UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TÍTULO

Sistema de detección de Equipos de Protección Personal (EPP) mediante visión artificial y redes neuronales para entornos industriales

AUTOR

Daquilema Aimacaña, Fernando Javier

TRABAJO DE TITULACIÓN

Previo a la obtención del grado académico en
MAGÍSTER EN ELECTRÓNICA Y AUTOMATIZACIÓN

TUTOR

Morales Escobar, Luis Alberto

Santa Elena, Ecuador

Año 2026

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TRIBUNAL DE SUSTENTACIÓN

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
ALICIA GERMANIA ANDRADE VERA

Ing. Alicia Andrade Vera, Mgtr.
COORDINADORA DEL PROGRAMA

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
LUIS ALBERTO MORALES ESCOBAR

Ing. Luis Morales Escobar, Ph.D.
TUTOR

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
LUIS ENRIQUE CHUQUIMARCA JIMENEZ

Ing. Luis Chuquimarca Jimenez, Ph.D.
DOCENTE ESPECIALISTA

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
JUNIOR RAFAEL FIGUEROA OLMEDO

Ing. Junior Figueroa Olmedo, Mgtr.
DOCENTE ESPECIALISTA

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
MARIA MARGARITA RIVERA GONZALEZ

Abg. María Rivera González, Mgtr.
SECRETARIA GENERAL
UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

CERTIFICACIÓN

Certifico que luego de haber dirigido científica y técnicamente el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos, razón por el cual apruebo en todas sus partes el presente trabajo de titulación que fue realizado en su totalidad por Daquilema Aimacaña Fernando Javier, como requerimiento para la obtención del título de Magíster en Electrónica y Automatización.

TUTOR

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
LUIS ALBERTO
MORALES ESCOBAR

Ing. Luis Morales Escobar, Ph. D.

Santa Elena, 31 de marzo de 2026

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Daquilema Aimacaña Fernando Javier**

DECLARO QUE:

El trabajo de Titulación, Sistema de detección de Equipos de Protección Personal (EPP) mediante visión artificial y redes neuronales para entornos industriales previo a la obtención del título en Magíster en Electrónica y Automatización, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Santa Elena, 31 de marzo de 2026

EL AUTOR

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
FERNANDO JAVIER
DAQUILEMA AIMACANA

Fernando Javier Daquilema Aimacaña

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

CERTIFICACIÓN DE ANTIPLAGIO

Certifico que después de revisar el documento final del trabajo de titulación denominado Sistema de detección de Equipos de Protección Personal (EPP) mediante visión artificial y redes neuronales para entornos industriales presentado por el estudiante, Daquilema Aimacaña Fernando Javier fue enviado al Sistema Antiplagio COMPILATIO, presentando un porcentaje de similitud correspondiente al 9%, por lo que se aprueba el trabajo para que continúe con el proceso de titulación.


Informe de análisis
Compilatio Magister+ | UPSE-ECU

Daquilema
ID : db7e145ff37f21b0723724c2b3920ae550cbc77e

9%
Textos sospechosos

Nombre del fichero : Daquilema.txt
Tamaño del archivo original : 5,5 MB
Número de palabras : 19.766
Número de caracteres : 159978

Depositante : LUIS ALBERTO MORALES ESCOBAR
Fecha de depósito : 31 de marzo de 2026
Tipo de carga : interface
fecha de fin de análisis : 31 de marzo de 2026

TUTOR

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
LUIS ALBERTO MORALES ESCOBAR

Ing. Luis Alberto Morales Escobar, Ph. D.

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

AUTORIZACIÓN

Yo, **Daquilema Aimacaña Fernando Javier**

Autorizo a la Universidad Estatal Península de Santa Elena, para que haga de este trabajo de titulación o parte de él, un documento disponible para su lectura consulta y procesos de investigación, según las normas de la Institución.

Cedo los derechos en línea patrimoniales de este proyecto de titulación con componentes de investigación aplicada y/o de desarrollo con fines de difusión pública, además apruebo la reproducción de este proyecto de titulación con componentes de investigación aplicada y/o de desarrollo dentro de las regulaciones de la Universidad, siempre y cuando esta reproducción no suponga una ganancia económica y se realice respetando mis derechos de autor.

Santa Elena, 31 de marzo de 2026

EL AUTOR

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
FERNANDO JAVIER
DAQUILEMA AIMACANA

Daquilema Aimacaña Fernando Javier

AGRADECIMIENTO

A Dios por brindarme salud y bendecirme en todas las actividades que se ha realizado para llegar a obtener tan anhelado título, por darme fortaleza, responsabilidad, sabiduría, A mis padres por brindarme el apoyo en todo momento de manera incondicional durante el transcurso de mis estudios, por dedicarme tiempo y esfuerzo para ser persona de bien y darme excelentes consejos en nuestro camino diario, a mis familiares por darme ejemplo, dedicación e instruirme para seguir adelante en mi vida profesional. A la Universidad Estatal Península de Santa Elena, a sus autoridades y docentes por abrir sus puertas y darme la oportunidad de seguirme formando profesionalmente para triunfar en la vida. A mi tutor de titulación Ph.D. Luis Morales quien con su dedicación y ayuda se logró concretar mi trabajo de titulación.

Fernando Javier, Daquilema Aimacaña

DEDICATORIA

A Dios. Por haberme guiado por el buen camino por haberme dado salud para lograr mis objetivos y no desmayar en los problemas que se presentaban y poder llagar a cumplir mi sueño, a mis padres: Carmen Aimacaña y Agustín Daquilema por ser el pilar fundamental motivándome y dándome apoyo en todo momento, por sus concejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada por su amor. A mis hermanos, por ser el ejemplo de los cuales he aprendido muchos aciertos, a mis sobrinos que siempre han estado a mi lado brindado su cariño y apoyo en el transcurso de este camino.

Fernando Javier, Daquilema Aimacaña

ÍNDICE GENERAL

TÍTULO I

TRIBUNAL DE SUSTENTACIÓN II

CERTIFICACIÓN III

DECLARACIÓN DE RESPONSABILIDAD IV

DECLARO QUE: IV

CERTIFICACIÓN DE ANTIPLAGIO V

AUTORIZACIÓN VI

AGRADECIMIENTO VII

DEDICATORIA VIII

ÍNDICE GENERAL IX

ÍNDICE DE TABLAS XIII

ÍNDICE DE FIGURAS XIV

RESUMEN XIX

ABSTRACT XX

INTRODUCCIÓN 1

CAPÍTULO 1. MARCO TEÓRICO REFERENCIAL 3

1.1. Revisión de literatura 3

1.2. Desarrollo teórico y conceptual 5

1.2.1 Tecnología de Hardware 5

1.2.2 Visión Artificial 5

1.2.3 Redes Neuronales y Yolov8 6

1.2.4 Machine Learning (ML) 7

1.2.5 Internet de las cosas (IoT) 8

1.2.6 Software 9
1.2.7 Python 9
1.2.8 Node-RED 10
1.2.9 ESP 32 11
1.2.10 RELE 12
CAPÍTULO 2. METODOLOGÍA 14
2.1. Contexto de la investigación. 14
2.2. Diseño y alcance de la investigación 14
2.3. Tipo y métodos de investigación 14
2.4. Población y muestra 15
2.5. Técnicas e instrumentos de recolección de datos 15
2.6. Procesamiento de la evaluación: Validez y confiabilidad de los instrumentos aplicados para el levantamiento de información. 16
2.7. Metodología de Desarrollo 18
2.7.1 Configuración del entorno de desarrollo 18
2.7.2 Instalación del lenguaje de programación Python 18
2.7.3 Instalación de librerías para visión artificial y comunicación 19
2.7.4 Configuración de Node-RED 20
2.7.5 Configuración de bróker MQTT (Mosquito) 22
2.7.6 Fases y etapas del desarrollo y funcionamiento del sistema 23
2.7.7 Fase l: Desarrollo del modelo de visión artificial 25
2.7.7.1 Etapa 1: Preparación del conjunto de datos (Dataset) 25
2.7.7.2 Etapa 2 Entrenamiento y validación del modelo YOLOv8 29
2.7.8 Fase II: Funcionamiento del sistema en tiempo real 33
2.7.8.1 Etapa 3: Selección y configuración de la cámara IP 34
2.7.8.2 Etapa 4: Captura de imágenes en tiempo real mediante Python y OpenCV ...36
2.7.8.3 Etapa 5: Procesamiento de imágenes mediante YOLOv8 37

2.7.8.4 Etapa 6: Evaluación del cumplimiento y uso de EPP 40
2.7.8.5 Etapa 7: Visualización de resultados.....42
2.7.8.6 Etapa 8: Comunicación con Node-RED mediante HTTP 43
2.7.8.7 Etapa 9: Configuración y activación de alertas físicas mediante MQTT.....52
Configuración del Broker mosquito 52
- Suscripción a topics (MQTT IN) 54
- Configuración del cliente MQTT en Node-RED.....55
- Lógica de decisión para control de dispositivo (Fuction ESP32).....56
- Publicación de comandos hacia el ESP32 (MQTT OUT).....56

2.7.8.8 Etapa 10: Implementación del sistema de alertas físicas basado en ESP32....58
- Diseño electrónico del sistema de alertas físicas 58
- Funcionamiento y conexión (ESP32 GPIO, Rele, Luz estroboscópica).....59
- Configuración del entorno de desarrollo (Arduino IDE) 60

Programación del dispositivo ESP32 para activación de alertas físicas.....64
- Verificación de comunicación y activación del relé 66

2.7.8.9 Etapa 11: Envío de notificaciones remotas mediante Telegram.....67
- Obtención del TOKEN del bot 68
- Obtención del Chat ID.....68
- Configuración e integración del bot en Node-RED 69
- Integración del nodo Telegram y Function 2 dentro del flujo de alertas 69
- Configuración del Token en el nodo Telegram 71
- Verificación del envió de notificaciones.....72

CAPÍTULO 3. RESULTADOS Y DISCUSIÓN..... 73
3.1 Resultados del modelo de detección de EPP.....73
3.1.1 Sistema Implementado 73

3.1.2 Flujo de operación del sistema en Node-RED....74
3.1.3 Interfaz de visualización y monitoreo real mediante Node-RED....74
3.1.4 Registro automático de eventos de incumplimiento de EPP....75
3.1.5 Supervisión remota y registro de eventos detectados por el sistema....76
3.1.6 Escenario de validación....76
Discusión....88
CONCLUSIONES....89
RECOMENDACIONES....90
REFERENCIAS....91
ANEXOS....92

ÍNDICE DE TABLAS

Tabla 1 Parámetros de evaluación del sistema.....17

Tabla 2 Métricas para evaluación del sistema 17

Tabla 3 Comandos para verificar instalación correcta de Python.....19

Tabla 4 Comandos para activar librerías 19

Tabla 5 Configuración e instalación de Node-RED. 20

Tabla 6 Configuración e Instalación Bróker Mosquito. 23

Tabla 7 Comando estándar que especifica la tarea de detección. 29

Tabla 8 Métricas del proceso de entrenamiento del modelo YOLOv8. 30

Tabla 9 Lista técnica de configuración Nodo HTTP In.....46

Tabla 10 Componentes y parámetros arquitectura MQTT 54

Tabla 11 Componentes Utilizados. 59

Tabla 12 Matriz de confusión para la detección de casco. 77

Tabla 13 Matriz de confusión para la detección de chaleco. 80

Tabla 14 Matriz de confusión para la detección de guantes. 82

Tabla 15 Comparación del desempeño por clase. 85

Tabla 16 Matriz de confusión global para la detección de EPP. 86

ÍNDICE DE FIGURAS

Figura 1 Camara Dahua.....5

Figura 2 Deteccion de Humanos.....6

Figura 3 Red Neuronal 6

Figura 4 Machine learning y deep learning 8

Figura 5 Imagen IoT.....9

Figura 6 Python con sus librerías.....10

Figura 7 Node-Red.....11

Figura 8 Tarjeta ESP 32 WI-FI.....12

Figura 9 Modulo Rele 5V.....13

Figura 10 Ejecución de Node-RED desde la línea de comandos y URL de acceso al entorno.....22

Figura 11 Fases y etapas del desarrollo del sistema.....24

Figura 12 Labellmg repositorio para crear etiquetas e implementar modelos de visión artificial.....26

Figura 13 Imágenes etiquetadas.....27

Figura 14 Estructura jerárquica del dataset en formato YOLO.....28

Figura 15 Definición de rutas y clases del dataset mediante el archivo data.yaml.....28

Figura 16 Proceso de entrenamiento en tiempo real del modelo YOLOv8 mediante consola.....30

Figura 17 Métricas de entrenamiento y validación del modelo YOLOv8 durante el proceso de aprendizaje.....32

Figura 18 Arquitectura general del sistema de detección de EPP en tiempo real mediante visión artificial. 33

Figura 19 Diagrama de bloques y funcionamiento del sistema en tiempo real. 34

Figura 20 Camara IP Dahua para adquisición de video 35

Figura 21 Configuración de red para la comunicación cámara-PC 35

Figura 22 Configuración de la URL RTSP de la cámara IP y rutas del sistema. 36

Figura 23 Código de inicialización y verificación de la captura de video RTPS mediante OpenCV 36

Figura 24 Captura de imágenes en tiempo real desde la cámara IP.....37

Figura 25 Parte del código que se utiliza para procesamiento.....38

Figura 26 Diagrama de flujo del procesamiento de imágenes con YOLOv8.....38

Figura 27 Resultado del procesamiento y detección de EPP.....39

Figura 28 Detección y código para la evaluar lógica del cumplimiento. 40

Figura 29 Diagrama de flujo de la evaluación del cumplimiento del uso de EPP. 41

Figura 30 Resultado de la evaluación del cumplimiento de EPP 42

Figura 31 Resultado visual del sistema en tiempo real donde se identifica incumplimiento. 43

Figura 32 Diagrama de flujo de la arquitectura de comunicación entre Python y Node-RED mediante HTTP. 44

Figura 33 Configuración para comunicación entre Python y Node-RED. 45

Figura 34 Configuración del Nodo HTTP IN.....45

Figura 35 Configuración nodo JSON para interpretación de datos. 46

Figura 36 Nodo Function se encarga de realizar procesamiento lógico.....47

Figura 37 Nodo HTTP Response 48

Figura 38 Nodo Debug para visualizar datos 48

Figura 39 Nodo Fuction dashboard encargado del procesamiento lógico de EPP para su visualización en Dashboard. 49

Figura 40 Nodo template utilizado para la visualización del estado de EPP.....50

Figura 41 Nodo template muestra imagen en tiempo real.....50

Figura 42 Nodo fuction para calcular porcentajes de cumplimiento de EPP. 51

Figura 43 Nodo Chart muestra visualmente el porcentaje de cumplimiento. 52

Figura 44 Pantalla principal de instalación Eclipse Mosquito.....53

Figura 45 Se verifica el correcto funcionamiento del Broker MQTT.....53

Figura 46 Nodo MQTT In el cual permite la suscripción al topic.....55

Figura 47 Conexión entre Node-RED y el broker Mosquito.....55

Figura 48 Nodo Fuction contiene lógica de decisión para dispositivo ESP32.....56

Figura 49 Nodo MQTT OUT para publicar mensaje de activación o desactivación.....57

Figura 50 Nodo MQTT OUT para establecer la comunicación. 58

Figura 51 Diagrama de conexión del sistema ESP32-Rele-Luz estroboscópica. 60

Figura 52 Entorno Arduino IDE instalado y operando. 60

Figura 53 Configuración del Arduino IDE para habilitar el soporte del microcontrolador ESP32.....61

Figura 54 Selección de la placa ESP32 Dev Module.....62

Figura 55 Librería PubSubClient para establecer comunicación MQTT.....63

Figura 56 Configuración del puerto serial de comunicación, Puerto COM7 63

Figura 57 Diagrama de flujo de la programación ESP32 para activar alertas físicas...64

Figura 58 Fragmento del código implementado en el ESP32. 65

Figura 59 Monitor serial Arduino IDE mostrando la recepción de comandos MQTT y activación de relé.....66

Figura 60 Proceso para la creación de bot mediante @BotFather.....67

Figura 61 Creación del TOKEN 68

Figura 62 Obtención del Chat ID.....69

Figura 63 Conexión del nodo Telegram Sender y Fuction 2 Decisión dentro del flujo principal. 70

Figura 64 Configuración interna del nodo Function para construcción del mensaje de alerta. 70

Figura 65 Configuración del nodo Telegram Sender. 71

Figura 66 Notificación automática recibida en el dispositivo móvil mediante Telegram ante incumplimiento de EPP. 72

Figura 67 Sistema completo implementado para detección y generación de alertas de incumplimientos de EPP. 73

Figura 68 Flujo de procesamiento del sistema en Node-RED. 74

Figura 69 Dashboard del sistema durante la fase de validación. experimental. 75

Figura 70 Registro automático de evidencias ante incumplimientos de EPP. 75

Figura 71 Notificación automática de incumplimiento de EPP enviada mediante Telegram al supervisor. 76

Figura 72 Resultado de la matriz de confusión del sistema para la detección de casco. 78

Figura 73 Muestra un ejemplo de detección de cumplimiento de casco. 79

Figura 74 Resultado de la matriz de confusion para la detección de chaleco 81

Figura 75 Muestra un ejemplo de detección del cumplimiento de chaleco 81

Figura 76 Resultado de la matriz de confusion para la detección de guantes 83

Figura 77 Muestra un ejemplo de detección de guantes 84

Figura 78 Resultado de la matriz de confusión para sistema global 87

Figura 79 Muestra un ejemplo de detección de cumplimiento de todos los EPP 87

Figura 80 Conexión RTSP para captura de video 92

Figura 81 Imagen modelo YOLOv8 92

Figura 82 Diseño de serigrafia instalado en sistema de detección. 107

Figura 83 Sistema de detección de EPP 108

Figura 84 Flujo de nodos en Node-RED 108

Figura 85 Monitoreo en tiempo real de detección de EPP 109

Figura 86 Diagrama de flujo para comunicación MQTT y activación de alertas físicas 111

RESUMEN

El propósito de este proyecto es desarrollar e implementar un sistema de detección autónoma de Equipos de Protección Personal (EPP) basado en visión artificial y redes neuronales profundas. La solución propuesta utiliza el algoritmo Yolov8 para la detección de objetos en tiempo real, utilizando una cámara IP configurada con dirección fija y transmisión mediante el protocolo de comunicación RTSP, el procesamiento del video se realizó mediante Python-OpenCV, utilizando el modelo convolucional preentrenado YOLOv8 para la identificación de elementos de protección personal como casco, chaleco y guantes. El sistema permite generar alertas y visualizar eventos relevantes. Facilitando la supervisión automatizada del cumplimiento de la norma técnica de seguridad en el trabajo.

Las pruebas realizadas han demostrado que el sistema permite una detección eficiente de objetos (EPP) en tiempo real, como por ejemplo el casco, el chaleco y los guantes. También se corroboró que el sistema genera una correcta exportación de datos hacia Node-RED y dashboard para la activación de alertas físicas y notificaciones al móvil. Esta aplicación facilita la viabilidad de integrar redes neuronales profundas con dispositivos de videovigilancia, ofreciendo una solución flexible, escalable y aplicable a sistemas de seguridad inteligente.

Palabras claves: Visión artificial, Yolov8, Python

ABSTRACT

The purpose of this project is to develop and implement an autonomous detection system for Personal Protective Equipment (PPE) based on artificial vision and deep neural networks. The proposed solution uses the Yolov8 algorithm for real-time object detection. Using an IP camera configured with a fixed address and transmission via the RTSP communication protocol, the video processing was performed using Python-OpenCV, using the pre-trained YOLOv8 convolutional model for the identification of personal protective equipment such as helmet, vest and gloves. The system allows for the generation of alerts and the visualization of relevant events, facilitating automated monitoring of compliance with workplace safety standards.

Tests have shown that the system allows for efficient, real-time detection of personal protective equipment (PPE), such as helmet, vest, and gloves. It was also verified that the system generated a correct export of data to Node-RED and dashboard for the activation of physical alerts and mobile notifications. This application facilitates the feasibility of integrating deep neural networks with video surveillance devices, offering a flexible scalable and solution applicable to intelligent security systems.

Keywords: Computer vision, Yolov8, Python

INTRODUCCIÓN

Durante los últimos años, las tecnologías de visión artificial y las redes neuronales se han consolidado como herramientas eficaces para automatizar tareas de supervisión y control en la industria. En particular, la integración de estos avances con algoritmos modernos de detección basados en redes neuronales convolucionales permite desarrollar sistemas eficientes, escalables y de bajo costo, capaces de operar en entornos reales con requerimientos mínimos de infraestructura y sin necesidad de equipamiento especializado.

Un gran desafío para las industrias es garantizar la seguridad física de sus trabajadores mediante el uso adecuado de Equipos de Protección Personal (EPP), en las industrias por normativa interna es reglamentario utilizar EPP para prevenir accidentes laborales, esto con lleva a depender del factor humano y supervisión manual, los mismos que pueden ser imprecisos, discontinuaos y muy costosos. además, esto puede inferir en accidentes laborales si la empresa no proporciona protección o integración de nuevas tecnologías, exceso de confianza o poca capacitación y supervisión para el uso adecuado de estos EPP, bajo presupuesto para implementación de nuevas tecnologías conlleva a altos índices de accidentes laborales en las industrias, esto además afecta a la disminución de la productividad en líneas de producción, aumento de costos por indemnizaciones a obreros por accidentes laborales, tiempos muertos en control por parte del supervisor, detecciones tardías para prevenir accidentes laborales, todo esto puede derivar en sanciones legales o pérdidas económicas para la empresa.

Debido a esta problemática, se propone una solución automatizada basado en visión artificial y redes neuronales YOLOv8. La investigación se orientó al diseño e implementación de un sistema de detección de objetos en tiempo real, específicamente casco, chaleco y guantes, el sistema identifica a la persona en movimiento y verifica si cuenta con los equipos de protección personal, caso contrario se emitirán las alertas respectivas cuando la persona incumpla las normas de seguridad, utilizando una cámara IP y técnicas de visión artificial basadas en el algoritmo Yolov8. El proyecto se centra en la adquisición del video desde una cámara IP, su procesamiento mediante un modelo de aprendizaje profundo preentrenado mediante una base de datos, la generación de resultados que van a ser integrados a plataformas de automatización y visualización como

Node-Red donde se podrá monitorear los datos obtenidos por el sistema, además se activarán alertas visuales y él envió de notificaciones ante el incumplimiento de algún EPP. Desde el ámbito industrial el sistema propuesto aporta una solución tecnológica viable y escalable adaptando aplicaciones de videovigilancia inteligentes, monitoreo y apoyo a la toma de decisiones, reduciendo la dependencia de supervisión manual y prevención de accidentes laborales mediante el uso de tecnología accesible y de bajo costo.

Formulación del problema de investigación

¿Qué aporte puede generar el implementar un sistema de detección automática de elementos de protección personal basado en visión artificial y redes neuronales en el ámbito industrial?

Objetivo General:

Diseñar e implementar un sistema para la detección de Equipos de Protección Personal (EPP) mediante visión artificial y redes neuronales para entornos industriales.

Objetivos Específicos:

- Entrenar un modelo de redes neuronales basado en visión artificial para la detección automática de EPP, tales como casco, chaleco y guantes, utilizando una base de datos de imágenes que incluya personas con y sin el uso de estos EPP.
- Implementar un sistema de detección y alertas en un área critica de la empresa que permita identificar la ausencia de EPP en el personal.
- Validar el funcionamiento del sistema mediante pruebas reales que permita evidenciar su adecuada operación y aplicabilidad.

Planteamiento hipotético

¿Qué nivel de eficiencia presenta el algoritmo Yolov8 en la detección de casco, chaleco y guantes a partir de flujos de video obtenidos desde la cámara IP mediante el protocolo de comunicación RTSP?

¿De qué manera un sistema de visión artificial y redes neuronales profundas puede optimizar los procesos de monitoreo y videovigilancia frente a métodos tradicionales?

CAPÍTULO 1. MARCO TEÓRICO REFERENCIAL

Mediante el análisis de diversas fuentes académicas, se realizó una búsqueda detallada para recopilar información relevante relacionada con el tema de investigación.

1.1. Revisión de literatura

Durante los últimos años, las nuevas tecnologías de visión artificial y redes neuronales se han convertido en una herramienta eficaz en la autonomía de tareas de supervisión y control en la industria, especialmente para detectar el uso adecuado de los equipos de protección personal (EPP) Internacionalmente se han desarrollado diversos proyectos que demuestran la eficacia de los sistemas de visión artificial en la detección autónoma del uso de (EPP).

Asimismo, diversos estudios como el de (Massiris et al., 2021) plantean propuestas innovadoras para el monitoreo del uso de EPP en el sector de la construcción. Dichos sistemas, basados en visión por computador y redes neuronales profundas, permiten automatizar la detección mediante análisis de imágenes digitales.

Esta fase consiste en analizar y diseñar un prototipo de solución que sea capaz de satisfacer la necesidad de identificar los EPP. Uno de los trabajos de referencia obtiene en un muy buen porcentaje de confianza (0.99) en la identificación de EPP en la industria de la construcción. Aunque para ello fue necesario la inversión de un alto costo computacional, para lograrlo utilizan procesamiento de videos de alta calidad, se obtienen las imágenes, se identifican los puntos antropométricos y luego de ello se identifica si poseen los EPP. Con el objetivo de reducir los costos computacionales elevados que acarrea dicha implementación, se opta por resolver el problema de una manera más sencilla empleando una red neuronal convolucional de detección de objetos en una imagen. Para esta implementación se decide utilizar la red YOLOv5 debido a su gran precisión, bajo costo computacional y completa documentación en el momento que se inicia el presente proyecto, razón por la cual se elige esa versión de la red. Se decide realizar el prototipo en lenguaje Python utilizando la librería Pytorch, que permite integrar fácilmente el modelo YOLOv5 y manipular sus resultados con facilidad.

El prototipo consta de dos etapas, entrenamiento y funcionamiento. La etapa de entrenamiento de la red requiere de un conjunto de imágenes etiquetadas con sus

respectivas clases a fin de producir un conjunto de pesos que permitan ser utilizados a posteriori. En la etapa de funcionamiento, los pesos obtenidos en el aprendizaje son utilizados para configurar la red y las imágenes capturadas por un dispositivo son procesadas por la misma. En función de los pesos introducidos a la red y la fotografía utilizada como imagen de entrada, se obtendrá un conjunto de detecciones, las cuales sirven de entrada a un algoritmo de decisión que determinará si cada persona en la imagen está utilizando sus EPP.(Rufino, 2024)

En cuanto al ámbito Nacional, Existen investigaciones relevantes en el desarrollo tecnológico en seguridad industrial mediante visión artificial. En la escuela superior politécnica de Chimborazo se desarrolló un sistema basado en visión artificial para el reconocimiento y control del epp en el personal operativo de la línea de extrusión de alimentos para mascotas petfood en balanceados exibal matriz chambo" Se utilizó Python como lenguaje de programación, una NVIDIA Jetson Nano B01 como SBC, además se empleó el modelo de Deep Learning ResNet18-Body para la segmentación del cuerpo humano y la detección de color se realizó con Visión Artificial, logrando desarrollar un prototipo con una eficiencia del 72%. (Chaguancallo & León, 2023)

El proyecto plantea la implementación de un sistema de detección de (EPP) basado en visión artificial utilizando una cámara IP para el monitoreo de los trabajadores. El sistema permite analizar las imágenes capturadas por la cámara IP para identificar el uso adecuado de los equipos de protección personal por parte del trabajador con el fin de disminuir riesgos laborales. Para ello, se emplean técnicas de visión artificial y redes neuronales profundas. En una primera etapa, se utiliza la red neuronal OpenPose para identificar la estructura corporal del trabajador, lo que permite localizar regiones de interés como cabeza y brazos del trabajador. El sistema será implementado en el entorno operativo de la empresa Distribución Eléctrica S.A., donde se aplicará un software que permitirá a los usuarios finales visualizar las alertas generadas por el sistema, además de recibir notificaciones mediante correo electrónico. Posteriormente, el desempeño del sistema de clasificación será evaluado a través de métricas de análisis como la precisión, la sensibilidad y la especificidad (Carpio et al., 2021).

Una cámara monitorea el sector, y mediante un algoritmo, se detecta si un trabajador está utilizando o no los E.P.P. obligatorios. El personal de seguridad industrial tendrá a su

disposición una pantalla donde se ve el video, y sonará una alerta cuando un operario no utilice los E.P.P. durante un tiempo determinado, para alertar al encargado mientras esté realizando otra actividad. Además, el sistema realizará una captura de imagen y la almacenará para que el personal de seguridad e higiene la vea posteriormente en caso de que en ese momento no se encuentre en su oficina.(Digitales Iii et al.,2023)

1.2. Desarrollo teórico y conceptual

1.2.1 Tecnología de Hardware

Cámara IP según los requerimientos y presupuesto, se puede utilizar una cámara conectada por wifi o por red loca, procesando las imágenes a una Pc, cámara IP dispositivo digital de videovigilancia que captura y transmite video y audio a través de una red IP (Protocolo de Internet) como internet o una red local (LAN). A diferencia de las cámaras analógicas tradicionales. Se puede observar en la Figura 1.

Figura 1

Camara Dahua

Nota. Fuente: Esta imagen pertenece, Dahua,2025, Cámaras Dahua FlexVu,

(https://www.hikvision.com)

1.2.2 Visión Artificial

La visión artificial conforma un campo de la inteligencia artificial orientado al análisis e interpretación de información visual proveniente de secuencias de video. Esta área permite a los sistemas computacionales identificar objetos, personas o situaciones específicas dentro de un entorno determinado mediante el análisis de información visual.

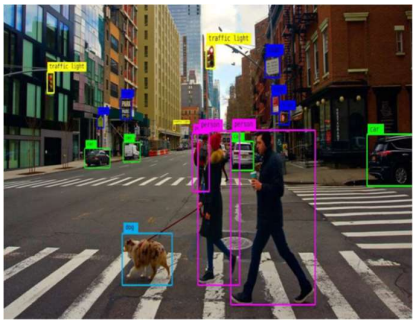
Su propósito es procesar imágenes capturadas de un entorno real mediante técnicas computacionales, con la finalidad de que los sistemas puedan comprender y extraer información relevante de dichas imágenes. La evolución de la visión artificial se remonta a la década de 1960 y con el paso del tiempo se ha convertido en una de las herramientas

más utilizadas para la identificación y reconocimiento de patrones de imágenes. Actualmente sus aplicaciones abarcan diversos campos, entre ellos la video vigilancia, el reconocimiento facial y el funcionamiento de sistemas robóticos (Borrella Petisco, 2022).

La imagen correspondiente se observa en la Figura 2.

Figura 2

Detección de Humanos

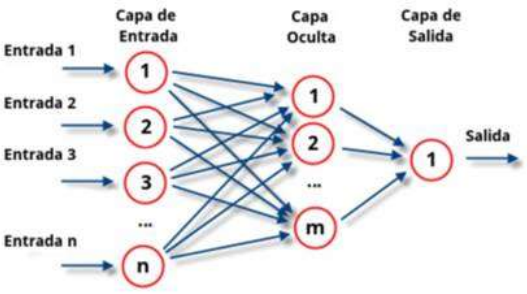


Nota. Fuente: Adaptado de Algotive (p. 1), Algotiv, 2022, Algorithmic Objective Corp.(https://www.algotive.ai/es-mx/blog/que-es-la-vision-artificial-y-como-funciona-con-la-inteligencia-artificial).

1.2.3 Redes Neuronales y Yolov8

Figura 3

Red Neuronal



Nota. Sciel, Revista Cubana de ciencias Informaticas,2020, (http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992020000300165)

Las redes neuronales convolucionales (CNN) como se puede apreciar en la Figura 3, son ampliamente utilizadas para el reconocimiento de imágenes YOLOv8 (You Only Look Once Versión 8) es un modelo de detección de objetos en tiempo real que, a demostrado alta precisión y velocidad, ideal para entornos industriales donde se requiere respuesta inmediata. Según (Yaseen, 2024) Un avance significativo en la detección de objetos llegó con la introducción del algoritmo You Only Look Once (YOLO) La serie YOLO revolucionó el campo al enmarcar la detección de objetos como un problema de regresión única, donde una red neuronal convolucional procesa una imagen completa en una sola pasada para predecir cuadros delimitadores y probabilidades de clase Este enfoque marcó un cambio con respecto a los métodos tradicionales de detección multietapa, ofreciendo mejoras significativas en velocidad y eficiencia. Basándose en el éxito de sus predecesores, YOLOv8 introduce innovaciones arquitectónicas y metodológicas avanzadas que mejoran significativamente su precisión, eficiencia y usabilidad en la detección de objetos en tiempo real. (Yaseen, 2024)

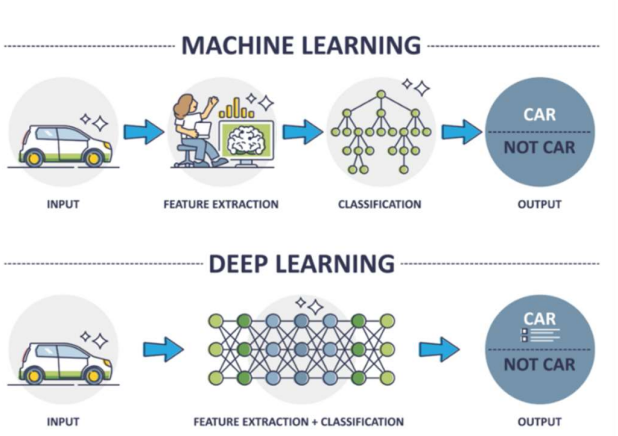
1.2.4 Machine Learning (ML)

El aprendizaje automático, conocido como Machine Learning, constituye una subdisciplina de la inteligencia artificial que permite a los sistemas informáticos aprender a partir del análisis de datos y experiencias previas. A través de este enfoque, los algoritmos pueden identificar patrones y mejorar su desempeño sin necesidad de ser programados explícitamente para cada tarea específica. Sintetizan e interpretan la información para el entendimiento humano, de acuerdo a parámetros preestablecidos, ayudando ahorrar tiempo, disminuir errores, crear acciones preventivas y automatizar procesos en grandes operaciones y empresas. En la actualidad, la visión artificial presenta un avance significativo destacándose por sus diversas aplicaciones y su proyección dentro del campo de la inteligencia artificial. (Algoviye, 2021) El Deep Learning es un subconjunto del machine learning que utiliza redes neuronales multicapa, llamadas redes neuronales profundas, para simular el complejo poder de toma de decisiones del cerebro humano. Algunas formas de deep learning impulsan la mayoría de las aplicaciones de inteligencia artificial (IA) en nuestra vida actual. La principal diferencia entre el deep learning y el machine learning es la estructura de la arquitectura de red neuronal subyacente. Los modelos tradicionales de machine learning “no profundos” utilizan redes neuronales simples con una o dos capas computacionales. Los modelos de deep learning

utilizan tres o más capas, pero normalmente cientos o miles de capas, para entrenar los modelos. (www.Tech Global University, 2025) se puede apreciar en la Figura 4.

Figura 4

Machine learning y deep learning



Nota. Adaptada de Deep Learning Tutorial for Beginners – A Complete Roading, por V.Gupta,2026, igmGuru (https://www.igmguru.com/blog/deep-learning-tutorial).

1.2.5 Internet de las cosas (IoT)

Los dispositivos IoT, también conocidos como "objetos inteligentes", pueden variar desde simples dispositivos domésticos inteligentes., como termostatos inteligentes, hasta dispositivos portátiles, como relojes inteligentes y ropa con RFID, hasta complejas maquinaria industrial y sistemas de transporte. Los tecnólogos incluso están imaginando “ciudades inteligentes” enteras basadas en tecnologías IoT. El internet de las cosas (IoT) permite que estos dispositivos inteligentes se comuniquen entre sí y con otros dispositivos habilitados para Internet. Al igual que los teléfonos inteligentes y las puertas de enlace, se crea una amplia red de dispositivos interconectados que pueden intercambiar datos y realizar diversas tareas de forma autónoma. Como se puede apreciar en la Figura 5.

Figura 5

Imagen IoT

Nota. Adoptado de Tecno seguros IOT, Jairo Rojas Campo, 2023, (https://www.tecnoseguro.com/analisis/pro/que-es-iot-su-impacto-industria-seguridad-electronica).

1.2.6 Software

El software corresponde al conjunto de programas e instrucciones que permiten a un sistema informático ejecutar diferentes tareas y procesos específicos. También se conoce como aplicaciones de software, paquetes de software, herramientas de software y programas de software. El software puede utilizarse para gestionar datos, automatizar procesos y crear aplicaciones o productos informáticos. Su complejidad puede variar desde un simple programa de tratamiento de textos hasta complejos sistemas informáticos que controlan infraestructuras críticas en sectores como la sanidad y el transporte. (Qué Es Software - Definición, Tipos y Ejemplos de Software, 2025)

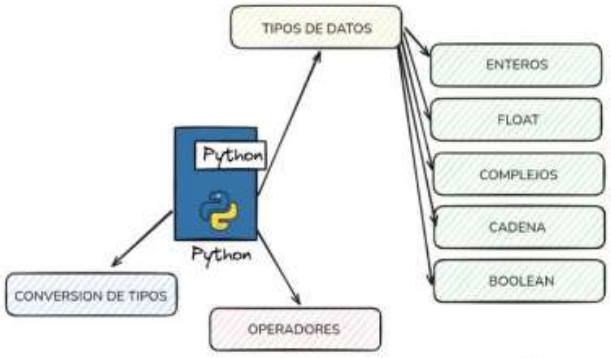
1.2.7 Python

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo.

(¿Qué Es Python? - Explicación Del Lenguaje Python - AWS, 2025), como se puede apreciar en la Figura 6.

Figura 6

Python con sus librerías



Nota. Apuntes de Walther Curo, Walther Curo De La Cruz ,2025

(http://blog.walthercuro.com/tipos-de-datos-en-python/)

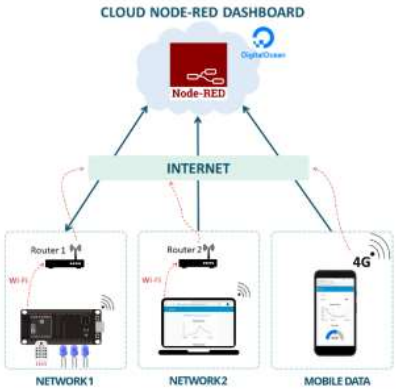
1.2.8 Node-RED

Node-RED es una plataforma de programación orientada a la integración de dispositivos de hardware, interfaces de programación de aplicaciones (API) y diversos servicios que pueden conectarse en línea. Es una herramienta que proporciona un entorno visual, permite crear y gestionar flujos de trabajo mediante la conexión de diferentes nodos disponibles en su biblioteca.

A través de esta interfaz gráfica, los usuarios pueden diseñar de forma sencilla la comunicación entre distintos componentes del sistema, facilitando la construcción de aplicaciones del internet de las cosas (IoT) y sistemas de automatización. Los flujos desarrollados pueden desplegarse directamente en el entorno de ejecución con una única acción, lo que simplifica el proceso de desarrollo y la implementación del sistema (Tupunatron, 2025), el esquema de funcionamiento de Node-RED se puede apreciar en la Figura 7.

Figura 7

Node-Red



Nota. Adaptado de Tupunatron, Node-red software de desarrolllo,2025,

(https://randomnerdtutorials.com/access-node-red-dashboard-anywhere-digital-ocean/)

1.2.9 ESP 32

El ESP32 es un microcontrolador ampliamente utilizado en proyectos relacionados con el internet de las cosas (IoT), gracias a sus capacidades de conectividad inalámbrica y su alto rendimiento para aplicaciones embebidas, dicha tarjeta se la puede apreciar en la figura 8.

- Estándares: soporta IEEE 802.11 b/g/n en la banda de 2.4 GHz.
- Velocidad: permite velocidades de transferencia de datos que pueden alcanzar aproximadamente 150 Mbps mediante tecnología HT40.
- Seguridad: soporta distintos mecanismos de protección inalámbrica como WEP, WPA/WPA2/WPA3, además de WPS para facilitar la configuración de red.
- Protocolo ESP-NOW: Un protocolo propietario de baja potencia que permite comunicación directa entre dispositivos ESP32 sin necesidad de un router, logrando alcances de hasta 480 metros o más.

Modos de Operación

El microcontrolador ESP32 puede funcionar en diferentes modos de operación de acuerdo con los requerimientos del sistema o del proyecto:

1. Station (STA): en este modo el dispositivo se conecta a un punto de acceso existente (como el router del hogar) para acceder a internet o a una red local.
2. Access Point (AP): en este modo el ESP32 crea su propia red Wi-Fi (SoftAP), permitiendo que otros dispositivos (celulares, laptops) se conecten directamente a él sin necesidad de un router.
3. Híbrido (AP+STA): Funciona como estación y punto de acceso al mismo tiempo, ideal para configurar credenciales de red de forma remota.
4. Sniffer: este modo permite monitorear y capturar paquetes Wi-Fi que circulan en el aire dentro de su rango.

Figura 8

Tarjeta ESP 32 WI-FI

Nota. Fuente: Adaptada de SounFounder, ESP32 Placa,2025, (https://docs.sunfounder.com)

1.2.10 RELE

Módulo relé de 1 canal, con el aislamiento fotoeléctrico, LED de energizado e indicador de acción del relé, como se puede apreciar en la figura 9.

El voltaje de funcionamiento del módulo es de 5Voltios, el módulo utiliza relé de calidad, carga máxima: CA 250V/10A, DC 30V/10A. Utiliza diodo freewheeling (volante) en paralelo con cargas inductivas para evitar sobrevoltajes.

Tamaño: 44x 24x 17 mm

Peso: 16g

Interfaz del módulo:

DC +: alimentación positiva (VCC) (5V)

DC - : fuente de alimentación negativo(GND)

IN: Terminal de activación, corriente de excitación debe ser superior a 4 mA

COM: Control de bucle de cable a tierra

GND: Tierra

Figura 9

Modulo Rele 5V

Nota. Fuente: Adaptado de Electronics,2025,(https://avelectronics.cc/producto/modulo-rele-1-canal)

CAPÍTULO 2. METODOLOGÍA

2.1. Contexto de la investigación

El proyecto se desarrolla en la zona 3 Provincia de Chimborazo, y está ubicado en el cantón Riobamba en la empresa "Alumvid, Mz C - Sarajevo y Zagreb. La empresa está dedicada a la fabricación, ensamblaje e instalación de estructuras de aluminio y vidrio, cuenta con áreas de trabajo donde el uso adecuado de equipos de protección personal es un requisito fundamental para garantizar la seguridad ocupacional. En este entorno, la empresa dispone de un sistema de videovigilancia instalados con fines de control y seguridad; sin embargo, dicho sistema no es monitoreado de manera continua ni es utilizado para la verificación automática del cumplimiento de normas de seguridad. Asimismo, la empresa cuenta con una oficina técnica equipada con una computadora de alto rendimiento, la cual dispone de las características necesarias para la implementación de un sistema de detección de equipos de protección personal mediante visión artificial y redes neuronales, lo que permite la ejecución del presente proyecto en un entorno real y controlado.

2.2. Diseño y alcance de la investigación

Mediante el enfoque tecnológico-aplicado se han reforzado los conocimientos en visión artificial, y mediante la investigación Experimental, se ha revisado a profundidad conceptos relacionados con la red neuronal convolucional Yolov8. ya que se pudo manipular variables técnicas como son: la fuente del video (cámara IP), el procesamiento de flujo (RTSP), la ejecución del modelo Yolov8, los parámetros de detección y visualización y sobre todo el enfoque tecnológico que se basó en el uso, integración y validación utilizando tecnologías existentes como son: cámara IP, protocolo RSTP, lenguaje Python, bibliotecas Opencv, redes neuronales profundas, plataformas de automatización (Node-red) brindando así una solución tecnológica moderna para la detección de elementos de protección personal en una área critica de la empresa Alumvid, asimismo se espera que esta investigación aporte a la empresa en los altos índices de accidentes laborales .

2.3. Tipo y métodos de investigación

El enfoque de investigación es cuantitativo ya que la investigación se basa en una medición objetiva es decir de uso o no uso de los EPP mediante un sistema automático

que detecta patrones visuales. Los resultados se expresarán en datos numéricos tales como:

• Porcentaje de aciertos en la detección en EPP

• Tasa de falsos positivos o negativos

• Cantidad de alertas generadas

• Tiempo de respuesta del sistema

2.4. Población y muestra

La población de este estudio está orientada a los entornos industriales en la cual se requiere el uso obligatorio de equipos de protección personal, como en fábricas, plantas de producción, obras de construcción y empresas logísticas. En estos entornos los obreros o técnicos operativos de la empresa Alumvid son los que desempeñan actividades de riesgo, ellos representan el grupo humano objetivo de la investigación.

La muestra se fundamentará en un área de trabajo real donde se requiere el uso de EPP, con un grupo de 3 personas interactuando en el área monitoreada usando y no usando los equipos de protección personal. El objetivo principal es probar el desempeño del sistema automatizado para la detección mediante visión artificial. se desarrollará en la zona 3 Provincia de Chimborazo cantón Riobamba en la empresa Prosetel.

2.5. Técnicas e instrumentos de recolección de datos

A través de una entrevista con el propietario de la empresa Alumvid, se abordó la preocupación sobre los accidentes laborales en la empresa debido a la no utilización de elementos de protección personal por parte de los técnicos, además la falta de supervisión continua, se propuso la implementación de un sistema de innovación tecnológica de visión artificial y redes neuronales para la detección del no uso de elementos de protección personal en el área más crítica de la empresa.

La implementación de un sistema automatizado surge como una solución para fortalecer la supervisión de EPP para prevenir accidentes laborales.

En el sistema propuesto, los datos fueron obtenidos de manera automática, la técnica empleada es la observación directa, ya que el sistema observa el entorno industrial de manera continua, utilizando una cámara IP, sin intervención humana, sus características

no dependen del criterio del observador, se realizan en tiempo real, se basa en procesamiento digital de imágenes.

Los instrumentos de recolección de datos cámara IP recolecta datos como son los frames de video, posición y presencia de persona, elementos visibles de protección personal, la cámara actúa como sensor visual del sistema, YOLOv8 procesa cada imagen capturada, detecta objetos de interés (EPP), los datos recolectados son las clases (casco, chaleco, guantes) este instrumento es el que transforma imágenes en datos numéricos.

Script de Python ejecuta la lógica del sistema, registra resultados, envía datos a Node-red, los datos recolectados y procesados son número de detecciones por clase, eventos de incumplimiento, los tiempos de procesamiento y la frecuencia de las alertas.

Node-red almacena y visualiza la información, gestiona las alertas, registra historial de eventos, fecha y hora de detección.

2.6. Procesamiento de la evaluación: Validez y confiabilidad de los instrumentos aplicados para el levantamiento de información.

En el desarrollo de este proyecto, los instrumentos para el levantamiento de información estuvieron constituidos principalmente por el sistema de visión artificial basado en el modelo YOLOv8, la cámara IP, el módulo ESP32 y el sistema de monitoreo y alertas implementadas en Node-RED.

La validación de los instrumentos se realizó a través de pruebas realizadas en un entorno real de trabajo, comprobando que el sistema detecte correctamente el uso o mal uso de los EPP establecidos en normativas de seguridad industrial. Asimismo, se utilizó un conjunto de datos previamente etiquetados, es decir una base de datos que contengan fotografías de personas utilizando y no utilizando casco, chaleco, guantes, y posteriormente validados durante el entrenamiento del modelo YOLOv8, lo que permitió asegurar que las detecciones correspondan al planteamiento del sistema.

En cuanto a la confiabilidad, se evaluó el comportamiento repetitivo del sistema ejecutando múltiples pruebas en diferentes momentos y condiciones de iluminación, comprobando que los resultados obtenidos sean consistentes en el tiempo. La confiabilidad estadística del sistema se midió a través de métricas de desempeño, usando

la matriz de confusión como herramienta de evaluación cuantitativa. La matriz de confusión se basa en cuatro posibles resultados, como se puede observar en la Tabla 1.

Tabla 1

Parámetros de evaluación del sistema

Verdaderos positivos (TP)	Corresponde a los casos en los que el sistema detecta correctamente la ausencia de EPP cuando realmente el trabajador no la está utilizando.
Verdaderos negativos (TN)	Representa los casos en los que el sistema identifica correctamente que el trabajador cumple con el uso de EPP.
Falsos positivos (FP)	Representa cuando el sistema genera una alerta de incumplimiento pese a que el trabajador si está utilizando correctamente el EPP.
Falsos negativos (FN)	Representa cuando el sistema no genera alerta a pesar de que existe una ausencia real de EPP.

Nota. Fuente: El autor.

Una vez obtenido los parámetros de evaluación del sistema, a partir de ellos se calculan las siguientes métricas, dichas métricas se pueden observar en la Tabla 2.

Tabla 2

Métricas para evaluación del sistema

Precisión (Precisión) $\text{Precisión} = \frac{TP}{TP+FP}$	Indica la proporción de alertas generadas por el sistema que corresponde a incumplimientos.
Sensibilidad (Recall) $\text{Recall} = \frac{TP}{TP+FN}$	Mide la capacidad del sistema para detectar correctamente los casos reales de incumplimiento.
Exactitud (Accuracy) $\text{Accuracy} = \frac{TP+TN}{Total}$	Representa el porcentaje global de decisiones correctas realizadas por el sistema respecto al total de pruebas realizadas.

Nota. Fuente: El autor.

La confiabilidad del sistema de comunicación se evaluó mediante ensayos continuos de trasmisión de datos entre Python, Node-RED y el Esp32, con el fin de verificar la correcta activación de alertas visuales y notificaciones ante la detección de incumplimientos de los EPP.

2.7. Metodología de Desarrollo

La metodología aplicada en este proyecto se basa en el desarrollo e implementación de un sistema de detección de EPP. El proceso incluye la configuración del entorno de desarrollo, la preparación del conjunto de datos, el entrenamiento del modelo de detección y la integración del sistema para generar alertas ante incumplimiento de EPP, cada una de estas etapas se describe a continuación.

2.7.1 Configuración del entorno de desarrollo

Para el desarrollo del sistema se configuro un entorno de trabajo que permite la ejecución y prueba de los algoritmos de visión artificial. Se utilizaron herramientas de programación en Python junto con librerías especializadas para el procesamiento de imágenes y aprendizaje profundo.

2.7.2 Instalación del lenguaje de programación Python

Se ha seleccionado el lenguaje de programación Python por su amplio uso en sistemas embebidos, aplicaciones de visión artificial, aprendizaje profundo, así como por la gran disponibilidad de sus librerías especializadas y su compatibilidad con plataformas de procesamiento en tiempo real.

La correcta instalación de Python constituye la base fundamental para la implementación del sistema, ya que sobre este lenguaje se ejecutan los algoritmos de captura de video, procesamiento de imágenes, detección mediante redes neuronales y comunicación con otros módulos del sistema.

La instalación se ha realizado en un computador personal el cual actúa como unidad central de procesamiento del sistema. Se ha instalado la versión Python 3.10 recomendada por frameworks de visión artificial como YOLOv8 y librerías de aprendizaje profundo, durante el proceso de instalación se activó la opción “Add Python to PATH”, lo que permite ejecutar Python y sus herramientas asociadas desde la línea de comandos del sistema, modo de instalación estándar, y el gestor de paquetes pip y las herramientas

básicas de desarrollo, para verificar su correcta instalación se verifico ejecutando el comando python --versión desde la terminal cmd, confirmando que la versión instalada sea correcta, cuyos comandos de verificación se encuentran en la Tabla 3.

Tabla 3

Comandos para verificar instalación correcta de Python

Descarga de Python desde sitio oficial	(https://www.python.org)
Selección de versión	(Python 3.10.11)
Activación para ejecución de Python	Add Python to PATH
Verificación de instalación correcta	python --version pip --version

Nota. Fuente: El autor.

2.7.3 Instalación de librerías para visión artificial y comunicación

Como siguiente paso se ha instalado las librerías necesarias cuyos comandos se muestran en la Tabla 4, y se ha empleado un entorno virtual de Python el cual ha permitido aislar las librerías del sistema operativo, inicialmente, se creó y activo un entorno virtual mediante el gestor de entornos integrado en Python, además se utilizó el gestor de paquetes pip para la instalación de las librerías requeridas, estas librerías instaladas permiten el procesamiento de imágenes, la detección automática de elementos de protección personal y la comunicación entre los distintos módulos del sistema, constituyendo así la base del entorno de desarrollo del proyecto.

Tabla 4

Comandos para activar librerías

Etapa	Descripción	Comando ejecutado en CMD
Creación del entorno virtual	Se creo un entorno virtual para aislar las dependencias del proyecto	python -m venv yoloenv
Activación del entorno virtual	Se activo en entorno virtual para instalar las librerías dentro del proyecto	yoloenv\Scripts\activate
Actualización de pip	Se actualizo el gestor de paquetes para evitar errores de compatibilidad	python -m pip install --upgrade pip

Instalación de YOLOv8	Se instalo la librería Ultralytics para detección de objetos mediante redes neuronales	pip install ultralytics
OpenCV-python (cv2)	Para captura y procesamiento de video RTSP	pip install OpenCV-python
Numpy	Para operaciones con matrices de imágenes (OpenCV las usa internamente)	pip install numpy
Paho-mqtt	Comunicación MQTT con ESP32 y Node-RED	pip install paho-mqtt
Instalación de Requests	Se instalo la librería para él envió de datos mediante solicitudes HTTP	pip install requests
Verificación de librerías	Se verifico la correcta instalación importando las librerías en Python	Python -c "import cv2, ultralytics, paho,mqtt.client,requests"

Nota. Fuente: El autor.

2.7.4 Configuración de Node-RED

Para la integración, visualización y gestión de la información que se generó del sistema de detección de los elementos de protección personal, se utilizó la herramienta Node-RED, cuyos comandos utilizados se muestran en la Tabla 5.

Tabla 5

Configuración e instalación de Node-RED.

Elemento	**Descripción/Ejecución en CMD**
Plataforma	Node-RED
Sistema operativo	Windows
Instalamos el entorno de ejecución Node.js	(https://nodejs.org/es/download)
Corroborar instalación correcta de Node.js	En CMD ejecutamos node --version
Directorio de trabajo	Directorio de usuario de Node-RED node-red -p 1882 -u C:\UPSE_1882
Verificamos la correcta instalación del gestor de paquetes	npm --version
Comando de instalación de Node-RED	npm install -g –unsafe-perm node-red

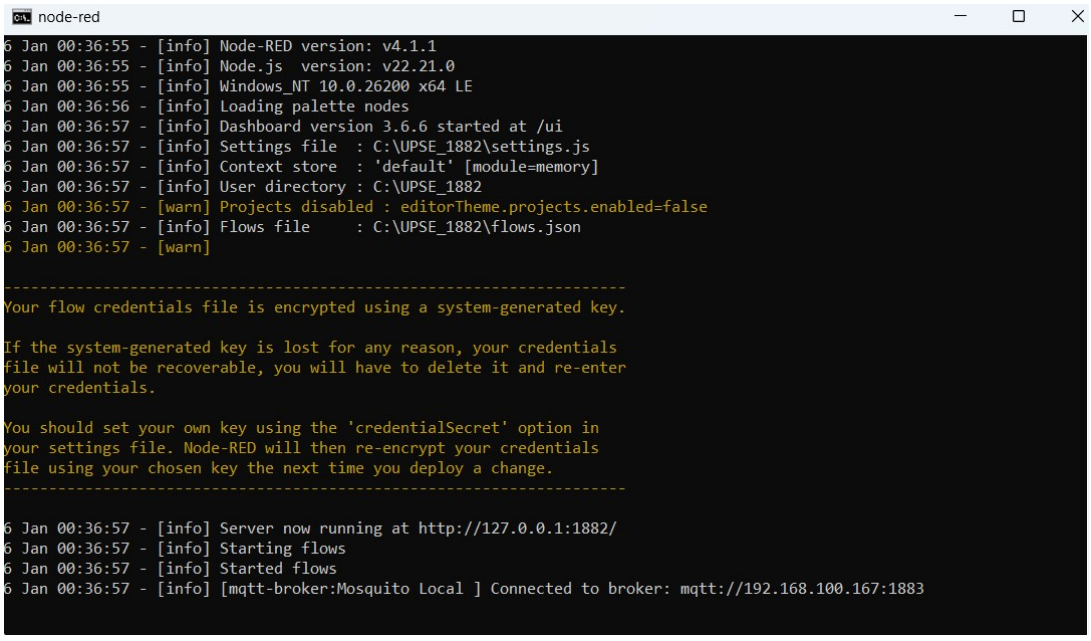
Comando para ejecutar Node-red para que empiece a correr en nuestra plataforma Windows	Node-RED
Puerto de ejecución por defecto	1880
URL de acceso	http://localhost:1880
Protocolo de entrada	HTTP(POST)
Puerto utilizado e indicado a Node-RED para su ejecución	1882
Cambia el puerto de servicio	-p 1882
Define donde se guardan los archivos	-u C:\UPSE_1882
Protocolo de salida	MQTT
Función principal	Gestión de la lógica del sistema y visualización de alertas
Dashboard	Diseño de pagina
Función del dashboard	Visualización del estado del sistema y alertas EPP

Nota. Fuente: El autor.

Se ha instalado como plataforma de integración y orquestación de flujos mediante el entorno de ejecución Node.js. la instalación se realizó a través del gestor de paquetes npm**,** permitiendo la ejecución del servicio mediante líneas de comandos. Una vez iniciado, el entorno de desarrollo se accede a través de un navegador web utilizando la URL local correspondiente al puerto de ejecución configurado, el acceso a Node-RED se realizó a través del puerto 1882 debido a que el servidor fue iniciado manualmente especificando dicho puerto con el parámetro -p, en lugar del puerto por defecto. node-red -p 1882 -u C:\UPSE_1882 como se muestra en la Figura 10.

Figura 10

Ejecución de Node-RED desde la línea de comandos y URL de acceso al entorno



Nota. Fuente: El autor.

2.7.5 Configuración de bróker MQTT (Mosquito)

La configuración del bróker MQTT Mosquito ha sido realizada mediante comandos ejecutados en la línea de comandos del sistema operativo Windows. Mosquito fue instalado y configurado como un servicio automático, permitiendo su ejecución permanente sin intervención manual. El bróker escucha en el puerto estándar 1883 y permite conexiones anónimas dentro de la red local, facilitando la comunicación entre el sistema de detección desarrollado en Python, la plataforma Nore-RED y el microcontrolador ESP32. Para ello se utilizó los comandos mostrados en la Tabla 6.

Tabla 6

Configuración e Instalación Bróker Mosquito

Elemento	Descripción/Comando utilizado
Bróker MQTT	Mosquito
Sistema operativo	Windows
Puerto de comunicación	1883
Protocolo	MQTT
Método de instalación	(https://mosquitto.org/)
Directorio de instalación	C\ProgramFiles\mosquito\
Archivo de configuración	mosquito.conf
Comando para verificar servicio	sc query mosquito
Configuración como servicio	sc config mosquito start= auto
Comando para iniciar el servicio	net start mosquito
Comando para detener el servicio	net stop mosquito
Modo de ejecución	Servicio automático del sistema
Permitir conexiones externas	listener 1883
Autenticación	allow_anonymous true
Función dentro del sistema	Intermediario de mensajes entre Python, Node-RED y ESP32
Dispositivos conectados	Pc (Python y Node-RED), ESP32
Tipo de red	Red local (LAN/Wi-Fi)

Nota. Fuente: El autor.

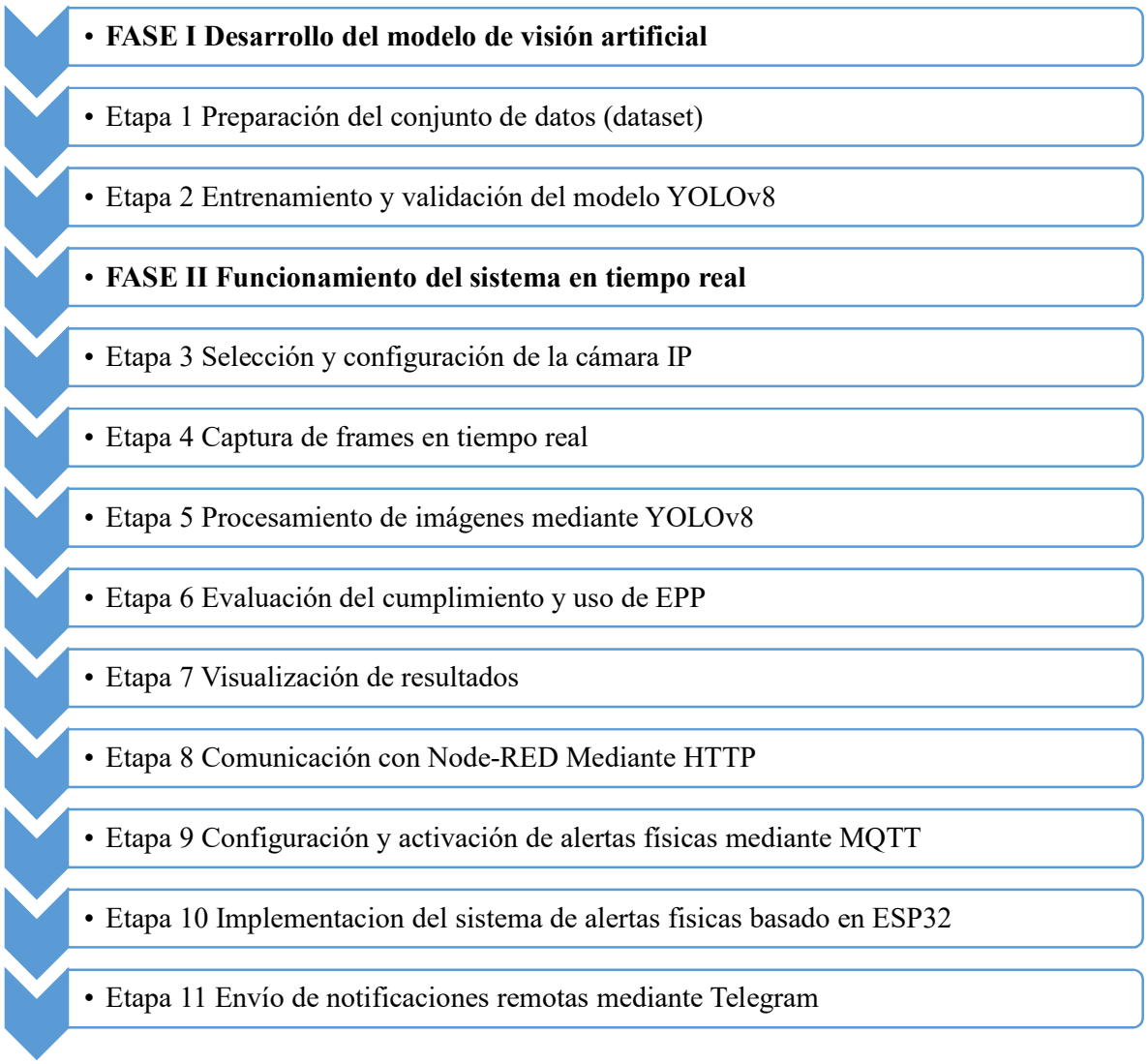
2.7.6 Fases y etapas del desarrollo y funcionamiento del sistema

El desarrollo del sistema propuesto se llevó a cabo de manera estructurada y se dividió en dos fases principales: (i) el desarrollo del modelo de visión artificial y (ii) la implementación y operación del sistema en tiempo real. Cada fase, junto con sus

respectivas etapas, debía cumplirse íntegramente antes de avanzar al siguiente procedimiento. A continuación, se describen los detalles de cada etapa, conforme se ilustra en la Figura 11.

Figura 11

Fases y etapas del desarrollo del sistema



Nota. Fuente: El autor.

2.7.7 Fase l: Desarrollo del modelo de visión artificial

Esta fase corresponde al proceso de preparación, entrenamiento y validación del modelo de detección de objetos en el sistema.

2.7.7.1 Etapa 1: Preparación del conjunto de datos (Dataset)

Con el objetivo de garantizar variabilidad y robustez en el modelo entrenado en esta etapa se realizó la recopilación de imágenes personalizadas correspondientes al entorno industrial, considerando diferentes condiciones de iluminación, posición de la persona, distancia de la cámara y escenarios de trabajo. Posteriormente, las imágenes fueron etiquetadas identificando la presencia y ausencia de los elementos de protección personal (casco, chaleco y guantes).

El conjunto de datos fue conformado a partir de múltiples fuentes, se realizaron capturas propias con el celular en entornos reales, así mismo imágenes obtenidas de repositorios públicos (dataset abiertos)

Para el proceso de etiquetado se utilizó la plataforma Labellmg, la cual facilita la anotación, organización y exportación de dataset para modelos de visión artificial, las imágenes fueron etiquetadas manualmente mediante cajas de limitación (bounding boxes), asignando una clase específica a cada objeto detectado, las anotaciones fueron exportadas en formato YOLO (.txt).

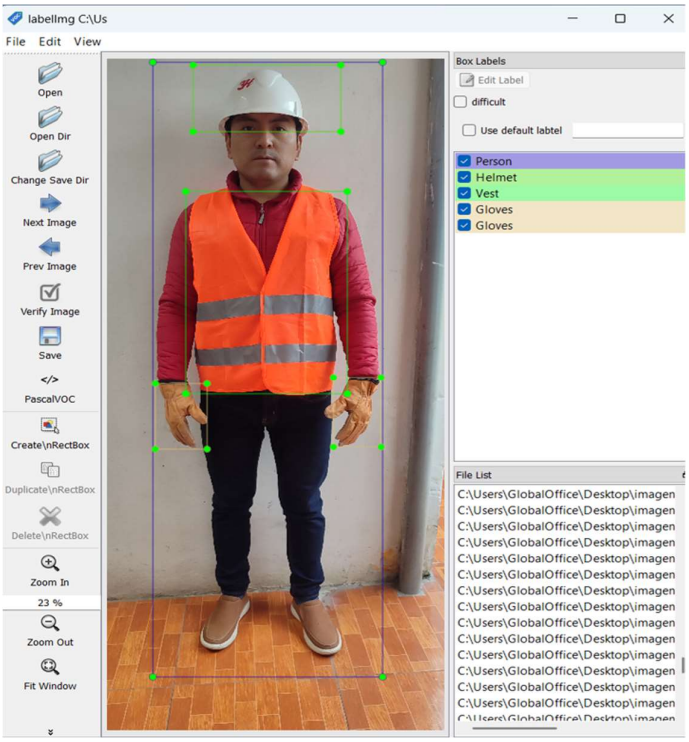
Las clases definidas para el proyecto fueron:

- Helmet (casco)
- Vest (chaleco)
- Gloves (guantes)

El proceso de etiquetado no modifica visualmente la imagen original, sino que genera archivos de texto asociados que contienen las coordenadas normalizadas de cada boulding boxes, las cuales son interpretadas por el modelo YOLO durante el entrenamiento, como se muestra en la Figura 12.

Figura 12

Labellmg repositorio para crear etiquetas e implementar modelos de visión artificial

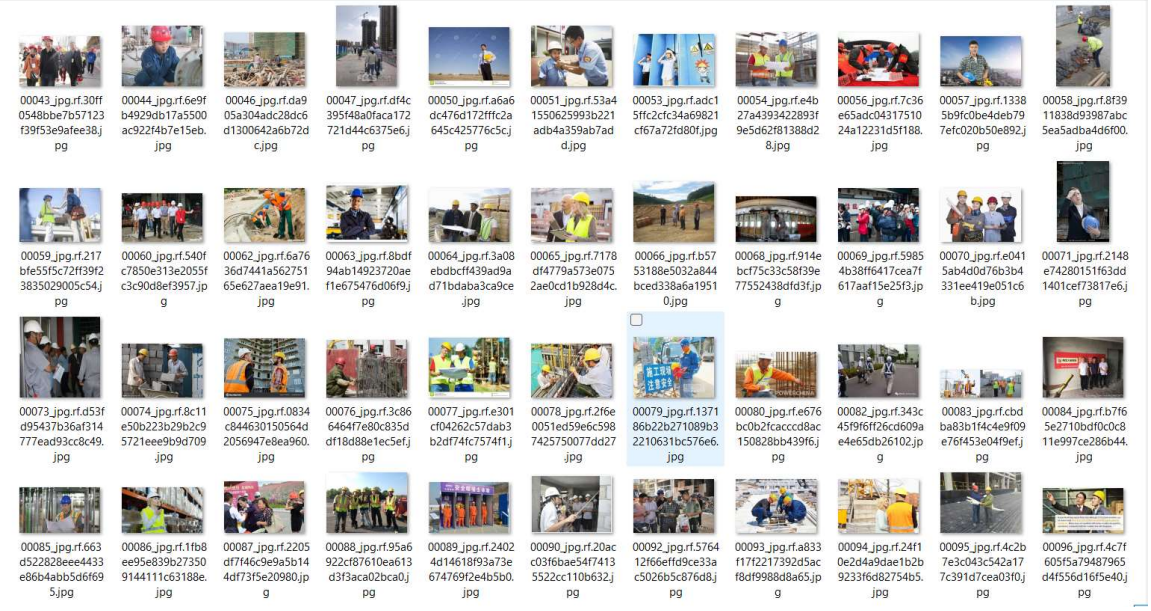


Nota. Fuente: Adaptado de la herramienta Labellmg, (p. 1)

En este trabajo no se emplearon técnicas de segmentación de imágenes, ya que el sistema se basa detección de objetos mediante el uso de cajas delimitadoras (bounding boxes). El modelo YOLO permite identificar directamente la presencia de elementos de protección personal dentro de la imagen, sin necesidad de segmentar la figura humana, así mismo se usaron imágenes obtenidas de repositorios públicos (dataset) como se puede apreciar en la Figura 13.

Figura 13

Imágenes etiquetadas



Nota. Fuente: Adaptado de roboflow universo,

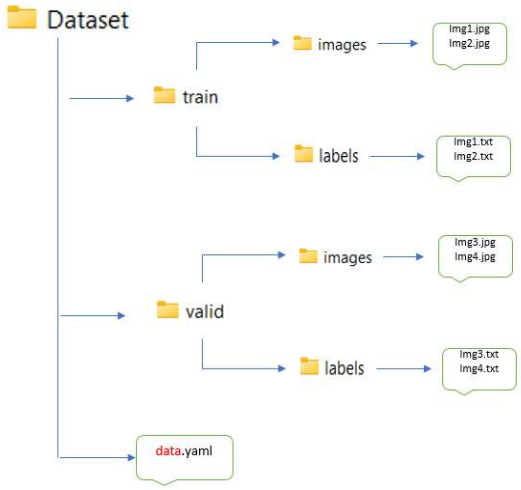
(P.1),https://universe.roboflow.com/search?q=epp+solo+casco+chaleco+y+guantes

El conjunto de datos fue organizado siguiendo la estructura requerida por el modelo YOLOv8, separando las imágenes y sus etiquetas correspondientes en subconjuntos de entrenamiento (train) y validación (val).

Las imágenes fueron divididas en conjuntos de entrenamiento y validación con el objetivo de evaluar el desempeño, el sistema se evalúa automáticamente en cada época usando el conjunto de validación, dicha distribución de carpetas la podemos observar en la Figura 14.

Figura 14

Estructura jerárquica del dataset en formato YOLO

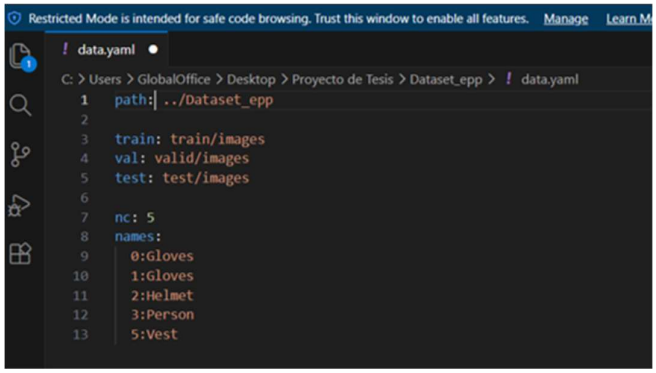


Nota. Fuente: El autor.

Es muy importante mencionar que YOLO no se guía por el nombre de las carpetas, sino por las rutas que sean puestas en data.yaml, se debe especificar las rutas de los subconjuntos de entrenamiento y validación, así como el número de clases y sus respectivos nombres, como se muestra en la Figura 15.

Figura 15

Definición de rutas y clases del dataset mediante el archivo data.yaml



Nota. Fuente: El autor.

2.7.7.2 Etapa 2 Entrenamiento y validación del modelo YOLOv8

Una vez concluida la preparación y organización del conjunto de datos, se procede al entrenamiento del modelo de detección de objetos, utilizando un enfoque de aprendizaje supervisado, dentro de un entorno virtual del lenguaje de programación Python. Durante esta etapa, se ajustaron los parámetros internos del modelo a partir de las imágenes previamente etiquetadas, con el objetivo de que la red neuronal aprendiera a identificar de manera automática el uso correcto e incorrecto de los EPP, para iniciar el proceso de entrenamiento, se empleó el comando estándar proporcionado por la librería Ultralytics YOLOv8, como se puede evidenciar en la Tabla 7.

Tabla 7

Comando estándar que especifica la tarea de detección

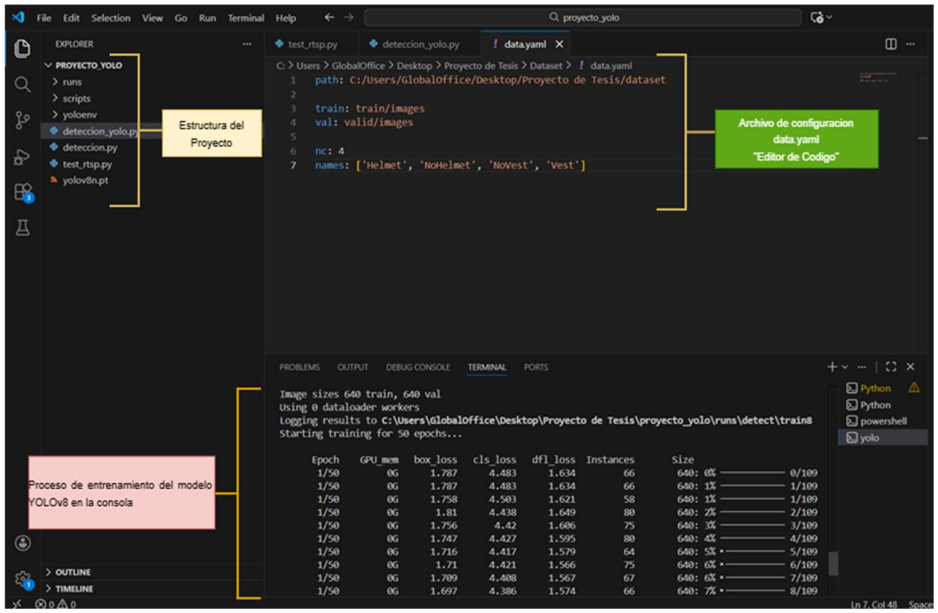
Descripción	Comando
Comando estándar proporcionado por la librería Ultralytics.	yolo task=detect mode=train model=yolov8n.pt data=data. yaml epochs=50 imgsz=640

Nota. Fuente: El autor.

Dicho comando se ejecuta en la consola Powershell desde la carpeta del proyecto, como se muestra en la Figura 16. A partir de las métricas observadas durante las primeras épocas del entrenamiento, se evidencia que el modelo inicio correctamente el proceso de aprendizaje. Las perdidas asociadas a la localización (box_loss y dfl_loss) y a la clasificación (cls_loss) muestran una tendencia decreciente, lo que indica una mejora progresiva en la detección, clasificación y confirman la correcta configuración del entrenamiento.

Figura 16

Proceso de entrenamiento en tiempo real del modelo YOLOv8 mediante consola



Nota. Fuente: El autor.

En la Tabla 8, se presenta métricas durante el proceso de entrenamiento del modelo YOLOv8, las cuales permiten analizar el comportamiento del modelo en las primeras etapas de aprendizaje.

Tabla 8

Métricas del proceso de entrenamiento del modelo YOLOv8

Epoch	Numero de épocas actual	Valores observados 1/50 inicio del entrenamiento.
GPU_mem	Memoria de la GPU	6 GB de memoria GPU
box_loss	Error en predicción de las cajas delimitadoras (bounding boxes)	1.78 – 1.68 El modelo empieza a aprender donde están ubicados los objetos
cls_loss	Error en la clasificación de las clases (Helmet, NoHelmet, Vest, etc.)	4.48-4.38 corresponde al error en la clasificación de las clases. El modelo aún está aprendiendo a

		diferenciar uso correcto vs incorrecto.
dfl_loss	Error relacionado con la precisión de la localización.	1.63-1.57 mejora gradual en la exactitud de detección.
Instances	Numero de objetos detectados batch	58-88 el dataset contiene múltiples instancias por imagen, adecuado para el entrenamiento.
size	Tamaño de entrada de las imágenes	640- las imágenes fueron redimensionadas a 640x640 pixeles, recomendado por YOLO.

Nota. Fuente: El autor.

Durante el proceso de entrenamiento, el modelo YOLOv8 ejecuta automáticamente un proceso de validación al finalizar cada época utilizando el conjunto de datos definido previamente en la carpeta validation. Esta carpeta es independiente del utilizado para el entrenamiento (training), esto permite evaluar el desempeño del modelo sobre datos no utilizados durante el aprendizaje en cada época del entrenamiento, el modelo genera predicciones sobre las imágenes del conjunto de validación y estas predicciones son comparadas con las etiquetas reales previamente definidas en el dataset, generando métricas de evaluación como precisión, recall, mAP95, loss, conforme se ilustra en la Figura 17. Estas métricas permiten analizar la evolución y desempeño del modelo a lo largo del proceso de entrenamiento, proporcionando información de la capacidad del modelo para detectar correctamente los EPP.

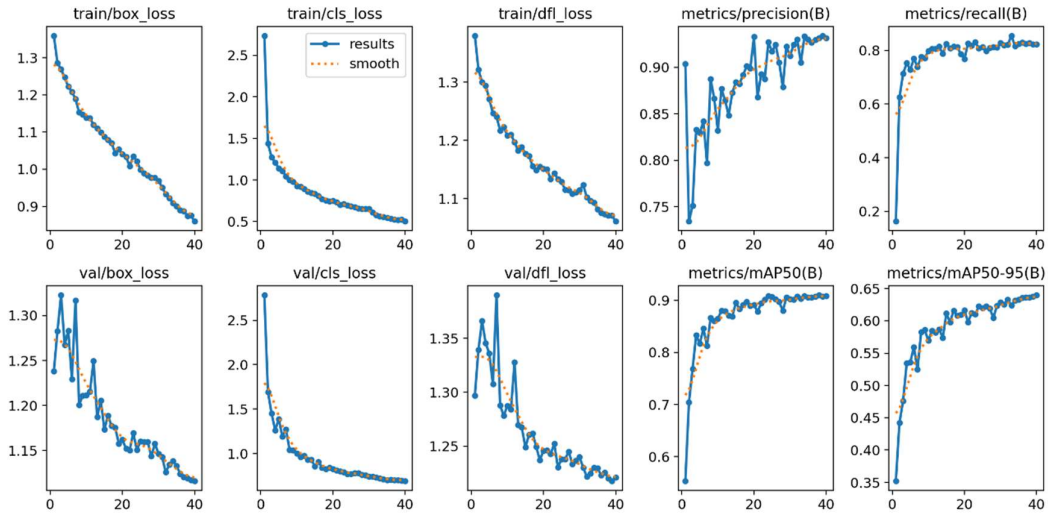
Además, YOLOv8 selecciona automáticamente el archivo de pesos correspondiente al mejor desempeño (best.pt), este archivo corresponde a la época con mayor rendimiento en el conjunto de validación. Este mecanismo permite garantizar que el modelo seleccionado para la etapa de inferencias sea aquel que presenta la mejor capacidad de generalización y detección de los equipos de protección personal en imágenes no vistas durante el entrenamiento.

Considerando que el dataset fue dividido previamente en conjuntos de training y validation, y que YOLOv8 realiza una evaluación automática al finalizar cada época de

entrenamiento, se obtiene una validación continua del desempeño del modelo sobre datos no utilizados durante el aprendizaje. Por esta razón, no se aplicó técnicas adicionales como K-Fold Cross Validation, ya que este método suele emplearse principalmente en dataset de menor tamaño o cuando no existe una división previa entre conjuntos de entrenamiento y validación.

Figura 17

Métricas de entrenamiento y validación del modelo YOLOv8 durante el proceso de aprendizaje



Nota. Fuente: Imagen asignada por el sistema de entrenamiento y validación YOLOv8.

La métrica mAP50-95 permite evaluar si el modelo detecta correctamente los elementos de protección personal y si las cajas generales coinciden con la ubicación real de los objetos en la imagen. A través de esta métrica se observó una mejora progresiva del desempeño durante el entrenamiento, hasta alcanzar valores estables, lo que indica que el modelo aprendió de manera adecuada y puede aplicarse correctamente a nuevas imágenes. 0.50 en mAP-95 es un resultado aceptable, la estabilidad de la curva nos indica que el modelo ya no mejora aunque sigan pasando las épocas, los pesos ya están ajustados de forma óptica, el modelo ya aprendió todo lo que podía aprender con estos datos, el eje x representa las épocas de entrenamiento, el eje y representa el valor de la métrica, sus valores van aproximadamente de 0 a 1, donde 0.50 significa que el modelo detecta con precisión y localización aceptable, en el proceso de entrenamiento del modelo también se analizan las funciones de perdida (loss), las cuales representan una medida de error que

comete el modelo al realizar predicciones, la perdida indica que tan alejada esta la predicción del resultado real, por lo tanto valores más bajos de perdidas significa que el modelo está aprendiendo de forma más precisa.

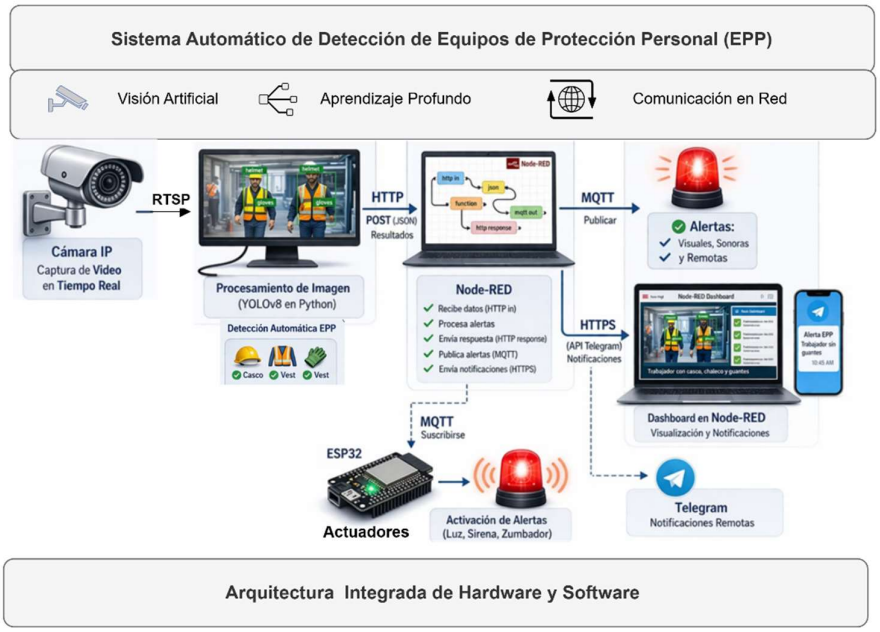
2.7.8 Fase II: Funcionamiento del sistema en tiempo real

En esta fase se describe el funcionamiento del sistema de detección de elementos de protección personal (EPP) en tiempo real. La arquitectura general del sistema se presenta en la Figura 18, donde se muestra el flujo completo de información desde la captura de imágenes hasta la generación de alertas.

El sistema inicia con la adquisición de imágenes mediante una cámara IP, las cuales son transmitidas al módulo de procesamiento. Posteriormente, dichas imágenes son analizadas por el modelo de visión artificial YOLOv8, encargado de detectar y evaluar el uso o no uso de los EPP. Los resultados obtenidos son enviados a través de los protocolos de comunicación MQTT y HTTP hacia Node-RED, donde se gestiona la visualización de resultados, el registro de eventos y la activación de alertas visuales, sonoras y notificaciones móviles, permitiendo una supervisión efectiva en tiempo real.

Figura 18

Arquitectura general del sistema de detección de EPP en tiempo real mediante visión artificial

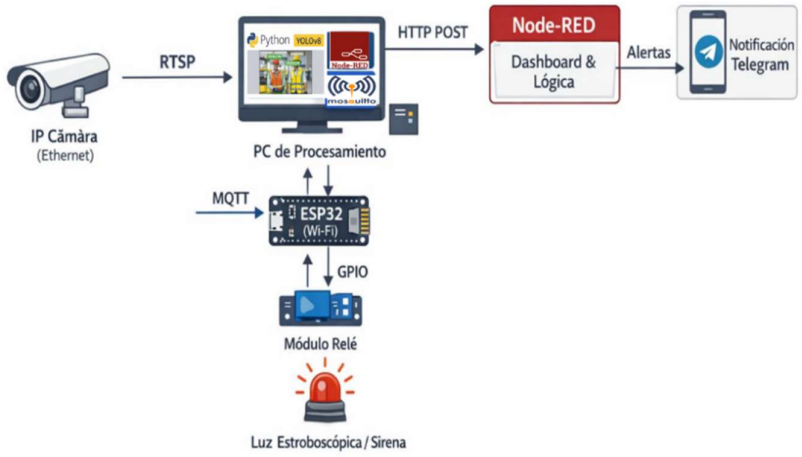


Nota. Fuente: El autor.

La Figura 19, muestra el diagrama de bloques del sistema de detección, en este diagrama se representa el flujo secuencial de información desde la adquisición de imágenes hasta la generación de alertas, integrando los distintos módulos de hardware y software que conforman el sistema, a partir de este diagrama, el funcionamiento del sistema se describe de manera detallada a través de la siguiente etapa.

Figura 19

Diagrama de bloques y funcionamiento del sistema en tiempo real



Nota. Fuente: El autor.

2.7.8.1 Etapa 3: Selección y configuración de la cámara IP

Camara industrial con conectividad Ethernet mostrada en la Figura 20, fue configurada para transmitir el flujo de video mediante el protocolo RTSP (Real Time Streaming Protocol), se asignó una dirección IP fija lo que permite conexión constante con el computador de procesamiento, el acceso al video se realizó a través de una URL RTSP, utilizada por la aplicación desarrollada en Python para establecer la conexión y recibir el flujo de imágenes de manera continua.

Figura 20

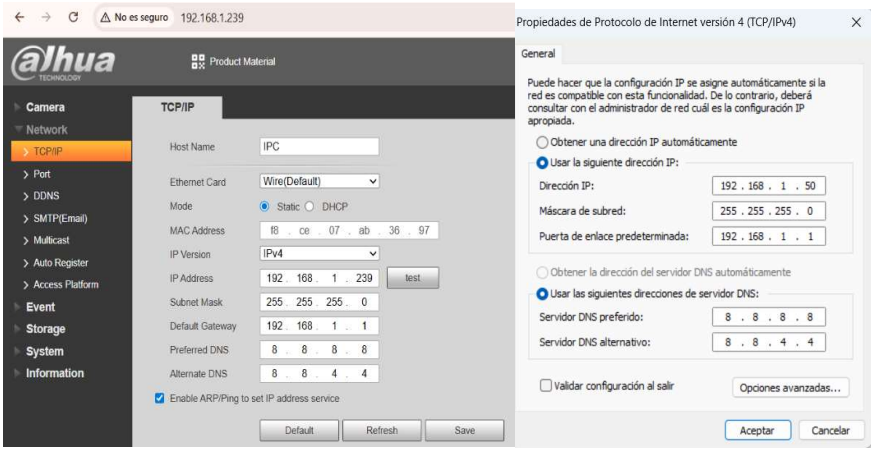
Camara IP Dahua para adquisición de video

Nota. Fuente: El autor

Para asegurar una comunicación estable entre la cámara y el computador de procesamiento se asignaron IP fijas a ambos dispositivos dentro de la misma red local como se muestra en la Figura 21, la cual presenta la configuración de red realizada, donde se observa la asignación de direcciones IP tanto en la cámara como en el equipo de procesamiento.

Figura 21

Configuración de red para la comunicación cámara-PC



Nota. Fuente: El autor.

2.7.8.2 Etapa 4: Captura de imágenes en tiempo real mediante Python y OpenCV

En esta etapa se realizó la captura de imágenes en tiempo real a partir del flujo de video trasmitido por la cámara IP. Para ello, se desarrollarlo una aplicación en el lenguaje de programación Python utilizando la librería OpenCV, la cual permite el acceso, procesamiento y visualización de secuencias de video.

La conexión con la cámara IP se estableció mediante una URL RTSP, como se puede observar en la Figura 22, la cual contiene dirección IP de la cámara, el puerto de transmisión, y las credenciales de acceso. Esta URL fue utilizada por OpenCV para abrir el flujo de video y obtener continuamente las imágenes que conforman la secuencia visual.

Figura 22

Configuración de la URL RTSP de la cámara IP y rutas del sistema

```
# ================ CONFIGURACIÓN ================
RTSP_URL = "rtsp://admin:Prosetel2025!@192.168.1.239:554/cam/realmonitor?channel=1&subtype=1"
NODE_RED_URL = "http://127.0.0.1:1882/alerta_epp"
MODEL_PATH = r"C:\Users\GlobalOffice\Desktop\Proyecto de Tesis\proyecto_yolo\runs\detect\train8\weights\best.pt"
```

Nota. Fuente: El autor.

Figura 23

Código de inicialización y verificación de la captura de video RTPS mediante OpenCV

```
# ================ CÁMARA ================
cap = cv2.VideoCapture(RTSP_URL, cv2.CAP_FFMPEG)
cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)

if not cap.isOpened():
    print("❌ No se pudo abrir la cámara RTSP")
    exit()

print("📷 Cámara conectada")
print("🟢 Sistema EPP en ejecución\n")
```

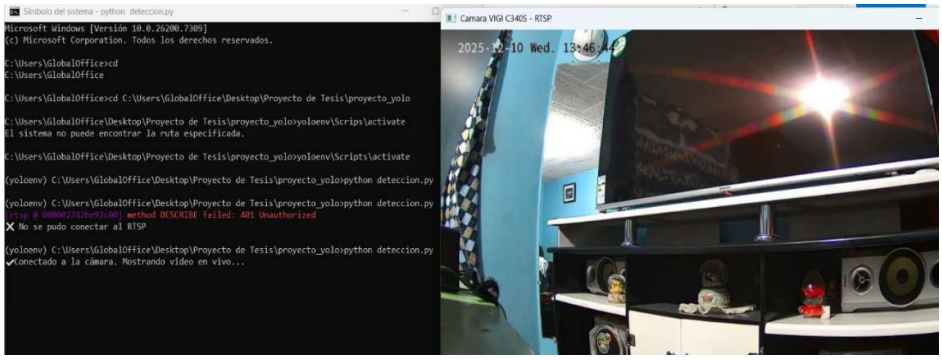
Nota. Fuente: El autor.

Se puede apreciar en la Figura 24, la captura de imágenes en tiempo real desde la cámara IP mediante el protocolo RTSP, constituyendo la entrada visual utilizada posteriormente para el procesamiento y análisis mediante el modelo de detección de equipos de protección personal.

- VideoCapture = abre el flujo RTSP
- CAP_FFMPEG = asegura compatibilidad con RTSP
- BUFFERSIZE = reduce latencia
- imshow () muestra las imágenes

Figura 24

Captura de imágenes en tiempo real desde la cámara



Nota. Fuente: El autor.

2.7.8.3 Etapa 5: Procesamiento de imágenes mediante YOLOv8

En esta etapa se describe el procesamiento de las imágenes capturadas en tiempo real mediante la aplicación del modelo de detección de objetos YOLOv8. El objetivo de esta fase es analizar cada frame proveniente de la cámara para identificar la presencia y uso adecuado de los EPP, permitiendo una evaluación automática y continua del cumplimiento de las normas de seguridad en el entorno industrial.

Una vez capturado el frame estos son enviados al modelo previamente entrenado, el modelo realiza inferencias sobre cada imagen, identificando objetos de interés mediante redes neuronales convolucionales, generando como salidas las clases detectadas, la probabilidad de confianza y las coordenadas de los cuadros delimitadores (bounding boxes). Parte del código de procesamiento se puede observar en la Figura 25.

Figura 25

Parte del código que se utiliza para procesamiento

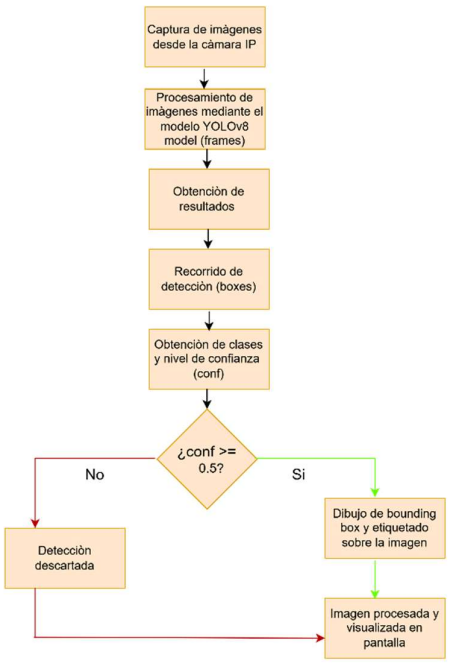
```
results = model(frame, conf=CONF_TH, verbose=False)[0]

for box in results.boxes:
    cls_id = int(box.cls[0])
    conf = float(box.conf[0])
    label = model.names[cls_id]
    x1, y1, x2, y2 = map(int, box.xyxy[0])

    color = VERDE

    if label == "Helmet":
        casco = "SI"
    elif label == "NoHelmet":
        casco = "NO"
        color = ROJO
        alerta_frame = True
```

Nota. Fuente: El autor.

El procesamiento se ejecuta en tiempo real, lo que permite analizar continuamente la escena sin necesidad de almacenamiento previo de las imágenes. Cada detección es clasificada según el tipo de elemento de protección personal presente o ausente, y se asigna un color distinto a los cuadros delimitadores para facilitar la interpretación visual de los resultados, el diagrama de flujo se puede observar en la Figura 26.

Figura 26

Diagrama de flujo del procesamiento de imágenes con YOLOv8



Nota: Fuente: El autor.

En la Figura 27 se presenta un ejemplo de la detección realizada por el sistema en tiempo real. En la imagen se observan los bounding boxes generados por el modelo YOLOv8, junto con la etiqueta de la clase detectada y su respectivo nivel de confianza que corresponden a:

- Person 82.3 %
- NoHelmet 87.4 % el modelo detecta no casco con 87 % de confianza
- Vest 91.9%, el modelo detecta un chaleco con 91.9% de confianza
- Cajas verdes, indica cumplimiento de EPP
- Caja roja, indica incumplimiento de EPP

Figura 27

Resultado del procesamiento y detección de EPP

Nota. Fuente: El autor.

2.7.8.4 Etapa 6: Evaluación del cumplimiento y uso de EPP

En esta etapa se realiza la evaluación lógica del cumplimiento del uso de los EPP a partir de los resultados generados por el modelo de detección, el objetivo es determinar, para cada imagen procesada, si el trabajador cumple o no con las normas de seguridad establecidas, considerando la presencia y confiabilidad de los EPP.

Esta evaluación constituye un punto clave del sistema, ya que trasforma las decisiones visuales en decisiones de seguridad, permitiendo posteriormente la generación de alertas y acciones correctivas.

Parte de la lógica de evaluación implementada en el sistema se observa en la Figura 28, en este fragmento de código se realiza el análisis de las detecciones generadas por el modelo YOLOv8, donde el modelo identifica las clases detectadas y evalúa si el trabajador cumple o no con los EPP. Además, se establecen condiciones para generar alertas y modificar el color de las cajas delimitadoras en función del cumplimiento o incumplimiento del uso de EPP.

Figura 28

Detección y código para la evaluar lógica del cumplimiento

Nota. Fuente: El autor.

Para cada objeto detectado, se evalúa la clase identificada (casco, chaleco, guantes) y se determina si el uso es correcto o incorrecto. En caso de detectarse la ausencia de algún EPP, se activa una condición de alerta.

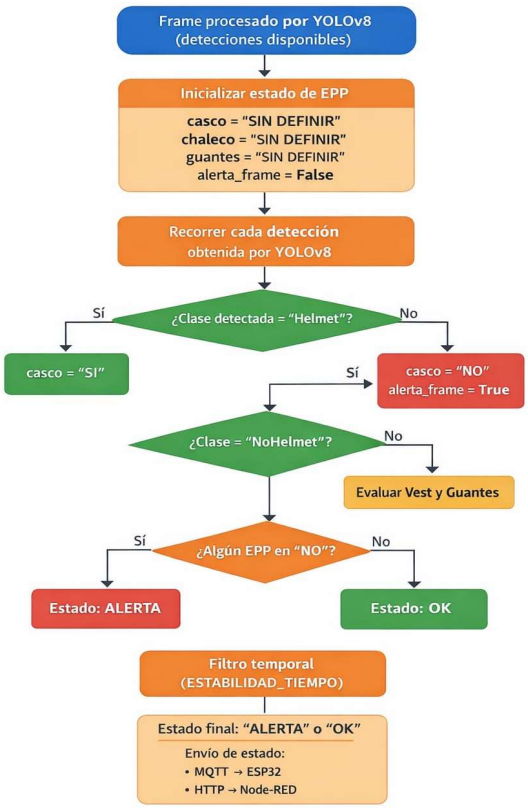
Posteriormente, se aplica un filtro temporal de estabilidad con el fin de enviar falsas alarmas producidas por detecciones momentáneas.

Finalmente, el sistema establece un estado de operación, clasificándolo como “OK” o “ALERTA”, el cual es enviado mediante protocolos MQTT y HTTP hacia los módulos de control y visualización, permitiendo la activación de alertas físicas y el registro del evento en el sistema.

El proceso de evaluación del cumplimiento y uso de EPP se basa en el análisis de las clases detectadas por el modelo YOLOv8. El flujo de evaluación del sistema se muestra en la Figura 29.

Figura 29

Diagrama de flujo de la evaluación del cumplimiento del uso de EPP

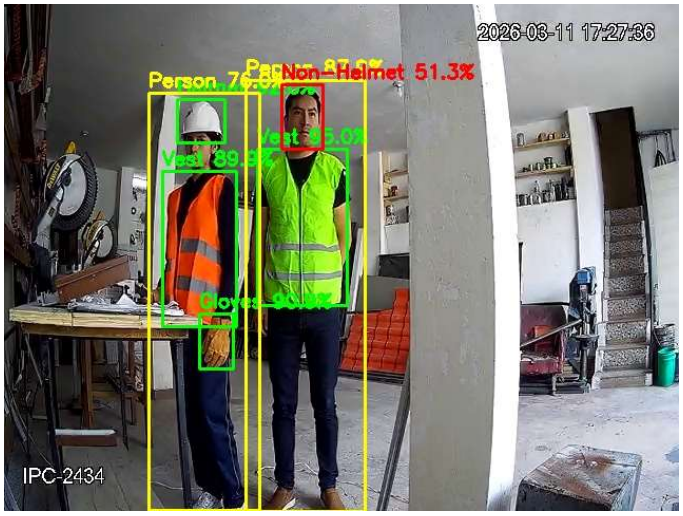


Nota. Fuente: El autor.

Como resultado de la evaluación del cumplimiento del uso de los EPP, el sistema analiza las clases detectadas por el modelo y determina el estado de cada EPP, cuando se identifica un incumplimiento, el sistema resalta visualmente la detección mediante un bouding box de color rojo y activa el estado de alerta correspondiente, el resultado se puede observar en la Figura 30.

Figura 30

Resultado de la evaluación del cumplimiento de EPP



Nota. Fuente: El autor.

2.7.8.5 Etapa 7: Visualización de resultados

Durante la ejecución del sistema, la visualización incluye los siguientes elementos:

Bouding boxes, cada objeto detectado es resaltado mediante un rectángulo delimitador (bounding box), el cual encierra la región donde el modelo YOLOv8 ha identificado un elemento de interés.

- Color verde: indica cumplimiento del EPP (por ejemplo, Helmet, Vest, Gloves).
- Color rojo: indica incumplimiento del EPP(por ejemplo, NoHelmet, NoVest, NoGloves).

- Este criterio visual permite identificar rápidamente situaciones seguras e inseguras dentro de la escena.
- Sobre cada Boulding box se muestra una etiqueta que contiene, el nombre de la clase detectada, el nivel de confianza asociado a la detección,
  NoHelmet: 59.4% Vest: 88.4 %, un ejemplo de la detección de incumplimiento de EPP en tiempo real se visualiza en la Figura 31.

Figura 31

Resultado visual del sistema en tiempo real donde se identifica incumplimiento

Nota. Fuente: El autor.

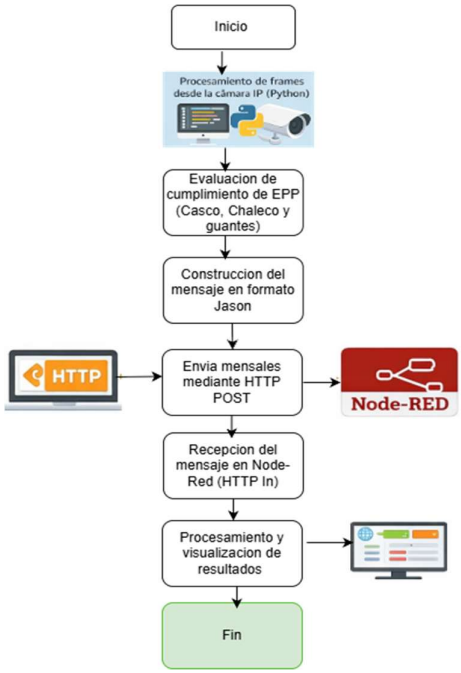
2.7.8.6 Etapa 8: Comunicación con Node-RED mediante HTTP

En esta etapa se implementa el mecanismo de comunicación entre el sistema de visión artificial desarrollado en Python y la plataforma Node-RED, utilizando el protocolo HTTP. El objetivo principal de esta comunicación es enviar el estado del uso de los EPP, así como alertas visuales acompañadas de evidencias gráficas, para su posterior visualización y gestión en un dashboard o sistema de notificaciones, así mismo se almacenarán las evidencias en una carpeta interna del proyecto.

Node-RED actúa como un servidor intermedio, permitiendo recibir los datos procesados por el modelo YOLOv8 y distribuirlos hacia interfaces gráficas, sistemas de alerta o dispositivos externos. El flujo de comunicación se puede apreciar en la Figura 32.

Figura 32

Diagrama de flujo de la arquitectura de comunicación entre Python y Node-RED mediante HTTP



Nota. Fuente: El autor.

Una vez realizada la evaluación, el sistema construye un mensaje estructurado en formato JSON, el cual contiene el estado de cada EPP y el nivel de confianza asociado a la detección, este mensaje se envía mediante una solicitud HTTP utilizando el método POST hacia un endpoint configurado en Node-RED

Node-RED recibe la información a través de un nodo HTTP In donde los datos son procesados, visualizados en un panel de control o utilizados para genera alertas adicionales, registros o acciones externas, los nodos utilizados son:

- HTTP In
- JSON
- Debug

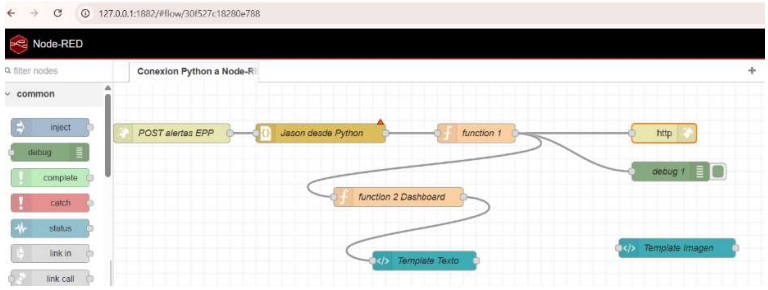
- HTTP Response

Nodo HTTP In recibe las peticiones enviadas desde el sistema Python, Method:POST, URL:/alertas_epp, esto genera el endpoint completo: htpp://127.0.0.1:1882/alerta_epp.

La estructura del flujo utilizada para recibir y procesar la información enviada desde Python hacia Node-RED se muestra en la Figura 33.

Figura 33

Configuración para comunicación entre Python y Node-RED

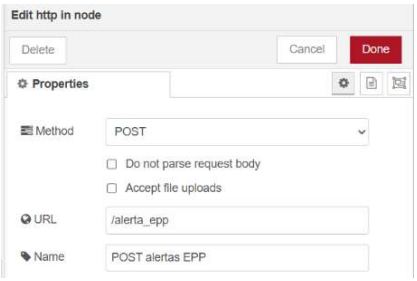


Nota. Fuente: El autor.

El nodo HTTP In de Node-RED actúa como endpoint del sistema obteniendo los datos enviados desde el script que fue desarrollado en Python. Fue configurado utilizando el método Post de HTTP, el cual permite recibir información enviada desde el sistema de visión artificial. La URL configurada cono /alertas_epp define el endpoint al cual el sistema envía las solicitudes utilizando el protocolo de comunicación HTTP. La representación de la configuración la podemos observar en la Figura 34.

Figura 34

Configuración del Nodo HTTP IN



Nota. Fuente: El autor.

Los parámetros utilizados para la configuración del nodo HTTP In en Node-RED se presentan en la Tabla 9.

Tabla 9

Lista técnica de configuración Nodo HTTP In

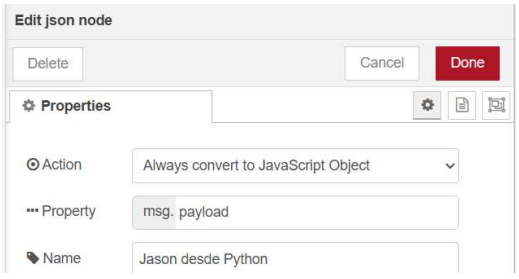
Método	POST
URL (Endpoint): NODE_RED_URL=	http://127.0.0.1:1822/alertas_epp
Función:	Recepción de datos del sistema de visión artificial
Formato de datos:	Payload es el JSON enviado
Origen de la solicitud:	Script Python (YOLOv8)

Nota. Fuente: El autor

Se implementó un nodo JSON el cual cumple la función de interpretar y transformar los datos recibidos desde el script de Python, dicho nodo se encarga de convertir la información a un objeto JavaScript facilitando una estructura de datos accesibles (msg.payload) para el funcionamiento de los nodos posteriores, como funciones lógicas, visualización en el dashboard, generación de alertas y comunicación mediante MQTT. La configuración del nodo se observa en la Figura 35.

Figura 35

Configuración nodo JSON para interpretación de datos

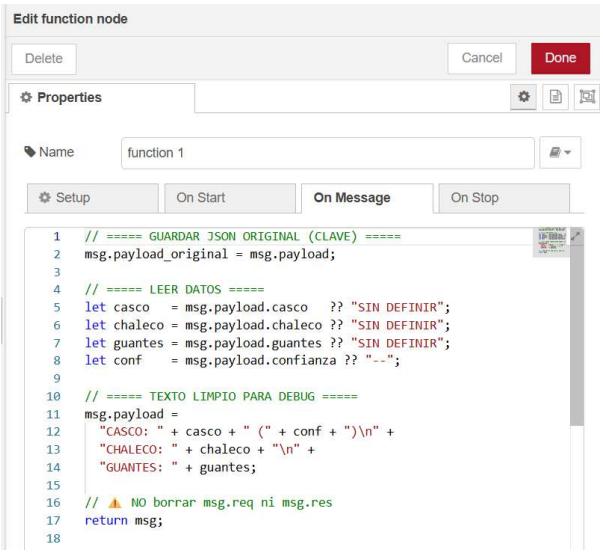


Nota. Fuente: El autor**.**

Después de haber convertido los datos a un objeto JavaScript a través del nodo JSON se implementó un nodo Function para realizar el procesamiento lógico de los datos recibidos desde el script de Python, como se observa en la Figura 36. En dicho nodo se implementa la lógica de decisión del sistema, analizando el contenido del Payload, que incluye la información sobre los estados del uso de EPP, como casco, chaleco y guantes, además del estado general del sistema (OK o ALERTA). Utilizando esa información el nodo Function determina las acciones a ejecutar, tales como generar mensajes de alerta o enviar datos al dashboard para su visualización en tiempo real, o activar notificaciones externas,

Figura 36

Nodo Function se encarga de realizar procesamiento lógico



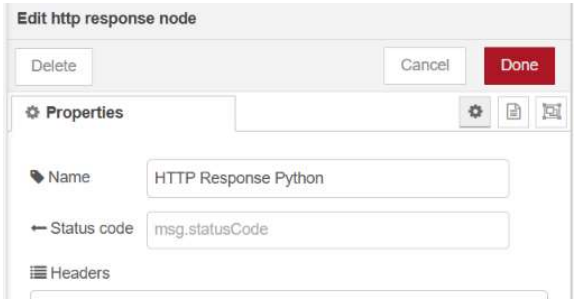
Nota. Fuente: El autor.

Para cerrar y confirmar la comunicación entre Python y Node-RED se utilizó el nodo HTTP Response, el cual es el encargado de responder oficialmente a la solicitud que fue recibida previamente por el nodo HTTP In. El flujo completo consiste de la siguiente manera: Python detecta una condición como es en uso o no uso de EPP, Python envía datos mediante HTTP In, Node-RED HTTP In recibe la solicitud, Node-RED procesa los datos (JSON, funciones, lógica) HTTP Response envía la respuesta de vuelta al cliente Python indicando que la solicitud fue exitosa, fallida o mal formada, sin este nodo la conexión HTTP quedaría abierta y por lo tanto Python podría lanzar errores de timeout

y la arquitectura HTTP quedaría mal estructurada. La configuración de nodo se puede observar en la Figura 37.

Figura 37

Nodo HTTP Response

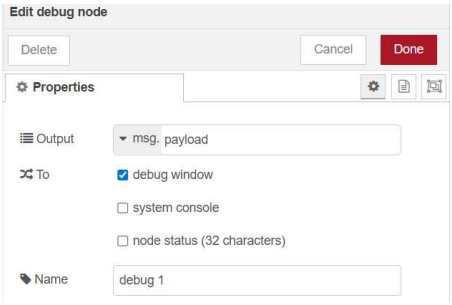


Nota. Fuente: El autor.

Para monitorear y verificar en tiempo real los estados del sistema que circulan dentro del flujo de Node-RED se implementó un nodo Debug, su función es visualizar datos, muestra el contenido del mensaje recibido (Payload) es decir es una herramienta de validación y diagnostico que confirma los datos de detección enviados desde Python, facilitando el monitoreo del estado del sistema, valores de confianza del modelo de visión artificial y la correcta ejecución de la lógica implementada. Dicha configuración podemos observar en la Figura 38.

Figura 38

Nodo Debug para visualizar datos

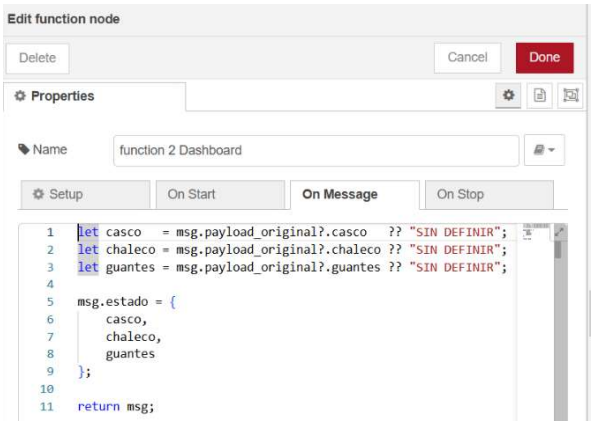


Nota. Fuente: El autor.

Se implementó el nodo Function para el procesamiento lógico y adaptación de los datos, funciona como intermediario entre el modelo de visión artificial y la interfaz gráfica tomando los datos enviados desde Python, este nodo convierte datos técnicos a información interpretable, prepara los estados en tiempo real de uso de los EPP (casco, chaleco, guantes) para ser visualizados en el dashboard de Node-RED. Dicha configuración se muestra en la Figura 39.

Figura 39

Nodo Fuction dashboard encargado del procesamiento lógico de EPP para su visualización en Dashboard

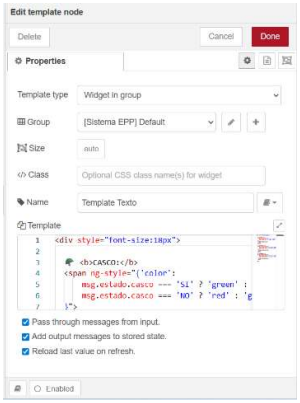


Nota. Fuente: El autor.

Se implementó el nodo Template para visualización dinámica de los resultados dentro del dashboard de Nore-RED, este nodo utiliza HTML para presentar visualmente estados del uso de EPP de forma clara y en tiempo real para el usuario. Dicha configuración se muestra en la Figura 40.

Figura 40

Nodo template utilizado para la visualización del estado de EPP

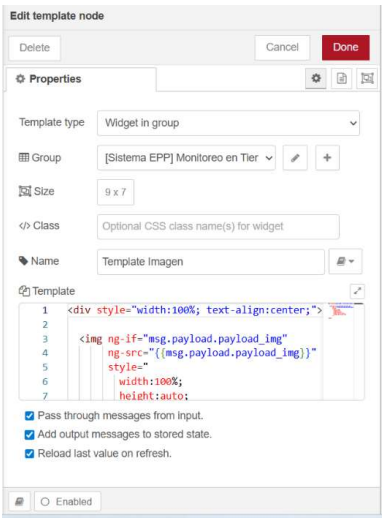


Nota. Fuente: El autor.

Se implementó un nuevo nodo Template imagen, el cual muestra en el dashboard la imagen procesada por Yolo en tiempo real, la imagen muestra los estados de los EPP incluida los bouding boxes de casco, chaleco, guantes y sus porcentajes, dicha configuración se muestra en la Figura 41.

Figura 41

Nodo template muestra imagen en tiempo real

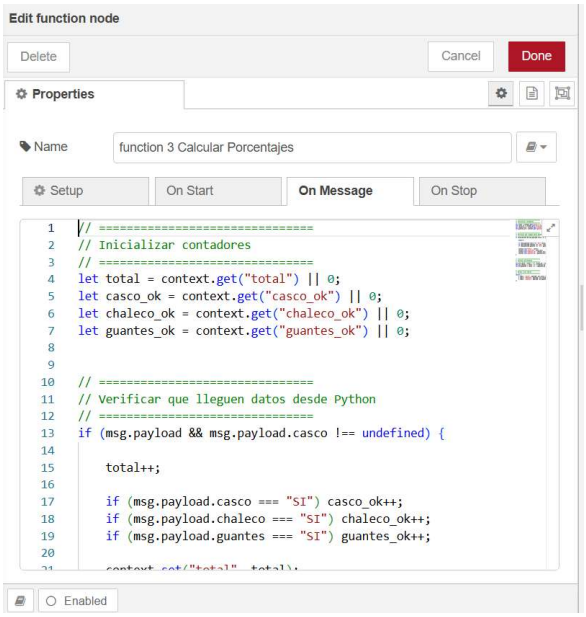


Note. Fuente: El autor.

Se implementó un nodo Fuction el cual transforma las detecciones individuales generadas por el modelo YOLOV8 en indicadores estadísticos, permite el cálculo dinámico del cumplimiento en tiempo real. Dicha configuración se observa en la Figura 42.

Figura 42

Nodo fuction para calcular porcentajes de cumplimiento de EPP

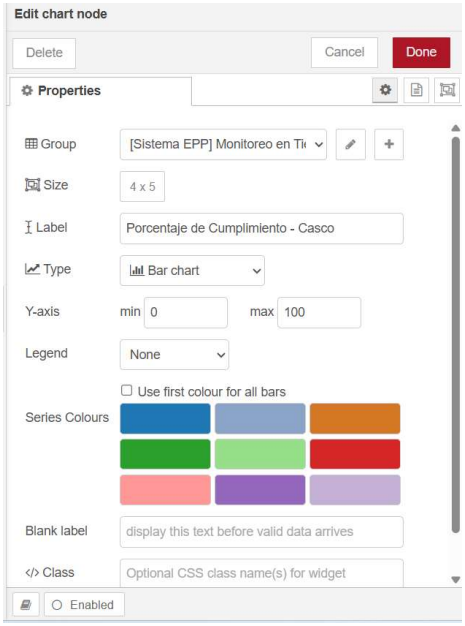


Nota. Fuente: El autor.

Se implementó el nodo Chart el cual muestra visualmente los porcentajes de cumplimiento de los EPP en forma gráfica de barras, el modelo YOLOv8 detecta cumplimento SI/NO. El nodo Fuction calcula el porcentaje mientras que el nodo chart recibe ese porcentaje y lo dibuja en una barra de 0 a 100. Facilitando la visualización estadística de los indicadores generados por el sistema de detección en tiempo real. Dicha configuración es mostrada en la Figura 43.

Figura 43

Nodo Chart muestra visualmente el porcentaje de cumplimiento



Nota. Fuente: El autor.

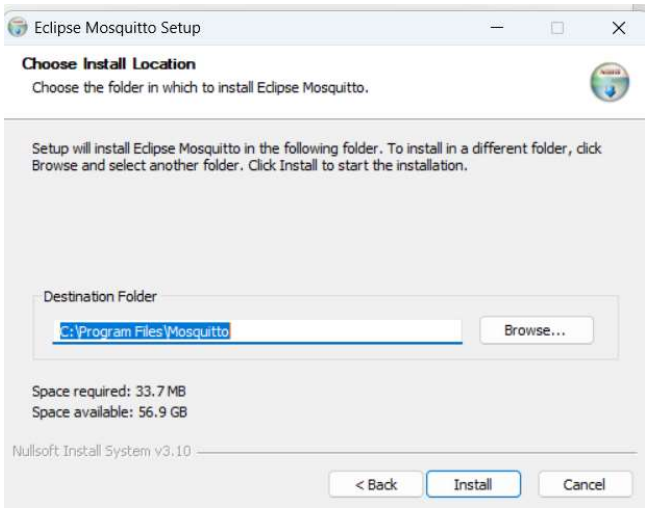
2.7.8.7 Etapa 9: Configuración y activación de alertas físicas mediante MQTT

Configuración del Broker mosquito

Consiste en configurar el protocolo de comunicación MQTT que permite transmitir información de alertas hacia el dispositivo físico de notificaciones, para ello se instaló el broker MQTT (Mosquito) Eclipse Mosquito es un bróker MQTT de open-source el cual implementa el protocolo de comunicación MQTT. Mosquito actúa como un servidor intermediario que recibe mensajes y los distribuye a los clientes suscritos, solo enruta datos, no tiene interfaz gráfica, no muestra mensajes, una vez descargado y descomprimido el software se selecciona el archivo ejecutable, posteriormente aparece la pantalla principal de instalación como podemos observar en la Figura 44.

Figura 44

Pantalla principal de instalación Eclipse Mosquito

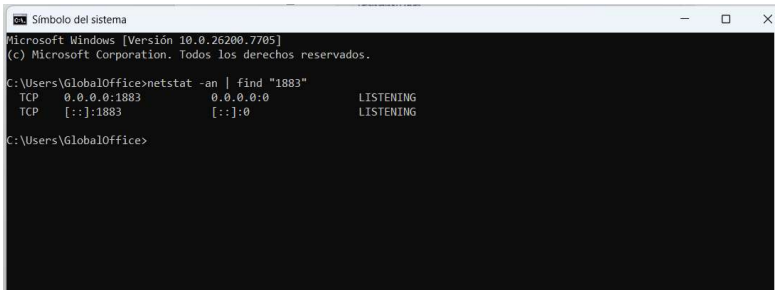


Nota. Fuete: El autor.

Una vez que el bróker mosquito está instalado, se debe verificar que utiliza el puerto 1883 el cual corresponde al puerto estándar asignado oficialmente al protocolo MQTT. Este puerto se configura automáticamente al instalar el bróker, permitiendo así que los clientes MQTT (ESP32 y Node-RED) se puedan conectar mediante TCP/IP dentro de la red local. El correcto funcionamiento del bróker mosquito se muestra en la Figura 45.

Figura 45

Se verifica el correcto funcionamiento del Broker MQTT



Nota. Fuente: El autor.

Los componentes y parámetros utilizados en la arquitectura de comunicación MQTT del sistema se presentan en la Tabla 10.

Tabla 10

Componentes y parámetros arquitectura MQTT

Arquitectura MQTT del Sistema	
Broker mosquito	Localhost-puerto 1883
Cliente publicador	Node-RED
Cliente suscritor	ESP32
Cliente de prueba	MQTTX (solo para monitorear y visualizar)
Modelo usado	Publish/Subscribe
Protocolo	MQTT sobre TCP/IP
Puerto	1883
IP usada	127.0.0.1 (bróker local)

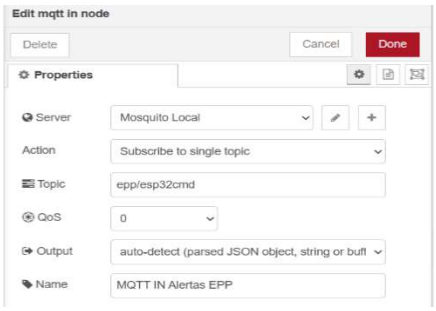
Nota. Fuente: El autor.

- **Suscripción a topics (MQTT IN)**

Se implementó el nodo MQTT IN el cual permite la suscripción al topic epp/esp32cmd. Mediante esta suscripción, Node-RED puede recibir mensajes publicados en el bróker Mosquito, permitiendo así la conexión entre el sistema de monitoreo y el dispositivo ESP32 a través del protocolo de comunicación MQTT sobre TCP/IP. Dicha configuración se la puede observar en la Figura 46.

Figura 46

Nodo MQTT In el cual permite la suscripción al topic



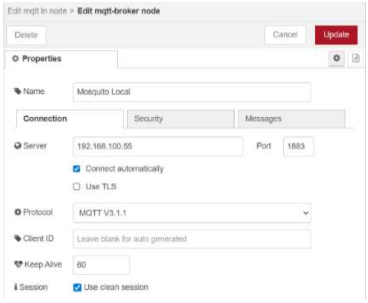
Nota. Fuente: El autor.

- **Configuración del cliente MQTT en Node-RED**

Para establecer la comunicación entre Node-RED y el bróker Mosquito Local mediante el protocolo de comunicación MQTT. Se define al servidor con la dirección IP 192.168.100.55 que corresponde al equipo donde se encuentra instalado el bróker Mosquito dentro de la red LAN. Se utilizo el puerto 1883 que es el puerto estándar asignado oficialmente al protocolo MQTT (sobre TCP/IP), esta configuración permite que Node-RED actúe como cliente MQTT capaz de publicar y suscribirse a topics dentro del sistema. Dicha configuración es mostrada en la Figura 47.

Figura 47

Conexión entre Node-RED y el broker Mosquito



Nota. Fuente: El autor.

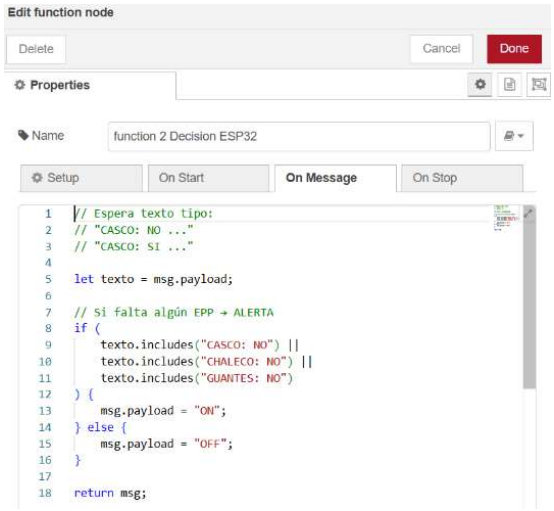
- **Lógica de decisión para control de dispositivo (Fuction ESP32)**

Se implementó el nodo Fuction el cual permite cumplir la lógica de decisión para activar alertas físicas en el sistema, consiste en analizar resultados enviados por el sistema de visión artificial (YOLO + Python), verifica si algún elemento de protección personal (casco, chaleco, guantes) presenta estado de incumplimiento (“NO”).

En caso de presentarse el incumplimiento de al menos un EPP, el nodo Fuction genera un mensaje con valor “ON”, el mismo que es enviado mediante el protocolo de comunicación MQTT hacia el broker mosquito para su posterior trasmisión al ESP32. Si el cumplimiento es adecuado de todos los EPP el nodo envía el valor “OFF”, desactivando el sistema de alertas, garantizando una respuesta inmediata ante incumplimientos de seguridad industrial. Dicha configuración se observa en la figura 48.

Figura 48

Nodo Fuction contiene lógica de decisión para dispositivo ESP32



Nota. Fuente: El autor.

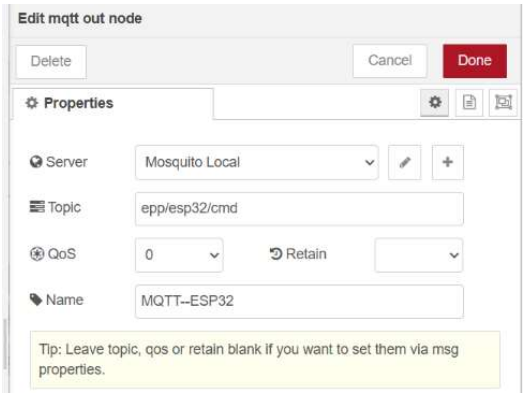
- **Publicación de comandos hacia el ESP32 (MQTT OUT)**

Se implementó el nodo MQTT OUT su función es publicar los comandos de activación y desactivación del sistema de alertas físicas hacia el broker Mosquito, el broker Mosquito

recibe el mensaje y lo redistribuye a todos los clientes suscritos al topic correspondiente, en este caso envía hacia el ESP23 el cual está suscrito al topic epp/esp32/cmd. Es el último paso de software antes que la orden llegue hacia el hardware. Este nodo actúa como cliente MQTT publicador, el mismo que envía el valor lógico generado por el Nodo de decisión (“ON” u “OFF”) dicha suscripción es mostrada en la Figura 49.

Figura 49

Nodo MQTT OUT para publicar mensaje de activación o desactivación

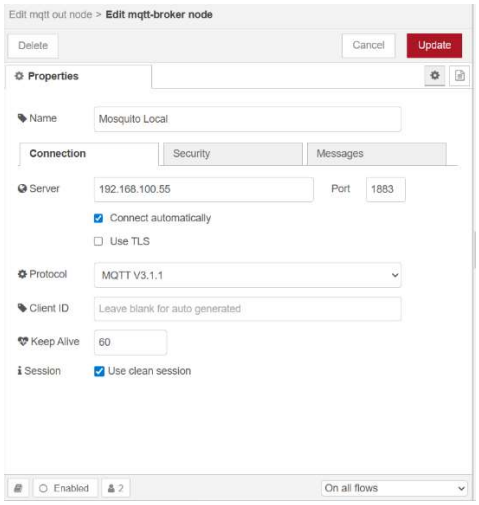


Nota. Fuente: El autor.

Para establecer la comunicación con el broker Mosquito se utilizó la dirección IP 192.168.100.55 correspondiente al equipo servidor, utilizando el puerto estándar 1883 asignado al protocolo MQTT, se seleccionó la version MQTT 3.1.1 por su compatibilidad con el dispositivo ESP32 y el broker mosquito, esta configuración permite que Node-RED actúe como cliente MQTT tanto para la publicación como para la suscripción de mensajes dentro de la red local. Dicha configuración podemos observar en la Figura 50.

Figura 50

Nodo MQTT OUT para establecer la comunicación



Nota. Fuente. El autor.

2.7.8.8 Etapa 10: Implementación del sistema de alertas físicas basado en ESP32

Después de que el sistema de visión artificial procesa la información y Node-RED gestiona la lógica de decisión mediante el protocolo de comunicación MQTT, se requiere un mecanismo físico el cual emita una señal visible en tiempo real. Esta etapa detalla el diseño electrónico del circuito, la configuración del entorno de desarrollo y programación del ESP32 para activar automáticamente la alerta visual.

- **Diseño electrónico del sistema de alertas físicas**

Este circuito electrónico permite integrar el microcontrolador ESP32 con un módulo relé para controlar la luz estroboscópica, desarrollando un esquema eléctrico en KiCad se garantiza una correcta distribución de señales y aislamiento de etapas, la etapa de control y la etapa de potencia.

Para el siguiente circuito se utilizó el software KiCad que permite diseñar de manera formal las conexiones eléctricas entre los componentes del sistema, el diagrama incluye los siguientes componentes que se puede observar en la Tabla 11.

Tabla 11

Componentes Utilizados

ESP32	- Microcontrolador con conectividad Wifi, opera a 3.3V - Ejecuta el programa de suscripción MQTT - Controla el pin GPIO encargado de activar el relé.
Modulo Rele (5V)	- Permite controlar cargas de mayor voltaje. - Aísla la etapa de control y potencia.
Luz estroboscópica	- Dispositivo de señalización visual - Se activa cuando el relé cierra el circuito. - Funciona como alerta física ante incumplimiento de EPP.

Nota. Fuente: El autor.

- **Funcionamiento y conexión (ESP32 GPIO, Rele, Luz estroboscópica)**

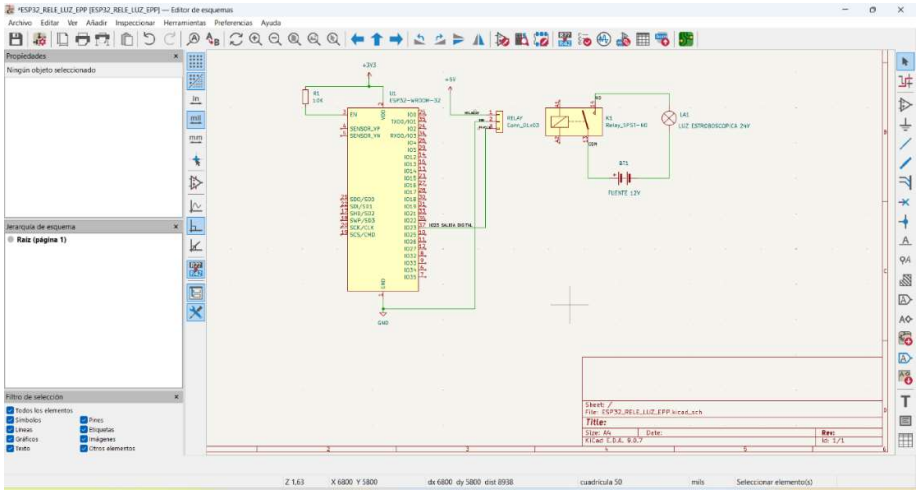
El dispositivo ESP32 recibe una señal lógica (ON/OFF) desde el sistema de visión artificial. Cuando la señal es activa (ON), el pin GPIO del ESP32 configurado como salida digital cambia a su nivel alto (HIGH) activando el módulo relé, el relé cierra el contacto normalmente abierto (NO) completando el circuito de alimentación de la luz estroboscópica. posteriormente la luz se enciende.

Cuando la señal emitida es OFF, el pin GPIO del ESP32 pasa a nivel bajo (LOW), el relé se desactiva permitiendo que el circuito de la luz estroboscópica se abra y posteriormente se apague.

La conexión eléctrica se dividió en dos etapas, la etapa de control que incluye el dispositivo ESP32 y la señal digital del pin GPIO (3.3V) y la etapa de potencia que incluye los contactos del relé y alimentación de la luz estroboscópica, asegurando la protección del microcontrolador. Dicha conexión la podemos observar en la Figura 51.

Figura 51

Diagrama de conexión del sistema ESP32-Rele-Luz estroboscópica



Nota. Fuente: El autor.

- **Configuración del entorno de desarrollo (Arduino IDE)**

Se utilizó el entorno de desarrollo Arduino ID debido a su facilidad de integración con la tarjeta embebida ESP32 además de su compatibilidad con las librerías de comunicación Wifi y MQTT garantizando la adecuada compilación, carga y ejecución del firmware el cual activa el sistema de alertas físicas.

Se procedió a la descarga del software Arduino IDE desde la página oficial, obteniendo así la interfaz principal del entorno de desarrollo como se puede observar en la Figura 52.

Figura 52

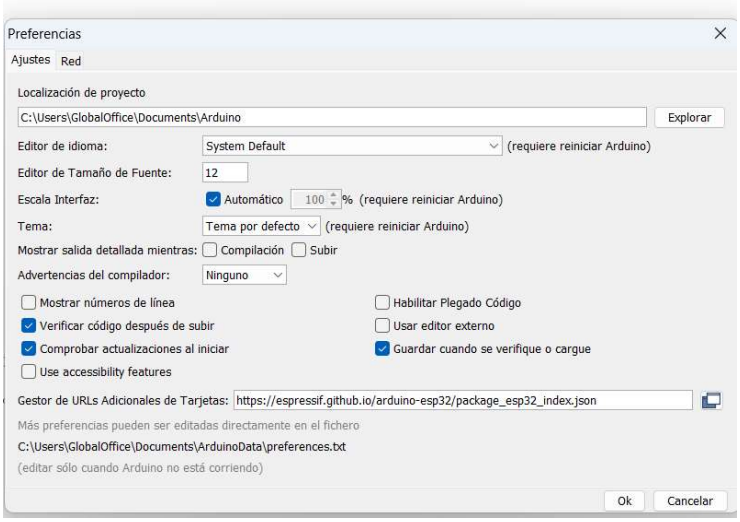
Entorno Arduino IDE instalado y operando

Nota. Fuente: El autor.

Posteriormente dado que Arduino IDE no posee por defecto las placas ESP32, fue necesario configurar el paquete de soporte para la tarjeta ESP32 donde incluye el repositorio oficial de Espressif Systems a través del Gestor URLs adicionales de tarjetas, esta acción permite instar el núcleo ESP32, el cual habilita el uso de librerías específicas, y la selección de la placa correspondiente y comunicación mediante puerto serial. Dicha configuración la podemos observar en la Figura 53.

Figura 53

Configuración del Arduino IDE para habilitar el soporte del microcontrolador ESP32

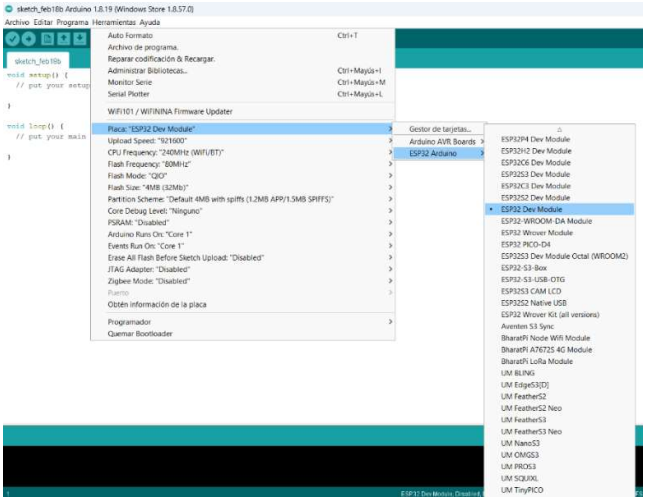


Nota. Fuente: El autor.

Posteriormente se selecciona la placa ESP32n Dev Module dentro del entorno Arduino IDE, para seleccionar la placa accedemos al menú herramientas, esta configuración permite que el compilador utilice la arquitectura y los parámetros adecuados del microcontrolador ESP32 para la correcta carga del firmware. Dicha configuración la podemos observar en la Figura 54.

Figura 54

Selección de la placa ESP32 Dev Module



Nota. Fuente: El autor.

Para permitir la conectividad Wifi y comunicación mediante el protocolo MQTT, se utilizaron las librerías correspondientes.

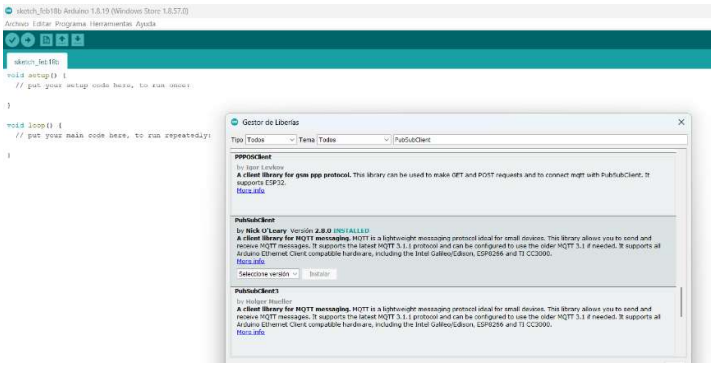
- WiFi.h esta librería permite la conexión a redes inalámbricas.
- PubSubClient.h Permite la comunicación mediante el protocolo de comunicación MQTT.

Cabe mencionar que la librería WiFi ya incluye el paquete ESP32, por lo tanto, no es necesario volverla a descargar.

la librería PubSubClient se la debe descargar para poder implementar el protocolo de comunicación con el broker MQTT, suscribirse al topic epp/esp32/cmd y posteriormente recibir mensajes “ON” y “OFF” que activa y desactiva el sistema de alertas físicas, su instalación se la puede observar en la Figura 55.

Figura 55

Librería PubSubClient para establecer comunicación MQTT

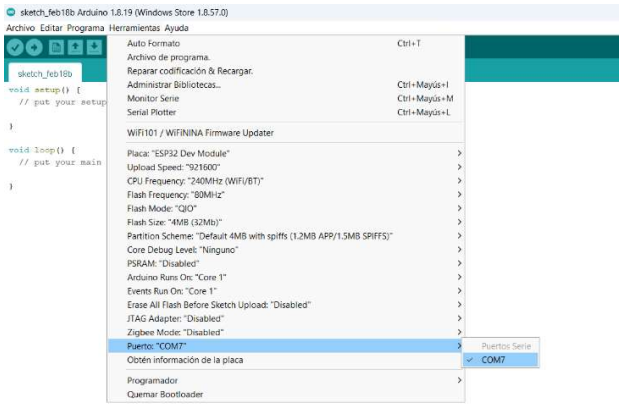


Nota. Fuente: El autor.

Posteriormente se configuró el puerto de comunicación entre el computador y el microcontrolador, el puerto COM7 es asignado automáticamente por el sistema operativo al detectar el dispositivo USB-Serial del ESP32. La correcta selección del puerto serial es fundamental para la carga del firmware en el microcontrolador, su configuración es mostrada en la Figura 56.

Figura 56

Configuración del puerto serial de comunicación, Puerto COM7



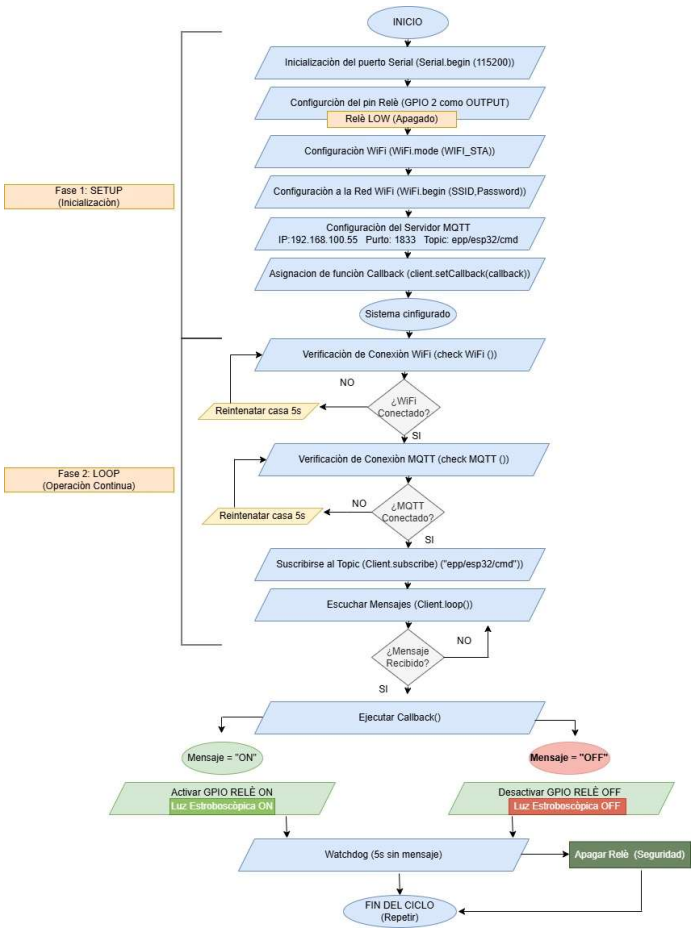
Nota. Fuente El autor.

Programación del dispositivo ESP32 para activación de alertas físicas

El siguiente programa implementa conectividad WiFi, comunicación MQTT y el control digital de salidas, además define el SSID y contraseña, se destacan las librerías utilizadas como son la configuración de la comunicación WiFi y MQTT, y la recepción de comandos remotos que controlan el estado del relé encargado de activar la luz estroboscópica. En el diagrama de flujo de la Figura 57, se muestra la programación implementada en el microcontrolador ESp32 y se divide en dos fases, fase de configuración (setup) y fase de operación continua (loop).

Figura 57

Diagrama de flujo de la programación ESP32 para activar alertas físicas



Nota. Fuente: El autor.

En base al diagrama de flujo presentado anteriormente se desarrolló el programa en el entorno Arduino IDE utilizando el lenguaje C++, donde se observa la inclusión de librerías necesarias, definición de variables de configuración y parámetros de conexión WiFi y MQTT. Como podemos observar en la Figura 58.

Figura 58

Fragmento del código implementado en el ESP32

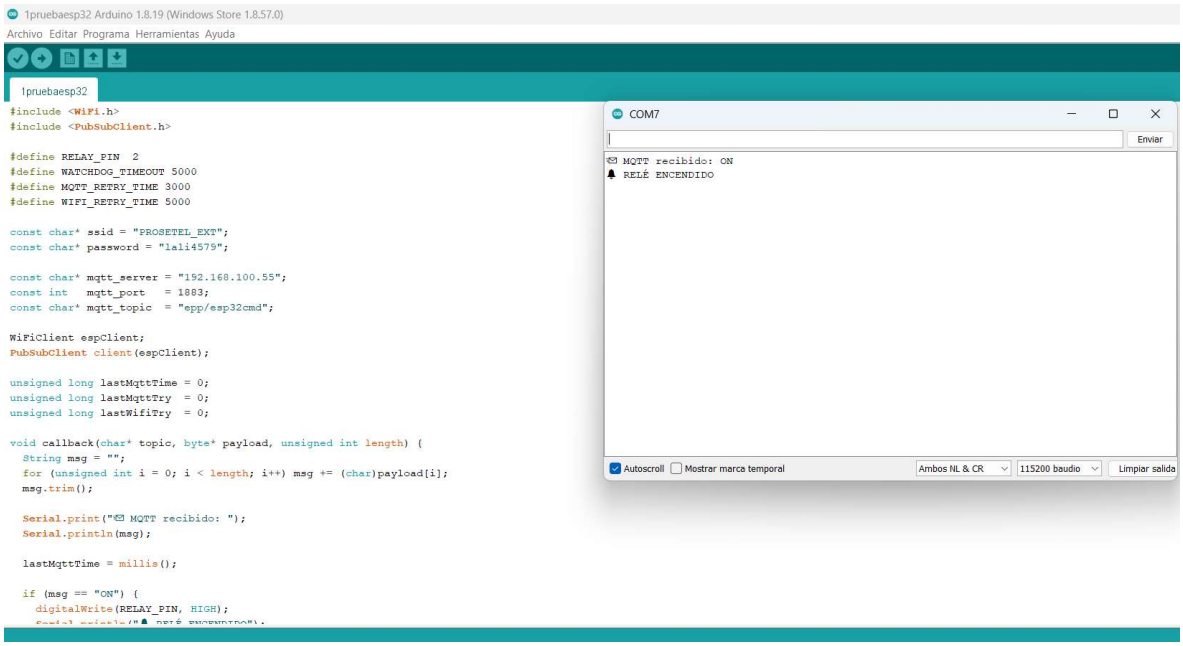
Nota. Fuente: El autor.

- **Verificación de comunicación y activación del relé**

Una vez copilado y cargado el programa el dispositivo ESP32, se procedió a verificar la comunicación con el broker MQTT mediante Monitor Serial del entorno Arduino IDE, donde se observa la correcta comunicación y recepción del mensaje “ON” enviado desde el sistema central Python, confirmando que el dispositivo ESP32 se encuentra correctamente suscrito al topic correspondiente como se puede observar en la Figura 59.

Figura 59

Monitor serial Arduino IDE mostrando la recepción de comandos MQTT y activación de relé



Nota. Fuente: El autor.

El dispositivo Esp32 ejecuta la función callback() al recibir el mensaje “ON” activando el puerto GPIO configurado como salida digital, lo que energiza el módulo relé y permite la activación de la alerta ante incumplimiento de EPP.

De igual manera al recibir el mensaje “OFF”, el sistema desactiva el GPIO correspondiente, apaga el relé y restablece el estado normal del sistema.

2.7.8.9 Etapa 11: Envío de notificaciones remotas mediante Telegram

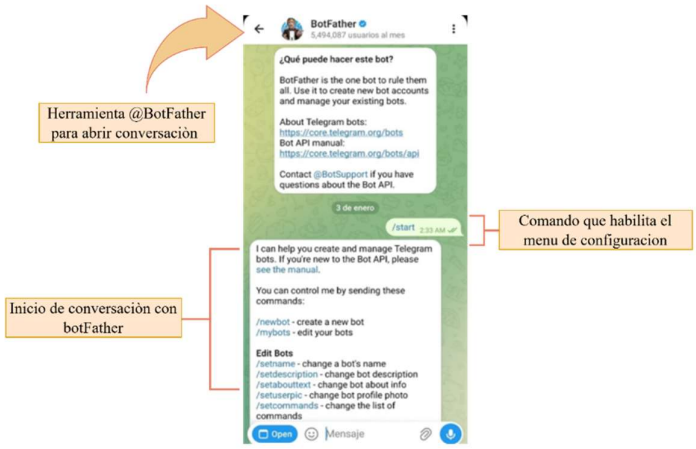
Para complementar el sistema de alertas físicas se incorporó un mecanismo de notificación remota a través de la plataforma Telegram. Este mecanismo permite enviar alertas automáticas al móvil del supervisor cuando el sistema de visión artificial detecta incumplimiento de EPP. Como requisito previo es indispensable contar con la aplicación de Telegram instalada en el dispositivo móvil, conexión a internet, creación de un bot oficial mediante la herramienta @BotFather.

- **Creación del bot mediante @BotFather**

Dentro de la aplicación de Telegram previamente ya instalada para generar el bot buscamos @BotFather para abrir una conversación, posteriormente se ejecuta el comando "/start." Este comando habilita el menú de configuración. Posteriormente ejecutamos "/newbot." Este comando inicia el proceso de creación de un nuevo bot dentro de la plataforma, luego el sistema solicita ingresar un nombre descriptivo para el bot. En este proyecto se asignó un nombre relacionado con el sistema. "EPP_Alertas_Bot", este nombre permite que el bot sea identificado dentro de la red de Telegram. el inicio de creación del bot mediante @BotFather se muestra en la Figura 60.

Figura 60

Proceso para la creación de bot mediante @BotFather



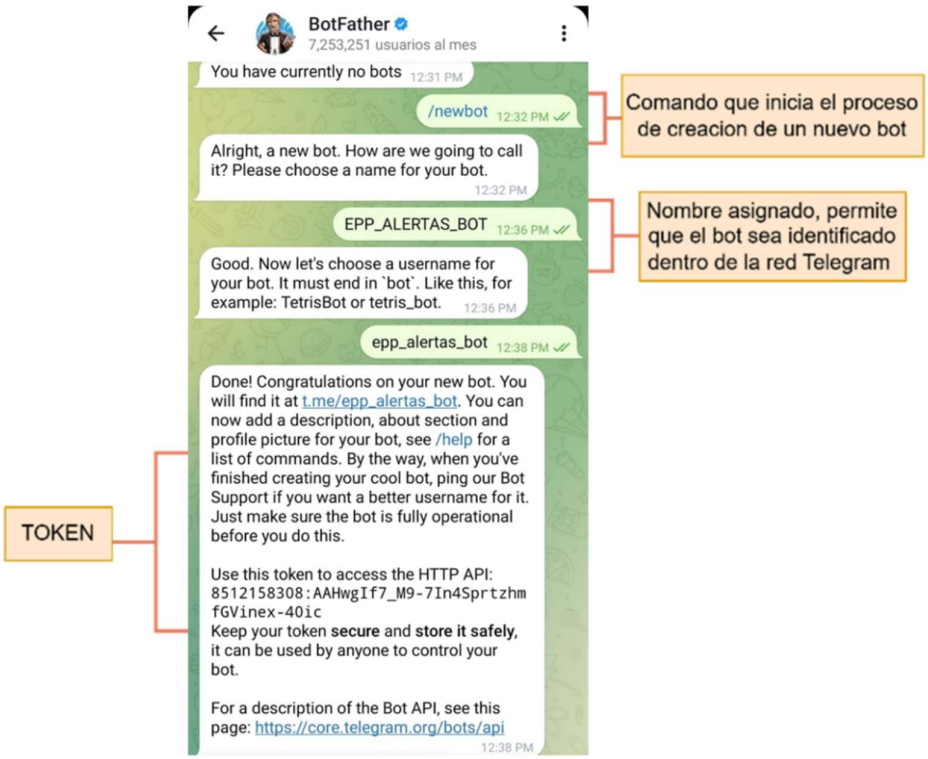
Nota. Fuente: El autor.

- **Obtención del TOKEN del bot**

Después de haber creado el bot, BotFather envía un mensaje tal y como se observa en la Figura 61. El sistema genera un TOKEN de autenticación único, la que se coloca en Node-RED y permite la comunicación con la API oficial de Telegram.

Figura 61

Creación del TOKEN



Nota. Fuente: El autor.

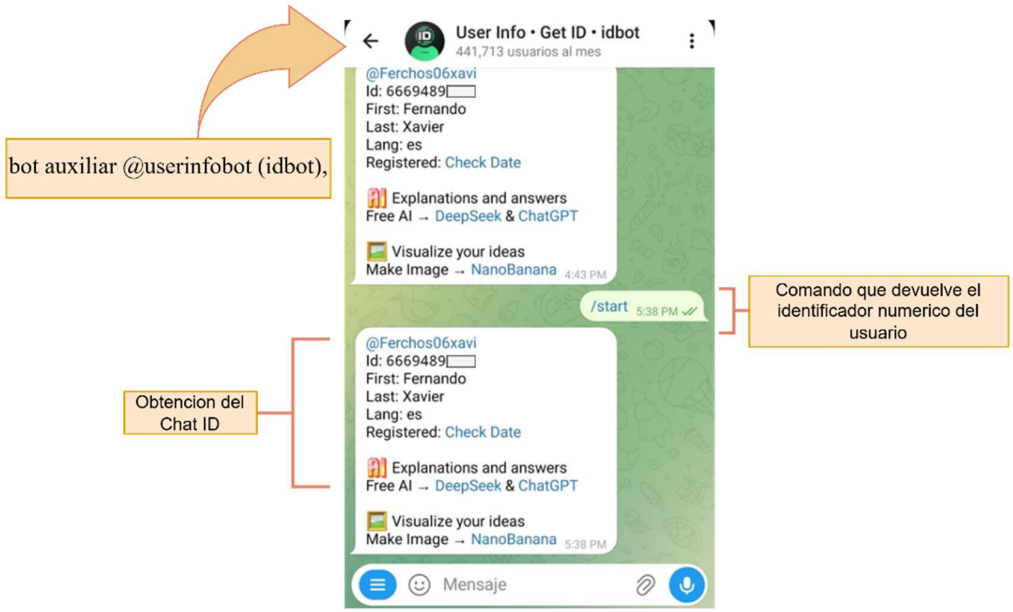
- **Obtención del Chat ID**

El sistema de visión artificial pueda enviar mensajes al usuario correcto, fue necesario obtener el identificador único de usuario (Chat ID).

Se utilizó el bot auxiliar @userinfobot (idbot), ejecutando el comando /start, el cual devolvió el identificador numérico del usuario como podemos observar en la Figura 62.

Figura 62

Obtención del Chat ID



Nota. Fuente: El autor.

- **Configuración e integración del bot en Node-RED**

Una vez obtenido el Chat ID y el Token, se procedió a la configuración del entorno de Nore-RED para habilitar él envió automático de notificaciones.

Para ello dentro de Node-RED se instaló el paquete “node-red-contrib-telegrambot” el cual se encuentra en el botón menú, manage palette.

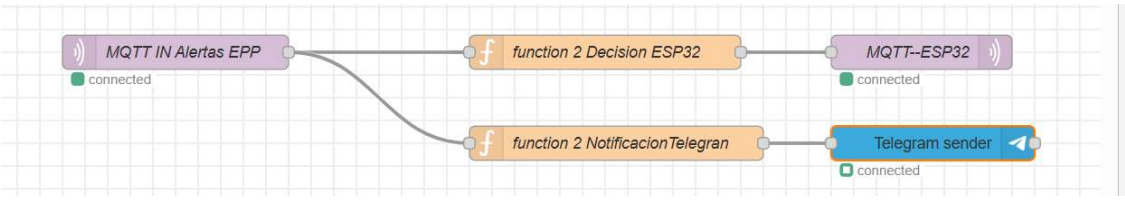
- **Integración del nodo Telegram y Function 2 dentro del flujo de alertas**

Se ha integrado al sistema de notificaciones los nodos Telegram Sender y nodo Function dentro del flujo principal, integra el nodo MQTT IN que es el encargado de recibir los mensajes publicados por el sistema central cuando se detecta incumplimiento de EPP,

este nodo esta suscrito al topic epp/esp32cmd y recibe el mensaje mediante el protocolo MQTT, Dicha integración de nodos se muestra en la Figura 63.

Figura 63

Conexión del nodo Telegram Sender y Fuction 2 Decisión dentro del flujo principal

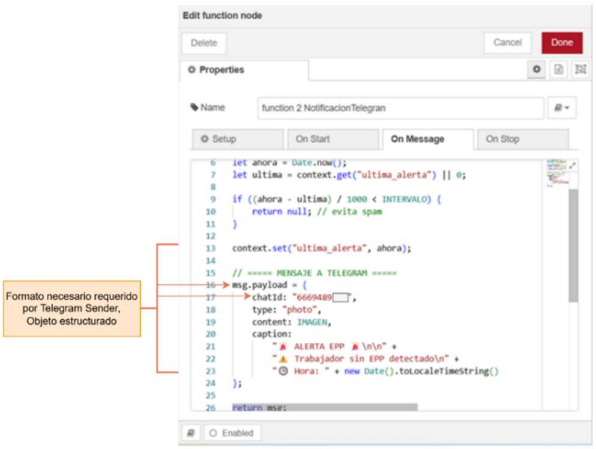


Nota. Fuente: El autor.

El nodo Function 2 Notificación Telegram cumple la función de interpretar el mensaje recibido desde MQTT IN, construye el texto de la alerta, adjunta información adicional como (hora, tiempo de incumplimiento), formatea el mensaje según el requerimiento del nodo Telegram Sender, dentro del nodo Function se genera una estructura de mensaje, este formato es necesario porque el nodo Telegram Sender espera un objeto estructurado con ChatId, type, Content, como se puede observar en la Figura 64.

Figura 64

Configuración interna del nodo Function para construcción del mensaje de alerta



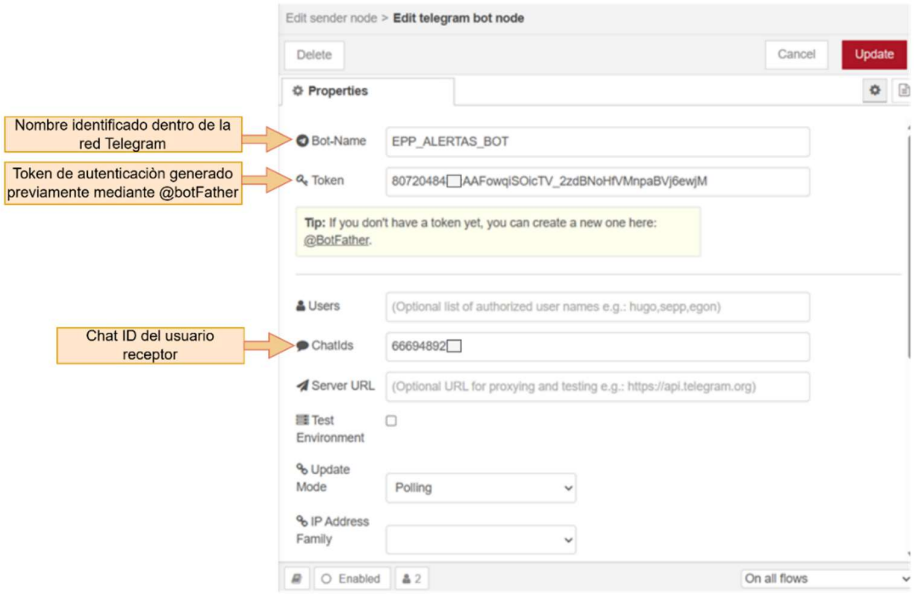
Nota. Fuente: El autor.

- **Configuración del Token en el nodo Telegram**

Para establecer la comunicación entre el sistema y la plataforma Telegram, es necesario configurar el nodo Telegram Sender dentro de Node-RED. En este nodo se ingresa los siguientes parámetros, el Token permite que Node-RED se autentique ante la API oficial de Telegram, mientras que Chat ID identifica el destinatario de la notificación, dicha configuración se puede observar en la Figura 65.

Figura 65

Configuración del nodo Telegram Sender



Nota. Fuente: El autor.

Telegram proporciona una API REST oficial, la cual permite enviar mensajes mediante solicitud web, su estructura general, https:/api.telegram.org/bot<TOKEN>/sendMessage permite que Node-Red realiza una solicitud web, el servidor Telegram recibe el mensaje, y Telegram lo redirige al dispositivo móvil del usuario, en esta comunicación se utilizó el protocolo HTTPS el cual implementa cifrado TLS asegurando que el TOKEN no sea interceptado, las alertas no puedan ser alteradas, y la comunicación sea segura.

- **Verificación del envió de notificaciones**

Finalmente, se realizó pruebas funcionales del sistema para validar el correcto envío de notificaciones remotas al móvil del supervisor. La recepción de la notificación se muestra en la Figura 66.

Figura 66

Notificación automática recibida en el dispositivo móvil mediante

Telegram ante incumplimiento de EPP

Nota. Fuente: El autor.

CAPÍTULO 3. RESULTADOS Y DISCUSIÓN

3.1 Resultados del modelo de detección de EPP

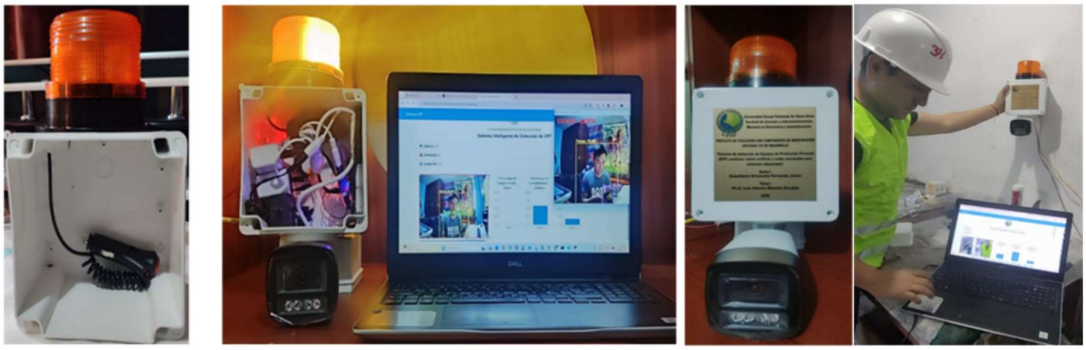
Con la finalidad de evaluar el desempeño del proyecto planteado se presentan los resultados obtenidos durante la validación del sistema de detección de EPP en condiciones reales de operación. Las pruebas se realizaron mediante transmisión en tiempo real desde una cámara IP, evaluando la capacidad del sistema para identificar la presencia de casco, chaleco y guantes, así como la correcta activación de alertas físicas y notificaciones remotas ante situaciones de incumplimiento.

3.1.1 Sistema Implementado

El sistema desarrollado fue implementado y puesto en marcha utilizando los componentes de captura, procesamiento y activación de alertas físicas. La arquitectura final incluye una cámara IP para la adquisición de video en tiempo real, un módulo de procesamiento ejecutado el modelo YOLOV8 para la detección de EPP, un dashboard en Nore-RED para monitoreo visual y un sistema de alertas físicas basado en ESP32 conectado a un relé y una luz estroboscópica. La Figura 67 muestra el sistema completo en funcionamiento durante la experimentación de pruebas.

Figura 67

Sistema completo implementado para detección y generación de alertas de incumplimientos de EPP



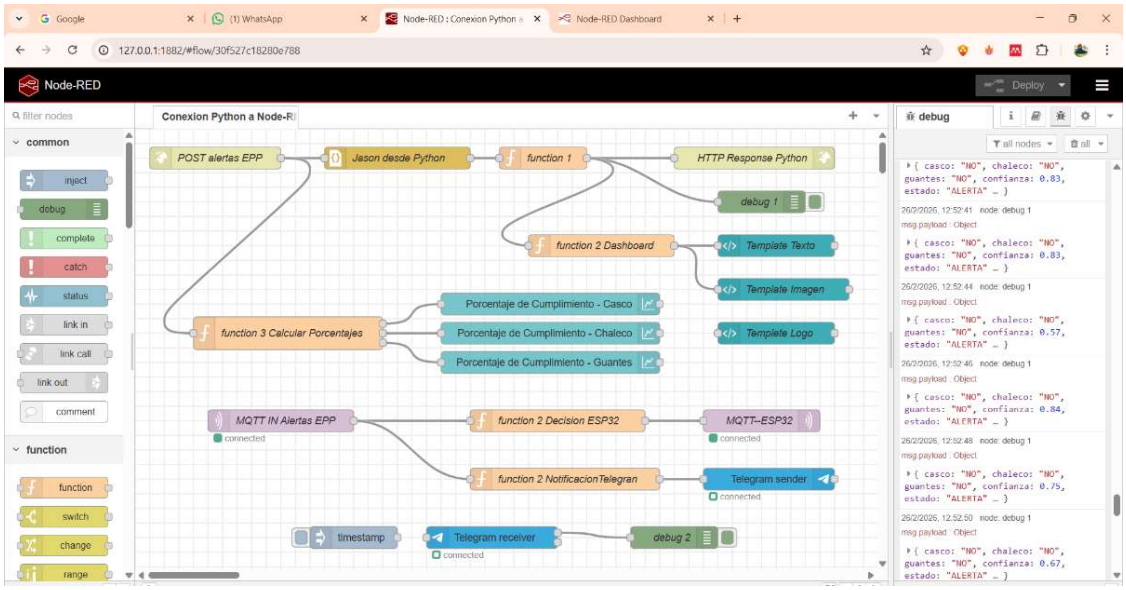
Nota. Fuente: El autor.

3.1.2 Flujo de operación del sistema en Node-RED

Entorno operativo del sistema en Node-RED durante las pruebas experimentales, mostrando la integración entre la detección de EPP, la lógica de decisión y la activación de notificaciones, como se puede observar en la Figura 68.

Figura 68

Flujo de procesamiento del sistema en Node-RED



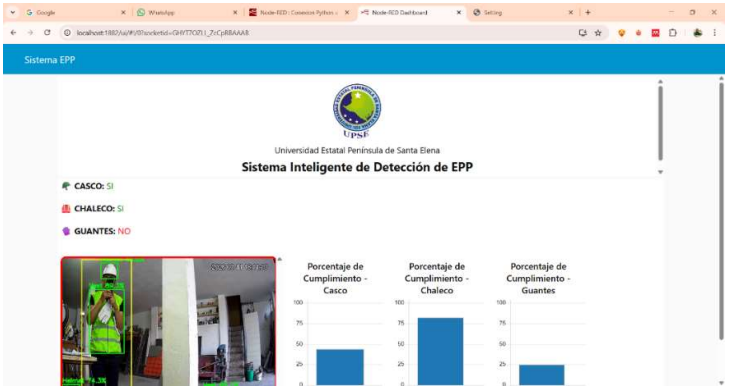
Nota. Fuente: El autor.

3.1.3 Interfaz de visualización y monitoreo real mediante Node-RED

Interfaz de monitoreo en tiempo real, donde se visualiza el estado individual de casco, chaleco, guantes, mediante un dashboard interactivo desarrollado en Nore-RED. Donde muestra también la imagen procesada por el modelo de visión artificial. Dicha interfaz la podemos observar en la Figura 69.

Figura 69

Dashboard del sistema durante la fase de validación. experimental



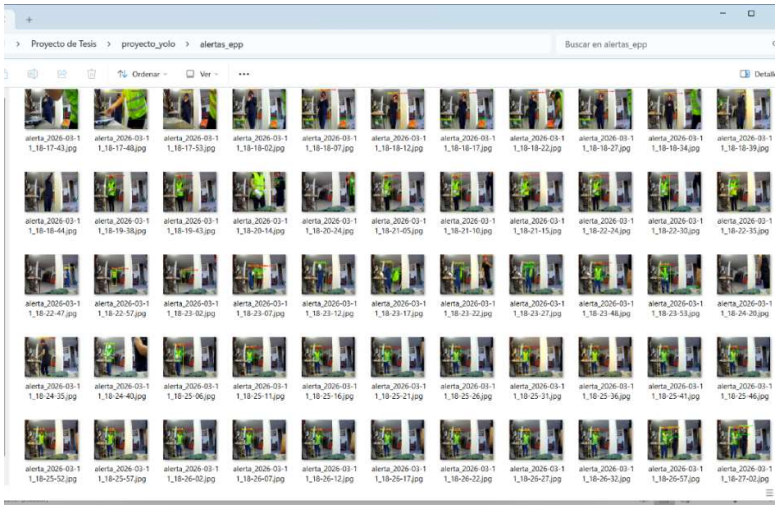
Nota. Fuente: El autor.

3.1.4 Registro automático de eventos de incumplimiento de EPP

Registro automático de imágenes que corresponden a eventos de incumplimiento de EPP, se almacenaron en la carpeta del sistema como evidencia digital incluyendo fecha y hora del evento, como se puede observar en la **Figura 70.**

Figura 70

Registro automático de evidencias ante incumplimientos de EPP



Nota. Fuente: El autor.

3.1.5 Supervisión remota y registro de eventos detectados por el sistema

La imagen procesada por el modelo de visión artificial permite realizar la supervisión remota en tiempo real y registro de eventos críticos, como se muestra en la Figura 71.

Figura 71

Notificación automática de incumplimiento de EPP enviada mediante Telegram al supervisor

Nota. Fuente: El autor.

3.1.6 Escenario de validación

La validación del sistema se realizó mediante la construcción de la matriz de confusión y el cálculo de métricas de desempeño derivadas de los indicadores TP, TN, FP, FN, se

evaluó el desempeño del sistema ante presencia y ausencia de los siguientes EPP. Casco, Chaleco, guantes: para cada clase se realizaron 40 pruebas controladas, divididas en: 20 casos con uso correcto del EPP, 20 casos sin uso del EPP.

20 pruebas con casco:

- Verdaderos negativos (TN) = 17
- Falsos positivos (FP) = 3

20 pruebas sin casco:

- Verdaderos positivos (TP) = 18
- Falsos negativos (FN) = 2

La Tabla 12, muestra la matriz de confusión obtenida para la detección de casco, donde se realizaron 40 pruebas y se comparó las predicciones del sistema con los datos reales registrados durante las pruebas experimentales. A partir de los resultados obtenidos se puede calcular las métricas como precisión, sensibilidad (reacll) y exactitud.

MATRIZ DE CONFUSIÒN (Casco)

Tabla 12

Matriz de confusión para la detección de casco

	Sistema (Casco No)	**Sistema (Casco Si)**	**Total, Real**
Incumplimiento: Real	18 (TP)	2 (FN)	20
Cumplimiento EPP	3 (FP)	17 (TN)	20
Total, Sistema	21	19	40

Nota. Fuente: El autor.

Precisión = $\frac{18}{18+3} = \mathbf{0,85\ x\ 100\% = 85\%}$

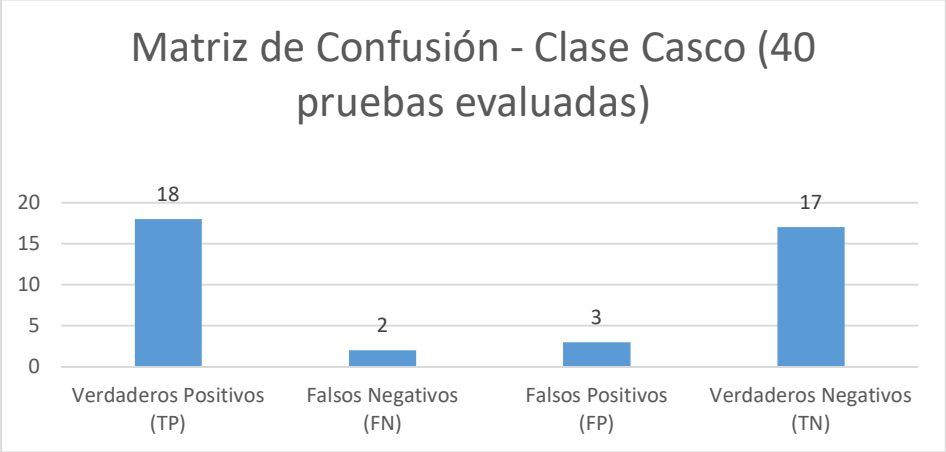
Recall = $\frac{18}{18+2} = \mathbf{0,9}$ x 100% = 90%

Accuracy = $\frac{18+17}{40}$ = 0,87 x 100% = 87%

En la Figura 72, se muestra la representación gráfica de la matriz de confusión correspondiente a la detección de la clase casco. Los resultados evidencian la cantidad de verdaderos positivos (TP), falsos negativos (FN), falsos positivos (FP), y verdaderos negativos (TN) obtenidos durante las pruebas realizadas, permitiendo analizar el desempeño del sistema en la identificación correcta del uso de la clase casco. Se observa que la clase casco presenta un mayor número de verdaderos positivos, lo que indica que el sistema logra identificar correctamente la ausencia de este EPP.

Figura 72

Resultado de la matriz de confusión del sistema para la detección de casco

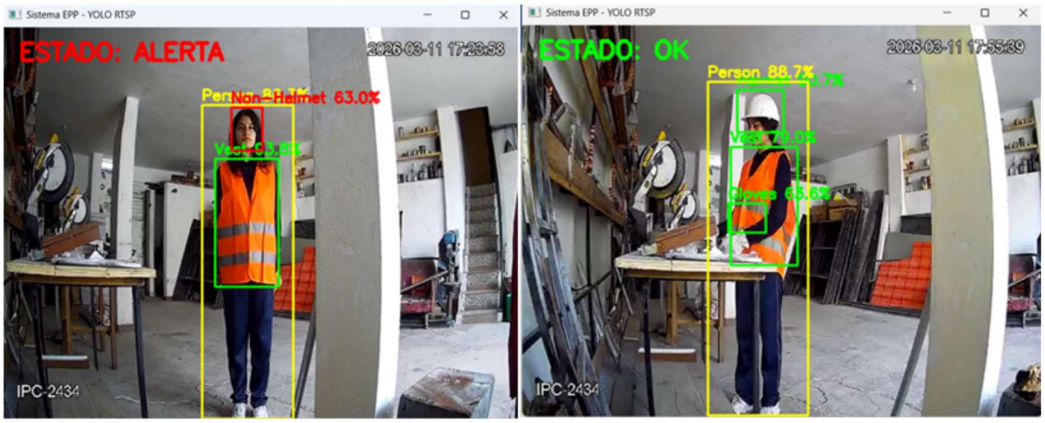


Nota. Fuente: El autor.

La Figura 73, muestra un ejemplo de detección de cumplimiento del uso de casco, en la imagen se observan las cajas delimitadoras (bounding boxes) generadas por el sistema, junto con las etiquetas de clasificación y el nivel de confianza asociado a cada detección, estos resultados evidencian que el modelo identifica correctamente los EPP.

Figura 73

Muestra un ejemplo de detección de cumplimiento de casco



Nota. Fuente: El autor.

La Tabla 13, muestra la matriz de confusión obtenida para la detección de chaleco, donde se realizaron 40 pruebas y se comparó las predicciones del sistema con los datos reales registrados durante las pruebas experimentales. A partir de los resultados obtenidos en la matriz de confusión se puede calcular las métricas como precisión, sensibilidad (recall) y exactitud.

pruebas con chaleco:

- Verdaderos negativos (TN) = 18
- Falsos positivos (FP) = 2

20 pruebas sin chaleco:

- Verdaderos positivos (TP) = 17
- Falsos negativos (FN) = 3

Total = 40 pruebas realizadas

MATRIZ DE CONFUSIÒN (Chaleco)

Tabla 13

Matriz de confusión para la detección de chaleco

	Sistema (Chaleco No)	**Sistema (Chaleco Si)**	**Total, Real**
Incumplimiento: Real	17 (TP)	3 (FN)	20
Cumplimiento EPP	2 (FP)	18 (TN)	20
Total, Sistema	19	21	40

Nota. Fuente: El autor.

Precisión = $\frac{\mathbf{17}}{\mathbf{17+2}} = \mathbf{0,89\ x\ 100\%} = \mathbf{89\%}$

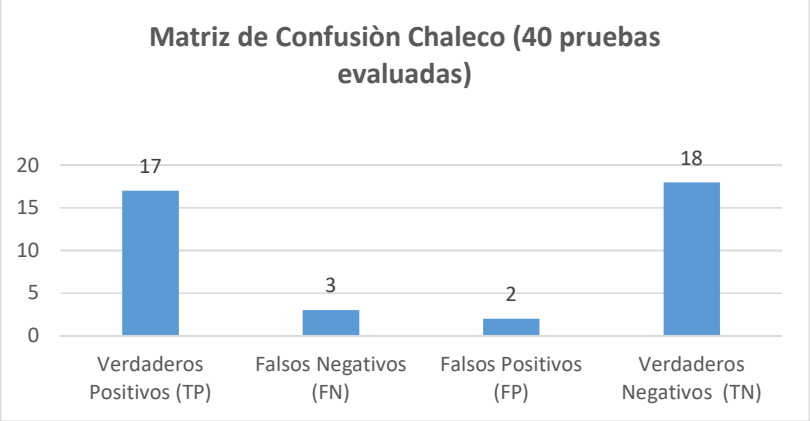
Recall = $\frac{\mathbf{17}}{\mathbf{17+3}} = \mathbf{0,85}$ x 100% = 85%

Accuracy = $\frac{17+18}{40} =$ 0,87 x 100% = 87%

En la Figura 74, muestra la representación gráfica de la matriz de confusión correspondiente a la detección de la clase chaleco, los resultados muestran la cantidad de verdaderos positivos (TP), falsos negativos (FN), falsos positivos (FP), y verdaderos negativos (TN) obtenidos durante las pruebas realizadas. Estos valores permiten analizar el desempeño del sistema. Se observa también que la clase chaleco presenta un alto número de verdaderos positivos y verdaderos negativos, lo que indica que el sistema logra identificar de forma adecuada la presencia de este EPP.

Figura 74

Resultado de la matriz de confusion para la detección de chaleco

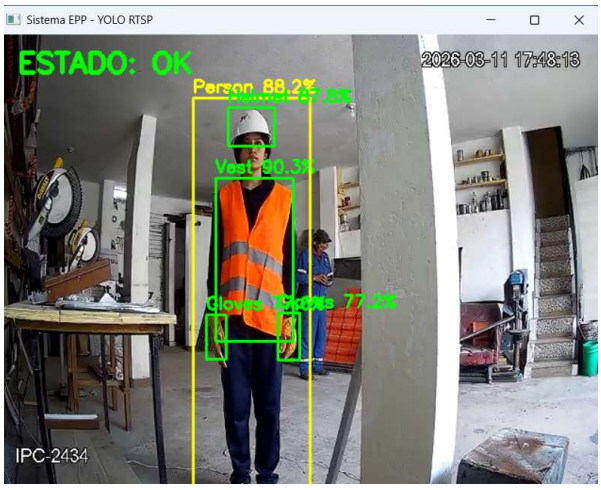


Nota. Fuente: El autor.

Como se muestra en la Figura 75, el sistema de visión artificial logra identificar correctamente el uso de chaleco. En la imagen se observan cajas delimitadoras (bounding boxes) generados por el modelo de detección, junto con las etiquetas de clasificación y el nivel de confianza asociado a cada detección.

Figura 75

Muestra un ejemplo de detección del cumplimiento de chaleco



Nota. Fuente: El autor.

La Tabla 14, muestra la matriz de confusión obtenida para la detección de guantes, donde se realizaron 40 pruebas y se comparó las predicciones del sistema con los datos reales registrados durante las pruebas experimentales. A partir de los resultados obtenidos en la matriz de confusión se puede calcular las métricas como precisión, sensibilidad (reacll) y exactitud.

20 pruebas con guantes:

- Verdaderos negativos (TN) = 15
- Falsos positivos (FP) = 5

20 pruebas sin guantes:

- Verdaderos positivos (TP) = 16
- Falsos negativos (FN) = 4

Total = 40 pruebas realizadas

MATRIZ DE CONFUSIÒN (Guantes)

Tabla 14

Matriz de confusión para la detección de guantes

	Sistema (Guantes No)	**Sistema (Guantes Si)**	**Total, Real**
Incumplimiento: Real	16 (TP)	4 (FN)	20
Cumplimiento EPP	5 (FP)	15 (TN)	20
Total, Sistema	21	19	40

Nota. Fuente: El autor.

Precisión = $\frac{16}{16+5} = \mathbf{0,76\ x\ 100\%} = \mathbf{76\%}$

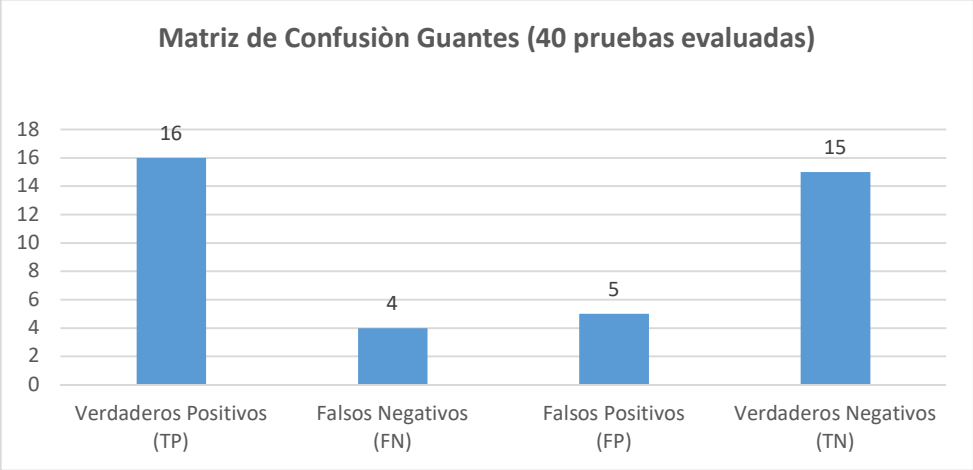
Recall = $\frac{16}{16+4} = \mathbf{0,8}$ x 100% = 80%

Accuracy = $\frac{16+1}{40} =$ 0,77 x 100% = 77%

Como se observa en la Figura 76, se representa la gráfica de la matriz de confusión correspondiente a la detección de la clase guantes. Se puede observar que el modelo presenta un número considerable de verdaderos positivos y verdaderos negativos, lo que indica que el sistema logra identificar de manera adecuada la presencia de estos EPP.

Figura 76

Resultado de la matriz de confusion para la detección de guantes



Nota. Fuente: El autor.

Como se muestra en la Figura 77, el sistema de visión artificial logra identificar correctamente el uso de guantes. En la imagen se observan cajas delimitadoras (bounding boxes) generados por el modelo de detección, junto con las etiquetas de clasificación y el nivel de confianza asociado a cada detección.

Figura 77

Muestra un ejemplo de detección de guantes

Nota. Fuente: El autor.

3.1.7 Cálculo general de las métricas de evaluación por clases

Observamos que las clases casco y chaleco presentan un mayor desempeño general del sistema, alcanzando mayores valores de precisión y exactitud. La clase guantes presenta el menor rendimiento, lo cual puede atribuirse a variaciones en iluminación, y posición del trabajador durante la captura de imágenes. Asimismo, se calculó el promedio general de las métricas de precisión, recall y exactitud con el fin de obtener una evaluación general del desempeño del sistema considerando todas las clases analizadas, como podemos observar en la Tabla 15.

Tabla 15

Comparación del desempeño por clase

EPP	Precisión	Recall	Accuracy
Casco	85%	90%	87%
Chaleco	89%	85%	87%
Guantes	76%	80%	77%
Promedio general	**83.3%**	**85%**	**83.7%**

Nota. Fuente: El autor.

3.1.8 Evaluación general del sistema completo

Al evaluar el sistema completo definimos que tan bien detecta el sistema el cumplimiento general de EPP, por lo tanto, cuando el sistema detecta al menos la falta de un EPP, el sistema genera un estado Positivo (Alerta), cuando el sistema detecta la presencia de todos los EPP, casco, chaleco y guantes el sistema permanece estado Negativo (OK), los resultados los podemos observar en la Tabla 16.

20 pruebas con todos los EPP:

- Verdaderos negativos (TN) = 15
- Falsos positivos (FP) = 5

20 con incumplimiento (falta de al menos un EPP)

- Verdaderos positivos (TP) = 17
- Falsos negativos (FN) = 3

Total = 40 pruebas realizadas

Matriz de confusión Global

Tabla 16

Matriz de confusión global para la detección de EPP

	Sistema Alerta	Sistema OK	Total, real
Incumplimiento real	17 (TP)	3 (FN)	20
Cumplimiento real	5 (FP)	15 (TN)	20
Total, Sistema	22	18	40

Nota. Fuente: El autor.

Precisión = $\frac{\mathbf{17}}{\mathbf{17+5}} = \mathbf{0,77\ x\ 100\%} = \mathbf{77\%}$

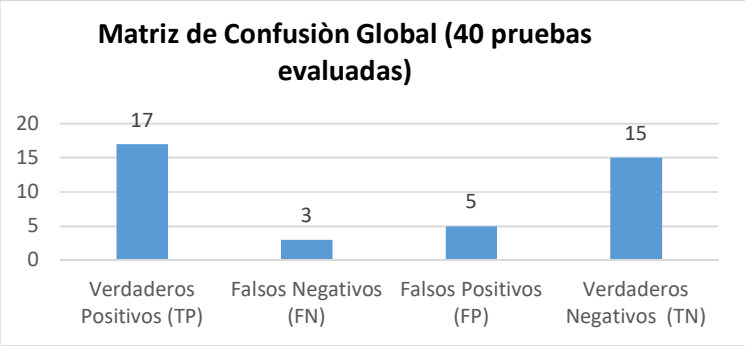
Recall = $\frac{\mathbf{17}}{\mathbf{17+3}} = \mathbf{0,85}$ x 100% = 85%

Accuracy = $\frac{17+15}{40} =$ 0,8 x 100% = 80%

Como se observa en la Figura 78, se presenta la matriz de confusión global del sistema obtenida a partir de las 40 pruebas evaluadas. En esta representación se muestran los valores correspondientes a verdaderos positivos (TP), falsos negativos (FN), falsos positivos (FP) y verdaderos negativos (TN), los cuales permiten analizar el desempeño general del modelo en la detección de EPP. Se puede observar que el número de verdaderos negativos y verdaderos negativos es superior al de errores de clasificación, lo que indica que el modelo presenta un comportamiento adecuado en la identificación de EPP.

Figura 78

Resultado de la matriz de confusión para sistema global

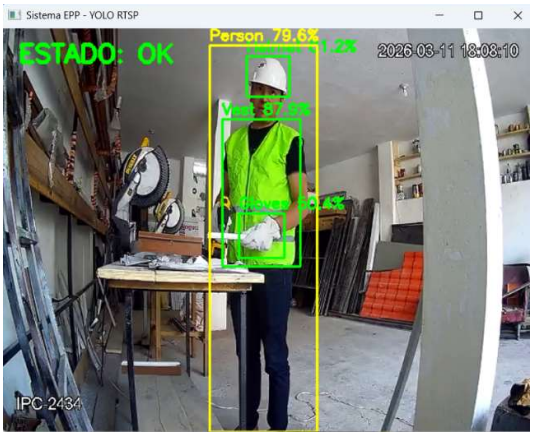


Nota. Fuente: El autor.

Como se observa en la Figura 79, el sistema de visión artificial permite identificar simultáneamente distintos EPP, como casco, chaleco y guantes. En la imagen se muestra las cajas delimitadoras (bounding boxes) generados por el modelo de detección, junto con las etiquetas de clasificación y el nivel de confianza asociado a cada objeto detectado. Estos resultados evidencian la capacidad del sistema para reconocer múltiples elementos de seguridad en una misma escena.

Figura 79

Muestra un ejemplo de detección de cumplimiento de todos los EPP



Nota. Fuente: El autor

Discusión

La clase casco es la que presenta el mejor desempeño general. Este resultado puede atribuirse a que el tamaño y visibilidad en la región superior del cuerpo. Lo que facilita la detección por el modelo YOLOv8.

Por otra parte, la clase guantes es la que menor desempeño presentó en términos de precisión y exactitud, posiblemente por oclusión parcial, variación de iluminación y tamaño reducido del objeto en la captura de imágenes.

Durante la evaluación general del sistema alcanzó un accuracy global del 80% lo que evidencia su viabilidad para aplicaciones en entornos de trabajo donde se requiere la supervisión de EPP.

En cuanto a la limitación del sistema se puede observar sensibilidad ante variaciones de iluminación, cambios en la postura del trabajador y distancia respecto a la cámara, factores que pueden afectar la precisión de detección.

De manera general, el sistema demostró un funcionamiento estable durante las pruebas, ya que integró correctamente la captura de video, el procesamiento del modelo, y la generación de alertas en tiempo real a través de Node-RED y Telegram, la comunicación entre los módulos no presenta fallos, evidenciando una adecuada sincronización entre la detección y la notificación. Estas pruebas y resultados confirman que la arquitectura que se implemento es funcional y aplicable en un entorno real de supervisión.

CONCLUSIONES

- Se desarrolló e implementó un sistema que permite identificar y evaluar el uso de equipos de protección personal (EPP). Para su funcionamiento, se entrenó un modelo utilizando el algoritmo YOLOv8 con un conjunto de imágenes que incluye personas con y sin el uso de casco, chaleco y guantes.
- La implementación del sistema de detección de EPP en la empresa Alumvid evidenció mejoras en el cumplimiento de normas de seguridad.
- El uso de herramientas de monitoreo, visón artificial y protocolos de comunicación permitió identificar el incumplimiento en el uso de EPP, lo cual permite reducir los riesgos laborales dentro de la empresa.
- Para la evaluación del sistema se realizaron pruebas experimentales y se analizaron métricas de desempeño como precisión, recall y accuracy, con los resultados obtenidos, se verificó el funcionamiento, mostrando un desempeño eficiente en la detección de los distintos EPP.
- Los resultados obtenidos del sistema confirman la utilidad de redes neuronales convolucionales profundas para la supervisión automática del cumplimiento de normas de seguridad en entornos industriales.

RECOMENDACIONES

- Se recomienda optimizar el modelo YOLOv8 mediante el uso de un mayor número de imágenes reales del entorno de trabajo, considerando diferentes condiciones de iluminación, distancia y ángulos de captura, con el fin de mejorar la precisión de detección y reducir falsos positivos y falsos negativos.
- Ubicar la cámara IP en posiciones estratégicas que permitan una visualización frontal del trabajador, evitando errores ocasionados por ángulo u oclusión, ya que estos factores influyen directamente en errores de detección.
- Se sugiere analizar el uso de hardware dedicado, como GPU o dispositivos embebidos con aceleración, antes de la implementación, ya que este tipo de hardware puede mejorar la velocidad de procesamiento y reducir la latencia en sistemas de monitoreo.
- Adaptar el sistema a diferentes sectores industriales o entornos laborables donde el uso de EPP sea obligatorio.

REFERENCIAS

Borrella Petisco, B. (2022). *Introducción a la visión artificial: procesos y aplicaciones*. https://hdl.handle.net/20.500.14352/3290

Carpio, A., Carlos, J., Astete, P., & Felipe, R. (2021). *Desarrollo de un algoritmo computacional de detección de equipos de protección eléctrica en personas, orientado a sistemas de vigilancia basados en cámaras IP*. https://repositorioacademico.upc.edu.pe/handle/10757/657930

Chaguancallo, J. B., & León, C. S. (2023). *de un sistema basado en visión artificial para el reconocimiento y control del EPP en el personal operativo de la línea de extrusión de alimentos para mascotas …*. https://dspace.espoch.edu.ec:8080/server/api/core/bitstreams/c174fe79-1c35-4b87-a0b7-a60bd39a473d/content

Digitales Iii, T., Jesús Rivero, F., Nicolás Taglialegne, A., & Jesús, I. (n.d.). *UNIVERSIDAD TECNOLÓGICA NACIONAL FACULTAD REGIONAL SAN NICOLÁS INGENIERÍA ELECTRÓNICA PROBLEMA DE INGENIERÍA Integrantes*.

Massiris, M., Fernández, J. A., Bajo, J., & Delrieux, C. (2021). An automated system for monitoring the use of personal protective equipment in the construction industry. *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, *18*(1), 68–74. https://doi.org/10.4995/RIAI.2020.13243

Rufino, K. A. (2024). *Desarrollo de un algoritmo de reconocimiento facial y detección de EPPS con Python para la industria de construcción*. https://repositorio.upn.edu.pe/handle/11537/37448

Yaseen, M. (2024). *W yolo 8: a i -d e i f n -g o d. 8*, 1–10. https://arxiv.org/pdf/2408.15857v1

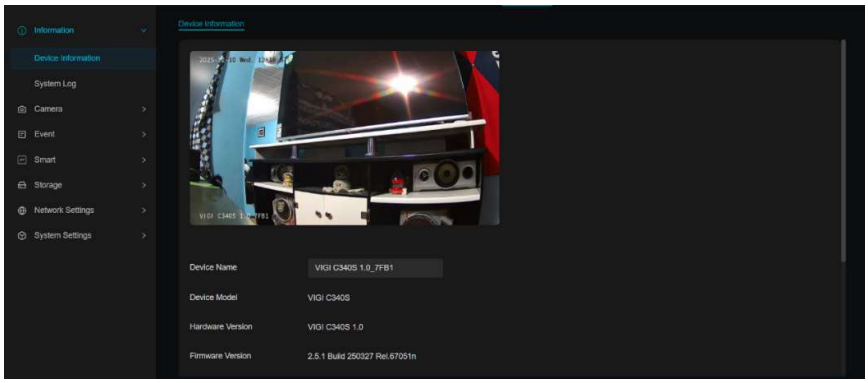
Gupta, V. (2026). *Deep learning tutorial for beginners – A complete roadmap. igmGuru. https://www.igmguru.com/blog/deep-learning-tutorial*

ANEXOS

Anexo 1: Primeras imágenes obtenidas al conectar cámara IP mediante protocolo RTSP

Figura 80

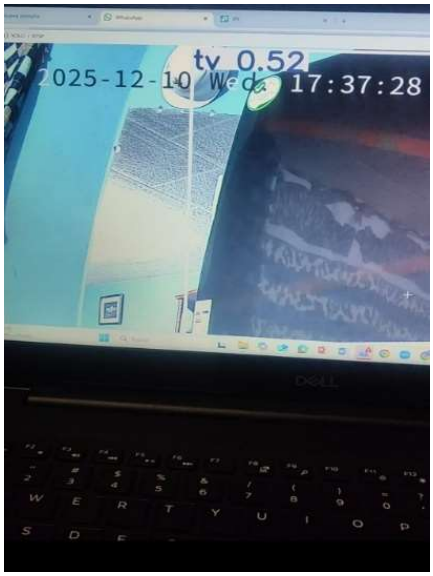
Conexión RTSP para captura de video



Anexo 2: Primeras imágenes obtenidas al instalar YOLOv8 mediante su dataset estándar coco.

Figura 81

Imagen modelo YOLOv8



Anexo 3: Programación del modelo para detectar EPP

```
import cv2
import os
import time
import base64
import requests
from datetime import datetime
from ultralytics import YOLO
import paho.mqtt.client as mqtt

# ================= CONFIGURACIÓN DE URL =================
RTSP_URL =
"rtsp://admin:Prosetel2025!@192.168.1.239:554/cam/realmonitor?channel
=1&subtype=1"
NODE_RED_URL = "http://127.0.0.1:1882/alerta_epp"
MODEL_PATH = r"C:\Users\GlobalOffice\Desktop\Proyecto de
Tesis\proyecto_yolo\runs\detect\train8\weights\best.pt"

# ===== MQTT =====
MQTT_BROKER = "192.168.100.167"
MQTT_PORT = 1883
MQTT_TOPIC = "epp/esp32cmd"

CONF_TH = 0.5
ALERT_INTERVAL = 5
ENVIO_ESTADO_INTERVAL = 2

# ===== NUEVO =====
ESTABILIDAD_TIEMPO = 1.5   # segundos
MQTT_INTERVAL = 1.0        # segundos

SAVE_DIR = "alertas_epp"
os.makedirs(SAVE_DIR, exist_ok=True)

VERDE = (0, 255, 0)
ROJO  = (0, 0, 255)

# ================= COMUNICACIÓN MQTT =================
mqtt_client = mqtt.Client(client_id="python_yolo_epp")
mqtt_client.connect(MQTT_BROKER, MQTT_PORT, 60)
mqtt_client.loop_start()

def mqtt_enviar(msg):
    mqtt_client.publish(MQTT_TOPIC, msg)


# ================= MODELO ENTRENADO =================

print(" Cargando modelo YOLO...")

model = YOLO(MODEL_PATH)

print("Modelo cargado")


# ================= CÁMARA IP=================
```

```
cap = cv2.VideoCapture(RTSP_URL, cv2.CAP_FFMPEG)
cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)

if not cap.isOpened():
    print(" No se pudo abrir la cámara RTSP")
    exit()

print("Cámara conectada")
print("Sistema EPP en ejecución\n")

# ================ ESTADOS DE ALERTAS =================
ultimo_envio_alerta = 0
ultimo_envio_estado = 0
ultimo_envio_mqtt = 0

alerta_inicio = None
ok_inicio = None
estado_estable = "OK"

# ================ PROCESO LOOP =================
while True:
    ret, frame = cap.read()
    if not ret:
        continue

    casco = chaleco = guantes = "SIN DEFINIR"
    alerta_frame = False
    max_conf = 0.0
    results = model(frame, conf=CONF_TH, verbose=False)[0]
    for box in results.boxes:
        cls_id = int(box.cls[0])
        conf = float(box.conf[0])
        label = model.names[cls_id]
        x1, y1, x2, y2 = map(int, box.xyxy[0])
        color = VERDE
        if label == "Helmet":
            casco = "SI"
        elif label == "NoHelmet":
            casco = "NO"
            color = ROJO
```

```
            alerta_frame = True

        elif label == "Vest":
            chaleco = "SI"
        elif label == "NoVest":
            chaleco = "NO"
            color = ROJO
            alerta_frame = True

        elif label == "Gloves":
            guantes = "SI"
        elif label == "NoGloves":
            guantes = "NO"
            color = ROJO
            alerta_frame = True

        max_conf = max(max_conf, conf)

        cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
        cv2.putText(frame, f"{label} {conf:.2f}",
                    (x1, y1 - 6),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.6, color, 2)

    ahora = time.time()

    # ============ FILTRO============
    if alerta_frame:
        ok_inicio = None
        if alerta_inicio is None:
            alerta_inicio = ahora
        elif ahora - alerta_inicio >= ESTABILIDAD_TIEMPO:
            estado_estable = "ALERTA"
    else:
        alerta_inicio = None
        if ok_inicio is None:
            ok_inicio = ahora
        elif ahora - ok_inicio >= ESTABILIDAD_TIEMPO:
            estado_estable = "OK"

    # ============ MQTT ============
    if ahora - ultimo_envio_mqtt >= MQTT_INTERVAL:
        if estado_estable == "ALERTA":
            mqtt_enviar("ON")
        else:
            mqtt_enviar("OFF")
        ultimo_envio_mqtt = ahora

    # ============ NODE-RED ============
    if ahora - ultimo_envio_estado >= ENVIO_ESTADO_INTERVAL:
        payload_estado = {
            "casco": casco,
            "chaleco": chaleco,
            "guantes": guantes,
            "confianza": round(max_conf, 2)
        }
        try:
           requests.post(NODE_RED_URL, json=payload_estado, timeout=1)
            ultimo_envio_estado = ahora
        except:pass
```

```
    # ============ ALERTA Y FOTO ============
    if estado_estable == "ALERTA" and ahora - ultimo_envio_alerta >=
ALERT_INTERVAL:
        ts = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
        ruta = f"{SAVE_DIR}/alerta_{ts}.jpg"
        cv2.imwrite(ruta, frame)

        _, buffer = cv2.imencode(".jpg", frame)
        img_base64 = base64.b64encode(buffer).decode()

        payload_alerta = {
            "imagen": f"{img_base64}",
            "casco": casco,
            "chaleco": chaleco,
            "guantes": guantes,
            "confianza": round(max_conf, 2),
            "archivo": ruta
        }

        try:
           requests.post(NODE_RED_URL, json=payload_alerta, timeout=2)
        except:
            pass

        ultimo_envio_alerta = ahora

    cv2.imshow("Sistema EPP - YOLO + Dahua", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break

# ================= CIERRE =================
cap.release()
cv2.destroyAllWindows()
mqtt_client.loop_stop()
mqtt_client.disconnect()
print("Sistema detenido")
```

Anexo 4: Programación Fuciton 1 para lectura de datos que llegados desde sistema de visión artificial.

```
// ===== GUARDAR JSON ORIGINAL (CLAVE) =====
msg.payload_original = msg.payload;

// ===== LEER DATOS =====
let casco   = msg.payload.casco  ?? "SIN DEFINIR";
let chaleco = msg.payload.chaleco ?? "SIN DEFINIR";
let guantes = msg.payload.guantes ?? "SIN DEFINIR";
let conf    = msg.payload.confianza ?? "--";

// ===== TEXTO  PARA DEBUG =====
msg.payload =
  "CASCO: " + casco + " (" + conf + ")\n" +
  "CHALECO: " + chaleco + "\n" +
  "GUANTES: " + guantes;
```

```
// NO borrar msg.req ni msg.res
return msg;
```

Programación Fuction 2

```
// Espera texto tipo:
// "CASCO: NO ..."
// "CASCO: SI ..."

let texto = msg.payload;

// Si falta algún EPP → ALERTA
if (
    texto.includes("CASCO: NO") ||
    texto.includes("CHALECO: NO") ||
    texto.includes("GUANTES: NO")
) {
    msg.payload = "ON";
} else {
    msg.payload = "OFF";
}

return msg;
```

Anexo 5: Diseño de serigrafia donde consta el título del sistema de detección de EPP desarrollado.

Figura 82

Diseño de serigrafia instalado en sistema de detección

Anexo 6: Imagen de sistema de detección de equipos de protección personal

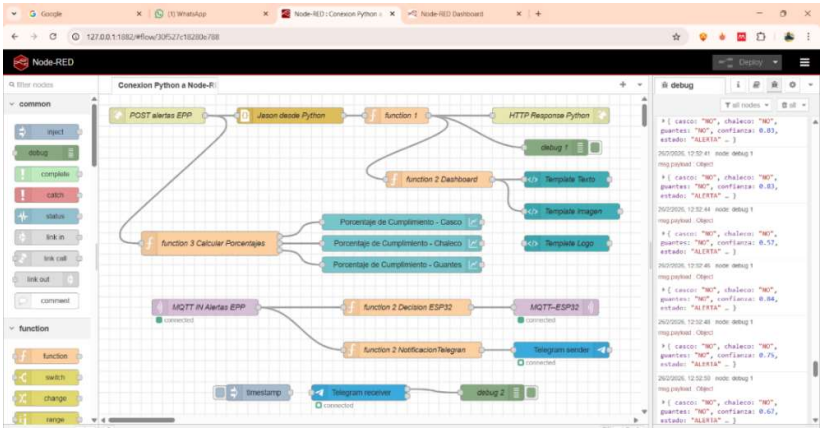
Figura 83

Sistema de detección de EPP

Anexo 7: Captura de Node-Red donde se visualiza resultados.

Figura 84

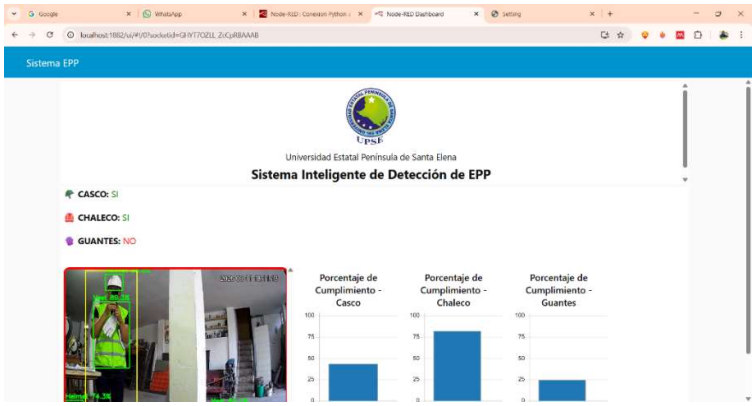
Flujo de nodos en Node-RED



Anexo 8: Dashboard donde se muestra alertas

Figura 85

Monitoreo en tiempo real de detección de EPP



Anexo 9: Programación ESP32 para activación de alertas físicas.

```
#include <WiFi.h>
#include <PubSubClient.h>
#define RELAY_PIN  2
#define WATCHDOG_TIMEOUT 5000
#define MQTT_RETRY_TIME 3000
#define WIFI_RETRY_TIME 5000
const char* ssid = "RED_LOCAL";
const char* password = "********";
const char* mqtt_server = "192.168.100.55";
const int   mqtt_port   = 1883;
const char* mqtt_topic  = "epp/esp32cmd"
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMqttTime = 0;
unsigned long lastMqttTry  = 0;
unsigned long lastWifiTry  = 0;
void callback(char* topic, byte* payload, unsigned int length) {
  String msg = "";
  for (unsigned int i = 0; i < length; i++) msg += (char)payload[i];
```

```
  msg.trim();
  Serial.print("MQTT recibido: ");
  Serial.println(msg);
  lastMqttTime = millis();
  if (msg == "ON") {
    digitalWrite(RELAY_PIN, HIGH);
    Serial.println(" RELÉ ENCENDIDO");
  } else if (msg == "OFF") {
    digitalWrite(RELAY_PIN, LOW);
    Serial.println(" RELÉ APAGADO");
  }
}
void checkWiFi() {
  if (WiFi.status() == WL_CONNECTED) return;

  unsigned long now = millis();
  if (now - lastWifiTry < WIFI_RETRY_TIME) return;
  lastWifiTry = now;
  Serial.println(" Reconectando WiFi...");
  WiFi.disconnect();
  WiFi.begin(ssid, password);
}
void checkMQTT() {
  if (WiFi.status() != WL_CONNECTED) return;
  if (client.connected()) return;
  unsigned long now = millis();
  if (now - lastMqttTry < MQTT_RETRY_TIME) return;
  lastMqttTry = now;
  Serial.print(" Conectando a MQTT...");
  if (client.connect("ESP32_EPP")) {
    Serial.println("  OK");
    client.subscribe(mqtt_topic);
    lastMqttTime = millis();
  } else {
    Serial.print("  rc=");
    Serial.println(client.state());
void setup() {
  Serial.begin(115200);
```

```
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
Serial.println(" ESP32 iniciado");
}void loop() {
  checkWiFi();
  checkMQTT();
  client.loop();  // ← CORRECCIÓN CLAVE
  if (millis() - lastMqttTime > WATCHDOG_TIMEOUT) {
    digitalWrite(RELAY_PIN, LOW)}
```

Anexo 10 Diagrama de flujo para comunicación MQTT a Node-RED y activación de alertas físicas.

Figura 86

Diagrama de flujo para comunicación MQTT y activación de alertas físicas