



UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

FACSISTEL

**INGENIERÍA EN ELECTRÓNICA Y
AUTOMATIZACIÓN**

TRABAJO DE INTEGRACIÓN CURRICULAR

**Sistema autónomo para detección de
presencia humana en situaciones anómalas en
espacios cerrados usando un dron con visión
artificial**

Carlos Joel Conforme Suárez

Dirigido por:
Ing. Junior Figueroa, M.Sc.

La Libertad - 2026

DEDICATORIA

Este trabajo, en primer lugar, se lo dedico a Dios, quien supo guiarme por el buen camino y darme salud, sabiduría y la fortaleza necesaria para continuar aun cuando había obstáculos.

A mis padres, por creer siempre en mí, por el esfuerzo que realizan todos los días para darme siempre lo mejor, por sus consejos, que me ayudaron a convertirme en una mejor persona, y por su apoyo incondicional durante toda mi formación.

A mis hermanas, Génesis, Allison y Valentina, por estar presentes en los buenos y malos momentos; por escucharme y animarme a seguir adelante. Las llevo siempre en mi corazón.

A mi novia, Domenica, por su amor, paciencia y compañía durante todo este proceso; por acompañarme en las noches difíciles y darme fuerzas para continuar; por confiar siempre en mí y motivarme hasta culminar esta hermosa etapa.

Con mucho cariño, les dedico este logro, porque fueron mi mayor apoyo y una parte fundamental de este camino.

AGRADECIMIENTO

Agradezco a Dios por darme la salud, la paciencia y la fuerza necesarias para seguir adelante y poder culminar esta etapa tan importante de mi vida.

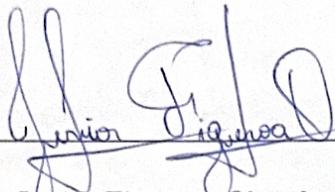
A mis padres, por todo el esfuerzo que han hecho para apoyarme durante mi formación. Gracias por confiar en mí, por sus consejos y por estar presentes en cada momento. A mis hermanas y a mi novia, Domenica, por acompañarme, comprender el tiempo que dediqué a este proyecto y animarme cuando las cosas se volvían difíciles.

A mi tutor, por su tiempo, orientación y observaciones durante el desarrollo de este trabajo. Sus recomendaciones fueron de gran ayuda para corregir y mejorar cada apartado. También agradezco a los docentes de la carrera por los conocimientos y experiencias compartidos a lo largo de estos años.

Finalmente, agradezco a la Universidad Estatal Península de Santa Elena por abrirme sus puertas y brindarme la oportunidad de crecer tanto en lo académico como en lo personal.

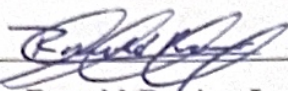
APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación denominado: **"Sistema autónomo para detección de presencia humana en situaciones anómalas en espacios cerrados usando un dron con visión artificial"**, elaborado por el estudiante **Carlos Joel Conforme Suárez**, de la carrera de **Ingeniería en electrónica y automatización** de la Universidad Estatal Península de Santa Elena, declaro que luego de haber orientado, estudiado y revisado, lo apruebo en todas sus partes y autorizo al estudiante que inicie los trámites legales correspondientes.

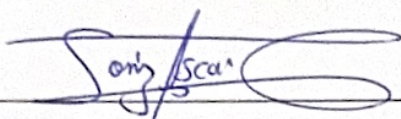


Ing. Junior Figueroa Olmedo, M.Sc.
Tutor

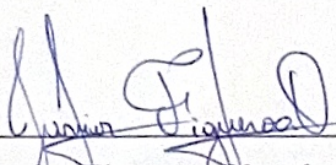
TRIBUNAL DE GRADO



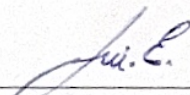
Ing. Ronald Riquira Jurado, Ph.D.
DIRECTOR DE CARRERA



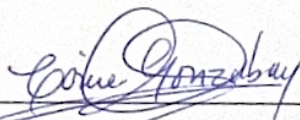
Ing. Óscar W. Gómez Morales, Ph.D.
DOCENTE ESPECIALISTA



Ing. Junior Figueroa Olmedo, Mgtr.
TUTOR



Ing. Luis Chuquimarca Jiménez, PhD.
DOCENTE GUÍA UIC



Ing. Corina Gonzabay De La A, Mgtr.
SECRETARIA DEL TRIBUNAL

Resumen

En este trabajo se desarrolló un sistema de monitoreo para identificar situaciones anómalas en espacios interiores mediante un dron DJI Tello y técnicas de visión artificial. El video captado por la cámara del dron se transmite por medio de una conexión Wi-Fi hacia una computadora, donde cada fotograma es procesado en Python. Para el tratamiento de las imágenes se utilizó OpenCV, mientras que la detección se realizó con un modelo YOLOv8 entrenado para reconocer las clases caída, comida, dormir y teléfono.

El sistema cuenta con una interfaz hombre-máquina que permite observar el video en tiempo real, consultar datos básicos de telemetría, visualizar las detecciones y conocer el nivel de confianza entregado por el modelo. Además, se incorporó un módulo de alertas mediante WhatsApp, el cual envía un mensaje con la clase detectada, la confianza, la fecha y la hora del evento. De forma paralela, el programa almacena una captura de la anomalía en la computadora.

El desarrollo del proyecto incluyó la revisión de información relacionada con drones y visión artificial, la preparación y etiquetado del conjunto de imágenes, el entrenamiento del modelo y la realización de pruebas en un entorno interior controlado. Durante la validación, el modelo alcanzó un valor global de mAP@0.5 de 0.839 y un valor máximo de F1 de 0.75 con un umbral de confianza aproximado de 0.70. Para el funcionamiento del sistema se estableció un umbral de 0.65, con el propósito de trabajar con detecciones más seguras y reducir alertas generadas por predicciones poco confiables.

Las pruebas permitieron comprobar que el DJI Tello, el modelo YOLOv8, la interfaz gráfica y el módulo de notificaciones podían funcionar de manera integrada. Entre las principales limitaciones observadas se encontraron las variaciones de iluminación, el ángulo de la cámara, la estabilidad de la conexión Wi-Fi y el tiempo de autonomía de la batería. Con base en los resultados obtenidos, se determinó que el sistema puede servir como una herramienta de apoyo para la supervisión de anomalías en tiempo real. Sin embargo, para una aplicación en entornos reales sería conveniente ampliar el conjunto de datos y realizar pruebas bajo diferentes condiciones de operación.

Palabras clave: Detección de anomalías, DJI Tello, interfaz hombre-máquina, visión artificial, WhatsApp.

Abstract

This study presents the development of a monitoring system designed to identify anomalous situations in indoor environments using a DJI Tello drone and computer vision techniques. The video captured by the drone's onboard camera is transmitted through a Wi-Fi connection to a computer, where each frame is processed in Python. OpenCV was used for image processing, while detection was carried out using a YOLOv8 model trained to recognize the classes Fall, Eating, Sleeping, and Phone Use.

The system includes a human-machine interface that displays the live video stream, basic telemetry data, detected events, and the confidence level provided by the model. A WhatsApp alert module was also incorporated to send a message containing the detected class, confidence level, date, and time of the event. At the same time, the program stores an image of the detected anomaly on the computer.

The project involved reviewing information related to drones and computer vision, preparing and labeling the image dataset, training the model, and conducting tests in a controlled indoor environment. During validation, the model achieved an overall mAP@0.5 of 0.839 and a maximum F1 score of 0.75 at an approximate confidence threshold of 0.70. For system operation, a confidence threshold of 0.65 was selected to prioritize more reliable detections and reduce alerts caused by low-confidence predictions.

The tests confirmed that the DJI Tello, the YOLOv8 model, the graphical interface, and the notification module could operate as an integrated system. The main limitations observed were changes in lighting conditions, camera angle, Wi-Fi connection stability, and battery life. Based on the results, the proposed system can serve as a support tool for real-time anomaly monitoring. However, before being used in real-world environments, the dataset should be expanded and further tests should be carried out under different operating conditions.

Keywords: Anomaly detection, computer vision, DJI Tello, human-machine interface, WhatsApp.

Índice

Índice de figuras	8
Índice de cuadros	10
Nomenclatura y abreviaturas	11
1. Introducción	13
1.1. Justificación	13
1.2. Panorama actual	14
1.3. Objetivos	15
1.3.1. Objetivo general	15
1.3.2. Objetivos específicos	15
1.4. Fundamentos teóricos	16
1.4.1. Robótica aérea	16
1.4.2. Vehículos aéreos no tripulados	17
1.4.3. Componentes de un dron cuadricóptero	18
1.4.4. Procesamiento de imágenes	22
1.4.5. Visión artificial aplicada a UAVs	25
1.4.6. Red neuronal convolucional YOLO (You only look on-	
ce)	27
Evolución de las versiones de YOLO:	27
Preprocesamiento y extracción de características:	29
Fusión multiescala y cabeza de detección:	29
Salida del modelo:	29
Función de pérdida:	30
Posprocesamiento y supresión de no máximos:	30
Ventajas técnicas:	31
1.4.7. Detección de anomalías visuales en UAVs	31
1.4.8. Operación de UAVs en espacios cerrados	32
1.4.9. Interfaces hombre-máquina (HMI)	34
1.4.10. Lenguajes de programación aplicados a UAVs	35
Python:	35
MATLAB (Matrix Laboratory):	36
Scratch:	36
1.4.11. Sistema de comunicación	37
Cable USB:	37
Bluetooth:	38
Wi-Fi:	38

1.4.12. Seguridad operacional y normativas legales para UAVs	
en espacios interiores	38
México: NOM-107-SCT3-2019	39
Unión Europea: Reglamentos (UE) 2019/945 y 2019/947	39
Argentina: Resolución ANAC 550/2025	39
Ecuador: RDAC 101	40
1.5. Marco contextual	40
2. Metodología del proyecto y diseño experimental	43
2.1. Metodología del proyecto	43
2.1.1. Metodología exploratoria	43
2.1.2. Metodología descriptiva	43
2.1.3. Metodología experimental	44
2.1.4. Metodología cuantitativa	44
2.1.5. Metodología de estudio de caso	44
2.1.6. Metodología diagnóstica	45
2.1.7. Descripción del proyecto	45
2.2. Componentes de la propuesta	46
2.2.1. Componentes físicos	46
Dron DJI Tello:	46
Sistema de posicionamiento visual:	48
Motores:	49
Hélices y funcionamiento del dron:	50
2.2.2. Componentes lógicos	53
Entorno de desarrollo PyCharm:	53
Librería DJITelloPy:	55
Librería NumPy (Numerical Python):	57
Librería OpenCV (Open source computer vision library):	58
2.3. Diseño experimental	60
2.3.1. Diseño e implementación del hardware	61
2.3.2. Diseño del sistema de comunicación	63
2.3.3. Diseño del software	64
Selección del entorno de desarrollo:	64
Arquitectura del software:	66
2.3.4. Sistema de visión artificial para la detección de personas	67
2.3.5. Diseño del sistema de control del dron	68
2.3.6. Diseño de la interfaz de usuario	70
2.4. Desarrollo experimental	72
2.4.1. Implementación del sistema de adquisición de datos . .	72
2.4.2. Preparación del dataset y entrenamiento del modelo . .	76
2.4.3. Implementación del sistema de procesamiento	79

2.4.4.	Implementación del sistema de detección de anomalías	81
2.4.5.	Implementación del sistema de alertas mediante WhatsApp	83
	Obtención de la clave API:	84
	Programación del envío de la alerta:	84
2.4.6.	Implementación de la interfaz gráfica	87
2.4.7.	Configuración del sistema	88
2.4.8.	Condiciones de implementación y limitaciones del sistema	90
3.	Resultados y conclusiones	92
3.1.	Resultados	92
3.1.1.	Resultados del entrenamiento del modelo	92
3.1.2.	Resultados de validación del modelo YOLOv8	93
	Curva F1-Confidence:	93
	Curva Precision-Recall:	94
	Matriz de confusión normalizada:	95
	Resumen de los resultados de validación:	97
3.1.3.	Resultados de detección de anomalías	97
3.1.4.	Resultados del sistema de alertas mediante WhatsApp	99
3.1.5.	Resultados del sistema de monitoreo	101
3.1.6.	Resultados de las pruebas de vuelo	103
3.1.7.	Evaluación general del sistema	104
3.2.	Conclusiones y recomendaciones	106
3.2.1.	Conclusiones	106
3.2.2.	Recomendaciones	106
	Referencias	108

Índice de figuras

1. Clasificación de la robótica aérea.	17
2. UAV (Vehículo aéreo no tripulado)	18
3. Partes y componentes principales de un dron tipo cuadricóptero.	19
4. Esquema de funcionamiento de un controlador de motor eléctrico.	20
5. Diagrama de conexión de un controlador de vuelo	21
6. Representación gráfica del proceso de segmentación mediante superpíxeles.	24
7. Arquitectura de redes convolucionales	26
8. Funcionamiento general del algoritmo YOLO para la detección de objetos en imágenes.	31
9. Esquema de un dron en un espacio cerrado.	33
10. Dron FIXAR	34
11. Sistema de comunicación	37
12. Protocolo de comunicación Wi-Fi	38
13. Laboratorio de electrónica ubicado en la UPSE	41
14. Esquema general del sistema de detección de presencia humana.	46
15. Dron DJI Tello	47
16. Sistema de posicionamiento visual en el dron DJI Tello	49
17. Motores sin escobillas del dron DJI Tello	50
18. Movimiento ascendente de un dron.	51
19. Movimiento descendente de un dron.	51
20. Desplazamiento a la izquierda de un dron.	51
21. Desplazamiento a la derecha de un dron.	52
22. Avance de un dron.	52
23. Retroceso de un dron.	52
24. Rotación en sentido horario.	53
25. Rotación en sentido antihorario.	53
26. Áreas de trabajo en PyCharm	54
27. Arreglos en 1D, 2D y 3D.	58
28. Arquitectura general del sistema de detección de presencia humana mediante dron y visión artificial	61
29. Visualización y reconocimiento de elementos conectados al dron	62
30. Conexión entre ordenador y red Wi-Fi del dron DJI Tello.	64
31. Sistema de control	67
32. Arquitectura interna del sistema de control del dron Tello.	70
33. Diseño general de la interfaz de usuario.	72
34. Entorno utilizado para las pruebas de adquisición de datos.	73

35. Conexión inalámbrica entre el dron DJI Tello y la computadora.	74
36. Visualización en tiempo real de la cámara del DJI Tello mediante OpenCV.	75
37. Ejemplos de imágenes capturadas para la construcción del dataset.	76
38. Imágenes utilizadas para la construcción del dataset.	77
39. Proceso de etiquetado de imágenes en formato YOLO.	78
40. Estructura general del dataset utilizado para el entrenamiento del modelo.	78
41. Proceso de entrenamiento del modelo YOLOv8.	79
42. Procesamiento inicial de fotogramas mediante OpenCV.	80
43. Aplicación del modelo YOLOv8 sobre los fotogramas procesados.	80
44. Visualización de anomalías detectadas mediante cuadros delimitadores.	81
45. Detección de anomalías en tiempo real mediante YOLOv8.	82
46. Código utilizado para el envío de alertas mediante WhatsApp	86
47. Esquema general de la interfaz gráfica propuesta para el sistema de monitoreo.	87
48. Implementación de la interfaz gráfica desarrollada en Python.	88
49. Configuración general del sistema de detección implementado.	90
50. Pruebas realizadas durante la implementación experimental del sistema.	91
51. Resultados del entrenamiento del modelo YOLOv8.	92
52. Curva F1 en función del umbral de confianza	94
53. Curva Precision-Recall del modelo YOLOv8	95
54. Matriz de confusión normalizada del modelo YOLOv8	96
55. Detección de anomalías mediante regiones de interés (ROI).	98
56. Mensaje de alerta recibido mediante WhatsApp	100
57. Interfaz gráfica del sistema durante el monitoreo en tiempo real.	102
58. Pruebas experimentales realizadas con el dron DJI Tello.	104
59. Reporte de compilatio	116

Índice de cuadros

2.	Evolución de las principales versiones de YOLO	28
3.	<i>Especificaciones técnicas del Dron DJI Tello</i> [72, 74, 73]	48
4.	<i>Especificaciones de los motores del dron Tello</i> [76]	50
5.	<i>Comandos del dron DJI Tello</i> [Autoría propia]	56
6.	Comandos de OpenCV [Autoría propia]	59
7.	Comparación de entornos de desarrollo para Python	65
8.	Resultados generales del entrenamiento del modelo	93
9.	Métricas de validación del modelo YOLOv8	97
10.	Resultados de detección por tipo de anomalía	98
11.	Funciones implementadas en la interfaz gráfica	103
12.	Resultados de pruebas de vuelo	104
13.	Evaluación general del sistema	105

Nomenclatura y Abreviaturas

A continuación, se presentan las nomenclaturas y abreviaturas utilizadas en el presente trabajo.

Símbolo	Significado
A	Área.
V	Voltaje.
I	Corriente eléctrica.
f	Frecuencia.
Δt	Intervalo de tiempo.
μ	Permeabilidad magnética.
ε_0	Permitividad eléctrica del vacío.
UAV	<i>Unmanned Aerial Vehicle</i> ; vehículo aéreo no tripulado.
UAS	<i>Unmanned Aircraft System</i> ; sistema de aeronave no tripulada.
RPA	<i>Remotely Piloted Aircraft</i> ; aeronave pilotada a distancia.
RPAS	<i>Remotely Piloted Aircraft System</i> ; sistema de aeronave pilotada a distancia.
HMI	<i>Human-Machine Interface</i> ; interfaz hombre-máquina.
CNN	<i>Convolutional Neural Network</i> ; red neuronal convolucional.
YOLO	<i>You Only Look Once</i> ; familia de modelos para la detección de objetos.
ROI	<i>Region of Interest</i> ; región de interés.
FPS	<i>Frames per Second</i> ; fotogramas por segundo.
IMU	<i>Inertial Measurement Unit</i> ; unidad de medición inercial.
GPS	<i>Global Positioning System</i> ; sistema de posicionamiento global.

Símbolo	Significado
SDK	<i>Software Development Kit</i> ; kit de desarrollo de software.
UDP	<i>User Datagram Protocol</i> ; protocolo de datagramas de usuario.
API	<i>Application Programming Interface</i> ; interfaz de programación de aplicaciones.
IoU	<i>Intersection over Union</i> ; intersección sobre unión.
mAP	<i>Mean Average Precision</i> ; precisión promedio media.
NMS	<i>Non-Maximum Suppression</i> ; supresión de máximos no relevantes.
Wi-Fi	Tecnología de comunicación inalámbrica utilizada para conectar el DJI Tello con la computadora.

1. Introducción

1.1. Justificación

La supervisión de fábricas, hospitales, almacenes, túneles y otros espacios interiores o críticos, donde las estructuras limitan la visibilidad, la accesibilidad de las personas y la capacidad de respuesta ante emergencias, constituye un reto. Por esta razón, existen proyectos orientados a mejorar la seguridad y la eficiencia en estos espacios mediante el uso de tecnologías inteligentes capaces de operar de manera autónoma y precisa.

Las limitaciones de los sistemas de vigilancia tradicionales basados en cámaras fijas o sensores estáticos incluyen los ángulos muertos, los obstáculos presentes en el espacio físico, la falta de adaptabilidad ante los cambios del entorno, el coste de la infraestructura y la necesidad de intervención humana. Además, estos sistemas pueden presentar dificultades para reaccionar oportunamente ante eventos inesperados, como una intrusión, un accidente laboral, un incidente de seguridad o una emergencia médica.

En este contexto, los drones equipados con visión artificial constituyen una alternativa móvil para la inspección y el monitoreo. Su capacidad para cambiar el punto de vista permite reducir algunos de los ángulos muertos de las cámaras fijas, mientras que los modelos de reconocimiento de acciones posibilitan el análisis de conductas humanas capturadas desde plataformas aéreas, como señalan Abbas y Jalal [1]. No obstante, en espacios interiores, la ausencia de señal GPS, la proximidad de obstáculos y las restricciones computacionales exigen un diseño cuidadoso del sistema, tal como indican Khosiawan y Nielsen [2].

El sistema propuesto también integra la detección de objetos, la transmisión de video y una interfaz de control, con el propósito de identificar situaciones anómalas, como caídas, uso de teléfonos móviles, ingesta de alimentos, inactividad. Este trabajo es consistente con el creciente uso de drones comerciales en actividades de inspección y monitoreo, como lo indica el informe de mercado de Grand View Research [3]; sin embargo, su alcance se limita a pruebas experimentales realizadas en espacios interiores.

Por lo tanto, el presente trabajo tiene como finalidad desarrollar una plataforma de supervisión de bajo coste y flexible, centrada en la detección automática de situaciones anómalas y en la generación de las alertas correspondientes. No obstante, el sistema no contempla un nivel de autonomía de navegación completamente implementado y validado, debido a que el desplazamiento del dron se realiza bajo la supervisión del operador.

1.2. Panorama actual

La integración con la visión artificial ha incrementado el uso de los drones en aplicaciones de control y vigilancia. Grand View Research [3] estimó que el mercado mundial de drones comerciales alcanzaría los USD 30,02 mil millones en 2024 y que crecería entre 2025 y 2030 a una tasa anual del 10,6 %. Para el mercado de drones de interiores, Delvens [4] estima un crecimiento de USD 11.210 millones en 2024 a USD 22.124,6 millones en 2031. Sin embargo, estas cifras corresponden a una previsión sobre la adopción de esta tecnología y no a los resultados de un estudio técnico realizado sobre un prototipo específico de dron.

Recientemente, se ha determinado que el aprendizaje profundo puede utilizarse para detectar personas, acciones y anomalías en imágenes aéreas. Sin embargo, su rendimiento varía en función del conjunto de datos empleado, las condiciones de iluminación, la resolución de las imágenes, el ángulo de captura y el hardware utilizado para la inferencia, por lo que no es posible establecer un rendimiento general aplicable a todos los sistemas [1, 5].

La navegación en espacios interiores presenta diversos retos, como la ausencia de señal GPS, el espacio limitado, la presencia de ruido y la necesidad de estimar continuamente la posición del vehículo aéreo. Khosiawan y Nielsen [2] presentan una arquitectura modular que coordina los procesos de localización, control, planificación y asignación de tareas. Por su parte, Sindhwani et al. [6] presentan la aplicación de métodos de detección no supervisada para identificar anomalías durante las misiones realizadas por drones.

Además, se han desarrollado prototipos de bajo coste destinados a la detección de objetos y a diferentes aplicaciones de inspección. Por ejemplo, Valdovinos Miranda [7] estudió un enjambre de vehículos aéreos no tripulados de bajo coste con capacidad de detección de objetos en tiempo real, mientras que Calin [8] analizó la aplicación de drones en los procesos de inspección y gestión de la seguridad en buques. Los antecedentes mencionados permiten establecer la factibilidad de utilizar plataformas de dimensiones reducidas, siempre que sus limitaciones operativas sean identificadas, evaluadas y documentadas adecuadamente.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un sistema autónomo de monitoreo basado en un dron equipado con visión artificial que, mediante la implementación de algoritmos de procesamiento de imágenes y técnicas de navegación autónoma, permita la detección de presencia humana en espacios cerrados para la optimización de la seguridad y la eficiencia en la supervisión de entornos comerciales.

1.3.2. Objetivos específicos

- Implementar un algoritmo de visión artificial que detecte y rastree en tiempo real la presencia humana en espacios cerrados.
- Desarrollar un sistema de generación y procesamiento de datos mediante un software específico para alimentar y optimizar el algoritmo de detección.
- Desarrollar una interfaz hombre-máquina (HMI) para la supervisión y control de la navegación autónoma del dron en entornos confinados.
- Validar el rendimiento del sistema mediante pruebas en escenarios controlados, evaluando la precisión y eficiencia del monitoreo.

1.4. Fundamentos teóricos

1.4.1. Robótica aérea

La robótica aérea es una disciplina enfocada en el desarrollo de plataformas capaces de operar en el espacio aéreo sin necesidad de un piloto a bordo. Estas plataformas, comúnmente conocidas como drones o vehículos aéreos no tripulados (UAV), se han convertido en herramientas versátiles para tareas como inspección, vigilancia, monitoreo ambiental y asistencia en emergencias. Su capacidad de acceso a zonas de difícil alcance y la integración de sensores, cámaras y algoritmos inteligentes permiten optimizar procesos con mayor seguridad y eficiencia [9].

Tal como explica Umiles Group [10], desde el punto de vista técnico y normativo, existen distintas categorías dentro de esta tecnología, que varían según el grado de autonomía y la infraestructura asociada:

- **UAV (Unmanned aerial vehicle):** Vehículo no tripulado que puede ser operado de forma remota o seguir trayectorias preprogramadas.
- **RPA (Remotely piloted aircraft):** Tipo de UAV que necesita intervención humana continua mediante una estación de control.
- **RPAS (Remotely piloted aircraft system):** Sistema completo compuesto por el dron, la estación de control y los enlaces de comunicación.
- **UAS (Unmanned aircraft system):** Conjunto más amplio que incluye el UAV, sensores, software, subsistemas de control y telecomunicaciones.

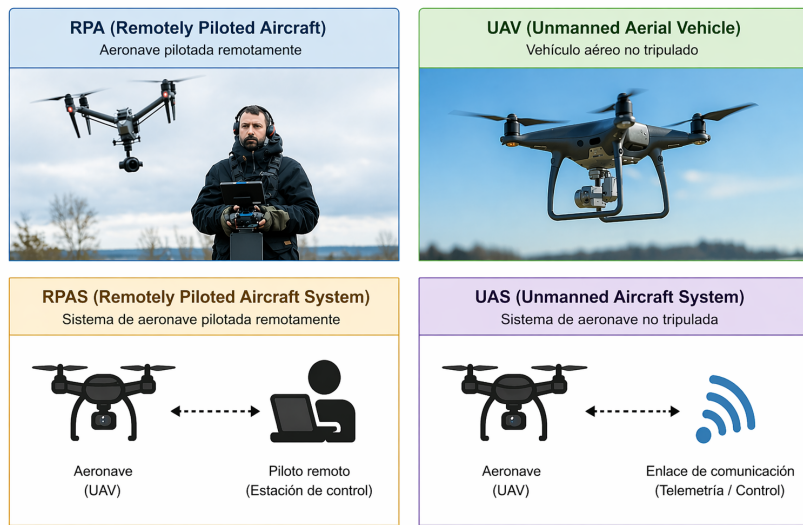


Figura 1. Clasificación de la robótica aérea.

Fuente: [10]

En cualquier caso, estas denominaciones no proporcionan por sí solas información sobre el nivel de autonomía de un UAS. Aunque una aeronave puede volar de forma automática o seguir una trayectoria preprogramada, la autonomía operacional también depende de las capacidades del sistema, el grado de supervisión humana y las condiciones del entorno de operación.

La integración de visión artificial en estas plataformas permite ampliar sus capacidades para realizar tareas de inspección. Silva Bohórquez et al. [9] presentan un robot aéreo capaz de efectuar tareas de inspección y vigilancia mediante visión artificial, lo que demuestra la posibilidad de combinar la percepción visual con la movilidad de una plataforma aérea.

1.4.2. Vehículos aéreos no tripulados

Los vehículos aéreos no tripulados (UAVs, por sus siglas en inglés) son sistemas aéreos diseñados para volar sin la presencia directa de un piloto a bordo. Su operación puede ser completamente autónoma o guiada de manera remota mediante una estación de control en tierra. Estos dispositivos forman parte integral de la robótica aérea y se emplean en múltiples sectores, desde la agricultura y la cartografía hasta la seguridad, la investigación y la vigilancia de espacios interiores [10].

Un UAV está compuesto, entre otras cosas, por su estructura, el sistema de propulsión, el sistema de control de vuelo, el sistema de energía, los sensores, el sistema de comunicación y, dependiendo de la misión, la cámara o

algún otro tipo de carga útil, y su comportamiento depende de la interacción de todos estos subsistemas [11].

Por lo general, los UAV se dividen en tres categorías según su configuración: ala fija, ala rotatoria y configuración híbrida. Los multirrotores presentan la ventaja de proporcionar despegues verticales, vuelos estables y maniobrabilidad en entornos confinados, aunque su autonomía suele ser más baja que la de los vehículos de ala fija [12].

Para llevar a cabo estos experimentos, se utilizó un UAV en forma de cuadricóptero. Estos vehículos aéreos no tripulados para operar en interiores se caracterizan por su geometría compacta y su bajo peso, lo que les otorga una gran capacidad de maniobra. También se le puede incorporar cámara para observar el entorno, como se observa en la Figura 2.



Figura 2. UAV (Vehículo aéreo no tripulado)

Fuente: [13]

1.4.3. Componentes de un dron cuadricóptero

Los drones cuadricópteros están compuestos por una serie de elementos que garantizan su funcionamiento eficiente, estable y seguro en entornos interiores. La Figura 3 presenta de manera esquemática los principales componentes del sistema. A continuación, se describe cada uno de ellos:

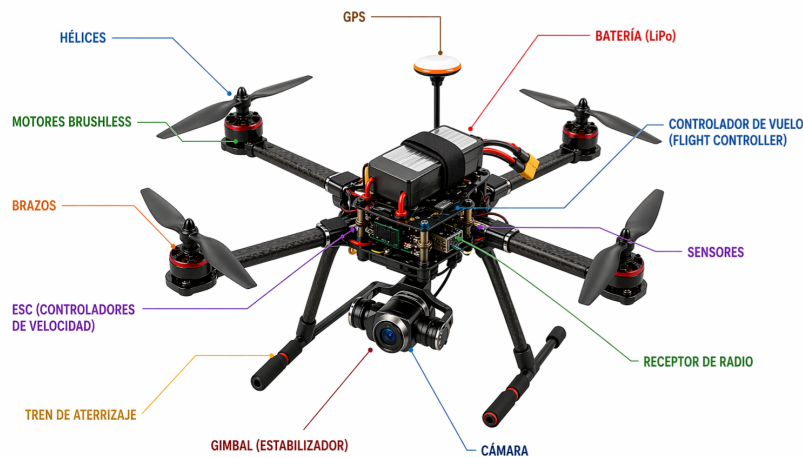


Figura 3. Partes y componentes principales de un dron tipo cuadricóptero.

Fuente: [13]

- **Estructura (chasis):** Es el armazón que sostiene todos los elementos del dron, incluyendo motores, hélices, controlador de vuelo, batería y sensores. Está diseñada para ser ligera y resistente, utilizando materiales como fibra de carbono, aluminio o plásticos técnicos, lo cual garantiza una buena relación peso-resistencia [14].
- **Hélices:** En un cuadricóptero, los pares de hélices giran en cada dirección (dos en sentido horario y dos en sentido antihorario) y las diferencias en la velocidad de rotación de las hélices se utilizan para controlar la elevación, el alabeo, el cabeceo y la guiñada, mientras que los pares aerodinámicos opuestos equilibran el vehículo [12].
- **Motores eléctricos:** Convierten la electricidad en movimiento. Los multirrotores pueden utilizar motores de corriente continua con escobillas o motores sin escobillas, en función de su tamaño, coste y potencia necesaria. Consideraciones para la selección del motor incluyen el voltaje, la corriente, el empuje, la eficiencia y la capacidad de trabajar con la hélice y el controlador de velocidad [11].
- **Controladores de velocidad (ESC):** Los Electronic Speed Controllers (ESC), o controladores electrónicos de velocidad, son dispositivos que regulan la velocidad de los motores eléctricos del dron. Reciben instrucciones del controlador de vuelo y ajustan la potencia de salida de cada motor, lo que permite un control adecuado de la orientación y el movimiento del cuadricóptero.

En sistemas con motores sin escobillas, el controlador electrónico de velocidad modula la energía suministrada a cada fase del motor a partir de las órdenes del controlador de vuelo. Algunos módulos incorporan regulación de voltaje para alimentar circuitos auxiliares. La Figura 4 presenta un esquema funcional de esta relación [15].

La elección del controlador de velocidad depende de la corriente máxima, el voltaje de la batería, el tipo de motor y el protocolo de control. Un dimensionamiento inadecuado puede reducir la eficiencia o comprometer la respuesta del sistema [11].

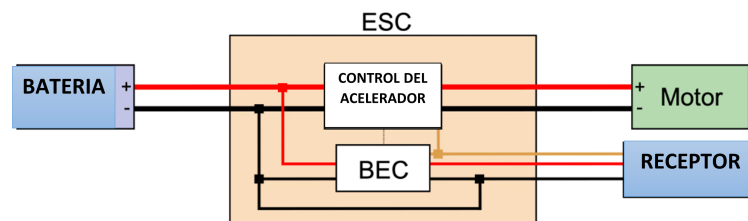


Figura 4. Esquema de funcionamiento de un controlador de motor eléctrico.

Fuente: [15]

- **Controlador de vuelo:** Es el componente central del sistema de control de un cuadricóptero, considerado como el “cerebro” del dron. Su función principal es procesar la información proveniente de los sensores (como el giroscopio, el acelerómetro, el magnetómetro y el barómetro) y calcular las correcciones necesarias para mantener el equilibrio, orientación y trayectoria deseada durante el vuelo.

El controlador de vuelo también recibe señales del piloto (ya sea desde un transmisor manual o mediante comandos automatizados) y las interpreta para ajustar dinámicamente la velocidad de los motores a través de los ESC. Además, se encarga de gestionar módulos adicionales como el GPS, módulos de telemetría o cámaras de visión artificial.

En la Figura 5 se muestra un ejemplo típico de conexión del controlador de vuelo, en el que se puede observar cómo se integran los ESC, la batería, los receptores y otros periféricos como transmisores de video y cámaras FPV [16].

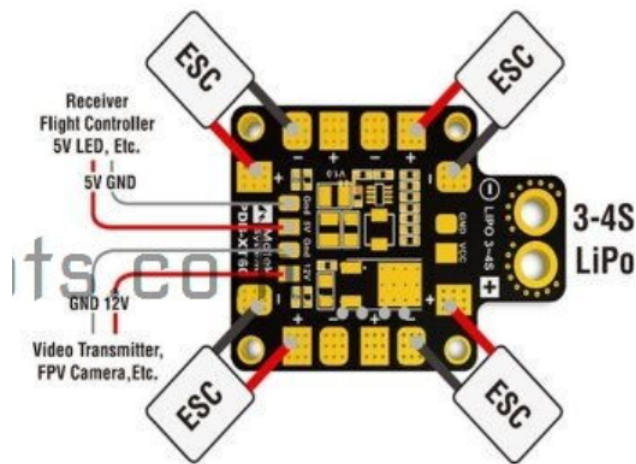


Figura 5. Diagrama de conexión de un controlador de vuelo

Fuente: [16]

- **Batería Li-Po:** Provee la energía necesaria para el funcionamiento del sistema. Estas baterías tienen alta densidad energética, bajo peso y son ampliamente utilizadas en aplicaciones aeronáuticas debido a su alto rendimiento [17].

Las tecnologías de almacenamiento utilizadas en UAV incluyen baterías Li-Po y Li-ion, además de soluciones híbridas y celdas de combustible. Las Li-Po ofrecen altas tasas de descarga; las Li-ion suelen proporcionar mayor densidad energética, aunque la elección final depende del perfil de potencia y de la misión [18].

El tiempo de vuelo depende de la capacidad de la batería, el consumo de los motores, la masa total, la temperatura y las maniobras realizadas. Por ello, la autonomía debe reportarse para cada plataforma y condición de prueba, en lugar de asumir un intervalo universal [18].

- **Receptor de control (Receiver):** El receptor es el componente que permite la comunicación unidireccional desde el transmisor (mando de control) hacia el dron. Su función principal es recibir las señales enviadas por el piloto y transmitir las al controlador de vuelo para ejecutar las maniobras correspondientes [19].

Las señales del transmisor son leídas por el receptor y enviadas al controlador de vuelo mediante PWM, PPM o buses seriales, según el tipo de enlace y la interoperabilidad del equipo [11].

- **Transmisor de video (VTX):** El transmisor de video, comúnmente conocido como VTX (Video Transmitter), es el componente encargado de enviar la señal de video capturada por la cámara del dron hacia un receptor en tierra, como unas gafas FPV o un monitor. Esta transmisión permite al operador tener una vista en primera persona (First Person View) del entorno del dron, lo cual es crucial para tareas de navegación, inspección o reconocimiento en espacios cerrados.

Los VTX varían en potencia de emisión (por ejemplo, 25 mW, 200 mW y 800 mW), lo que influye directamente en el alcance y la calidad de la señal. En ambientes interiores, es recomendable usar potencias moderadas para evitar interferencias. Además, estos dispositivos cuentan con múltiples canales y frecuencias configurables para facilitar la compatibilidad con diferentes sistemas receptores [20].

- **Cámara:** La cámara es el sensor encargado de capturar la información visual del entorno en el que opera el dron. Este componente es fundamental en aplicaciones que requieren percepción del entorno, como navegación autónoma, inspección, vigilancia y sistemas de visión artificial.

Las cámaras se diferencian por resolución, frecuencia de captura, campo de visión, rango dinámico y latencia. En procesamiento en tiempo real, la tasa de fotogramas y el tiempo de transferencia condicionan la capacidad de respuesta del sistema [21].

Ya sea que la señal se envíe a una estación terrestre o se procese a bordo, la cámara se convierte en la fuente de datos para los algoritmos de detección, clasificación, seguimiento y segmentación [21].

La selección de la cámara se basa en el FOV, la resolución, la latencia de la transmisión y la iluminación. En interiores, se prefieren las cámaras con alta sensibilidad a la luz y baja latencia.

1.4.4. Procesamiento de imágenes

El procesamiento de imágenes digitales es una disciplina de la ingeniería y las ciencias computacionales que se enfoca en la manipulación computacional de imágenes adquiridas a través de sensores o cámaras digitales. Su propósito es mejorar la calidad visual de las imágenes, extraer información relevante o automatizar tareas de análisis visual que tradicionalmente realizaba el ojo humano [22]. Esta técnica ha cobrado especial relevancia en aplicaciones de visión por computador, inteligencia artificial, robótica y sistemas autónomos.

A diferencia del procesamiento de señales en el dominio temporal, las imágenes son señales bidimensionales que contienen información espacial en forma de píxeles. Cada píxel representa una intensidad luminosa que, organizada en una matriz, forma la imagen completa. La capacidad de interpretar esa información y derivar conocimiento útil de ella depende de una correcta cadena de procesamiento digital [23].

El procesamiento digital de imágenes puede dividirse en una serie de etapas secuenciales, que son fundamentales para el análisis de la imagen:

- **Adquisición:** En esta fase se obtienen las imágenes a partir de las cámaras digitales o por medio de sensores RGB o de infrarrojos. En nuestro sistema, esta fase se realiza mediante la cámara del dron, que proporciona una secuencia de imágenes de forma continua y en tiempo real.
- **Preprocesado:** Las imágenes obtenidas pueden ser objeto de varios procesos para facilitar su análisis posterior, como la conversión de color (RGB a escala de grises), la corrección geométrica, la eliminación de ruido a través de filtros (gaussiano, mediana, etc.) o la normalización del tamaño. Estos procesos ayudan a mejorar la calidad de la imagen y a eliminar distorsiones que podrían dificultar su análisis.
- **Segmentación:** Esta etapa consiste en dividir la imagen en partes que pueden corresponder a contornos o grupos de objetos y separar el fondo de las regiones que se van a considerar. Por ejemplo, en el sistema desarrollado se puede aislar a la persona detectada.
- **Extracción de características:** A partir de cada una de las regiones resultantes de la segmentación, se pueden obtener características o atributos como bordes, esquinas, texturas, formas o patrones que serán utilizados por el algoritmo para clasificar o reconocer comportamientos.
- **Clasificación y análisis:** Los modelos de aprendizaje profundo aprenden representaciones jerárquicas a partir de los datos. Las redes neuronales convolucionales han demostrado un desempeño destacado en clasificación, detección y segmentación, debido a su capacidad para extraer patrones locales y combinarlos en estructuras más abstractas [25].

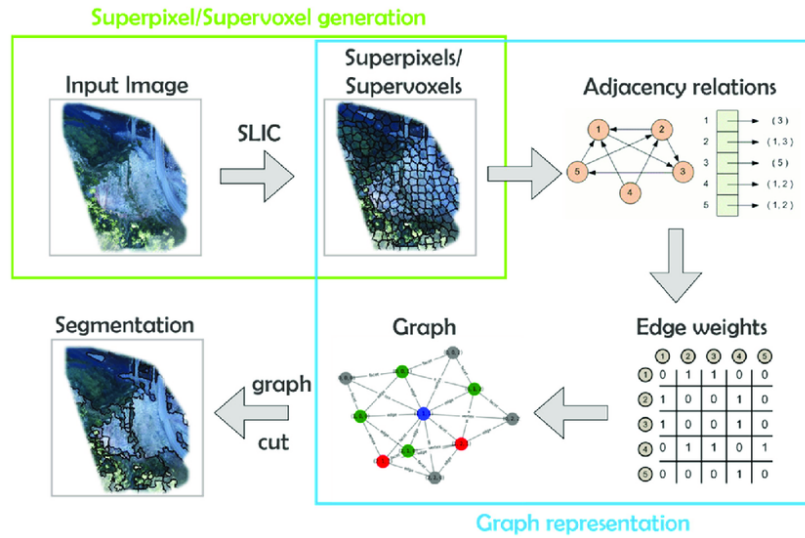


Figura 6. Representación gráfica del proceso de segmentación mediante superpíxeles.

Fuente: [26]

En el proceso de análisis de imágenes digitales, una de las etapas más importantes es la segmentación, ya que permite dividir la imagen en regiones significativas que facilitan la posterior extracción de información. Un enfoque ampliamente utilizado consiste en aplicar algoritmos de generación de superpíxeles, como el método SLIC (Simple linear iterative clustering), el cual agrupa píxeles vecinos con características similares para formar bloques coherentes dentro de la imagen.

Como se observa en la Figura 6, a partir de las regiones (superpíxeles) obtenidas en el paso anterior, se puede construir un grafo que represente la imagen. En este grafo, las regiones (superpíxeles) son los nodos y las aristas representan la cercanía entre cada uno de los nodos. A partir de esta representación, se puede aplicar teoría de grafos para analizar la topología de la imagen, donde el peso de cada arista representa la similitud entre regiones adyacentes.

En la siguiente fase se procede a la segmentación del grafo mediante técnicas como el Graph Cut. En esta etapa se separan las regiones de interés del resto de la imagen. Este tipo de segmentación es especialmente útil para la detección de objetos o personas en imágenes con fondos complejos, ya que permite al sistema concentrar su atención en las áreas relevantes y minimizar el impacto del ruido en el análisis de la imagen [26]. Además, el uso de procesamiento de imágenes en sistemas embebidos o móviles, como un dron, impone retos adicionales como la optimización de recursos, la latencia

y la precisión, lo que refuerza la importancia de elegir modelos eficientes y adecuados para la tarea, como el caso de YOLOv8.

1.4.5. Visión artificial aplicada a UAVs

El procesamiento de imágenes mediante visión artificial en UAV se utiliza en diversos tipos de aplicaciones, como la navegación autónoma, el seguimiento de objetivos o la vigilancia visual. Esto se consigue gracias a la utilización de cámaras y algoritmos de procesamiento de imágenes que permiten a los UAV captar su entorno y reaccionar ante él de manera autónoma.

Uno de los principales desafíos de la visión artificial aplicada a los UAV es la limitación de energía y de capacidad de cómputo, ya sea en el propio UAV o en la estación terrena. Los modelos compactos y las plataformas embebidas permiten ejecutar inferencia cerca de la fuente de datos, mientras que enfoques no supervisados pueden apoyar la identificación de desviaciones en misiones de vuelo [39, 6].

Las redes neuronales convolucionales (CNN) transforman progresivamente una imagen en representaciones de mayor nivel. Sus componentes habituales incluyen capas convolucionales, funciones de activación, operaciones de reducción y capas de salida adecuadas a la tarea [25]:

- Capas convolucionales, que aplican filtros o kernels a la imagen para extraer características como bordes, texturas u otros patrones.
- Capas de activación (comúnmente ReLU), que introducen la no linealidad necesaria para que el modelo pueda aprender relaciones complejas.
- Capas de agrupamiento (*pooling*), que reducen la dimensionalidad de los datos y conservan solo las características más relevantes para mejorar la eficiencia del cálculo.
- Capas completamente conectadas (*fully connected*), que conectan todas las neuronas de una capa con la siguiente para realizar tareas de clasificación o regresión.

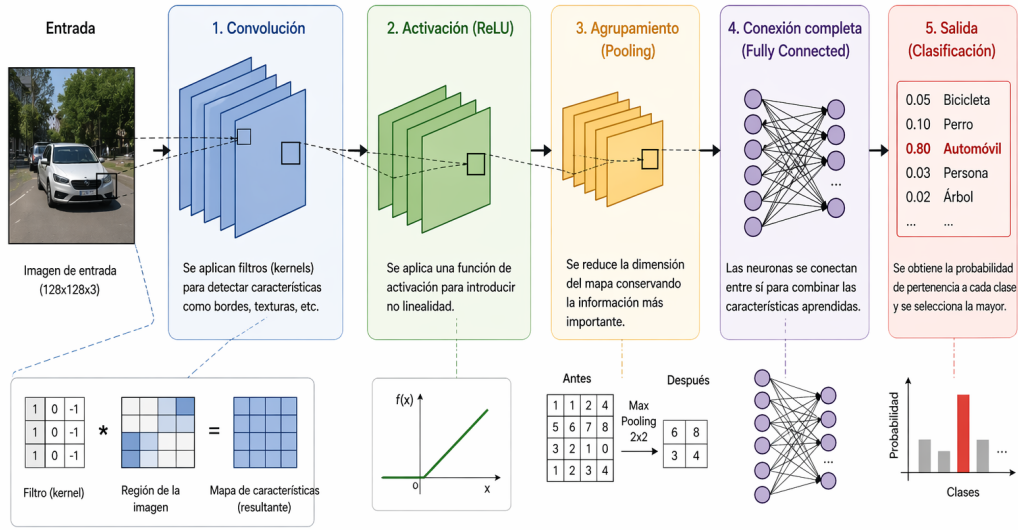


Figura 7. Arquitectura de redes convolucionales

Fuente: [28]

La Figura 7 ilustra el flujo de procesamiento típico en una CNN. El proceso inicia con la capa de entrada, donde se recibe la imagen cruda (matriz de píxeles). A continuación, se aplican una o más capas de convolución, encargadas de extraer características locales como bordes, texturas o patrones, utilizando filtros entrenables que barren la imagen.

Después de cada operación de convolución, es común aplicar una capa de agrupamiento (*pooling*) que reduzca la dimensionalidad, al tiempo que conserva las características más relevantes para que el modelo sea más eficiente. Este patrón de convolución y reducción se repite varias veces, de modo que se obtiene una jerarquía de características de la imagen. Estas características se aplanan y se pasan a través de una capa completamente conectada para la clasificación, que predice la categoría de la imagen de entrada en función de la probabilidad de cada clase [25, 28].

Este tipo de arquitectura ha demostrado ser muy eficaz en las tareas de clasificación, detección y segmentación de imágenes dentro de los sistemas de visión artificial como el desarrollado en este proyecto [25].

En el artículo de Joshi et al. [5], los autores comparan distintas técnicas de ML y DL en un caso de detección de anomalías en imágenes de drones. Este artículo muestra cómo las CNN y otros clasificadores pueden facilitar el análisis de imágenes aéreas, a pesar de que su rendimiento depende de la calidad de los datos y de las clases.

Wang et al. [29] propusieron GTAF, una GNN con mecanismos de aten-

ción para la detección de anomalías en datos multivariantes de sistemas no tripulados. Con la fusión de canales se pueden modelar las relaciones temporales y priorizar las variables, aunque debe ser adaptada para funcionar con video.

Otros trabajos [30] han demostrado que la visión artificial es importante en los sistemas de enjambre, ya que cada dron tiene que ser capaz de percibir su entorno e interactuar con otros miembros del enjambre para alcanzar un objetivo común. La visión se convierte así en un componente importante para la coordinación del comportamiento colectivo y la respuesta a las perturbaciones del entorno en los sistemas de enjambre.

Los modelos más complejos, como YOLO (You only look once), han sido diseñados para que la detección de objetos se realice en tiempo real y con una alta precisión. Dado que estos modelos están orientados a computadoras de bajo rendimiento, pueden instalarse fácilmente en drones.

1.4.6. Red neuronal convolucional YOLO (You only look once)

YOLO (You only look once) es una familia de detectores de objetos de una sola etapa. El modelo recibe una imagen, extrae características y predice directamente la ubicación de los objetos, sus clases y sus niveles de confianza. La formulación original consolidó la detección como un problema unificado de regresión, y las versiones posteriores incorporaron cambios en el backbone, la fusión multiescala y la cabeza de detección [31, 39].

Evolución de las versiones de YOLO: Con el paso del tiempo, el algoritmo YOLO ha atravesado sucesivas versiones, cada una aportando avances en cuanto a rapidez, exactitud, diseño arquitectónico y facilidad de puesta en marcha. En el cuadro 2 se presenta una síntesis de las versiones más relevantes de YOLO junto con sus rasgos distintivos.

Cuadro 2. Evolución de las principales versiones de YOLO

Versión	Año	Autor / entidad	Características principales
YOLOv1	2016	Redmon et al. [31]	Versión inicial del modelo. Abordó la detección de objetos como un problema de regresión, analizando la imagen en su totalidad en un único paso de la red.
YOLOv2 / YOLO9000	2017	Redmon y Farhadi [32]	Elevó el nivel de precisión y velocidad al incorporar cajas ancla, normalización por lotes y un esquema de entrenamiento a múltiples escalas.
YOLOv3	2018	Redmon y Farhadi [33]	Añadió refinamientos en la arquitectura y permitió la detección en distintas escalas simultáneamente, mejorando el reconocimiento de objetos de tamaños variados.
YOLOv4	2020	Bochkovskiy et al. [34]	Buscó un mejor equilibrio entre velocidad y exactitud mediante CSP, aumento de datos con Mosaic, la función de activación Mish y la función de pérdida CIOU.
YOLOv5	2020	Ultralytics [35]	Desarrollado sobre PyTorch, ofreció mayor flexibilidad en el entrenamiento, facilidad de implementación y varias escalas de modelo (nano, small, medium, large y extra large).
YOLOv6	2022	Meituan [36]	Pensado para entornos industriales, aportó mejoras en eficiencia computacional, cuantización y adaptación a hardware de producción.
YOLOv7	2022	Wang et al. [37]	Incorporó estrategias de entrenamiento denominadas <i>trainable bag-of-freebies</i> , elevando el desempeño en la detección de objetos en tiempo real.
YOLOv8	2023	Ultralytics [38]	Edición más reciente, empleada en este trabajo. Cuenta con una arquitectura optimizada, un flujo de entrenamiento simplificado y compatibilidad con detección, segmentación y clasificación, incluyendo variantes ligeras como YOLOv8n.

Preprocesamiento y extracción de características: Antes de la inferencia, la imagen se redimensiona al tamaño de entrada del modelo y sus valores se convierten al formato numérico esperado. Esta normalización permite procesar secuencias provenientes de distintas resoluciones de cámara.

- **Redimensionamiento:** Adapta el fotograma a la resolución configurada para el modelo, conservando o rellenando la relación de aspecto según la implementación.
- **Backbone:** Se utilizan bloques convolucionales para extraer características visuales de diferentes niveles de complejidad.
- **SPPF:** Agrega información espacial a diferentes escalas y amplía el campo receptivo con un costo computacional moderado.

La detección también utiliza un enfoque sin cajas ancla predefinidas (*anchor-free*), lo que facilita la estimación de ubicaciones y minimiza la necesidad de ajuste [39].

A diferencia de la arquitectura original de YOLO, las versiones más recientes no utilizan una capa de red neuronal completamente conectada para hacer todas las predicciones, sino que utilizan cabezas convolucionales sobre mapas de características.

Fusión multiescala y cabeza de detección: Las características del backbone se combinan en un neck multiescala para conservar información semántica y espacial. La cabeza desacoplada estima por separado la localización y la clasificación, lo que favorece la optimización de ambas tareas.

Para cada detección, la red entrega coordenadas del cuadro delimitador, una clase y una puntuación de confianza. Estas salidas se convierten al sistema de coordenadas del fotograma original.

Salida del modelo: La salida de inferencia contiene un conjunto de candidatos, cada uno descrito mediante los siguientes elementos:

- **Caja delimitadora:** Coordenadas que definen la posición y el tamaño del objeto en la imagen.
- **Clase:** Categoría con mayor probabilidad entre las clases aprendidas durante el entrenamiento.
- **Confianza:** Puntuación utilizada para filtrar detecciones débiles y priorizar los candidatos más consistentes.

El tamaño y la variante del modelo se seleccionan según el equilibrio requerido entre velocidad, memoria y precisión. En este proyecto se utilizó YOLOv8n por su menor costo computacional [39].

Función de pérdida: Durante el entrenamiento, YOLOv8 optimiza una función compuesta que considera tres grupos principales de error:

1. **Localización:** Penaliza la diferencia entre el cuadro predicho y el cuadro anotado, empleando medidas basadas en la intersección sobre unión.
2. **Clasificación:** Penaliza cuando la clase predicha no coincide con la etiqueta real.
3. **Distribution focal loss:** Refina los límites del cuadro mediante una representación discreta de las distancias.

La salida de estas pérdidas combinadas y ponderadas se utiliza para actualizar los parámetros del modelo durante la retropropagación.

Los valores y pesos específicos dependen de la variante y la configuración de entrenamiento de Ultralytics [38].

Posprocesamiento y supresión de no máximos: La red puede producir varios candidatos superpuestos para un mismo objeto. El posprocesamiento filtra las predicciones mediante los siguientes pasos:

- Se eliminan las detecciones cuya confianza es inferior al umbral configurado.
- Los candidatos restantes se ordenan por confianza y se compara su intersección sobre unión (IoU).
- Cuando dos cajas de la misma clase se superponen por encima del umbral establecido, se conserva la de mayor confianza y se descarta la redundante.

Este procedimiento, conocido como supresión de no máximos, reduce duplicados y produce una salida interpretable para cada fotograma [42].

Ventajas técnicas: YOLO se caracteriza por realizar el proceso de detección y clasificación de objetos en una sola etapa, lo que permite un procesamiento unificado y más eficiente en comparación con otros métodos tradicionales. Además, destaca por su alta velocidad de ejecución, siendo capaz de procesar más de 30 fotogramas por segundo en tiempo real, característica fundamental para aplicaciones de visión artificial en drones y sistemas de monitoreo dinámico. Otra de sus ventajas es la predicción global de la imagen, ya que la red neuronal analiza toda la escena durante la inferencia, reduciendo significativamente la aparición de falsos positivos. Asimismo, su estructura modular facilita la implementación en dispositivos embebidos y sistemas con recursos limitados, permitiendo su integración en plataformas de procesamiento en tiempo real. La Figura 8 muestra el funcionamiento general del algoritmo YOLO, donde la imagen de entrada es procesada mediante una red neuronal que permite predecir simultáneamente las cajas delimitadoras y las clases de los objetos en una sola pasada.

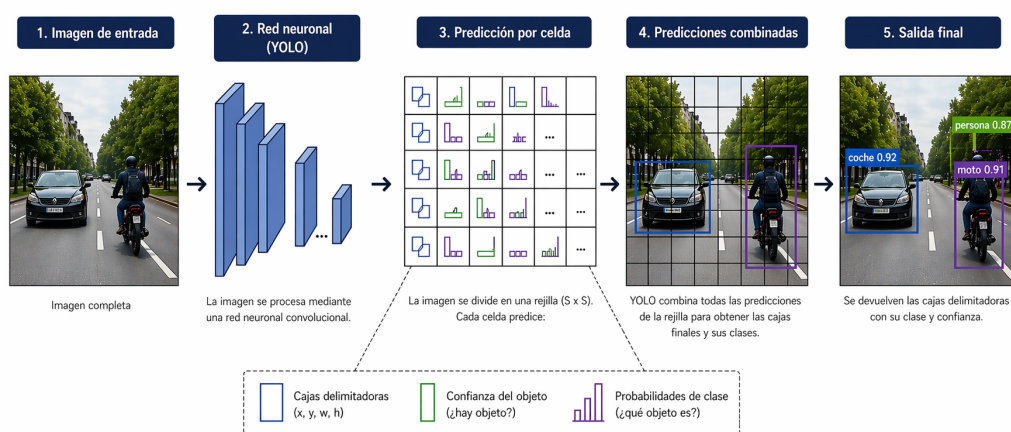


Figura 8. Funcionamiento general del algoritmo YOLO para la detección de objetos en imágenes.

Fuente: [38, 40, 41, 42]

1.4.7. Detección de anomalías visuales en UAVs

El uso de vehículos aéreos no tripulados (UAVs) equipados con cámaras ha abierto nuevas posibilidades para la vigilancia autónoma, la inspección visual y el reconocimiento de patrones. En este contexto, una de las aplicaciones más relevantes de la visión artificial es la detección de anomalías en video,

que permite detectar sucesos o comportamientos inusuales en tiempo real y sin intervención humana.

Joshi et al. [5] abordan la detección de anomalías en imágenes de drones mediante aprendizaje automático y redes neuronales profundas, y proponen representaciones y clasificadores visuales para detectar patrones anómalos. Sus conclusiones apuntan a que la formulación del problema y la calidad del conjunto de datos son determinantes para el éxito del enfoque.

Wang et al. [29] proponen un modelo de diagnóstico de anomalías de telemetría de sistemas no tripulados basado en redes neuronales gráficas y atención para la fusión de múltiples variables temporales y la detección de anomalías. Aunque no está orientado al video, este trabajo proporciona un marco para la fusión de información heterogénea.

Ahn y Chung [30] propusieron un sistema de detección de anomalías de un solo dron que realiza una misión de enjambre utilizando un modelo de aprendizaje profundo que se entrena con el comportamiento normal de cada dron para detectar comportamientos anómalos y fallos en su funcionamiento.

Sindhwani et al. [6] presentan un enfoque no supervisado para el aprendizaje de patrones de operación de drones de entrega autónomos sin etiquetas exhaustivas y la detección de observaciones anómalas cuando el comportamiento anómalo real es raro.

1.4.8. Operación de UAVs en espacios cerrados

La operación de vehículos aéreos no tripulados (UAVs) en espacios cerrados representa un escenario particularmente desafiante desde el punto de vista tecnológico. A diferencia del vuelo en exteriores, donde los drones pueden apoyarse en el sistema de posicionamiento global (GPS) y en trayectorias abiertas, los entornos interiores (como pasillos estrechos, bodegas, fábricas o instalaciones subterráneas) exigen una navegación de alta precisión debido a la cercanía de obstáculos, la limitada visibilidad y la falta de señal satelital.



Figura 9. Esquema de un dron en un espacio cerrado.

Fuente: Autoría propia

Como se muestra en la Figura 9, la navegación interior requiere percepción local y control preciso. La combinación de cámaras, sensores inerciales, telémetros, LiDAR o sensores de profundidad puede apoyar la estimación de posición y la evitación de obstáculos, pero cada configuración debe validarse para el espacio y la misión previstos [2].

Khosiawan y Nielsen [2] propusieron una arquitectura modular integrada por posicionamiento interior, control de vuelo, planificación de rutas y programación central de tareas. Separar funciones permite la coordinación entre múltiples UAV y la adaptación a cambios en el entorno.

Dado que los drones no pueden incorporar GPS de forma fiable, algunas aplicaciones utilizan sistemas de localización mediante visión por computadora, fusión de sensores inerciales (IMU) o sistemas de captura de movimiento que permiten el seguimiento de la posición del dron en un espacio tridimensional. Con estos sistemas de localización, se puede generar un modelo tridimensional que se puede utilizar para la navegación y la detección de anomalías.



Figura 10. Dron FIXAR

Fuente: [43]

Un ejemplo comercial es el FIXAR Indoor, ilustrado en la Figura 10. La plataforma fue concebida para inspección y mapeo en entornos sin GPS, mediante sensores de proximidad y percepción tridimensional. Sus características muestran cómo una solución especializada puede integrar protección física, localización y captura de datos en espacios confinados [43].

Este tipo de plataforma se orienta a minas, túneles, almacenes e instalaciones industriales donde el acceso humano puede ser limitado. Su mayor nivel de sensorización no es equivalente al DJI Tello utilizado en esta investigación; se presenta únicamente como referencia del estado de la tecnología [43].

1.4.9. Interfaces hombre-máquina (HMI)

Las interfaces hombre-máquina (HMI) constituyen el vínculo fundamental entre el operador humano y el sistema robótico, permitiendo la comunicación fluida mediante dispositivos como computadoras personales, tabletas o teléfonos inteligentes. A través de estas interfaces, el operador puede consultar el estado del dron, modificar su configuración, ejecutar determinadas tareas y realizar análisis de datos en tiempo real, sin necesidad de modificar el código del sistema.

Las aplicaciones más exigentes, como la logística, la vigilancia autónoma o la inspección industrial, requieren una HMI de alta calidad que permita interactuar con la máquina de una forma muy intuitiva y controlarla con gran precisión para garantizar su seguridad y correcto funcionamiento. En estos casos, la HMI no solo muestra información, sino que también permite capacidades interactivas avanzadas como mapas en tiempo real, programación de rutas y gestión de misiones de forma dinámica.

Un ejemplo de este tipo de HMI es el sistema que la empresa Robotnik proporciona para el robot RB-WATCHER en misiones de vigilancia, donde el operador puede establecer itinerarios, supervisar la misión, teleoperar el robot y recibir en tiempo real alarmas por fallos a través de diferentes canales,

entre los que se incluyen el correo electrónico y la mensajería instantánea. La fusión de estas capacidades proporciona al sistema una mayor autonomía, y al operador una menor carga cognitiva al poder gestionar tareas complejas a través de un entorno visual.

Para los UAV, la HMI es la interfaz entre la autonomía y el operador humano, en la que se muestran estados y se envían órdenes y alertas. Una buena organización en la HMI disminuye la carga de trabajo y reduce los tiempos de reacción [44].

1.4.10. Lenguajes de programación aplicados a UAVs

Para la implementación de los algoritmos de control y navegación en un UAV en interiores, se necesita un lenguaje de programación que consuma pocos recursos, sea flexible y disponga de bibliotecas de visión artificial, control de hardware e interfaces gráficas. Por estas características, los lenguajes más utilizados son Python, MATLAB y Scratch.

Python: Python es un lenguaje de propósito general, interpretado y multiplataforma, caracterizado por una sintaxis legible y un amplio ecosistema de bibliotecas. Su documentación oficial y la disponibilidad de herramientas para cómputo científico, visión artificial y comunicaciones facilitan el prototipado de sistemas robóticos [45].

En este proyecto, las bibliotecas principales fueron OpenCV para el procesamiento de imágenes, NumPy para operaciones matriciales, Ultralytics para ejecutar YOLOv8 y DJITelloPy para comunicarse con el dron [46, 47, 39].

- **OpenCV:** Biblioteca de visión por computadora y procesamiento de imágenes de código abierto que incluye detección de bordes, reconocimiento de objetos, seguimiento visual y navegación autónoma.
- **TensorFlow:** Una biblioteca de código abierto de Google para redes neuronales profundas que se utiliza para la detección de anomalías y el reconocimiento visual en UAVs.
- **PyTorch:** Un competidor popular de TensorFlow, PyTorch utiliza grafos computacionales dinámicos para una construcción más sencilla de modelos de aprendizaje profundo.
- **Scikit-image:** Biblioteca para realizar análisis y segmentación de imágenes, análisis morfológico y extracción de medidas estructurales.
- **Pillow:** Enfocada en operaciones básicas de manipulación de imágenes como recortar, rotar y convertir formatos.

- **SimpleITK y Mahotas:** Son bibliotecas de software científico que pueden ser utilizadas para analizar estructuras observadas por UAVs.

Python también se utiliza con ROS (Robot Operating System) y plataformas como Tello para desarrollar sistemas autónomos de interiores.

Además, permite crear interfaces gráficas mediante PyQt o Tkinter para el control visual del sistema.

MATLAB (Matrix Laboratory): MATLAB es un entorno de programación numérica y visualización desarrollado por MathWorks. Sus herramientas para procesamiento de imágenes, visión por computador, simulación y generación de código permiten modelar algoritmos y sistemas de control [48].

Las herramientas de MATLAB más relacionadas con UAV y visión artificial son:

- **Computer vision toolbox:** Incluye funciones para detección de características, calibración de cámaras, reconocimiento de objetos y seguimiento visual.
- **Image processing toolbox:** Segmentación, filtrado y análisis estructural de imágenes.
- **Simulink:** Entorno de simulación integrado a MATLAB que se emplea para simular sistemas de control, navegación sin GPS y trayectorias de UAVs en interiores.
- **Parallel computing toolbox:** permite el procesamiento en GPU, que es especialmente útil para las aplicaciones de procesamiento de imágenes en tiempo real.

MATLAB también genera código C/C++ para sistemas embebidos, y realiza procesamiento de imágenes en UAV en hardware como Jetson Nano.

Scratch: Es un lenguaje de programación visual basado en bloques, desarrollado por el MIT para facilitar la enseñanza de la lógica computacional y el pensamiento algorítmico. Está especialmente orientado a la educación, permitiendo a los usuarios sin experiencia previa desarrollar programas mediante una interfaz intuitiva y gráfica [49, 50].

Las aplicaciones en UAV educativos son:

- **DroneBlocks y ScratchX:** Extensiones que permiten programar drones como el Tello para tareas simples de movimiento, captura de imágenes y automatización básica.

- **Educación y prototipado:** Scratch se utiliza en cursos introductorios para familiarizar a los estudiantes con conceptos de programación aplicados a la robótica aérea.

Aunque sus capacidades son limitadas para visión artificial avanzada, es una herramienta eficaz para fases iniciales de prototipado o formación en entornos educativos.

1.4.11. Sistema de comunicación

Un sistema de comunicación se basa en un conjunto de dispositivos que permiten transmitir, emitir y recibir señales como voz, datos, audio o video. Estas señales pueden ser digitales o analógicas, y su transmisión requiere tres elementos fundamentales: un transmisor, un medio de transmisión y un receptor [51].

El transmisor genera y manipula la señal mediante procesos como modulación, codificación y filtrado. El medio de transmisión (ya sea alámbrico o inalámbrico) transporta la señal, mientras que el receptor la reconstruye para recuperar la información original [52].

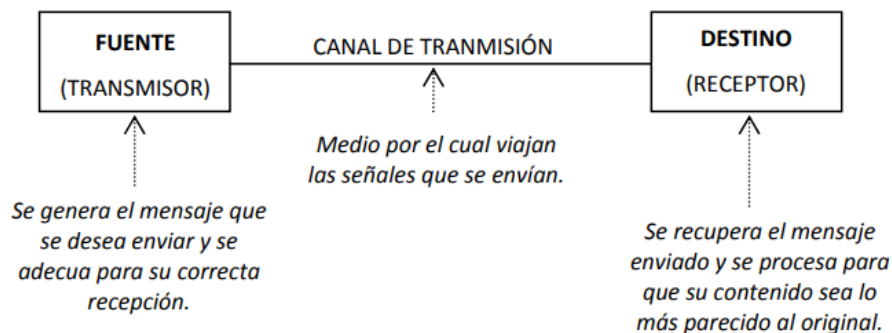


Figura 11. Sistema de comunicación

Fuente: [51]

A continuación, se describen los tipos de comunicación más comunes empleados en sistemas UAV:

Cable USB: La comunicación por USB emplea una arquitectura serial con cuatro conductores: dos para alimentación y dos para datos. Puede operar a velocidades de 1,5 Mbps (LOW), 12 Mbps (FULL) y 480 Mbps (HIGH). Además, integra mecanismos de detección, reconocimiento e interacción entre dispositivos conectados [53].

Bluetooth: Bluetooth permite la transmisión inalámbrica de datos a corto alcance (hasta 10 m), operando en 2.4 GHz. Se basa en enlaces por salto de frecuencia (frequency hopping), lo que garantiza una comunicación estable incluso en entornos con interferencia electromagnética. Es ideal para la sincronización de datos entre móviles y sistemas embarcados [54].

Wi-Fi: Wi-Fi agrupa tecnologías de red local inalámbrica basadas en la familia IEEE 802.11. Las implementaciones pueden operar en bandas de 2,4, 5 o 6 GHz; la cobertura y el rendimiento dependen del estándar, el canal y las condiciones de propagación [55].

En el sistema propuesto, la computadora se conecta a la red Wi-Fi de 2,4 GHz generada por el DJI Tello. Mediante el SDK y el protocolo UDP se envían comandos de control, se reciben datos de estado y se obtiene el flujo de video para procesamiento en tiempo real [56].

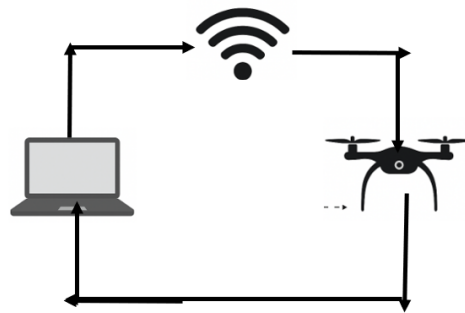


Figura 12. Protocolo de comunicación Wi-Fi

Fuente: Autoría propia

1.4.12. Seguridad operacional y normativas legales para UAVs en espacios interiores

Las operaciones de drones en espacios cerrados presentan riesgos que deben tenerse en cuenta. Entre otras cosas, no hay GPS, se vuela cerca de personas y objetos, hay superficies reflectantes y puede haber pérdida de enlace (es decir, pérdida de la conexión entre el piloto y el dron). Por lo tanto, antes de cada prueba se deben definir los límites de altitud y velocidad, las zonas de exclusión, los procedimientos de emergencia y las responsabilidades del operador.

Las mejores prácticas incluyen: Utilizar protectores de hélices, revisar la batería y los propulsores, mantener la línea de visión, no sobrevolar a las personas, controlar el acceso a la zona y tener despejado un camino de

aterrizaje [57]. También se sugiere que el dron esté adaptado al espacio y que se realice una inspección previa [57].

En interiores, por su parte, se deben desactivar las funciones de retorno al punto de origen, entre otras que dependan del GPS, y el piloto debe contar con un protocolo de aterrizaje en caso de emergencia, verificar el nivel de batería e interrumpir el vuelo si pierde la conexión con el dron, si el dron presenta un comportamiento anómalo o si acceden personas no autorizadas a la zona.

Las normas aeronáuticas se orientan principalmente al uso del espacio aéreo exterior. Su aplicabilidad a recintos cerrados varía por jurisdicción; por tanto, además de la regulación aeronáutica, deben cumplirse las políticas institucionales, las reglas de seguridad ocupacional, la protección de datos y el consentimiento para capturar imágenes. A continuación, se resume el alcance de cuatro marcos normativos relevantes.

México: NOM-107-SCT3-2019 La NOM-107-SCT3-2019 establece requisitos para operar RPAS en el espacio aéreo mexicano. La propia norma indica que no se aplica a RPAS operados en espacios cerrados y atribuye al propietario de la instalación y al organizador la responsabilidad de establecer las medidas de seguridad correspondientes. Si la operación se traslada al exterior, deben revisarse los requisitos de registro, limitaciones operativas y autorizaciones previstos por la autoridad aeronáutica [58].

Unión Europea: Reglamentos (UE) 2019/945 y 2019/947 El Reglamento Delegado (UE) 2019/945 establece requisitos de diseño y comercialización de UAS, mientras que el Reglamento de Ejecución (UE) 2019/947 organiza las operaciones exteriores en categorías abierta, específica y certificada según su nivel de riesgo. Estos reglamentos no sustituyen la evaluación de seguridad del propietario del recinto para vuelos que permanecen completamente dentro de edificios [59, 60].

Argentina: Resolución ANAC 550/2025 La Resolución ANAC 550/2025 aprobó las Partes 100, 101 y 102 de las Regulaciones Argentinas de Aviación Civil para aeronaves pilotadas a distancia y sustituyó el marco anterior. La norma clasifica las operaciones por riesgo en categorías abierta, específica y certificada. Para cualquier operación exterior deben consultarse los requisitos vigentes; en recintos cerrados continúan siendo esenciales el control del lugar, la seguridad de las personas y la protección de la imagen [61].

Ecuador: RDAC 101 En Ecuador, la RDAC 101 (revisada y aprobada por última vez en diciembre de 2024) regula los sistemas de aeronaves no tripuladas, la certificación de los operadores y la instrucción UAS. Al operar un UAS, el operador debe verificar el registro, la categoría operacional, las limitaciones del espacio aéreo y las autorizaciones. El enfoque del laboratorio estuvo en la autorización institucional, el control de acceso, la seguridad de los participantes y los procedimientos de emergencia [62].

Tomando en cuenta estos marcos, se puede concluir que la seguridad no solo depende de la regulación de la aviación y que, para la operación interior, se debe documentar el consentimiento de la persona, el uso previsto de las imágenes, el almacenamiento de la información, la supervisión del operador y la posibilidad de detener el vuelo en caso de inseguridad.

1.5. Marco contextual

El proyecto se desarrolló en el laboratorio de electrónica de la Universidad Estatal Península de Santa Elena (UPSE), un espacio académico equipado para la implementación de soluciones tecnológicas en entornos controlados. Para este caso, el laboratorio se adaptó como una maqueta representativa de un entorno cerrado, incluyendo elementos como sillas, escritorios, equipos electrónicos, pasillos y obstáculos que simulan un espacio real como una oficina, sala de control o centro de operaciones.



Figura 13. Laboratorio de electrónica ubicado en la UPSE

Fuente: Autoría propia

En este ambiente se recrearán diversas situaciones que permitan validar la capacidad del sistema para diferenciar entre comportamientos normales y anómalos dentro de un entorno laboral controlado. En condiciones normales, se espera que las personas realicen actividades propias del entorno de trabajo, tales como permanecer en sus estaciones asignadas, desplazarse de manera ordenada por los pasillos, interactuar de forma adecuada con equipos electrónicos y mantener una conducta acorde a las normas institucionales.

Por otro lado, se simularán comportamientos anómalos o inusuales que se desvían de estas condiciones normales. Entre estas acciones se incluyen caídas de personas, carreras dentro del área de trabajo, uso indebido del teléfono celular, consumo de alimentos en lugares restringidos.

Estas condiciones permitirán no solo evaluar la efectividad del sistema de detección de presencia humana, sino también su capacidad para identificar situaciones que podrían representar riesgos, faltas de disciplina o condiciones que afecten el ambiente institucional o laboral.

Los principales beneficiarios de este proyecto serán los estudiantes y docentes de la Facultad de Sistemas y Telecomunicaciones de la UPSE, ya que podrán fortalecer los procesos de enseñanza-aprendizaje en visión artificial, robótica, automatización e inteligencia artificial, al poner en práctica sus

conocimientos en un proyecto real y funcional.

Además, el sistema podrá utilizarse como una herramienta académica para futuras investigaciones, prácticas supervisadas o pruebas de nuevos algoritmos, beneficiando a generaciones posteriores. A largo plazo, este tipo de soluciones puede trasladarse a instituciones, oficinas o entornos donde se requiera monitoreo automatizado de comportamientos, reforzando así la seguridad y el control en espacios cerrados.

2. Metodología del proyecto y diseño experimental

2.1. Metodología del proyecto

El enfoque fue mixto, combinando la exploración y descripción de un experimento controlado, análisis cuantitativos y un estudio de caso. Esto hizo posible la selección de tecnologías, la caracterización del laboratorio, el entrenamiento del modelo y la evaluación del prototipo en el lugar de su aplicación. El uso de métodos mixtos fue motivado por la naturaleza aplicada de este trabajo [63, 64, 65].

2.1.1. Metodología exploratoria

En la etapa inicial se aplicó un alcance exploratorio para identificar tecnologías de detección, comunicación y control adecuadas para un prototipo en interiores. El propósito fue delimitar el problema y establecer criterios de selección, no realizar una revisión sistemática con protocolo de metaanálisis [63].

Se efectuó una revisión bibliográfica estructurada en IEEE Xplore, ScienceDirect, Springer y MDPI, complementada con documentación oficial de hardware y software. A partir de esta revisión se seleccionaron YOLOv8n, OpenCV, Python, PyCharm y DJITelloPy como componentes del prototipo.

La revisión también permitió anticipar posibles dificultades, como la interferencia lumínica en interiores o la pérdida de conectividad Wi-Fi, lo que facilitó la toma de decisiones desde una base informada para etapas posteriores del desarrollo.

2.1.2. Metodología descriptiva

El alcance descriptivo se utilizó para caracterizar la distribución del mobiliario, la iluminación, los obstáculos, las zonas de interés del laboratorio de electrónica de la UPSE, las clases visuales y las condiciones de prueba que se debían reproducir [63].

Asimismo, se identificaron y describieron los patrones de comportamiento humano que el sistema debía reconocer, como caídas, entrada a áreas restringidas, estado de alerta, falta de actividad, posiciones inusuales o movimientos erráticos, fueron identificados y descritos. La conducta se modeló a partir de registros visuales en un ambiente de laboratorio. Para ello, se tuvo que crear un conjunto de clases y una base de datos con imágenes reales que se etiquetaron manualmente.

Este análisis descriptivo sirvió como fundamento para el diseño de escenarios representativos y la planificación de pruebas, alineando el funcionamiento del sistema con las condiciones reales del entorno de aplicación.

2.1.3. Metodología experimental

La etapa experimental evaluó el sistema bajo condiciones controladas, modificando la iluminación, la posición de las personas y la distribución de obstáculos. El registro de configuraciones y resultados permitió comparar el comportamiento del modelo entre ensayos [64].

YOLOv8 fue entrenado con una base de datos propia que se generó en el laboratorio con diferentes ángulos de visión, condiciones de iluminación y comportamientos humanos. Se definieron los hiperparámetros más relevantes en el entrenamiento (*learning rate* o tasa de aprendizaje, *epochs* o número de épocas, y *batch size* o tamaño de lote) y se optimizaron en función de los resultados que se obtenían.

Las pruebas incluyeron vuelos asistidos y trayectorias programadas cortas en entornos estáticos y dinámicos. Se observaron la estabilidad del vuelo, la continuidad del video y la detección de las clases definidas. No se implementó una navegación autónoma completa ni evasión automática de obstáculos.

2.1.4. Metodología cuantitativa

Los datos obtenidos durante las pruebas experimentales fueron sometidos a análisis estadístico mediante un enfoque cuantitativo. Se utilizaron bibliotecas en Python como NumPy y SciPy para procesar los resultados y aplicar métricas estándar de evaluación en sistemas de clasificación: exactitud (*accuracy*), precisión (*precision*), sensibilidad (*recall*) y puntuación F1 (*F1-score*).

Estas métricas permitieron evaluar la calidad de la clasificación e identificar posibles sesgos o errores sistemáticos. Los resultados de la clasificación que ha dado el modelo se compararon entre las distintas versiones del modelo entrenado con distintos parámetros para poder seleccionar el modelo con la mejor relación entre velocidad y precisión.

El análisis cuantitativo también permitió una evaluación objetiva del rendimiento del sistema que ayudó a probar su viabilidad técnica en un entorno del mundo real.

2.1.5. Metodología de estudio de caso

Se adoptó un estudio de caso centrado en el Laboratorio de electrónica de la UPSE, por ser el entorno específico donde se integró y probó el prototipo. Este diseño permite analizar en profundidad una solución dentro

de su contexto real, sin generalizar automáticamente los resultados a otros espacios [65].

La prueba incluyó la resistencia del sistema al ruido visual, la usabilidad de la interfaz de control, las interacciones del usuario con la plataforma de programación y su respuesta a situaciones imprevistas. También se evaluaron la capacidad de detección de comportamientos (puntos fuertes), la incidencia de falsos positivos, la pérdida de señal y los errores de clasificación (puntos débiles).

La validación práctica no solo fue importante para comprobar la viabilidad de la propuesta, sino que también proporcionó información sobre el ajuste del modelo y la arquitectura del sistema.

2.1.6. Metodología diagnóstica

Finalmente, se aplicó una metodología diagnóstica orientada a identificar y corregir factores críticos que podrían afectar el funcionamiento del sistema. Entre los hallazgos más relevantes estuvieron las interferencias generadas por reflejos en superficies brillantes, sombras proyectadas que confundían al modelo y pérdidas de conectividad temporal del dron con la estación base.

Para mitigar estos problemas, se realizaron ajustes en el preprocesamiento de imágenes (normalización de iluminación, uso de filtros adaptativos), se mejoró la segmentación por color y se redefinieron algunos umbrales de activación en el modelo YOLO. Además, se verificó la compatibilidad de las bibliotecas de software empleadas con el hardware del dron y se realizaron pruebas de estabilidad en la transmisión Wi-Fi.

Este enfoque diagnóstico permitió afinar la funcionalidad del sistema, aumentando su fiabilidad operativa y facilitando su futura implementación en otros contextos.

2.1.7. Descripción del proyecto

El proyecto desarrolla un sistema de detección automática de situaciones anómalas a partir del video de un dron en interiores. El análisis visual identifica clases definidas previamente, mientras el vuelo permanece bajo control asistido del operador.

La detección se realizó con YOLOv8, previamente entrenado con una base de datos personalizada, mientras que OpenCV se utilizó para procesar las imágenes. Para seleccionar el dron se consideraron la estabilidad en el vuelo, la facilidad de implementar las librerías de visión artificial, el procesamiento en tiempo real y la conectividad inalámbrica, ya que la plataforma aérea debe ser capaz de realizar las misiones en un entorno como el del laboratorio

de electrónica de la Facultad de Sistemas y Telecomunicaciones de la UPSE, donde se realizan las pruebas y validaciones.

El sistema se compone de cuatro bloques principales: planificación de la trayectoria de prueba, captura de video, inferencia con YOLOv8 y generación de alertas en la interfaz. La Figura 14 presenta los bloques funcionales de la propuesta.

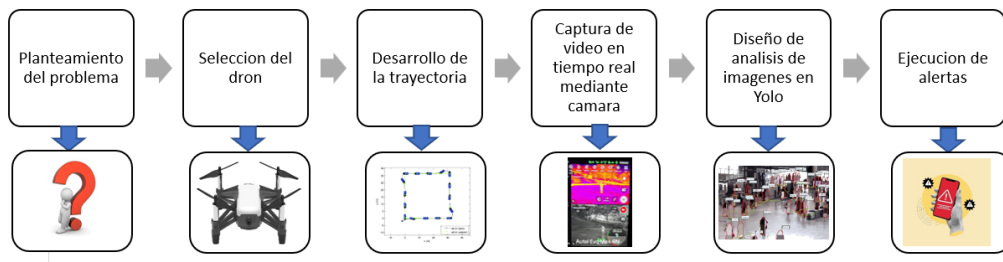


Figura 14. Esquema general del sistema de detección de presencia humana.

Fuente: Autoría propia.

2.2. Componentes de la propuesta

En esta sección, se presenta una descripción detallada de los componentes físicos y lógicos requeridos para el desarrollo del proyecto de visión artificial aplicada a UAVs en espacios cerrados, con énfasis en inspección, vigilancia y detección de anomalías. Se destacan las características técnicas principales de cada componente, junto con una explicación concisa de su funcionalidad, subrayando su importancia en el diseño e implementación del sistema propuesto.

2.2.1. Componentes físicos

El dron DJI Tello fue seleccionado como plataforma base por su estabilidad en interiores, facilidad de programación, bajo peso y compatibilidad con bibliotecas de visión artificial como OpenCV. Este cuadricóptero compacto incluye una cámara integrada, un sistema de posicionamiento visual y motores eficientes, lo cual lo hace ideal para proyectos de detección de comportamientos en espacios cerrados como aulas, oficinas o laboratorios.

Dron DJI Tello: El DJI Tello es un dron de bajo costo y fácil programación, ampliamente utilizado para fines educativos y de investigación. Su cámara integrada permite capturar imágenes y videos en resolución HD, y

puede ser controlado mediante un SDK compatible con Python. Además, su sistema de posicionamiento visual permite una mayor estabilidad durante el vuelo en interiores [72, 73, 74].



Figura 15. Dron DJI Tello

Fuente: [74]

En la Figura 15 se muestra una imagen representativa del dron DJI Tello utilizado en este proyecto. Además, en el cuadro 3 se presentan sus principales especificaciones técnicas, que justifican su elección como plataforma de prueba debido a su bajo peso, cámara integrada, capacidad de programación mediante SDK y buena estabilidad en entornos interiores.

Cuadro 3. *Especificaciones técnicas del Dron DJI Tello* [72, 74, 73]

Especificaciones técnicas	
Aeronave	Modelo TLW004
Peso (con protectores)	87 g
Velocidad máxima	28,8 km/h (17.8 mph)
Tiempo de vuelo máximo	13 minutos (sin viento a 15 km/h)
Rango de temperatura funcionamiento	De 0 °C a 40 °C
Frecuencia de funcionamiento	2,4 a 2,4835 GHz
Transmisor (PIRE)	20 dBm (FCC), 19 dBm (CE/SRRC)
Cámara	2592 x 1936 (foto), 1280 x 720 30p (video)
Formato de video	MP4
Batería de vuelo	1100 mAh, 3,8 V, LiPo, 4,18 Wh
Peso neto de la batería	2 g
Temp. de carga	De 5 °C a 45 °C
Potencia de carga máxima	10 W

Sistema de posicionamiento visual: El sistema de posicionamiento visual del dron DJI Tello permite mantener una posición estable durante el vuelo en interiores. Está compuesto por una cámara y un sensor infrarrojo 3D ubicados en la parte inferior del dron, los cuales trabajan en conjunto para detectar el movimiento relativo respecto al suelo. Este sistema calcula los desplazamientos basándose en patrones visuales y datos de altitud, permitiendo que el dron mantenga una altura constante sin necesidad de GPS [75].



Figura 16. Sistema de posicionamiento visual en el dron DJI Tello

Fuente: Autoría propia

La estabilidad depende de la calidad de la superficie, la iluminación y la altura de operación. Superficies uniformes, reflectantes o con poca luz pueden degradar el posicionamiento, por lo que las pruebas se realizaron en un entorno controlado [70, 72].

Motores: El DJI Tello utiliza cuatro micromotores de corriente continua integrados en su estructura. El control de actitud se obtiene variando la velocidad de cada motor. Ryze Tech no publica una ficha eléctrica individual de estos motores; por ello, no se atribuyen al fabricante valores de corriente, velocidad en vacío o tipo de conmutación que no consten en su documentación final.

Las especificaciones de los motores del dron DJI Tello se establecen en el cuadro 4. Estos motores se han diseñado para ofrecer las características de velocidad, eficiencia energética y capacidad de respuesta necesarias para un vuelo en interiores y son un componente clave del rendimiento y la estabilidad del dron en vuelo.

Cuadro 4. *Especificaciones de los motores del dron Tello* [76]

Especificaciones de los motores	
Voltaje	3.7 V DC
Juego final del eje	0,05 – 0,3 mm
Velocidad sin carga	52000 RPM
Corriente sin carga	93 mA (MÁX)
Corriente de carga	2200 mA (MÁX)
Voltaje de arranque	0.8 V DC (MÁX)
Peso bruto	5.5 g/pc

La Figura 17 muestra una imagen ilustrativa del motor sin escobilla.



Figura 17. Motores sin escobillas del dron DJI Tello

Fuente: [76]

Hélices y funcionamiento del dron: Las hélices del dron Tello son de dos palas, livianas y de fácil reemplazo. El funcionamiento del dron, en cuanto a desplazamiento y control, se basa en la variación de velocidad entre sus cuatro motores. A continuación, se describe el principio de funcionamiento paso a paso [77]:

- **Ascenso:** Todos los motores giran a alta velocidad, generando una fuerza de sustentación superior al peso del dron (Figura 18) [77].



Figura 18. Movimiento ascendente de un dron.
Fuente: Autoría propia.

- **Descenso:** Se reduce la velocidad de los motores, disminuyendo la sustentación (Figura 19) [77].



Figura 19. Movimiento descendente de un dron.
Fuente: Autoría propia.

- **Desplazamiento lateral:** Se aumenta la velocidad de un par de motores y se reduce la del lado opuesto, permitiendo moverse hacia la izquierda como se ve en la Figura 20 y hacia la derecha como se ve en la Figura 21 [77].



Figura 20. Desplazamiento a la izquierda de un dron.
Fuente: Autoría propia.



Figura 21. Desplazamiento a la derecha de un dron.

Fuente: Autoría propia.

- **Avance/retroceso:** Para avanzar, los motores traseros giran más rápido como se ve en la Figura 22 y para retroceder, los delanteros lo hacen como se ve en la Figura 23 [77].



Figura 22. Avance de un dron.

Fuente: Autoría propia.



Figura 23. Retroceso de un dron.

Fuente: Autoría propia.

- **Giros:** Los motores que giran en sentido horario (CW) y antihorario (CCW) alternan su velocidad para generar rotación sobre el eje vertical tal y como se puede visualizar en la Figura 24 y en la Figura 25 [77].



Figura 24. Rotación en sentido horario.

Fuente: Autoría propia.



Figura 25. Rotación en sentido antihorario.

Fuente: Autoría propia.

2.2.2. Componentes lógicos

Los componentes lógicos permiten programar y controlar el dron DJI Tello mediante un entorno de desarrollo y librerías especializadas. A continuación, se describen los principales elementos utilizados:

Entorno de desarrollo PyCharm: PyCharm es un entorno de desarrollo integrado (IDE) especializado en el lenguaje Python, ampliamente utilizado en el ámbito de la inteligencia artificial, la visión por computador y el análisis de datos. Este entorno, desarrollado por la empresa JetBrains, proporciona herramientas avanzadas para facilitar el desarrollo de proyectos complejos, como los que integran control de drones, procesamiento de imágenes y automatización de tareas.

PyCharm permite gestionar intérpretes, entornos virtuales, paquetes, depuración y control de versiones en proyectos Python. Estas funciones facilitan mantener dependencias separadas y reproducir el entorno de desarrollo utilizado para el prototipo [78].

El editor ofrece autocompletado, inspección de código, navegación entre archivos y depuración interactiva. También integra Git y herramientas de

pruebas, características útiles para organizar los módulos de control, video e inferencia [78].

En este proyecto, PyCharm se utilizó para escribir, probar y depurar el código de comunicación con el DJI Tello y el procesamiento de video. La Figura 26 presenta el entorno de trabajo.



Figura 26. Áreas de trabajo en PyCharm
Fuente: Autoría propia

La Figura 26 identifica los componentes principales del IDE utilizados durante el desarrollo:

- **Panel de proyecto:** ubicado en la parte izquierda, permite gestionar la estructura del proyecto, acceder a los archivos del programa y navegar entre carpetas, scripts y recursos asociados.
- **Editor de código:** Es la parte más importante del IDE, ya que es donde se escribe, edita y analiza el código fuente. Incluye herramientas de autocompletado, resaltado de sintaxis y navegación rápida entre funciones.
- **Barra de herramientas:** Ofrece accesos directos para ejecutar código, depurar, gestionar entornos virtuales y versiones, y otras herramientas que hacen de este un entorno de desarrollo muy ágil.
- **Consola de ejecución:** En la parte inferior se pueden ver los resultados de la ejecución de programas, así como los mensajes de `print()`, errores, advertencias y los resultados de las pruebas.

Este entorno robusto facilita la implementación, depuración y prueba del sistema basado en visión artificial, permitiendo al desarrollador mantener un control detallado sobre cada etapa del código.

Librería DJITelloPy: DJITelloPy es una biblioteca de Python que implementa una interfaz de alto nivel para el SDK del DJI Tello. Permite enviar comandos de vuelo, consultar datos de estado y acceder al flujo de video mediante los canales UDP definidos por el fabricante [79, 56]. Entre sus ventajas se encuentran:

- Los comandos de vuelo `takeoff()`, `land()`, `moveforward()` y `rotateclockwise()` se pueden usar de forma sencilla.
- Estado del dron: batería, velocidad, temperatura, calidad de la señal.
- Soporte para varios modelos de dron DJI Tello en modo manual o con bibliotecas como OpenCV o NumPy.
- Para realizar maniobras más complejas se puede enviar el comando “`send rc control()`”, que controla el dron en tiempo real en términos de velocidades en cada eje y rotación.

En este proyecto, DJITelloPy se empleó para establecer la conexión, activar la transmisión de video, ejecutar comandos básicos y consultar el nivel de batería. La biblioteca simplifica la integración, pero no elimina las restricciones de seguridad ni las limitaciones del SDK [79].

Los comandos más relevantes que se utilizaron para la implementación del proyecto, usando la librería de control de dron `djitellopy`, están en el cuadro [5]. Estos comandos permiten de forma básica despegar, aterrizar y moverse, así como el control manual del dron y son la base para lograr un sistema de navegación y análisis en tiempo real totalmente autónomo.

Cuadro 5. *Comandos del dron DJI Tello* [Autoría propia]

Comandos del dron DJI Tello	
Comando	Descripción
DRON = tello.Tello()	Crea una instancia del objeto Tello vinculada a la biblioteca djitellopy.
DRON.connect()	Establece la comunicación inicial entre el dron y el programa.
DRON.streamon()	Activa la transmisión de video en tiempo real desde la cámara del dron.
DRON.takeoff()	Ordena el despegue automático del dron desde su posición actual.
DRON.land()	Indica al dron que inicie el proceso de aterrizaje.
DRON.move_forward()	Mueve al dron hacia adelante en línea recta.
DRON.move_back()	Hace retroceder al dron.
DRON.move_left()	Traslada lateralmente al dron hacia la izquierda.
DRON.move_right()	Traslada lateralmente al dron hacia la derecha.
DRON.move_up()	Eleva al dron hacia una mayor altitud.
DRON.move_down()	Provoca que el dron reduzca su altitud.
DRON.rotate_clockwise()	Hace que el dron gire en dirección de las agujas del reloj.
DRON.rotate_counter_clockwise()	Genera un giro del dron en sentido antihorario.
DRON.go_xyz_speed()	Ordena al dron moverse en coordenadas XYZ definidas con una velocidad asignada.
DRON.flip_forward()	Ejecuta una voltereta hacia adelante.
DRON.flip_back()	Ejecuta una voltereta hacia atrás.
DRON.flip_left()	Ejecuta una voltereta hacia la izquierda.
DRON.flip_right()	Ejecuta una voltereta hacia la derecha.
DRON.send_rc_control(left_right_velocity, for- ward_backward_velocity, up_down_velocity, yaw_velocity)	Permite el control manual de velocidades horizontales, verticales y de rotación, enviadas directamente al dron por medio de los canales de control remoto.

Librería NumPy (Numerical Python): Es una biblioteca fundamental en el ecosistema de Python, utilizada para el manejo eficiente de estructuras de datos numéricos, especialmente arreglos multidimensionales denominados *ndarrays* [80]. En el contexto del procesamiento de imágenes, la biblioteca NumPy permite representar cada imagen como una matriz de píxeles, facilitando operaciones matemáticas como transformaciones, filtrados y combinaciones entre matrices.

Una de sus principales ventajas es su rendimiento computacional: al estar escrita en C, las operaciones sobre grandes volúmenes de datos se ejecutan de forma mucho más rápida que usando listas estándar de Python. NumPy también incluye una gran cantidad de funciones matemáticas avanzadas, como álgebra lineal, transformadas, estadísticas, y operaciones vectorizadas, esenciales en visión artificial y aprendizaje automático [80]. Las funcionalidades clave aplicadas incluyen:

- Creación y manipulación de arreglos multidimensionales con `np.array()` y `np.reshape()`.
- Operaciones vectorizadas sobre imágenes (como suma, resta, multiplicación entre píxeles).
- Filtrado de datos, umbralización y estadísticas para análisis de regiones de interés (ROI).

Con estas propiedades, se obtuvo una herramienta que permitió la creación de algoritmos óptimos para la búsqueda de patrones anómalos en el entorno del dron.

Los ejemplos de arreglos de una, dos y tres dimensiones son los de la Figura 27. Una imagen en escala de grises puede modelarse como una matriz 2D, mientras que una imagen RGB se representa habitualmente mediante un arreglo 3D con canales de color [47].

Estas representaciones son fundamentales en aplicaciones de visión artificial, ya que permiten manipular imágenes como matrices de valores numéricos en diferentes dimensiones. NumPy proporciona funciones específicas para acceder, modificar y procesar estos arreglos, facilitando la integración con otras bibliotecas como OpenCV y TensorFlow.

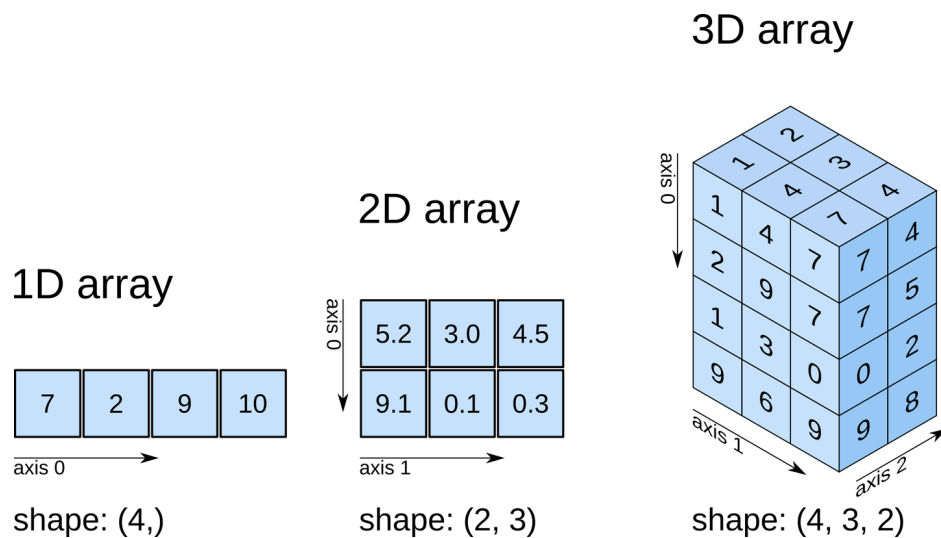


Figura 27. Arreglos en 1D, 2D y 3D.
Fuente: [80].

Librería OpenCV (Open source computer vision library): Es una biblioteca de código abierto ampliamente utilizada en aplicaciones de visión por computador. Desarrollada inicialmente por Intel, está optimizada para la ejecución en tiempo real y se ha convertido en un estándar de facto en proyectos que involucran análisis de imágenes y video [81].

La librería OpenCV proporciona funciones especializadas para tareas como detección de objetos, seguimiento, reconocimiento de rostros, transformación de imágenes, segmentación, y calibración de cámaras. Una de sus mayores ventajas es su capacidad para trabajar de forma integrada con la librería NumPy, lo que permite aplicar operaciones matemáticas sobre imágenes representadas como matrices. Entre las funciones más utilizadas destacan:

- `cv2.VideoCapture()` para capturar el video en tiempo real.
- `cv2.cvtColor()` para convertir imágenes entre diferentes espacios de color.
- `cv2.inRange()` para realizar segmentaciones por color.
- `cv2.findContours()` y `cv2.drawContours()` para identificar bordes y objetos.
- `cv2.resize()`, `cv2.flip()`, `cv2.putText()` y otras para manipulación visual.

Cuadro 6. Comandos de OpenCV [Autoría propia]

Comando	Descripción
<code>cv2.imread()</code>	Importa una imagen desde el disco y la transforma en una matriz de datos.
<code>cv2.imshow()</code>	Visualiza una imagen dentro de una ventana emergente.
<code>cv2.imwrite()</code>	Almacena una imagen en un archivo del sistema.
<code>cv2.IMREAD_COLOR</code>	Interpreta la imagen con sus colores originales.
<code>cv2.IMREAD_GRAYSCALE</code>	Interpreta la imagen en escala de grises.
<code>cv2.IMREAD_UNCHANGED</code>	Mantiene todos los canales de la imagen sin alteraciones.
<code>cv2.waitKey()</code>	Detiene el programa hasta que se presione una tecla.
<code>cv2.destroyAllWindows()</code>	Cierra todas las ventanas creadas por OpenCV.
<code>cv2.VideoCapture()</code>	Habilita la captura de video desde una cámara o archivo.
<code>cv2.line()</code>	Traza una línea sobre la imagen.
<code>cv2.circle()</code>	Dibuja un círculo sobre la imagen.
<code>cv2.putText()</code>	Inserta un texto sobre una imagen.
<code>cv2.createTrackbar()</code>	Crea una barra deslizante interactiva dentro de una ventana.
<code>cv2.getTrackbarPos()</code>	Lee la posición actual de una barra deslizante.
<code>cv2.resize()</code>	Modifica las dimensiones de la imagen.
<code>cv2.add()</code>	Realiza una suma entre dos imágenes o matrices.
<code>cv2.bitwise()</code>	Aplica operaciones bit a bit entre imágenes.
<code>cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)</code>	Transforma una imagen de formato BGR a HSV.
<code>cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)</code>	Transforma una imagen BGR a escala de grises.

Continúa en la siguiente página

Cuadro 6. Comandos de OpenCV [Autoría propia]

Comando	Descripción [Autoría propia]
<code>cv2.namedWindow()</code>	Crea una ventana personalizada con un nombre específico.
<code>cv2.inRange()</code>	Permite realizar una segmentación por color dentro de un rango.
<code>cv2.countNonZero()</code>	Cuenta los píxeles distintos de cero en una imagen.
<code>cv2.findContours()</code>	Detecta y extrae los contornos de una imagen.
<code>cv2.RETR_EXTERNAL</code>	Extrae únicamente los contornos externos de la imagen.
<code>cv2.CHAIN_APPROX_SIMPLE</code>	Reduce los puntos necesarios para representar los contornos y optimiza el uso de memoria.

En el cuadro 6 se resumen algunos de los comandos más representativos y frecuentemente utilizados en la librería OpenCV. Estas funciones permiten capturar, procesar y manipular imágenes y videos de forma eficiente, siendo herramientas esenciales para el desarrollo de sistemas de visión por computador como el que se implementa en este proyecto.

2.3. Diseño experimental

En esta sección se describe el diseño experimental del sistema propuesto para la detección de presencia humana en situaciones anómalas dentro de espacios cerrados mediante el uso de un dron equipado con visión artificial. El diseño del sistema se fundamenta en la integración de diferentes subsistemas que permiten la adquisición de imágenes, la transmisión de datos, el procesamiento en tiempo real y la visualización de resultados.

El sistema desarrollado se basa en el uso de un vehículo aéreo no tripulado de tipo cuadricóptero, específicamente el dron DJI Tello, el cual fue seleccionado debido a su facilidad de operación en entornos interiores, su capacidad de transmisión de video en tiempo real y su compatibilidad con herramientas de programación en Python. Este dispositivo actúa como plataforma de adquisición de datos, permitiendo capturar imágenes del entorno mediante su cámara integrada.

La arquitectura del sistema está organizada en cuatro bloques principales: la plataforma aérea, el sistema de comunicación, el sistema de procesamiento

y la interfaz de monitoreo. Estos bloques trabajan de manera conjunta para garantizar el funcionamiento autónomo del sistema de detección.

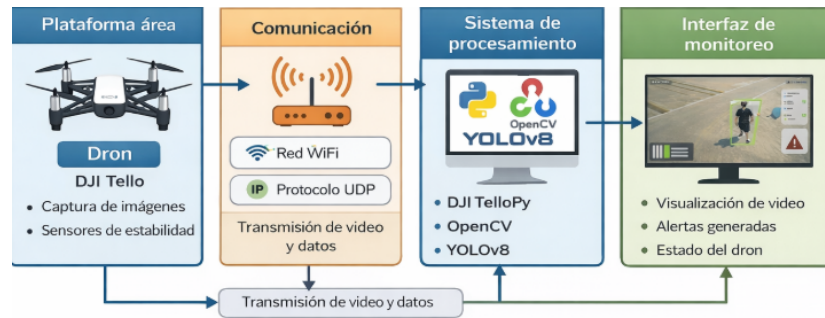


Figura 28. Arquitectura general del sistema de detección de presencia humana mediante dron y visión artificial

Fuente: Autoría propia.

La plataforma aérea está constituida por el dron, el cual incorpora una cámara para la captura de video y sensores internos que permiten mantener la estabilidad durante el vuelo. Este componente es responsable de obtener la información visual del entorno monitoreado.

El sistema de comunicación permite establecer un enlace inalámbrico entre el dron y la computadora de procesamiento mediante una red Wi-Fi generada por el propio dispositivo. A través de esta conexión se transmiten tanto los comandos de control como el flujo de video capturado. La comunicación se realiza utilizando el protocolo UDP, lo que permite un intercambio de datos en tiempo real.

El módulo de procesamiento se ejecuta en una computadora en la que se ha instalado el software creado con el lenguaje de programación Python, librerías DJITelloPy para la comunicación con el dron, OpenCV para el procesamiento de imágenes y modelo de detección YOLOv8 para la detección de humanos en tiempo real. Este módulo se encarga del procesamiento de los fotogramas que le envía el dron para determinar si hay o no humanos en la zona monitoreada.

La interfaz muestra el video, los cuadros delimitadores y la información básica del dron. Su arquitectura y la interacción entre módulos se muestra en la Figura 28.

2.3.1. Diseño e implementación del hardware

El hardware del sistema propuesto está constituido por la plataforma aérea utilizada para la adquisición de datos y la estación de procesamiento

encargada del análisis de la información. En este trabajo se empleó el dron DJI Tello como vehículo aéreo no tripulado, debido a su diseño compacto, facilidad de operación en espacios interiores y capacidad de transmisión de video en tiempo real.

El DJI Tello es una plataforma de arquitectura cerrada, por lo que no permite modificaciones internas directas. Sin embargo, como se muestra en la Figura 29, se identifican sus principales componentes funcionales, entre los que destacan la cámara integrada, los motores, las hélices y el sistema de posicionamiento visual. Estos elementos cumplen un rol fundamental dentro del sistema, ya que permiten la captura de imágenes del entorno y la estabilidad del dron durante el vuelo en espacios confinados.

La cámara del sistema se encuentra integrada en la plataforma, al igual que el resto de sensores, es un componente esencial en el proceso de adquisición de la información visual. A partir de la señal de video generada por la cámara, el sistema de visión artificial procesará la información. Los sensores de localización y los motores serán los encargados de mantener el dron estable y evitar así que se produzcan perturbaciones en la toma de imágenes.

Además del sistema de cámaras, también hay una estación de procesamiento de tipo ordenador que se encarga de la visión artificial y del control del dron. Esta estación, que recibe el flujo de video en tiempo real que envía el UAV, se encarga de procesar los fotogramas y de ejecutar el modelo de detección de personas.

Por lo tanto, el hardware del sistema no se limita al dron, sino que también incluye la estación de procesamiento, de modo que se trata de un sistema distribuido en el que el UAV es la unidad de adquisición y la computadora es la unidad de análisis y control.

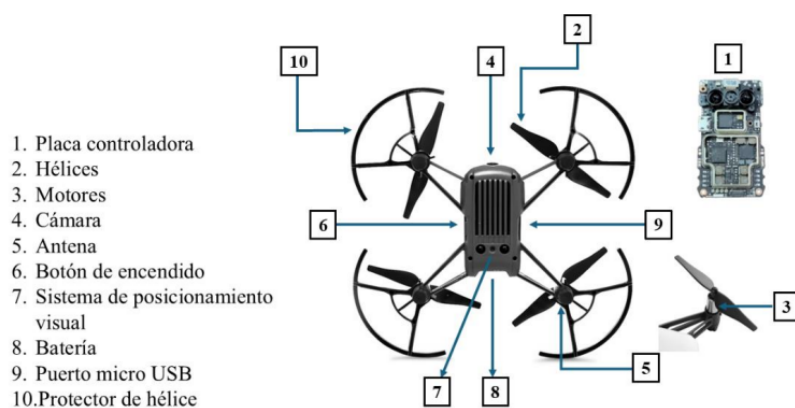


Figura 29. Visualización y reconocimiento de elementos conectados al dron
Fuente: Autoría propia.

2.3.2. Diseño del sistema de comunicación

El sistema de comunicación permite el intercambio de información entre el dron y la estación de procesamiento, constituyendo un componente fundamental para el funcionamiento del sistema propuesto. Esta comunicación se realiza de manera inalámbrica mediante una red Wi-Fi generada por el propio dron, lo que permite establecer un enlace directo entre el vehículo aéreo no tripulado y la computadora de control.

Una vez activado el dron, este habilita una red inalámbrica identificada con el nombre “TELLO”, a la cual se conecta la computadora. Esta conexión establece un enlace punto a punto sin necesidad de infraestructura adicional, lo que resulta adecuado para entornos cerrados donde se desarrollan las pruebas experimentales.

El protocolo de comunicación se basa en el User Datagram Protocol (UDP), de bajo retardo. Esto permite el envío de comandos en tiempo real, para el control del dron. Es la estación de procesamiento la que envía estos comandos por el protocolo, tales como el despegue, el aterrizaje, el desplazamiento y la rotación.

La comunicación con el dron fue realizada a través de la librería DJITelloPy, que proporciona una API de alto nivel que permite comunicarse con el dron a través de Python y enviarle órdenes así como recibir información del estado del dron, como el nivel de batería o parámetros de vuelo.

Además del control de la trayectoria del UAV, el sistema de comunicación permite transmitir el video de la escena captado por la cámara del UAV a la estación de procesamiento. Esta información es importante para que el sistema de visión artificial funcione correctamente, ya que sobre estas imágenes se realizarán las tareas de detección de presencia de personas en tiempo real.

La comunicación inalámbrica permite un control en tiempo real durante el experimento, aunque tiene la desventaja de su limitado alcance (entre 10 y 15 metros en campo abierto), además de que puede verse interferida por otras redes Wi-Fi cercanas.

El proceso de establecimiento de la conexión se resume en la Figura [30](#).

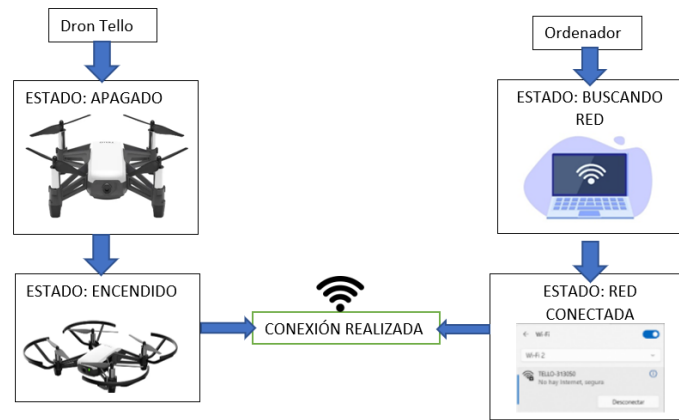


Figura 30. Conexión entre ordenador y red Wi-Fi del dron DJI Tello.
Fuente: Autoría propia.

2.3.3. Diseño del software

El software del sistema fue desarrollado utilizando el lenguaje de programación Python, debido a su amplia disponibilidad de bibliotecas especializadas en visión artificial, procesamiento de imágenes y control de dispositivos externos. El desarrollo se realizó en el entorno de programación PyCharm, el cual facilita la organización y depuración del código.

Selección del entorno de desarrollo: Antes de comenzar con la programación del sistema, se revisaron varias herramientas que permiten trabajar con Python. Para esta comparación se tomaron en cuenta PyCharm, Visual Studio Code, Jupyter Notebook y Spyder, considerando principalmente la facilidad de uso, la organización del proyecto, la depuración del código y la integración de las bibliotecas utilizadas.

En el Cuadro 7 se resumen las características de cada alternativa.

Cuadro 7. Comparación de entornos de desarrollo para Python

Entorno	Características	Ventajas	Consideraciones para el proyecto
PyCharm	Entorno diseñado principalmente para el desarrollo en Python. Incluye editor de código, terminal, depurador y administración de entornos virtuales [66].	Facilita la organización de los archivos, la instalación de bibliotecas y la identificación de errores durante la programación.	Permitió integrar en un mismo proyecto el modelo YOLOv8, la cámara del DJI Tello, la interfaz gráfica y el sistema de alertas.
Visual Studio Code	Editor de código que admite Python mediante la instalación de extensiones [67].	Es ligero, flexible y permite incorporar herramientas según las necesidades del usuario.	Requiere instalar y configurar varias extensiones para disponer de las funciones necesarias.
Jupyter Notebook	Herramienta que permite ejecutar el código por bloques y visualizar los resultados de manera inmediata [68].	Resulta útil para realizar pruebas, analizar datos y mostrar gráficos durante el desarrollo.	Su trabajo por celdas resulta menos adecuado para ejecutar continuamente el video, la interfaz y el control del dron.
Spyder	Entorno orientado al cálculo científico y al análisis de datos mediante Python [69].	Permite observar variables y resultados numéricos durante la ejecución del programa.	Está enfocado principalmente en el análisis científico y no tanto en el desarrollo de una aplicación formada por varios módulos.

Fuente: Autoría propia.

Luego de revisar las alternativas, se eligió PyCharm como entorno de desarrollo, ya que permitió organizar en un solo proyecto las diferentes funciones del sistema. Desde este entorno se trabajó con la transmisión de video, el procesamiento de imágenes, el modelo YOLOv8, el control del DJI Tello, la interfaz gráfica y el envío de alertas mediante WhatsApp.

Otra razón para utilizar PyCharm fue la facilidad para crear un entorno virtual e instalar las bibliotecas necesarias, entre ellas OpenCV, Ultralytics, `djitellopy`, CustomTkinter, Pillow y Requests. También permitió ejecutar y revisar cada parte del programa por separado antes de unir todos los componentes.

Por estas razones, PyCharm fue la herramienta que mejor se ajustó a las

necesidades del proyecto. Aunque las demás alternativas también permiten programar en Python, su forma de trabajo resultaba menos conveniente para una aplicación que debía procesar video en tiempo real, controlar el dron y mantener activa una interfaz gráfica.

Arquitectura del software: El sistema está basado en una arquitectura modular que permite que las diferentes funcionalidades del sistema sean independientes para facilitar su mantenimiento y su integración, que permite que el software esté dividido en distintos módulos que trabajan conjuntamente para la detección de la presencia humana en tiempo real.

Los módulos principales son: Conexión con el dron, captura de video, procesamiento de imágenes, detección de personas y visualización de resultados.

El flujo de trabajo del software comienza con la conexión al dron y la transmisión del video en vivo. A continuación, cada imagen del video es analizada mediante algoritmos de visión artificial y el modelo YOLOv8, que permite la detección de personas con un alto nivel de precisión. El módulo de procesamiento de imágenes se encarga de redimensionar y cambiar el formato de las imágenes obtenidas de la cámara y de prepararlas para ser enviadas al modelo de detección. El modelo YOLOv8 analiza las imágenes y determina si hay personas en su campo de visión.

Finalmente, los resultados se envían al módulo de visualización, que muestra cuadros delimitadores en las personas detectadas, permitiendo al operador del sistema ver el entorno en tiempo real.

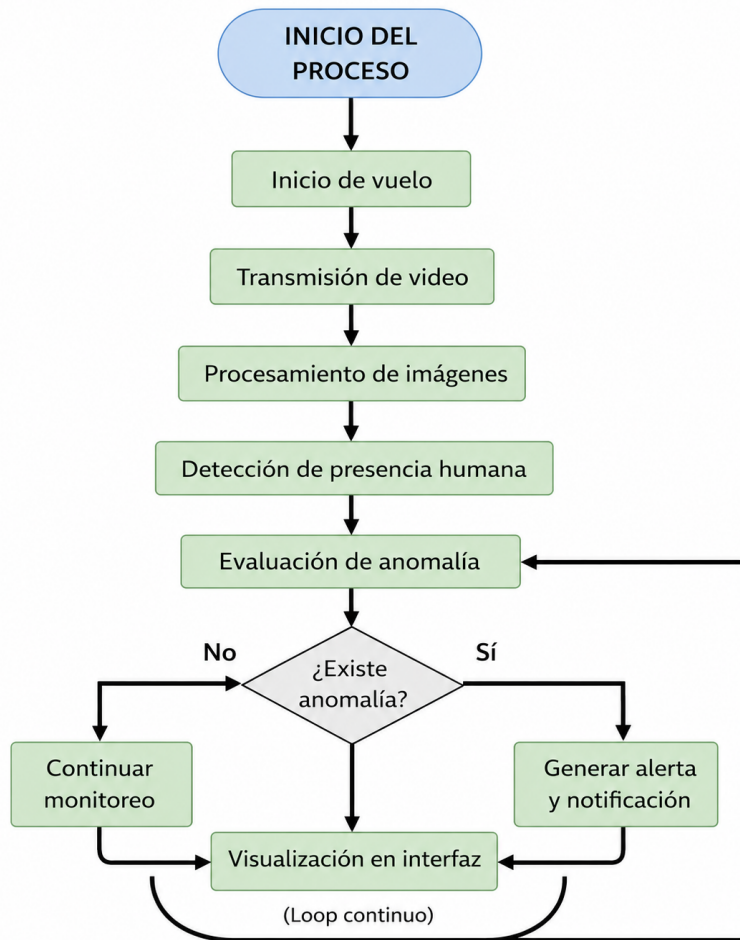


Figura 31. Sistema de control

Fuente: Autoría propia.

2.3.4. Sistema de visión artificial para la detección de personas

El sistema de visión artificial constituye el núcleo funcional del sistema propuesto, ya que es el encargado de analizar las imágenes capturadas por la cámara del dron para identificar la presencia humana dentro del entorno monitoreado. Su implementación se basa en el uso de técnicas de detección de objetos en tiempo real, permitiendo procesar continuamente los fotogramas recibidos desde el flujo de video del UAV.

Desde el punto de vista computacional, cada fotograma puede representarse como una matriz de valores de intensidad en el espacio de color RGB, definida como $I(x, y) \in R^{H \times W \times 3}$, donde H y W representan la altura y el

ancho de la imagen, respectivamente, y el valor 3 corresponde a los canales de color.

Para la detección de personas se empleó el modelo YOLOv8, perteneciente a la familia de algoritmos YOLO (*You only look once*), ampliamente utilizados en aplicaciones de visión artificial por su capacidad para realizar detección de objetos en tiempo real con un equilibrio adecuado entre precisión y velocidad de procesamiento. Este modelo permite identificar la clase de interés dentro de una imagen, así como determinar su ubicación mediante cuadros delimitadores.

El funcionamiento del sistema inicia con la recepción de los fotogramas transmitidos por la cámara del dron. Posteriormente, cada imagen es sometida a una etapa de preprocesamiento, en la cual se realizan operaciones de redimensionamiento, conversión de formato y preparación de los datos de entrada requeridos por el modelo. Estas operaciones son implementadas mediante la biblioteca OpenCV.

Una vez realizado el preprocesamiento, el fotograma es procesado por el modelo YOLOv8 mediante una función de detección que puede expresarse como:

$$f(I) = \{(b_i, c_i, p_i)\}_{i=1}^N \quad (1)$$

Donde b_i es la localización del objeto en un cuadro delimitador, c_i es la clase del objeto (en este caso, persona) y p_i es la probabilidad de que la detección sea correcta.

Cuando el modelo detecta una persona en un área determinada, genera un cuadro delimitador en la ubicación de la detección y una puntuación de confianza. Esta información permite que la interfaz marque la presencia humana en tiempo real. La implementación permite analizar automáticamente el entorno y reduce la necesidad de vigilancia humana. Por lo tanto, el subsistema de visión artificial se hace cargo de la detección de presencia humana, que es el objetivo de esta investigación.

En una toma en la que se haya detectado una persona, el modelo genera un cuadro delimitador que encapsula la región de la imagen donde se ha detectado a la persona y un nivel de confianza que se utiliza para mostrar gráficamente las detecciones de personas en tiempo real en la interfaz de usuario.

2.3.5. Diseño del sistema de control del dron

El sistema de control del dron tiene como objetivo ejecutar acciones de navegación en función de la información obtenida por el sistema de visión

artificial. Este subsistema permite que el dron responda de manera dinámica ante la detección de presencia humana dentro del entorno monitoreado.

El control del dron se implementa mediante el envío de comandos desde la estación de procesamiento hacia el UAV, utilizando la librería *DJITelloPy*. A través de esta interfaz, es posible controlar los movimientos del dron, tales como despegue, aterrizaje, desplazamientos laterales, avance, retroceso y rotación.

El sistema de control sigue un enfoque reactivo, en el cual las decisiones se toman en tiempo real en función de los resultados generados por el modelo de visión artificial. Cuando el sistema detecta la presencia de una persona, se pueden ejecutar acciones específicas como mantener la posición, ajustar la orientación del dron o generar alertas visuales en la interfaz de monitoreo.

En condiciones normales, cuando no se detecta ninguna anomalía, el dron continúa en estado de observación, realizando la captura continua de imágenes del entorno. Este comportamiento permite garantizar un monitoreo constante sin intervención manual.

El control del dron se realiza mediante el envío de comandos discretos a través del protocolo de comunicación UDP, lo que permite una respuesta rápida y adecuada durante las pruebas experimentales. Sin embargo, debido a las limitaciones del entorno interior, el sistema no implementa un control autónomo completo de navegación, sino un control asistido basado en la supervisión del sistema de visión artificial.

De esta manera, el sistema de control actúa como un puente entre la percepción del entorno (visión artificial) y la acción del dron, permitiendo la ejecución de respuestas ante eventos detectados y contribuyendo al cumplimiento de los objetivos del sistema.

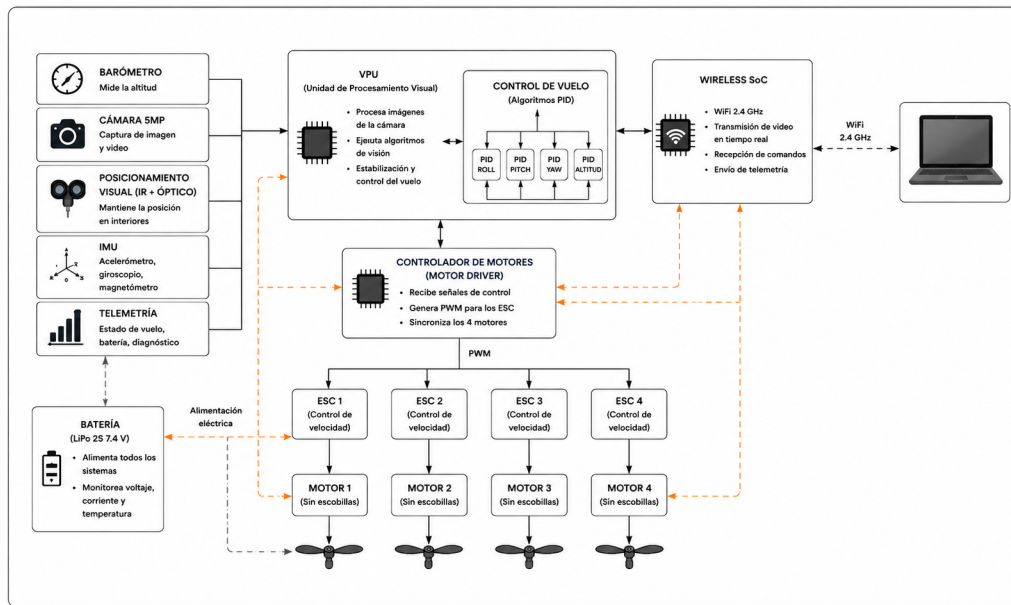


Figura 32. Arquitectura interna del sistema de control del dron Tello.

Fuente: [70].

La Figura 32 presenta la arquitectura interna del sistema de control del dron Tello. El sistema integra sensores como el barómetro, la cámara de 5 MP, el sistema de posicionamiento visual y la unidad de medición inercial (IMU), encargados de recopilar información del entorno y del estado del vuelo.

La unidad de procesamiento visual (VPU) ejecuta algoritmos de visión y estabilización, mientras que el módulo de control de vuelo procesa las señales necesarias para mantener la estabilidad del cuadricóptero mediante algoritmos PID. Posteriormente, el controlador de motores genera señales PWM hacia los ESC y motores brushless para controlar la velocidad de las hélices.

Además, el módulo inalámbrico permite la comunicación Wi-Fi a 2,4 GHz con la computadora encargada del monitoreo y procesamiento adicional mediante técnicas de visión artificial con YOLOv8.

2.3.6. Diseño de la interfaz de usuario

La interfaz de usuario fue diseñada con el objetivo de proporcionar una interacción clara e intuitiva entre el operador y el sistema de monitoreo basado en el dron Tello. La estructura propuesta organiza las diferentes funciones del sistema en módulos independientes, permitiendo controlar el dron, supervisar el estado del vuelo y visualizar las detecciones realizadas mediante visión artificial en tiempo real.

La interfaz se divide en diferentes secciones funcionales. El módulo de conexión permite establecer la comunicación entre el dron y la computadora mediante Wi-Fi, además de mostrar el estado actual de la conexión. El módulo de control de vuelo integra funciones básicas como despegue, aterrizaje, emergencia y calibración del sistema IMU del dron.

Por otra parte, la sección de vista de video está destinada a la visualización en tiempo real de la transmisión capturada por la cámara del dron, permitiendo observar el entorno monitoreado y las detecciones generadas por el modelo de inteligencia artificial. Asimismo, la interfaz incorpora un módulo de telemetría encargado de mostrar parámetros importantes del vuelo, como altitud, batería, velocidad, modo de vuelo y calidad de señal inalámbrica.

El sistema también incluye un módulo de detección de personas, donde se presentan los resultados obtenidos mediante el modelo YOLOv8, mostrando información relacionada con las personas detectadas y sus respectivas posiciones dentro de la escena analizada. Adicionalmente, se incorpora una consola de eventos destinada al registro de mensajes y estados del sistema durante el funcionamiento en tiempo real.

Finalmente, la interfaz contempla módulos adicionales para la selección del modo de vuelo, visualización de posición y configuración general del sistema, permitiendo adaptar parámetros internos relacionados con el monitoreo y funcionamiento del dron.

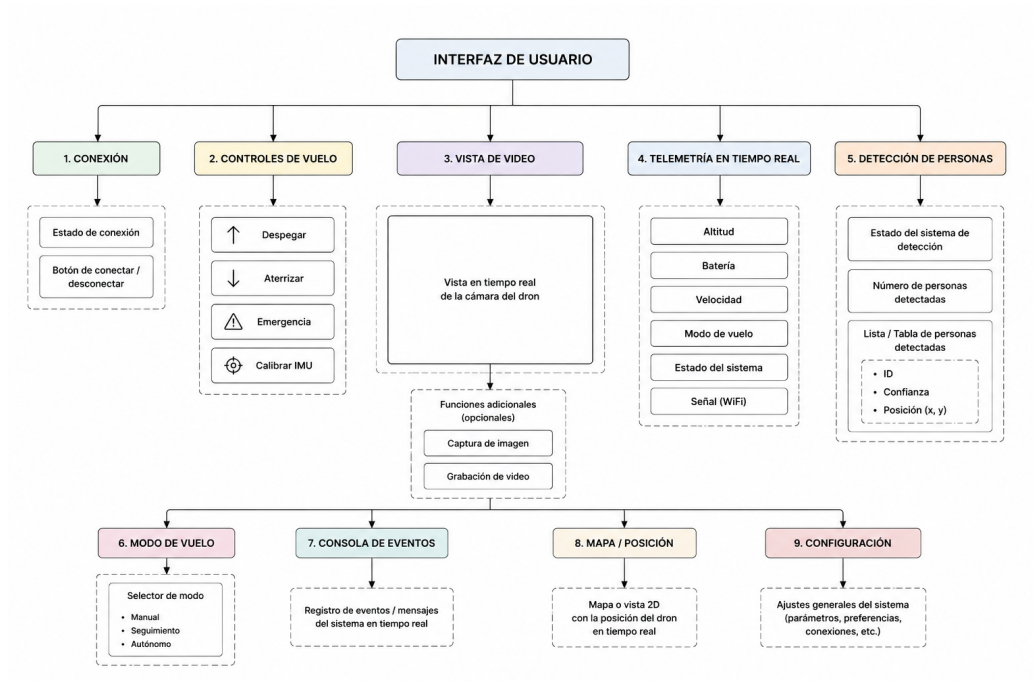


Figura 33. Diseño general de la interfaz de usuario.

Fuente: Autoría propia.

La Figura 33 presenta la estructura general propuesta para la interfaz de usuario del sistema. En ella se observan los diferentes módulos funcionales encargados de la conexión, control del dron, visualización de video, monitoreo de telemetría, detección de personas y configuración del sistema. Esta organización modular permite mejorar la interacción entre el operador y el sistema durante el monitoreo en tiempo real.

2.4. Desarrollo experimental

En esta sección se detalla la ejecución práctica del sistema propuesto, enfocándose en la integración física y lógica de sus componentes. Se documenta cómo se realizó la implementación del prototipo funcional para la detección de presencia humana mediante visión artificial, resaltando el flujo del desarrollo, las herramientas empleadas y las decisiones tomadas a lo largo del proceso.

2.4.1. Implementación del sistema de adquisición de datos

La implementación del sistema de adquisición de datos se realizó utilizando el dron DJI Tello como plataforma principal para la captura de imágenes

y video en tiempo real. Esta etapa tuvo como finalidad establecer la comunicación entre el dron y la computadora, así como obtener información visual necesaria para el desarrollo del sistema de detección de anomalías mediante visión artificial.

Inicialmente, se preparó el entorno de pruebas dentro de un espacio cerrado, permitiendo realizar vuelos controlados y capturas de imágenes bajo diferentes condiciones de iluminación y posición. El dron fue ubicado dentro del área de monitoreo para posteriormente establecer la conexión inalámbrica mediante la red Wi-Fi generada por el DJI Tello.



Figura 34. Entorno utilizado para las pruebas de adquisición de datos.

Fuente: Autoría propia.

Una vez establecida la conexión, se utilizó Python junto con la librería `djitellopy` para controlar el dron y acceder a la transmisión de video proporcionada por la cámara integrada. Esto permitió enviar comandos básicos de vuelo, obtener información de telemetría y recibir fotogramas en tiempo real desde la cámara del dron.

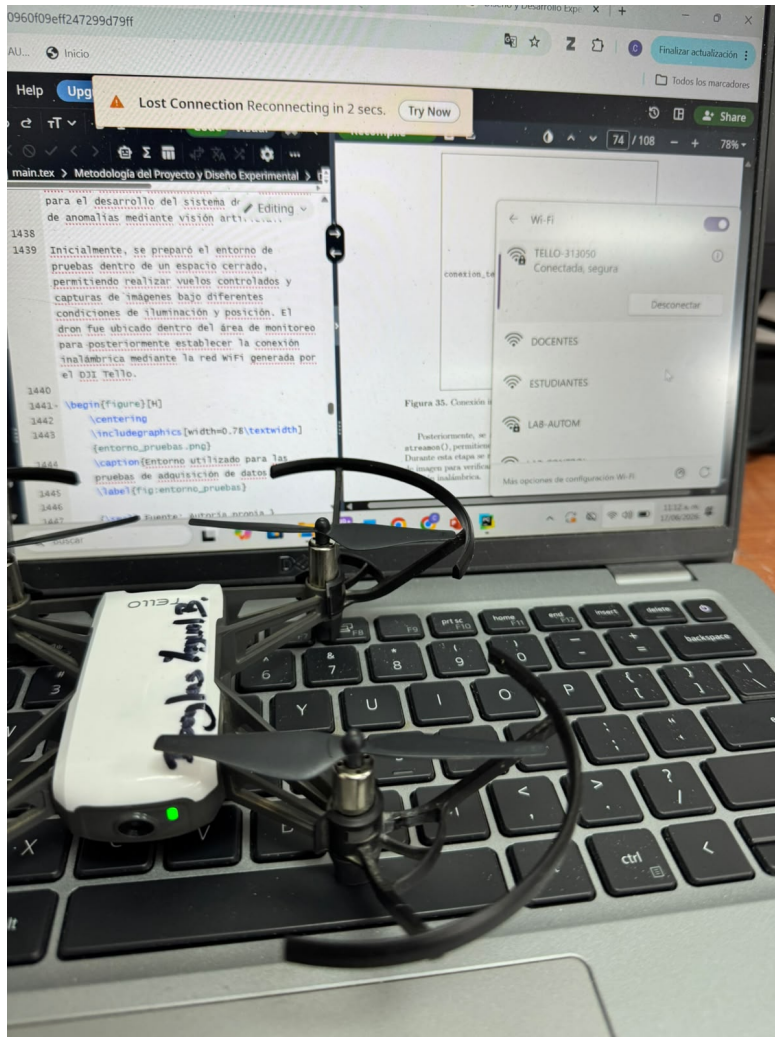


Figura 35. Conexión inalámbrica entre el dron DJI Tello y la computadora.
Fuente: Autoría propia.

Posteriormente, se activó la transmisión de video mediante la función `streamon()`, permitiendo visualizar las imágenes capturadas utilizando OpenCV. Durante esta etapa se realizaron diferentes pruebas de estabilidad y calidad de imagen para verificar el correcto funcionamiento de la cámara y la comunicación inalámbrica.

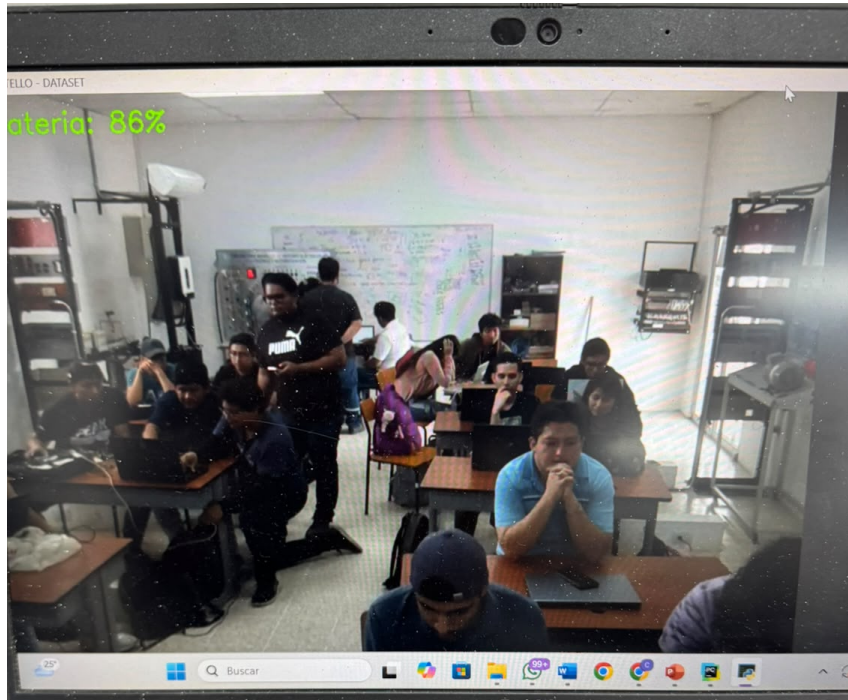


Figura 36. Visualización en tiempo real de la cámara del DJI Tello mediante OpenCV.

Fuente: Autoría propia.

Además, se realizaron ajustes relacionados con la representación de color debido a diferencias entre los formatos RGB y BGR utilizados por OpenCV, permitiendo mejorar la visualización de los fotogramas obtenidos desde el dron.

Finalmente, las imágenes capturadas durante las pruebas fueron utilizadas como parte del dataset personalizado del proyecto. Estas capturas permitieron incorporar escenarios reales obtenidos desde la perspectiva del dron, mejorando posteriormente el entrenamiento y desempeño del modelo YOLOv8 para la detección de anomalías en espacios cerrados.



Figura 37. Ejemplos de imágenes capturadas para la construcción del dataset.

Fuente: Autoría propia.

2.4.2. Preparación del dataset y entrenamiento del modelo

En esta etapa se realizó la construcción y preparación del dataset utilizado para el entrenamiento del modelo de visión artificial YOLOv8. El objetivo

principal fue recopilar imágenes relacionadas con diferentes anomalías presentes en espacios cerrados, permitiendo posteriormente entrenar un modelo capaz de detectar dichas situaciones en tiempo real.

Inicialmente, se recopilaron imágenes provenientes de diferentes fuentes, incluyendo capturas reales obtenidas mediante el dron DJI Tello y fotografías obtenidas de internet. Las clases definidas para el sistema son las de personas usando teléfono celular, personas durmiendo, personas consumiendo alimentos y caídas. El conjunto de datos final contiene 1694 imágenes en 4 clases (teléfono, dormir, comida y caída.). Las imágenes se obtuvieron mediante fotografías propias y fuentes de internet, bajo distintas condiciones de iluminación, posición y ángulo. Para que el resultado se pueda reproducir, se debe incluir el número de imágenes por clase en el archivo de entrenamiento.



Figura 38. Imágenes utilizadas para la construcción del dataset.

Fuente: Autoría propia.

Posteriormente, las imágenes fueron etiquetadas utilizando el formato YOLO, asignando manualmente cuadros delimitadores (*bounding boxes*) sobre las regiones de interés correspondientes a cada anomalía detectada. Este procedimiento permitió generar archivos `.txt` asociados a cada imagen, conteniendo la información de posición y clase requerida por el modelo YOLOv8.

```
0 0.505068 0.422720 0.496622 0.704392
```

Figura 39. Proceso de etiquetado de imágenes en formato YOLO.
Fuente: Autoría propia.

Una vez finalizado el etiquetado, se procedió a reorganizar el dataset para su utilización dentro del entrenamiento del modelo. Para ello, se unificaron las diferentes clases en una sola estructura de entrenamiento compuesta por carpetas de imágenes y etiquetas para entrenamiento y validación. Además, se configuró el archivo `data.yaml`, donde se definieron las rutas del dataset y las clases correspondientes al sistema de detección.

```
! data.yaml X
C: > Users > Admin > Documents > Dataset > DATASET_FINAL > ! data.yaml
1 path: C:/Users/Admin/Documents/Dataset/DATASET_FINAL
2 train: train/images
3 val: valid/images
4
5 nc: 5
6 names:
7   0: Arma
8   1: Caída
9   2: Comida
10  3: Dormir
11  4: Telefono
12
```

Figura 40. Estructura general del dataset utilizado para el entrenamiento del modelo.
Fuente: Autoría propia.

Posteriormente, se realizó el entrenamiento del modelo utilizando YOLOv8 mediante Python y la librería Ultralytics. El entrenamiento se ejecutó empleando el modelo base YOLOv8n debido a su capacidad de procesamiento en tiempo real y menor consumo computacional.

Durante el proceso de entrenamiento se configuraron parámetros como el número de épocas, resolución de entrada, tamaño de lote y rutas del dataset. El objetivo del entrenamiento fue permitir que el modelo aprendiera características visuales asociadas a cada anomalía presente en el entorno monitoreado.

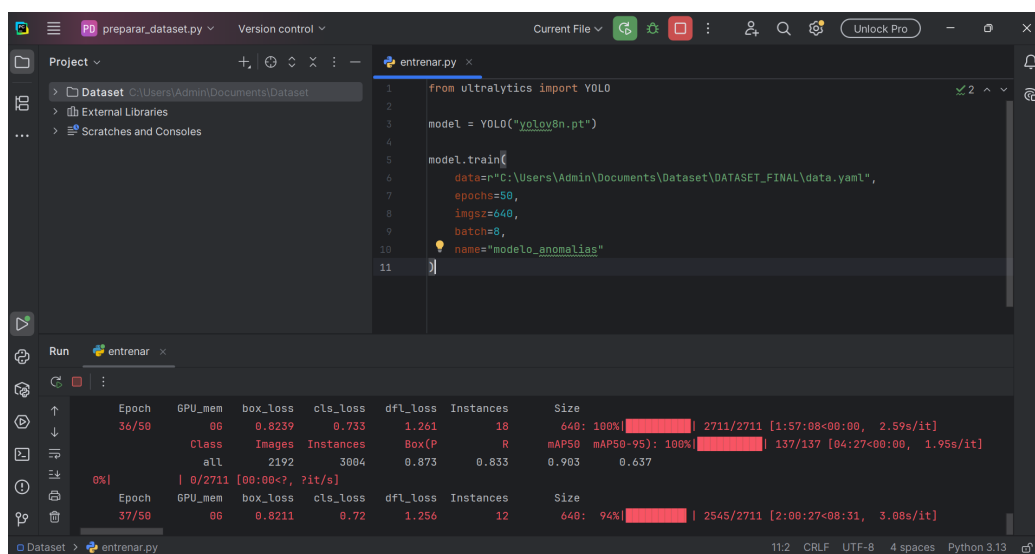


Figura 41. Proceso de entrenamiento del modelo YOLOv8.

Fuente: Autoría propia.

Finalmente, el entrenamiento generó un modelo .pt capaz de realizar detección de anomalías en tiempo real mediante la cámara del dron DJI Tello, permitiendo posteriormente integrarlo dentro del sistema de monitoreo desarrollado.

2.4.3. Implementación del sistema de procesamiento

La implementación del sistema de procesamiento se realizó con el objetivo de analizar los fotogramas obtenidos desde la cámara del dron DJI Tello y aplicar el modelo de visión artificial entrenado para la detección de anomalías. Esta etapa constituye el núcleo lógico del sistema, ya que permite transformar la información visual capturada en datos útiles para la toma de decisiones.

Inicialmente, los fotogramas recibidos desde el dron fueron procesados mediante la biblioteca OpenCV, la cual permitió realizar operaciones básicas como redimensionamiento, conversión de color y visualización en tiempo real. Durante las pruebas iniciales se corrigió la representación de color de los fotogramas mediante la conversión de RGB a BGR, debido a que OpenCV utiliza el formato BGR para la visualización correcta de imágenes.

```

frame = frame_read.frame

if frame is not None:
    frame = cv2.resize(frame, dsize=(560, 360))
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

    results = model(frame, conf=CONF, verbose=False)
    annotated = results[0].plot()
    detecciones = results[0].boxes

```

Figura 42. Procesamiento inicial de fotogramas mediante OpenCV.

Fuente: Autoría propia.

Posteriormente, se integró el modelo YOLOv8 entrenado con el dataset personalizado del proyecto. Para ello, se cargó el archivo `best.pt`, generado durante el proceso de entrenamiento, el cual contiene los pesos finales del modelo ajustado para detectar las clases de anomalías definidas: caída, comida, dormir y teléfono.

El procesamiento se ejecutó fotograma por fotograma, aplicando inferencia sobre cada imagen recibida. Como resultado, el modelo generó cuadros delimitadores, etiquetas de clase y valores de confianza asociados a cada detección. Esta información permitió identificar visualmente la anomalía detectada dentro de la escena monitoreada.

```

model = YOLO(MODEL_PATH)

results = model(frame, conf=CONF, verbose=False)

detecciones = results[0].boxes

annotated = results[0].plot()

```

Figura 43. Aplicación del modelo YOLOv8 sobre los fotogramas procesados.

Fuente: Autoría propia.

Para facilitar la interpretación de los resultados, las detecciones fueron superpuestas sobre el fotograma original mediante cuadros delimitadores y etiquetas generadas por el modelo. Esta visualización también proporcionó

una comprobación de la clase identificada y de la confianza que se tenía en cada predicción.

```
if len(detecciones) > 0:

    mejor = detecciones[0]

    cls_id = int(mejor.cls[0])
    conf = float(mejor.conf[0])

    clase = model.names[cls_id]
```

Figura 44. Visualización de anomalías detectadas mediante cuadros delimitadores.

Fuente: Autoría propia.

Asimismo, el sistema de procesamiento fue conectado a la HMI (Interfaz hombre-máquina), donde se visualizaba en tiempo real el video procesado, el estado de alerta y la clase de anomalía detectada. De esta manera, fue posible establecer la comunicación entre el sistema de visión artificial y el sistema de supervisión desarrollado.

2.4.4. Implementación del sistema de detección de anomalías

La implementación del sistema de detección de anomalías se desarrolló con el propósito de identificar comportamientos y situaciones consideradas anormales dentro del entorno monitoreado mediante visión artificial y aprendizaje profundo. Esta etapa permitió integrar el modelo YOLOv8 entrenado con el flujo de video capturado por el dron DJI Tello para realizar detecciones en tiempo real.

Inicialmente, el sistema recibió los fotogramas provenientes de la cámara del dron, los cuales fueron procesados continuamente mediante el modelo de detección entrenado. Cada imagen capturada fue analizada utilizando inferencia en tiempo real, permitiendo identificar las diferentes clases de anomalías definidas durante la construcción del dataset.

Las anomalías consideradas dentro del sistema corresponden a:

- Personas usando teléfono celular.
- Personas durmiendo.
- Personas consumiendo alimentos.

- Caídas.

Durante el procesamiento, el modelo generó cuadros delimitadores (*bounding boxes*) alrededor de los objetos o personas detectadas, mostrando además el nombre de la clase y el nivel de confianza asociado a cada predicción. Esta información permitió identificar visualmente la anomalía presente dentro de la escena monitoreada.



Figura 45. Detección de anomalías en tiempo real mediante YOLOv8.

Fuente: Autoría propia.

La detección de anomalías es un paso posterior a la detección de personas, y se centra en encontrar situaciones que no se comportan con normalidad en el entorno que capturan las cámaras.

Dado que el sistema puede implementarse en espacios interiores donde las personas son comunes, el mero hecho de detectar personas no se consideró un evento anómalo y se definió un segundo criterio, que tendría en cuenta la ubicación espacial de las personas en el campo visual.

Para ello, el sistema utiliza la información proporcionada por el modelo de detección YOLOv8, que consiste en las coordenadas de los cuadros delimitadores de los objetos detectados en el frame, expresadas como (x_1, y_1, x_2, y_2) , y a partir de estos valores se extrae el centro del objeto detectado con la siguiente fórmula:

$$x_c = \frac{x_1 + x_2}{2}, \quad y_c = \frac{y_1 + y_2}{2} \quad (2)$$

Este punto central permite determinar de manera eficiente la posición de la persona dentro del fotograma.

La detección de anomalías se basa en la definición de una región de interés (ROI, por sus siglas en inglés) dentro del entorno monitoreado, la cual representa una zona restringida. Esta región es definida mediante límites espaciales en el plano de la imagen.

Una detección se considera anómala cuando el punto central (x_c, y_c) se encuentra dentro de los límites de la región de interés definida. En caso contrario, la detección es clasificada como una condición normal.

Si se cumplen las condiciones que permiten determinar que existe una anomalía, el sistema generará una señal de alerta que aparecerá en un panel de control. Si no se detectan anomalías, el sistema continuará funcionando en modo de supervisión y no se generarán alertas.

Esto no sólo mejora la precisión del sistema y reduce la tasa de falsas alarmas, sino que también permite más flexibilidad a la hora de adaptar el sistema a diferentes escenarios al cambiar la región de interés.

Dado que la detección de anomalías se realiza de forma continua en cada fotograma, el sistema puede reaccionar en tiempo real a la aparición de eventos que cumplan las condiciones.

Posteriormente, las detecciones obtenidas mediante YOLOv8 fueron integradas con el sistema de procesamiento en tiempo real, permitiendo visualizar las anomalías detectadas directamente sobre los fotogramas capturados por el dron DJI Tello. Para ello, el sistema superpone cuadros delimitadores, etiquetas de clase y niveles de confianza sobre cada detección realizada.

Asimismo, el sistema fue integrado con la interfaz HMI desarrollada para el monitoreo, permitiendo mostrar el estado de alerta y la anomalía detectada durante la supervisión en tiempo real.

Finalmente, el sistema permitió realizar detecciones automáticas sobre el video transmitido por el dron DJI Tello, facilitando la supervisión de anomalías en espacios cerrados mediante técnicas de visión artificial y aprendizaje profundo.

2.4.5. Implementación del sistema de alertas mediante WhatsApp

Para informar al usuario sobre las anomalías detectadas, se incorporó al programa un sistema de alertas mediante WhatsApp. Este módulo trabaja en conjunto con el modelo YOLOv8 y se activa cuando una detección alcanza el umbral de confianza establecido.

En el programa se configuró un umbral de confianza del 65 %. Por lo tanto, las detecciones con valores inferiores no generan una alerta. Esta condición ayuda a reducir el envío de mensajes producidos por predicciones poco

confiables.

La condición utilizada para activar una alerta se presenta en la Ecuación 3.

$$A = \{ 1, \text{ si } C \geq C_{\min} y k \in \mathcal{K}_A, 0, \text{ en caso contrario} \} \quad (3)$$

donde A representa el estado de la alerta, C es la confianza obtenida por YOLOv8, $C_{\min}$ corresponde al umbral mínimo configurado, k representa la clase detectada y $\mathcal{K}_A$ agrupa las clases consideradas anómalas: Caída, comida, dormir y teléfono.

Obtención de la clave API: Para enviar los mensajes se utilizó el servicio CallMeBot, el cual permite realizar la comunicación entre un programa desarrollado en Python y WhatsApp mediante una solicitud por Internet [71].

Antes de integrar el servicio al programa fue necesario obtener una clave API. Para ello, se agregó a los contactos del teléfono el número proporcionado por CallMeBot:

+34 644 95 73 56

Luego, desde el mismo número de WhatsApp que recibiría las alertas, se envió el siguiente mensaje:

I allow callmebot to send me messages

Después de enviar el mensaje, el servicio confirmó la activación y proporcionó una clave API asociada al número telefónico. Esta clave se utilizó posteriormente en el programa para autorizar el envío de las notificaciones.

El número receptor se ingresó con el código internacional correspondiente. En el caso de Ecuador, se utilizó el prefijo +593, seguido del número telefónico sin el cero inicial.

Dentro del código, el número y la clave se guardaron en las variables TELEFONO y APIKEY. Por seguridad, estos datos no se muestran en el documento y fueron ocultados en la captura del programa.

Programación del envío de la alerta: Cuando YOLOv8 detecta una anomalía, el programa obtiene el nombre de la clase y el nivel de confianza. También registra la fecha y la hora en la que ocurrió la detección. Con estos datos se genera el mensaje que se envía mediante WhatsApp.

La notificación presenta una estructura similar a la siguiente:

ALERTA DE ANOMALÍA
Clase detectada: Teléfono
Confianza: 0.91
Fecha y hora: 16/07/2026 22:30:15

El nombre de la clase y el nivel de confianza cambian según la detección realizada por el modelo. De esta forma, cada mensaje contiene la información correspondiente al evento observado por la cámara del DJI Tello.

Antes de realizar el envío, el contenido del mensaje se procesa con la función `quote()`. Esta función permite incluir correctamente espacios, saltos de línea, tildes y otros caracteres especiales dentro de la dirección utilizada por la API.

Después, el programa construye la dirección de envío con el número telefónico, el mensaje y la clave API. La solicitud se realiza mediante la biblioteca `requests`. También se estableció un tiempo máximo de espera de diez segundos para evitar que el programa quede detenido si existe un problema con la conexión.

La Figura [46](#) muestra la parte del código utilizada para configurar y enviar las alertas.

```

# =====
# CONFIGURACIÓN DE WHATSAPP
# =====

TELEFONO = "+593xxxxxxxx"
APIKEY = "66xxxx"

ENVIAR_WHATSAPP = True
TIEMPO_ENTRE_ALERTAS = 30 # segundos para evitar mensajes repetidos

ultima_alerta_whatsapp = 0
enviando_whatsapp = False

ultima_alerta_whatsapp = ahora
enviando_whatsapp = True

fecha = datetime.now().strftime("%d/%m/%Y %H:%M:%S")

mensaje = (
    "🚨 ALERTA DE ANOMALÍA\n"
    f"Clase detectada: {clase}\n"
    f"Confianza: {confianza:.2f}\n"
    f"Fecha y hora: {fecha}"
)

texto = quote(mensaje)
url = f"https://api.callmebot.com/whatsapp.php?phone={TELEFONO}&text={texto}&apikey={APIKEY}"

try:
    respuesta = requests.get(url, timeout=10)

```

Figura 46. Código utilizado para el envío de alertas mediante WhatsApp

Fuente: Autoría propia.

La variable `ENVIAR_WHATSAPP` permite activar o desactivar el envío de mensajes. Cuando esta opción se encuentra desactivada, el sistema continúa detectando anomalías, pero no envía notificaciones.

También se estableció un intervalo de treinta segundos entre alertas. Este tiempo evita que una misma anomalía produzca varios mensajes mientras permanece visible en fotogramas consecutivos.

El envío se ejecuta en un hilo independiente mediante la biblioteca `threading`. Esto permite realizar la solicitud en segundo plano, sin detener la reproducción del video ni bloquear la interfaz gráfica.

Además del mensaje, el programa guarda una captura de la anomalía en la carpeta `capturas_anomalias`. El nombre del archivo contiene la clase

detectada, el nivel de confianza y la fecha del evento, lo que permite identificar las imágenes con mayor facilidad.

En esta implementación, la imagen queda almacenada como evidencia en la computadora, mientras que el mensaje enviado mediante CallMeBot contiene la clase, la confianza y la fecha de la detección. Los mensajes recibidos durante las pruebas se presentan posteriormente en el capítulo de resultados.

2.4.6. Implementación de la interfaz gráfica

En esta etapa se implementó una interfaz gráfica tipo HMI para permitir la supervisión del sistema de detección de anomalías en tiempo real. La interfaz fue desarrollada en Python con el objetivo de integrar en una sola ventana la visualización del video, el estado del sistema, las detecciones realizadas por el modelo YOLOv8 y la información básica del monitoreo.

La interfaz se estructuró en diferentes módulos funcionales. El área principal corresponde al video en vivo, donde se muestra la transmisión capturada desde la cámara del dron DJI Tello o desde una cámara de prueba durante las fases iniciales. Sobre esta imagen se visualizan los cuadros delimitadores generados por el modelo de detección, indicando la clase identificada y su nivel de confianza.

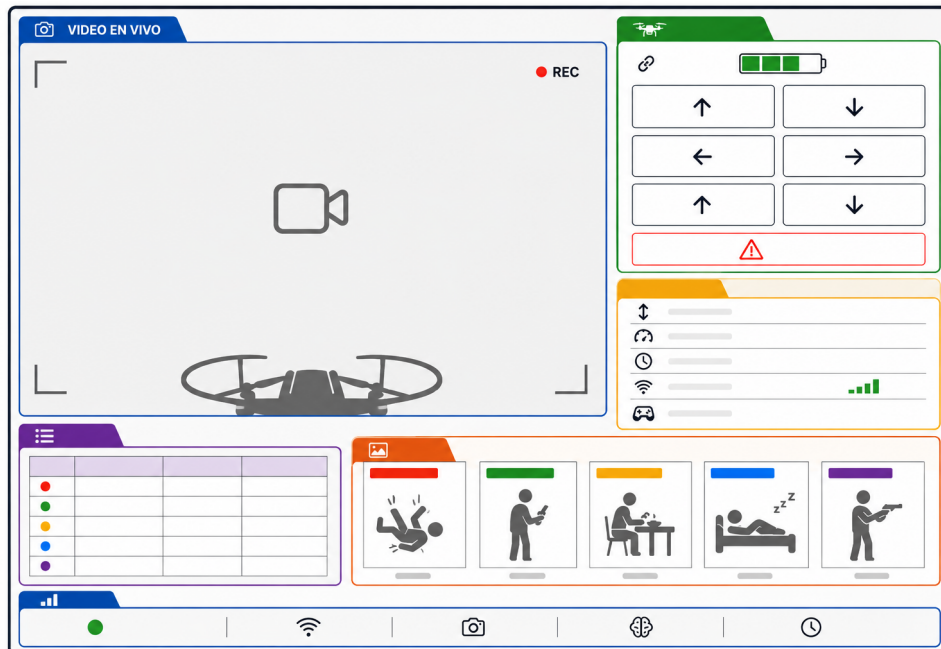


Figura 47. Esquema general de la interfaz gráfica propuesta para el sistema de monitoreo.

Fuente: Autoría propia.

Se incluyó un panel de control que proporciona información sobre el estado del sistema, la batería, la conexión y los comandos básicos de operación del dron, para que el operador tenga una visión general del estado del sistema durante el monitoreo.

Aparte de esto también hay una parte dedicada a las detecciones donde se puede ver lo que ha detectado el modelo YOLOv8, su confianza y la hora a la que lo ha detectado, lo que permite hacer un seguimiento visual de los eventos que se han producido en la supervisión. También se incluyen paneles como la velocidad del procesador y el estado de la conexión Wi-Fi que permiten al usuario monitorear el rendimiento del sistema en tiempo real mientras el sistema está en uso.

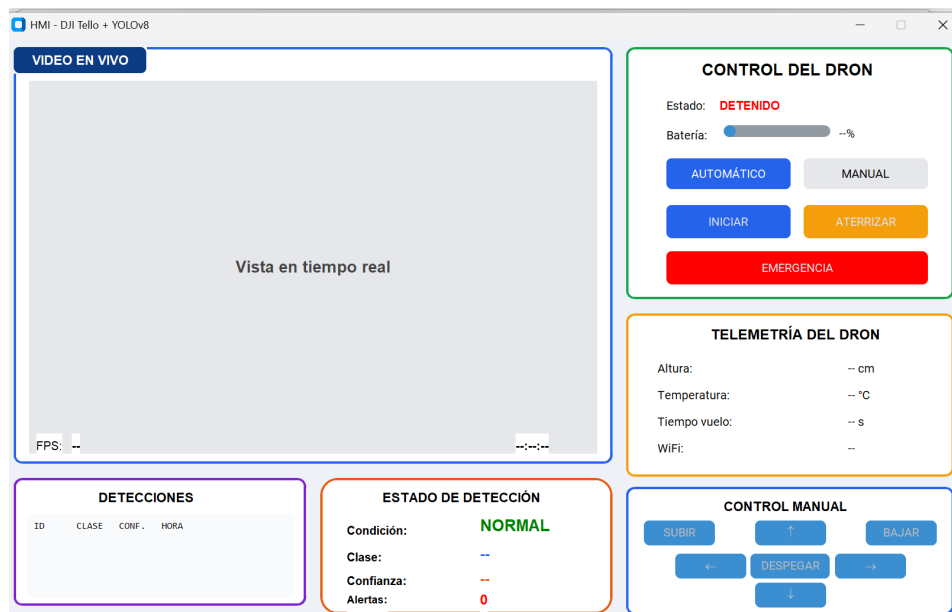


Figura 48. Implementación de la interfaz gráfica desarrollada en Python.
Fuente: Autoría propia.

Finalmente, se añadieron botones de control para iniciar y detener el monitoreo, así como una opción de cierre seguro del sistema. Esta estructura permitió integrar en una sola ventana el procesamiento de video, la detección de anomalías y la supervisión general del sistema basado en visión artificial.

2.4.7. Configuración del sistema

La configuración del sistema se realizó para lograr que el sistema opere de la mejor manera posible y de forma estable al momento de realizar la

detección de anomalías en tiempo real. Para esto, los parámetros de operación del sistema fueron definidos y calibrados experimentalmente, en base al rendimiento del modelo de visión artificial y las características del entorno laboral.

Para la implementación del modelo YOLOv8 se utilizó el archivo de pesos entrenado `best.pt`, el cual se genera durante la etapa de entrenamiento del modelo sobre el dataset personalizado. El sistema utiliza un umbral de confianza que permite filtrar las detecciones que no tienen una probabilidad alta de acierto, de manera que se disminuyan las falsas alarmas y se incremente la confiabilidad del sistema.

A través de la librería OpenCV se realizaba la captura de fotogramas uno a uno desde la cámara de un dron DJI Tello. Se alcanzó un balance entre la velocidad de procesamiento y la precisión del modelo de detección de objetos al ajustar la resolución con la que se procesaba la imagen.

Para la identificación de anomalías se definieron regiones de interés (ROI), estableciendo zonas específicas dentro del campo visual donde el sistema debía realizar el monitoreo y generar alertas únicamente cuando una detección se encontraba dentro de dichas regiones.

Finalmente, durante las pruebas experimentales se realizaron ajustes empíricos sobre los parámetros de funcionamiento, incluyendo la resolución de procesamiento, el umbral de confianza y la configuración de las regiones de interés, obteniendo un sistema con un desempeño estable, capaz de realizar detección continua en tiempo real y adaptarse a las condiciones del entorno de operación.



Figura 49. Configuración general del sistema de detección implementado.
Fuente: Autoría propia.

2.4.8. Condiciones de implementación y limitaciones del sistema

El sistema fue implementado y probado en un entorno interior controlado, simulando un espacio cerrado para realizar las pruebas de monitoreo y detección de anomalías. Durante las pruebas, el dron DJI Tello operó a una altura moderada, permitiendo obtener imágenes adecuadas para el procesamiento mediante visión artificial.



Figura 50. Pruebas realizadas durante la implementación experimental del sistema.

Fuente: Autoría propia.

Las pruebas se realizaron bajo diferentes condiciones de iluminación y disposición del entorno, permitiendo evaluar el comportamiento del sistema ante variaciones reales durante el monitoreo.

El procesamiento de video y la ejecución del modelo YOLOv8 se realizaron en una computadora convencional, encargada de recibir el flujo de video y ejecutar la detección en tiempo real.

A pesar del funcionamiento obtenido, el sistema presenta ciertas limitaciones relacionadas con la iluminación del entorno, estabilidad de la conexión inalámbrica y perspectiva de la cámara del dron. Asimismo, la duración de la batería del DJI Tello limita el tiempo continuo de operación durante las pruebas experimentales.

3. Resultados y conclusiones

3.1. Resultados

3.1.1. Resultados del entrenamiento del modelo

El modelo YOLOv8 fue entrenado utilizando imágenes correspondientes a diferentes comportamientos normales y anómalos dentro del entorno monitoreado. Durante el proceso de entrenamiento se evaluaron parámetros relacionados con precisión, capacidad de detección y desempeño general del sistema de visión artificial.

El entrenamiento permitió obtener un modelo capaz de identificar personas y detectar comportamientos específicos en tiempo real mediante el análisis de video capturado por el dron DJI Tello.

La Figura 51 presenta los resultados obtenidos durante el entrenamiento del modelo.

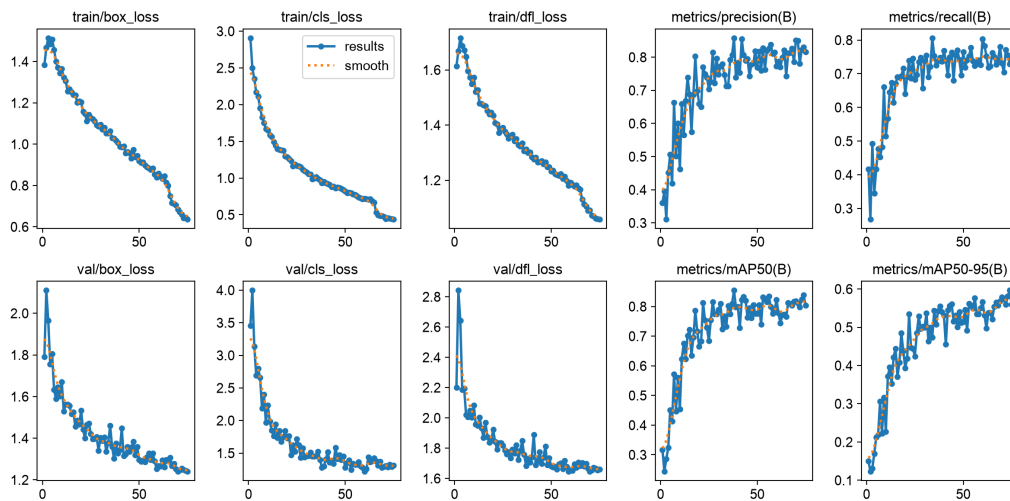


Figura 51. Resultados del entrenamiento del modelo YOLOv8.

Fuente: Autoría propia.

La Figura 51 muestra la evolución de las principales métricas obtenidas durante el entrenamiento del modelo YOLOv8. Se observa que las funciones de pérdida (Box Loss, Classification Loss y Distribution Focal Loss) disminuyen progresivamente tanto en entrenamiento como en validación, indicando una adecuada convergencia del modelo. De igual manera, las métricas de Precisión, Recall y mAP presentan una tendencia creciente conforme avanzan las

épocas de entrenamiento, evidenciando una mejora continua en la capacidad del modelo para detectar las anomalías definidas en el dataset.

Cuadro 8. Resultados generales del entrenamiento del modelo

Parámetro	Valor
Épocas	75
Precisión	82 %
Recall	74 %
mAP50	84 %
Tiempo de entrenamiento	7000 min
Número de imágenes	1694

Los resultados obtenidos evidencian que el modelo alcanzó un desempeño adecuado para la detección de personas y anomalías dentro del entorno monitoreado, permitiendo posteriormente su implementación en tiempo real mediante el sistema embarcado con el dron.

3.1.2. Resultados de validación del modelo YOLOv8

Después de completar el entrenamiento, se revisó el comportamiento del modelo con las imágenes destinadas a validación. Para ello, se analizaron la curva F1-Confidence, la curva Precision-Recall y la matriz de confusión normalizada. Estas gráficas permitieron conocer el rendimiento final del modelo y observar las diferencias existentes entre las clases Caída, comida, dormir y teléfono.

Curva F1-Confidence: La Figura 52 presenta la variación del valor F1 con respecto al umbral de confianza. Esta métrica considera al mismo tiempo la precisión y el recall, por lo que resulta útil para identificar un punto de funcionamiento en el que ambos valores mantengan un equilibrio.

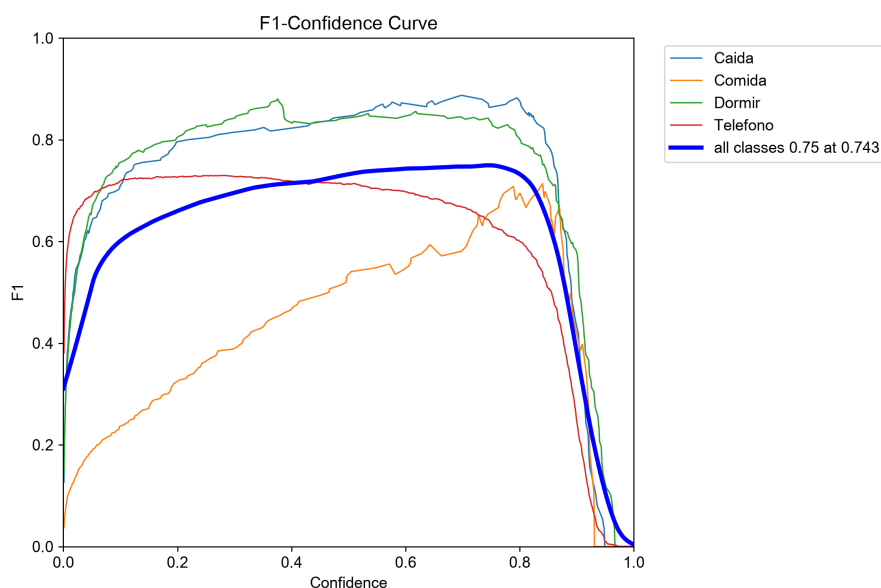


Figura 52. Curva F1 en función del umbral de confianza

Fuente: Autoría propia.

Como se observa en la Figura 52, el valor F1 máximo para el conjunto de clases fue de 0.75, alcanzado con un umbral de confianza cercano a 0.743. En ese punto, el modelo presentó una relación favorable entre las detecciones correctas y las anomalías que podían quedar sin identificar.

A pesar de que la gráfica indica un valor óptimo de 0.743, para las pruebas realizadas con el sistema se utilizó un umbral de 0.65. Se eligió este valor con la intención de tener un equilibrio en las alertas generadas. Al revisar cada clase por separado, la clase caída presentó los valores F1 más altos en buena parte de la curva. Comida y dormir también presentaron un comportamiento estable. En cambio, teléfono obtuvo resultados inferiores, principalmente cuando el umbral de confianza comenzó a aumentar.

Curva Precision-Recall: La Figura 53 muestra la relación entre la precisión y el recall alcanzados por el modelo. Esta gráfica también permite observar el valor AP de cada clase y el mAP@0.5 obtenido de forma general.

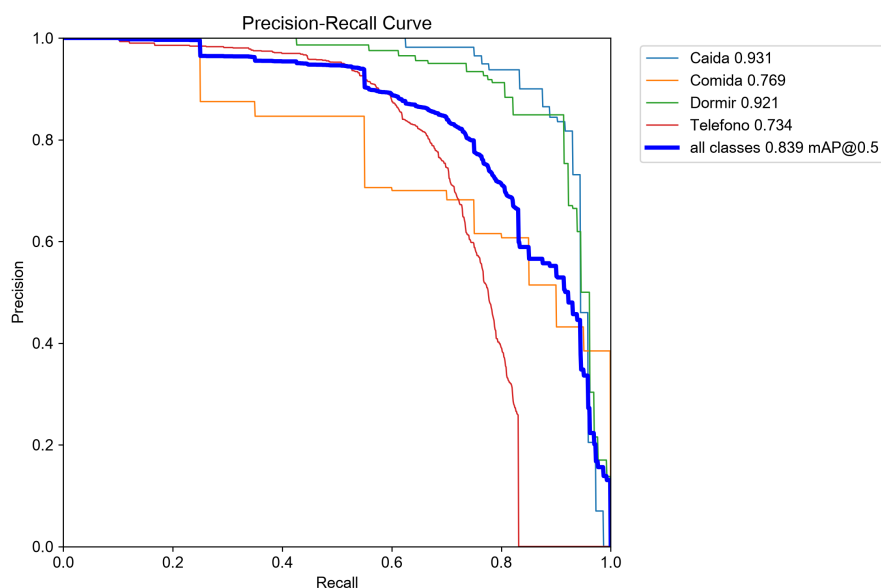


Figura 53. Curva Precision-Recall del modelo YOLOv8

Fuente: Autoría propia.

El modelo alcanzó un mAP@0.5 global de 0.839. Este resultado muestra que, en términos generales, logró reconocer de manera adecuada las anomalías incluidas en el conjunto de validación.

La clase con el valor AP más alto fue caída con 0.931. Después se ubicaron dormir con 0.921 y comida, con 0.769. Por último, teléfono obtuvo un valor de 0.734, siendo la clase con menor rendimiento entre las cuatro evaluadas.

El resultado de teléfono y comida puede estar relacionado con la variedad de posiciones que adopta una persona al realizar esta actividad. Además, ciertos movimientos de las manos y algunos objetos presentes en la escena pueden parecerse visualmente a otras acciones, como permanecer sentado o caída hasta incluso parecer una persona durmiendo.

Matriz de confusión normalizada: Para revisar con mayor detalle los aciertos y errores de clasificación, se utilizó la matriz de confusión normalizada presentada en la Figura 54. En ella se muestran las clases reales en el eje horizontal y las predicciones realizadas por el modelo en el eje vertical.



Figura 54. Matriz de confusión normalizada del modelo YOLOv8

Fuente: Autoría propia.

Los valores de la diagonal principal corresponden a las muestras clasificadas correctamente. Caída alcanzó un valor de 0.94, comida obtuvo 1.00, dormir presentó 0.91 y teléfono registró 0.67. Esto indica que la mayor parte de las imágenes de validación fueron asignadas a la clase correspondiente.

Fuera de la diagonal también se observan algunos errores entre categorías. Por ejemplo, el 0.03 de las muestras reales de comida fue clasificado como dormir. Asimismo, una pequeña parte de las muestras de dormir fue identificada como caída, con valores de 0.01 y el otro porcentaje fue identificada como fondo real.

La mayor dificultad apareció en la clase teléfono respecto al fondo de la imagen. La matriz presenta un valor de 0.67 en la relación entre el fondo real y la predicción teléfono. Esto significa que algunos elementos que no pertenecían a una anomalía fueron interpretados por el modelo como una persona utilizando el teléfono.

Este comportamiento puede producirse cuando aparecen manos cerca del rostro, objetos pequeños sostenidos por una persona o posturas parecidas a las utilizadas durante el manejo de un dispositivo móvil. El umbral de confianza de 0.65 empleado en el programa permitió reducir parte de estas detecciones antes de generar una alerta.

Resumen de los resultados de validación: En la Tabla 9 se reúnen los valores más importantes obtenidos durante la validación del modelo.

Cuadro 9. Métricas de validación del modelo YOLOv8

Métrica	Resultado
F1 global máximo	0.75
Umbral asociado al F1 máximo	0.743
Umbral utilizado en el sistema	0.65
mAP@0.5 global	0.839
AP Caída	0.943
AP Comida	0.769
AP Dormir	0.938
AP Teléfono	0.734
Clasificación correcta de caída	0.94
Clasificación correcta de comida	0.95
Clasificación correcta de dormir	0.91
Clasificación correcta de teléfono	0.67

En términos generales, el modelo mostró un desempeño satisfactorio para su integración en el sistema de monitoreo basado en el DJI Tello. Las clases caída y dormir exhibieron un comportamiento más sólido durante la etapa de validación, en tanto que la clase teléfono resultó más problemática, sobre todo debido a su confusión con elementos presentes en el fondo de la imagen.

Si bien los resultados alcanzados fueron aceptables para el desarrollo del prototipo, la confusión entre la clase teléfono y el fondo constituye una limitación a tener en cuenta durante su operación en tiempo real. Debido a esto, el programa exige que la confianza sea igual o mayor a 0.65 antes de desplegar una detección en la interfaz o de emitir una alerta vía WhatsApp.

3.1.3. Resultados de detección de anomalías

Las pruebas de detección fueron realizadas en tiempo real utilizando el dron DJI Tello como sistema de adquisición de video. El sistema logró identificar correctamente diferentes comportamientos considerados anómalos dentro del área supervisada.

La detección de anomalías se realizó mediante la utilización de regiones de interés (ROI), permitiendo identificar eventos relevantes cuando una persona ingresaba a zonas restringidas o realizaba acciones consideradas fuera del comportamiento esperado.

La Figura 55 muestra el funcionamiento del sistema de detección de anomalías utilizando regiones de interés.



Figura 55. Detección de anomalías mediante regiones de interés (ROI).

Fuente: Autoría propia.

Cuadro 10. Resultados de detección por tipo de anomalía

Anomalía	Tasa de detección	Confianza promedio	Resultado
Uso de celular	85 %	69 %	Correcto
Persona dormida	80 %	83 %	Correcto
Persona comiendo	90 %	73 %	Correcto
Caída de persona	90 %	88 %	Correcto

Los resultados experimentales muestran que el sistema presentó un mejor desempeño en la detección de personas dormidas y en caídas de personas

obteniendo altos niveles de precisión durante las pruebas realizadas. Algunas anomalías relacionadas con posturas corporales como personas usando el teléfono celular y personas consumiendo alimentos presentaron ligeras variaciones debido a cambios de iluminación y perspectiva de la cámara.

3.1.4. Resultados del sistema de alertas mediante WhatsApp

Durante las pruebas realizadas se comprobó el funcionamiento del sistema de alertas por WhatsApp. Cada vez que el modelo detectaba una anomalía con una confianza igual o superior al 65 %, el programa preparaba el mensaje y lo enviaba al número previamente configurado mediante CallMeBot.

El mensaje recibido mostraba la clase identificada, el valor de confianza y la fecha y hora de la detección. Con esta información fue posible conocer el tipo de anomalía registrada sin tener que observar directamente la interfaz del programa.

En la Figura [56](#) se presenta uno de los mensajes recibidos durante las pruebas. En este caso, la notificación fue generada a partir de una anomalía detectada por el modelo YOLOv8.

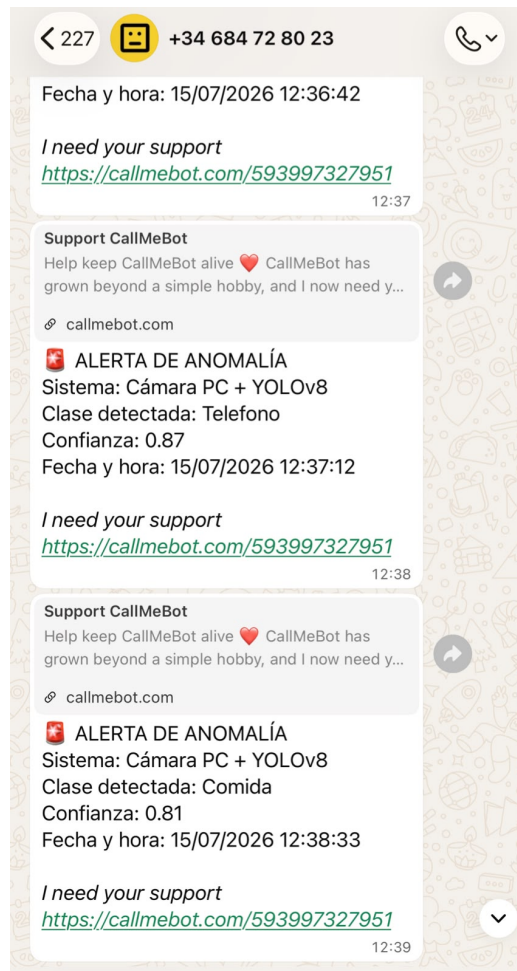


Figura 56. Mensaje de alerta recibido mediante WhatsApp

Fuente: Autoría propia.

Como se observa en la Figura 56, el mensaje enviado contiene los datos principales de la detección. La clase, la confianza y la hora registrada correspondieron con la información mostrada por el programa durante la prueba.

También se revisó el intervalo de treinta segundos colocado entre cada alerta. Esta configuración evitó que el sistema enviara varios mensajes seguidos cuando una misma anomalía aparecía durante varios fotogramas.

Mientras no se cumplía el tiempo establecido, el programa continuaba realizando las detecciones, pero no generaba una nueva notificación. Después de transcurrir los treinta segundos, el envío quedaba habilitado nuevamente.

El proceso de envío no interrumpió la visualización del video ni el funcionamiento de la interfaz. Esto se debe a que la comunicación con WhatsApp se ejecutó en un hilo separado, mientras el programa continuaba procesando

las imágenes obtenidas por la cámara del DJI Tello.

Además del mensaje, el sistema guardó automáticamente una captura de la anomalía en la carpeta `capturas_anomalias`. El archivo se almacenó con el nombre de la clase, el valor de confianza y la fecha de la detección, lo que facilitó su búsqueda posterior.

Con estas pruebas se verificó que el sistema podía informar de manera remota sobre las anomalías detectadas. Sin embargo, el envío de las alertas dependió de la conexión a Internet y de la disponibilidad del servicio CallMeBot. En esta implementación se enviaron únicamente los datos de la detección por WhatsApp, mientras que la captura quedó guardada en la computadora.

3.1.5. Resultados del sistema de monitoreo

La interfaz gráfica desarrollada permitió visualizar en tiempo real el flujo de video capturado por el dron, así como información relacionada con el estado general del sistema y las detecciones realizadas por el modelo de inteligencia artificial.

El sistema de monitoreo incluyó funciones como visualización de video en tiempo real, telemetría del dron, detección de anomalías, generación de alertas visuales y modos de operación automático y manual.

La Figura [57](#) presenta la interfaz implementada y utilizada durante las pruebas de monitoreo.

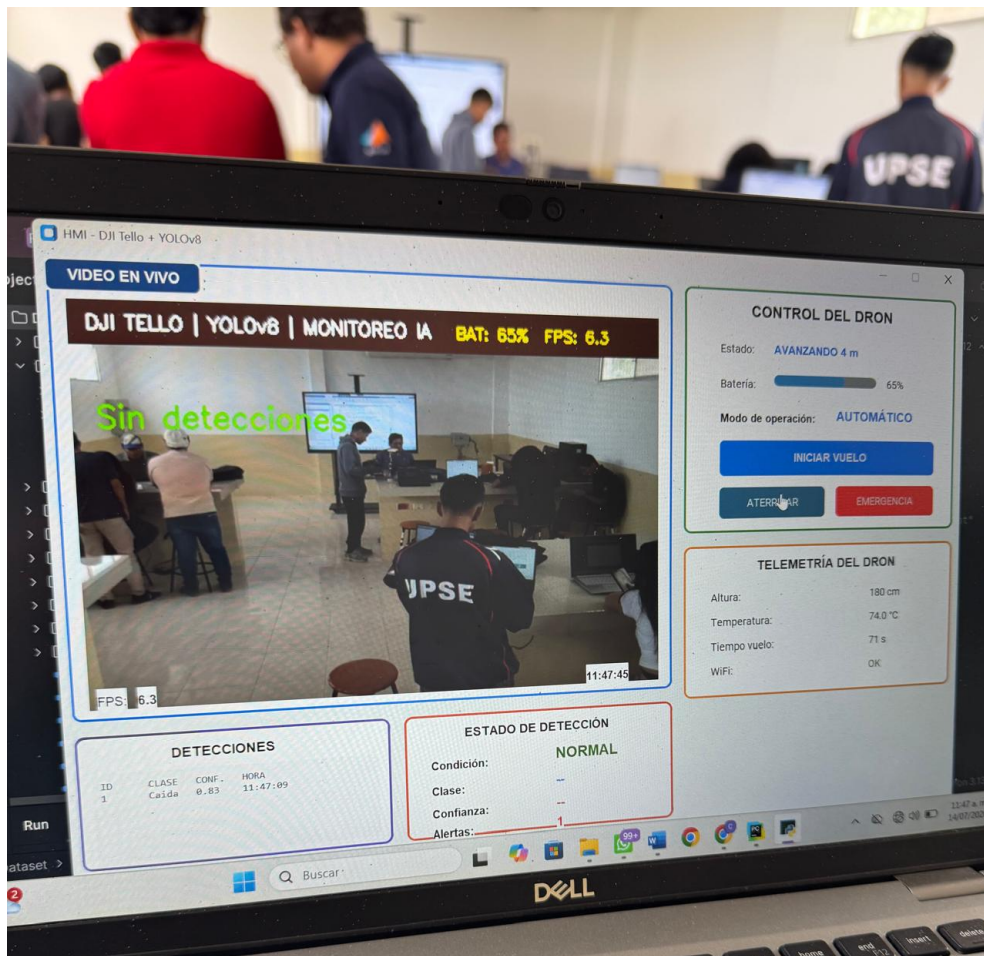


Figura 57. Interfaz gráfica del sistema durante el monitoreo en tiempo real.

Fuente: Autoría propia.

Cuadro 11. Funciones implementadas en la interfaz gráfica

Función	Estado
Video en tiempo real	Implementado
Modo automático	Implementado
Telemetría del dron	Implementado
Detección con YOLOv8	Implementado
Alertas visuales	Implementado
Sistema ROI	Implementado
Comunicación Wi-Fi	Implementado

La interfaz desarrollada permitió supervisar el entorno monitoreado de manera continua, mostrando información relacionada con batería, altura, temperatura, FPS, estado de conexión y alertas generadas por el sistema de visión artificial.

3.1.6. Resultados de las pruebas de vuelo

Las pruebas de vuelo fueron realizadas en entornos interiores controlados, permitiendo evaluar la estabilidad del dron, la calidad del flujo de video y el desempeño general del sistema durante la supervisión aérea.

Durante las pruebas, el dron logró mantener una conexión estable mediante comunicación Wi-Fi, permitiendo transmitir video en tiempo real hacia la estación de procesamiento.

La Figura 58 presenta una de las pruebas realizadas durante el funcionamiento experimental del sistema.



Figura 58. Pruebas experimentales realizadas con el dron DJI Tello.

Fuente: Autoría propia.

Cuadro 12. Resultados de pruebas de vuelo

Parámetro	Resultado
Altura promedio	180 cm
Tiempo de vuelo	5 min
Velocidad de respuesta	0.01 s
Conexión Wi-Fi	2.4 GHz
FPS promedio	6.3 Fps
Tiempo de detección	5 s

Los resultados muestran la efectividad del sistema en tiempo real, que es capaz de ejecutar tareas de seguimiento en el aire e identificar automáticamente anomalías utilizando visión artificial.

3.1.7. Evaluación general del sistema

Finalmente, se realizó una evaluación general del desempeño del sistema considerando la detección de anomalías, estabilidad del vuelo, procesamiento en tiempo real y funcionamiento general del sistema de monitoreo.

Cuadro 13. Evaluación general del sistema

Criterio	Resultado
Detección en tiempo real	Correcta
Generación de alertas	Correcta
Monitoreo remoto	Correcto
Estabilidad del vuelo	Adecuada
Comunicación Wi-Fi	Estable
Procesamiento en tiempo real	Correcto
Funcionamiento del HMI	Correcto

De manera general, el sistema desarrollado logró cumplir con los objetivos planteados, permitiendo implementar un sistema de monitoreo aéreo basado en visión artificial capaz de detectar anomalías en tiempo real mediante el uso de un dron y técnicas de inteligencia artificial.

3.2. Conclusiones y recomendaciones

3.2.1. Conclusiones

1. Se implementó un sistema de supervisión basado en el dron DJI Tello, así como Python, OpenCV, DJITelloPy y YOLOv8n, con procesamiento de video y detección de anomalías en un entorno interior controlado.
2. El modelo entrenado produce predicciones para las clases caída, comida, dormir y teléfono. Las métricas globales fueron del 82 % de exactitud, del 74 % de *recall* y del 84 % de mAP@0.5.
3. La conexión Wi-Fi se utilizaba para enviar comandos y video en tiempo real. Sin embargo, la duración y el alcance de las pruebas se vieron limitados por la duración de la batería y la fiabilidad del enlace.
4. Se implementó la interfaz hombre-máquina que incluía el video procesado, estado de conexión, estado de batería y alertas, esta mejoraba la capacidad de seguimiento y la interpretación de las detecciones por parte del operador.
5. Se logró integrar la supervisión asistida como un prototipo funcional, incorporando los componentes necesarios para el monitoreo y análisis del sistema. Además, se realizaron las pruebas correspondientes en diferentes escenarios, considerando métricas de desempeño y validación de los resultados obtenidos. La documentación de los datos y la formalización de los protocolos de seguridad complementan la propuesta, demostrando la viabilidad del sistema desarrollado y su potencial para ser aplicado en condiciones reales.

3.2.2. Recomendaciones

1. Aumentar el conjunto de datos utilizado para entrenar el modelo con un mayor número de imágenes reales en diferentes condiciones de iluminación, ángulos de cámara, distancias y escenarios para mejorar la precisión y la capacidad de generalización del sistema.
2. Un dron más estable, que aguante más tiempo en el aire y que tenga más sensores, ya que el DJI Tello tiene una duración de vuelo corta y no está ideado para vigilar durante varias horas en entornos industriales o de mayor tamaño.

3. Incorporar un sistema de evasión de obstáculos mediante sensores de proximidad, visión artificial o técnicas de navegación autónoma para optimizar la seguridad durante el vuelo en entornos interiores.
4. Optimizar el procesador de visión artificial o bien a través de un hardware más potente o de un modelo más liviano para aumentar la velocidad y disminuir la latencia en la detección en tiempo real.
5. Realizar pruebas experimentales en entornos más completos y más variables, tanto en el número de rutas aéreas y de personas implicadas, como en la complejidad de las situaciones que se pueden dar, para conocer cuál es el funcionamiento del sistema en un entorno más cercano al real.

Bibliografía

Referencias

- [1] Y. Abbas y A. Jalal, “Drone-Based Human Action Recognition for Surveillance: A Multi-Feature Approach,” en *2024 International Conference on Engineering & Computing Technologies (ICECT)*, pp. 1–6, IEEE, 2024. [En línea]. Disponible en: <https://doi.org/10.1109/ICECT61618.2024.10581378>
- [2] Y. Khosiawan e I. E. Nielsen, “A System of UAV Application in Indoor Environment,” *Production & Manufacturing Research*, vol. 4, no. 1, pp. 2–22, 2016. [En línea]. Disponible en: <https://doi.org/10.1080/21693277.2016.1195304>
- [3] Grand View Research, “Commercial Drones Market Size, Share & Trends Analysis Report, 2025–2030,” *Grand View Research*, 2024. [En línea]. Disponible en: <https://www.grandviewresearch.com/industry-analysis/global-commercial-drones-market>
- [4] Delvens, “Indoor Drones Market Size, Share & Trends Analysis Report,” *Delvens*, 2024. [En línea]. Disponible en: <https://www.delvens.com/report/indoor-drones-market>
- [5] K. Joshi, G. Vidya, S. Shaw, A. K. Thyagarajan, A. Pathak, R. H. Shah, V. R. Kishan y J. S. R. Alex, “Anomaly Detection in Drone-Captured Images Using Machine Learning Techniques and Deep Learning Architectures,” en *Futuristic Communication and Network Technologies*, pp. 783–791, Springer, 2022. [En línea]. Disponible en: https://doi.org/10.1007/978-981-16-4625-6_78
- [6] V. Sindhvani, H. Sidahmed, K. Choromanski y B. Jones, “Unsupervised Anomaly Detection for Self-Flying Delivery Drones,” en *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 186–192, IEEE, 2020. [En línea]. Disponible en: <https://doi.org/10.1109/ICRA40945.2020.9197074>
- [7] J. Valdovinos Miranda, “Low-Cost UAV Swarm for Real-Time Object Detection Applications,” tesis de maestría, California Polytechnic State University, 2022. [En línea]. Disponible en: <https://doi.org/10.15368/theses.2022.51>

- [8] C. L. Calin, “Aplicación de los drones en los procesos de inspección y gestión de la seguridad en buques,” trabajo de fin de grado, Universitat Politècnica de Catalunya, 2023. [En línea]. Disponible en: <https://hdl.handle.net/2117/382633>
- [9] A. F. Silva Bohórquez, C. A. Peña Cortés, y L. E. Mendoza, “Sistema de inspección y vigilancia utilizando un robot aéreo guiado mediante visión artificial,” *Iteckne*, vol. 10, no. 2, pp. 190–198, dic. 2013.
- [10] Javi, “RPA, RPAS, UAS y UAV: ¿Qué son y en qué se diferencian?”, *UMILES*, [En línea]. Disponible en: <https://umilesgroup.com/rpas-uas-uav-diferencias/>.
- [11] K. P. Valavanis y G. J. Vachtsevanos, eds., *Handbook of Unmanned Aerial Vehicles*. Springer, 2015. [En línea]. Disponible en: <https://doi.org/10.1007/978-90-481-9707-1>
- [12] R. W. Beard y T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.
- [13] A. Sánchez, “¿Cuáles son las partes de un Dron? Listado completo”, *UMILES*, [En línea]. Disponible en: <https://umilesgroup.com/partes-de-un-dron/>.
- [14] Amazon, *TBS Source One V3 Frame*. Disponible en: <https://www.amazon.com/TBS-Source-One-V3/dp/B07TT8B8NV>.
- [15] Tyto Robotics, *¿Qué es un ESC y cómo funciona?*. Disponible en: <https://www.tytorobotics.com/blog/what-is-an-esc-and-how-it-works/>.
- [16] Prometec, *Controladores de vuelo para drones: ¿Cuál elegir?*. Disponible en: <https://www.prometec.net/controladores-de-vuelo-para-drones/>.
- [17] Dro-bots, *Tipos de baterías LiPo para drones y recomendaciones*. Disponible en: <https://dro-bots.com/blogs/guias/baterias-lipo-para-drones>.
- [18] A. Townsend, I. N. Jiya, C. Martinson, D. Bessarabov y R. Gouws, “A Comprehensive Review of Energy Sources for Unmanned Aerial Vehicles, Their Shortfalls and Opportunities for Improvements,” *Heliyon*, vol. 6, no. 11, art. e05285, 2020. [En línea]. Disponible en: <https://doi.org/10.1016/j.heliyon.2020.e05285>

- [19] Decero a Dronero, *¿Qué es el receptor del dron y cómo funciona?*. Disponible en: <https://deceroadronero.es/componentes-que-tienes-que-saber-de-tu-dron/receptor-dron-que-es/>.
- [20] Prometec, *Cómo elegir el VTX para tu dron FPV*. Disponible en: <https://www.prometec.net/elegir-vtx-para-dron/>.
- [21] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Springer, 2022. [En línea]. Disponible en: <https://doi.org/10.1007/978-3-030-34372-9>
- [22] MathWorks. (s.f.). *Procesamiento digital de imágenes*. Recuperado de: <https://la.mathworks.com/discovery/digital-image-processing.html>
- [23] Foqum. (s.f.). *¿Qué es el procesamiento de imágenes?*. Recuperado de: <https://foqum.io/blog/termino/procesamiento-de-imagenes/>
- [24] G. Jocher, A. Chaurasia y J. Qiu, *Ultralytics YOLOv8*, software informático, 2023. [En línea]. Disponible en: <https://github.com/ultralytics/ultralytics>
- [25] Y. LeCun, Y. Bengio y G. Hinton, “Deep Learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [En línea]. Disponible en: <https://doi.org/10.1038/nature14539>
- [26] García Serrano, Á. (2022). *Procesamiento digital de imágenes*. Trabajo de Integración Curricular, Universidad de Sevilla. Disponible en: https://biblus.us.es/bibing/proyectos/abreproy/12112/fichero/Documento_por_capitulos%252F3_Cap%C3%ADtulo_3.pdf
- [27] ILCE. (s.f.). *Procesamiento de imágenes digitales*. Biblioteca Digital ILCE. Recuperado de: https://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen2/ciencia3/084/htm/sec_9.htm
- [28] H. Par, “Crear una red neuronal convolucional en Python desde cero,” *Blog de Hadson Par*, 2021. [En línea]. Disponible en: <http://blog.hadsonpar.com/2021/08/crear-una-red-neuronal-convolucional-en.html>
- [29] G. Wang, J. Ai, L. Mo, X. Yi, P. Wu, X. Wu y L. Kong, “Anomaly Detection for Data from Unmanned Systems via Improved Graph Neural Networks with Attention Mechanism,” *Drones*, vol. 7, no. 5, art. 326, 2023. [En línea]. Disponible en: <https://doi.org/10.3390/drones7050326>

- [30] C. Kyrkou, T. Theocharides y C. Panayiotou, “Deep learning-based anomaly detection for individual drone vehicles performing swarm missions,” *Expert Systems with Applications*, vol. 242, p. 122717, 2024. Disponible en: <https://doi.org/10.1016/j.eswa.2023.122717>
- [31] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” en *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, IEEE, 2016. [En línea]. Disponible en: <https://doi.org/10.1109/CVPR.2016.91>
- [32] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7263–7271, 2017. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Redmon_YOLO9000_Better_Faster_CVPR_2017_paper.html
- [33] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv preprint arXiv:1804.02767*, 2018. [Online]. Available: <https://arxiv.org/abs/1804.02767>
- [34] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” *arXiv preprint arXiv:2004.10934*, 2020. [Online]. Available: <https://arxiv.org/abs/2004.10934>
- [35] Ultralytics, “YOLOv5,” *GitHub repository*, 2020. [Online]. Available: <https://github.com/ultralytics/yolov5>
- [36] C. Li et al., “YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications,” *arXiv preprint arXiv:2209.02976*, 2022. [Online]. Available: <https://arxiv.org/abs/2209.02976>
- [37] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors,” *arXiv preprint arXiv:2207.02696*, 2022. [Online]. Available: <https://arxiv.org/abs/2207.02696>
- [38] Ultralytics, “YOLOv8 Docs,” Ultralytics Official Documentation, 2024. [En línea]. Disponible en: <https://docs.ultralytics.com/>
- [39] G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics YOLO,” *GitHub repository*, 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>

- [40] V7 Labs, “YOLO (You Only Look Once) Explained,” V7 Labs, 2024. [En línea]. Disponible en: <https://www.v7labs.com/blog/yolo-object-detection>
- [41] Encord, “YOLO Object Detection Explained: Evolution, Models, and Comparison,” Encord Blog, 2024. [En línea]. Disponible en: <https://encord.com/blog/yolo-object-detection-guide/>
- [42] DataCamp, “Object Detection with YOLO: The Complete Guide,” DataCamp, 2024. [En línea]. Disponible en: <https://www.datacamp.com/tutorial/yolo-object-detection-guide>
- [43] M. Ball, “FIXAR Develops Autonomous Hybrid VTOL and Indoor Drones,” *Unmanned Systems Technology*, 2021. Disponible en: <https://fixar.pro/products/fixar-indoor/>
- [44] Robotnik Automation, “Human-Machine Interface in robotics: definition, benefits and future innovations,” 2024. [En línea]. Disponible en: <https://robotnik.eu/human-machine-interface-in-robotics-definition-benefits-future-innovations/>
- [45] Python Software Foundation, “Python 3 Documentation.” [En línea]. Disponible en: <https://docs.python.org/3/>
- [46] G. Bradski, “The OpenCV Library,” *Dr. Dobbs’s Journal of Software Tools*, 2000.
- [47] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith et al., “Array Programming with NumPy,” *Nature*, vol. 585, pp. 357–362, 2020. [En línea]. Disponible en: <https://doi.org/10.1038/s41586-020-2649-2>
- [48] MathWorks, “Productos MATLAB,” [En línea]. Disponible: <https://la.mathworks.com/products/matlab.html>
- [49] Profile, “¿Qué es Scratch?” [En línea]. Disponible: <https://profile.es/blog/que-es-scratch/> [
- [50] Campus Norte UNC, “Aprende a crear tu primer videojuego con Scratch,” [En línea]. Disponible: <https://campusnorte.unc.edu.ar/diplomaturas/aprende-a-crear-tu-primer-videojuego-con-scratch/>

- [51] «A6.pdf». Accedido: 23 de junio de 2024. [En línea]. Disponible en: <http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/861/A6.pdf?sequence=6>
- [52] R. G. Méndez y C. H. Arias, «Metodología Para el Análisis de Redes en Instituciones Públicas Basada en Prácticas ITIL v3», 2020.
- [53] B. A. P. Anaguano, «INTEGRANTES: LIZARDO ALEXIS LUCERO MOLINA».
- [54] J. C. P. Domínguez, «TECNOLOGÍA BLUETOOTH», Tesis para obtener el grado de Ingeniero en Comunicaciones y Electrónica. Instituto Politécnico Nacional, Accedido: 26 de marzo de 2025. Disponible en: https://www.academia.edu/31776999/_TECNOLOG%C3%8DA_BLUETOOTH_Tesis_para_obtener_el_grado_de_Ingeniero_en_Comunicaciones_y_Electr%C3%B3nica_
- [55] Wi-Fi Alliance, “Discover Wi-Fi.” [En línea]. Disponible en: <https://www.wi-fi.org/discover-wi-fi>
- [56] Ryze Tech, *Tello SDK 2.0 User Guide*, 2018. [En línea]. Disponible en: <https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf>
- [57] Southeast Missouri State University, “Indoor Drone Safety Guidelines,” 2024. [En línea]. Disponible en: <https://semo.edu/drone-compliance/indoor-safety.html>
- [58] Secretaría de Comunicaciones y Transportes, “Norma Oficial Mexicana NOM-107-SCT3-2019, que establece los requerimientos para operar un sistema de aeronave pilotada a distancia (RPAS) en el espacio aéreo mexicano,” *Diario Oficial de la Federación*, 2019. [En línea]. Disponible en: https://www.dof.gob.mx/normasOficiales/8006/sct11_C/sct11_C.html
- [59] Comisión Europea, “Reglamento Delegado (UE) 2019/945 de la Comisión, de 12 de marzo de 2019, sobre los sistemas de aeronaves no tripuladas y los operadores de terceros países,” 2019. [En línea]. Disponible en: https://eur-lex.europa.eu/eli/reg_del/2019/945/oj
- [60] Comisión Europea, “Reglamento de Ejecución (UE) 2019/947 de la Comisión, de 24 de mayo de 2019, relativo a las normas y los procedimientos aplicables a la utilización de aeronaves no tripuladas,”

2019. [En línea]. Disponible en: https://eur-lex.europa.eu/eli/reg_impl/2019/947/oj
- [61] Administración Nacional de Aviación Civil, “Resolución 550/2025: Nuevo marco normativo para la operación de drones,” 2025. [En línea]. Disponible en: <https://www.argentina.gob.ar/normativa/nacional/resoluci%C3%B3n-550-2025-416149/texto>
- [62] Dirección General de Aviación Civil, “RDAC 101: Sistemas de aeronaves no tripuladas (UAS),” Ecuador, 2024. [En línea]. Disponible en: <https://www.aviacioncivil.gob.ec/wp-content/uploads/downloads/2024/12/RDAC-101-UAS.pdf>
- [63] J. W. Creswell y J. D. Creswell, *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, 5th ed. SAGE, 2018.
- [64] D. C. Montgomery, *Design and Analysis of Experiments*, 9th ed. Wiley, 2017.
- [65] R. K. Yin, *Case Study Research and Applications: Design and Methods*, 6th ed. SAGE, 2018.
- [66] JetBrains, “PyCharm Features”, 2026. [En línea]. Disponible en: <https://www.jetbrains.com/pycharm/features/>.
- [67] Microsoft, “Python in Visual Studio Code”, 2026. [En línea]. Disponible en: <https://code.visualstudio.com/docs/languages/python>.
- [68] Project Jupyter, “Jupyter Notebook”, 2026. [En línea]. Disponible en: <https://jupyter.org/>.
- [69] Spyder Project Contributors, “Spyder Documentation”, 2026. [En línea]. Disponible en: <https://docs.spyder-ide.org/current/>.
- [70] Ryze Tech, “Tello User Manual,” DJI, 2018. [En línea]. Disponible en: <https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20User%20Manual%20v1.4.pdf>
- [71] CallMeBot, “Free API to Send WhatsApp Messages”, 2021. [En línea]. Disponible en: <https://www.callmebot.com/blog/free-api-whatsapp-messages/>.
- [72] A. Vicianá Gálvez, “Implementación de un sistema de navegación autónomo basado en visión por computador aplicado a dron Tello,” sep. 2022. [En línea]. Disponible en: <https://hdl.handle.net/10953.1/19827>.

- [73] MundoDrone, “DJI Tello: análisis, especificaciones y experiencia de uso,” [En línea]. Disponible en: <https://mundodrone.es/dji-tello/>.
- [74] MercadoLibre, “Drone Dji Tello Boost Combo Dji020 Mini Cámara Hd Con 3 Baterías,” [En línea]. Disponible en: <https://www.mercadolibre.com.ec/drone-dji-tello-boost-combo-dji020-mini-camara-hd-con-3-baterias/p/MEC24030630>.
- [75] Ryze Tech, «Tello User Manual V1.0_ES.pdf», Accedido: 15 de marzo de 2025. [En línea]. Disponible en: https://dl-cdn.ryzerobotics.com/downloads/Tello/201806mul/Tello%20User%20Manual%20V1.0_ES.pdf.
- [76] AriesRC, “DJI Ryze Tello Drones Motor Engine Repair Parts,” [En línea]. Disponible en: <https://www.ariesrc.gr/en/motors/19530-dji-ryze-tello-drones-motor-engine-repair-parts-4pcs-1set.html>.
- [77] Universidad de Jaén, «Diseño de un sistema de navegación autónoma mediante visión artificial aplicado a un dron cuadróptero DJI Tello», Trabajo Fin de Grado, 2021. [En línea]. Disponible en: <https://crea.ujaen.es/server/api/core/bitstreams/843c1e27-c7f2-49d0-976a-aa0662f411fa/content>.
- [78] JetBrains. (2024). *PyCharm: IDE para desarrollo en Python*. Disponible en: <https://www.jetbrains.com/pycharm/>
- [79] DJITelloPy. (2025). *Documentación oficial de la biblioteca djitelopy para Python*. Disponible en: <https://github.com/damiafuentes/DJITelloPy>
- [80] Alberca, A. S. (2025). *Manual de Numpy en Python*. Aprende con Alf. Disponible en: <https://aprendeconalf.es/docencia/python/manual/numpy/>
- [81] OpenCV. (2025). *Documentación oficial de OpenCV: Referencia de funciones*. Disponible en: <https://docs.opencv.org/4.x/>

Anexos

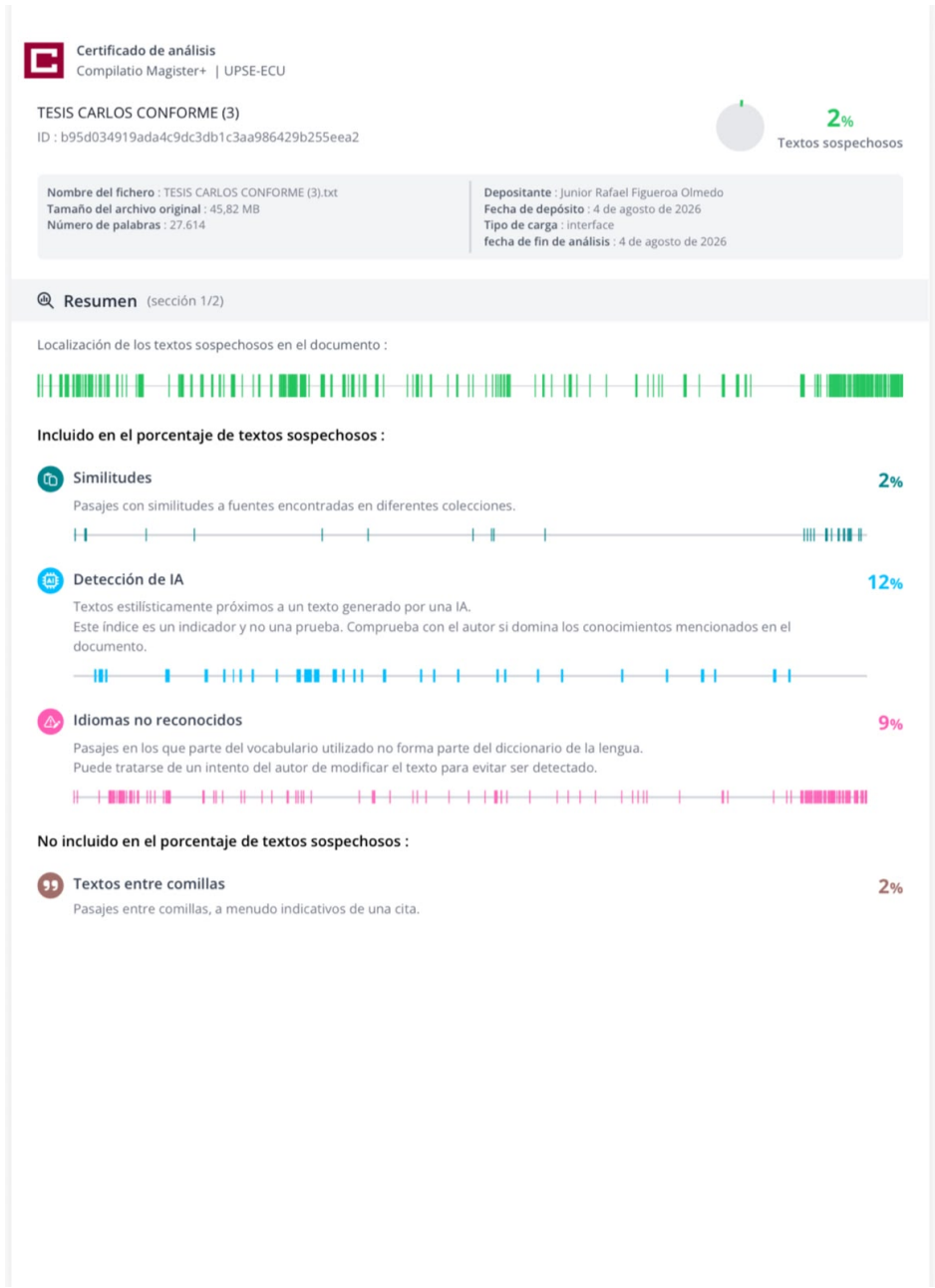


Figura 59. Reporte de compilatio