor bajo indica que está aprendiendo correctamente.

1. train/box_loss & val/box_loss
    a. Indican el error en la predicción de las cajas de detección (bounding boxes)
    b. Un valor alto significa que las cajas están mal ubicadas

c. Debe disminuir progresivamente durante el entrenamiento

2. train/cls_loss & val/cls_loss
    a. Representan el error en la clasificación de los objetos dentro de las cajas
    b. Si el valor es alto, el modelo está confundiendo clases
    c. Se espera que este valor baje con más entrenamiento y mejores datos
3. train/dfl_loss & val/dfl_loss
    a. Relacionado con la regresión de coordenadas de las cajas
    b. Un DFL alto indica que el modelo tiene dificultades para predecir los bordes exactos

Métricas de rendimiento (evaluación de la precisión del modelo)

Estas métricas nos indican qué tan bien el modelo detecta y clasifica los objetos.

1. metrics/precisión(B)
    a. Mide qué tan preciso es el modelo en sus predicciones
    b. Si es alto, significa que cuando el modelo dice que un objeto está presente, casi siempre es correcto
    c. Un valor bajo indica muchos falsos positivos
2. metrics/recall(B)
    a. Mide qué tan bien el modelo detecta todos los objetos reales
    b. Un valor bajo significa que el modelo se está perdiendo muchos objetos (falsos negativos)
    c. Idealmente, se busca un equilibrio entre precisión y recall
3. metrics/mAP50(B) (Mean Average Precision al 50%)
    a. Mide la precisión global del modelo usando un umbral del 50% de IoU (*Intersection over Union*)
    b. Un valor alto indica que el modelo predice bien las cajas de detección
4. metrics/mAP50-95(B)
    a. Similar a mAP50, pero calcula el promedio en distintos umbrales de IoU (50% a 95%)
    b. Es la métrica más exigente que evalúa qué tan bien se ajustan las predicciones a los objetos reales

Métrica fundamental

En definitiva, y como resumen de las comparaciones más importantes para evaluar las métricas:

- Si la precisión es alta, pero el recall es bajo, el modelo es **muy estricto** (no detecta algunos objetos)
- Si el recall es alto pero la precisión es baja, el modelo es **muy permisivo** (detecta demasiadas cosas incorrectamente)
- Si el mAP50 y mAP50-95 son bajos, significa que el modelo no está prediciendo correctamente las cajas de los objetos

4.5.3.2. F1 curve.png

Muestra la relación entre Precisión y Sensibilidad (Recall) en diferentes umbrales de confianza del modelo.

- Cuanto más arriba esté la curva, mejor rendimiento del modelo
- Si la curva está cerca del 1, el modelo tiene buen balance entre precisión y recall y es capaz de detectar objetos correctamente y sin muchos falsos positivos
- Si la curva es baja o irregular, tiene un mal rendimiento global y el modelo puede ser inconsistente

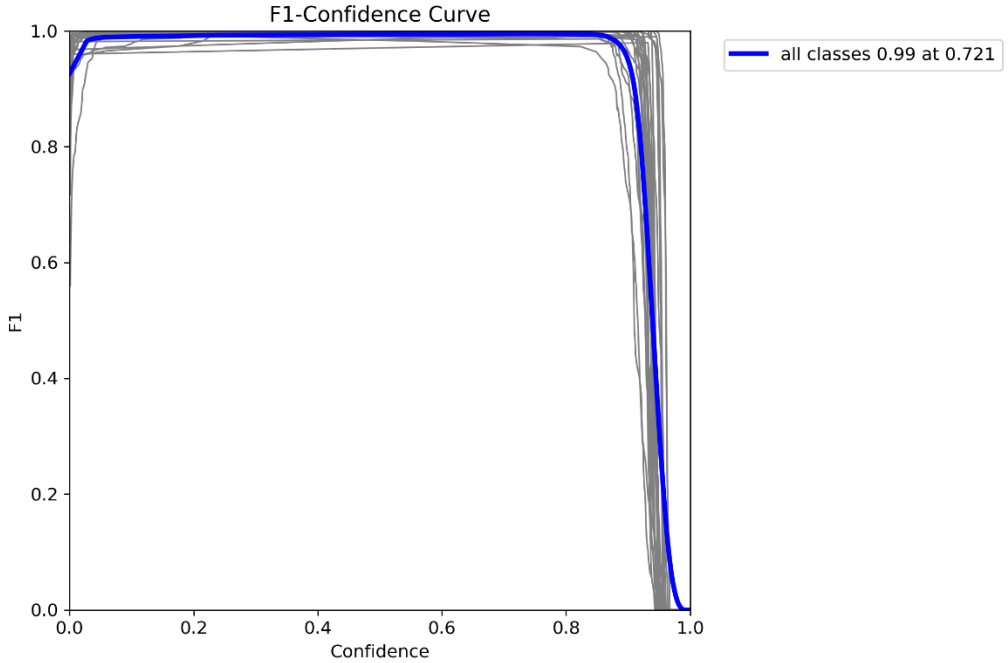


Figura 4. 110: Curva F1

Fuente: Elaboración Propia

4.5.3.3. confusion_matrix.png

Muestra cómo el modelo clasifica cada clase y cuántos errores comete al confundir unas clases con otras.

- Eje horizontal: Clases predichas por el modelo
- Eje vertical: Clases reales en el dataset
- Valores en la diagonal: Predicciones correctas (cuanto más altos, mejor)
- Valores fuera de la diagonal: Errores del modelo (queremos que sean lo más bajos posible)

Si la diagonal principal tiene valores altos:

- El modelo predice correctamente la mayoría de los objetos

Si hay valores altos fuera de la diagonal:

- El modelo está confundiendo unas clases con otras
- Puede haber problemas de etiquetado en el dataset o la red no está aprendiendo bien las diferencias entre clases

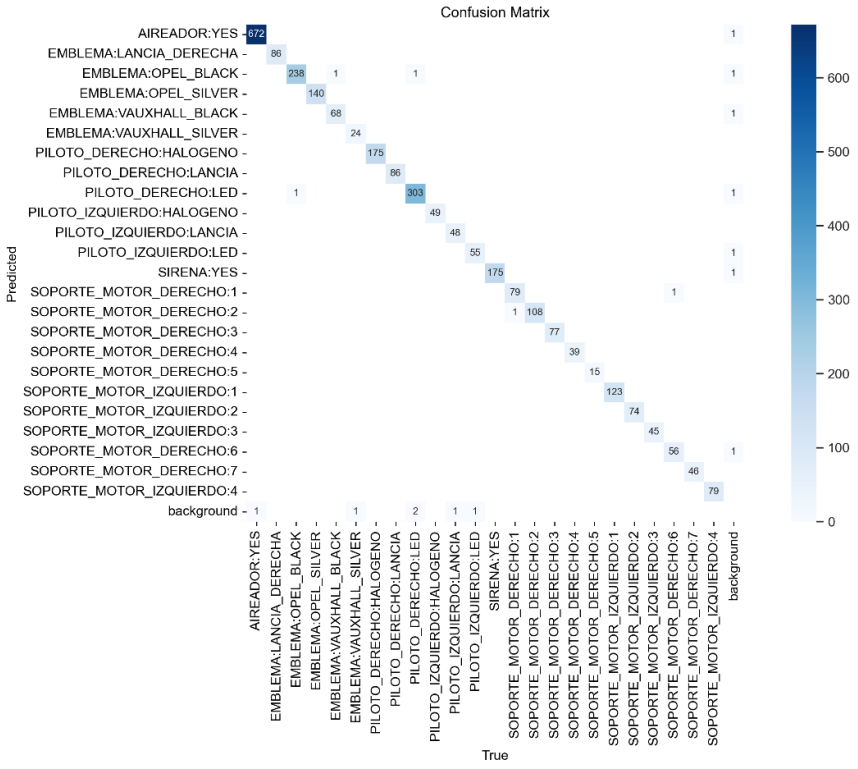


Figura 4. 111: Matriz de confusión

Fuente: Elaboración Propia

4.5.3.4. Mejora del modelo e interpretación de los resultados

Si map50 es bajo, indica baja precisión general

- Causas posibles:

1. Dataset insuficiente o mal etiquetado
2. Modelo subentrenado con pocos datos

- Soluciones:

1. Aumentar el dataset con más imágenes variadas
2. Mejorar la calidad del etiquetado
3. Entrenar más épocas (epochs), pero sin sobreajustar. Es decir, entrenar más épocas puede mejorar el rendimiento del modelo, pero si se excede, puede producir sobreajuste, haciendo que el modelo funcione bien con los datos de entrenamiento pero falle al generalizar con imágenes nuevas
4. Revisar el tamaño del batch (batch_size) y la tasa de aprendizaje (lr)

Si la precisión es baja, indica muchos falsos positivos (detecta cosas incorrectas)

- Causas posibles:

1. Umbral de confianza bajo, el modelo detecta más, pero con errores
2. Clases muy similares entre sí

- Soluciones:

1. Aumentar el umbral de confianza (conf-thres) para evitar predicciones erróneas
2. Revisar y mejorar el etiquetado de clases parecidas
3. Usar más imágenes específicas de la clase que confunde

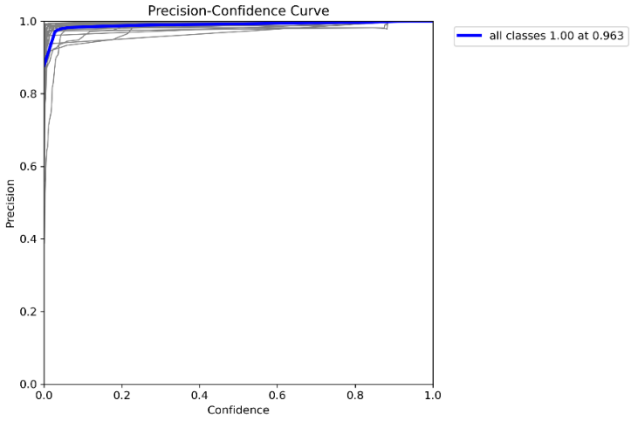


Figura 4. 112: Curva de precisión - umbral de confianza

Fuente: Elaboración Propia

Si Recall es bajo, indica muchos falsos negativos (no detecta cosas que debería)

- Causas posibles:

1. Dataset insuficiente para ciertas clases
2. Modelo demasiado estricto con la detección

- Soluciones:

1. Reducir el umbral de confianza para que detecte más objetos
2. Asegurar que haya suficientes ejemplos por clase en el dataset
3. Revisar si el modelo ha entrenado suficiente (variar epochs)

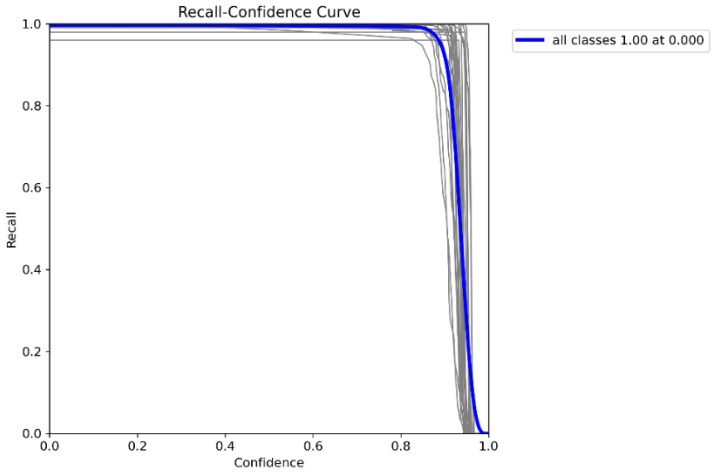


Figura 4. 113: Curva de sensibilidad - umbral de confianza

Fuente: Elaboración Propia

Si la matriz de confusión muestra muchas confusiones entre clases

- Causas posibles:

1. Clases visualmente similares (por ejemplo: coches turismo vs coches sub)
2. Etiquetado inconsistente
3. Falta de datos para algunas clases

- Soluciones:

1. Revisar y corregir etiquetas
2. Asegurar que cada clase tenga suficientes imágenes bien balanceadas
3. Ajustar los pesos de las clases (class_weighting) si alguna es muy minoritaria

5. RESULTADOS

El resultado principal del sistema desarrollado se materializa en las imágenes analizadas por la red neuronal YOLOv5u, entrenada específicamente para detectar mutilaciones y errores de montaje en componentes del vehículo, que gracias a la combinación de Hardware y Software que hemos visto hasta este punto, da como resultado una gran ayuda para los operarios y supervisores.

Como hemos visto, una vez se ha realizado la comparación entre la lectura de la hoja de montaje y lo que ha analizado YOLO, esas imágenes son procesadas y constituyen la salida final del sistema de visión artificial, y son fundamentales para asistir en la toma de decisiones dentro de la planta.

A continuación, se presentan dos ejemplos representativos de resultados obtenidos durante el día a día del turno productivo:

- **IMAGEN CORRECTAMENTE ANALIZADA (OK)**

  En este caso:

  1. Se almacenan las fotos analizadas por la red en su propia carpeta
  2. Se muestran al operario por medio de una interfaz
  3. Espera las siguientes imágenes del vehículo para ser analizadas

Figura 5. 1: Imagen frontal del vehículo analizada por la red neuronal OK

Fuente: Elaboración Propia

En la Figura 5.1 se muestra una captura frontal de un vehículo en la línea de montaje, en la que, tras procesar la imagen por la red neuronal, ha detectado correctamente el control que nosotros hemos entrenado durante un cierto tiempo, en este caso el soporte izquierdo que se encarga de sujetar el motor ensamblado.

El recuadro amarillo resalta la región de interés localizada por el modelo entrenado, y la anotación textual indica no solo el tipo de componente (:1), sino también el tipo de identificador del tipo de motor que nosotros tenemos asignado en nuestras tablas y que corresponde con un LCD, una especie de identificador interno de la planta asignado a un tipo de soporte de motor concreto.

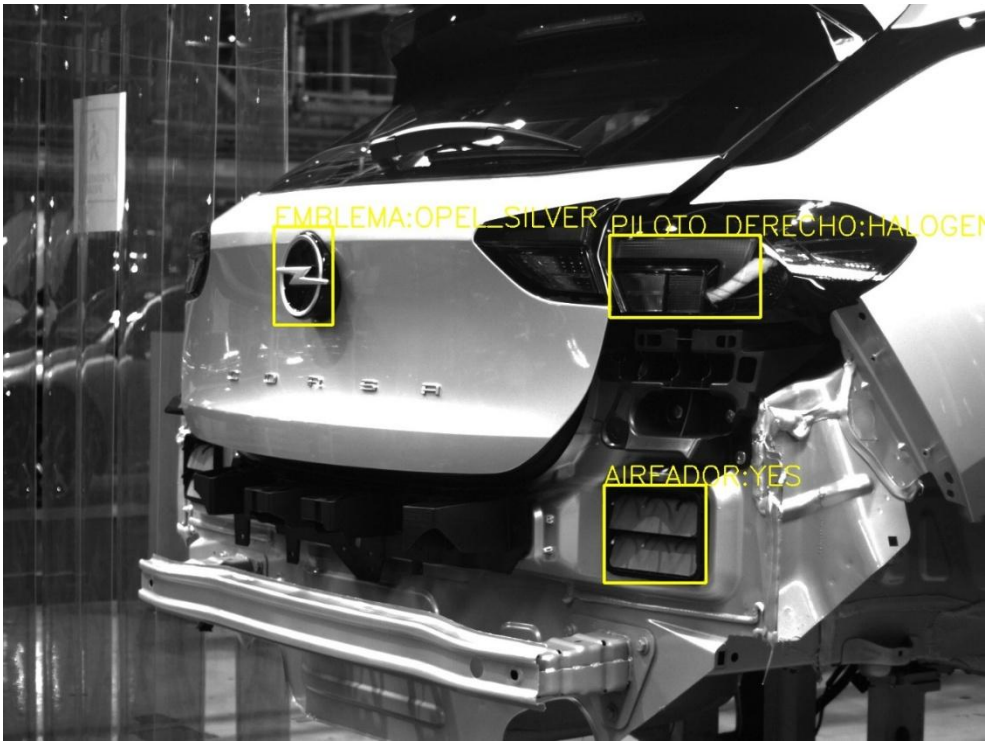


Figura 5. 2: Imagen trasera del vehículo analizada por la red neuronal OK

Fuente: Elaboración Propia

En la Figura 5.2 se presenta una imagen trasera del vehículo en la que la red neuronal ha identificado correctamente tres componentes clave en la fase del montaje: el emblema del portón trasero, el piloto trasero derecho y el aireador. De nuevo, cada uno de estos elementos ha sido localizado mediante cuadros delimitadores amarillos (bounding boxes), acompañados de etiquetas que indican tanto el tipo de componente como su versión específica del mismo.

El proceso del entrenamiento del modelo ha requerido un conjunto variado de imágenes etiquetadas para cada uno de estos componentes, abarcando distintos tipos de emblemas, pilotos y configuraciones de montaje, lo cual ha permitido que el sistema clasifica correctamente ante diferentes versiones del mismo control. Esta parte también requirió de un estudio de todas las diferentes configuraciones encontradas en los distintos controles (Figuras 5.3 y 5.4).

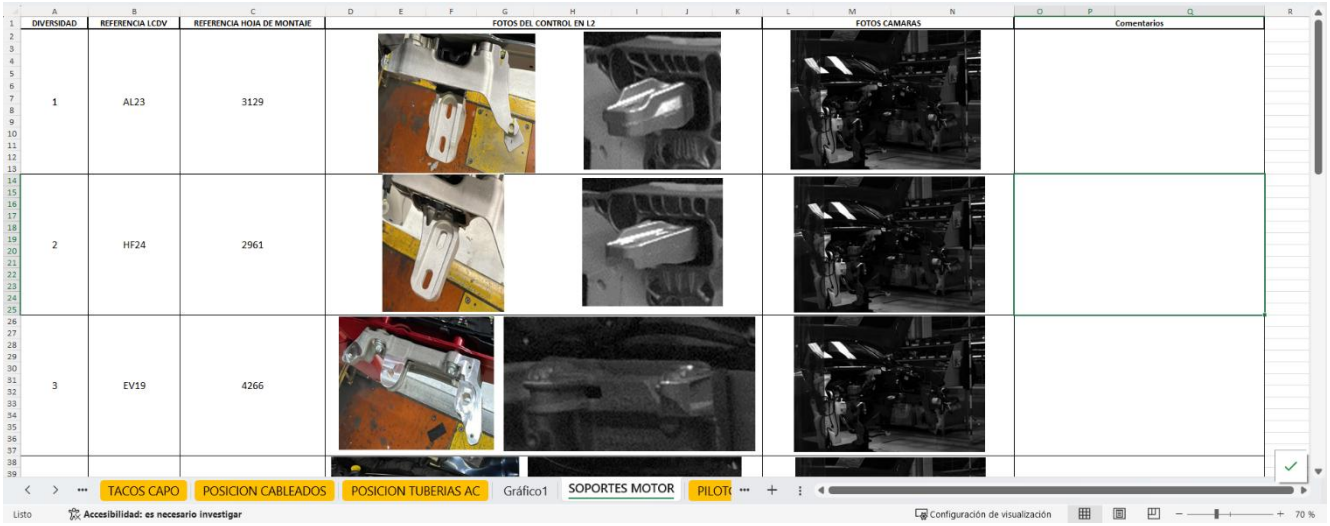


Figura 5. 4: Tabla Excel con los diferentes tipos de configuración de soportes de motor

Fuente: Elaboración Propia

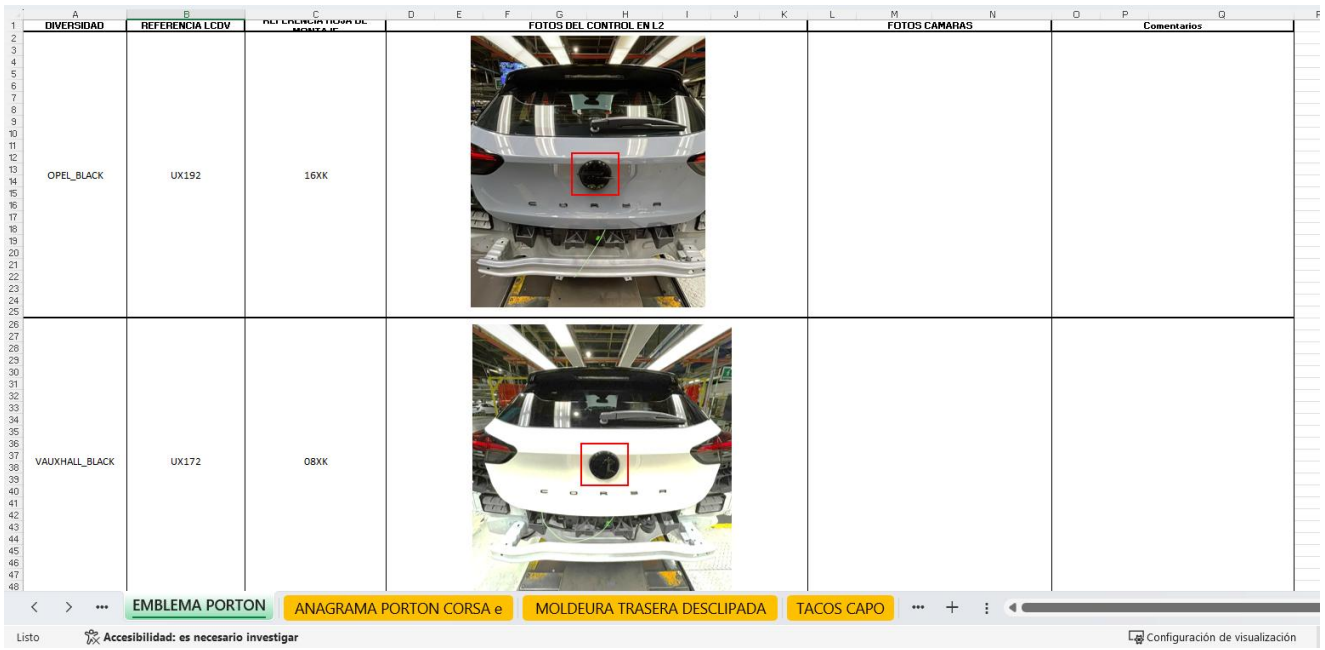


Figura 5. 3: Tabla Excel con las diferentes configuraciones del emblema trasero del vehículo

Fuente: Elaboración Propia

- IMAGEN INCORRECTAMENTE ANALIZADA (NOK)

En este caso:

1. Se activa la baliza, que emite una señal sonora y visual
2. Se muestra al operario por medio de la interfaz las imágenes del vehículo que ha salido como NOK
3. Se almacenan las fotos de ese vehículo en una carpeta propia (NOK) para su posterior consulta y análisis

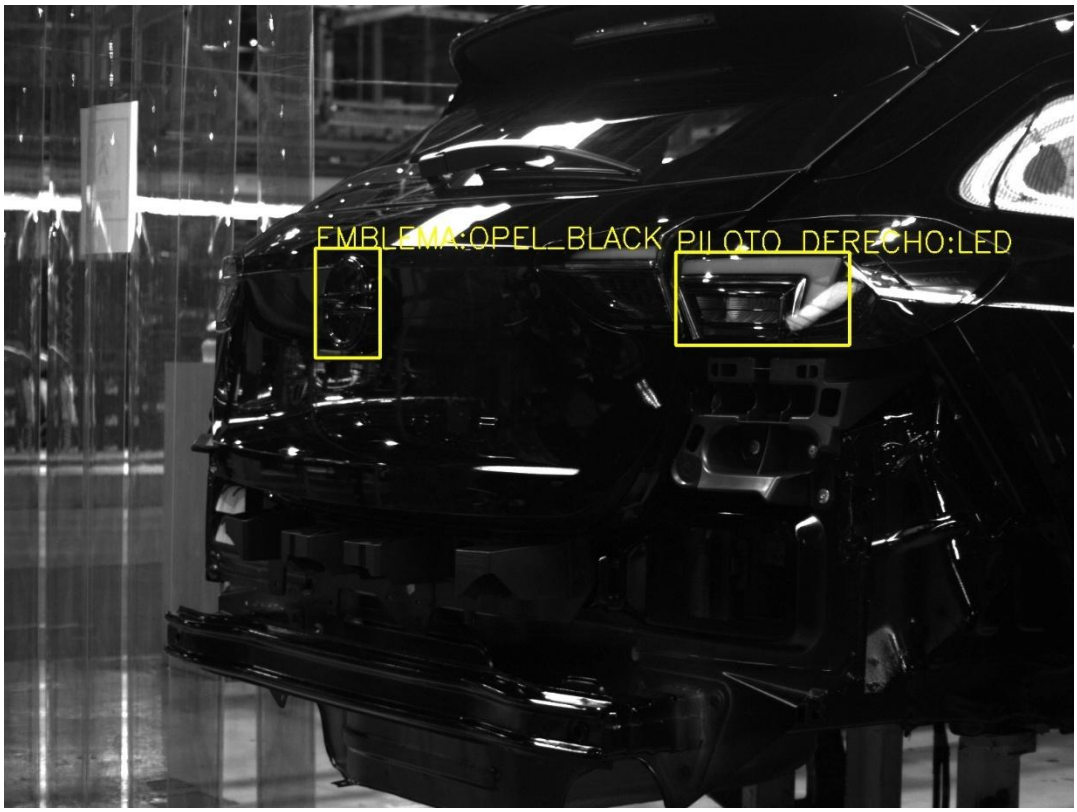


Figura 5. 5: Imagen trasera del vehículo analizada por la red neuronal NOK

Fuente: Elaboración Propia

En la Figura 5.5 se muestra nuevamente una captura de la parte trasera del vehículo, pero en este caso con una condición: falta por completo el aireador derecho. Esta situación representa un caso real de montaje incompleto, que el sistema detecto de forma automática. Tal como se observa, la red neuronal ha procesado la imagen y no ha detectado el aireador derecho, generando una salida de tipo NOK.

Pero la instalación no termina aquí, en caso de error deberemos alertar al operario por medio de la baliza. Para esta baliza se utiliza como intermediario el microcontrolador ESP32, pues este está constantemente leyendo el puerto serie. La señal del ESP32, como ya hemos visto, es acondicionada para aumentar su tensión de 3,3 V a 24 V mediante un optoacoplador insertado en una PCB de fabricación propia, para hacer que la baliza funcione.

Finalmente, para facilitar al operario la interpretación de los resultados y la visualización de las imágenes, se ha desarrollado en Python una Interfaz de Usuario Guiada o GUI (por sus siglas en inglés, Guided Users Interface), mediante la librería de PyQT5. En esta interfaz aparecen las fotos tomadas y lo que está analizando la red neuronal, además de otra información adicional (Figuras 5.6 y 5.7).

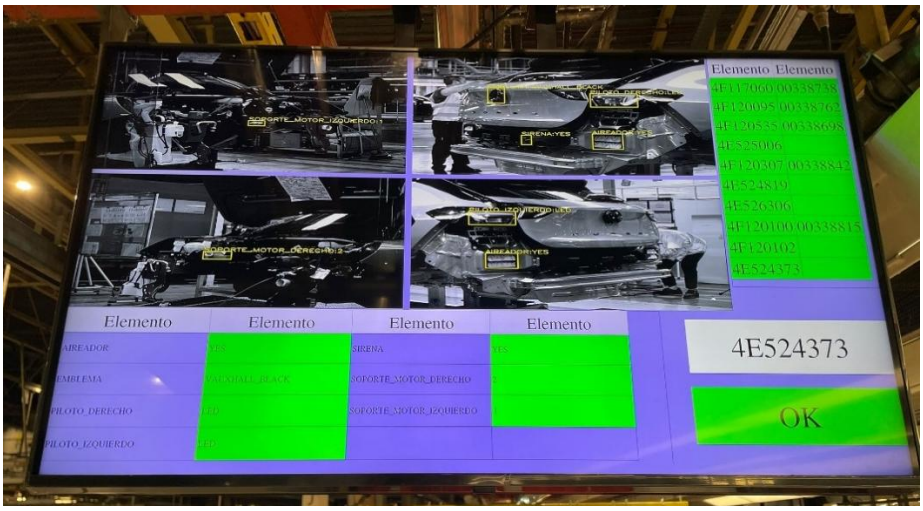


Figura 5. 6: GUI con resultado OK

Fuente: Elaboración Propia

Figura 5. 7: GUI con resultado NOK

Fuente: Elaboración Propia

La Tabla 2 de abajo muestra los presupuestos totales y reales de la instalación. Cabe destacar que, si bien el coste total estimado de los materiales y componentes asciende a 16.843,18 euros, el coste real de la instalación ha sido significativamente menor con un total de 7.433,11 euros. Esta notable reducción se ha logrado gracias al aprovechamiento de materiales reciclados y componentes reutilizados procedentes de instalaciones anteriores, lo cual no solo optimiza los recursos disponibles, sino que también refuerza el carácter sostenible y low cost del proyecto.

Hay que mencionar también que los materiales marcados con un asterisco (*) de la tabla de presupuestos corresponden al ordenador utilizado en la Workstation. Este equipo se utiliza principalmente para el entrenamiento de los modelos de visión artificial que se utilizan en 3 estaciones de visión, por tanto, su coste real se ha dividido proporcionalmente entre ellas, imputando únicamente un tercio del valor total a la estación actual.

Además, el presente proyecto constituye una base para futuras instalaciones, que ya se están empezando a implementar en otras zonas de la planta, como CAR o Pinturas, lo que permitirá escalar la solución de manera eficiente y dividir todavía más esos costes asociados a la Workstation.

ELEMENTO	PROCEDENCIA	COSTE UNITARIO	COSTE TOTAL	COSTE REAL
Procesador Intel Core i7-6800K 3.4 GHz	Reciclado	205,93 €	205,93 €	0,00 €
Placa Gigabyte B760M DS3H DDR4	Reciclado	119,57 €	119,57 €	0,00 €
Memoria RAM 32GB Kingston FURY Beast DDR4 3200 MHz 2×16 GB	Reciclado	74,90 €	74,90 €	0,00 €
Gráfica MSI GTX 1080 Ti GAMING Nvidia GeForce Go PCI Express x16 3.0	Reciclado	329,00 €	329,00 €	0,00 €
Disco duro 2.5 TB Samsung 870 EVO SSD 2 TB	Reciclado	154,95 €	154,95 €	0,00 €
Fuente de alimentación 500 W L-Link ATX 500 W	Reciclado	26,63 €	26,63 €	0,00 €
Caja torre 3Go Nain2	Reciclado	38,89 €	38,89 €	0,00 €
Monitor HP LE1901w	Reciclado	90,00 €	90,00 €	0,00 €
Teclado Dell Multimedia KB216	Reciclado	25,00 €	25,00 €	0,00 €
Ratón Logitech MX Master 3S	Reciclado	30,00 €	30,00 €	0,00 €
Televisor LG 43UR78	Almacén IM01467529	312,76 €	312,76 €	312,76 €
8WD4420-5AD SIEMENS señalizador luminoso; amarillo; LED	Almacén IM01509704	52,89 €	52,89 €	52,89 €
8WD4420-0FA SIEMENS Módulo: señalizador acústico; negro	Almacén IM01299209	51,32 €	51,32 €	51,32 €
Base baliza	Almacén IM01372630	54,14 €	54,14 €	54,14 €
NVIDIA Jetson Xavier NX 16GB	The Imagin Source (Iberoptics)	N/A	N/A	850,00 €
6 channel board	The Imagin Source (Iberoptics)	N/A	N/A	846,00 €
Cámaras BFLY-PGE-20E4M-CS	Reciclado	546,09 €	3.276,54 €	0,00 €
Cámaras DFM 36CX290-ML	The Imagin Source (Iberoptics)	329,00 €	1.974,00 €	1.974,00 €
Objetivos FJN-DF6HA-1S	The Imagin Source (Iberoptics)	135,00 €	810,00 €	810,00 €
Objetivos Tamron M118FM16	Reciclado	158,70 €	952,20 €	0,00 €
Sensor SICK WL12L-2B530	Almacén IM1427840	376,18 €	752,36 €	752,36 €
Reflector	Reciclado	20,46 €	40,92 €	0,00 €
Switch D-LINK	Reciclado	82,39 €	82,39 €	0,00 €
deleyCON 15m Cable de Red Ethernet RJ45 CAT6 1000Mbit	Amazon	11,06 €	66,36 €	66,36 €
TIS-CA-FAKRA-IP67-15m - Camera - Computer Cables	The Imagin Source (Iberoptics)	58,00 €	348,00 €	348,00 €
ESP32	AliExpress	3,28 €	3,28 €	3,28 €
Armario	Reciclado	40,00 €	40,00 €	0,00 €
Soporte Ballhead KJ-7	Reciclado	17,25 €	17,25 €	0,00 €
Adaptador trípode	Reciclado	8,37 €	50,22 €	0,00 €
Placas PVC	Reciclado	0,07 €	0,84 €	0,00 €
Soporte PVC	Fabricado en 3D con impresora	0,64 €	3,84 €	3,84 €
Perfiles estructura 75x75 de Paletti	Reciclado	144,16 €	1.243,38 €	0,00 €
Perfiles estructura 75x50 de Paletti	Reciclado	92,50 €	272,41 €	0,00 €
Precision Bolt 50 Km6	Reciclado	14,00 €	840,00 €	0,00 €
Precision Bolt 75 Km6	Reciclado	18,20 €	691,60 €	0,00 €
Set DIN912 A2 - 900 ud	Reciclado	38,99 €	38,99 €	0,00 €
Caja cuadro eléctrico	Reciclado	24,79 €	24,79 €	0,00 €
PIA 16A	Almacén IM01426875	9,42 €	9,42 €	9,42 €
Siemens 6EP3332-6SB00 Logo!Power 24 V/2.5 A	Almacén IM00526419	57,50 €	57,50 €	57,50 €
PCB	Fabricada	3,40 €	3,40 €	3,40 €
Borneras WAGO 870 2,5 mm2	Reciclado	1,34 €	13,40 €	0,00 €

ELEMENTO	PROCEDENCIA	COSTE UNITARIO	COSTE TOTAL	COSTE REAL
Borneras VIKING3 2,5 mm2	Electricidad y Luz Aragón S.L	1,73 €	25,95 €	25,95 €
Señalizadores precortados amarillo Ø5 PVC	Reciclado	0,02 €	0,46 €	0,00 €
Carril DIN	Reciclado	4,53 €	2,54 €	0,00 €
Canaletas para pasar cables	Reciclado	2,65 €	3,58 €	0,00 €
Punteras aisladas "TTE" 1 mm2 - ROJO	Electricidad y Luz Aragón S.L	0,02 €	8,00 €	8,00 €
Cable 1,5 mm2	Reciclado	0,78 €	1,56 €	0,00 €
Cable 1 mm2	Reciclado	0,41 €	1,64 €	0,00 €
Prensaestopas	Reciclado	2,18 €	8,72 €	0,00 €
*Placa Base Asus ProArt	PC Componentes	347,11 €	347,11 €	115,70 €
*Procesador AMD Ryzen	PC Componentes	330,58 €	330,58 €	110,19 €
*Memoria RAM 2,X32GB 3200MHZ CL16	PC Componentes	396,69 €	396,69 €	132,23 €
*SSD Samsung 980 PRO 1TB	PC Componentes	169,42 €	169,42 €	56,47 €
*Fuente de alimentación Corsair HX1200	PC Componentes	202,48 €	202,48 €	67,49 €
*Caja Torre Pure Base 500	PC Componentes	115,70 €	115,70 €	38,57 €
*Disipador Noctua NH-D15 SE-AM4	PC Componentes	99,17 €	99,17 €	33,06 €
*Tarjeta Gráfica GeForce RTX 3080 Ti	PC Componentes	1.636,36 €	1.636,36 €	545,45 €
*Teclado Logitech G213	PC Componentes	53,72 €	53,72 €	17,91 €
*Ratón Logitech G502	PC Componentes	53,72 €	53,72 €	17,91 €
*Monitor ViewSonic 27" WQHD	PC Componentes	206,71 €	206,71 €	68,90 €
TOTAL			16.843,18€	7.433,11€

Tabla 2: Presupuestos de la estación de visión

También se adjunta un análisis de los resultados obtenidos por la estación de visión artificial de TRIM mediante una estadística hecha en PowerBI Desktop de los controles entrenados hasta la fecha, diferenciando entre porcentaje de detecciones OK y NOK por cada uno de los elementos inspeccionados.

Esta estadística presente en la Figura 5.8 permite identificar con precisión que controles presentan mayor tasa de fallo, facilitando así decisiones de mejora tanto a nivel de diseño de sistema como dentro de la línea.

Figura 5. 8: Estadística de la estación desde el 05/05/2025 hasta el 25/06/2025

Fuente: Elaboración Propia con PowerBI Dekstop

Además, al analizar la evolución temporal de estos porcentajes, se observa una mejora progresiva en la tasa de aciertos (OK) . Este comportamiento se explica por el hecho de que, a medida que se recopila mayor numero de imágenes y resultados reales, el sistema ha sido reentrenado periódicamente para mejorar su capacidad de detección.

Esto queda patente en la Figura 5.9 con el control del anagrama, la tasa de NOK en los primeros meses era muy alta, del 25.20% aproximadamente. Sin embargo, tras aplicar mejoras en el dataset y realizar nuevos entrenamientos con imágenes especificas de fallos detectados, esta tasa se reduce significativamente hasta un 12.28% y seguimos trabajando en ella.

Figura 5. 9: Estadística de la estación desde el 10/06/2025 hasta el 25/06/2025

Fuente: Elaboración Propia

6. CONCLUSIONES

El desarrollo e implementación e implementación de esta estación de visión artificial ha permitido cumplir satisfactoriamente los objetivos propuestos al inicio del trabajo.

Mediante el uso de una red neuronal YOLO entrenada específicamente para el entorno de producción en STELLANTIS, se ha conseguido un sistema capaz de:

1. Detectar automáticamente errores de montaje en los componentes del vehículo y mutilaciones en la superficie de este
2. Procesar imágenes en tiempo real, proporcionando una respuesta inmediata ante cualquier fallo, gracias a la implementación de sensores y la combinación entre un hardware robusto y un software eficiente
3. Generar trazabilidad visual, almacenando imágenes etiquetadas correcta e incorrectamente como evidencia de cada detección, para su posterior consulta

El sistema ha demostrado ser de gran utilidad para los supervisores de la línea, ya que les facilita la verificación de fallos mediante evidencias visuales claras y objetivas. Las imágenes almacenadas permiten no solo justificar la imputación de errores en el proceso, sino también identificar el punto exacto de la línea donde se ha producido el fallo. Esto repercute directamente en la calidad del producto final y en una optimización de los procesos de supervisión, mejora continua y mantenimiento.

Además, uno de los aspectos más destacables del sistema a tener en cuenta es su capacidad de adaptación continua y modularidad. La industria automovilística está en constante evolución tecnológica, incorporando nuevos modelos de vehículos, actualizaciones de software de cada vehículo, nuevos componentes y procesos de ensamblajes cada vez más complejos. En este contexto dinámico, la estación desarrollada se encuentra en constante actualización, incorporando de forma progresiva nuevos controles, nuevo hardware más potente y nuevas lógicas de análisis que permiten mantener el sistema alineado con los requerimientos actuales de producción.

Esta flexibilidad asegura que la solución no quede obsoleta, sino que evolucione al ritmo de la fábrica, a la par que globalmente con la tecnología, ofreciendo una herramienta de control de calidad robusta, escalable y completamente integrada en la realidad de la industria 4.0.

6.1. TRABAJOS FUTUROS

Como continuación de este trabajo, se han identificado distintas líneas de mejora y ampliación que, a día de hoy, ya se están comenzando a implementar y que permiten reforzar el rendimiento del sistema y aumentar su aplicabilidad en otras áreas de la planta. A continuación, se destacan las principales:

1. **Problema al ejecutar la interfaz gráfica (PyQT5)**

Tras implementar la separación de usuarios, nos encontramos con una problemática relacionada con la ejecución de programas con interfaz gráfica desde cuentas con permisos restringidos.

El software de la estación incluye una interfaz gráfica basada en Qt, denominada QTInterface la cual normalmente es lanzada desde terminal. Sin embargo, al intentar ejecutar esta interface desde terminal con el usuario administrador (trimdesktop) desde la sesión de los operarios (supts12) nos era imposible debido a que en Linux un usuario no puede iniciar una aplicación gráfica en un entorno de otro usuario sin configurar correctamente el acceso al DISPLAY (servidor gráfico).

Para permitir que trimdesktop (administrador) pudiera ejecutar QTInterface dentro de la sesión iniciada por supts12, tuvimos que aplicar una configuración manual del DISPLAY:

- Desde la sesión de supts12, permitimos el acceso al servidor gráfico mediante el comando `xhost +` que permite que otros usuarios puedan utilizar la pantalla gráfica
- Obtuvimos el valor actual del DISPLAY en la sesión de supts12 ejecutando `echo $DISPLAY` obteniendo un resultado (`:0`) que indica la pantalla en uso
- Iniciamos sesión con trimdesktop y exportamos manualmente el valor del DISPLAY obtenido con el comando `export DISPLAY=:0`

Con esto, trimdesktop ya podía ejecutar QTInterface dentro de la sesión gráfica de supts12 sin errores.

No obstante, esta solución es temporal y requiere intervención manual, por lo que la solución a largo plazo y en la cual ya estamos trabajando, pasa por desarrollar una interfaz web accesible desde cualquier estación de trabajo, utilizando Angular como framework de frontend. Esto no solo eliminará las limitaciones impuestas por el entorno grafico de Linux, sino que permitirá una mayor flexibilidad, mantenimiento remoto y escalabilidad del sistema.

2. Software en la placa embebida NVIDIA Xavier NX

Uno de los principales inconvenientes encontrados durante el desarrollo del proyecto fue la incompatibilidad del entorno de software en la NVIDIA Xavier con las herramientas y librerías actuales. La Xavier funciona con el sistema operativo JetPack 4.6, el cual se basa en una versión antigua de Ubuntu y viene preinstalado con Python 3.6. Esto generó múltiples problemas, ya que Python 3.6 ha sido descontinuado y, en consecuencia, muchas librerías modernas no ofrecen soporte para esta versión. Por ejemplo, para la implementación de YOLOv5, pues su versión estándar requiere versiones más recientes de Python (como 3.8 o superior), lo que hizo imposible su instalación y ejecución sin realizar modificaciones en el código o en las dependencias.

Para evitar estos problemas en futuros proyectos, una alternativa viable seria delegar el procesamiento de imágenes a color por parte de las cámaras encargadas de las mutilaciones, que actualmente están conectadas a la Xavier (FPD-Link) a un ordenador externo con hardware más potente y un sistema operativo actualizado. Esto permitiría utilizar versiones recientes de Python y librerías sin las restricciones impuestas por el JetPack de Xavier.

3. Problemática del OCR y la lectura de la hoja de montaje

Durante el desarrollo del proyecto, encontramos un grave inconveniente en la implementación del sistema de reconocimiento óptico de caracteres (OCR) para la lectura de la hoja de montaje del coche. Las cámaras utilizadas para la capturar las imágenes y procesarlas con OCR presentan muchos problemas que afectan a la fiabilidad del sistema y obligan a realizar constantes ajustes tanto en el software, como en el hardware ajustando las cámaras, para corregir estos errores.

Uno de los problemas más significativos es la sensibilidad del sistema a los movimientos de la cámara. Cualquier desplazamiento, por mínimo que sea, afecta a la capacidad del OCR provocando fallos en la identificación de los códigos (referencias) de la hoja de montaje. Además, observamos casos en los que el sistema no detecta correctamente las referencias de algunos coches, lo que genera inconsistencias en los registros y requiere soluciones temporales en el software para intentar compensar estos errores, generando un código “parcheado”.

Otro importante inconveniente fue la presencia de códigos repetidos en la propia hoja de montaje. En varias ocasiones, el OCR encontraba estos valores duplicados y los asignaba incorrectamente, haciendo que el sistema procesara información errónea, afectando a la trazabilidad y al correcto análisis de las imágenes.

Para resolver estas deficiencias, se propone reemplazar el actual sistema basado en OCR por un dispositivo ESP100 con base ESP32. Este dispositivo, alimentado a 24V y equipado con una antena Wi-Fi, se conecta directamente a los servidores de producción a través de un servidor MQTT. Mediante esta comunicación, el EdgeBox recibe los datos del vehículo que ingresa a la estación y

transmite esta información al ordenador a través del puerto serie, lo que presenta ventajas clave en la estación:

- Eliminar la dependencia del OCR, evitando errores de lectura causados por la calidad de la imagen, la iluminación o el movimiento de la cámara
- Garantiza una transmisión de datos más precisa y fiable
- Mejora la integración con los sistemas de producción, minimizando la perdida de información

Figura 6. 1: Dispositivo ESP100 con base ESP32

Fuente: Elaboración Propia

4. Problemas de ancho de banda

Otro de los principales desafíos técnicos es la recepción incompleta de imágenes provenientes de las cámaras. En muchas ocasiones, las imágenes presentan defectos como franjas negras o, en algunos casos, ni siquiera llegan al sistema. Esto afecta directamente al procesamiento posterior y a fiabilidad del sistema, ya que una imagen corrupta o incompleta no podía ser utilizada por la red neurona YOLO, lo cual generaba falsos negativos o directamente ausencia de detección.

Nos dimos cuenta de que una de las causas del problema estaba relacionada con limitaciones de ancho de banda en la tarjeta de red integrada del equipo de la estación. Como solución, se propone

incorporar una tarjeta HP NC552SFP de 10GbE, que permite una transmisión más estable y rápida de imágenes.

5. Problemas en la gestión de entradas y salidas digitales

Tenemos el problema de que la gestión de entradas y salidas digitales, que se realiza a través de una PCB propia conectada a un microcontrolador, aunque funcional, no es la solución más adecuada para un entorno industrial ya que no ofrece el nivel de fiabilidad, robustez ni profesionalidad que requiere una estación de visión artificial integrada en una línea de producción. Además, implica una conversión manual de tensiones, escasa protección ante interferencias, y un diseño más artesanal que industrial.

Para solucionar esto, estamos realizando pruebas con el uso del IFM AL1321, un módulo IO-Link Master con entradas digitales de 24 V que actúa como un concentrador de señales, especialmente diseñado para entornos industriales. Algunas características del dispositivo:

- Cuenta con 4 puertos IO-Link tipo A, con capacidad para alimentar dispositivos conectados con hasta 3,6 A en total
- Soporta comunicación PROFINET IO, Ethernet/IP y protocolo MQTT facilitando su integración en redes industriales estándar
- Ofrece aislamiento galvánico para proteger la electrónica y evitar interferencias externas
- Incluye diagnostico avanzado en tiempo real que detecta fallos en sensores o cables, facilitando el mantenimiento predictivo

Por tanto, nos permite sustituir el microcontrolador y centralizar la gestión de señales, con múltiples ventajas: aislamiento eléctrico, diagnóstico de entradas, configuración de los sensores conectados (tiempos de subida y bajada de la señal) y una instalación mucho más limpia y compacta. Aunque supone una inversión mayor, el AL1103 aporta una solución robusta, fiable y escalable, más acorde con los estándares de la industria automovilística.

Actualmente, nos encontramos probándolo en campo en otra instalación de visión para su futura integración completa en el sistema y su posible sustitución en la estación de la que trata este trabajo.

Figura 6. 2: Maestro IO-Link con interfaz PROFINET AL1103

Fuente: (IFM, 2024)

7. OBJETIVOS DE DESARROLLO SOSTENIBLE

Los objetivos de este Trabajo Fin de Grado están alineados con los siguientes Objetivos de Desarrollo Sostenible (ODS) y metas, de la Agenda 2030:

- Objetivo 4 - Garantizar una educación inclusiva y equitativa de calidad y promover oportunidades de aprendizaje permanente para todos
- Meta 4.4. De aquí a 2030, aumentar considerablemente el número de jóvenes y adultos que tienen las competencias necesarias, en particular técnicas y profesionales, para acceder al empleo, el trabajo decente y el emprendimiento

- Objetivo 8 - Promover el crecimiento económico sostenido, inclusivo y sostenible, el empleo pleno y productivo y el trabajo decente para todos
- Meta 8.2. Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra

- Objetivo 9 – Conseguir infraestructuras sostenibles, resilientes y de calidad para todos, impulsar una nueva industria bajo criterios de sostenibilidad que adopte tecnologías y procesos industriales limpios y ambientalmente racionales, fomentar la tecnología, la innovación y la investigación y lograr el acceso igualitario a la información y al conocimiento, principalmente a través de internet
- Meta 9.4. De aquí a 2030, modernizar la infraestructura y reconvertir las industrias para que sean sostenibles, utilizando los recursos con mayor eficacia y promoviendo la adaptación de tecnologías y procesos industriales limpios y ambientalmente racionales, y logrando que todos los países tomen medidas de acuerdo con sus capacidades respectivas.

- Objetivo 12 – Garantizar modalidades de consumo y producción sostenibles: la reutilización de componentes en el diseño de la estación fomenta practicas sostenibles, reduciendo la generación de residuos electrónicos y el consumo de recursos nuevos.

- Meta 12.2. De aquí a 2030, reducir considerablemente la generación de desechos mediante actividades de prevención, reducción, reciclado y reutilización.

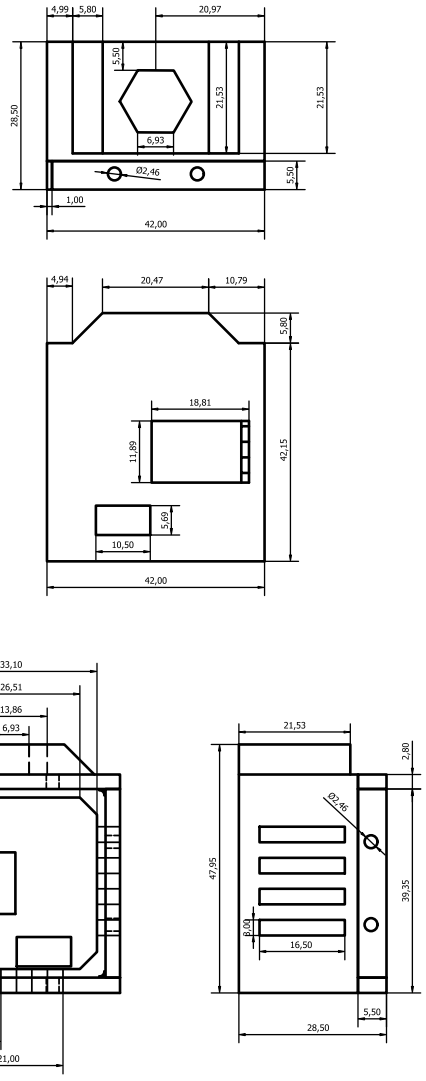
- Objetivo 13 – Adoptar medidas urgentes para combatir el cambio climático y sus efectos: al reutilizar materiales y optimizar el uso de recursos, el proyecto contribuye a la reducción del impacto ambiental y a la sostenibilidad de los procesos industriales

- Meta 13.2. Incorporar medidas relativas al cambio climático en las políticas, estrategias y planes nacionales

8. ANEXOS

8.1. ANEXO A: PLANOS DE SOPORTE PARA CÁMARAS MIPI

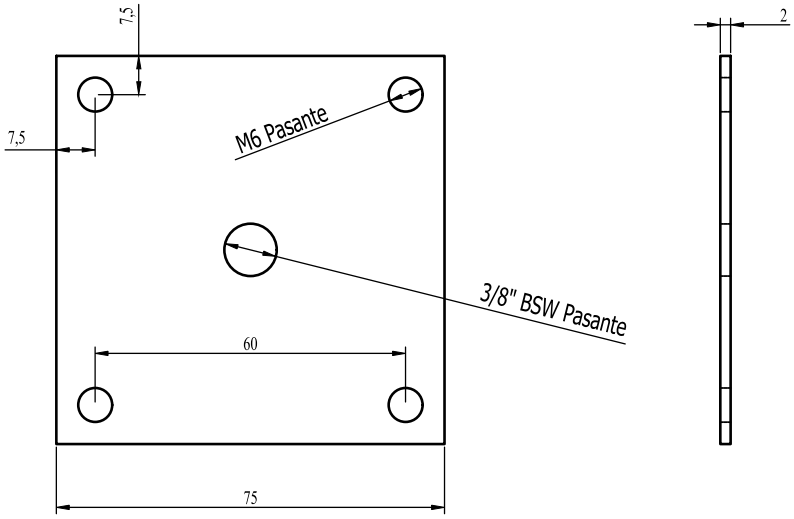
A4



Revisión: A 06/2025		Fecha	Nombre y apellidos	Firma:	Escuela Universitaria Politécnica - La Almunia, Centro adscrito, Universidad Zaragoza
	Dibujado:	05/2025	Jorge Lasarte Aneiros		
	Comprobado;	06/2025	Tribunal Mecatrónica		
ESCALA: 2:1	ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST CARCASA PARA CÁMARAS MIPI EN PLA			HOJA: 1/1	CÓDIGO DE PLANO: 424.23.01.0L.01

8.2. Anexos B: Placas De PVC Para Anclaje De Cámaras

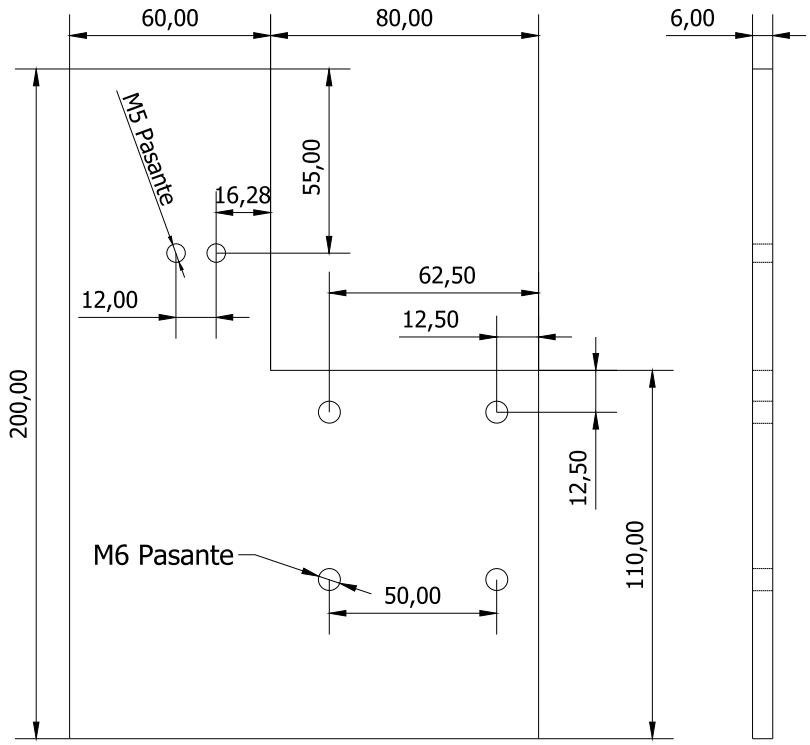
A4



Revisión: A 06/2025		Fecha	Nombre y apellidos	Firma:	Escuela Universitaria Politécnica - La Almunia, Centro adscrito, Universidad Zaragoza
	Dibujado:	05/2025	Jorge Lasarte Aneiros		
	Comprobado:	06/2025	Tribunal Mecatrónica		
ESCALA: 1:1	ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST PLACA PVC PARA SOPORTE DE CÁMARAS EN LA ESTRUCTURA			HOJA: 1/1	CÓDIGO DE PLANO: 424.23.02.0L.01

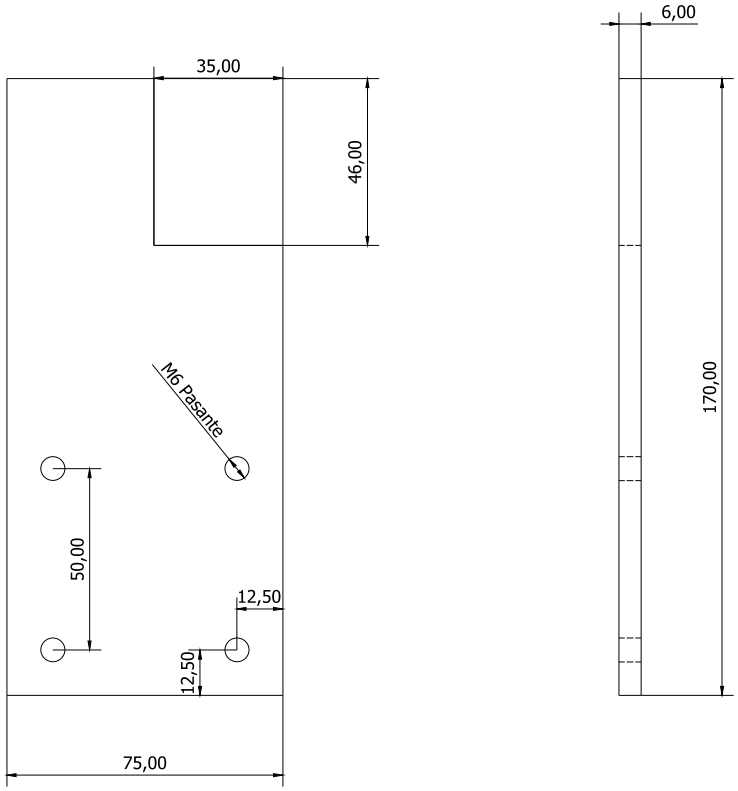
8.3. ANEXOS C: PLACAS DE PVC PARA ANCLAJE DE SENSORES

A4



Revisión: A 06/2025		Fecha	Nombre y apellidos	Firma:	Escuela Universitaria Politécnica - La Almunia, Centro adscrito, Universidad Zaragoza
	Dibujado:	05/2025	Jorge Lasarte Aneiros		
	Comprobado;	06/2025	Tribunal Mecatrónica		
ESCALA: 1:2	ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST PLACA PVC PARA ADAPTAR SENSOR A LA ESTRUCTURA			HOJA: 1/2	CÓDIGO DE PLANO: 424.23.03.0L.01

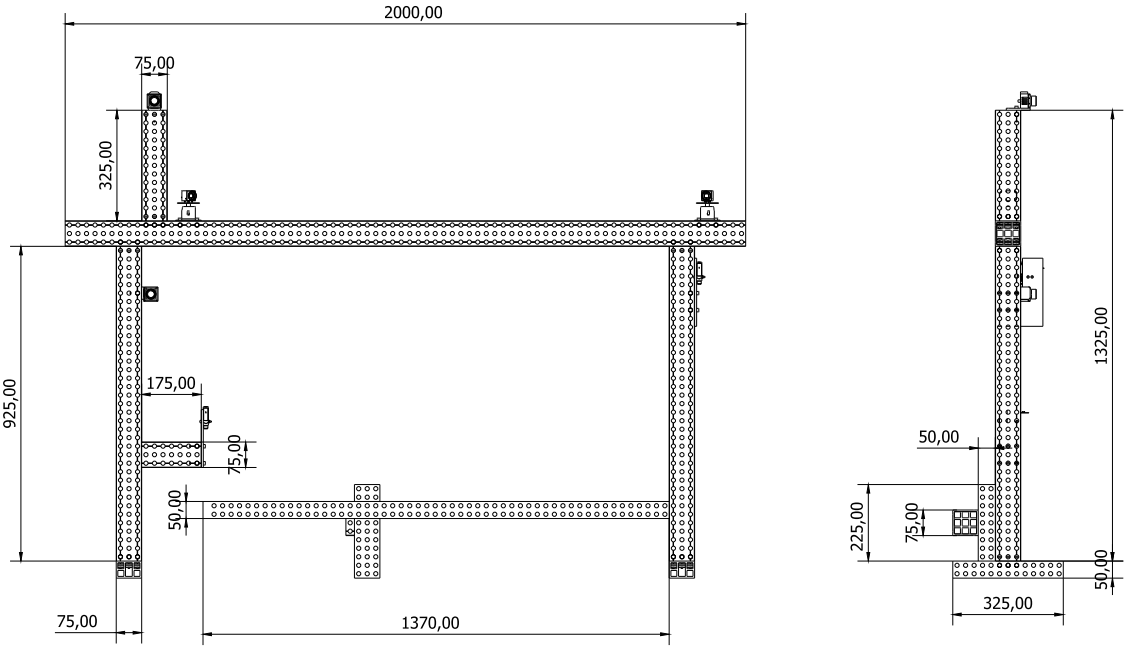
A4



Revisión: A 06/2025		Fecha	Nombre y apellidos	Firma:	Escuela Universitaria Politécnica - La Almunia, Centro adscrito, Universidad Zaragoza
	Dibujado:	05/2025	Jorge Lasarte Aneiros		
	Comprobado;	06/2025	Tribunal Mecatrónica		
ESCALA: 1:2	ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST PLACA PVC PARA ADAPTAR SENSOR A LA ESTRUCTURA			HOJA: 2/2	CÓDIGO DE PLANO: 424.23.03.0L.02

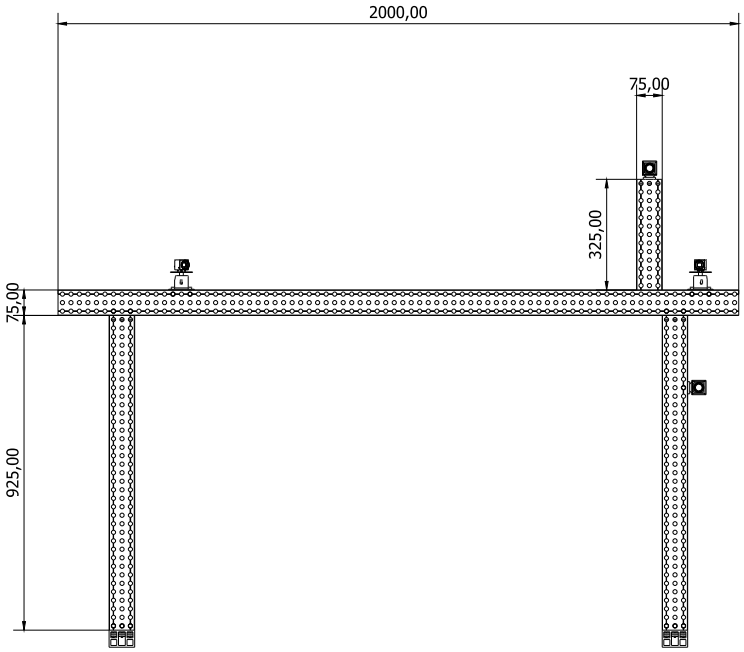
8.4. ANEXOS D: ESTACIÓN DE VISIÓN ARTIFICIAL

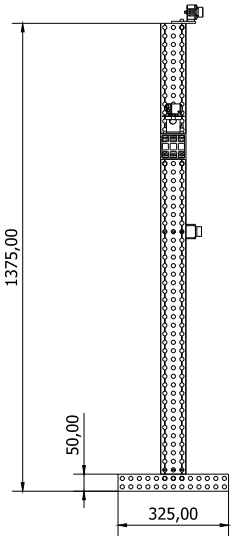
A4



Revisión: A 06/2025		Fecha	Nombre y apellidos	Firma:	Escuela Universitaria Politécnica - La Almunia, Centro adscrito, Universidad Zaragoza
	Dibujado:	06/2025	Jorge Lasarte Aneiros		
	Comprobado:	06/2025	Tribunal Mecatrónica		
ESCALA: 1:10	ESTACIÓN DE VISIÓN ARTIFICIAL LOWCOST ESTRUCTURA DE LA ESTACIÓN DE VISIÓN			HOJA: 1/3	CÓDIGO DE PLANO: 424.23.04.0L.01

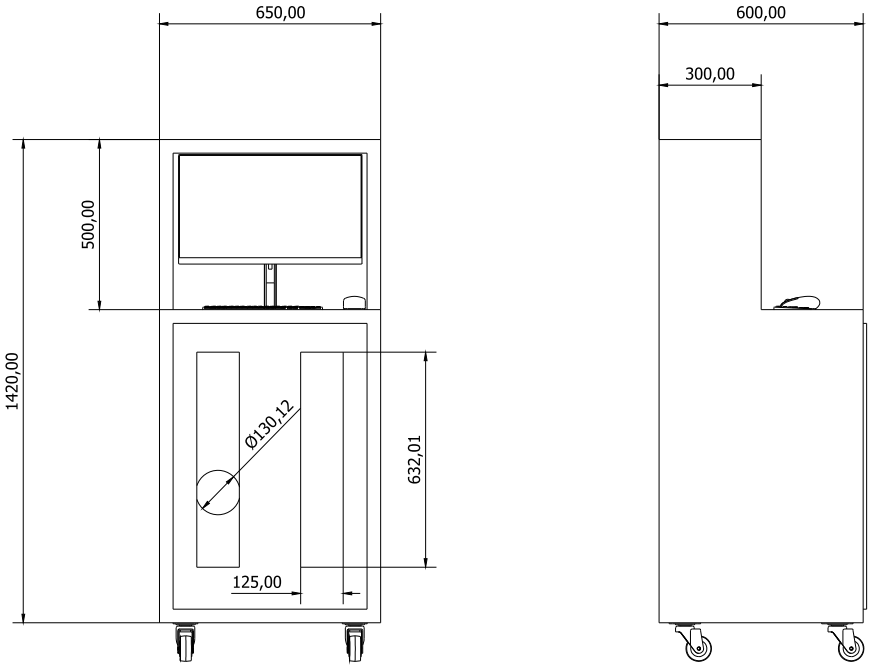
A4





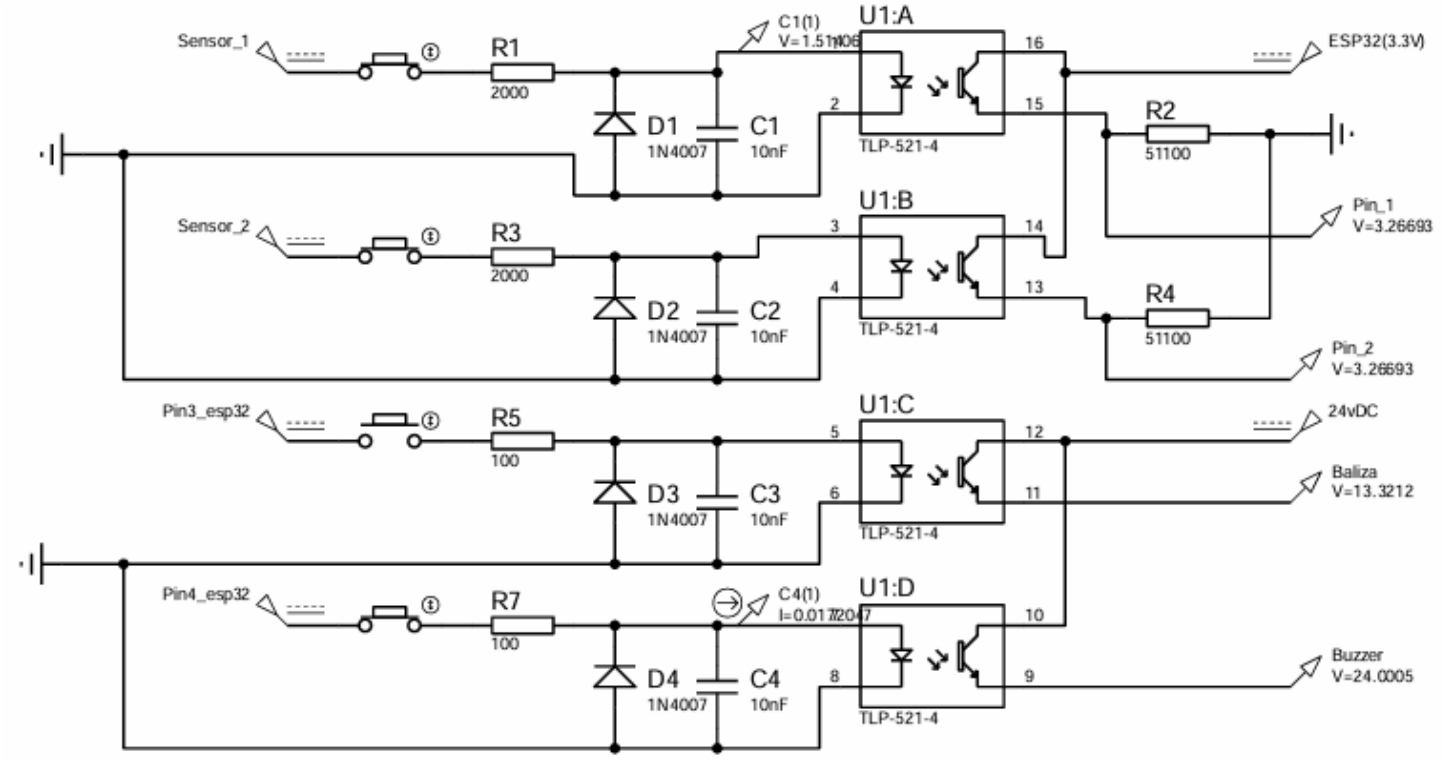
Revisión: A 06/2025		Fecha	Nombre y apellidos	Firma:	Escuela Universitaria Politécnica - La Almunia, Centro adscrito, Universidad Zaragoza
	Dibujado:	06/2025	Jorge Lasarte Aneiros		
	Comprobado:	06/2025	Tribunal Mecatrónica		
ESCALA: 1:10	ESTACIÓN DE VISIÓN ARTIFICIAL LOWCOST ESTRUCTURA DE LA ESTACIÓN DE VISIÓN			HOJA: 2/3	CÓDIGO DE PLANO: 424.23.04.0L.02

A4

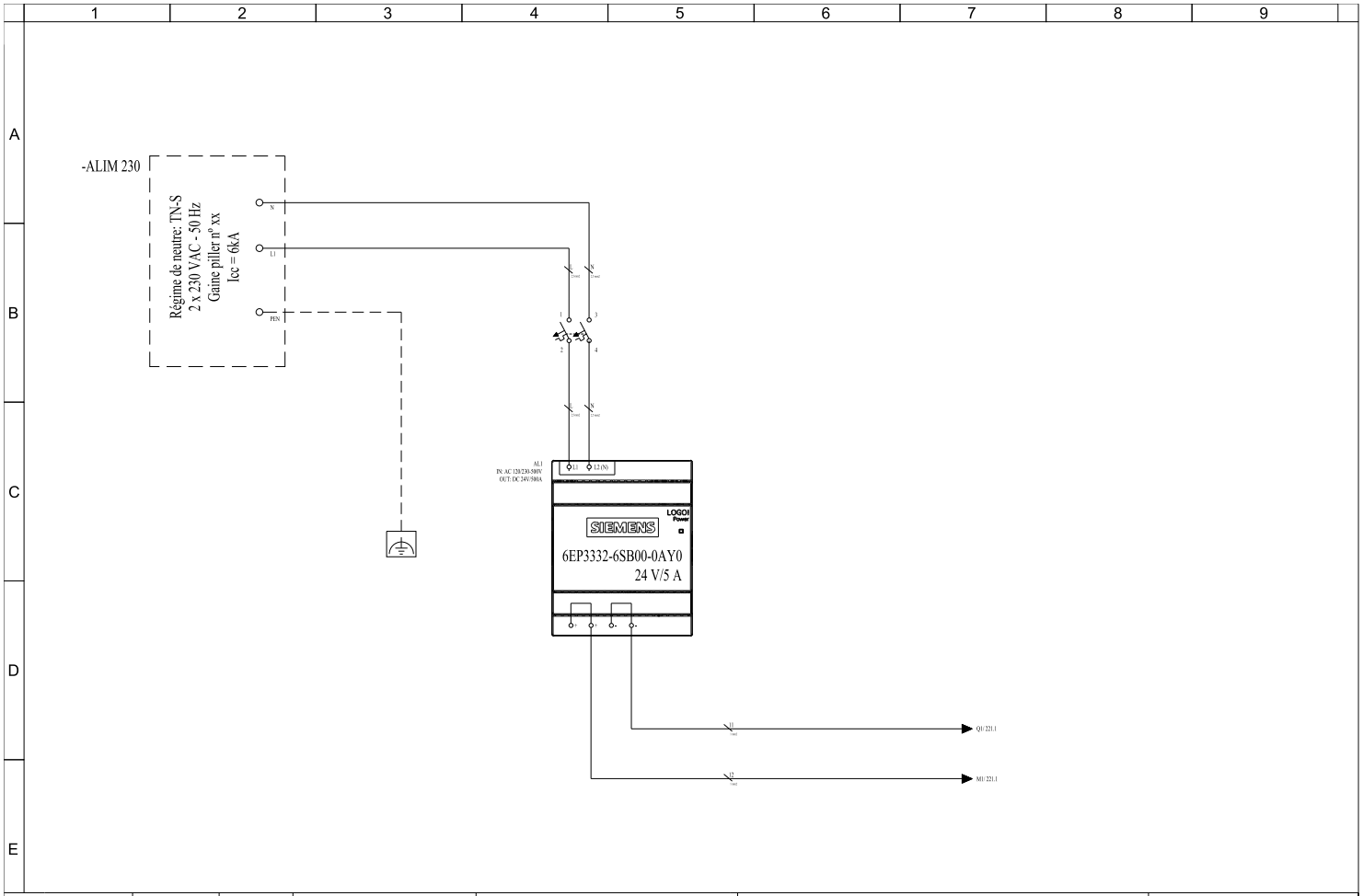


Revisión:		Fecha	Nombre y apellidos	Firma:	Escuela Universitaria Politécnica - La Almunia, Centro adscrito, Universidad Zaragoza
A	Dibujado:	06/2025	Jorge Lasarte Aneiros		
06/2025	Comprobado:	06/2025	Tribunal Mecatrónica		
ESCALA:	ESTACIÓN DE VISIÓN ARTIFICIAL LOWCOST			HOJA:	CÓDIGO DE PLANO:
1:10	ESTRUCTURA DE LA ESTACIÓN DE VISIÓN			3/3	424.23.04.0L.03

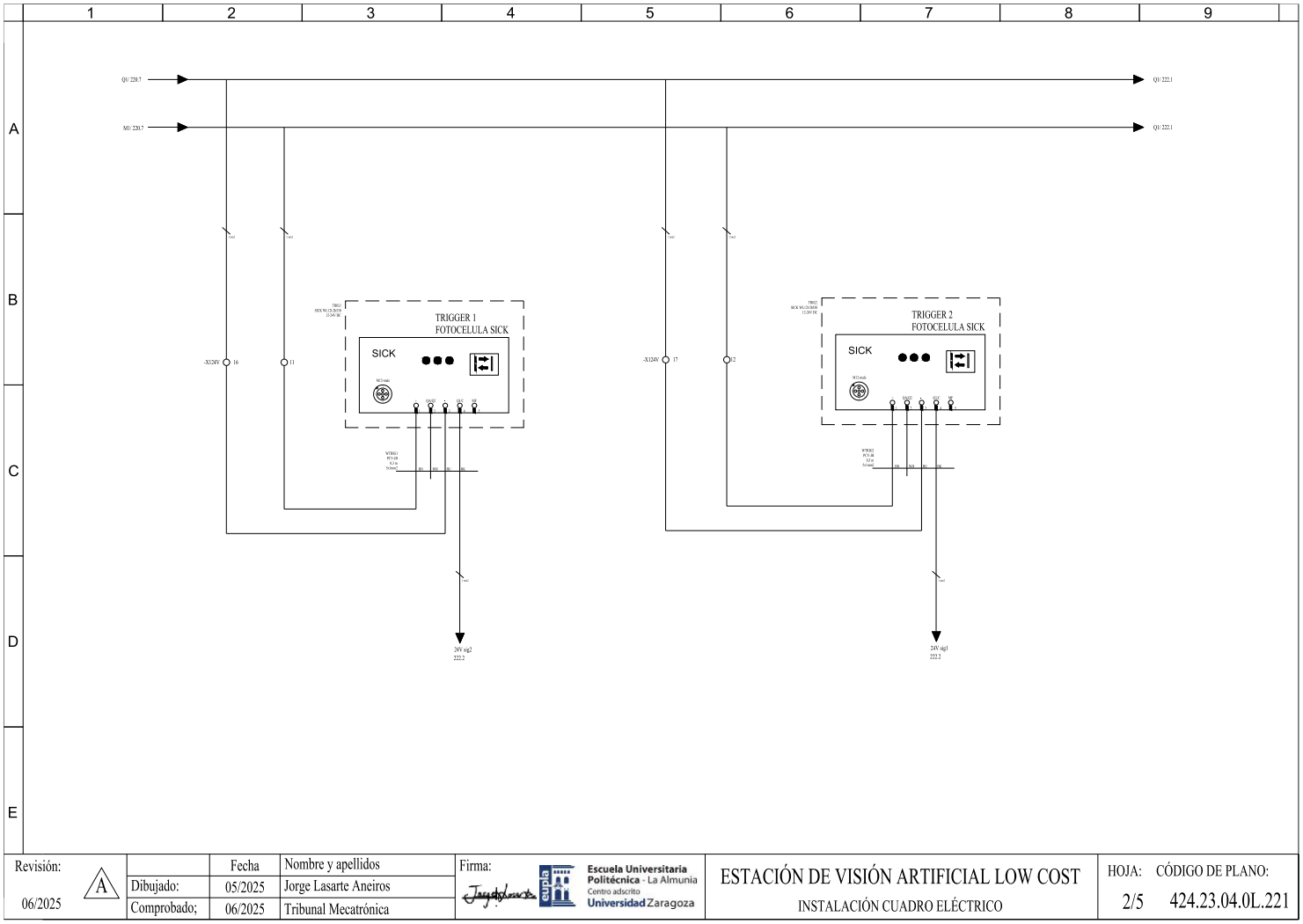
8.5. ANEXOS E: PLANO CIRCUITO PCB



8.6. ANEXOS F: PLANO ELÉCTRICO DE LA ESTACIÓN



Revisión: 06/2025	A		Fecha	Nombre y apellidos	Firma:	ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST	HOJA: 1/5
		Dibujado:	05/2025	Jorge Lasarte Aneiros		INSTALACIÓN CUADRO ELÉCTRICO	CÓDIGO DE PLANO: 424.23.04.0L.220
		Comprobado;	06/2025	Tribunal Mecatrónica			

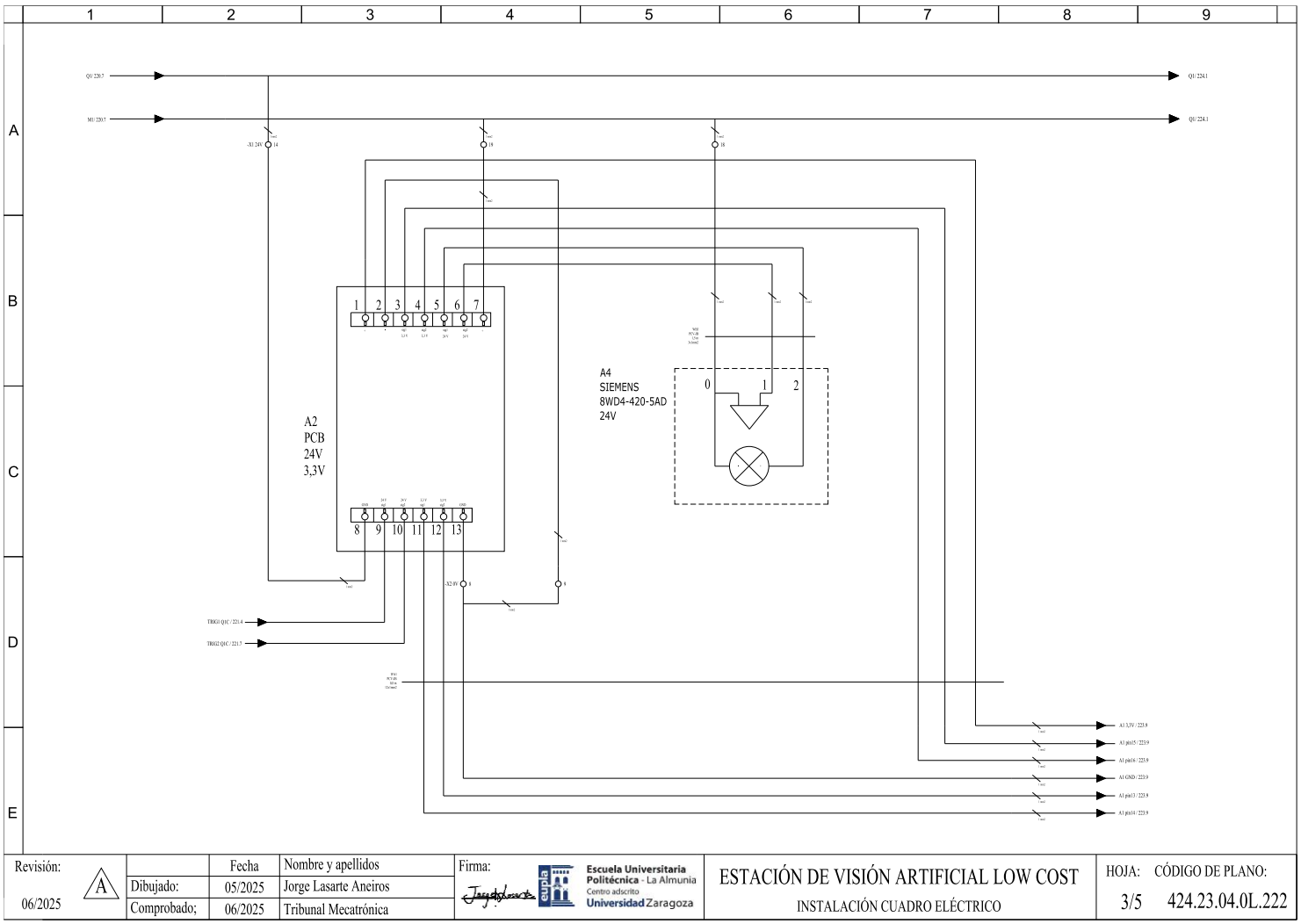


Revisión: 06/2025	A		Fecha	Nombre y apellidos	Firma:
		Dibujado:	05/2025	Jorge Lasarte Aneiros	
		Comprobado;	06/2025	Tribunal Mecatrónica	

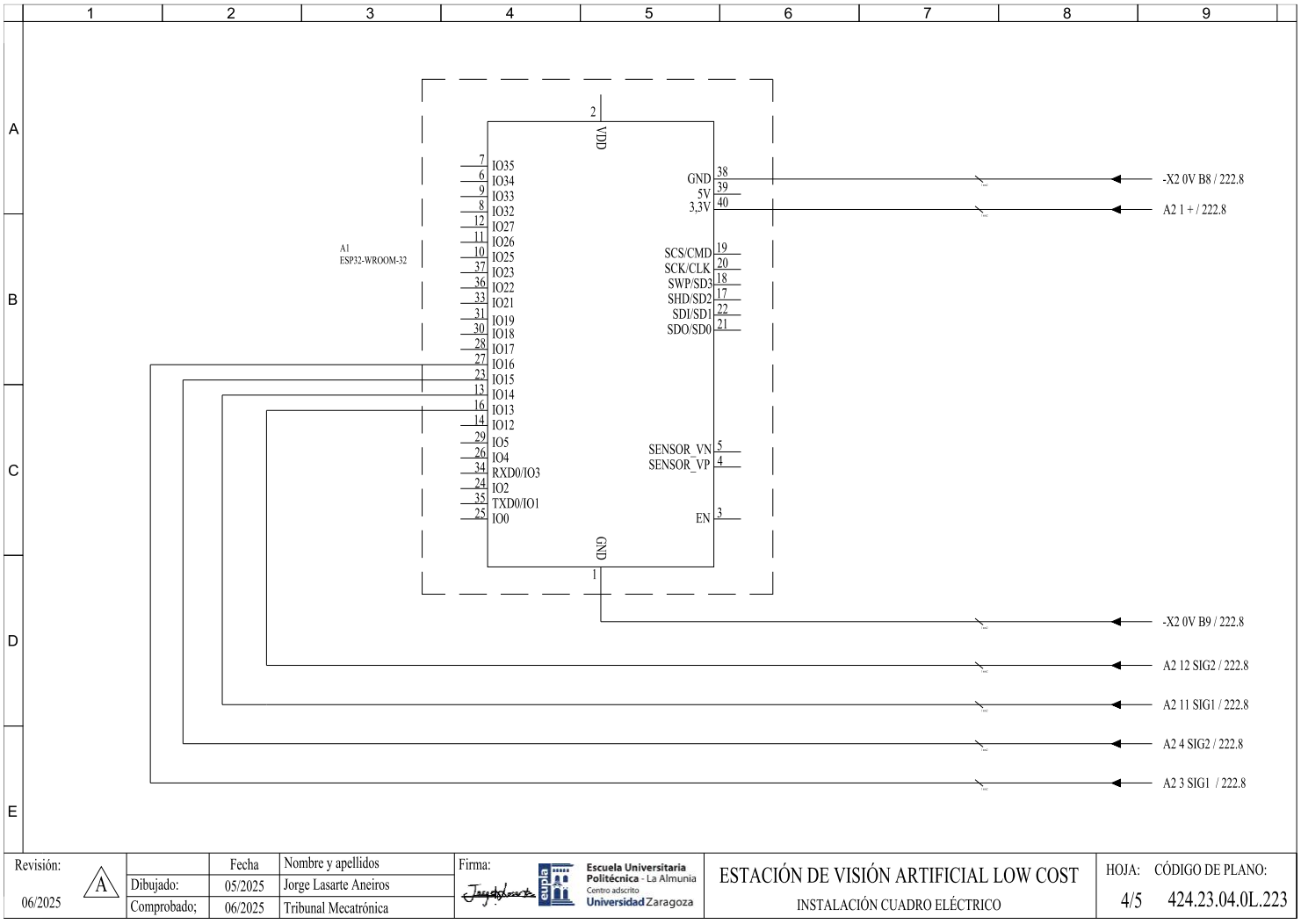
Escuela Universitaria Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST
INSTALACIÓN CUADRO ELÉCTRICO

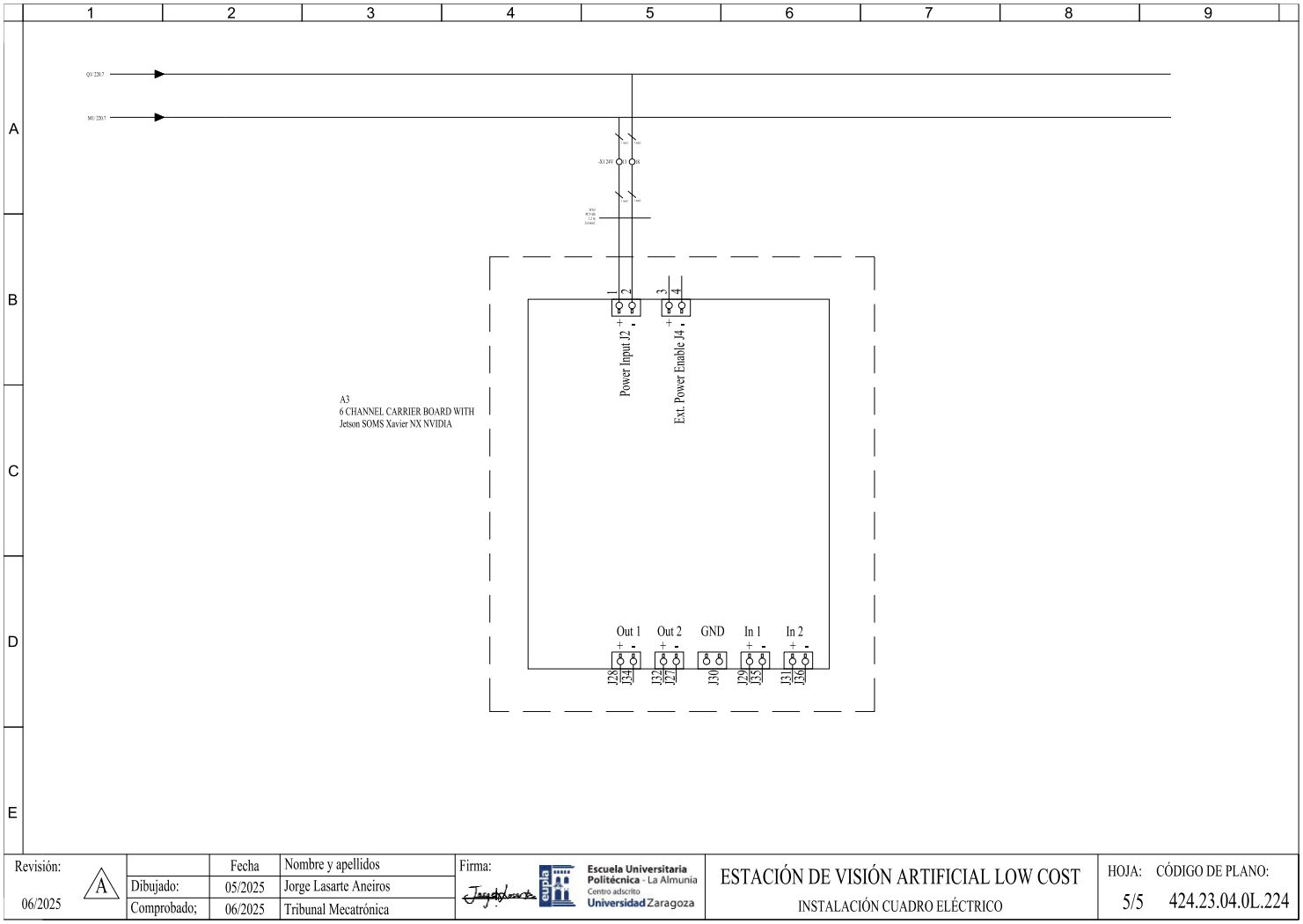
HOJA: 2/5 CÓDIGO DE PLANO: 424.23.04.0L.221



Revisión: 06/2025	A		Fecha	Nombre y apellidos	Firma:	ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST	HOJA: 3/5
		Dibujado:	05/2025	Jorge Lasarte Aneiros		INSTALACIÓN CUADRO ELÉCTRICO	CÓDIGO DE PLANO: 424.23.04.0L.222
		Comprobado;	06/2025	Tribunal Mecatrónica			

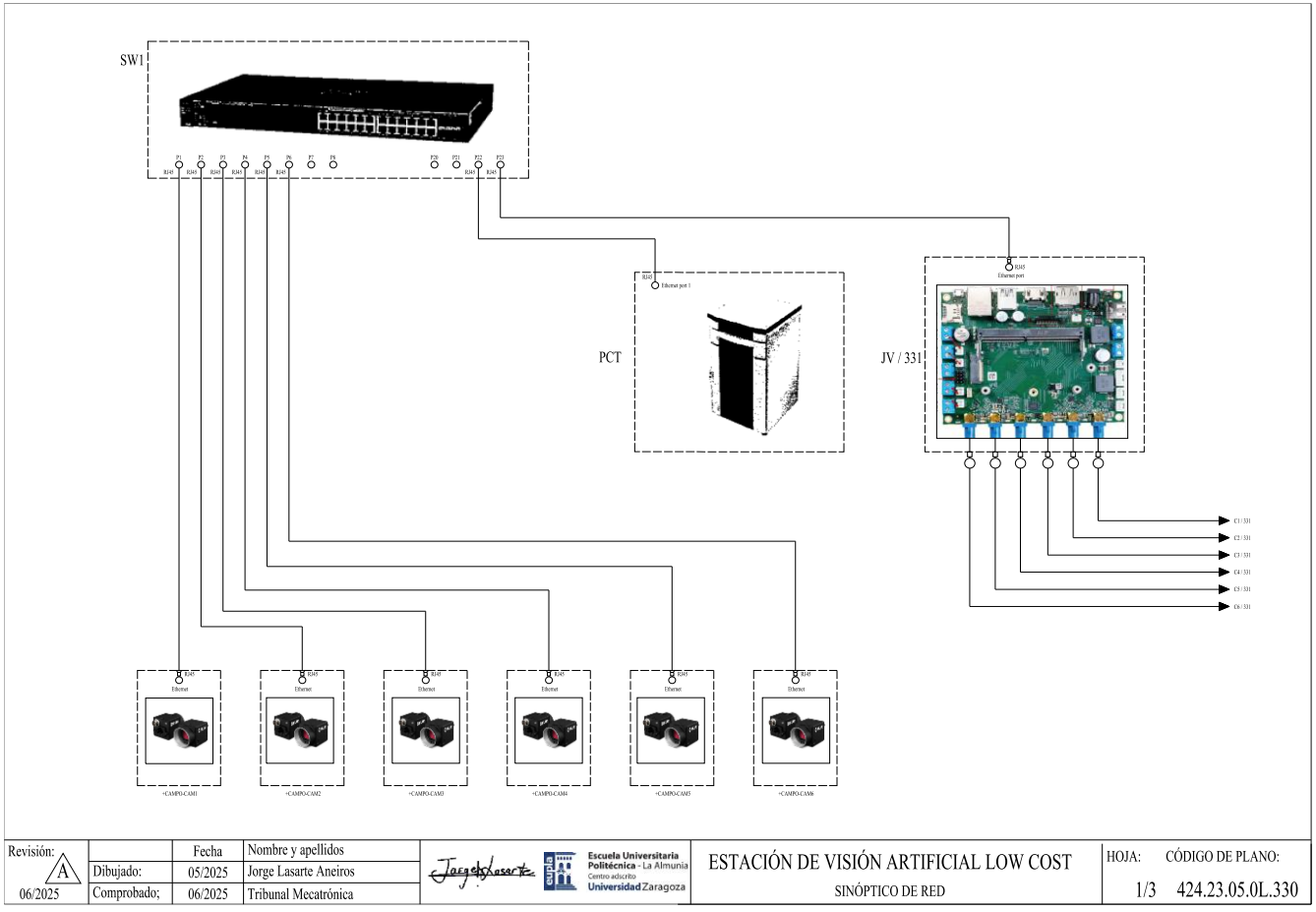


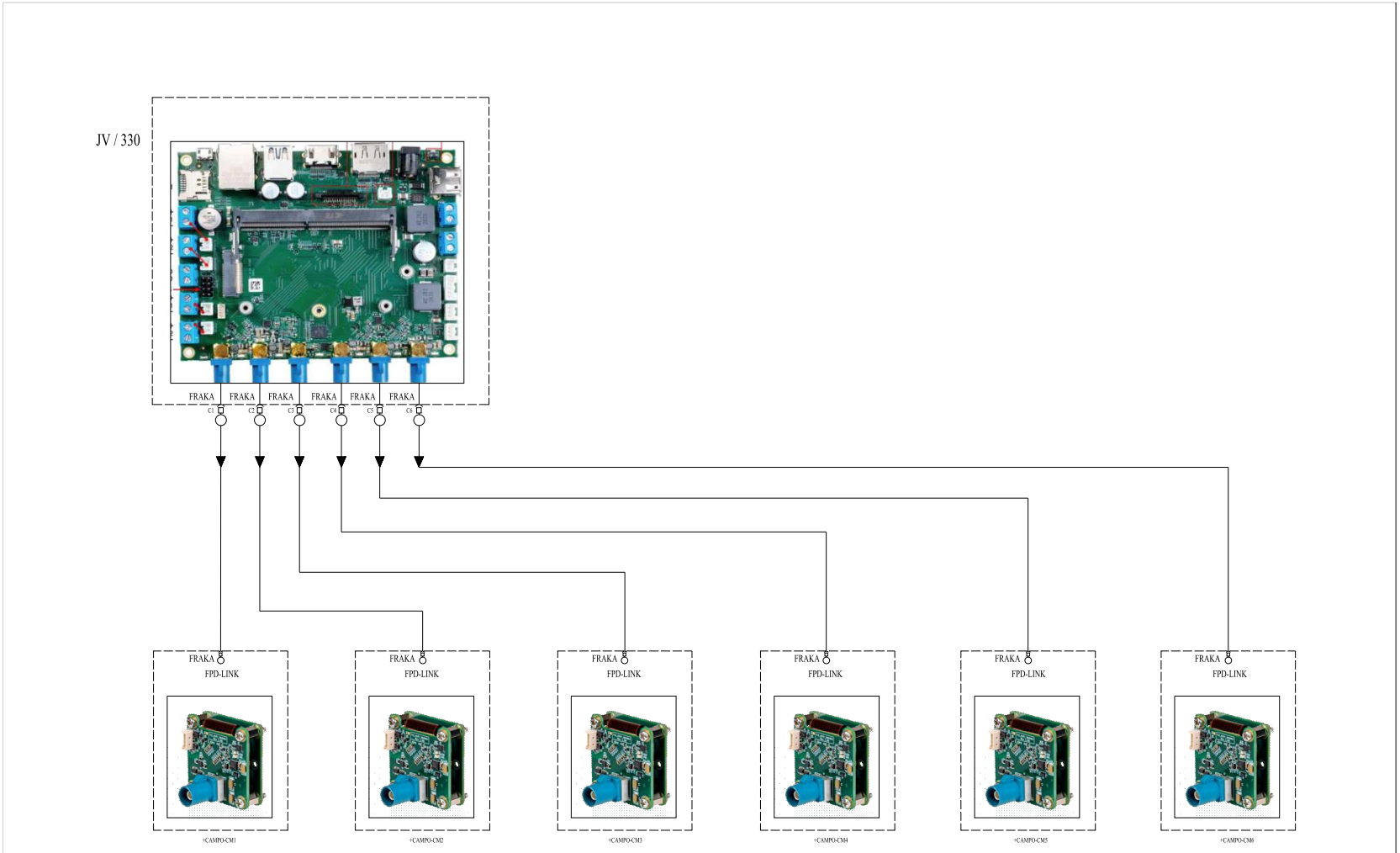
Revisión: 06/2025	A		Fecha	Nombre y apellidos	Firma:	ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST	HOJA: 4/5
		Dibujado:	05/2025	Jorge Lasarte Aneiros	Escuela Universitaria Politécnica - La Almunia, Centro adscrito Universidad Zaragoza	INSTALACIÓN CUADRO ELÉCTRICO	CÓDIGO DE PLANO: 424.23.04.0L.223
		Comprobado;	06/2025	Tribunal Mecatrónica			



Revisión: 06/2025	A		Fecha	Nombre y apellidos	Firma:	ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST	HOJA: 5/5
		Dibujado:	05/2025	Jorge Lasarte Aneiros		INSTALACIÓN CUADRO ELÉCTRICO	CÓDIGO DE PLANO: 424.23.04.0L.224
		Comprobado;	06/2025	Tribunal Mecatrónica			

8.7. ANEXOS G: PLANO SINÓPTICO DE RED DE LA ESTACIÓN





Revisión: A 06/2025		Fecha	Nombre y apellidos	Escuela Universitaria Politécnica - La Almunia, Centro adscrito, Universidad Zaragoza	ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST	HOJA:	CÓDIGO DE PLANO:
	Dibujado:	05/2025	Jorge Lasarte Aneiros		SINÓPTICO DE RED	2/3	424.23.05.0L.331
	Comprobado;	06/2025	Tribunal Mecatrónica				

Indentificador de medios de explotación	Cantidad	Designación	Número de tipo	Proveedor	Número de artículo
SW1	1	Switch D-LINK 24 puertos DGS-1024D 100Mb/s		D-LINK	
PCT	1	PC industrial		PC Componentes	
JV	1	6 Channel Carrier Board con Jetson SomS + Xavier NX		The Imagin Source	
+CAMPO-CAM1	1	Cámara monocromática de 2Mpx, 1600x1200, 50fps	BFLY-PGE-20E4M-CS	Teledyne FLIR	BFLY-PGE-20E4M-CS
+CAMPO-CAM2	1	Cámara monocromática de 2Mpx, 1600x1200, 50fps	BFLY-PGE-20E4M-CS	Teledyne FLIR	BFLY-PGE-20E4M-CS
+CAMPO-CAM3	1	Cámara monocromática de 2Mpx, 1600x1200, 50fps	BFLY-PGE-20E4M-CS	Teledyne FLIR	BFLY-PGE-20E4M-CS
+CAMPO-CAM4	1	Cámara monocromática de 2Mpx, 1600x1200, 50fps	BFLY-PGE-20E4M-CS	Teledyne FLIR	BFLY-PGE-20E4M-CS
+CAMPO-CAM5	1	Cámara monocromática de 2Mpx, 1600x1200, 50fps	BFLY-PGE-20E4M-CS	Teledyne FLIR	BFLY-PGE-20E4M-CS
+CAMPO-CAM6	1	Cámara monocromática de 2Mpx, 1600x1200, 50fps	BFLY-PGE-20E4M-CS	Teledyne FLIR	BFLY-PGE-20E4M-CS
+CAMPO-CM1	1	Cámara color de 2,1Mpx, 1920x1080, 60fps	DFM 36CX290	The Imagin Source	DFM 36CX290
+CAMPO-CM2	1	Cámara color de 2,1Mpx, 1920x1080, 60fps	DFM 36CX290	The Imagin Source	DFM 36CX290
+CAMPO-CM3	1	Cámara color de 2,1Mpx, 1920x1080, 60fps	DFM 36CX290	The Imagin Source	DFM 36CX290
+CAMPO-CM4	1	Cámara color de 2,1Mpx, 1920x1080, 60fps	DFM 36CX290	The Imagin Source	DFM 36CX290
+CAMPO-CM5	1	Cámara color de 2,1Mpx, 1920x1080, 60fps	DFM 36CX290	The Imagin Source	DFM 36CX290
+CAMPO-CM6	1	Cámara color de 2,1Mpx, 1920x1080, 60fps	DFM 36CX290	The Imagin Source	DFM 36CX290

Revisión:		Fecha	Nombre y apellidos		ESTACIÓN DE VISIÓN ARTIFICIAL LOW COST	HOJA:	CÓDIGO DE PLANO:
A	Dibujado:	05/2025	Jorge Lasarte Aneiros	Escuela Universitaria Politécnica - La Almunia, Centro adscrito, Universidad Zaragoza	SINÓPTICO DE RED	3/3	424.23.05.0L.332
06/2025	Comprobado;	06/2025	Tribunal Mecatrónica				

9. BIBLIOGRAFÍA

AliExpress. (2024a, May 6). *Módulo ESP32-WROOM-32 WiFi + Bluetooth*. https://es.aliexpress.com/item/1005006456519790.html?spm=a2g0o.productlist.main.1.2ad57305mBCHu2&aem_p4p_detail=20250616101845842731412788872000050238 3&algo_pvid=e76d216e-6906-4dc8-bbc5-a947623e9244&algo_exp_id=e76d216e-6906-4dc8-bbc5-a947623e9244-0&pdp_ext_f=%7B%22order%22%3A%2211221%22%2C%22eval%22%3A%221%22%7D&pdp_npi=4%40dis%21EUR%2111.95%212.84%21%21%2196.72%2123.04%21%402103868817500943257313300e0642%21120000372653173 61%21sea%21ES%210%21ABX&curPageLogUid=15rzkhMdTVYh&utparam-url=scene%3Asearch%7Cquery_from%3A&search_p4p_id=20250616101845842731412788872000050238 3_1#nav-description

AliExpress. (2024b, May 12). *Cable de red Ethernet RJ45 Cat6 blindado (deleyCON)*. https://www.amazon.es/deleyCON-Gigabit-apantallado-contactos-conectores/dp/B01LLOH4VQ?th=1

De Nova Guerrero, A. (2021). *Trabajo Fin de Máster Detección y segmentación de objetos en imágenes panorámicas Object detection and segmentation on panoramic images*.

D-Link. (2023). *Switch D-Link 24 puertos 10/100/1000Mbps DGS-1024D*. https://www.dlink.com/es/es/products/dgs-1024d-24-port-copper-gigabit-switch

Duan, X. (2018). *Python Machine Learning from Scratch*. Tsinghua University Press. http://www.wqbook.com

Durán Suárez, J. (2017). *Redes Neuronales Convolucionales en R. Reconocimiento de caracteres escritos a mano*.

FUJIFILM. (2024). *FUJIFILM Serie HF-HA-1S*. https://www.fujifilm.com/es/es-es/business/optical-devices/mvlens/hfha

Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). *Deep Learning*.

López Perez, H. (2021). *Visión Artificial Aplicada A La Detección de Personas*.

Martin Romero, M. (2022). *Reconocimiento de objetos mediante técnicas de visión por computador y aprendizaje automático*.

Murphy, K. P. . (2012). *Machine learning : a probabilistic perspective*. MIT Press.

Navarro, S. (2024, June 12). *Tipos de Machine Learning*. https://keepcoding.io/blog/tipos-de-machine-learning/

NVIDIA. (2024). *NVIDIA Jetson Xavier NX Module*. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/

Paletti Profilsysteme GmbH & Co. KG. (2023, May 12). *Precision Technology – Aluminum Profile Systems*. https://paletti-group.com/wp-content/uploads/PrecisionTechnology.pdf

Pérez-Aguilar, D. A., Pérez-Aguilar, J. M., Pérez-Aguilar, A. P., Risco-Ramos, R. H., & Malpica-Rodríguez, M. E. (2024). Electric substation inspection: YOLOv5 in hotspot detection through thermal imaging. *Ingenius*, *2024*(31), 43–54. https://doi.org/10.17163/ings.n31.2024.04

Raimon Blanes, S. (2022). *Procesado de Imagen con deep learning*.

SICK AG. (2024). *Sensor fotoeléctrico WL12L-2B530S06*. SICK España. https://www.sick.com/es/es/catalog/productos/sensores-de-deteccion/fotocelulas/w12/wl12l-2b530s06/p/p300644

Silva Guzmán, E. R. (2020). *Entrenamiento de la Red Neuronal Convolucional YOLO para objetos propios*.

Suat Rojas, N. E., Montoya Serna, B. S., Pinzón Velásquez, E. M., & Rodríguez Galeano, O. S. (2021). Reconocimiento del abecedario de la lengua de señas colombiana con Redes Neuronales Convolucionales. *Orinoquia*, *25*(1), 25–30. https://doi.org/10.22579/20112629.680

Teledyne Vision Solutions. (2024). *Teledyne FLIR. (2024). Blackfly GigE BFLY-PGE-20E4M-CS*. https://www.teledynevisionsolutions.com/es-es/products/blackfly-gige/?model=BFLY-PGE-20E4M-CS&vertical=machine+vision&segment=iis&docTypes=technicalreference&docPage=1

The Imaging Source. (2024a). *6 Channel Carrier Board – FPD-Link III*. https://www.theimagingsource.com/en-us/embedded/carrier/6-chan-fpd-link-iii/

The Imaging Source. (2024b). *Cables FAKRA para placas embebidas conexión FPD-Link III*. https://www.theimagingsource.com/en-us/embedded/accessory/fakra-cable/

The Imaging Source. (2024c). *The Imaging Source. (2024). DFM 36CX290-ML – FPD-Link III camera*. https://www.theimagingsource.com/en-us/embedded/fpd-link-iii/36c-board/dfm36cx290ml/

VITRONIC. (2023). La IA en la inspección de calidad óptica. *VITRONIC*. https://www.vitronic.com/es-es/blog/produccion-inteligente/la-ia-en-la-inspeccion-de-calidad-optica

Zhou, Z. H. (2021). Machine Learning. In *Machine Learning*. Springer Nature. https://doi.org/10.1007/978-981-15-1967-3

Relación de documentos

(X) Memoria 110 páginas

(_) Anexos 16 páginas

La Almunia, a 29 de Junio de 2025

Firmado: Jorge Lasarte Aneiros