UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Mecatrónica

Algoritmo de visión artificial para el mantenimiento preventivo de un equipo de impresión 3D

Trabajo de grado previo a la obtención del título de Ingeniero en Mecatrónica

Autor:

Apuango Morales Alexander Dario

Director:

Ing. Ojeda Peña David Alberto PhD

Ibarra - Ecuador
2024

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	172738715-9		
APELLIDOS Y NOMBRES:	Apuango Morales Alexander Dario		
DIRECCIÓN:	Simón Bolívar y 24 de mayo, Atuntaqui, Andrade Marín		
EMAIL:	adapuangom@utn.edu.ec		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0998737694

DATOS DE LA OBRA	
TÍTULO:	Algoritmo de visión artificial para el mantenimiento preventivo de un equipo de impresión 3D
AUTOR (ES):	Apuango Morales Alexander Dario
FECHA DE APROBACIÓN: DD/MM/AAAA	22/01/2024
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERO EN MECATRÓNICA
ASESOR /DIRECTOR:	Ing. David Ojeda Peña, PhD / Ing. Iván García Santillán, PhD

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 22 días del mes de enero de 2024

EL AUTOR:

(Firma).....

Nombre: Apuango Morales Alexander Dario

Universidad Técnica del Norte

Facultad de Ingeniería en Ciencias Aplicadas

CERTIFICACIÓN DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

En mi calidad de director del trabajo de grado "Algoritmo de visión artificial para el mantenimiento preventivo de un equipo de impresión 3D", presentado por el egresado Apuango Morales Alexander Dario, que opta por el título de ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, 22 de enero de 2024

Ing. David Alberto Ojeda Peña, PhD

Director de Tesis

Universidad Técnica del Norte

Facultad de Ingeniería en Ciencias Aplicadas

APROBACIÓN DEL COMITÉ CALIFICADOR

El Tribunal Examinador del trabajo de titulación. Algoritmo de visión artificial para el mantenimiento preventivo de un equipo de impresión 3D. Elaborado por Alexander Dario Apuango Morales, previo a la obtención del título de Ingeniero en Mecatrónica, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

Ing. David Alberto Peña Ojeda PhD

Director de Tesis

Ing. Iván Danilo García Santillán PhD

Asesor de Tesis

Dedicatorias

A Miguel Apuango y Margarita Morales, mis queridos padres, cuyo apoyo incondicional ha sido la fuerza que guía mi travesía académica. Vuestra dedicación y amor constante han sido la luz que ilumina cada paso de este camino. En cada desafío, sus palabras sabias y su aliento han sido mi ancla, recordándome que el camino hacia el conocimiento está pavimentado con determinación y apoyo familiar.

A mi apreciada hermana, Gaby, quien siempre me ha alentado a seguir adelante con valentía y determinación. Tu apoyo ha sido un pilar fundamental en mi camino hacia el logro académico. En los momentos de duda, tus palabras alentadoras han sido un recordatorio poderoso de que cada esfuerzo vale la pena y que la perseverancia conduce al éxito.

A mi fiel compañero, Cocky, cuyo amor incondicional y compañía han sido un bálsamo en las largas noches de estudio. Tu presencia silenciosa ha convertido las horas de investigación en momentos más ligeros y reconfortantes. En cada rincón de este logro, dejo huellas de gratitud por tu lealtad y afecto.

A mis amigos, cómplices de risas y testigos de nuestras experiencias compartidas. Su amistad ha enriquecido mi viaje académico, convirtiendo los desafíos en oportunidades para crecer juntos.

En este viaje, encontré en sus palabras y gestos el impulso necesario para superar desafíos y alcanzar metas. Este trabajo de tesis es, en parte, un tributo a ustedes, quienes han sido mi inspiración y apoyo constante. Con gratitud profunda, dedico este logro a la familia que ha sido mi roca y mi fuente de fortaleza. Su amor ha sido el cimiento sobre el cual he construido mi educación y mi futuro.

Agradecimientos

En el culminar de este viaje académico, deseo expresar mi sincero agradecimiento a aquellos que han sido pilares fundamentales en la realización de este trabajo de titulación. A mi familia, fuente inagotable de apoyo incondicional. Cada paso que he dado en este camino ha sido respaldado por su amor, dedicación y aliento constante. Su inquebrantable respaldo ha sido el cimiento sobre el cual he construido mi educación y mi futuro. A mis amigos, Javier y Anderson, quienes han sido amigos leales en cada etapa de este viaje. Atravesamos juntos no solo los buenos momentos, sino también los desafíos, y su amistad ha sido una fuente inigualable de fortaleza. A la Universidad Técnica del Norte, por proporcionar el entorno propicio para mi formación académica. Cada aula, cada desafío académico, ha sido una oportunidad para crecer y aprender. Esta institución ha sido un hogar intelectual donde las semillas del conocimiento han germinado y florecido. A mi asesor de tesis, el PhD David Ojeda, agradezco su conocimiento, paciencia y consejos sabios que han sido la brújula que guió este proyecto hacia su realización. Su dedicación y experiencia han sido la chispa que encendió mi pasión por la investigación y para ser mejor cada día. A todos aquellos que, de alguna manera, contribuyeron a este logro, les expreso mi profundo agradecimiento. Cada interacción, cada palabra de estímulo, ha dejado una huella imborrable en mi travesía académica. Este trabajo no es solo el resultado de esfuerzo individual, sino el producto de un tejido de apoyo, conocimiento compartido y amistades duraderas. Les dedico este logro con gratitud sincera.

Índice general

Índice de figuras

Índice de tablas

Resumen

El presente trabajo de grado se centra en el desarrollo de un sistema innovador para el monitoreo en tiempo real de un equipo de impresión 3D, la investigación comprende tres componentes esenciales: detección de objetos, seguimiento ocular y comunicación IoT. En la fase de detección de objetos, se implementó una red neuronal convolucional, específicamente la RCNN Faster Inception V2, entrenada con un conjunto de datos compuesto por imágenes de tres equipos: un torno, una fresadora y una impresora 3D, el modelo permite identificar y distinguir los equipos en tiempo real, lo que constituye un avance fundamental. El seguimiento ocular se logra mediante el uso de un predictor, que permite identificar que equipo está siendo observado por el usuario, facilitando la interacción y el monitoreo específico. La comunicación IoT se establece através del protocolo MQTT, conectando sensores de temperatura y humedad ambiental, así como un sensor de temperatura y vibración en la impresora 3D, los datos de estos sensores se trasmiten en tiempo real a través de una red IoT, lo que proporciona información vital para el mantenimiento preventivo. Una interfaz de usuario desarrollada con PyQt5 integra la información de la visión por computadora y los datos de los sensores, brindando al usuario una visión completa del estado de los equipos y sus sensores en tiempo real. Este sistema no solo representa un avance tecnológico significativo en la gestión y mantenimiento de equipos de impresión 3D, sino que también demuestra la sinergia entre la visión artificial, el seguimiento ocular y la comunicación IoT. Los resultados de las pruebas confirman la eficacia y la viabilidad de esta solución en entornos de fabricación. Este enfoque innovador tiene el potencial de mejorar la eficiencia y la confiabilidad en la operación de impresoras 3D y puede aplicarse a otras áreas de la industria con necesidades de monitoreo en tiempo real.

Palabras clave: Mantenimiento preventivo, Visión artificial, Detección de objetos, Seguimiento ocular, Protocolos IoT.

Abstract

This degree work focuses on the development of an innovative system for real-time monitoring of a 3D printing equipment, the research comprises three essential components: object detection, eye tracking and IoT communication. In the object detection phase, a convolutional neural network was implemented, specifically the RCNN Faster Inception V2, trained with a dataset composed of images from three pieces of equipment: a lathe, a milling machine and a 3D printer, the model allows to identify and distinguish the equipment in real time, which is a fundamental advance. Eye tracking is achieved through the use of a predictor, which identifies which equipment is being observed by the user, facilitating interaction and specific monitoring. IoT communication is established through the MQTT protocol, connecting environmental temperature and humidity sensors, as well as a temperature and vibration sensor on the 3D printer; data from these sensors is transmitted in real time through an IoT network, providing vital information for preventive maintenance. A user interface developed with PyQt5 integrates computer vision information and sensor data, giving the user a complete view of the status of the equipment and its sensors in real time. This system not only represents a significant technological breakthrough in the management and maintenance of 3D printing equipment, but also demonstrates the synergy between computer vision, eye tracking and IoT communication. Test results confirm the effectiveness and feasibility of this solution in manufacturing environments. This innovative approach has the potential to improve efficiency and reliability in the operation of 3D printers and can be applied to other areas of industry with real-time monitoring needs.

Keywords: Preventive maintenance, Machine vision, Object detection, Eye tracking, IoT protocols.

Introducción

Problema

El mantenimiento es un factor crucial para asegurar el correcto funcionamiento de las empresas, ya que permite prever posibles riesgos y fallos de los equipos y maquinarias que se utilizan en las tareas diarias, lo que es esencial para cumplir con los plazos establecidos. Cuando algo no funciona adecuadamente, el mantenimiento se encarga de resolver el problema, pero si además el proceso es efectivo al detectar riesgos anticipadamente, puede mejorar significativamente el rendimiento de la empresa. De esta forma, el mantenimiento se convierte en un elemento clave para prevenir problemas y optimizar el desempeño empresarial [1].

La mayoría de los trabajos de mantenimiento se ralentizan por procesos manuales y, debido a que son repetitivos y tediosos, aumentan la posibilidad de ocasionar errores, esto afecta también a la eficiencia de los empleados y equipos dentro de una empresa. Cuando las tareas se realizan manualmente, los datos precisos no están disponibles y, si las decisiones se toman en función de estos datos, la mayoría de las posibilidades apuntarán a que no se obtendrán los resultados esperados. [2].

Para la gestión de mantenimiento, el seguimiento y monitoreo de un equipo es fundamental para garantizar el correcto funcionamiento por lo que en este trabajo de investigación se propone abordar el problema en tiempos de mantenimiento y monitoreo de una máquina, mediante el uso de visión artificial con “eye tracking”. Por lo que se propone el desarrollo de un algoritmo capaz de dar información en tiempo real para realizar el mantenimiento preventivo de un equipo.

Objetivos

Objetivos generales

- Desarrollar un algoritmo para el mantenimiento preventivo en un equipo de impresión 3D utilizando visión artificial.

Objetivos Específicos

- Clasificar los modelos de detección de objetos basados en inteligencia artificial.
- Desarrollar una base de datos para almacenar la información recolectada y la incorporación del "eye tracking" con la detección de objetos.
- Evaluar el correcto funcionamiento del algoritmo realizado para el sistema.

Alcance

La investigación se limita a la realización de un algoritmo para un proceso de automatización, capaz de brindar información en tiempo real para monitoreo de un equipo, mediante protocolos IoT. Esto se llevará a cabo por medio de algoritmos con librerías propias del lenguaje de programación Python.

La visión artificial permitirá la captación, procesamiento y el análisis de imágenes, mediante el aprendizaje automático y redes neuronales, incrementando la eficacia de actuación y reduciendo costos. Se realizará varios algoritmos como: El del "Deep Learning", que generará redes neuronales artificiales con comportamiento similar a las redes neuronales biológicas. Para lograr esto, el algoritmo de clasificación utilizado escaneará pequeñas regiones de la imagen en lugar de la imagen completa.

Para el entrenamiento de la red neuronal se hará uso de un "dataset" organizado por carpetas y etiquetadas de forma manual, para llegar a esto se necesita la obtención y etiquetado de imágenes que posteriormente pasarán a una validación mediante scripts de las librerías propias de Python, las imágenes obtenidas para el entrenamiento serán de una impresora 3D, una vez validado este proceso se continuará con la exportación del modelo entrenado, se hará uso de una matriz de confusión para comprobar el rendimiento y funcionamiento adecuado del algoritmo.

La creación de la base de datos servirá para recopilar y organizar información, la cual almacenará dicha información acerca de los componentes de un equipo, para respaldar y obtener

datos de los dispositivos a reconocer, que posteriormente ayudará a compartir estos valores mediante protocolos IoT, estos protocolos permitirán el entendimiento e intercambio de información, facilitando la comunicación “Machine2Machine” (M2M) en tiempo real, con la creación de otro algoritmo este permitirá emitir un reporte de las variables registradas en la base de datos y las magnitudes físicas captadas mediante sensores implementados en el equipo.

Al usar el “eye tracking”, combinado con la detección de objetos, permitirá enfocar un solo elemento y no varios a la vez, en un ambiente controlado, mostrando la información de un equipo en específico.

Al combinar todos estos parámetros se obtendrá el algoritmo final cuyo funcionamiento será el de brindar información y especificaciones del equipo seleccionado, además los datos captados mediante los sensores implementados y los registrados en la base de datos.

El algoritmo final se implementará en una interfaz gráfica de un computador, que al momento de captar el equipo con la cámara del celular, mediante “eye tracking” emitirá el reporte en tiempo real del equipo seleccionado.

Capítulo I

Marco referencial

1.1. Antecedentes

En 2018, Lanzarini et al, desarrollaron una plataforma de detección de objetos en tiempo real para solucionar problemas complejos, implementando técnicas de visión artificial y herramientas de software libre, utilizando modelos pre-entrenados que están disponibles y preparados para reconocer objetos, se configuró varias cámaras Ip como entrada, para definir un área y de esta manera, aplicar el proceso de detección, a través de un sistema de reglas condicionales, logrando ejecutar diversas acciones, como enviar una alarma, un mensaje instantáneo, ajustar una variable de control, entre otras, dependiendo de las necesidades y objetivos que se deseen accionar, con esta plataforma se definen zonas seguras e inseguras para el tránsito de personas, el reconocimiento de objetos permitidos y no permitidos en una cinta transportadora. En términos de seguridad, se establecen zonas y horarios específicos donde la detección de personas puede activar una alarma, o detectar y reconocer rostros autorizados o no autorizados [3].

Castelo y Mosquera, en 2018, implementaron un sistema para la detección y seguimiento de una persona utilizando un cuadricóptero mediante visión artificial, usando un dron modelo DJI Ryze Tello, para monitorear un ambiente al aire libre, con la ayuda de un algoritmo que combina una red neuronal (SSD MobilNet V2) para la detección de objetos y otro algoritmo de seguimiento (KCF) haciendo que el dron siga a la persona que porta un chaleco distintivo, los algoritmos se probaron en una Raspberry Pi de 4 de 8Gb para posteriormente comandar el dron, Durante la fase de investigación, se aplicó un enfoque de campo realizando pruebas al aire libre. Se emplearon vídeos breves de diez segundos en los que una persona aparecía, desaparecía y reaparecía detrás de un objeto. En estas pruebas, se compararon dos algoritmos de seguimiento, centrándose especialmente en la medición del tiempo de seguimiento. Los resultados de la com-

paración indicaron que el algoritmo KCF exhibió una velocidad superior en comparación con el otro algoritmo, evidenciándose mediante una mayor cantidad de fotogramas por segundo. [4].

En 2022, Ramírez y Toro, desarrollaron una aplicación móvil para detectar y reconocer las matrículas de las motocicletas. Para lograr esto, se utilizaron herramientas de desarrollo libres, como la librería de visión por computador OpenCV para procesar la imagen y localizar el área de la matrícula. También se usó Tesseract, una librería de reconocimiento óptico de caracteres (OCR), para identificar los caracteres de las matrículas. Para mejorar la precisión y exactitud en la detección de cascos y parrilleros, se utilizó el algoritmo de inteligencia artificial YOLO entrenado para la detección de objetos. Además, se compararon los resultados obtenidos en la detección de cascos con los obtenidos mediante la transformada circular de Hough. En las pruebas realizadas se obtuvo una tasa de precisión para la detección de placas fue del 77 %, mientras que la tasa de exactitud fue del 85,5 % [5].

Neisser y Junior, en 2019, aplicaron técnicas de visión artificial para detectar el estado fisiológico de los ojos en conductores, para ello se utilizaron algunas de las librerías más importantes de Python, como Numpy, Pandas, Scikit-learn y OpenCV, para el procesamiento y clasificación de imágenes. El conjunto de datos empleado comprende una cantidad de fotografías de personas con los ojos cerrados y abiertos, con ayuda de varios clasificadores se pudo observar que el rendimiento del clasificador Majority Voting fue superior en la detección del estado fisiológico del ojo izquierdo y derecho en su estado abierto o cerrado en comparación con los otros modelos, con un promedio de precisión media (mAP) del 93,39 %. Por otro lado, el modelo Random Forests obtuvo un mejor rendimiento en la detección del estado del ojo izquierdo, con una precisión del 93,60 %, mientras que Majority Voting obtuvo una precisión del 94,22 %. Por el contrario, el clasificador SVM tuvo un rendimiento inferior en comparación con los otros modelos, obteniendo la precisión más baja con un 91,75 % y 92,98 % para el ojo derecho e izquierdo, respectivamente [6].

En 2018, Martínez, creó un sistema de Visión Artificial interactivo que pudo reconocer y seguir los movimientos de los fisicoculturistas para determinar si sus posturas son correctas en tiempo real. Para esto, se hizo un algoritmo que compara la posición de las articulaciones del usuario con las preestablecidas en el sistema de ejes de tres dimensiones utilizando el dispositivo de captura de movimiento Kinect. La evaluación final del sistema se llevó a cabo en un entorno real donde se realizó una calibración del sensor para ajustar la altura, la posición y la iluminación para asegurar que el ambiente fuera adecuado para la prueba. El resultado final fue una herramienta computacional que ayuda a los fisicoculturistas a estimular de manera adecuada los músculos involucrados en cada ejercicio, lo que evitará las posibles consecuencias negativas a

corto y largo plazo que puedan surgir de una mala práctica [7].

Cuertán, en 2021, realizó una aplicación multimedia que utiliza realidad aumentada y se basa en el modelamiento digital de un sistema de suspensión, que proporciona información técnica y de mantenimiento a los usuarios. La aplicación se centra en la utilización de un sistema de suspensión específico y ofrece una experiencia interactiva. La realidad aumentada se usa para superponer elementos digitales sobre la vista en vivo de la cámara de un dispositivo móvil, lo que permite a los usuarios visualizar el sistema de suspensión de manera virtual y obtener información técnica sobre el mantenimiento del mismo, al implementarlo la aplicación proporcionó datos técnicos sobre los sistemas de suspensión, tales como el ángulo de salida, caída, convergencia, divergencia, fallos y diagnóstico [8].
En 2019, Gonzáles, creó un detector de baches mediante el uso de la tecnología de Deep Learning, con el objetivo de reducir los costos del mantenimiento de las carreteras y mejorar la seguridad de los usuarios. Para lograr este objetivo, se realizó un estudio exhaustivo de la tecnología de Deep Learning y su aplicación en la detección de objetos, seleccionando la plataforma más adecuada y entrenando una red neuronal para detectar imperfecciones en las carreteras. Se compararon diferentes modelos entrenados y se desarrolló un prototipo funcional de la aplicación en forma de APP.Los resultados obtenidos muestran un porcentaje de éxito y fracaso cercano al 50 %, también se ha observado que el número de predicciones verdaderas es bastante bajo y, en general, el resultado no cumple con nuestras expectativas. Para obtener mejores resultados el entrenamiento debe llevarse de manera completa, para evitar detecciones incorrectas [9].

En 2022, Gudiño et al, construyeron un dispositivo mecatrónico que permitió registrar el seguimiento ocular del ser humano para la manipulación de un brazo robótico de tres grados de libertad, y desarrollo de una interfaz gráfica hombre-máquina amigable para visualizar el ojo humano y los comandos necesarios para la manipulación del robot. Los resultados experimentales del dispositivo de seguimiento ocular desarrollado son muy satisfactorios, ya que se utilizó la técnica de video-oculografía para detectar el movimiento del ojo mediante una fuente de luz cercana al ojo humano. Además, se ha creado una interfaz hombre-computadora-robot lo que permite a cualquier persona manipular el software, el dispositivo de seguimiento ocular y el robot. Para demostrar la eficacia del prototipo, se realizaron diez pruebas con diferentes usuarios para evaluar la usabilidad del dispositivo de seguimiento ocular, obteniendo un 100 % de adaptación fácil al prototipo y a la interfaz gráfica [10].

Ibarracín, en 2022, desarrolló un sistema de realidad aumentada para mantenimiento correctivo industrial en el marco de la digitalización. La realidad aumentada es una tecnología que inte-

gra información digital con el entorno real de manera interactiva y complementaria. El sistema propuesto buscó mejorar la seguridad en el mantenimiento de maquinarias y procesos al proporcionar información relevante y actualizada de manera instantánea. Este sistema permite a los operarios escanear la maquinaria y desplegar etiquetas con información necesaria para solucionar inconvenientes de manera eficiente y con menor tiempo de respuesta. A pesar de que la realidad aumentada es una tecnología con amplia información disponible, no se suele utilizar en el área de mantenimiento correctivo industrial en Ecuador, lo que representa una oportunidad para fomentar nuevos métodos de trabajo y obtener resultados más eficientes [11].

En 2022, López, para mejorar el mantenimiento se implementó tecnologías de última generación, como la realidad aumentada, se utilizó el software Unity en conjunto con Vuforia, ya que son de código libre y ofrecen las mismas funcionalidades de Ecostruxture Mechanic de forma gratuita, al ejecutar el código permitió superponer información sobre imágenes captadas con una cámara, accesible a través de un dispositivo móvil en cualquier momento y lugar, proporcionando datos precisos y reales para el mantenimiento. La interfaz gráfica diseñada cumplió con las expectativas previstas, ya que permite visualizar en tiempo real las alertas generadas por el sistema de realidad aumentada a través de etiquetas, así como mostrar el estado de activación de cada elemento, lo que mejora la eficiencia en la respuesta ante una falla imprevista [12].

1.2. Marco teórico

1.2.1. Mantenimiento preventivo de máquinas

El mantenimiento predictivo, también conocido como mantenimiento basado en condición, consiste en examinar la integridad de cada uno de los componentes mecánicos o eléctricos que conforman un equipo. Para ello, se utilizan técnicas de seguimiento y verificación que permiten programar y planificar las tareas, solo cuando es necesario, es decir, justo antes de que se produzca una avería. En el mantenimiento predictivo, se estudian variables o parámetros específicos que están directamente relacionados con el estado de la instalación, como la temperatura o el aislamiento de los conductores. Al analizar estos datos, se puede obtener información sobre el estado de los componentes y su funcionamiento en la instalación, su base se encuentra en la monitorización de los dispositivos, ya que se debe evaluar su condición mientras están en funcionamiento. Gracias a diversas técnicas disponibles, es posible evaluar fallos en los componentes y realizar un seguimiento de su evolución durante períodos prolongados antes de decidir la intervención del técnico [13]. De esta forma, se puede coordinar el momento más adecuado para realizar el cuidado. Al implementar un plan de mantenimiento preventivo, es fundamental

seleccionar los equipos y elementos correctos. Esta selección se basa en tres factores principales:

- Relevancia del equipo en la instalación
- Facilidad de acceso
- Información técnica disponible sobre el equipo

Cuando se debe determinar si una falla debe corregirse de manera inmediata o programada, es esencial considerar la importancia del elemento defectuoso en el funcionamiento del sistema

Métodos y técnicas

El plan de mantenimiento preventivo es un conjunto detallado de tareas específicas asociadas a un equipo o máquina en particular, que describe las acciones, plazos y piezas de repuesto necesarias para llevar a cabo la tarea. Estas tareas suelen incluir limpieza, comprobación, ajuste, lubricación y reemplazo, para su efectividad requiere una investigación previa sobre los equipos y su historial de mantenimiento, lo que permite planificar y ejecutar el proceso de manera más eficiente. En general, el plan de mantenimiento se desarrolla en varias fases que incluyen la creación de un inventario técnico con manuales y planos, la definición de procedimientos técnicos, el control de las frecuencias y el registro de reparaciones, repuestos y costos para ayudar en la planificación futura. Las técnicas consisten en recopilar y analizar datos estadísticos sobre el comportamiento y los parámetros de control de los procesos y las máquinas en función de su funcionamiento y rendimiento. De esta manera, se puede crear un modelo de evaluación preventivo para anticipar posibles problemas. Este enfoque utiliza herramientas de inspección que se basan en mediciones técnicas para garantizar que el equipo esté en óptimas condiciones [14].

Laboratorio de mecatrónica

El laboratorio proporciona un entorno adecuado para la realización de prácticas, proyectos y tareas integradoras a los estudiantes para permitirles adquirir habilidades y conocimientos que les serán de utilidad en su futura carrera profesional. Específicamente, en el caso de las aplicaciones mecatrónicas, es fundamental que los estudiantes tengan la oportunidad de interactuar con equipos y tecnologías relevantes para la industria y que puedan desarrollar destrezas prácticas en el manejo y aplicación de dichos equipos. De esta manera, la práctica en entornos industriales y sus aplicaciones, no solo permite que los estudiantes adquieran habilidades

valiosas para su carrera, sino que también les proporciona la experiencia necesaria para entender mejor los desafíos y oportunidades del campo y su aplicación en la industria. Uno de sus laboratorios es el de manufactura, tiene como objetivo desarrollar actividades prácticas relacionadas con procesos de mecanizado, generación de código "G", y manejo de equipos CNC utilizando herramientas computacionales como CAD, CAM, CAE y PR para complementar los conocimientos teóricos. La utilización de estos software permite una simulación más precisa y un mejor entendimiento del proceso de mecanizado y manufactura. En la Figura 1.1 se muestra el laboratorio de mecatrónica, que incluye máquinas CNC y herramientas para procesos de manufactura.

Figura 1.1: Laboratorio de procesos de manufactura

1.2.2. Visión artificial

Para mejorar la interacción del usuario con su entorno, es necesario implementar sistemas avanzados, que utilicen nuevas técnicas más complejas, además de los sensores. Una de estas técnicas puede ser la visión artificial, que utiliza una cámara para capturar imágenes y procesar la escena capturada para obtener datos que puedan ser utilizados de la misma manera que los datos de los sensores tradicionales. La visión artificial se llama así porque imita el funcionamiento de la visión humana al obtener datos a partir de lo que se puede "ver.[en] la imagen capturada [15].

Visión por computadora

La visión por computadora es una disciplina que busca comprender las imágenes y extraer información sobre su contenido, incluyendo la forma, el brillo y la distribución de colores. Los sistemas desarrollados en este campo utilizan representaciones de la realidad para analizar y actuar en consecuencia. Aunque el reconocimiento de formas y colores es una tarea fácil para los humanos y los animales, su implementación en sistemas automatizados representa un gran desafío para los investigadores de esta área [16].

Técnicas de visión artificial

Machine learning

El machine learning es un conjunto de técnicas que permiten a las computadoras aprender a partir de los datos y aplicar ese conocimiento a situaciones que no han sido vistas previamente. Estas técnicas se basan en el uso de algoritmos de aprendizaje que son capaces de detectar patrones y asociaciones en grandes conjuntos de datos. En los últimos años, el machine learning ha experimentado un gran crecimiento gracias al aumento en la cantidad de datos disponibles y la capacidad de procesarlos de manera eficiente, además de la aparición de nuevos métodos de aprendizaje. Aunque la idea de que las computadoras aprendan de los datos no es nueva, la disponibilidad de grandes cantidades de datos y la capacidad de procesarlos de manera eficiente ha permitido la aplicación de estas técnicas en una amplia variedad de áreas, incluyendo la biología molecular, la imagenología y la práctica clínica. Hay dos enfoques principales en el machine learning: el aprendizaje supervisado y el no supervisado. En el aprendizaje supervisado, la computadora aprende a predecir el valor de una variable de salida a partir de un conjunto de variables de entrada y un conjunto de ejemplos de entrenamiento. En el aprendizaje no supervisado, la computadora busca patrones y estructuras en los datos sin ninguna información previa sobre las variables o categorías de interés. Los modelos de aprendizaje no supervisado tienden a ser más difíciles de interpretar desde un punto de vista fisiológico, mientras que los modelos de aprendizaje supervisado a menudo se basan en modelos fisiológicos previos y por lo tanto son más interpretables [17].

Deep Learning

Para ser considerado como un algoritmo de Deep Learning, se requiere que tenga alguna de las siguientes características: el uso de múltiples capas con unidades de procesamiento no lineales para extraer y transformar variables, el aprendizaje de múltiples niveles de características o

representaciones de datos, y la capacidad de aprender múltiples niveles de representación que correspondan a diferentes niveles de abstracción. Todas estas características involucran el uso de múltiples capas de procesamiento no lineal que forman una jerarquía de características, desde un nivel de abstracción más bajo a uno más alto. Aunque no existe un estándar para determinar cuántas capas son necesarias para considerar que un algoritmo es de Deep Learning, la mayoría de los investigadores en este campo coinciden en que debe tener más de dos transformaciones intermedias [18].

1.2.3. Elección de entorno

Open CV

OpenCV (Open source computer vision library) es una biblioteca enfocada en el procesamiento y análisis de imágenes. Fue creada por Intel en 1999 y ha ganado rápidamente reconocimiento y popularidad, convirtiéndose en una de las más utilizadas en el desarrollo de aplicaciones de visión artificial en la actualidad [19].

Python

Python es un lenguaje de programación de alto nivel y fácil de leer. Es ampliamente utilizado en áreas como desarrollo web, análisis de datos, inteligencia artificial y automatización de tareas. Su sintaxis clara y su amplia biblioteca estándar hacen que sea fácil de aprender y utilizar. Python cuenta con una gran comunidad de desarrolladores y ofrece una amplia gama de bibliotecas especializadas para diversas aplicaciones. Es una opción popular tanto para principiantes como para profesionales en programación.

1.2.4. Redes neuronales artificiales

La red neuronal artificial (RNA) es un modelo computacional que puede simular las funciones fundamentales del cerebro humano, como la capacidad de adaptarse, organizarse y tolerar errores. Durante las últimas dos décadas, ha habido un crecimiento exponencial en la aplicación de la RNA en casi todas las áreas de investigación, debido a su capacidad para resolver problemas que son difíciles de abordar con la estadística tradicional. Los modelos generados son apropiados para describir relaciones no lineales y predecir resultados a largo plazo, lo que puede ser difícil de lograr con modelos matemáticos convencionales. En resumen, la RNA se ha adoptado ampliamente para identificar, analizar, pronosticar, reconocer sistemas y optimizar el diseño de modelos [20].

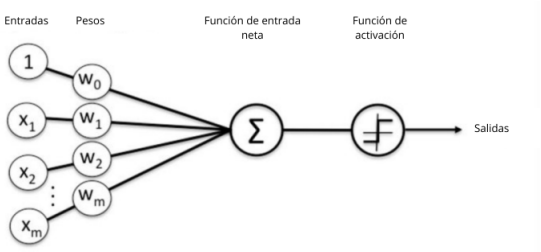


Figura 1.2: Ilustración de una red neuronal [21].

En la figura 1.2 indica un esquema que representa la apariencia de un nodo, donde se exhiben inicialmente un conjunto específico de entradas, a las cuales se les asigna un valor ponderado para agruparlas más adelante. Luego, el resultado se somete a una función de activación que, para simplificar, puede concebirse como un interruptor que, en función del valor de la señal de entrada, determina si se propaga hacia la salida o no.

Arquitectura

La estructura de una red neuronal se compone de un conjunto de neuronas interconectadas, organizadas en capas: entrada, intermedia y salida. La capa de entrada corresponde al número de variables de entrada, mientras que la capa de salida se relaciona con el número de variables a predecir. La capa intermedia se utiliza para aplicar transformaciones no lineales a las variables de entrada originales, como muestra la Figura 1.3 cada unidad neuronal está interconectada con muchas otras, y las conexiones entre ellas pueden aumentar o inhibir el nivel de activación de las neuronas vecinas.

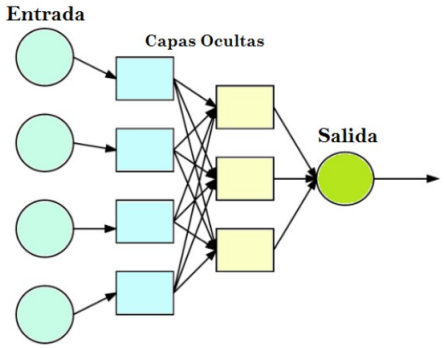


Figura 1.3: Arquitectura de una red neuronal artificial [22].

Redes neuronales convolucionales (CNN)

La CNN es un tipo de red neuronal artificial que simula el procesamiento del cortex visual humano para detectar distintas características en las entradas. Contiene varias capas ocultas especializadas, jerarquizadas de tal forma que las primeras capas detectan características básicas, mientras que las capas más profundas son capaces de reconocer formas complejas [23].

La Figura 1.4 indica el proceso de convolución para calcular mapas de características a partir de la imagen de entrada. Estos mapas se obtienen al aplicar múltiples kernels para detectar distintas propiedades y formas en la imagen, y luego reducir su tamaño mediante un proceso de agrupación. Este proceso se repite varias veces para extraer el mapa de características. Posteriormente, se utilizan capas completamente conectadas para generar la probabilidad de cada clase.

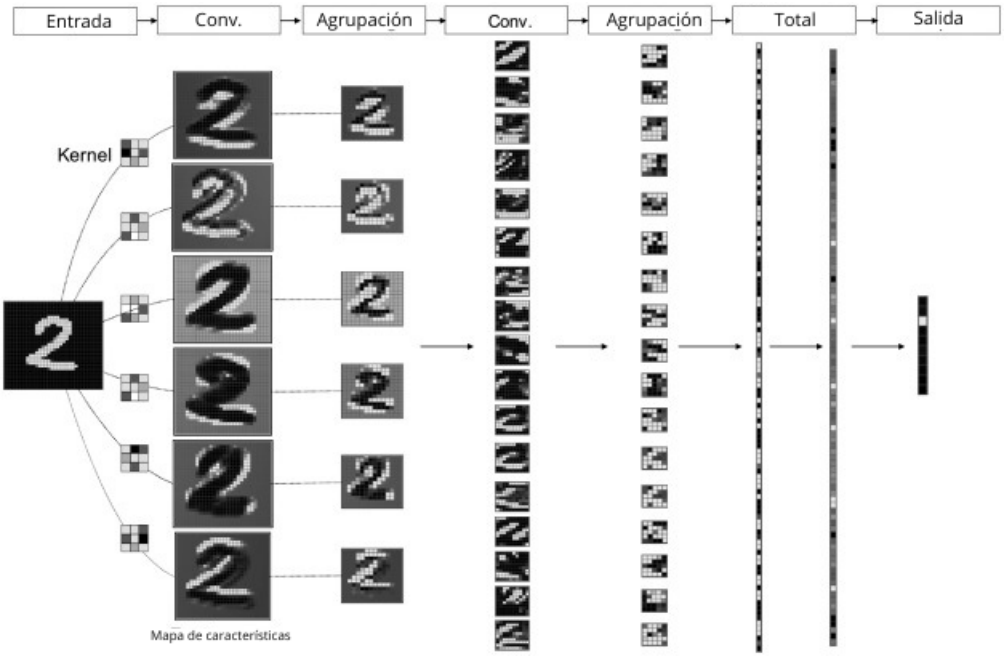


Figura 1.4: Estructura CNN [24].

Durante el proceso de entrenamiento, se ingresan los datos de entrenamiento a la red para obtener las predicciones, se calcula el error y se actualizan los parámetros desde la capa de salida hacia la capa de entrada mediante la retropropagación. El objetivo es repetir este proceso para obtener buenos parámetros que permitan reconocer las imágenes de manera efectiva.

Modelo Faster R-CNN

La arquitectura Faster R-CNN se compone de cuatro etapas fundamentales: extracción de características, Red de Regiones Propuestas (RPN), agrupación y clasificación de regiones de interés (ROI). La RPN genera un conjunto de propuestas de objetos en forma de rectángulos, cada una con una puntuación que indica su nivel de objetividad. Mediante el uso de anclas o cuadros de diferentes tamaños y escalas, junto con la aplicación de regresión softmax y regresión delimitadora, se logra ajustar y refinar las propuestas seleccionadas para obtener propuestas más precisas. En este sentido, la RPN se conecta con la estructura convolucional de la Fast R-CNN, permitiendo generar los contornos de las regiones y encontrar los objetos deseados [25].

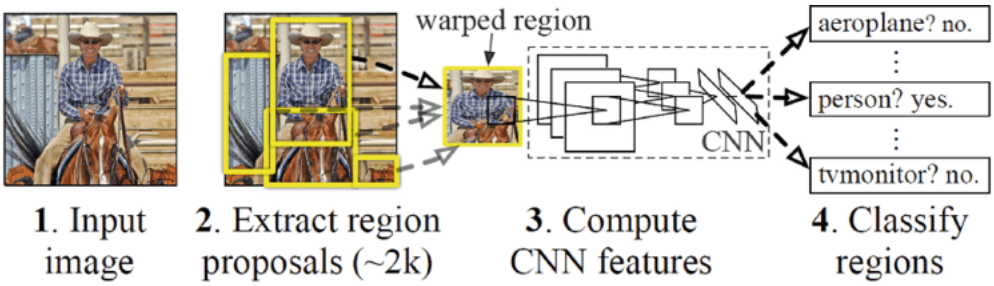


Figura 1.5: Arquitectura MobileNet-SSD [26].

En la Figura 1.5 se presenta un esquema que ilustra el funcionamiento de R-CNN. Esta técnica implica utilizar una red neuronal profunda previamente entrenada para la clasificación de imágenes, la cual se adapta para la detección de objetos.

El enfoque fundamental descrito en el primer artículo sobre R-CNN se describe de la siguiente manera: (1) se toma una imagen de entrada y (2) se generan numerosas propuestas de regiones mediante un proceso específico. Estas propuestas de regiones, conocidas como Regiones de Interés (ROIs), se envían de forma independiente a través de la red, generando un vector de valores de coma flotante, como por ejemplo 4.096 dimensiones, para cada ROI. Finalmente, (4) se entrena un clasificador que utiliza la representación en coma flotante de los 4.096 valores como entrada, y genera una etiqueta y una confianza para cada ROI detectada.

Redes neuronales recurrentes (RNN)

Las redes neuronales recurrentes son un tipo de red dinámica en la que el cálculo de una entrada en un paso determinado depende del paso anterior, lo que las hace ideales para el tratamiento de secuencias, análisis de trayectorias, predicciones no lineales y modelación de siste-

mas dinámicos. Las redes recurrentes se pueden clasificar en parcialmente recurrentes y totalmente recurrentes, y las conexiones recurrentes pueden ser fijas o variables. Las redes parcialmente recurrentes incluyen dos modelos principales: la red simple recurrente de Elman y la red secuencial de Jordan, mientras que las redes totalmente recurrentes se destacan por los modelos Real-Time Recurrent Learning (RTRL), Time-Depent Recurrent Back Propagation (TDRB) y Long short-term memory (LSTM). Este último es una arquitectura particular de una red totalmente recurrente, propuesta para resolver las dificultades presentadas en los modelos RTRL y TDRB [27].

Entrenamiento

En el aprendizaje de una red neuronal, se emplean técnicas como la minimización del error o la optimización de una función de recompensa para ajustar los pesos sinápticos en función de las entradas disponibles y mejorar la respuesta de la red a las salidas deseadas. Es importante establecer una función de error que evalúe el rendimiento de la red en un momento determinado y buscar los pesos que minimicen esta función durante el proceso de entrenamiento. Este consta de varias etapas, incluyendo la inicialización de la red con pesos aleatorios, el análisis de nuevos patrones de entrada y la refinación de los pesos para disminuir el error en cada iteración, la asignación del error y la propagación hacia atrás para modificar los pesos a través del algoritmo Backpropagation. Finalmente, se ajustan los pesos utilizando el método del gradiente descendiente para minimizar el error [28].

1.2.5. Detección de objetos

En el ámbito de visión por computadora, la detección de objetos se realiza mediante la identificación de un objeto específico dentro de una imagen, basándose en su apariencia visual y su posición. Esto se logra al asociar características propias del objeto, que pueden ser definidas previamente o extraídas mediante modelos matemáticos, estas pueden incluir histogramas de color, intensidad de luz, y técnicas más avanzadas, que utilizan métodos de aprendizaje automático para identificar características. Para localizar los objetos en las imágenes se utilizan (cajas delimitadoras) bouding boxes, que definen el espacio que ocupa el objeto en la imagen y su tamaño en pixeles [29].

En la Figura 1.6 se puede visualizar objetos detectados, resaltados con cuadros alrededor de ellos, en un entorno de prueba.

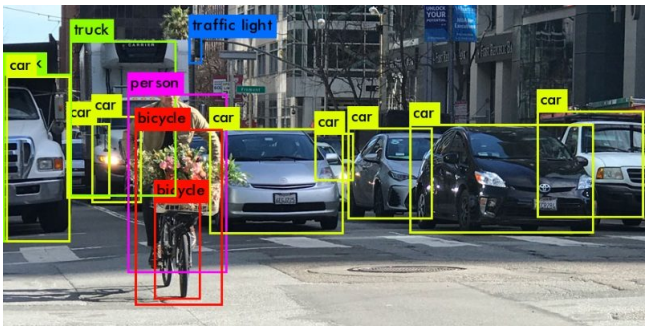


Figura 1.6: Modelo de detección de objetos [30].

Algoritmos de detección de objetos

Un algoritmo de detección de objetos es una técnica empleada en visión por computadora y aprendizaje automático para identificar y ubicar objetos específicos en imágenes o videos. Este proceso implica varias etapas, como el procesamiento previo de la imagen, la extracción de características distintivas, la elección de un modelo de detección, la detección y localización de los objetos, y el uso de técnicas de postprocesamiento. Estos algoritmos son utilizados en una amplia gama de aplicaciones, como sistemas de seguridad, vehículos autónomos y reconocimiento facial. Continúan evolucionando constantemente gracias a los avances en el aprendizaje automático y la visión por computadora. Existen varios algoritmos de detección de objetos algunos de ellos son:

Pytorch

Es una biblioteca de código abierto enfocada en el aprendizaje automático, que se basa en redes neuronales convolucionales y está escrita en el lenguaje de programación Python. Está diseñada para una integración profunda y ofrece tres características principales: cálculo de tensores, aceleración mediante GPUs y redes neuronales convolucionales integradas con un sistema de integración automática. PyTorch brinda una amplia gama de funciones para operaciones con tensores, satisfaciendo las necesidades de cómputo y permitiendo un desarrollo rápido y sencillo utilizando el lenguaje Python. También incorpora Redes Neuronales Dinámicas, que utilizan una implementación eficiente de una técnica llamada derivación automática en modo inverso, lo que permite a los usuarios modificar arbitrariamente el comportamiento de la red [31].

YOLO

El enfoque YOLO aborda la detección de objetos como un problema de regresión único. Mediante una red convolucional, se realizan predicciones simultáneas de múltiples cuadros delimitadores que encuadran los objetos en la imagen, así como probabilidades condicionales p(Clase | Objeto) para cada una de estas regiones. Esto permite identificar y clasificar los objetos de manera eficiente. El modelo de red neuronal logra una alta velocidad de ejecución, alcanzando hasta 45 fotogramas por segundo (fps) en computadoras de propósito general [32].

Se aplica un enfoque mediante el cual se utiliza una red neuronal única para analizar la imagen en su totalidad, dividiéndola en regiones específicas. Esto permite predecir cuadros delimitadores y la probabilidad correspondiente para cada región. Una de las ventajas clave de este modelo radica en el análisis completo de la imagen en el momento de la prueba, lo que proporciona un contexto global. El modelo hace uso de técnicas de Deep Learning y Redes Neuronales Convolucionales (CNN) en la detección de objetos. Su característica distintiva reside en "ver"la imagen únicamente una vez, lo que acelera considerablemente el proceso y contribuye al resultado final. Durante la división de la imagen en una cuadrícula SxS, en cada celda se generan predicciones de N posibles cuadros delimitadores, junto con el cálculo del nivel de probabilidad. El cálculo SxSxN se realiza de manera eficiente, con un bajo nivel de incertidumbre. Posteriormente, se eliminan los cuadros delimitadores que no cumplen con el límite establecido. A continuación, se aplica una técnica conocida como "supresión no máxima", que permite eliminar los elementos posiblemente duplicados. Esto garantiza que solo se mantenga el cuadro delimitador más preciso y exacto [33].

TensorFlow

Tensorflow, desarrollado por Google, es una librería de código abierto que tiene como objetivo simplificar el proceso de diseño, construcción y entrenamiento de sistemas de aprendizaje profundo. Una de las características distintivas de Tensorflow es su enfoque de programación basado en un grafo, en el cual las operaciones y los datos se almacenan internamente. Esto proporciona una forma eficiente de visualizar las dependencias entre las operaciones y asignarlas a distintos dispositivos, incluyendo procesadores gráficos [34].

Esta biblioteca simplifica las operaciones con matrices y vectores, colabora con las redes neuronales, que son capaces de reconocer patrones y son fundamentales para el funcionamiento del proyecto de inteligencia artificial. Las operaciones realizadas en los conjuntos de datos,

también conocidos como librerías, se asemejan a la estadística inferencial. Con una base de datos amplia, en este caso la información de cada raza, se busca identificar el patrón presente en ellos. Inicialmente, lo más importante es contar con los datos y una amplia y diversa librería de referencias que oriente a la red neuronal hacia una interpretación. Tensorflow utiliza grafos para crear un modelo en el cual trabaja, donde cada nodo representa una operación aritmética que genera tensores que son objetos geométricos que describen relaciones entre vectores geométricos, escalares y otros. Estos son los objetos que la red neuronal utiliza para producir valores [35].

MobileNet-SDD V2

Se ha notado una tendencia común en los modelos de visión por computadora, donde se vuelven progresivamente más profundos y complicados para alcanzar una mayor precisión. Sin embargo, estos avances vienen acompañados de un incremento en el tamaño y la demora, lo que restringe su aplicabilidad en sistemas con recursos computacionales limitados.

En este tipo de casos, MobileNet resulta muy útil, es un modelo diseñado específicamente para aplicaciones móviles e integradas que requieren alta velocidad. Su primera versión, MobileNetV1, utiliza convoluciones separables en profundidad para reducir el tamaño del modelo y el costo de complejidad de la red, haciéndolo adecuado para aplicaciones con baja capacidad de procesamiento. Posteriormente, en la segunda edición de la familia, se introdujo una estructura residual invertida para mejorar la modularidad del modelo, dando lugar a MobileNetV2. Esta mejora ha ayudado a eliminar las no linealidades en capas estrechas, lo que se traduce en un rendimiento de vanguardia para las aplicaciones mencionadas. Alrededor del mismo período en que se presentó la primera versión de MobileNet, Google lanzó el detector de un solo disparo (SSD) para aplicaciones que requieren tanto velocidad como precisión. SSD es capaz de detectar múltiples objetos en una imagen utilizando un solo disparo. MobileNet, a pesar de ofrecer una velocidad decente, presenta un inconveniente en cuanto a su precisión. Por lo tanto, se integró con SSD para mejorar su precisión sin sacrificar la velocidad, creando así el modelo MobileNet-SSD [36].

En la figura 1.7 se puede observar el proceso de clasificación, las capas finales de la red neuronal presentan la siguiente configuración: Normalmente, la salida de la red base tiene dimensiones de 7x7 píxeles. Para iniciar el proceso de clasificación, se utiliza una capa de agrupación global que reduce el tamaño de la imagen de 7x7 a 1x1 píxeles, lo que esencialmente implica trabajar con un conjunto de 49 predictores diferentes.

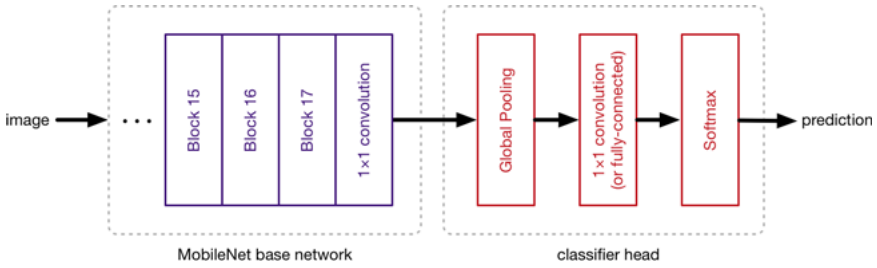


Figura 1.7: Arquitectura MobileNet-SSD [37].

Métodos de detección de objetos

En el campo de la visión por computadora, se utilizan diversas técnicas para analizar y procesar información visual. Estas técnicas se basan en conjuntos de imágenes anotadas, conocidos como datasets. Las principales técnicas incluyen la clasificación de imágenes, la localización de objetos y la detección de objetos.

La clasificación consiste en categorizar las imágenes en diferentes clases o categorías. Se utiliza un conjunto de datos etiquetados para entrenar modelos que puedan reconocer y asignar nuevas imágenes a las clases correspondientes. La localización de objetos se enfoca en detectar la posición y delimitar los elementos de interés en una imagen. Se utilizan algoritmos y técnicas para identificar y ubicar los objetos específicos mediante el uso de cuadros delimitadores. La detección de objetos combina la clasificación y la localización para identificar y etiquetar múltiples elementos de diferentes categorías en una imagen. Se busca tanto clasificar los objetos presentes como determinar su ubicación precisa en la imagen [38].

La segmentación semántica es un proceso que asigna diferentes clases a cada píxel de una imagen, lo que permite comprender su función en la escena. A diferencia de la detección de objetos, se busca delimitar los objetos a nivel de píxel, lo que proporciona una mayor precisión en la localización. La segmentación de instancia va más allá al permitir distinguir entre objetos superpuestos pertenecientes a la misma o diferentes clases, lo que mejora aún más la precisión en la detección [39].

1.2.6. Eyetracking

El seguimiento de pupila (EyeTracking) es una técnica experimental que permite registrar el desplazamiento de los ojos y la posición de la mirada, es una herramienta comúnmente utilizada para observar cómo se distribuye la atención visual [40].

Censado invasivo

El método invasivo de censado, se realiza típicamente mediante el uso de lentes de contacto o bobinas magnéticas, proporciona resultados precisos y exactos acerca de los puntos de atención visual que son utilizados durante una tarea dirigida [41].

Censado no invasivo

Se puede realizar un seguimiento ocular sin la necesidad de entrar en contacto directo con el ojo. Mediante el uso de una luz, como la infrarroja, se puede detectar el movimiento ocular a través del reflejo del haz de luz, se captura mediante una cámara de vídeo o un sensor óptico. Los dispositivos de seguimiento ocular no invasivos suelen utilizar el reflejo corneal y el centro de la pupila para determinar el movimiento del ojo. Algunos también analizan la parte frontal de la córnea y la parte posterior del cristalino. También existen dispositivos que registran el interior del ojo, centrándose en la posición de los vasos sanguíneos de la retina. En general, los métodos ópticos son ampliamente considerados en la investigación, ya que son de bajo costo y no invasivos. Sin embargo, pueden presentar fallas al momento de registrar el movimiento ocular, ya que a veces no detectan con precisión la pupila, la córnea u otros indicadores oculares utilizados para el seguimiento ocular. Además, si el sujeto cierra los ojos, no se podrá registrar su comportamiento visual [42].

Aplicaciones en la industria

En la industria, es posible combinar el seguimiento de movimientos oculares con la robótica para permitir que los operarios realicen acciones utilizando solo sus ojos. Por ejemplo, pueden dirigir una cámara hacia un lugar específico o tomar capturas de pantalla para ayudar en tareas de planificación, evaluación, entrenamiento o medición. Esta integración de tecnologías ofrece una mayor eficiencia y precisión en la ejecución de diversas actividades industriales [43].

Aplicado al marketing incluyen el desarrollo de productos, el comportamiento del consumidor, la optimización del diseño y la estrategia de mercado, entre otras. Lo que comparten estas áreas de estudio es su dependencia de la toma de decisiones, es decir, cuando se toma una deci-

sión, las personas implicadas en cada caso generan movimientos oculares. Estos movimientos se analizan para garantizar que el estudio o proyecto en cuestión se desarrolle con la menor cantidad de errores posible [44].

1.2.7. Base de datos para el mantenimiento

Una base de datos para el mantenimiento es un sistema que almacena y organiza información relacionada con las actividades de mantenimiento. Incluye registros de activos, historial de mantenimiento, programación de tareas preventivas, órdenes de trabajo, inventario de repuestos y generación de informes. Permite una gestión eficiente del mantenimiento, facilitando la planificación, seguimiento y análisis de las actividades. Contribuye a mejorar la eficiencia y prolongar la vida útil de los activos.

Definición y objetivos de la base de datos

Una base de datos es un conjunto de datos organizados de forma estructurada en diferentes campos. Estos, se almacenan en un repositorio con el propósito de ser utilizados y procesados para convertirse en información útil. Consta de tablas, que almacenan los datos en filas y columnas, y es importante que los datos almacenados en cada tabla estén relacionados con el tema de la base de datos en cuestión. Las consultas permiten filtrar la información y mostrar los datos específicos que se desean consultar. Los formularios proporcionan formatos para interactuar con la base de datos, permitiendo al usuario crear, modificar o consultar la información de manera más fácil. Los informes son presentaciones personalizadas de los registros almacenados, generados a partir de las consultas realizadas [45].

Desarrollo de una bases de datos

En la Figura 1.8 se observan los parámetros a seguir para la creación de una base de datos, esto asegura la implementación de una estructura bien organizada, un rendimiento óptimo, la preservación de la integridad de los datos y la implementación de medidas de seguridad adecuadas, lo que a su vez permite una adaptabilidad fluida. Estas prácticas contribuyen significativamente a la eficacia y el éxito del proyecto a largo plazo.

Interfaz de usuario para la gestión de datos

La interfaz de usuario de un sistema gestor de base de datos se define como la manera en que el usuario interactúa y utiliza la base de datos. Existen sistemas para la gestión de datos que

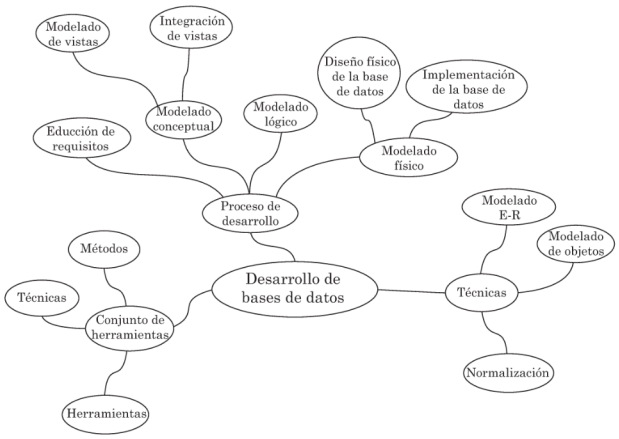


Figura 1.8: Desarrollo de base de datos [46].

ofrecen la posibilidad de personalizar la interfaz según las necesidades del usuario, permitiendo que vea y trabaje únicamente con los elementos que el administrador de la base de datos haya definido [47]

1.2.8. MQTT

MQTT se presenta como un protocolo de mensajería que se basa en estándares, funcionando como un conjunto de reglas que facilita la comunicación entre computadoras. En el ámbito de los dispositivos inteligentes, wearables y otros elementos pertenecientes al Internet de las cosas (IoT), es común la necesidad de transmitir y recibir datos a través de redes con recursos y ancho de banda limitados. En este contexto, los dispositivos IoT optan por emplear MQTT para la transmisión eficiente de datos, ya que su implementación resulta sencilla y permite una comunicación eficaz. MQTT proporciona soporte para la mensajería tanto de dispositivo a nube como de nube a dispositivo [48].

Capítulo II

Marco metodológico

2.1. Modelo de la investigación

En este capítulo se abordará un tema de gran relevancia en el ámbito de visión artificial y el mantenimiento preventivo de equipos: la detección de objetos mediante la integración de técnicas de seguimiento ocular, esta investigación se lleva a cabo con el propósito de desarrollar un algoritmo innovador que permita identificar y monitorear de manera precisa y eficaz diferentes equipos, con el fin de mejorar su mantenimiento y reducir los tiempos de inactividad.

La metodología empleada en esta investigación se caracteriza por su enfoque aplicado y multidisciplinario, ya que combina conocimientos técnicos en visión artificial y técnicas de recolección de datos, como el seguimiento ocular, con el objetivo de abordar una problemática concreta en el campo de la tecnología de fabricación aditiva. A través de la fusión de elementos, se pretende ofrecer una solución efectiva y precisa que permita detectar cualquier anomalía en los equipos de impresión 3D de manera temprana. La investigación se divide en tres fases bien definidas. En la primera fase, se realiza una revisión exhaustiva de la literatura para clasificar y analizar los modelos de detección de objetos basados en inteligencia artificial existentes. Esta etapa resulta fundamental para comprender las técnicas y enfoques previamente desarrollados en la detección de objetos y establecer una sólida base para el proyecto. La segunda fase implica la creación de una base de datos específicamente diseñada para almacenar información relevante sobre el funcionamiento de los equipos de impresión 3D. Además, se integra la tecnología de seguimiento ocular con los modelos de detección de objetos en un entorno práctico, con el propósito de adquirir datos precisos y evaluar su utilidad en la detección de elementos críticos. Finalmente, la tercera fase se concentra en validar el algoritmo desarrollado, empleando métricas como la matriz de confusión para evaluar su rendimiento. Se seleccionan conjuntos

de prueba representativos y se analizan los resultados obtenidos. En caso de que el algoritmo demuestre un desempeño satisfactorio, se procede a su implementación en el sistema final, lo cual impactará de manera significativa en el ámbito del mantenimiento predictivo de equipos de impresión 3D.

Esta metodología, que combina revisión bibliográfica, recopilación de datos en campo y evaluación rigurosa, se orienta hacia la obtención de resultados concretos y aplicables en un entorno tecnológico en constante evolución. La integración de la visión artificial y el seguimiento ocular promete ofrecer nuevas perspectivas y soluciones innovadoras en el mantenimiento de equipos de impresión 3D, contribuyendo al avance de la industria y la mejora de la eficiencia operativa en este campo.

La investigación será aplicada ya que implica la aplicación de conocimientos teóricos en un campo específico, como la visión artificial, para desarrollar un algoritmo que pueda ser utilizado en el mantenimiento de equipos de impresión 3D. Además, la utilización de técnicas de recolección de datos, como el eye tracking, y la elaboración de una base de datos para obtener información relevante para el desarrollo del algoritmo.

La investigación será bibliográfica porque clasificará los modelos de detección de objetos basados en inteligencia artificial, identificará los diferentes enfoques y modelos de detección de objetos que se han desarrollado, sus ventajas, desventajas, para conocer las últimas tendencias y avances en el campo de la visión artificial, y para obtener información sobre las técnicas y herramientas de análisis de datos que se pueden utilizar para evaluar el desempeño del algoritmo.

Esta investigación será de campo debido a que se obtendrá información relevante sobre las condiciones reales de uso del equipo de impresión 3D y los elementos que deben ser monitoreados para detectar cualquier problema en su funcionamiento. La utilización del eye tracking para identificar un solo elemento y no varios a la vez también requiere de datos de campo para evaluar su eficacia en un entorno real.

Los resultados obtenidos permitirán determinar la efectividad del algoritmo de eye tracking desarrollado en la detección de elementos críticos en equipos de impresión 3D, lo que tendrá implicaciones importantes en el campo del mantenimiento predictivo de equipos.

2.2. Diseño de la investigación

2.2.1. Clasificación de modelos de detección de objetos de inteligencia artificial

Actividad 1: revisión bibliográfica exhaustiva sobre los modelos de detección de objetos basados en inteligencia artificial"

Investigación exhaustiva sobre los diversos modelos de detección de objetos basados en inteligencia artificial que se han propuesto en la literatura. El objetivo es obtener un conocimiento sólido sobre los enfoques existentes y comprender las técnicas y metodologías utilizadas en este campo.

Actividad 2: identificación de modelos a partir de la revisión bibliográfica

Después de realizar la revisión bibliográfica, esta actividad consiste en identificar los modelos de detección de objetos específicos que se ajustan a los requisitos y objetivos del proyecto. Es importante seleccionar una variedad de modelos que sean relevantes y representativos de las técnicas utilizadas en el campo de la detección de objetos.

Actividad 3: análisis de características de los modelos, fortalezas y debilidades

En esta actividad, se analizan en detalle las características de los modelos identificados en la actividad anterior. Esto implica examinar aspectos como la precisión de detección, la velocidad de procesamiento, la eficiencia en el uso de recursos computacionales y otras métricas relevantes. También se deben identificar las fortalezas y debilidades de cada modelo en términos de su aplicabilidad y rendimiento en diferentes escenarios.

Actividad 4: clasificación de modelos de detección de objetos basados en inteligencia artificial según las características analizadas

Se clasifican los modelos de detección de objetos basados en inteligencia artificial en diferentes categorías o grupos en función de las características y el rendimiento analizados en la actividad anterior. Esto ayuda a organizar y comprender mejor la diversidad de enfoques existentes y proporciona una base para tomar decisiones informadas sobre qué modelos pueden ser más adecuados para la investigación.

2.2.2. Desarrollo de una base de datos e incorporación del eye tracking con la detección de objetos con IoT y MQTT

Actividad 1: elaboración de la base de datos que permita almacenar información recolectada del equipo de impresión 3D

Diseño de una base de datos versátil destinada a almacenar información no solo de la impresora 3D, sino también de un torno y una fresadora.

Actividad 2: elección de la tecnología de base de datos más adecuada en función de las características de la información a ser almacenada

Esto incluye consideraciones relacionadas con la escalabilidad, la compatibilidad con servicios en la nube, la seguridad de los datos y la eficacia en las consultas.

Actividad 3: implementación de la base de datos en un entorno en la nube

Junto con la definición de permisos de acceso para que garanticen la seguridad y privacidad de la información almacenada. Para asegurar la integridad de los datos, se establecerán niveles de acceso y roles de usuario apropiados, así como medidas de seguridad tal como el cifrado de datos.

Actividad 4: incorporación de protocolos IoT y MQTT para habilitar la transmisión en tiempo real de los valores generados por los sensores en las máquinas. Estos protocolos permiten la comunicación eficiente y segura de datos desde los dispositivos a una websocket. Esto, a su vez, posibilita la observación en tiempo real de los valores proporcionados por los sensores, lo que es esencial para el monitoreo y control de los equipos de impresión 3D, torno y fresadora.

Actividad 5: implementación del seguimiento ocular en tiempo real mediante un predictor que captura y procesa puntos faciales

La tecnología de seguimiento ocular se integra con el sistema existente y permite observar en tiempo real la dirección de la mirada del usuario. Este componente adicional resulta crucial para mejorar la interacción de las máquinas, ya que proporciona información valiosa sobre el enfoque visual del operador. Además, mediante el seguimiento ocular, se logra identificar de manera precisa un solo equipo a la vez, permitiendo así obtener información específica sobre el equipo que el usuario está observando en ese momento. Esta característica es esencial para proporcionar datos detallados y relevantes en función de la máquina que se observa.

Actividad 6: validación del funcionamiento de la base de datos

En este proceso, se verifica que los datos se almacenan de manera adecuada, se realizan consultas y análisis para extraer información relevante, y se evalúa la eficacia y utilidad de la base de datos en el contexto del proyecto.

2.2.3. Validación del algoritmo

Actividad 1: evaluación para medir el desempeño del algoritmo con matriz de confusión Se lleva a cabo una evaluación para medir el desempeño del algoritmo de detección de objetos mediante el uso de una matriz de confusión. Dicha matriz proporciona información relativa a la precisión, sensibilidad, especificidad y otras métricas que permiten evaluar la eficacia del algoritmo en la detección de objetos.

Actividad 2: selección de conjuntos de prueba Se enfoca en la selección de conjuntos de prueba adecuados que representen diversos escenarios y condiciones en los cuales se empleará el algoritmo de detección de objetos. Esto asegura una evaluación exhaustiva y la capacidad del algoritmo para generalizar su rendimiento en diferentes situaciones.

Actividad 3: análisis de resultados, si los resultados son satisfactorios, implementación del algoritmo en el sistema

Comprende el análisis de los resultados de la evaluación del algoritmo y la determinación de si cumple con los criterios de desempeño establecidos. Si los resultados son satisfactorios y el algoritmo demuestra ser eficaz en la detección de objetos, se procede a su implementación en el sistema final para su aplicación práctica.

2.2.4. Especificaciones del sistema a diseñar

En esta sección, se detallan las especificaciones del sistema de mantenimiento preventivo diseñado para equipos de laboratorio, que se basa en la detección de objetos y el seguimiento visual. El propósito de este sistema es proporcionar una solución efectiva y en tiempo real para la monitorización y diagnóstico de los equipos. El sistema consta de una arquitectura de red neuronal convolucional RCNN Faster Inception V2 para la detección de objetos, tecnología de seguimiento visual, una base de datos y una interfaz de usuario desarrollada con la biblioteca Qt5. Además, a través de protocolos IoT y MQTT, se logra que los valores de los sensores

se muestren en tiempo real en un websocket. Esto permite el monitoreo en tiempo real de los equipos, brindando a los usuarios una visión en tiempo real de los datos de los sensores, lo que facilita el diagnóstico y la toma de decisiones informadas para el mantenimiento preventivo de los equipos de laboratorio.

A continuación un listado de criterios del sistema:

- **Precisión de detección de objetos.** El sistema debe ser capaz de detectar con alta precisión los equipos en tiempo real, con un bajo índice de falsos positivos y falsos negativos.
- **Eficiencia en tiempo real.** El sistema debe funcionar en tiempo real, proporcionando detección de objetos y seguimiento ocular sin demoras significativas.
- **Seguimiento ocular preciso.** Debe rastrear con precisión la dirección de la mirada del usuario y determinar qué equipo está siendo observado en un momento dado.
- **Seguimiento visual personalizado.** La integración del seguimiento visual personalizado permitirá a los usuarios interactuar con los equipos y recibir información específica al mirarlos.
- **Comunicación IoT robusta.** La comunicación entre los sensores y el sistema central mediante MQTT debe ser confiable y tolerante a fallos para garantizar la recopilación de datos en tiempo real.
- **Interfaz de usuario intuitiva.** La información de detección y seguimiento visual debe presentarse de manera clara y comprensible, utilizando indicadores visuales.
- **Personalización de umbrales.** Los usuarios deben tener la opción de ajustar los umbrales de detección y confianza para adaptar el sistema a las necesidades específicas de cada equipo.
- **Mantenimiento preventivo efectivo.** Debe permitir la detección temprana de problemas o fallos en los equipos y proporcionar recomendaciones para el mantenimiento preventivo.
- **Seguridad de datos.** Se debe garantizar que los datos de seguimiento visual y detección se manejen de manera segura y se cumplan los requisitos de privacidad.

- **Compatibilidad y adaptabilidad.** El sistema debe ser compatible con diferentes cámaras y hardware de equipos y estar preparado para adaptarse a futuras tecnologías.
- **Facilidad de implementación.** El sistema debe ser relativamente sencillo de implementar y mantener, con una documentación clara y recursos disponibles.
- **Documentación y soporte técnico.** Debe existir documentación detallada y recursos de soporte técnico para facilitar la implementación y el mantenimiento del sistema.
- **Validación mediante pruebas.** Debe ser validado a través de pruebas exhaustivas que incluyan escenarios de uso real y se deben definir métricas de éxito.

2.2.5. Limitaciones asociadas a la solución

- La precisión del modelo de detección de objetos puede verse afectada por condiciones de iluminación inadecuadas, oclusión de componentes y variaciones en la apariencia de los equipos.
- La calibración del seguimiento visual puede ser un proceso inicial que requiere tiempo y puede afectar la precisión si no se realiza correctamente para cada usuario.
- La calidad y características de las cámaras y hardware de los equipos pueden afectar la calidad de las imágenes y la capacidad de detección.
- La detección y confianza de manera óptima puede requerir pruebas y ajustes iterativos para lograr resultados óptimos.
- La precisión del seguimiento visual puede disminuir en casos de miradas rápidas o movimientos oculares irregulares.
- La implementación del seguimiento visual plantea preocupaciones de privacidad que deben ser gestionadas de manera adecuada y cumplir con regulaciones de privacidad.
- La solución podría requerir actualizaciones periódicas para mantenerse al día con cambios en los equipos, modelos de detección y tecnologías de seguimiento visual.
- Dependiendo de la implementación y configuración del sistema podría estar limitado por la disponibilidad de energía eléctrica y conectividad de red.
- La implementación de tecnologías como seguimiento visual y cámaras de alta calidad podría implicar costos adicionales en términos de hardware y software.

- La ejecución simultánea de detección de objetos y seguimiento visual podría requerir una capacidad de procesamiento significativa, lo que podría limitar la cantidad de equipos monitoreados de manera concurrente.
- Los usuarios necesitarán tiempo para familiarizarse con la interfaz y los controles del sistema, así como para comprender y ajustar los umbrales de detección.
- Cambios en las condiciones ambientales del laboratorio, como la iluminación, podrían afectar la calidad de las imágenes y la precisión de la detección.

2.3. Diseño

A continuación, se presenta una descripción detallada de la solución propuesta para el mantenimiento preventivo, mediante la detección de objetos y el seguimiento visual. La solución está diseñada para proporcionar un sistema integral y eficiente que permita a los usuarios monitorear y diagnosticar los equipos en tiempo real, anticipando posibles problemas y asegurando un funcionamiento óptimo.

2.3.1. Solución propuesta

La solución propuesta en este trabajo de investigación es un sistema integral de mantenimiento en tiempo real para equipos de impresión 3D, que combina la visión artificial, el seguimiento ocular y la comunicación IoT para brindar un enfoque innovador en la gestión y el mantenimiento de estos equipos. La solución se basa en una serie de componentes y tecnologías clave:

- **Detección de Objetos:** la implementación de una red convolucional RCNN Faster Inception v2 permite la identificación precisa de equipos de impresión 3D en imágenes en tiempo real. Esta detección es fundamental para el seguimiento específico de cada equipo.
- **Seguimiento ocular:** A través del uso de un predictor facial,se realiza un seguimiento ocular preciso, permitiendo al sistema determinar qué equipo está siendo observado en un momento dado. Esto mejora la interacción y la capacidad de proporcionar información específica.
- **Comunicación IoT:** la solución incorpora una infraestructura de IoT robusta, que involucra sensores de temperatura, humedad y vibración, así como hardware IoT (ESP32 y

Arduino con DAQ). La comunicación se realiza a través del protocolo MQTT, lo que permite la adquisición de datos de sensores en tiempo real.

- **Interfaz de usuario:** la interfaz de usuario desarrollada con PyQt5 integra datos de visión por computadora y datos de sensores en una sola vista en tiempo real. Los usuarios pueden acceder a información detallada sobre el estado de los equipos y sus condiciones ambientales.

La solución propuesta ha sido diseñada para proporcionar una visión integral de los equipos de impresión 3D y su entorno, permitiendo la detección temprana de problemas y la implementación de un mantenimiento preventivo eficaz, esta solución representa un paso significativo hacia la aplicación de tecnologías avanzadas en la mejora de la eficiencia y la confiabilidad de equipos industriales, y ofrece una visión prometedora de cómo la convergencia de la visión por computadora, el seguimiento ocular y la comunicación IoT puede transformar la industria de la impresión 3D y otras aplicaciones industriales.
El diseño del sistema se divide en dos componentes esenciales: la construcción de hardware y el desarrollo de software. Cada uno de estos aspectos es fundamental para el funcionamiento integral del sistema de mantenimiento en tiempo real para equipos de impresión 3D. A continuación, se describe el procedimiento para la construcción de hardware y el desarrollo de software:
Construcción de Hardware

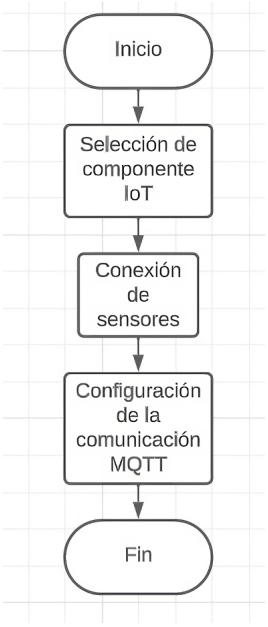


Figura 2.1: Diagrama por parte de hardware

El desarrollo del hardware en el proyecto se centró en la construcción de la infraestructura necesaria para la comunicación IoT y la adquisición de datos de sensores. El procedimiento incluyó la selección de componentes clave, como la ESP32 y el Arduino con DAQ, así como la elección de sensores específicos de temperatura y vibración. Se conectaron y configuraron los sensores para garantizar una adquisición precisa de datos. Luego, se configuró el protocolo MQTT en la ESP32 para permitir la transmisión de datos en tiempo real al servidor MQTT como se muestra en la Figura 2.1.

Los dispositivos electrónicos empleados en la ejecución de este este trabajo de investigación son componentes que presentan un alto nivel de rendimiento, son fáciles de integrar y están ampliamente disponibles en el entorno. Por tanto, se considera esencial el uso de:

ESP32

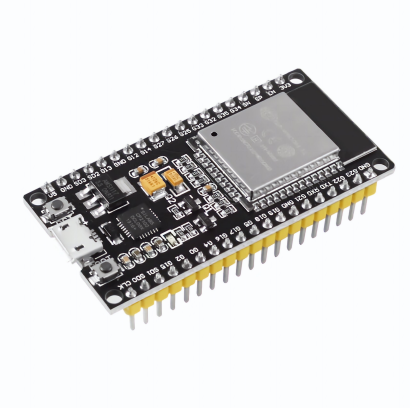


Figura 2.2: Módulo ESP32 [49].

El módulo ESP32, es un dispositivo basado en un microcontrolador de 32 bits que incorpora un periférico de comunicación inalámbrica operando en la banda de 2.4 GHz. Esto permite la conectividad a redes Wi-Fi y Bluetooth. Los ESP32 DevKits son placas de evaluación compactas que utilizan los módulos ESP32. El asequible precio de estas placas de evaluación y la capacidad de programar los ESP32 con Arduino han contribuido a que el proceso de desarrollo de aplicaciones IoT (Internet de las cosas) sea más accesible y menos complejo, lo que ha impulsado la popularidad de este módulo entre entusiastas y profesionales que se encuentran en etapas de prototipado [50].

Arduino Mega

Arduino es un conjunto de dispositivos electrónicos que funcionan como microcontrolado-

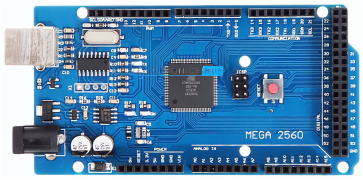


Figura 2.3: Arduino Mega 2560 [51].

res y permiten la adquisición de datos de código abierto. Estos dispositivos se basan en software libre y hardware flexible, lo que los hace fáciles de utilizar. Se utilizan para crear microcomputadoras en una sola placa con diversas aplicaciones, el software libre se refiere a programas informáticos de uso gratuito y que pueden ser personalizados por el usuario, Arduino utiliza la plataforma Arduino IDE para programar y adaptar sus dispositivos para una amplia variedad de aplicaciones [52].

Sensor de vibración SW-18010p

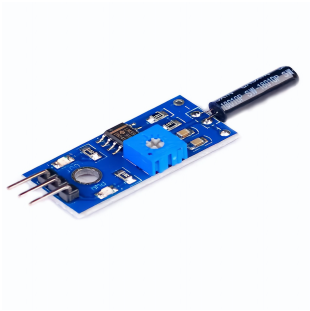


Figura 2.4: Sensor de vibración [53].

Conforme a su denominación, se emplea para registrar la vibración y/o el desplazamiento de un objeto. Con este módulo, es posible establecer alertas, como la detección de movimientos, como abrir una puerta. Además, se utiliza en aplicaciones lúdicas, como juguetes que generan sonidos en respuesta al movimiento, así como en otros sistemas donde se requiere evaluar la intensidad de la vibración [54].

Tabla 2.1: Características del sensor SW-18010p

Características	**Detalles**
Salida del comparador	señal limpia
Voltaje de funcionamiento	3v a 5v
Salida digital	0 y 1
Tamaño	3,1cm x 1,4 cm

Sensor de temperaura PT100

Figura 2.5: Sensor de temperatura PT-100 [55].

Los sensores Pt100 pertenecen a la categoría de detectores de temperatura conocidos como RTD (Detector de Temperatura de Resistencia). La característica distintiva y fundamental de los elementos Pt100 radica en su composición de platino, lo que les confiere una resistencia eléctrica de 100 ohmios a una temperatura de 0 °C. Este tipo de sensor RTD es, con mucho, el más ampliamente utilizado en la industria, en el caso de una sonda de temperatura Pt500, su resistencia es de 500 ohmios a 0 °C, mientras que una sonda Pt1000 presenta una resistencia de 1000 ohmios a la misma temperatura de referencia. Por lo general, los sensores Pt100 se insertan en una vaina o revestimiento protector para formar una sonda de temperatura más resistente, conocida comúnmente como "sonda Pt100.º "sonda de resistencia de platino Pt100"[56].

Tabla 2.2: Características del sensor de temperatura PT100

Características	Detalles
Rango de trabajo	-200ºC a +400ºC
Precisión	±0,3°C a 0ºC
Medidas de sonda	4mm x 30mm
Longitud de cable	100cm
Protección exterior	Acero inoxidable
Protección interna	Cerámica

Módulo max 31865

El Módulo transmisor MAX31865 ofrece una solución simple y fiable para obtener mediciones de temperatura de los sensores RTD PT100. Este dispositivo se encarga de la lectura de la resistencia del sensor PT100 y convierte la señal analógica en datos digitales interpretables por un microcontrolador. El chip MAX31865 está equipado con un conversor ADC delta-sigma de alta precisión que brinda una resolución de 15 bits, equivalente a 0,03125ºC. La interfaz de comunicación utiliza el protocolo SPI, lo que permite una integración sencilla con dispositivos como Arduino o PIC,el módulo MAX31865 es versátil y se puede configurar para trabajar con sensores PT100 de 2, 3 o 4 cables, aunque la configuración predeterminada es para sensores de 4 hilos [57].

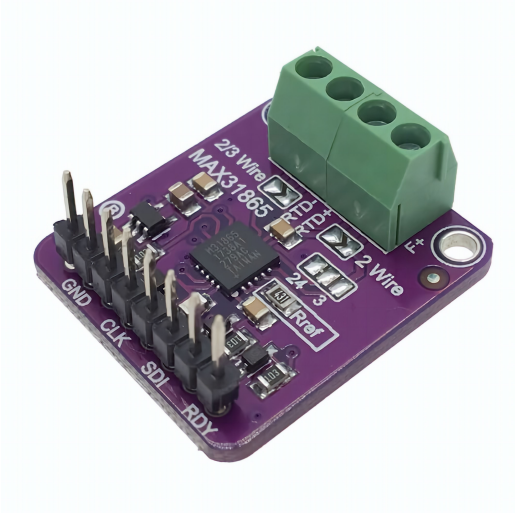


Figura 2.6: Modulo MAX 31865 [58].

Tabla 2.3: Características del módulo max 31865

Características	Detalles
Voltaje de operación	3 - 5VDC
Precisión	0,5ºC
Resolución ADC	15 bits (0.03125ºC)
Tiempo de conversión	21ms (máx)
Interfaz	SPI
Dimensiones	28,0x25,3x3,0 mm

Construcción de software

Como se muestra en la Figura 2.7. Se seleccionan herramientas de desarrollo, como PyQt5 y la arquitectura RCNN Faster Inception v2 de TensorFlow, para crear una interfaz de usuario y llevar a cabo la detección de objetos. La interfaz se diseña con PyQt5 para mostrar información en tiempo real sobre equipos de impresión 3D, datos de sensores y dirección de mirada. Luego, se entrena la red convolucional RCNN Faster Inception v2 con imágenes de los equipos, ajustando parámetros para una detección precisa. Se integran componentes de software, incluyendo el predictor facial y la interfaz, para funcionar conjuntamente. Se realizan pruebas exhaustivas para verificar el funcionamiento en tiempo real y se evalúan métricas de rendimiento.

Diagrama de la ejecución del software

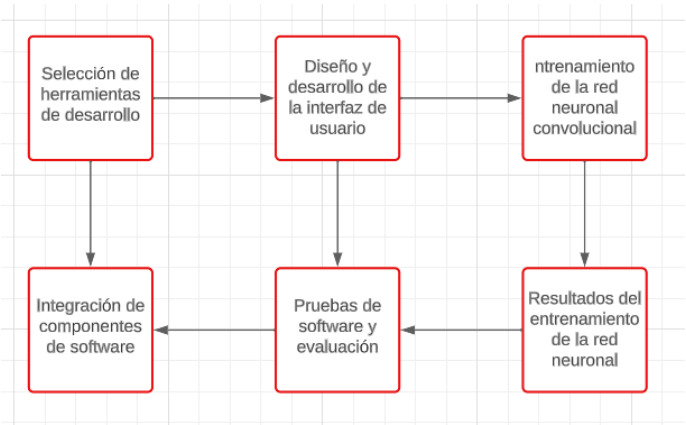


Figura 2.7: Diagrama de bloque de la ejecución del software

.

Diagrama de flujo de la programación

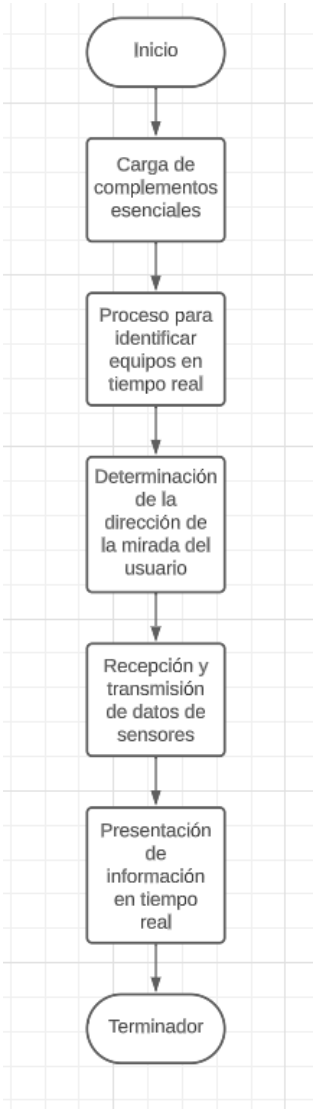


Figura 2.8: Diagrama de flujo de programación

.

2.4. Implementación

La figura 2.9indica la implementación del hardware en el sistema, se inicia con la selección de componentes IoT apropiados, como la ESP32 y el Arduino con DAQ, que permiten la adquisición de datos de sensores. Estos sensores incluyen mediciones de temperatura, humedad del ambiente y para la impresora 3D temperatura y vibración. La configuración del protocolo

MQTT en la ESP32 facilita la transmisión en tiempo real de los datos al servidor MQTT, que actúa como intermediario. Las pruebas exhaustivas se realizan para garantizar la precisión de los datos y el funcionamiento fiable del hardware en el sistema. En conjunto, el hardware desempeña un papel crucial en la adquisición de datos para el monitoreo y diagnóstico de equipos de impresión 3D en tiempo real.

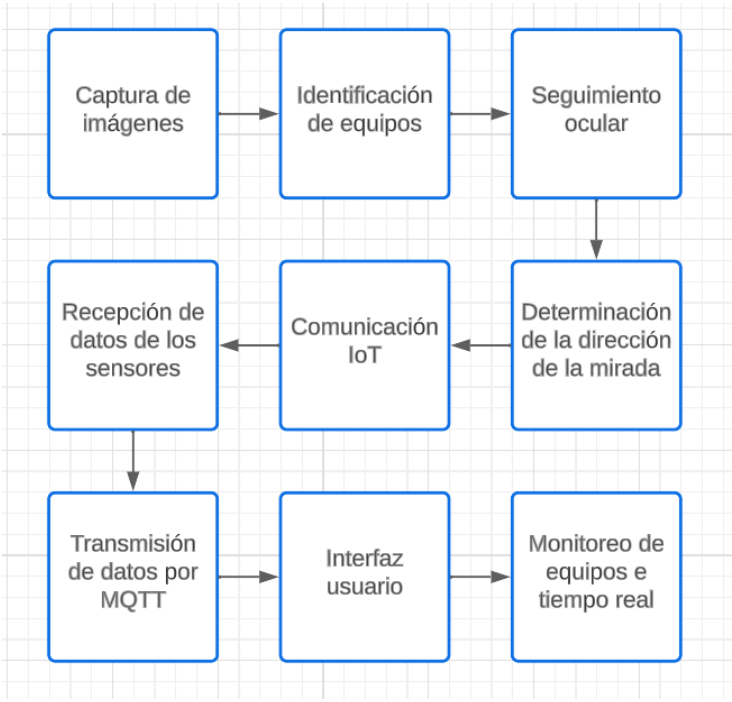


Figura 2.9: Diagrama de funcionamiento

.

La implementación del software se compone de la elección de herramientas de desarrollo específicas para cada módulo, como PyQt5 para la interfaz de usuario, TensorFlow con RCNN Faster Inception v2 para la detección de objetos y el predictor facial para el seguimiento ocular. Se desarrolla la interfaz de usuario para mostrar información en tiempo real, se entrena la red convolucional para la detección de objetos, y se integran eficazmente los componentes de software. Esto asegura la monitorización en tiempo real de equipos de impresión 3D y la provisión de información relevante a los usuarios y profesionales de mantenimiento.

2.5. Proceso de elaboración del algoritmo

2.5.1. Preparación del entorno y bibliotecas

Antes de comenzar el proceso, es esencial preparar el entorno de desarrollo. Se asegura que las bibliotecas y herramientas requeridas estén disponibles y configuradas correctamente. Las bibliotecas clave para el entrenamiento de detección de objetos incluyen TensorFlow, Keras y OpenCV. Estas herramientas proporcionan las capacidades necesarias para la construcción y el entrenamiento del modelo de detección.

Figura 2.10: Librerías para detección de objetos

La Figura 2.10 brinda una representación visual de los pasos necesarios para preparar el entorno de desarrollo. Se aprecia la correcta carga y configuración de las librerías relevantes, lo cual sienta las bases para la implementación exitosa del proceso de detección de objetos. Esta representación gráfica no solo visualiza el proceso, sino que también proporciona una guía visual que resulta valiosa para aquellos que desean llevar a cabo esta tarea de manera efectiva.

2.5.2. Preparación de datos y entrenamiento del modelo de detección

El proceso comienza con la preparación de datos, que implica la recopilación de un conjunto diverso de imágenes que representan los equipos de fresadora, torno CNC e impresora 3D en diversas condiciones de iluminación y perspectivas, mil imágenes de cada equipo, aproximadamente. Estas imágenes se anotan con etiquetas que indican la ubicación de los componentes clave. Posteriormente, el modelo de detección basado en RCNN Faster inception V2 se entrena utilizando este conjunto de datos anotado. El entrenamiento implica iteraciones de ajuste de

hiperparámetros y evaluaciones de rendimiento para lograr una detección precisa y rápida de los componentes.

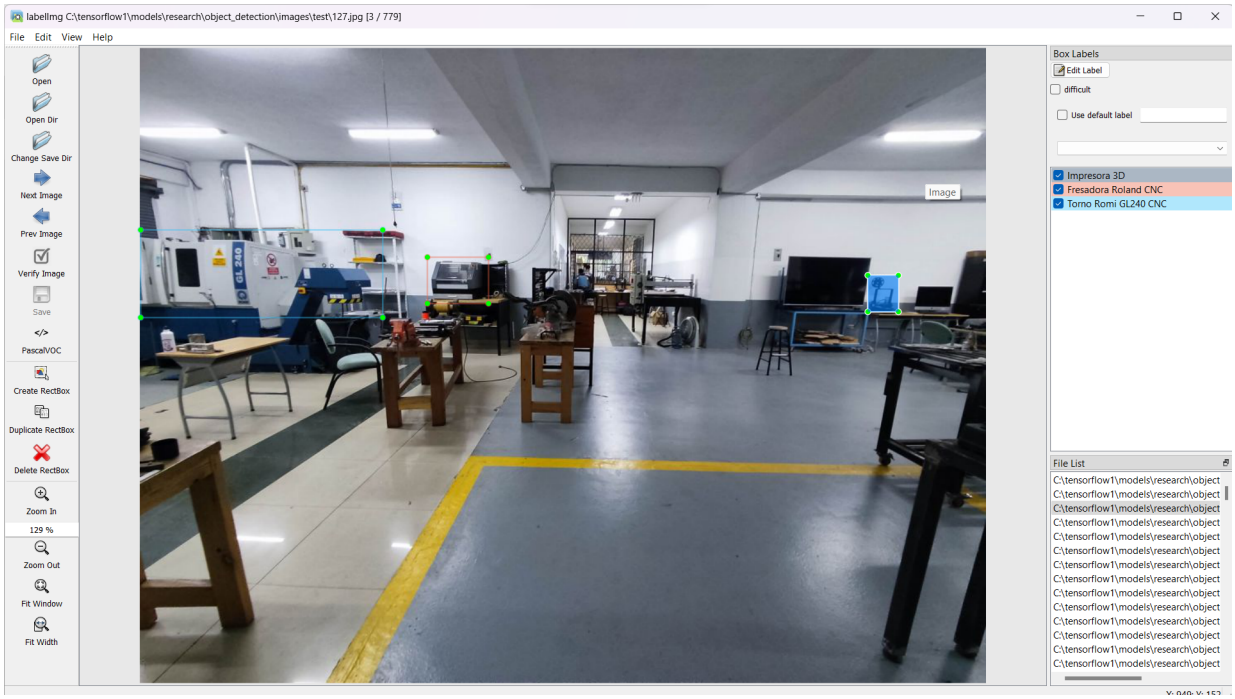


Figura 2.11: Etiquetado de imágenes

En la Figura 2.11 se puede observar las imágenes obtenidas y la interfaz Labellmg py, donde la imagen del laboratorio ocupa un lugar central, las áreas resaltadas se presentan en forma de cajas delimitadoras precisas, cada una de las cuales rodea un objeto en específico, estas cajas representan la forma en que Labellmg py identifica y anota los elementos clave.

La Figura 2.12 proporciona una visualización detallada del proceso de entrenamiento llevado a cabo, junto con los pasos correspondientes que ilustran la evolución de las pérdidas durante el entrenamiento y la cantidad de conocimiento que la máquina asimila en cada iteración.

2.5.3. Integración del seguimiento ocular

El código importa bibliotecas esenciales, como OpenCV, NumPy, dlib, argparse, imutils y serial para comunicación serial. Inicializa el detector de rostros dlib y el predictor de puntos de referencia faciales, preparando la cámara para capturar video en tiempo real. El programa realiza un seguimiento ocular detectando y extrayendo regiones de interés (ROIs) de los ojos derecho e izquierdo. Luego, aplica operaciones de procesamiento de imágenes, como desenfoque y umbralización, en estas ROIs. Posteriormente, se detectan contornos en las imágenes

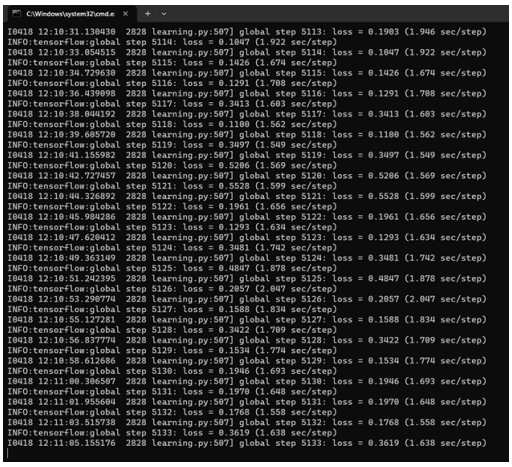


Figura 2.12: Entrenamiento del modelo utilizando la red neuronal convolucional RCNN Faster Inception V2

umbralizadas de los ojos, y se utiliza esta información para determinar la posición y el movimiento de los ojos. El programa define zonas de referencia, como izquierda, derecha, arriba y centro, para identificar la dirección de la mirada. La dirección de la mirada se muestra en la ventana de video, junto con las imágenes procesadas de los ojos. El programa incluye opciones de calibración y ajuste de umbrales, y permite reiniciar los valores de seguimiento cuando sea necesario. Para detener el programa, se presiona la tecla 'esc'. Simultáneamente, se integra la tecnología de seguimiento visual mediante Eye Tracker. El proceso de integración incluye la instalación y calibración precisa del rastreador ocular para adaptarlo a las características únicas de cada usuario. Se establece una comunicación bidireccional entre el módulo de detección de objetos y el seguimiento visual, lo que permite determinar el equipo específico observado por el usuario en función de su mirada.

Como se puede observar en la Figura 2.13 el proceso de seguimiento ocular, se capturan imágenes de la cámara y se detectan rostros en ellas. Luego, para cada rostro detectado, se localizan puntos de referencia faciales y se extraen regiones de interés (ROIs) correspondientes a los ojos derecho e izquierdo. Estas ROIs se someten a procesos de filtrado, como desenfoque y umbralización, y se buscan contornos en los ojos. Se determina la posición y el movimiento de los ojos, y se identifica la dirección de la mirada (izquierda, derecha, arriba o centro). Finalmente, la dirección de la mirada y las imágenes de los ojos procesados se muestran en una ventana de video.

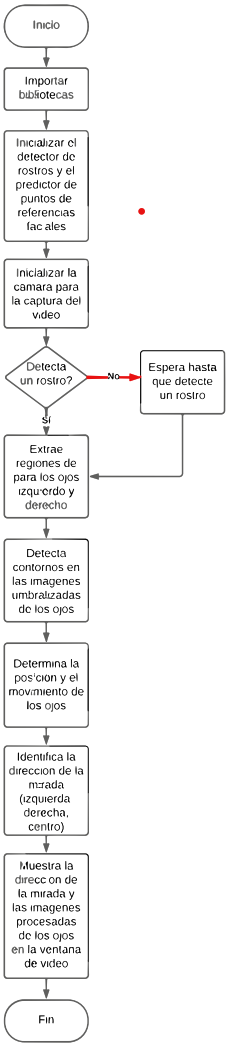


Figura 2.13: Diagrama de flujo de programación del seguimiento ocular

2.5.4. Comunicación IoT con MQTT y HiveMQ

En la ESP32 se realiza la programación en Arduino que consta de lo siguiente:

- **Configuración de red wifi:** el dispositivo se conecta a una red WiFi proporcionando el SSID (nombre de red) y la contraseña.
- **Configuración de comunicación MQTT:** Se establece la configuración para la comunicación MQTT, que incluye la información del servidor en HiveMQ, como la dirección del servidor y el puerto, así como detalles de autenticación, como el usuario y la contraseña. Además, se definen los temas de suscripción y publicación que se utilizarán para transmitir y recibir datos.

- **Bucle principal:** en cada iteración del bucle, se capturan datos de sensores, como temperatura, corriente, tiempo, humedad y un número de alerta.

  Estos datos se preparan para su transmisión a través del protocolo MQTT.

  El programa verifica si está conectado al servidor MQTT. En caso de no estarlo, intenta reconectar de manera automática.

  Si está conectado al servidor, publica los datos capturados en el tema correspondiente.

  El programa también monitorea mensajes entrantes y toma acciones basadas en su contenido. Por ejemplo, puede encender o apagar un dispositivo (como un compresor) o activar un relé según las instrucciones recibidas.

- **Callback MQTT:** Se define una función de callback que se encarga de manejar los mensajes recibidos. Esta función decodifica los mensajes y realiza acciones específicas en función de su contenido.

- **Serial event:** El dispositivo monitorea la comunicación serial en busca de datos entrantes y los procesa cuando se reciben, lo que le permite interactuar con otros dispositivos a través de la comunicación serial.

El proceso se repite continuamente en el bucle principal, lo que permite al dispositivo capturar, transmitir y actuar en función de los datos de sensores y las instrucciones MQTT, manteniendo una comunicación eficaz con la red WiFi y el servidor MQTT.

En resumen, este diagrama de flujo de la Figura 2.14 describe el proceso de configuración y comunicación de dispositivos IoT para adquirir y transmitir datos de sensores al servidor MQTT en HiveMQ, lo que permite la monitorización y recopilación de datos en tiempo real desde estos dispositivos.

DAQ en Arduino Como Indica la Figura 2.15 El código está diseñado para adquirir datos en tiempo real de varios sensores y transmitirlos a través de comunicación serial. Los sensores incluyen un sensor de temperatura y humedad DHT11, un sensor de temperatura PT100, y un sensor de vibración conectado a una entrada analógica. Los datos recopilados se envían a través de la comunicación serial tanto a la consola serie (Serial) como a otra interfaz serial (Serial2). Cada conjunto de datos contiene información sobre la temperatura, humedad, temperatura PT100, lectura analógica, un contador de tiempo y un valor constante. El código también verifica y maneja posibles fallos en el sensor PT100. Los datos se actualizan cada 500 milisegundos.

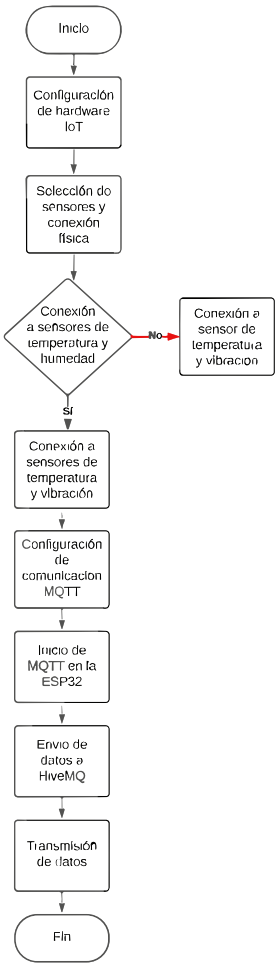


Figura 2.14: Diagrama de flujo de programación de ESP32

2.5.5. Adquisición de datos en tiempo real y visualización en el websocket

Recolección de datos de sensores: Adquisición de datos de los sensores, que incluyen temperatura, humedad y vibración.

Transmisión de datos a través de MQTT: Transmisión en tiempo real los datos de los sensores al servidor utilizando protocolos MQTT.

Visualización en el websocket: Configuramción de websocket con los topics correspondientes para mostrar información en tiempo real al usuario, incluyendo el equipo observado, los valores de sensores y otros datos relevantes.

La Figura 2.16 La interfaz de HiveMQ permite la visualización en tiempo real de los valores de múltiples sensores, incluyendo los sensores de temperatura y humedad del entorno, así como el sensor de temperatura y vibración de la impresora 3D.

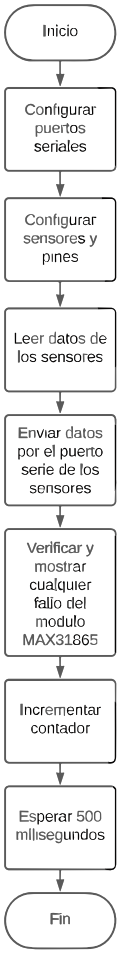


Figura 2.15: Diagrama de programación en Arduino Mega

Configuración del Websocket: Se implementa un servidor para establecer una comunicación bidireccional con la interfaz de usuario. La seguridad de la comunicación se garantiza mediante conexiones cifradas TLS/SSL.

Transmisión de datos en tiempo real: El servidor envía información actualizada constantemente al cliente, lo que incluye datos sobre equipos, valores de sensores y otros datos relevantes.

Interfaz de usuario: Los datos transmitidos a través del websocket se reflejan instantáneamente en la interfaz de usuario. Los cambios en la detección de objetos, la dirección de la mirada y los datos de sensores se actualizan dinámicamente.

Registro y almacenamiento de datos: Se implementa un registro de datos para conservar un historial de eventos y mediciones, permitiendo el acceso a información pasada y análisis de

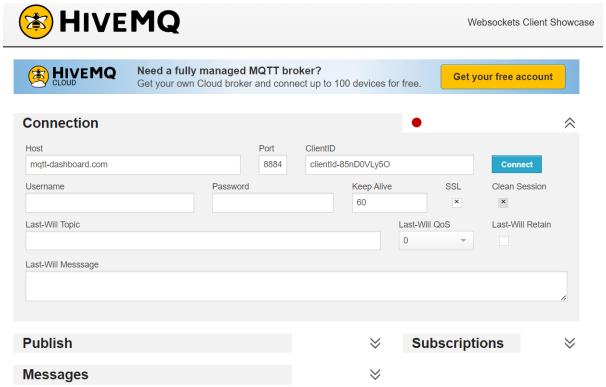


Figura 2.16: Interfaz HiveMQ [59].

tendencias.

Sincronización en múltiples dispositivos: Se garantiza que la visualización en tiempo real sea accesible en diferentes dispositivos, como computadoras, tabletas y dispositivos móviles.

Gestión de sesiones y seguridad: Se incluyen características de autenticación y seguridad para proteger la privacidad y la integridad de los datos transmitidos a través del websocket.

Optimización de la latencia: Se minimiza la latencia en la transmisión de datos para asegurar que la información llegue al usuario en tiempo real con la menor demora posible.

2.5.6. Interfaz PyQt5

En el proceso de diseño de la interfaz gráfica, se inicia definiendo una estructura visual intuitiva y atractiva. Se crean bocetos y mockups que consideran la disposición de elementos como paneles, gráficos y botones. Luego, se elige una paleta de colores y fuentes de texto para asegurar legibilidad y estética.

Seguidamente, se procede al desarrollo de la interfaz con PyQt5 en Python. Se crean ventanas principales, diálogos emergentes y componentes interactivos que permiten a los usuarios interactuar con el sistema. La interfaz se ajusta para adaptarse a las dimensiones de la pantalla del dispositivo. La conexión con módulos de procesamiento de datos permite que la información capturada, incluyendo la detección de objetos y el seguimiento ocular en tiempo real, se refleje de manera dinámica en la interfaz. Se establecen mecanismos de comunicación bidireccional para que las acciones del usuario afecten a los módulos subyacentes.

La interfaz muestra en tiempo real la detección de objetos, resaltando equipos de impresión 3D identificados con etiquetas descriptivas que indican su estado. Además, se integra la

representación gráfica de la dirección de la mirada del usuario. Se presentan datos de sensores en gráficos y tablas en tiempo real, incluyendo valores de temperatura, humedad y vibración, proporcionando contexto para entender su relevancia en el estado de los equipos de impresión 3D.

La interfaz se actualiza continuamente para reflejar cambios en la detección de objetos, dirección de la mirada y datos de sensores, garantizando que los usuarios tengan acceso a la información más reciente. Se implementan elementos interactivos para que los usuarios puedan tomar decisiones informadas y realizar acciones específicas, como seleccionar equipos para obtener detalles adicionales o iniciar procedimientos de mantenimiento.

Diagrama de flujo de la programación de la interfaz con PyQt5

El procedimiento de la Figura 2.17 juega un papel crucial en la operación y gestión eficiente de los equipos de impresión 3D. Al permitir una visualización en tiempo real de datos relevantes, como la detección de objetos, la dirección de la mirada del usuario y los valores de los sensores, los usuarios pueden tomar medidas inmediatas cuando se detectan problemas o condiciones anómalas. Esto se traduce en una mayor eficacia operativa, una reducción de los tiempos de inactividad no planificados y un ahorro en costos de mantenimiento al abordar los problemas en las primeras etapas. Además, la integración de funciones de detección de objetos y seguimiento ocular en el algoritmo principal ofrece una experiencia de usuario más intuitiva. La capacidad de asociar la dirección de la mirada con los objetos detectados permite una identificación más precisa de los equipos de impresión 3D observados.

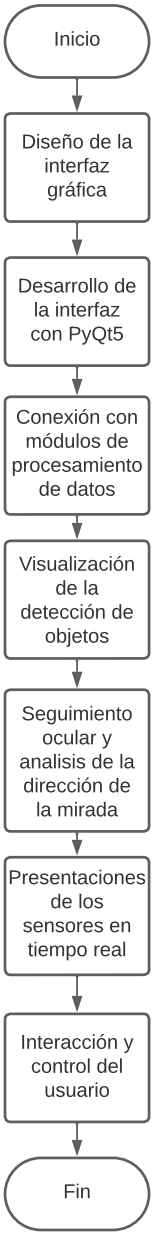


Figura 2.17: Diagrama de flujo programación interfaz gráfica

2.5.7. Algoritmo final

Se han definido funciones clave para la detección de objetos y el seguimiento ocular, que operan en imágenes de entrada y proporcionan información sobre los equipos de impresión 3D

detectados y la dirección de la mirada del usuario. Estas funciones se integran en el algoritmo principal, donde se importan los módulos de detección de objetos y seguimiento ocular. Luego, se llaman a estas funciones, pasando la imagen capturada en tiempo real como entrada.

La lógica del algoritmo principal asocia los resultados de la detección de objetos con la dirección de la mirada, permitiendo identificar el equipo de impresión 3D que el usuario está observando cuando la dirección de la mirada coincide con un objeto detectado.

Además, se ha integrado una función de comunicación MQTT que envía datos de detección de objetos y dirección de la mirada al servidor MQTT, junto con los valores de los sensores recopilados por el hardware IoT.

La interfaz gráfica, desarrollada con PyQt5, se mantiene como un módulo independiente. Utiliza funciones específicas para actualizar la pantalla continuamente y mostrar información actualizada. La interfaz también recibe datos de detección de objetos, dirección de la mirada y datos de sensores a través de funciones diseñadas para la comunicación entre los módulos.

Para compartir los datos en tiempo real con los usuarios en cualquier ubicación, se ha implementado una función que conecta la interfaz gráfica con el servidor MQTT de HiveMQ a través del websocket. Esto permite una comunicación bidireccional y garantiza que los usuarios tengan acceso a datos actualizados de manera instantánea. La programación en cascada de estas funciones proporciona un aumento en la eficiencia y la sincronización de todas las partes del sistema. La Figura 2.18 indica todo el proceso de construcción del algoritmo final.

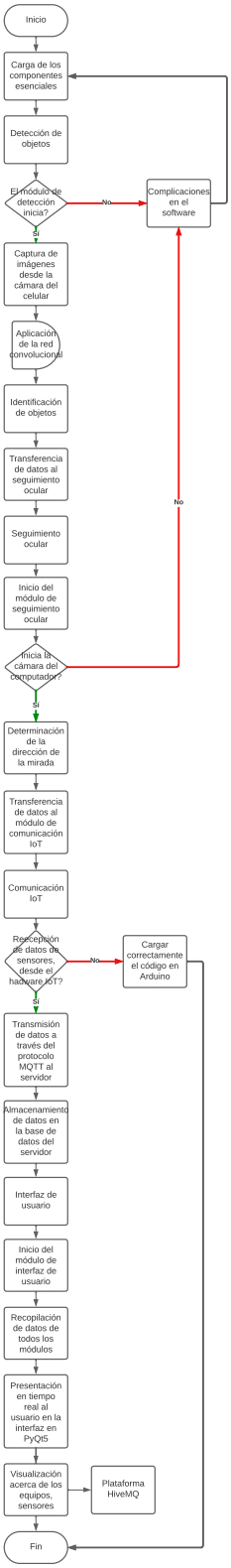


Figura 2.18: Diagrama de flujo programación completa

Capítulo III

Resultados y discusión

En esta sección, se presenta el punto culminante del trabajo de investigación, que busca mejorar el mantenimiento de un equipo de impresión 3D. Se ha desarrollado un algoritmo innovador que integra tecnologías de visión artificial, seguimiento ocular y comunicación IoT en tiempo real. Los resultados y el análisis que se presentan aquí ofrecen una visión detallada de los logros y descubrimientos clave obtenidos durante la implementación de la solución propuesta.

3.1. Comparación redes neuronales

Para la comparación, se emplearon dos arquitecturas de redes neuronales convolucionales para tareas de detección de objetos: SSD MobileNet V2 y Faster R-CNN con Inception V2. Estas dos arquitecturas son ampliamente reconocidas en el campo de la visión por computadora y se utilizaron en el estudio con el propósito de analizar su desempeño y comparar sus resultados.

3.1.1. SSD MobileNet V2

La Figura 3.1 muestra una visión detallada del proceso de entrenamiento de un modelo de detección de objetos en TensorFlow, abarcado un total de 50000 pasos de entrenamiento. Estos registros muestran el número de pasos de entrenamiento, el tiempo promedio necesario para cada paso y métricas relacionadas con la función de pérdida. Estas métricas incluyen la pérdida de localización que mide la precisión en la determinación de la posición de los objetos, la pérdida global minimizada durante el entrenamiento y la tasa de aprendizaje, que indica la tasa de ajuste de los pesos del modelo. A partir del paso 49900, las métricas no muestran mejoras significativas, lo que indica que el modelo ha alcanzado su límite de aprendizaje.

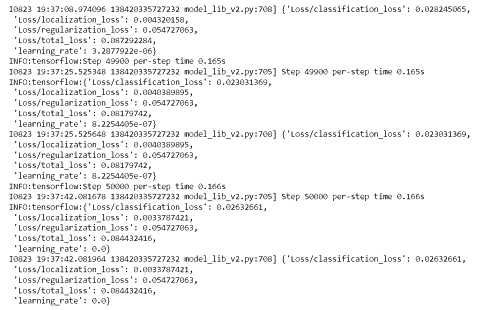


Figura 3.1: Entrenamiento del modelo

Tabla 3.1: Evaluación del modelo SSD Mobilenet V2

Promedio	**IoU**	**Área**	**Detecciones max. Total**	
AP	0,50:0,95	completa	100	0,684
AP	0,50	completa	100	0,978
AP	0,75	completa	100	0,848
AP	0,50:0,95	pequeña	100	0,259
AP	0,50:0,95	mediana	100	0,582
AP	0,50:0,95	grande	100	0,722
AR	0,50:0,95	completa	1	0,732
AR	0,50:0,95	completa	10	0,737
AR	0,50:0,95	completa	100	0,740
AR	0,50:0,95	pequeña	100	0,406
AR	0,50:0,95	mediana	100	0,652
AR	0,50:0,95	grande	100	0,776

La tabla 3.1 muestra los resultados de la evaluación de un modelo de detección de objetos después de completar 50.000 pasos de entrenamiento. Los resultados de la evaluación se presentan en términos de métricas de precisión promedio (AP) y métricas de recall promedio (AR) para diferentes umbrales de superposición (IoU) y áreas de detección (pequeña, mediana y grande).

- El AP representa la precisión promedio del modelo en la detección de objetos.
- El AR representa el recall promedio del modelo en la detección de objetos.
- Las métricas se dividen en diferentes categorías según el IoU y el tamaño de las áreas.
- Las métricas incluyen la precisión y el recall a diferentes niveles de superposición y tamaños de área.

En este caso, los resultados indican que el modelo alcanzó una precisión promedio (AP) de alrededor del 0,684 en un rango de umbrales de superposición y áreas, lo que proporciona información sobre su desempeño en la tarea de detección de objetos.

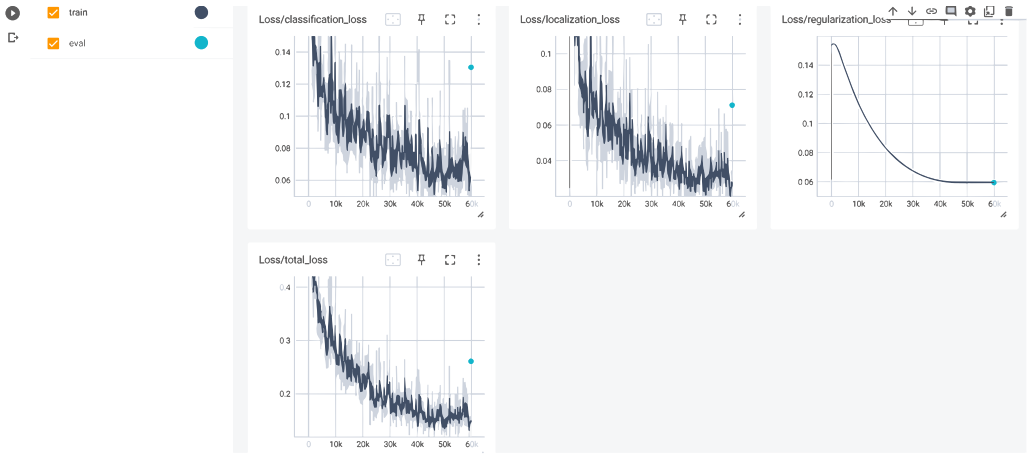


Figura 3.2: Gráficas de pérdidas del entrenamiento

La Figura 3.2 indica la evaluación de las pérdidas del modelo entrenado. En el eje vertical (eje Y), se muestran los valores de las pérdidas, mientras que en el eje horizontal (eje X), se representan los pasos o interaciones de entrenamiento. Estas gráficas permiten observar cómo las pérdidas varían a lo largo del proceso de entrenamiento, lo que facilita la evaluación del rendimiento del modelo.

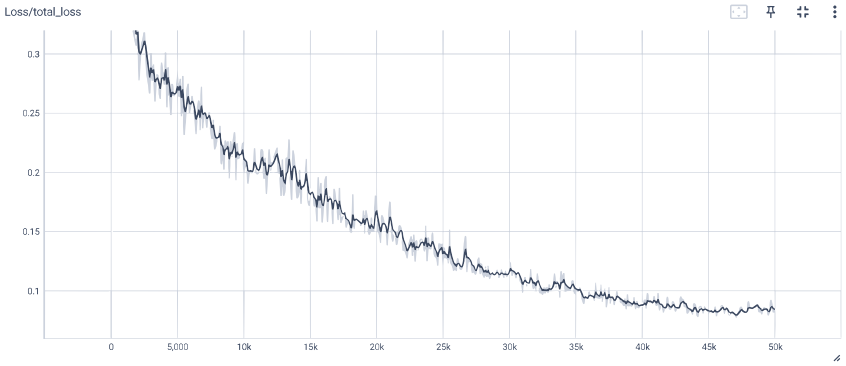


Figura 3.3: Gráfica de pérdida total del entrenamiento

En la Figura 3.3 se presenta de manera más detallada el gráfico de pérdida total. La curva exhibe una tendencia a mantenerse suave, sin presentar fluctuaciones significativas, y se estabiliza en un nivel de pérdida cercano a 0,1.

Figura 3.4: Inferencia del modelo con una impresora 3D

Se realizó un código que emplea TensorFlow para llevar a cabo la detección de objetos en una imagen de entrada utilizando un modelo previamente entrenado. En primer lugar, cargó la imagen y convirtió su contenido en un tensor, a continuación, se aplicó el modelo y se extrajo información sobre las detecciones. Luego, visualizó los resultados en la imagen original, superponiendo cuadros y etiquetas en los objetos detectados. En la Figura 3.4 se refleja una precisión del detector de objetos del 94 % al identificar una impresora 3D.

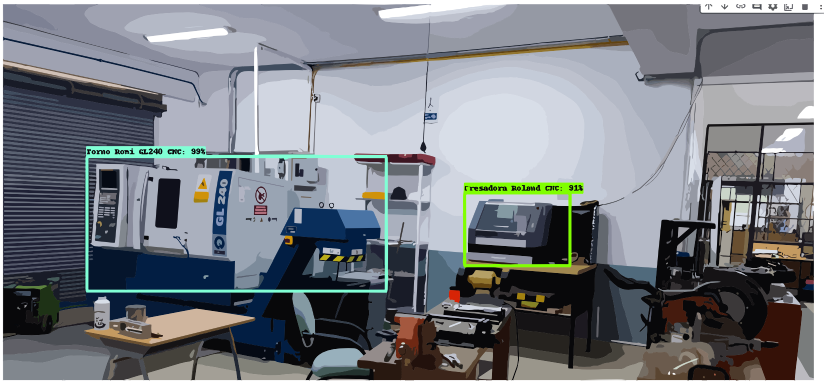


Figura 3.5: Inferencia del modelo con dos equipos del laboratorio de mecatrónica

En la Figura 3.5 se ilustra una precisión del 91 % en la detección de la Fresadora Roland CNC, junto con una precisión del 99 % del torno Romi GL240 CNC, ambos ubicado en el laboratorio de mecatrónica.

3.1.2. Tiempo de inferencia del modelo entrenado

Se realizó un script que implementa la detección de objetos en un conjunto de aproximadamente 500 imágenes de prueba, el 20 % del conjunto de datos, utilizando un modelo preentrenado en TensorFlow. Durante la ejecución, se registra el tiempo de inferencia para cada imagen. Los resultados de la detección, que incluyen cuadros delimitadores y etiquetas, se visualizan en las imágenes originales. Este proceso permite evaluar el rendimiento del modelo en el conjunto de datos de prueba y proporciona información valiosa, como el tiempo de inferencia asociado a cada imagen procesada. La ejecución se realizó en la carpeta "test"del dataset de entrenamiento. Para calcular el tiempo de inferencia se aplica la siguiente ecuación:

$$\text{Tiempo de inferencia} = (\text{tiempo final} - \text{tiempo inicial}) \times 1000 \tag{3.1}$$

Para el tiempo de inferencia por muestra se utiliza la siguiente ecuación:

$$\text{Tiempo de Inferencia por Muestra} = \frac{\text{Tiempo de Finalización} - \text{Tiempo de Inicio}}{N} \tag{3.2}$$

Donde:

- Tiempo de Finalización es el momento en el que se completa la inferencia para todas las muestras.
- Tiempo de Inicio es el momento en el que se inicia la inferencia.
- N es el número de muestras.

El tiempo de inferencia, derivado del cálculo según la ecuación 3.2, representa la duración que el modelo de detección de objetos necesita para identificar y procesar un objeto en una muestra dada. En un sentido más detallado, este intervalo de tiempo abarca desde el inicio hasta la finalización de la inferencia en un conjunto de datos de N tamaño. Durante este lapso, el modelo ejecuta operaciones de preprocesamiento, realiza la inferencia en cada muestra individual y ejecuta operaciones de postprocesamiento en los resultados obtenidos. Como se muestra en la Figura 3.6 da como resultado 2821,89 milisegundos.

Es crucial señalar que este tiempo de inferencia no solo evalúa la eficacia del modelo en la clasificación de objetos, sino que también considera las operaciones adicionales, como la preparación de la entrada y la manipulación de los resultados, proporcionando así una métrica comprehensiva del rendimiento del modelo en términos de velocidad y precisión durante la

Figura 3.6: Tiempo de inferencia del modelo con SSD MobileNet

identificación de objetos en el conjunto de datos específico.

El tiempo de inferencia general se determinó utilizando imágenes de prueba con el modelo de detección de objetos. Este tiempo se calculó mediante la medición del tiempo de inicio y fin de la ejecución del modelo para cada imagen. Este tiempo de inferencia proporciona una medida cuantitativa del rendimiento en la detección de objetos en las imágenes de prueba. En la Figura 3.7 indica los tiempos de inferencia del modelo entrenado, utilizando los nombres de las imágenes tal como están nombradas en la carpeta del conjunto de datos.

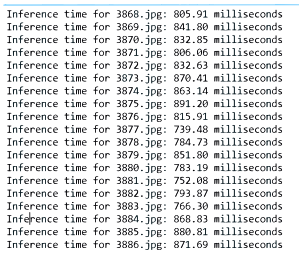


Figura 3.7: Tiempo de inferencia por imagen

Sumando todos los tiempos de inferencia proporcionados, se obtiene un total de 372911,87 milisegundos. Para calcular el promedio, se divide esta suma por el número total de imágenes, que es 500. El promedio de tiempo de inferencia para estas imágenes específicas es:

$$\text{Promedio} = \frac{594133,19}{500} \approx 1188{,}27 \text{ milisegundos por imagen} \tag{3.3}$$

3.1.3. RCNN Faster Inception V2

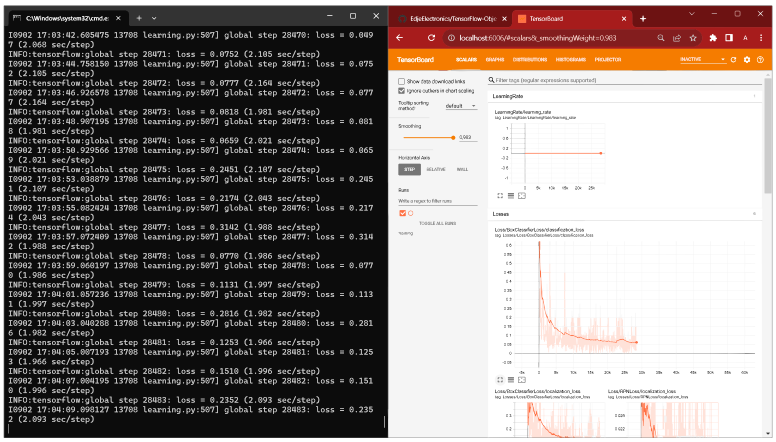


Figura 3.8: Entrenamiento con red neuronal convolucional

En el entrenamiento utilizando la red neuronal convolucional Faster RCNN inception V2, se llevaron a cabo 30000 pasos de entrenamiento. El gráfico de la Figura 3.8 proporciona información sobre las pérdidas por paso y el tiempo que requiere cada paso del proceso. El "global step"se refiere al número total de pasos que se han realizado durante el entrenamiento, lo que incluye tanto los pasos anteriores como los actuales. Este valor es una métrica importante para seguir el proceso y evaluar cuántos pasos se han realizado hasta el momento. El gráfico muestra como evolucionan las pérdidas a medida que los pasos avanzan y cuánto tiempo toma cada uno, lo que es esencial para monitorear y ajustar el proceso de entrenamiento.

La tabla 3.2 indica la evaluación del modelo.

El modelo fue evaluado siguiendo un procedimiento similar al descrito en la tabla 3.1, en el cual se presentan los distintos parámetros evaluados. Estos incluyen el promedio, IoU (índice de superposición de intersección), la clasificación y el resultado de la evaluación.

Se realiza una evaluación de las pérdidas del modelo entrenado, donde en la Figura 3.9, el eje vertical (eje Y) representa los valores de las pérdidas, mientras que en el eje horizontal (eje X) se representan los pasos o iteraciones de entrenamiento en este caso con 30000 pasos. Estas representaciones gráficas permiten visualizar cómo cambian las pérdidas a medida que avanza el proceso de entrenamiento, lo que resulta útil para valorar el desempeño del modelo en diferentes etapas del entrenamiento.

Tabla 3.2: Evaluación del modelo Faster RCNN inception V2

Promedio	**IoU**	**Área**	**Detecciones max. Total**	
AP	0,50:0,95	completa	100	0,934
AP	0,50	completa	100	0,982
AP	0,75	completa	100	0,836
AP	0,75:0,95	completa	100	0,959
AP	0,95	mediana	100	0,673
AP	0,95	grande	100	0,882
AR	0,95	completa	100	0,843
AR	0,80	completa	100	0,848
AR	0,95	completa	100	0,864
AR	0,95	completa	100	0,967
AR	0,95	mediana	100	0,786
AR	0,50:0,95	grande	100	0,884

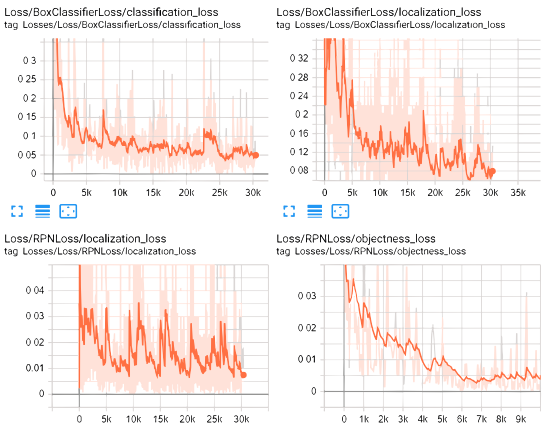


Figura 3.9: Gráficas del entrenamiento de la arquitectura Faster RCNN inception V2

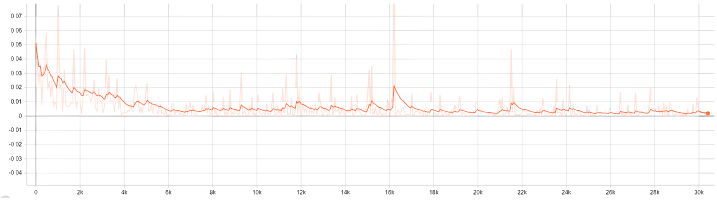


Figura 3.10: Gráfica de la pérdida total del entrenamiento

En la Figura 3.10 se representa el gráfico de pérdidas de la red convolucional Faster R-CNN Inception v2. Un aspecto destacado es la ausencia de picos pronunciados en la gráfica, lo que

indica una estabilidad en el proceso de entrenamiento. A partir de aproximadamente 25000 pasos, las pérdidas se mantienen constantes alrededor de 0,1. Sin embargo, es relevante señalar que esta red neuronal establece que los valores óptimos para un detector son aquellos por debajo de 0,05. Esto sugiere que, aunque la red ha alcanzado una relativa estabilidad, aún podría existir margen para mejorar el rendimiento del detector en términos de la pérdida.

3.1.4. Tiempo de inferencia del modelo entrenado

Para calcular el tiempo de inferencia del modelo entrenado, se empleó la ecuación 3.2, cuyo resultado se presenta en la Figura 3.11 23,59 milisegundos.

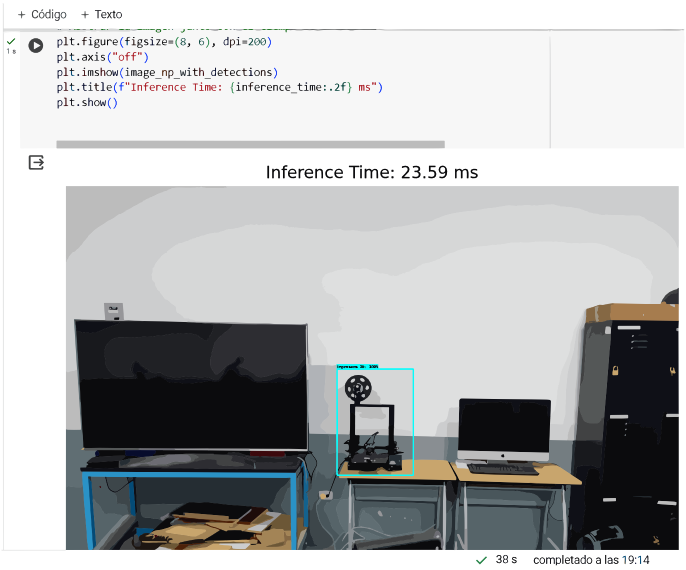
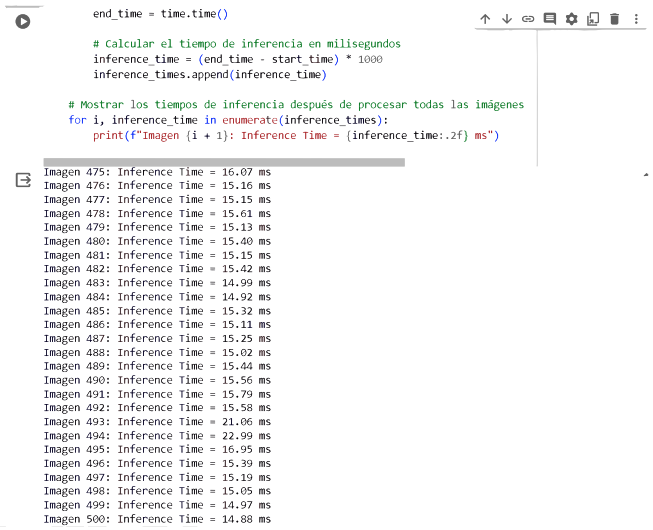


Figura 3.11: Tiempo de inferencia por muestra

Para determinar el tiempo de inferencia general del modelo, se aplicó la ecuación 3.1. Los valores resultantes de esta ecuación se exhiben en la Figura 3.12. Posteriormente, estos valores fueron obtenidos mediante la ejecución de un programa en Python, que recopiló información necesaria. La suma de estos valores proporcionó el resultado final del tiempo de inferencia general. Este enfoque garantiza una evaluación exhaustiva y precisa del rendimiento del modelo en términos de tiempo de inferencia, lo que contribuye a una comprensión más profunda de su eficiencia y capacidad de respuesta en situaciones del mundo real.



```
        end_time = time.time()

        # Calcular el tiempo de inferencia en milisegundos
        inference_time = (end_time - start_time) * 1000
        inference_times.append(inference_time)

    # Mostrar los tiempos de inferencia después de procesar todas las imágenes
    for i, inference_time in enumerate(inference_times):
        print(f"Imagen {i + 1}: Inference Time = {inference_time:.2f} ms")
```

```
Imagen 475: Inference Time = 16.07 ms
Imagen 476: Inference Time = 15.16 ms
Imagen 477: Inference Time = 15.15 ms
Imagen 478: Inference Time = 15.61 ms
Imagen 479: Inference Time = 15.13 ms
Imagen 480: Inference Time = 15.40 ms
Imagen 481: Inference Time = 15.15 ms
Imagen 482: Inference Time = 15.42 ms
Imagen 483: Inference Time = 14.99 ms
Imagen 484: Inference Time = 14.92 ms
Imagen 485: Inference Time = 15.32 ms
Imagen 486: Inference Time = 15.11 ms
Imagen 487: Inference Time = 15.25 ms
Imagen 488: Inference Time = 15.02 ms
Imagen 489: Inference Time = 15.44 ms
Imagen 490: Inference Time = 15.56 ms
Imagen 491: Inference Time = 15.79 ms
Imagen 492: Inference Time = 15.58 ms
Imagen 493: Inference Time = 21.06 ms
Imagen 494: Inference Time = 22.99 ms
Imagen 495: Inference Time = 16.95 ms
Imagen 496: Inference Time = 15.39 ms
Imagen 497: Inference Time = 15.19 ms
Imagen 498: Inference Time = 15.05 ms
Imagen 499: Inference Time = 14.97 ms
Imagen 500: Inference Time = 14.88 ms
```

Figura 3.12: Tiempo de inferencia del modelo general

$$\text{Promedio} = \frac{261801{,}76}{500} \approx 745{,}82 \,\text{milisegundos por imagen} \tag{3.4}$$

Tabla 3.3: Tiempos de inferencia

SSD Mobilenet V2	**RCNN Faster Inception**
1188,27 (ms)	745,82 (ms)
1,1883 (s)	0,7458 (s)

Como muestra la Tabla 3.3, la arquitectura de red neuronal Faster Inception R-CNN destaca con un tiempo de 745,82 milisegundos, superando significativamente a la SSD MobileNet, que registra un tiempo de 1188,27 milisegundos. Esta diferencia de tiempo subraya la capacidad superior para realizar inferencias de detección de objetos de manera más eficiente.

Tras analizar los valores en las tablas 3.1 y 3.2, se calculó un promedio de las precisiones totales, lo que resultó en los siguientes datos: la red neuronal SSD MobileNet V2 exhibió una precisión promedio del 62,66 %, mientras que la Faster R-CNN Inception V2 alcanzó una precisión promedio del 87,15 %. Como resultado de esta comparación, se optó por utilizar la red neuronal convolucional Faster R-CNN Inception V2 debido a su sustancialmente mayor

precisión en comparación con la SSD MobileNet. Esto respalda la elección de este modelo de entrenamiento como la mejor opción para la investigación.

Tabla 3.4: Precisión de las arquitecturas de entrenamiento

SSD MobileNet V2	**Faster RCNN inception V2**
0,6266	0,8715
62,66 %	87,15 %

Una vez completado el entrenamiento con la red neuronal convolucional, En la Figura 3.13 se puede apreciar un notable avance en la detección de objetos en comparación con la otra red neuronal SSD MobileNet v2 de la Figura 3.4 La imagen revela mejoras significativas, y el detector asigna con un 99 % de confianza la identificación del objeto como una impresora 3D. Este resultado sugiere que la red neuronal convolucional ha logrado una mayor precisión en la tarea de reconocimiento de objetos en comparación con el modelo anterior, proporcionando así una mejora sustancial en la calidad de la detección.

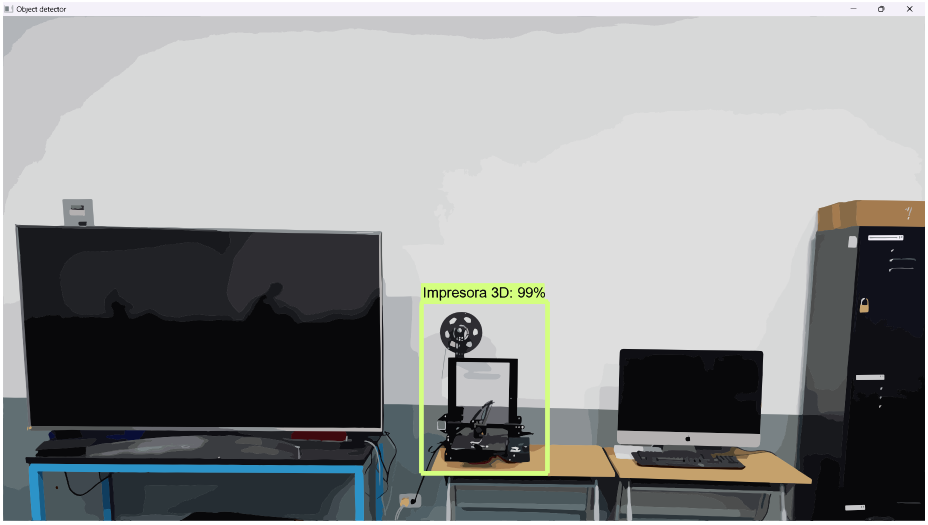


Figura 3.13: Inferencia del modelo

Después de analizar otra imagen del laboratorio de mecatrónica mediante el modelo de la red neuronal convolucional. En la Figura 3.14 se evidencia una mejora significativa en la precisión al identificar los distintos equipos previamente entrenados. En particular, la fresadora Roland CNC y el torno Romi GL240 son reconocidos con una asombrosa precisión del 99 % cada uno. Este nivel de exactitud resalta la eficacia y confiabilidad de la red neuronal utilizada en este trabajo de investigación.

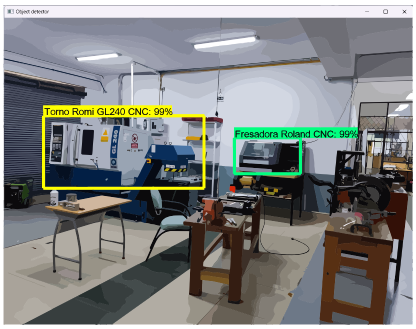


Figura 3.14: Inferencia del modelo en el torno y fresadora del laboratorio

3.2. Implementación de detección de objetos en tiempo real

Después de completar la comparación entre las redes neuronales, una vez que el modelo de detección ha sido entrenado, la implementación de la detección de objetos en tiempo real se lleva a cabo con eficiencia. Este proceso implica una secuencia de pasos diseñados para optimizar el rendimiento del modelo entrenado y proporcionar información precisa sobre los equipos de laboratorio. Cabe destacar que, según las comparaciones anteriores, la red neuronal convolucional Faster R-CNN Inception V2 ha demostrado ser superior a la SSD MobileNet en términos de rendimiento y precisión.

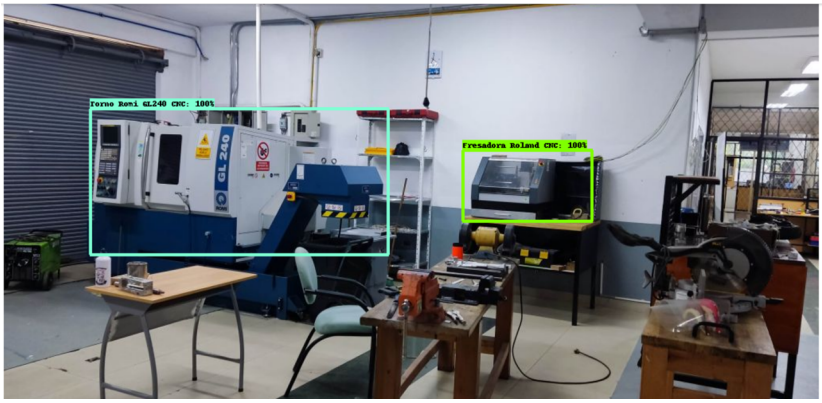


Figura 3.15: Funcionamiento de la detección de objetos

La Figura 3.15 expone con claridad la ejecución exitosa del algoritmo de detección de objetos. En cada recuadro del gráfico, se puede apreciar tanto la precisión del algoritmo como la correspondencia entre las detecciones y el conocimiento que la máquina ha adquirido previamente. Este diagrama brinda una representación visual completa de cómo el algoritmo interpreta y analiza datos en tiempo real. La precisión de las detecciones se refleja en la concordancia con

los objetos reales presentes en la imagen. Cada recuadro muestra cómo el modelo identifica y delimita los objetos de interés con una alta coincidencia entre las predicciones y la realidad.

3.3. Base de datos

La información de los equipos se ha almacenado en una etiqueta (label) dentro del algoritmo de detección de objetos. Esta etiqueta, asociada al equipo seleccionado, proporciona de manera específica las características y especificaciones del equipo correspondiente.

3.4. Integración del seguimiento visual

El código en Python emplea OpenCv y dlib para realizar un seguimiento en tiempo real de la dirección de la mirada del usuario, específicamente mediante el seguimiento ocular. En cada frame del vídeo. identifica rostros, extrae regiones de interés para los ojos, aplica filtros y técnicas de binarización, y determina la dirección de la mirada según la posición relativa de los ojos. La información sobre la dirección de la mirada se muestra de manera visual en la pantalla, como se muestra en la Figura 3.16

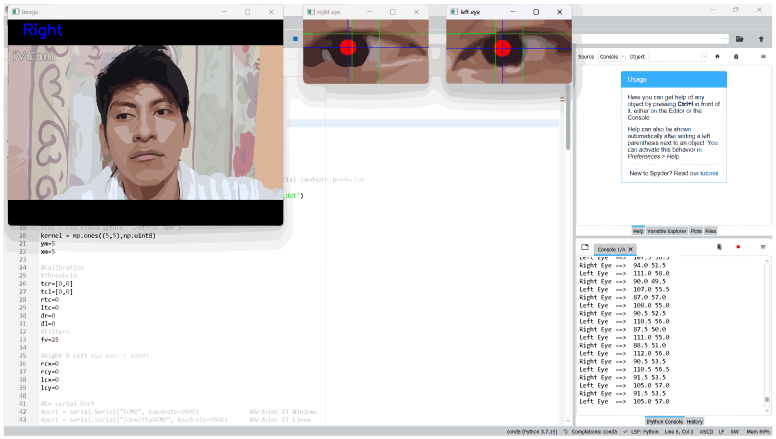


Figura 3.16: Funcionamiento del seguimiento ocular

La implementación exitosa de este algoritmo de seguimiento ocular, junto con el predictor, no solo garantiza un seguimiento preciso de los movimientos oculares, sino que también se integra perfectamente en el código final del sistema. Permite a los usuarios visualizar las especificaciones de los equipos presentes, al tiempo que el algoritmo identificará de manera singular y precisa un único elemento en lugar de múltiples simultáneamente. Esta funcionalidad se lleva

a cabo en un entorno controlado, asegurando una detección confiable y optimizada. Los resultados obtenidos confirman la capacidad del sistema para simplificar la interacción del usuario al proporcionar información detallada sobre los equipos, contribuyendo así a una experiencia más eficiente y precisa en el análisis de datos y la identificación de objetos específicos en el entorno. Para el funcionamiento del código, se emplearon tanto operaciones geométricas como lógicas. Específicamente, para calcular las coordenadas del centro del ojo, se utilizaron las siguientes ecuaciones: Ojo derecho:

$$rcx = x + \frac{w}{2} \tag{3.5}$$

$$rcy = y + \frac{h}{2} \tag{3.6}$$

Ojo izquierdo:

$$lcx = x + \frac{w}{2} \tag{3.7}$$

$$lcy = y + \frac{h}{2} \tag{3.8}$$

Donde, 'x' y 'e' y son las coordenadas de la esquina superior izquierda del rectángulo que rodea el ojo, y 'w' y 'h' son el ancho y la altura del rectángulo, respectivamente. Se calculan las coordenadas del centro del ojo sumando la mitad del ancho y la mitad de la altura a las coordenadas de la esquina superior izquierda.

Definición de Regiones de Interés (ROI):

Límite izquierdo ('lbx'):

$$lbx = \text{int}\left(\frac{\text{cols}}{9} \times 3{,}5\right) \tag{3.9}$$

Límite derecho ('rbx'):

$$rbx = \text{int}\left(\frac{\text{cols}}{9} \times 5{,}5\right) \tag{3.10}$$

Límite superior ('uby'):

$$uby = \text{int}\left(\frac{\text{rows}}{9} \times 2\right) \tag{3.11}$$

Donde: int = Función de redondeo al entero más cercano, utilizada para asegurar que el resultado de una operación sea un número entero.

cols = Representa el ancho de la imagen en píxeles.

rows = Representa la altura de la imagen.

3.4.1. Evaluación

Comparación de coordenadas mediante código de OpenCV
Comparación para el límite izquierdo ('lbx'):

$$\text{if } rcx < lbx \text{ or } lcx < lbx: \quad ... \tag{3.12}$$

Determina si el centro del ojo derecho ('rcx') o del ojo izquierdo ('lcx') está a la izquierda del límite definido ('lbx'). Utilizado para identificar movimientos oculares hacia la izquierda.

Comparación para la región central ('lbx', 'rbx', 'uby'):

$$\text{if } lbx \leq rcx \leq rbx \text{ and } lbx \leq lcx \leq rbx \text{ and } uby \leq rcy \text{ and } uby \leq lcy: \quad ... \tag{3.13}$$

Verifica si los centros de ambos ojos ('rcx' y 'lcx') están entre los límites izquierdo ('lbx') y derecho ('rbx'), y si las coordenadas y ('rcy' y 'lcy') están por encima del límite superior ('uby'). Utilizado para reconocer si los ojos están en el centro.

Comparación para el límite derecho ('rbx'):

$$\text{if } rcx > rbx \text{ or } lcx > rbx: \quad ... \tag{3.14}$$

Indica si el centro del ojo derecho ('rcx') o del ojo izquierdo ('lcx') está a la derecha del límite establecido ('rbx'). Se emplea para detectar movimientos oculares hacia la derecha.

Comparación para el límite superior ('uby'):

$$\text{if } rcy \leq uby \text{ and } lcy \leq uby: \quad ... \tag{3.15}$$

Verifica si las coordenadas y de ambos ojos ('rcy' y 'lcy') están por debajo del límite superior ('uby'). Utilizado para reconocer movimientos oculares hacia arriba.

En un conjunto de datos donde se registraron 30 instancias de mirar a la derecha y 35 de mirar a la izquierda. El algoritmo logró clasificar correctamente 25 de las veces que se miró hacia la derecha, mientras que en 5 ocasiones no logró detectar la dirección correcta de la mirada. La sensibilidad del algoritmo, que representa la proporción de veces que el algoritmo detectó correctamente la mirada hacia la derecha en comparación con el total real de miradas

hacia la derecha, fue calculada utilizando la Ecuación 3.16.

$$\text{Sensibilidad} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}} \tag{3.16}$$

El resultado obtenido fue aproximadamente del 83,33 %, indicando una tasa de detección exitosa de miradas hacia la derecha por parte del algoritmo.

$$\text{Sensibilidad(Mirar a la derecha} = \frac{25}{25+5} = \frac{25}{30} \approx 0{,}8333 \tag{3.17}$$

Mientras que la precisión de miradas hacia la izquierda fue:

$$\text{Sensibilidad (Mirar a la Izquierda)} = \frac{28}{28+7} = \frac{28}{35} \approx 0{,}8 \tag{3.18}$$

Se evaluó la especificidad para medir la capacidad del modelo en identificar correctamente los casos negativos con la Ecuación 3.19

$$\text{Especificidad} = \frac{\text{Verdaderos Negativos}}{\text{Verdaderos Negativos} + \text{Falsos Positivos}} \tag{3.19}$$

Sustituyendo los valores para mirar a la derecha:

$$\text{Especificidad} = \frac{25}{25+10} = \frac{25}{35} \approx 0{,}7143 \tag{3.20}$$

3.5. Ejecución MQTT

3.5.1. Evaluación

Se configuró el cliente MQTT para conectarse a un broker específico (en este caso, "broker.hivemq.com") y se definió un tema de comunicación. El programa simula la publicación de mensajes en un bucle y muestra cómo recibir mensajes utilizando funciones de callback. La medida del tiempo de latencia, se obtiene restando el instante de inicio del tiempo actual :

$$T_{\text{latencia}} = t_{\text{actual}} - t_{\text{inicio}} \tag{3.21}$$

La tasa de entrega de mensajes, se calcula dividiendo el número total de mensajes recibidos por el número total de mensajes esperados:

$$R_{\text{entrega}} = \frac{M_{\text{recibidos}}}{M_{\text{esperados}}} \tag{3.22}$$

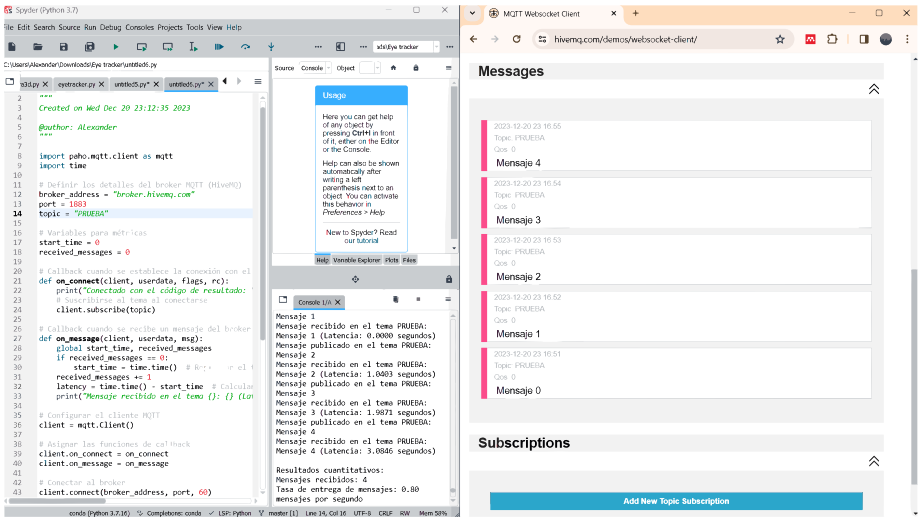


Figura 3.17: Evaluación MQTT

La Figura 3.17 ilustra el proceso de comunicación MQTT, en el cual se enviaron mensajes de manera aleatoria desde Python a un broker de HiveMQ. A través del uso de las bibliotecas de tiempo de Python, se registraron los valores de inicio y tiempo actual para calcular la latencia. Se aplicó la ecuación 3.21 y se verificó la coincidencia de los resultados con los obtenidos mediante el programa.

Mensaje 1

$$T_{\text{latencia}} = 1703132794{,}804566s - 1703132794{,}804566s = 0s \tag{3.23}$$

Mensaje 2

$$T_{\text{latencia}} = 1703132795{,}832640s - 1703132797s = 1{,}0403s \tag{3.24}$$

Mensaje 3

$$T_{\text{latencia}} = 1703132796{,}860046s - 1703132799s = 1{,}9871s \tag{3.25}$$

Mensaje 4

$$T_{\text{latencia}} = 1703132797{,}788105s - 1703132801s = 3{,}0846s \tag{3.26}$$

Tasa de entrega de mensajes

$$R_{\text{entrega}} = \frac{4}{5} = 0{,}8s \tag{3.27}$$

Posteriormente se realizó un sistema de monitoreo diseñado para máquinas, que integra diversos sensores, incluyendo mediciones de temperatura ambiente, vibración y temperatura al contacto. La infraestructura central del sistema utiliza una ESP32 como punto primordial para la comunicación eficiente mediante protocolo MQTT, permitiendo la transferencia fluida de datos. Para la observación inmediata de datos, se ha implementado un dispositivo de adquisición de datos, proporcionando una interfaz eficaz a través del monitor serie. Simultáneamente, se ha establecido un canal remoto de monitoreo utilizando websocket, permitiendo el seguimiento continuo y a distancia de los datos generados por los sensores.

La consistencia entre los valores mostrados en el monitor serie y los transmitidos a través del websocket valida la precisión del sistema en su conjunto. Como se visualiza en la Figura 3.18, en donde el primer valor es del sensor de temperatura, el segundo de humedad, el tercero de temperatura al contacto y el cuarto valor del sensor de vibración. Esta implementación ofrece una solución robusta para la adquisición y monitoreo en tiempo real, facilitando la toma de decisiones informadas y la optimización del rendimiento de las máquinas.

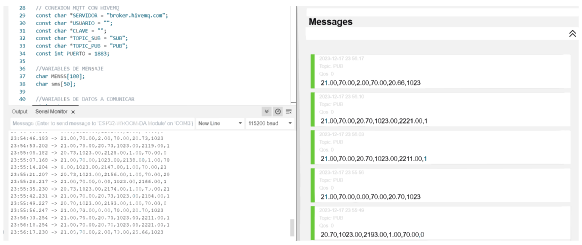


Figura 3.18: Datos monitor serie y websocket

3.5.2. Interfaz de usuario

La interfaz de usuario es una parte esencial de la solución. Utilizando la biblioteca Qt5, se ha diseñado una interfaz intuitiva que proporciona información clara y accesible sobre la detección y seguimiento visual, presenta la imagen del equipo a detectar, señala posibles fallas y datos de los sensores de acuerdo al equipo seleccionado. En la Figura 3.19 muestra la interfaz gráfica realizada con herramientas de Python.

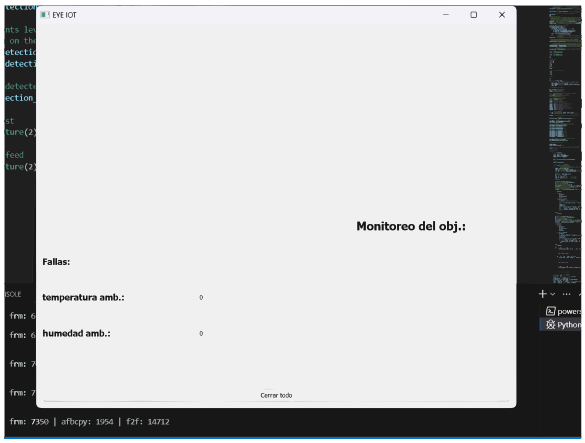


Figura 3.19: Interfaz gráfica

3.6. Pruebas y optimización

La evaluación de la precisión del modelo entrenado se llevó a cabo mediante la implementación de una matriz de confusión. Esta herramienta proporciona una visión detallada de cómo el modelo clasifica las instancias en un conjunto de datos. Se utilizó diversas ecuaciones para calcular parámetros clave que permitieron evaluar el rendimiento del modelo de manera más exhaustiva.

Precisión

$$\text{Precisión} = \frac{VP}{VP + FP} \tag{3.28}$$

Sensibilidad

$$\text{Sensibilidad} = \frac{VP}{VP + FN} \tag{3.29}$$

Especificidad

$$\text{Especificidad} = \frac{VN}{VN + FP} \tag{3.30}$$

Tasa de falsos positivos

$$\text{FPR} = \frac{FP}{FP + VN} \tag{3.31}$$

Donde:

- (VP) = Verdaderos Positivos
- (VN) = Verdaderos Negativos
- (FP) = Falsos Positivos

- (FN) = Falsos Negativos

3.6.1. Matriz de confusión

Para verificar la eficacia del modelo entrenado, es fundamental emplear el 20 % del conjunto total de datos, lo que se traduce en 780 imágenes. Esta distribución se realiza de manera equitativa, asignando 260 imágenes a cada una de las tres clases: Fresadora Roland CNC, Torno Romi GL240, Impresora 3D

El resultado crucial para el tercer objetivo específico se refleja en la matriz de confusión, como se ilustra en la Figura 3.20 Cada columna representa los valores predichos por el modelo con un umbral de confianza del 100 %, mientras que cada fila representa los valores reales, los cuales han sido etiquetados manualmente con ayuda de software.

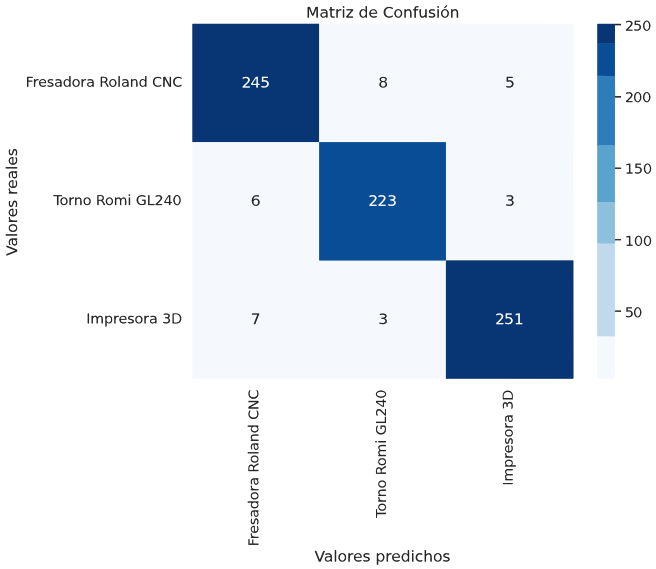


Figura 3.20: Validación del modelo mediante Matriz de confusión

Tabla 3.5: Análisis diagonal principal

Variable	Validación	Afirmación	Erróneas
Fresadora Roland CNC	260	245	15
Torno Romi GL240	260	223	37
Impresora 3D	260	251	9
Total	780	719	61

En la validación del modelo para diversas categorías de equipos, como se observa en la tabla 3.5. La Fresadora Roland CNC registra un 94,23 % de aciertos y un 5,77 % de errores, mientras que el Torno Romi GL240 exhibe un 85,77 % de afirmaciones correctas y un 14,23 % de imprecisiones. Por su parte, la Impresora 3D destaca con un 96,54 % de aciertos y tan solo un 3,46 % de desaciertos. En el conjunto total de 780 afirmaciones, el modelo alcanza una precisión global del 92,18 %, con un 7,82 % de afirmaciones incorrectas, subrayando así su eficacia en la clasificación de equipos.

Tabla 3.6: Evaluación de cada clase

Clase	Verdaderos Positivos	Verdaderos Negativos	Falsos Positivos	Falsos Negativos
Fresadora Roland CNC	245	503	13	15
Torno Romi GL240	223	508	11	37
Impresora 3D	251	495	14	9

Los datos presentados en la tabla de evaluación 3.6 proporcionan información sobre el rendimiento del modelo en términos de verdaderos positivos (VP), verdaderos negativos (VN), falsos positivos (FP) y falsos negativos (FN) para cada una de las clases evaluadas

- La clase 'Impresora 3D' revela que el modelo logró identificar correctamente 251 instancias, excluyendo con precisión 495 casos negativos.
- Sin embargo, se observaron 14 casos de falsos positivos y 9 de falsos negativos.

Tabla 3.7: Métricas de Clasificación por Clase

Clase	Precisión	Especificidad	F1-Score	Exactitud	Sensibilidad
Fresadora Roland CNC	0,91	0,98	0,97	0,94	0,92
Torno Romi GL240	0,95	0,97	0,97	0,95	0,87
Impresora 3D	0,97	0,98	0,96	0,97	0,94

La tabla 3.7 presenta la clasificación para tres categorías específicas: Fresadora Roland CNC, Torno Romi GL240 e Impresora 3D.

- La precisión indica la proporción de instancias correctamente clasificadas para Fresadora Roland CNC (91 %), Torno Romi GL240 (95 %) e Impresora 3D (97 %).
- La especificidad, que mide la capacidad del modelo para identificar correctamente instancias negativas, es del 98 % para Fresadora Roland CNC, 97 % para Torno Romi GL240 y 98 % para Impresora 3D.

- El F1-Score, que combina precisión y sensibilidad, es del 97 % para Fresadora Roland CNC, 97 % para Torno Romi GL240 y 96 % para Impresora 3D. La exactitud general es del 94 %, 95 % y 97 % respectivamente.
- La sensibilidad, que indica la capacidad de identificar correctamente instancias positivas, alcanza el 92 % para Fresadora Roland CNC, 87 % para Torno Romi GL240 y 94 % para Impresora 3D.

Optimización

Detección de objetos Es posible ajustar el umbral de confianza para mejorar la detección de objetos, permitiendo así una mayor flexibilidad en la precisión de las predicciones. Además, se puede implementar la quantización de modelos como una estrategia eficaz para disminuir la carga computacional, lo que resulta en un rendimiento más eficiente del algoritmo de detección.

Seguimiento ocular

Para optimizar el rendimiento, se sugiere utilizar la paralelización mediante el uso de GPU para acelerar el procesamiento.

MQTT Minimizar el tamaño de los mensajes y gestionar las conexiones de manera eficiente.

Discusión Se empleó la red neuronal convolucional Faster RCNN Inception V2 debido a su notable rendimiento durante la inferencia, registrando un tiempo de 745,82 ms, en comparación con SSD MobileNet V2, que arrojó 1188,27 ms. Además, Faster Inception demostró una precisión de entrenamiento del 87,15 %, superando significativamente la MobileNet con un 62,66 %. En la evaluación del algoritmo de seguimiento ocular, se definió la región de interés y se compararon coordenadas, calculando sensibilidad y especificidad. La implementación de software permitió almacenar las ubicaciones de la mirada para una posterior comparación. Para la componente MQTT, se evaluó la latencia y la tasa de entrega de mensajes. Se destaca la red neuronal convolucional Faster Inception V2 en términos de eficiencia y precisión, así como la efectividad del algoritmo de seguimiento ocular y el desempeño del protocolo MQTT en la aplicación desarrollada.

Limitaciones Las decisiones sobre la selección de modelos se vieron influidas por diversas limitaciones y consideraciones. En primer lugar, la elección de no utilizar las redes neuronales SSD MobileNet V3 y RCNN Faster Inception V3 se atribuye, en parte, a restricciones en los

recursos computacionales disponibles. Estas arquitecturas, conocidas por su complejidad y demanda computacional, podrían haber excedido los límites de capacidad del sistema. Además, al evaluar el rendimiento en tiempo de inferencia, se observó que estas arquitecturas no ofrecían la eficiencia deseada en comparación con la selección de Faster Inception V2, que demostró tiempos de inferencia más rápidos. La complejidad adicional de los modelos no utilizados, junto con la falta de modelos pre-entrenados y ajustados específicamente para la aplicación, también contribuyó a la decisión. En última instancia, se priorizó la simplicidad y el rendimiento eficiente en lugar de la complejidad de las alternativas disponibles, considerando la viabilidad y los requisitos específicos.

3.7. Resultados

La convergencia exitosa de la detección de objetos, el seguimiento ocular y la implementación de comunicación MQTT ha culminado en un sistema integral para el mantenimiento de equipos de impresión 3D. Este enfoque une estos tres componentes esenciales, ha sido materializado en una interfaz gráfica intuitiva y funcional. La ventana principal de la interfaz presenta de manera destacada el equipo detectado, seleccionado de manera precisa mediante el eye tracker. Este componente, a su vez, ha desempeñado un papel crucial al dividir la información en tres secciones contextualizadas según la dirección visual del usuario. En una ventana adicional del sistema, la capacidad del eye tracker para seguir la dirección de la mirada se ha amplificado. La interactividad toma forma al mostrar recortes de pantalla específicos cuando el usuario dirige la mirada hacia áreas particulares. Esta función no solo mejora la precisión en la identificación de equipos, sino que también agrega una dimensión de inmediatez y contextualización visual a la experiencia del usuario durante el proceso de mantenimiento. La integración de MQTT y el uso de un websocket han enriquecido aún más el sistema al proporcionar la visualización en tiempo real de los datos de los sensores. Esta característica esencial no solo valida la efectividad del mantenimiento, sino que también contribuye significativamente a la implementación de estrategias proactivas basadas en datos. El sistema, en su totalidad, representa una unión sinérgica de tecnologías avanzadas que elevan la eficiencia y la precisión en el mantenimiento de equipos de impresión 3D. Este logro no solo optimiza la operatividad del sistema, sino que también sienta las bases para futuras innovaciones en el campo del mantenimiento industrial y la integración de tecnologías emergentes. El algoritmo opera mediante el seguimiento ocular para permitir la observación exclusiva de la información asociada a un equipo seleccionado. Como se muestra en la Figura 3.21 las líneas verticales delinean las distintas zonas visuales, abarcando la izquierda, el centro y la derecha. En este contexto, al enfocarse en el centro, se

facilita la presentación específica de datos relacionados con un equipo determinado, aquel que se encuentra bajo la atención visual del usuario.

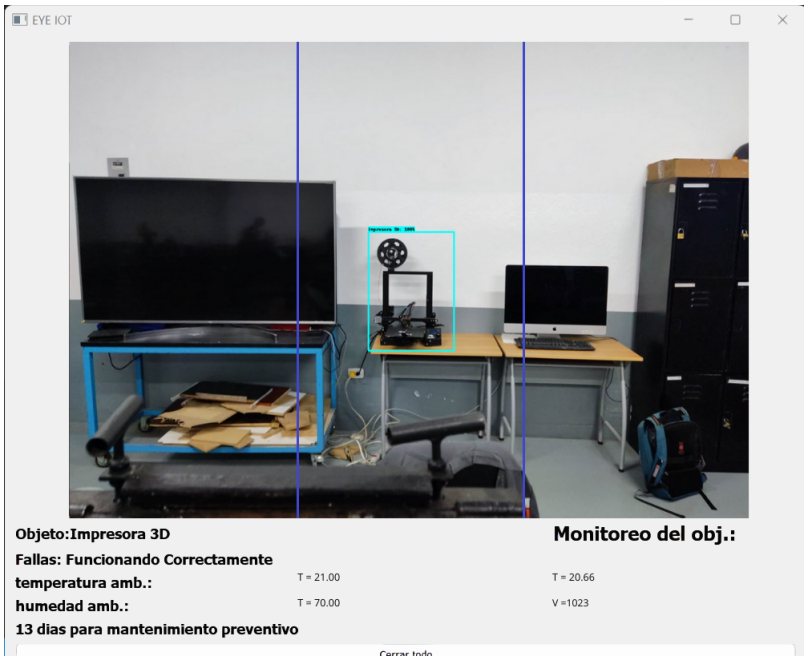


Figura 3.21: Funcionamiento final del algoritmo

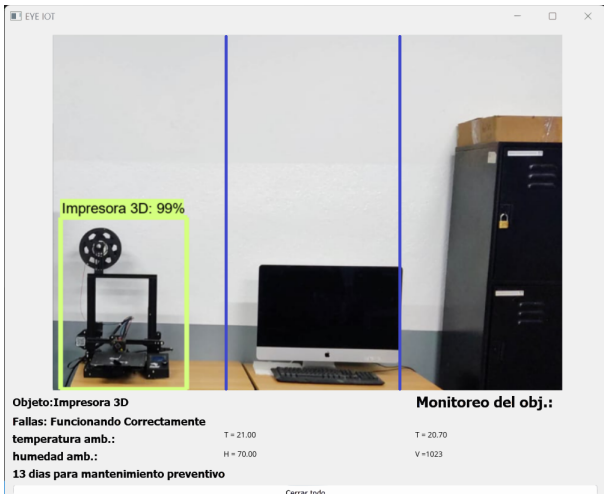


Figura 3.22: Funcionamiento del algoritmo mirando a la izquierda

Los resultados obtenidos de los experimentos demostraron la efectividad del algoritmo propuesto para el mantenimiento preventivo de equipos. Se observó una detección precisa en diferentes condiciones de iluminación. El seguimiento visual mostró una integración fluida con la

detección de objetos, proporcionando información relevante sobre el equipo observado. La retroalimentación en tiempo real sobre el estado de los equipos permitió a los usuarios identificar información crítica y tomar decisiones informadas sobre mantenimiento preventivo. En conclusión, el diseño de experimentos demostró que el algoritmo desarrollado es capaz de ofrecer una solución efectiva y precisa para el monitoreo y mantenimiento preventivo de equipos a través de la detección de objetos y el seguimiento visual.

Capítulo IV

Conclusiones, Recomendaciones y Trabajo a futuro

4.1. Conclusiones

- La integración de tecnologías avanzadas, como la detección de objetos, el seguimiento ocular y la comunicación IoT, ha resultado en un sistema integral para el mantenimiento de máquinas, la colaboración efectiva de estos componentes han demostrado ser esencial para mejorar la eficiencia y la precisión en el monitoreo y cuidado de los dispositivos.

- La combinación de detección de objetos y seguimiento ocular ha llevado a una optimización significativa del proceso de mantenimiento preventivo, la identificación precisa de equipos y la adaptabilidad en la presentación de información han permitido tomar decisiones informadas y ejecutar acciones de mantenimiento de manera más eficiente.

- La inclusión del seguimiento ocular ha actuado como una capa de validación adicional, contribuyendo a la reducción de posibles errores en la identificación de equipos. Esto ha mejorado la precisión global del sistema y ha fortalecido la confianza en los resultados obtenidos.

- La ejecución exitosa de los protocolos IoT, como la comunicación mediante MQTT y la adquisición de datos en tiempo real a través de sensores, ha demostrado ser un pilar fundamental en el sistema de mantenimiento preventivo de equipos de impresión 3D. La capacidad de transmitir datos de sensores relevantes, ha permitido no solo la monitorización constante, sino también la toma de decisiones basadas en información actualizada. Esto destaca el potencial transformador del IoT en la mejora de la eficiencia operativa,

la predicción de fallos y la posibilidad de establecer estrategias de mantenimiento pro activo.

4.2. Recomendaciones

- Se recomienda continuar optimizando el modelo de detección de objetos para mejorar aún más la precisión y la velocidad de detección. La exploración de arquitecturas más avanzadas y técnicas de optimización podría llevar a mejoras significativas en el rendimiento del modelo.
- Aunque la integración entre la detección de objetos y el seguimiento visual es exitosa, se sugiere explorar formas de mejorar la sincronización y la respuesta en tiempo real. Esto podría aumentar la eficacia de la solución y brindar una experiencia aún más fluida a los usuarios.
- La implementación progresiva de técnicas de aprendizaje continuo o adaptativo podría ser una dirección valiosa para investigaciones futuras. Permitir al algoritmo ajustarse y mejorar su rendimiento a medida que se enfrenta a nuevos datos o condiciones no previstas podría potencial la capacidad de adaptación del sistema a entornos cambiantes.
- Se sugiere realizar un análisis exhaustivo del impacto del sistema en las estrategias de mantenimiento preventivo. Evaluar de manera detallada cómo la implementación del algoritmo puede contribuir efectivamente a la prevención de fallos y prolongar la vida útil de los equipos proporcionará una comprensión más completa de su eficacia y aplicabilidad en escenarios específicos.

4.3. Trabajo a futuro

- Una extensión natural del algoritmo sería la detección de anomalías en los equipos. Explorar formas de identificar comportamientos inusuales o componentes defectuosos podría permitir un mantenimiento aún más proactivo y preventivo.
- Investigar la posibilidad de implementar interacciones más avanzadas con los equipos, como la activación de alertas o la realización de acciones específicas en función de las detecciones y el seguimiento visual.

- La integración de tecnologías de realidad aumentada podría transformar la experiencia de mantenimiento al proporcionar información contextual en tiempo real directamente en el campo de visión del técnico, esto podría suponer datos relevantes, como instrucciones de mantenimiento, diagnósticos y visualizaciones 3D, mejorando la comprensión y eficiencia durante las tareas de mantenimiento. Esta expansión permitiría una interacción más inmersiva y colaborativa entre el técnico y el sistema.
- El algoritmo desarrollado ha demostrado su valía como una solución sólida para el mantenimiento preventivo de equipos de laboratorio. Sin embargo, existen áreas para la mejora continua y la expansión de las capacidades, lo que abre oportunidades emocionantes para investigaciones futuras y mejoras en la aplicación práctica de la solución.

Bibliografía

[1] Equipo de redacción de Drew, "Principales desafíos a superar por un gerente de mantenimiento," 6 2022.

[2] fegemu solutions, "La automatización del mantenimiento predictivo. BLOG. Fegemu Solutions," 4 2021.

[3] L. C. Lanzarini, W. Hasperué, A. Villa Monte, P. Jimbo Santana, G. Reyes Zambrano, J. P. Corvi, A. Fernández Bariviera, and J. Olivas Varela, "Plataforma detección de objetos en tiempo real," *XXI Workshop de Investigadores en Ciencias de la Computación (WICC 2019)*, 2018.

[4] B. G. Castelo Pichucho and B. G. Mosquera López, *Diseño e implementación de un sistema para la detección y seguimiento de una persona a través de un cuadricóptero de tamaño reducido empleando visión artificial*. PhD thesis, Escuela Politécnica Nacional, Quito, 2 2022.

[5] T. Ramirez-Guerrero and M. Toro, "Detection of infractions and license plates in motorcycles by means of artificial vision, applied to Intelligent Transport Systems Ballast useful life optimization and replacement study for Metro de Medellín railway View project Virtual simulator for training in the overhead cable command and control system View project," *Revista Ibérica de sistemas e tecnologías de informaçäo*, 2020.

[6] N. Ale Ale, J. Fabián, N. Ale Ale, and J. Fabián, "Detección del estado fisiológico de los ojos en Conductores mediante técnicas de visión artificial," *Ingeniare. Revista chilena de ingeniería*, vol. 27, pp. 564–572, 12 2019.

[7] J. M. José Martínez, ""Sistema de Visión Artificial para la Detección y Corrección de Posturas en ejercicios realizados por fisicoculturistas

," tech. rep., Universidad Autónoma del Estado de México, Atlacomulco, 10 2018.

[8] P. S. Cuertán Ponce, *Creación de una aplicación multimedia para mantenimiento técnico basado en el modelamiento digital de un sistema de suspensión con realidad aumentada*. PhD thesis, Universidad Técnica del Norte, Ibarra, 5 2021.

[9] A. González Manzanares, *Detector de baches con deep learning*. PhD thesis, Universidad Pompeu Fabra, Barcelona, 2019.

[10] E. de Pädi, J. Gudiño-Lau, C. Llamas-Woodward, J. Alcalá-Rodríguez, S. Charre-Ibarra, A. Jarillo-Silva, and M. Durán-Fonseca, "Manipulación de un brazo robot mediante el seguimiento de los ojos," *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, vol. 10, pp. 114–120, 11 2022.

[11] M. Wilmer Fabian Albarracín Guarochico, *SISTEMA DE REALIDAD AUMENTADA PARA MANTENIMIENTO CORRECTIVO INDUSTRIAL.* PhD thesis, Universidad Israel, Quito, 9 2022.

[12] D. L. López Olson, *REALIDAD AUMENTADA PARA MANTENIMIENTO PREVENTIVO DEL SISTEMA DE CONTROL DEL GRUPO DE GENERACIÓN DE LA CENTRAL SAYMIRIN.* PhD thesis, Universidad Israel, Quito, 9 2022.

[13] R. Guerrero Pérez, *Mantenimiento preventivo de sistemas domóticos e inmóticos*. IC Editorial, 2015.

[14] A. Virgüez Gómez, *Diseño e implementación de un programa de mantenimiento preventivo en la sección de mezclado de planta de caucho de la EMPRESA ETERNA S.A. a partir de técnicas de mantenimiento predictivo*. PhD thesis, Universidad Distrital Francisco José de Caldas, Bogotá, 9 2019.

[15] E. Iona Jiva, *Desarrollo de la teleoperación de robots industriales y colaborativos mediante técnicas avanzadas de visión artificial*. PhD thesis, Universidad Politécnica de Valencia, Valencia, 10 2019.

[16] J. C. Miranda, *CLASIFICACIÓN AUTOMÁTICA DE NARANJAS POR TAMAÑO Y POR DEFECTOS UTILIZANDO TÉCNICAS DE VISIÓN POR COMPUTADORA*. PhD thesis, Universidad Nacional de Asunción, San Lorenzo, 2018.

[17] S. Badillo, B. Banfai, F. Birzele, I. I. Davydov, L. Hutchinson, T. Kam-Thong, J. Siebourg-Polster, B. Steiert, and J. D. Zhang, "An Introduction to Machine Learning," *Clinical Pharmacology and Therapeutics*, vol. 107, pp. 871–885, 4 2020.

[18] P. Agustín Granell, *Redes Neuronales Recurrentes: Una aplicación para los mercados bursátiles*. PhD thesis, Universidad de Barcelona, Barcelona, 7 2018.

[19] T. Domínguez Mínguez, *Visión artificial: Aplicaciones prácticas con OpenCV*. MARCOMBO, S.L., 1 ed., 2021.

[20] L. Quiñones Huatangari, L. Ochoa Toledo, O. Gamarra Torres, J. Bazán Correa, J. Delgado Soto, and N. Kemper Valverde, "Red neuronal artificial para estimar un índice de calidad de agua," *Enfoque UTE*, vol. 11, pp. 109–120, 4 2020.

[21] J. Sánchez-Alor Expósito, *Evaluación de algoritmos de detección de objetos basados en deep learning para detección de incidencias en carreteras*. PhD thesis, Escuela Técnica Superior de Ingenieros de Telecomunicación, Villadolid, 9 2020.

[22] E. Rodríguez Hernández, *Clonación de comportamiento basado en aprendizaje profundo para cruce de pasajes estrechos con VANT*. PhD thesis, Instituto Politécnico Nacional, México, 8 2019.

[23] Artola Moreno, *Clasificación de imágenes usando redes neuronales convolucionales en Python*. PhD thesis, Universidad de Sevilla, Sevilla, 2019.

[24] H. Fujiyoshi, T. Hirakawa, and T. Yamashita, "Deep learning-based image recognition for autonomous driving," *IATSS Research*, vol. 43, pp. 244–252, 12 2019.

[25] R. Gonzales Martínez, J. Machacuay, P. Rotta, and C. Chinguel, "Hyperparameters Tuning of Faster R-CNN Deep Learning Transfer for Persistent Object Detection in Radar Images," *IEEE LAT AM*, vol. 20, pp. 677–685, 4 2021.

[26] C. Lopez-Castaño, C. Ferrin-Bolaños, and L. Castillo-Ossa, "Computer vision and the internet of things ecosystem in the connected home," *Advances in Intelligent Systems and Computing*, vol. 800, pp. 213–220, 2019.

[27] D. A. Jarrín Rodríguez, *ESTUDIO COMPARATIVO ENTRE MODELOS DE APRENDIZAJE PROFUNDO, DESARROLLADOS A PARTIR DE REDES NEURONALES RECURRENTES Y REDES NEURONALES CONVOLUCIONALES, PARA LA DETECCIÓN DE INTRUSOS DE RED*. PhD thesis, Universidad Internacional SEK, Quito, 9 2019.

[28] M. Cabezón Manchado, *Implementación de redes neuronales recurrentes en Python*. PhD thesis, Universidad Complutense de Madrid, Madrid, 9 2018.

[29] D. Romero Ureña, *Detección y geolocalización de objetos mediante visión artificial desde vehículos en movimiento*. PhD thesis, Universidad Carlos III de Madrid, Madrid, 2020.

[30] 330 ohms, "¿Cómo detectar objetos con YOLO?," 11 2020.

[31] J. Benito Picazo, K. Thurnhofer Hemsi, M. A. Molina-Cabello, and E. Domínguez, "Comparación de marcos de trabajo de Aprendizaje Profundo para la detección de objetos," *RiUMA*, 11 2018.

[32] M. Massiris, C. Delrieux, and J. Álvaro Fernández, "DETECCIÓN DE EQUIPOS DE PROTECCIÓN PERSONAL MEDIANTE RED NEURONAL CONVOLUCIONAL YOLO," *Actas de las XXXIX Jornadas de Automática*, 9 2018.

[33] D. L. Moreira Ramos, *Aplicación de un modelo de reconocimiento de objetos utilizando Yolo (you only look once)*. PhD thesis, Universidad Estatal Península de Santa Elena, La Libertad, 4 2021.

[34] A. Robles Lagunas, J. S. Valdez Martínez, J. Villanueva Tavira, A. M. Beltrán Escobar, Contreras Calderón Enrique, I. Alcalá Barojas, L. J. López Vega, and A. Leyva Mancilla, "Aplicación de redes neuronales en la detección de seguridad para la prevención de contagios," *ResearchGate*, 2021.

[35] F. Arévalo Mesa, "El análisis de eficiencia de la red neuronal convolucional (cnn) y el sistema de aprendizaje tensorflow," *Investigación y Ciencia Aplicada a la Ingeniería*, vol. 4, pp. 14–27, 9 2021.

[36] R. Varadharajan Iyer, P. Shashikant Ringe, R. Varadharajan Iyer, and K. Prabhulal Bhensdadiya, "Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection," *Article in International Journal of Research in Engineering and Technology*, 2021.

[37] M. Hollemans, "MobileNet version 2," 4 2018.

[38] A. Torres Alonso, *Detección de frutas en árboles* . PhD thesis, Universidad Complutense Madrid, Madrid, 3 2020.

[39] O. Soto Orozco, "Análisis del desempeño de redes neuronales profundas para segmentación semántica en hardware limitado ," *Revista electrónica de Computación, Informática, Biomédica y Electrónica*, vol. 8, pp. 1–21, 11 2019.

[40] B. T. Carter and S. G. Luke, "Best practices in eye tracking research," *International Journal of Psychophysiology*, vol. 155, pp. 49–62, 9 2020.

[41] W. Guarquila Uzcha and H. Solis Guncay, *Psicodiagnóstico obtenido por "Eye tracking", revisión sistemática de la literatura* . PhD thesis, Universidad del Azuay, Cuenca, 2023.

[42] B. Porras-Garcia, M. Ferrer-Garcia, A. Ghita, M. Moreno, L. López-Jiménez, A. Vallvé-Romeu, E. Serrano-Troncoso, and J. Gutiérrez-Maldonado, "The influence of gender and body dissatisfaction on body-related attentional bias: An eye-tracking and virtual reality study," *International Journal of Eating Disorders*, vol. 52, pp. 1181–1190, 10 2019.

[43] M. Ramírez, "Eye tracking, la nueva tecnología de la Industria 4.0," *Técnica Industrial*, vol. 325, pp. 4–5, 2020.

[44] T. S. Blas Ayllón and J. A. Pariona Medina, "La aplicación del eye tracking como herramienta de neuromarketing para incrementar el consumo," *Universidad Tecnológica del Perú*, 2019.

[45] M. J. Coral Quinto, *Diseño e implementación de base de datos mediante el uso de web services con integración de unity 3D para apoyo de aplicaciones lúdicas en la materia de fundamentos de programación* . PhD thesis, Universidad de Guayaquil, Guayaquil, 2018.

[46] P. Beynon-Davies, *Sistemas de bases de datos*. 2018.

[47] E. Chicano Tejada and J. J. Trujillo Cebrian, *Utilización de las bases de datos relacionales en el sistema de gestión y almacenamiento de datos. ADGG0308*. IC Editorial, 2 ed., 2022.

[48] Amazon, "¿Qué es el MQTT? - Explicación del protocolo MQTT - AWS."

[49] ElectroStore, "MÓDULO ESP32 ESP-32 WIFI BLUETOOTH (38PINES) – Grupo Electrostore."

[50] Álvarez Carulla Albert, *Comunicación de un módulo ESP32 }con Ubidots mediante MQTT. PhD thesis, Universidad de Barcelona, 12 2021.*

[51] ElectronicsHub, "Arduino Mega Pinout | Arduino Mega 2560 Layout, Specifications."

[52] Mendoza Galindo Jorge, Implementación de un sistema de adquisición de datos }con la interfaz de Arduino Mega para el estudio de }fenómenos físicos. PhD thesis, 2019.

[53] SUCONEL, “Sensor De Vibración SW-18010p Lm393 - Suconel S.A.”

[54] ardobot, “Modulo Sensor de vibracion SW-18010P.”

[55] Electro Store, “SENSOR DE TEMPERATURA PT100 1METRO 2 HILOS – Grupo Electrostore.”

[56] Omega, “Introducción y tipos de sensores pt100.”

[57] Naylamp mechatronics, “Módulo transmisor RTD PT100 MAX31865.”

[58] Unit Electronics, “MAX31865 Detector de temperatura - UNIT Electronics.”

[59] HiveMQ, “MQTT Websocket Client.”

Anexos

- https://utneduec-my.sharepoint.com/:f:/g/personal/adapuangom_utn_edu_ec/EsNXAl3H1ldBkRQHM5e=9yapAI
- https://github.com/AlexanderApuango/Detector-de-objetos.git
- https://github.com/AlexanderApuango/Eye-tracker.git
- https://github.com/AlexanderApuango/MQTT.git