UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERÍA
CARRERA DE TELECOMUNICACIONES

Sistema de prevención de delitos contra vehículos estacionados aplicando procesamiento de imágenes.

Proyecto de investigación previo a la obtención del título de

Ingeniero en Telecomunicaciones

Autor:
Moncayo Romero Oscar Alexander

Tutor:
PhD. Daniel Antonio Santillán Haro

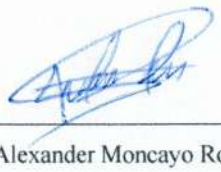
Riobamba, Ecuador. 2026

DECLARATORIA DE AUTORÍA

Yo, Oscar Alexander Moncayo Romero, con cédula de ciudadanía 210061642-0, autor del trabajo de investigación titulado: **Sistema de prevención de delitos contra vehículos estacionados aplicando procesamiento de imágenes**, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor (a) de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, a los 30 días del mes de junio de 2026.



Oscar Alexander Moncayo Romero

C.I: 210061642-0

DICTAMEN FAVORABLE DEL PROFESOR TUTOR

Quien suscribe, Daniel Antonio Santillán Haro catedrático adscrito a la Facultad de Ingeniería, por medio del presente documento certifico haber asesorado y revisado el desarrollo del trabajo de investigación titulado: **Sistema de prevención de delitos contra vehículos estacionados aplicando procesamiento de imágenes,** bajo la autoría de Oscar Alexander Moncayo Romero; por lo que se autoriza ejecutar los trámites legales para su sustentación.

Es todo cuanto informar en honor a la verdad; en Riobamba, a los 26 días del mes de junio de 2026

PhD. Daniel Antonio Santillán Haro

TUTOR

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación **Sistema de prevención de delitos contra vehículos estacionados aplicando procesamiento de imágenes** presentado por **Oscar Alexander Moncayo Romero**, con cédula de identidad número **2100616420**, bajo la tutoría de **PhD. Daniel Antonio Santillán Haro**; certificamos que recomendamos la **APROBACIÓN** de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba a los 30 días del mes de junio de 2026.

Pedro Fernando Escudero Villa, PhD.
PRESIDENTE DEL TRIBUNAL DE GRADO

Alejandra Del Pilar Pozo Jara, Mgs.
MIEMBRO DEL TRIBUNAL DE GRADO

Manuel Antonio Meneses Freire, PhD.
MIEMBRO DEL TRIBUNAL DE GRADO

C E R T I F I C A C I Ó N

Que, **MONCAYO ROMERO OSCAR ALEXANDER** con CC: **2100616420**, estudiante de la Carrera **TELECOMUNICACIONES**, Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado "**SISTEMA DE PREVENCIÓN DE DELITOS CONTRA VEHÍCULOS ESTACIONADOS APLICANDO PROCESAMIENTO DE IMÁGENES.**", cumple con el 1 %, de similitud y el 8% de Inteligencia Artificial, de acuerdo al reporte del sistema Anti plagio **COMPILATIO**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 17 de junio de 2026

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
DANIEL ANTONIO
SANTILLAN HARO

PhD. Daniel Antonio Santillán Haro
TUTOR

DEDICATORIA

A Dios, por ser el cimiento de mi vida, la fuente de toda sabiduría y entendimiento, y la guía constante que me ha brindado la fortaleza necesaria para superar cada adversidad en este camino.

A mis amados padres, Oscar y Guadalupe, cuyo amor, sacrificio y apoyo incondicional han sido el pilar fundamental que me permitió culminar mis estudios universitarios. Gracias por inculcarme valores inquebrantables, principios de vida y, sobre todo, por enseñarme a caminar con fe en el Señor.

Con profundo cariño a mis abuelitos, María y Jesús, cuyas bendiciones y sabiduría me han acompañado en cada etapa de mi crecimiento. A mis hermanos, Anthony, Jhohan, Jostin, Gabriel y Saray, por su presencia constante y su respaldo en los momentos que más lo necesité a lo largo de la carrera; espero de corazón que este logro les sirva de inspiración para alcanzar sus propias metas y convertirse en grandes profesionales.

Oscar Moncayo R.

AGRADECIMIENTO

Ante todo, expreso mi gratitud eterna a Dios por concederme la dicha de alcanzar una de mis más grandes metas, siendo mi amparo incondicional y por brindarme fuerzas en los momentos más difíciles. A mis padres, Oscar y Guadalupe, les agradezco con el alma su respaldo infinito; este éxito académico es el resultado directo de su apoyo incondicional, desvelos y sacrificios compartidos para verme triunfar. Asimismo, a mis abuelitos, María y Jesús, por cubrirme siempre con su amor y sus bendiciones. A mis hermanos, Anthony, Jhohan, Jostin, Sarahí y Gabriel, por creer firmemente en mi capacidad y brindarme ese aliento diario para no rendirme, extendiendo este agradecimiento a toda mi familia por estar pendiente de cada uno de mis pasos. Dedico una mención muy especial a mi pareja, Janela, agradeciéndole su infinita paciencia, su presencia constante y el ser ese motor esencial que me dio tranquilidad y empuje en los días más difíciles del proyecto. De igual forma, reconozco la labor de mi tutor de tesis, quien con paciencia y generosidad guio mi aprendizaje, ayudándome a mejorar mis capacidades técnicas. Mi sincero agradecimiento también a la Universidad y al cuerpo docente por brindarme una formación integral y de excelencia. Por último, mi afecto y gratitud para mis amigos de las aulas universitarias, con quienes compartí tantos momentos e historias felices. Su compañía sincera se queda grabada como recuerdos valiosos de esta pequeña etapa de mi vida.

Oscar Moncayo R.

ÍNDICE GENERAL

DECLARATORIA DE AUTORÍA

DICTAMEN FAVORABLE DE TUTOR

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

CERTIFICACIÓN DEL SISTEMA ANTI PLAGIO

DEDICATORIA

AGRADECIMIENTO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

ABSTRACT

CAPÍTULO I **17**

1.1. Introducción 17
1.2. Planteamiento del Problema 18
1.3. Justificación 19
1.4. Objetivos 20
1.4.1. General 20
1.4.2. Específicos 20

CAPÍTULO II **21**

2.1. Estado del Arte 21
2.2. Fundamentación Teórica 23
2.2.1. Análisis estadístico del robo de vehículos 23
2.2.1.1. Métodos de robos a vehículos 26
2.2.1.2. Criminología ambiental 27
2.2.1.3. Criterios de selección del vehículo objetivo 27
2.2.2. Capa 1: Percepción y Hardware Integrado 28
2.2.2.1. Principio de Funcionamiento de la Tecnología Time-of-Flight (ToF) 28
2.2.2.2. Características Técnicas del Sensor VL53L0X 30
2.2.2.3. Protocolo de Comunicación Inter-Integrated Circuit (I2C) 30

2.2.2.4. Direccionamiento Dinámico mediante Pines de Apagado por Hardware (XSHUT) . . . 30
2.2.2.5. Cerebro del procesamiento . . . 31
2.2.3. Capa 2: Inteligencia Artificial y Visión . . . 32
2.2.3.1. Procesamiento de imágenes . . . 32
2.2.3.2. Etapas Procesamiento de imágenes y videos. . . . 33
2.2.3.3. Visión artificial . . . 34
2.2.3.4. Etapas de procesamiento de imágenes con visión artificial 34
2.2.3.5. Algoritmos principales de reconocimiento biométrico . . 35
2.2.3.6. Aprendizaje Automático y Aprendizaje Profundo . . . 37
2.2.3.7. Arquitectura YOLO (You Only Look Once) . . . 39
2.2.3.8. Métricas de Evaluación . . . 41
2.2.3.9. Algoritmos Auxiliares de Procesamiento en Visión Artificial 42
2.2.4. Capa 3: Telecomunicaciones y Alertas . . . 43
2.2.4.1. Arquitectura de Redes Celulares GSM/LTE . . . 43
2.2.4.2. Interfaces UART/USB en Linux . . . 44
2.2.4.3. Protocolo de Comandos AT . . . 45
2.2.4.4. Telegram Bot API . . . 45

CAPÍTULO III **46**
3.1. Metodología . . . 46
3.1.1. Tipo de Investigación . . . 46
3.1.2. Diagrama Operativo . . . 46
3.1.3. Población y Muestra . . . 47
3.1.3.1. Población . . . 47
3.1.3.2. Muestra . . . 47
3.1.4. Operacionalización de las Variables . . . 48
3.1.4.1. Variables Independientes . . . 48
3.1.4.2. Variable Dependiente . . . 48
3.2. Fase 1: Investigación y Análisis . . . 49
3.2.1. Selección de la Plataforma de Procesamiento . . . 49
3.2.2. Selección del Modelo de Detección de Objetos . . . 51
3.2.3. Selección del Sensor de Distancia . . . 52
3.2.4. Selección del Módulo de Comunicación Celular . . . 53
3.3. Fase 2: Diseño del Sistema . . . 54
3.3.1. Descripción General del Sistema . . . 54
3.3.2. Componentes de Hardware . . . 55
3.3.3. Diagrama de Bloques del Sistema . . . 57
3.3.4. Secuencia de Procesos del Sistema . . . 59
3.4. Fase 3: Desarrollo e Implementación . . . 60
3.4.1. Configuración del Hardware . . . 60
3.4.2. Configuración de Montaje . . . 64

3.4.3. Prototipo Instalado 65
3.4.4. Configuración del Software 68
3.4.5. Construcción del Dataset y Entrenamiento del Modelo 68
3.4.6. Arquitectura del Software 70
3.5. Fase 4: Validación del Sistema 72
3.5.1. Prueba de Sensores 72
3.5.2. Prueba de Cámaras 73
3.5.3. Prueba de Detección 74
3.5.4. Prueba de Alertas 75

CAPÍTULO IV **77**
4.1. Resultados y Discusión 77
4.2. Fase 1: Diseño del sistema de vigilancia 77
4.2.1. Entrenamiento del modelo YOLOv8n 77
4.2.2. Curvas de pérdida del entrenamiento 79
4.2.3. Matriz de confusión del modelo 80
4.3. Fase 2: Implementación del sistema 82
4.3.1. Escenarios de prueba 82
4.3.1.1. Matriz de pruebas del sistema 82
4.4. Fase 3: Evaluación del sistema 86
4.4.0.1. Matriz de Confusión 86
4.4.0.2. Análisis de Detección de Atributos YOLOv8n 87
4.4.0.3. Diagrama de Cajas — Tiempo de Respuesta por Escenario 88
4.4.0.4. Diagrama de Dispersión — Atributos Detectados vs Tiempo de Respuesta 89
4.4.1. Discusión 90

CAPÍTULO V **92**
5.1. Conclusiones 92
5.2. Recomendaciones 94

BIBLIOGRAFÍA **95**

ANEXOS **102**

ÍNDICE DE TABLAS

2.1. Etapas de la arquitectura de sistemas electrónicos para el procesamiento de videos enfocada a la visión artificial [1] 34
2.2. Variantes del modelo YOLOv8 y sus características [2, 3]. 41
3.3. Operacionalización de variables 49
3.4. Comparación entre Arduino y Raspberry Pi 50
3.5. Comparación entre modelos de Raspberry Pi 51
3.6. Comparación de modelos de detección de objetos 52
3.7. Comparación de sensores de detección de presencia 53
3.8. Comparación de módulos de comunicación celular 54
3.9. Componentes utilizados en el hardware. 56
3.10. Tabla de conexiones entre componentes y Raspberry Pi 5 64
3.11. Librerías instaladas. 68
3.12. Parámetros principales para el entrenamiento. 69
4.13. Métricas obtenidas del modelo entrenado 78
4.14. Escenarios de prueba 82
4.15. Matriz de pruebas del sistema y resultados de detección. 83
4.16. Matriz de confusión del sistema. 86
4.17. Matriz de confusión de detección de atributos YOLOv8n. 87

ÍNDICE DE FIGURAS

2.1. estudios sobre robo de vehículos 24
2.2. estudios sobre robo de vehículos 24
2.3. Evolución del robo de vehículos en Ecuador. 25
2.4. Distribución de robos de vehículos en Ecuador hasta octubre de 2025 [4]. . 26
2.5. Diagrama de bloques de VL53L0X [5]. 28
2.6. Descripción funcional del sensor [5]. 29
2.7. Precisión COCO mAP vs. latencia de inferencia para distintas arquitecturas YOLO. Adaptado de Ultralytics [6]. 39
2.8. El sistema de detección YOLO [7]. 39
2.9. Arquitectura básica del modelo de detección de objetos YOLOv8 [8]. . . . 40
2.10. Arquitectura de integración GSM/LTE del módulo SIM7600G-H en el sistema. 44
3.11. Diagrama de Proceso de la Metodología de Proyecto. 46
3.12. Diagrama de bloques del sistema propuesto. 57
3.13. Procesos del funcionamiento del sistema propuesto. 59
3.14. Circuito de acondicionamiento del sensor VL53L0X. 61
3.15. Distribución e instalación de los componentes del sistema sobre el vehículo de prueba. 65
3.16. Raspberry Pi 5 instalada en el vehículo. 67
3.17. Módulo SIM7600G-H (4G Dongle). 67
3.18. Cámara y sensor zona delantera. 67
3.19. Cámara y sensor zona lateral. 67
3.20. Sensor VL53L0X zona delantera. 67
3.21. Sensor VL53L0X zona trasera. 67
3.22. Fly Cam (Z3) instalada en luneta trasera. 67
3.23. Componentes del prototipo instalados en el vehículo. 67
3.24. Detalles del dataset en la plataforma Roboflow. 69
3.25. Diagrama de flujo por zona. 71
3.26. Control del funcionamiento correcto de los sensores. 73
3.27. Control de cámaras. 74
3.28. Constancia de los atributos detectados. 75
3.29. Evidencias de las alertas. 76
4.30. Resultado del entrenamiento con YOLOv8n en Google Colab. 77
4.31. Gráficas de las curvas de pérdidas de entrenamiento. 79
4.32. Matriz de confusión del modelo entrenado. 80
4.33. Resultados de las imágenes de detección del modelo entrenado. 81
4.34. Diagrama de cajas del tiempo de respuesta por escenario de prueba. 89
4.35. Diagrama de dispersión entre atributos detectados y tiempo de respuesta. . . 90
1.36. Vista exterior del vehículo de prueba utilizado para la instalación y la posición de sus componentes. 102

1.37. Vista general del interior del vehículo, donde se ve el cableado del sistema repartido por el techo, con la Raspberry Pi 5 ubicada del lado izquierdo y el módulo SIM7600G-H a la vista, junto al espejo retrovisor. 102
1.38. Configuración del proceso de fine-tuning sobre el modelo base YOLOv8n. Se aprecia la arquitectura de 130 capas con 3,011,823 parámetros, la transferencia de 355 pesos preentrenados y los parámetros de augmentación aplicados al dataset de 2901 imágenes. 103
1.39. Resultados del entrenamiento del modelo YOLOv8n tras completar las 200 épocas en Google Colab con GPU Tesla T4. Se muestran las métricas finales por clase y la ruta del mejor modelo guardado. 103
1.40. Vista general del dataset en la plataforma Roboflow, compuesto por 1.315 imágenes de entrenamiento. Cada imagen muestra el número de anotaciones totales y las clases presentes, con entre 1 y 10 instancias etiquetadas por imagen correspondientes a las cinco clases del proyecto: gorra, gafas, mascarilla, persona y null. 104
1.41. Distribución de instancias por clase dentro del dataset del proyecto, según lo que se ve en la plataforma Roboflow. Ahí se nota claramente el desbalance entre clases: mascarilla tiene la mayor cantidad de instancias, con 1.448, seguida de persona con 1.155 y gorra con 241, mientras que gafas cuenta con 216 instancias y null apenas llega a 11. Este desbalance es justo lo que explica por qué el modelo rindió distinto según la clase, siendo mascarilla la que obtuvo el mejor mAP50 y gorra la más baja. 104
1.42. Portada del datasheet oficial del VL53L0X de STMicroelectronics, donde se describen sus características principales: medición de distancia por tiempo de vuelo (ToF) hasta 2 metros, emisor láser de 940 nm, comunicación I2C y cumplimiento de la norma de seguridad ocular IEC 60825-1:2014 [5]. . . . 105
1.43. Datasheet del VL53L0X resume sus principales características técnicas: encapsulado óptico LGA12 de 4,4 x 2,4 x 1 mm, voltaje de operación de 2,6 V a 3,5 V, rango de temperatura de -20 a 70°C, emisor infrarrojo de 940 nm y comunicación I2C a hasta 400 kHz [5]. 106
1.44. Diseño del soporte para las cámaras CSI Pi Camera Module 3, hecho en el software Creality Print, con dimensiones de 29,80 × 42,80 × 10,00 mm. El diseño tiene ranuras de ventilación y un sistema de fijación con tornillos, pensado para sujetar la cámara a la carrocería del vehículo. 106
1.45. Diseño del soporte para el sensor de distancia VL53L0X en Tinkercad, con una cavidad ajustada al encapsulado del sensor y una ranura frontal que permite la línea de visión directa del láser hacia el exterior del vehículo. . . 107

1.46. Escenario 6 — Reconocimiento del propietario en Zona Z0. La etiqueta *DUEÑO* en color verde confirma que el sistema identificó correctamente el rostro del propietario y no generó ninguna alerta. Dado que en este escenario el sistema no dispara alerta, tampoco guarda fotografía automáticamente, por lo que se presenta una captura de pantalla como evidencia del funcionamiento correcto. 107
1.47. Detección de atributo mascarilla en Zona Z2 a las 10:56:57. El sistema identificó el uso de mascarilla sobre el rostro de la persona y acumuló el atributo dentro de la ventana de 5 segundos para su posterior evaluación. . . 108
1.48. Detección en condiciones de contraluz en la Zona Z0, a las 10:41:46. A pesar de la iluminación adversa, el sistema logró detectar la presencia de la persona y mostró la etiqueta *Persona [sin atrib.]*, lo que muestra que no se juntaron suficientes atributos para activar la alerta por ese mecanismo, quedando el temporizador de permanencia como respaldo. 108
1.49. Detección al mismo tiempo de gorra y mascarilla en la Zona Z0. El sistema identificó ambos atributos de ocultamiento facial en la misma persona, superando así el umbral de dos atributos necesario para que se active la alerta por sospechoso. 109
1.50. Detección de atributos en una escena con dos personas dentro de la Zona Z1, a las 10:34:00. El sistema identificó gorra y gafas en la persona de la izquierda, y mascarilla en la persona de la derecha, lo que muestra la capacidad del modelo para analizar a varias personas dentro de un mismo cuadro al mismo tiempo. 109
1.51. Detección de mascarilla en un entorno exterior, desde la Zona Z0, a las 12:14:06. Esta prueba confirma que el sistema funciona bien en condiciones de luz natural, con la cámara apuntando hacia el exterior del vehículo. . . . 110

RESUMEN

El presente trabajo describe el diseño, desarrollo e implementación de un sistema de seguridad vehicular inteligente de vigilancia perimetral basado en hardware embebido, visión artificial y redes de telecomunicaciones móviles. El sistema propuesto enfrenta al incremento de delitos que se cometen a los vehículos, proponiendo una solución que busca ser más rápida y eficiente que la vigilancia tradicional. El núcleo del sistema es una mini computadora Raspberry Pi 5, la cual controla un conjunto de sensores láser medidores de distancia, basados en tecnología de Tiempo de Vuelo (ToF), distribuidos a lo largo del vehículo. Para ahorrar energía y no saturar el procesador, el sistema funciona de forma inteligente: los sensores vigilan todo el contorno del auto en silencio y, solo cuando alguien se acerca a menos de un metro, el programa ordena que se enciendan las cámaras de video para registrar lo que está pasando. Del lado del software, las imágenes que captan las cámaras llegan al sistema, encargado de procesarlas con la ayuda de un motor de inteligencia artificial, que va analizando de forma continua lo que sucede frente al vehículo, identificando rostros y extrayendo sus rasgos más relevantes. Con esa información, el sistema compara lo que ve con las fotografías almacenadas previamente en su base de datos, lo que le permite determinar si la persona frente al vehículo es el propietario o alguien desconocido. En caso de que el sistema identifique a alguien que no está autorizado, toma de forma automática una fotografía que queda como registro del evento y, a través del módulo de comunicación celular, le envía de inmediato un mensaje de texto (SMS) al dueño del vehículo para notificarle sobre la situación. Las pruebas realizadas con el prototipo real demostraron que el proyecto es totalmente viable. El sistema midió las distancias con gran precisión, manejó los videos de las cámaras con total estabilidad y envió las alertas celulares sin fallas. Se concluye que combinar sensores de corto alcance con inteligencia artificial, es una solución confiable y eficiente para desarrollar sistemas de seguridad vehicular modernos y avanzados.

Palabras clave: Hardware Embebido, Visión Artificial, Tiempo de Vuelo, Reconocimiento Facial, Comunicación Celular, Seguridad Vehicular.

ABSTRACT

This work describes the design, development, and implementation of an intelligent vehicle security system for perimeter surveillance based on embedded hardware, computer vision, and mobile telecommunications networks. The proposed system addresses the rising number of crimes committed against vehicles, offering a solution that aims to be faster and more efficient than traditional surveillance methods. The core of the system is a Raspberry Pi 5 single-board computer, which controls a set of laser distance sensors based on Time-of-Flight (ToF) technology, distributed around the vehicle. To save energy and avoid overloading the processor, the system operates intelligently: the sensors silently monitor the entire perimeter of the car, and only when someone approaches within one meter does the program trigger the video cameras to start recording what is happening. On the software side, the images captured by the cameras are sent to the system, which processes them with the help of an artificial intelligence engine that continuously analyzes what is happening in front of the vehicle, identifying faces and extracting their most relevant features. Using this information, the system compares what it sees against the photographs previously stored in its database, allowing it to determine whether the person in front of the vehicle is the owner or someone unknown. If the system identifies someone who is not authorized, it automatically takes a photograph as a record of the event and, through the cellular communication module, immediately sends a text message (SMS) to the vehicle owner to notify them of the situation. Tests carried out with the real prototype demonstrated that the project is fully viable. The system measured distances with high accuracy, handled the camera video feeds with complete stability, and sent cellular alerts without failures. It is concluded that combining short-range sensors with artificial intelligence is a reliable and efficient solution for developing modern, advanced vehicle security systems.

Keywords: Embedded Hardware, Computer Vision, Time-of-Flight, Facial Recognition, Cellular Communication, Vehicle Security.

Approved by Jacqueline Armijos

CAPÍTULO I

1.1 Introducción

En la actualidad, dejar el auto estacionado se ha convertido en una preocupación constante, ya que la delincuencia vehicular golpea no solo el bolsillo de los ciudadanos y comercios, sino también la tranquilidad pública a nivel global [9]. Cuando se habla de este tipo de delitos, no nos referimos únicamente al robo total del coche, sino también al mercado negro de autopartes y repuestos ilícitos [9]. Este escenario va más allá del impacto directo al propietario; genera pérdidas financieras masivas a las aseguradoras, afecta la reputación de los fabricantes automotrices y, por lo general, se encuentra estrechamente vinculado con redes de delincuencia organizada más complejas [9].

A nivel nacional, los datos de seguridad en el Ecuador reflejan un crecimiento preocupante de los delitos relacionados con vehículos —entre ellos el vandalismo, el hurto y el robo total—, con mayor incidencia en las ciudades más pobladas del país [10, 11]. Ante esta realidad, han surgido distintas propuestas tecnológicas orientadas a reducir los riesgos a los que están expuestos los propietarios de automóviles. Sin embargo, aunque hoy en día se cuenta con sistemas de videovigilancia, alarmas, sensores de presencia y localización GPS, la mayoría de estas herramientas operan de manera aislada [12]. Esta falta de integración limita una respuesta inmediata y coordinada frente a situaciones de vulnerabilidad real [12].

Por su parte, el procesamiento digital de imágenes es una disciplina informática enfocada en la manipulación y análisis de registros visuales mediante algoritmos, cuyo fin es extraer información valiosa o preparar el entorno para tareas complejas de visión artificial [1]. Bajo esa premisa, el presente trabajo tiene como finalidad el diseño e implementación de un sistema perimetral unificado que potencie los métodos de seguridad tradicionales a través de algoritmos de inteligencia artificial. Esta propuesta busca anticiparse al delito evaluando el comportamiento de las personas en las proximidades del vehículo, empleando técnicas de visión computacional para detectar conductas de movimiento atípicas que estadísticamente puedan asociarse a una intención de hurto [13].

En los siguientes capítulos de este documento se presentan las bases teóricas que respaldan la investigación, abarcando conceptos fundamentales de seguridad informática, visión artificial, aprendizaje automático y análisis del comportamiento humano. Asimismo, se detalla la metodología empleada para el diseño e implementación y las pruebas experimentales del sistema propuesto.

1.2 Planteamiento del Problema

En Ecuador, el número de robos de vehículos presenta un incremento en los últimos tres años. En el 2022, hubo 11.372 denuncias interpuestas en la fiscalía general del Estado. Significa un aumento de casi el 150 % si se compara con el 2020, que tuvo 4.596 robos. Mientras que hasta agosto de 2023 suman 6.858 [10, 11]. El robo de vehículos figura entre los delitos con mayor impacto social, en parte por la frecuencia con que ocurre y por el peso simbólico que el automóvil tiene en la vida diaria de gran parte de la población, sobre todo en los sectores de clase media. Para muchas personas, el vehículo representa una señal de progreso económico y estatus social; por eso, su sustracción —con o sin violencia— no solo implica una pérdida material, sino que también genera una sensación de vulnerabilidad, miedo e indefensión, y pone en entredicho la idea de que los bienes privados están protegidos [14, 15]. Un indicador claro de la repercusión social de este delito es la elevada tasa de denuncias que genera en comparación con otros tipos de crímenes. Prácticamente la totalidad de estos casos termina siendo reportada ante las autoridades policiales y judiciales, en buena medida porque las compañías aseguradoras lo exigen como requisito para activar los procesos de indemnización [15]. A su vez, este alto nivel de reporte permite que las circunstancias en que se producen estos hechos sean bien conocidas por las instituciones competentes, lo que en teoría debería favorecer el diseño e implementación de políticas públicas o mecanismos de seguridad orientados a reducir su incidencia.

1.3 Justificación

El desarrollo de este proyecto responde directamente a la urgente necesidad de contrarrestar la crisis de seguridad vehicular que afecta al territorio ecuatoriano, la cual vulnera de forma constante el patrimonio y el bienestar familiar. Para la mayoría de los ciudadanos, especialmente dentro de la clase media, un automóvil representa años de esfuerzo, una herramienta de trabajo indispensable y un símbolo de progreso [15]. Por consiguiente, sufrir su pérdida no solo provoca un grave impacto económico, sino que genera un profundo estado de vulnerabilidad, impotencia y desamparo en la sociedad.

Los mecanismos de protección tradicionales, como las alarmas sonoras o los sistemas de rastreo pasivos, han mostrado ser insuficientes frente a las estrategias que emplean los delincuentes hoy en día. Su mayor debilidad está en que actúan de manera reactiva: solo se activan una vez que el vehículo ya ha sido vulnerado o cuando el intruso logra acceder a su interior, lo que deja un margen de tiempo muy reducido para prevenir el daño [12, 16].

Frente a este panorama, el sistema propuesto se plantea como una solución tecnológica moderna y de carácter preventivo, con una justificación tanto técnica como práctica. A través de la combinación de visión computacional y sensores de distancia por tiempo de vuelo (ToF), el dispositivo es capaz de vigilar de forma activa el perímetro cercano del automóvil, diferenciando de manera automatizada entre el propietario y una persona con comportamiento sospechoso. Al superar el modelo de seguridad pasiva y automatizar tanto el envío de evidencia fotográfica como las alertas en tiempo real a través de mensajería móvil, esta investigación demuestra que es posible ofrecer un mecanismo de protección inteligente, eficiente y accesible, orientado a resguardar bienes de alto valor antes de que el delito llegue a consumarse.

1.4 Objetivos

1.4.1 General

- Desarrollar un sistema de prevención de delitos contra vehículos estacionados aplicando procesamiento de imágenes.

1.4.2 Específicos

- Diseñar un sistema de vigilancia en base al comportamiento de las personas alrededor de un vehículo.

- Implementar un sistema de vigilancia basado en visión artificial, capaz de identificar situaciones de riesgo en vehículos estacionados y generar alertas en tiempo real.

- Evaluar el rendimiento del sistema desarrollado mediante pruebas de campo, con el propósito de determinar su eficacia en la prevención de delitos contra vehículos estacionados.

CAPÍTULO II

2.1 Estado del Arte

Últimamente se ha investigado mucho sobre cómo usar la visión artificial para vigilar y prevenir incidentes. Se ha probado diferentes métodos y aplicaciones para lograr que estos sistemas sean más eficientes y precisos.

Según lo expuesto por Villa Yuquilema [13], el "Diseño e implementación de un sistema inteligente basado en visión artificial para el monitoreo y prevención de robos en tiempo real en la farmacia Pharmatodo de la ciudad de Riobamba" describe que el sistema de detección obtiene resultados alentadores con un rendimiento excepcional en las condiciones ideales analizadas, bajo una iluminación adecuada con un fondo simple, siendo capaz de detectar armas de fuego cortas y armas blancas.

Un ejemplo de esto es el estudio de Draghici [17], donde se usaron redes neuronales para analizar imágenes tomadas por cámaras y así ubicar las placas de los vehículos dentro de ellas. Según los resultados obtenidos, el sistema reconocía los caracteres con un 98 % de precisión, mientras que la lectura completa y correcta de la matrícula se logró en un 80 % de los casos evaluados.

Otro ejemplo es el trabajo de Ludeña Chica [18], quien diseñó un sistema para supervisar niños de 2 a 4 años y brindar seguridad en todos los entornos que se encuentren. Sus resultados indican que su propuesta funciona un 15 % mejor que los sistemas convencionales que se usan en el hogar, confirmando que la visión artificial realmente mejora el monitoreo y la prevención de incidentes.

También está el estudio de Morales Camacho [19], quien creó un sistema con monitoreo web para contar pasajeros en buses. Su proyecto utiliza un algoritmo que ayuda a que la cámara se adapte sola a los cambios de luz y no se confunda con las sombras o el movimiento del fondo, logrando un conteo más exacto.

El robo de vehículos ha impulsado el desarrollo de diversas soluciones tecnológicas a nivel mundial. En 2024, Pavithra y Muruganantham [20] presentaron un sistema de vigilancia que combina YOLOv7 con análisis de trayectorias para identificar comportamientos asociados a hurtos, tomando en cuenta variables como la velocidad, la energía del movimiento y los cambios de dirección de las personas dentro del área monitoreada. Sus resultados mostraron una mejora notable frente a modelos anteriores, con aplicabilidad tanto en viviendas como en zonas de estacionamiento, lo que lo convierte en un referente directo para el presente trabajo.

Como complemento a este enfoque, un estudio publicado en *Artificial Intelligence Review* [21] planteó un sistema basado en una versión mejorada de YOLOv5, entrenado específicamente para reconocer en tiempo real a personas con conductas sospechosas. Uno de los aportes más interesantes de este trabajo fue que el sistema lograba mantener un buen desempeño incluso en condiciones adversas, como poca iluminación, objetos parcialmente ocultos o imágenes de baja resolución, factores que normalmente representan un dolor de cabeza en entornos de vigilancia reales.

Desde una perspectiva más orientada al vehículo en sí, una patente tecnológica registrada en Estados Unidos [22] describe un sistema embebido directamente en el automóvil que, mientras este se encuentra estacionado, activa sus cámaras para detectar personas sospechosas en los alrededores mediante redes neuronales convolucionales. Lo que hace especialmente interesante a esta propuesta es su capacidad de aprendizaje continuo: con cada incidente registrado en la zona, el sistema mejora su capacidad de reconocimiento, adaptándose al entorno específico donde opera.

A partir de estos antecedentes, queda claro que combinar modelos YOLO con análisis de comportamiento y notificaciones en tiempo real es una estrategia viable y efectiva para prevenir delitos mediante visión artificial. No obstante, ninguno de estos sistemas fue diseñado pensando en la realidad de países como Ecuador, donde el robo a vehículos estacionados en zonas urbanas representa un problema creciente y sin soluciones tecnológicas locales consolidadas. Es precisamente esa brecha la que motiva el presente trabajo, que adapta estos principios al diseño e implementación de un sistema de prevención orientado al monitoreo de vehículos estacionados mediante procesamiento de imágenes.

2.2 Fundamentación Teórica

2.2.1 Análisis estadístico del robo de vehículos

La presente investigación emplea registros estadísticos oficiales sobre robos y hurtos de vehículos, los cuales constituyen datos cuantitativos respecto a la incidencia de conductas delictivas en el entorno social. Dichos datos provienen de los registros oficiales de la Policía Judicial y la Fiscalía General del Estado, que a nivel territorial opera mediante sedes provinciales, denominadas Fiscalía Provincial de Pichincha en Quito y Fiscalía Provincial del Guayas en Guayaquil [14].
En cuanto a la delimitación conceptual de hurto y robo, debemos aclarar que su aplicación se ancla al Código Penal Ecuatoriano (1971), que divide a los delitos contra la propiedad en los siguientes artículos:

- Art. 547.- "[Hurto]. - Son reos de hurto los que, sin violencia ni amenaza contra las personas, ni fuerza en las cosas, sustrajeren fraudulentamente una cosa ajena, con ánimo de apropiarse."

- Art. 550.- "[Robo]. - El que, mediante violencias o amenazas contra las personas o fuerza en las cosas, sustrajere fraudulentamente una cosa ajena, con ánimo de apropiarse, es culpado de robo, sea que la violencia tenga lugar antes del acto para facilitarlo, en el momento de cometerlo, o después de cometido para procurar su impunidad."

El gráfico 2.1 muestra cómo se distribuyen los delitos contra vehículos según la situación en que ocurren, comparando Quito y Guayaquil en el año 2008. Cuando se trata de vehículos estacionados, Quito tiene el número más alto con 83,75, mientras que Guayaquil tiene 54,29. En el caso de vehículos que están en movimiento, Guayaquil tiene más delitos que Quito, con 45,71 frente a 14,27. Estos datos indican que en Quito la mayoría de los delitos contra vehículos suceden cuando el vehículo está estacionado. En cambio, Guayaquil tiene una distribución más igualada entre vehículos estacionados y vehículos en movimiento [14].

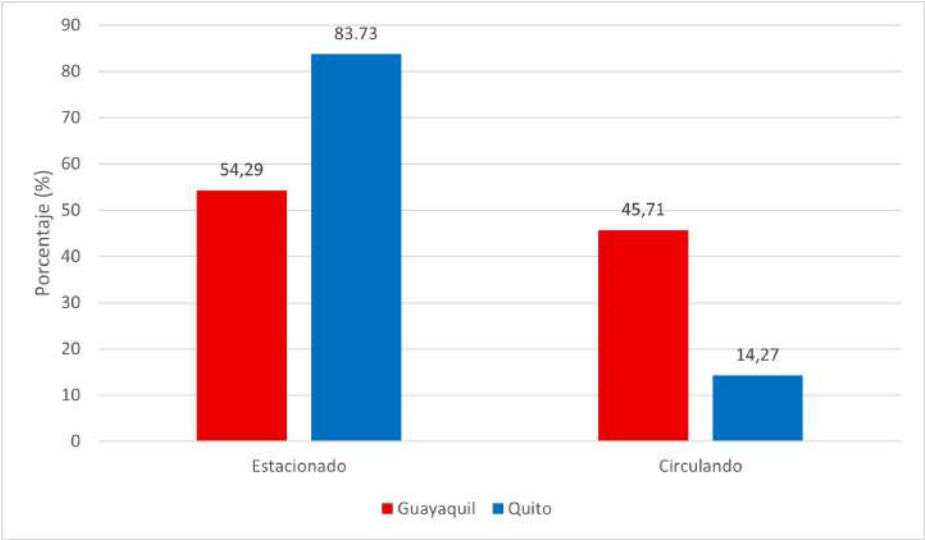


Figura. 2.1. Delitos a vehículos según la circunstancia (2008) [14].

En Quito, una de las recomendaciones pasa por algo tan simple como no dejar el carro solo en la calle o en parqueaderos que no ofrezcan mayor seguridad. En Guayaquil, en cambio, debería ir más por el lado de reforzar la vigilancia, con patrullajes constantes tanto a pie como en vehículo [14]. Este proyecto se centrará en uno de los delitos más comunes en las calles de Ecuador, el robo de vehículos. Las cifras hablan por sí solas:

- En 2017 se registraron 4.541 robos de vehículos, número que subió a 4.714 al año siguiente.

- En 2019, esta cifra ascendió a 5.653, lo que supone un incremento de 939 vehículos en comparación con el año anterior. En lo que respecta al año 2020, se obtuvo que en la Región Oriente ocurrieron tan solo 46 robos, a diferencia de Sierra, que tuvo 1.628 y Costa 2.904 [11].

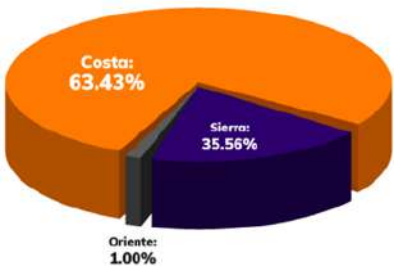


Figura. 2.2. Robo de autos por región. Cifras 2021 [11].

La Figura 2.2 muestra cómo se distribuyeron los robos de vehículos en Ecuador en el año 2020. La región Costa tuvo la mayor cantidad de robos de vehículos con un 63,43 por ciento.

La región Sierra sigue con un 35,56 por ciento. La región Oriente es la que menos robos de vehículos tuvo con solo un 1 por ciento. Esto muestra que la región Costa es donde más se roban vehículos en Ecuador. La región Costa acumula casi dos tercios de todos los robos de vehículos del país [11].

El robo de vehículos es, sin duda, uno de los problemas de seguridad que más preocupa al Ecuador en los últimos años. Basta con revisar las cifras: solo entre enero y junio de 2024, la Fiscalía General del Estado registró 5.665 casos de robo vehicular a nivel nacional, siendo Guayas la provincia más golpeada por este delito, seguida de El Oro, Los Ríos y Santo Domingo de los Tsáchilas [23]. Solo en Guayaquil se reportaron 2.918 denuncias formales durante ese año, aunque la propia ciudadanía sospecha que la cifra real es mucho más alta, ya que buena parte de los casos nunca llega a denunciarse [24]. Algo parecido ocurrió en Quito, donde los robos de carros prácticamente se duplicaron en los primeros meses de 2024 respecto al año anterior, mientras que el hurto de motocicletas tuvo un incremento adicional del 10 % [25].

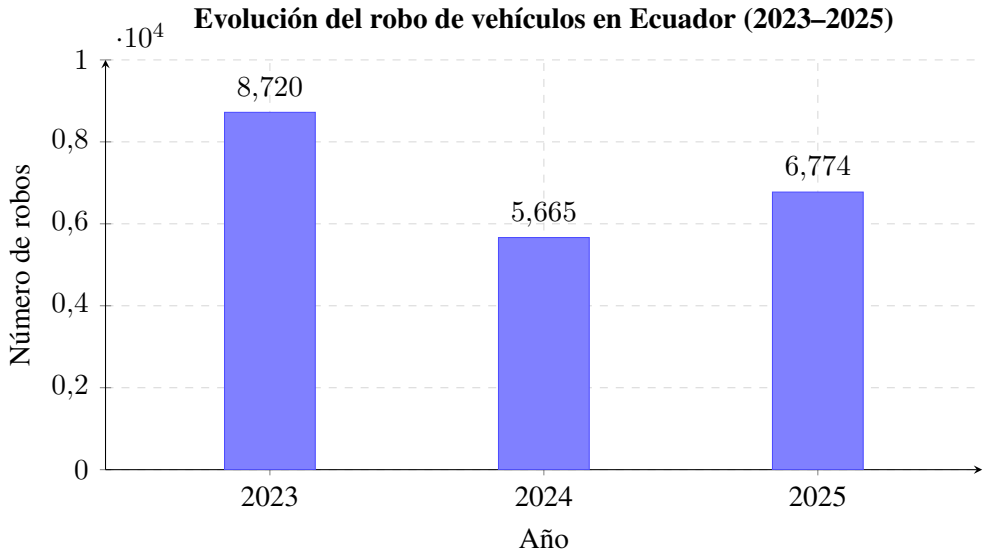


Figura. 2.3. Evolución del robo de vehículos en Ecuador entre 2023 y 2025. Fuente: Fiscalía General del Estado [4, 23].

Lejos de mejorar, el panorama se tornó más crítico en 2025. Hasta octubre de ese año, la Fiscalía contabilizó 6.774 vehículos robados a nivel nacional. Solo en Quito se reportaron 2.191 casos, superando incluso el total registrado durante todo 2024, lo que se traduce en aproximadamente un robo cada 3,5 horas en la capital [4]. La Figura 2.3 ilustra la evolución del robo de vehículos en Ecuador entre 2023 y 2025. En ella se observa que en 2023 se registraron aproximadamente 8.720 casos, cifra que descendió a 5.665 en 2024 durante el primer semestre, para luego repuntar a 6.774 casos hasta octubre de 2025, evidenciando que, pese a una leve reducción intermedia, la tendencia general se mantiene en niveles alarmantes y con señales de nuevo crecimiento hacia finales del período analizado. A nivel de distritos, el

sector Eugenio Espejo, ubicado en el norte de Quito, se posicionó como la zona de mayor riesgo del país en cuanto a sustracción de vehículos, de acuerdo con el Plan Operativo Anual Integral 2026 de la Policía Nacional [26].

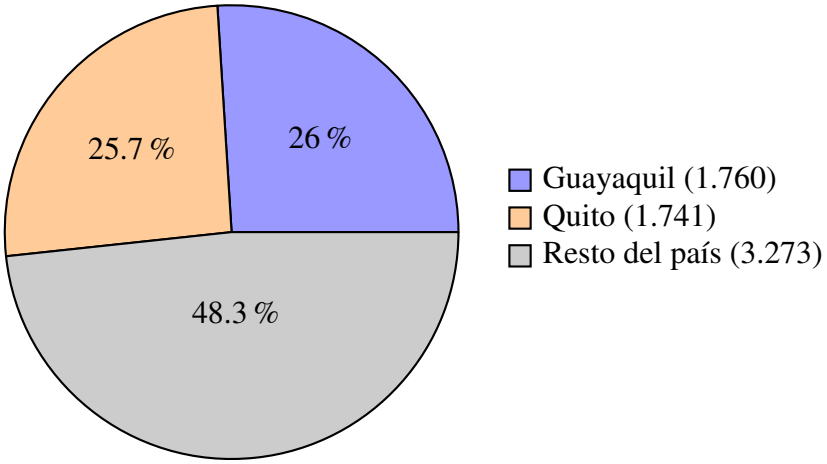


Figura. 2.4. Distribución de robos de vehículos en Ecuador hasta octubre de 2025 [4].

Por otro lado, la Figura 2.4 presenta la distribución geográfica de los robos vehiculares registrados entre enero y septiembre de 2025 a nivel nacional. El gráfico revela que Guayaquil concentró el 26 % del total de casos con 1.760 denuncias, mientras que Quito registró una proporción muy similar con el 25,7 % y 1.741 casos. En conjunto, ambas ciudades acumularon más del 51 % de todos los robos vehiculares del país en ese período. El 48,3 % restante, equivalente a 3.273 casos, se distribuyó entre las demás provincias del Ecuador, lo que demuestra que, si bien el problema se concentra principalmente en las dos urbes más grandes del país, ninguna región queda exenta de esta problemática [4].

Frente a este escenario, resulta evidente que el problema no se limita a una simple cifra: golpea de manera directa el bolsillo y la tranquilidad de muchas familias ecuatorianas. Por eso es tan importante apostar por soluciones tecnológicas que ayuden a prevenir y detectar a tiempo cualquier actividad sospechosa en zonas de parqueo, como la que se plantea en este trabajo.

2.2.1.1 Métodos de robos a vehículos

a) Robos mediante Rotura de Vidrio

Una modalidad común de robo a vehículos en Ecuador involucra la rotura de vidrios. Esta técnica suele ser la preferida por los delincuentes cuando buscan entrar al vehículo de forma rápida y sin levantar sospechas. Por lo general, los estacionamientos poco vigilados o las zonas urbanas con poca iluminación terminan

siendo el escenario ideal para este tipo de hechos. Y cuando no hay medidas preventivas de por medio, como alarmas o cámaras de seguridad, se vuelve mucho más fácil que ocurra un robo a través de la rotura de vidrios [27].

b) Robos mediante Forzamiento de la Puerta del Vehículo

Otra de las tácticas que suelen usar los ladrones es forzar directamente la puerta del vehículo, recurriendo a herramientas como palancas o destornilladores para manipular la cerradura y entrar sin levantar mayor sospecha. Este tipo de robo se da con frecuencia en lugares poco concurridos, donde el delincuente tiene más tiempo para actuar sin ser notado [27].

c) Acercamiento de Personas Sospechosas al Vehículo.

Antes de que ocurra el robo como tal, suele haber un momento previo en el que el delincuente simplemente se acerca al vehículo para echar un vistazo: busca objetos de valor a la vista, calcula qué tan fácil sería entrar, o simplemente espera el momento oportuno. Muchas veces, el descuido del propietario —ese segundo en que no presta atención a su alrededor, o esa ventana que quedó mal cerrada— es justo lo que termina convirtiendo una simple observación en una oportunidad de robo [27].

2.2.1.2 Criminología ambiental

La conducta criminal encuentra influencia en el ambiente inmediato donde ocurre, el cual participa como un elemento criminógeno que, según precisa Vázquez, afecta tanto al comportamiento como al proceso de toma de decisiones. Bajo esta idea, surge la conocida como "criminología ambiental". En este sentido, se muestra como una aproximación criminológica cuyo objetivo no es otro que aportar conocimientos a la par que soluciones en el campo tanto de análisis, como de intervención y prevención de la delincuencia [28].

2.2.1.3 Criterios de selección del vehículo objetivo

Los modelos de vehículos con mayor penetración en el mercado presentan una mayor susceptibilidad al robo. La alta disponibilidad de estas unidades alimenta tanto el mercado de vehículos robados por encargo como el comercio ilegal de autopartes. De acuerdo con [28], los consumidores pueden auditar esta vulnerabilidad mediante el análisis de la robustez de los sistemas de seguridad del automóvil y el índice de propensión al robo de sus piezas. Dichos indicadores son accesibles a través de plataformas de difusión masiva, oficinas estadísticas institucionales y consorcios aseguradores.

En el contexto ecuatoriano, esta tendencia se manifiesta de manera explícita en la composición del parque automotor siniestrado. La marca Chevrolet registra la mayor incidencia de sustracciones, representando aproximadamente el 40 % del total de los casos reportados a nivel nacional. Entre los modelos con mayor vulnerabilidad estadística destacan el Chevrolet Aveo, el Chevrolet D-Max, el Chevrolet Sail y el Chevrolet Grand Vitara. Asimismo, otras marcas de gran circulación en el país, como Toyota (en sus modelos Hilux y Land Cruiser) y Hyundai (con Accent y Tucson), registran una presencia recurrente en las bases de datos policiales [29].

2.2.2 Capa 1: Percepción y Hardware Integrado

2.2.2.1 Principio de Funcionamiento de la Tecnología Time-of-Flight (ToF)

La telemetría basada en el principio de Tiempo de Vuelo es una técnica que utiliza la luz para medir la distancia entre un sensor y un objeto. Esto se hace midiendo el tiempo que tarda una señal de luz en viajar desde el sensor hasta el objeto y regresar. La señal de luz se propaga por el aire y el tiempo que tarda en hacerlo nos da la distancia. Esta técnica se utiliza para determinar la distancia entre el sensor y el objeto objetivo. A diferencia de los métodos de triangulación estereoscópica o los sensores ultrasónicos tradicionales, que son altamente susceptibles al ruido geométrico y a las condiciones de eco ambiental, la tecnología ToF fundamenta su precisión en la constancia de la velocidad de la luz en el vacío ($c \approx 3 \times 10^8$ m/s) [5].

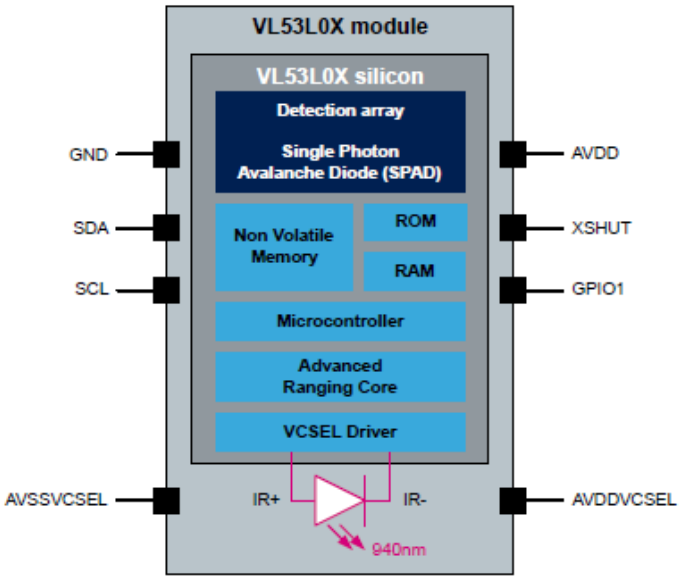


Figura. 2.5. Diagrama de bloques de VL53L0X [5].

La Figura 2.5 muestra el diagrama de bloques del módulo VL53L0X. Como se puede observar, el sensor integra en un solo chip todos los elementos necesarios para medir distancias. El componente principal es el arreglo de detección SPAD, que es el encargado de captar la luz que rebota desde el objeto detectado. Internamente, el módulo dispone de memoria y un microcontrolador propio que procesa la información recibida. Para emitir el pulso de luz, el sensor utiliza un láser infrarrojo de 940 nm, invisible al ojo humano, controlado por el driver VCSEL. Finalmente, la comunicación con el sistema externo se realiza a través del protocolo I2C mediante los pines SDA y SCL [5].

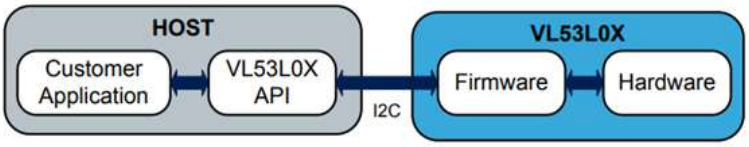


Figura. 2.6. Descripción funcional del sensor [5].

El modelo funcional de interconexión del sensor VL53L0X se presenta en la Figura 2.6. El sistema se divide en el bloque Host (dispositivo de control principal) y el módulo periférico VL53L0X. La interacción entre la aplicación del usuario y el dispositivo se realiza a nivel de software a través de la API del sensor, comunicándose mediante el protocolo I2C como canal físico. Por dentro, el VL53L0X cuenta con un firmware propio que se encarga de controlar el hardware óptico (emisor láser y detector) para la ejecución y procesamiento de las lecturas telemétricas [5].

En sistemas de rango de corto alcance, se emplea predominantemente la técnica de ToF de onda continua modulada (CW-ToF) o ToF por pulsos. El dispositivo emite un haz de luz infrarroja no visible (generalmente a una longitud de onda de 940 nm mediante un Láser de Emisión Superficial con Cavidad Vertical o VCSEL). Los fotones viajan hacia el objetivo, se reflejan en su superficie y regresan hacia una matriz de fotonivelación o sensor de fotodiodos de avalancha en el receptor. La distancia (d) se calcula matemáticamente mediante la siguiente ecuación cinemática fundamental (2.1):

$$d = \frac{c \cdot \Delta t}{2} \tag{2.1}$$

Donde Δt representa el tiempo de tránsito de ida y vuelta de los fotones. El factor de división para dos (2) compensa la naturaleza bidireccional del trayecto de la señal [30].

2.2.2.2 Características Técnicas del Sensor VL53L0X

Para la implementación de la monitorización perimetral del sistema de vigilancia vehicular predictivo, se seleccionó el sensor de rango láser VL53L0X de STMicroelectronics. Este componente reúne en un solo encapsulado modular un emisor VCSEL de 940 nm junto con una matriz de detectores de fotones únicos de avalancha (SPAD, Single Photon Avalanche Diodes). La inclusión de tecnología SPAD permite al sensor discriminar el ruido óptico ambiental y los reflejos secundarios, lo que permite obtener mediciones precisas sin que la reflectancia del objeto afecte el resultado (colores de vestimenta, carrocería o texturas del sujeto) con un rango operativo de hasta 2000 mm en condiciones óptimas [5], esto se lo puede apreciar en la sección de anexos, en la Figura 1.43.

2.2.2.3 Protocolo de Comunicación Inter-Integrated Circuit (I2C)

La transferencia de datos de telemetría desde los nodos sensores hacia la unidad de procesamiento central (Raspberry Pi 5) se ejecuta a través del bus serie síncrono Inter-Integrated Circuit (I2C). Este protocolo de arquitectura Maestro-Esclavo utiliza únicamente dos líneas bifásicas activas: SDA (Serial Data Line) para el envío de datos binarios, y SCL (Serial Clock Line), encargada de la señal de reloj que sincroniza la comunicación, generada por el dispositivo maestro [31].
Las líneas del bus I2C presentan una configuración de colector abierto o drenador abierto, lo que exige acoplar resistencias de Pull-Up conectadas a la línea de voltaje lógico positivo ($V_{CC} = 3{,}3$ V) para mantener el bus en un estado lógico alto pasivo cuando los dispositivos se encuentran en ralentí. La velocidad de transferencia configurada corresponde al modo estándar (Standard-mode a 100 kbps) o modo rápido (Fast-mode a 400 kbps), garantizando un comportamiento temporal predecible, adecuado para la lectura secuencial de los entornos perimetrales [31].

2.2.2.4 Direccionamiento Dinámico mediante Pines de Apagado por Hardware (XSHUT)

Por diseño de hardware de fábrica, todos los circuitos integrados VL53L0X comparten una dirección esclava física idéntica de 7 bits indexada en el valor hexadecimal 0x29. Debido a que el sistema requiere la coexistencia de cuatro sensores distribuidos espacialmente para cubrir un rango omnidireccional de 360 grados, conectar los dispositivos de forma directa al mismo bus I2C generaría un conflicto de colisión de direcciones en el bus de datos, inhabilitando la comunicación [5].

Para mitigar esta restricción sin recurrir a hardware multiplexor externo (como el TCA9548A),

se implementó una metodología de direccionamiento dinámico por software aprovechando el pin de apagado por hardware, denominado XSHUT. El pin XSHUT es una entrada lógica digital de tipo activa en bajo (Active-Low). Cuando este pin se drena a un potencial de tierra (0 V o LOW), las etapas internas del sensor y sus interfaces lógicas se quedan sin energía, entrando así en un estado de aislamiento o hibernación profunda (Shutdown), desconectándose lógicamente del bus I2C.
A continuación se describe el proceso de inicialización y direccionamiento dinámico, de acuerdo con los criterios de ingeniería que se aplicaron en el firmware:

1. Al arrancar el sistema, la unidad central (maestro) pone en estado LOW todos los pines GPIO conectados a las líneas XSHUT, con lo que se apagan los cuatro sensores al mismo tiempo.

2. Luego se energiza solamente el primer sensor (el Nodo Z0, por ejemplo), llevando su GPIO XSHUT a un estado de alta impedancia o HIGH (3,3 V). Esta transición queda estabilizada gracias al resistor de Pull-Up acoplado, y el sensor arranca con su dirección por defecto, que es 0x29.

3. Acto seguido, el maestro envía a través del bus I2C un comando para reconfigurar los registros, indicándole al dispositivo que cambie su dirección interna volátil por un nuevo identificador único (por ejemplo, 0x30).

4. Una vez guardado el cambio, el maestro repite el procedimiento de forma iterativa con el resto de los sensores (Z1, Z2, Z3), asignándoles las direcciones consecutivas 0x31, 0x32 y 0x33 respectivamente. Al finalizar la rutina, los cuatro nodos ToF operan de manera simultánea e independiente dentro del mismo bus de datos de telecomunicaciones, siguiendo los criterios de ingeniería del firmware [5].

2.2.2.5 Cerebro del procesamiento

Raspberry Pi 5 como Unidad Central de Procesamiento
Cuando se diseña un sistema de vigilancia inteligente que debe funcionar de manera autónoma, una de las decisiones más importantes es elegir qué dispositivo va a ser el "cerebro" que procese toda la información. Para el sistema propuesto se eligió la Raspberry Pi 5 con 8 GB de memoria RAM, una pequeña computadora del tamaño de una tarjeta de crédito que, a pesar de su tamaño reducido, tiene la capacidad suficiente para ejecutar modelos de inteligencia artificial, controlar múltiples cámaras y gestionar comunicaciones en tiempo real [32]. La Raspberry Pi 5 fue lanzada en octubre de 2023 por la Raspberry Pi Foundation y representa un salto importante respecto a sus versiones anteriores [32]. Su procesador cuenta con cuatro núcleos ARM Cortex-A76 funcionando a 2.4 GHz, lo que en términos sencillos significa que puede realizar varias tareas al mismo tiempo sin que unas interfieran con otras [33].

Esto es especialmente importante para este trabajo, donde simultáneamente se están leyendo cuatro cámaras, analizando imágenes con inteligencia artificial, monitoreando sensores de distancia y esperando enviar alertas cuando sea necesario. Los 8 GB de memoria RAM permiten que el modelo de detección YOLOv8 permanezca cargado en memoria durante toda la operación del sistema, evitando los tiempos de espera que ocurrirían si hubiera que cargarlo cada vez que se necesita analizar una imagen [32]. Además, la Raspberry Pi 5 incluye un nuevo chip de entrada y salida llamado RP1, diseñado internamente por la Raspberry Pi Foundation, que mejora significativamente el rendimiento de los puertos USB y las interfaces de cámara respecto a modelos anteriores [34].

Libcamera y la Gestión de Cámaras CSI

Las cámaras utilizadas en las zonas Z0 y Z1 del sistema se conectan a la Raspberry Pi 5 mediante una interfaz denominada CSI (Camera Serial Interface), que es básicamente un canal de comunicación de alta velocidad directamente integrado en la placa [35]. A diferencia de las cámaras USB que compiten por el mismo canal de datos que otros dispositivos, las cámaras CSI tienen su propio camino dedicado hacia el procesador, lo que reduce los retrasos y permite obtener imágenes más fluidas [35]. Para gestionar estas cámaras, la Raspberry Pi 5 utiliza libcamera, una librería de software de código abierto que reemplazó al sistema anterior (raspistill/raspivid) a partir de 2022 [36]. Libcamera actúa como intermediario entre el hardware de la cámara y las aplicaciones que necesitan usar las imágenes, encargándose automáticamente de ajustes como la exposición, el balance de blancos y el enfoque [36].

En términos cotidianos, es como el sistema que en un teléfono inteligente se encarga de que las fotos salgan bien iluminadas y nítidas sin que el usuario tenga que configurar nada manualmente.

Un aspecto destacable de libcamera es que entrega las imágenes en formato YUV420, una forma eficiente de representar el color que reduce el tamaño de los datos sin sacrificar la información visual necesaria para detectar objetos [36]. Gracias a esto, el sistema logra procesar las imágenes con mayor rapidez y un menor uso de memoria, algo fundamental considerando que se están analizando cuatro zonas al mismo tiempo en un equipo de bajo consumo como la Raspberry Pi 5.

2.2.3 Capa 2: Inteligencia Artificial y Visión

2.2.3.1 Procesamiento de imágenes

Los sistemas electrónicos para el procesamiento de imágenes son dispositivos diseñados para capturar, procesar y analizar imágenes de manera eficiente. Estos sistemas incluyen componentes como sensores de imagen, circuitos de captura y unidades de procesamiento de

imágenes. Los sistemas electrónicos utilizan algoritmos avanzados para realizar operaciones como filtrado, segmentación y reconocimiento de patrones. Las técnicas de proceso de imágenes mejoran la apariencia visual de las imágenes para que sean más fáciles de ver y también preparan las imágenes para que las máquinas puedan entenderlas [1].

Vista desde un punto más conceptual, el procesamiento digital de imágenes se puede entender a partir de cuatro operaciones básicas:

- **Adquisición o captura:** tiene que ver con los distintos medios para obtener la imagen, sea mediante cámaras digitales o digitalizando fotografías analógicas.

- **Realce y mejora:** busca optimizar la apariencia visual de la imagen o recuperar información que se haya perdido en imágenes deterioradas.

- **Segmentación:** se encarga de dividir la imagen en regiones o áreas que tengan sentido para su análisis.

- **Extracción de características:** consiste en identificar y ubicar elementos geométricos dentro de la imagen, desde algo tan simple como líneas y puntos, hasta formas más complejas como curvas y cuádricas [1].

Estas cuatro operaciones son, en cierto modo, la base teórica sobre la que luego se construyen arquitecturas más completas de procesamiento. La Tabla 2.1, por ejemplo, muestra cómo estas operaciones se organizan en la práctica dentro de un sistema electrónico orientado a la visión artificial, sumando además otras etapas como la transformación y filtrado, y el almacenamiento y gestión de los datos obtenidos.

2.2.3.2 Etapas Procesamiento de imágenes y videos.

Comprende una serie de componentes y etapas que trabajan en conjunto para analizar y procesar imágenes de manera eficiente y efectiva [27]. Además, permite obtener información valiosa a partir de los datos visuales. La Tabla 2.1 presenta las etapas que conforman la arquitectura de un sistema electrónico de procesamiento de video orientado a la visión artificial. El proceso inicia con la adquisición del video mediante cámaras o dispositivos de entrada, seguido del procesamiento para mejorar la calidad de la imagen. Posteriormente se realiza la extracción de características relevantes como bordes, texturas y formas, para luego aplicar técnicas de transformación y filtrado. Finalmente, el sistema ejecuta la segmentación para separar objetos de interés del fondo, y concluye con el almacenamiento y gestión eficiente de los datos obtenidos [37].

Tabla 2.1. Etapas de la arquitectura de sistemas electrónicos para el procesamiento de videos enfocada a la visión artificial [1]

Etapa	Descripción
Adquisición de videos	Captura de videos utilizando cámaras, escáneres u otros dispositivos de entrada.
Procesamiento de imágenes y videos	Ajustes de contraste, reducción de ruido y otras mejoras que aumentan la calidad y claridad del video.
Extracción de características	Identificación de elementos relevantes dentro del video, como bordes, texturas y formas.
Transformación y filtrado	Aplicación de distintas técnicas para modificar y optimizar el contenido visual.
Segmentación de videos	Separación de los objetos o regiones de interés respecto al fondo, mediante algoritmos especializados.
Almacenamiento y gestión de datos	Organización y manejo eficiente de las imágenes y la información asociada dentro de los sistemas de almacenamiento.

2.2.3.3 Visión artificial

La visión computacional, también llamada visión artificial, es una parte de la inteligencia artificial que permite a los sistemas digitales ver como lo hace un ser humano. Se estudia y se aplica para desarrollar sistemas que puedan procesar, analizar y entender imágenes o videos. El objetivo principal de la visión computacional es que las computadoras puedan observar y entender lo que las rodea. Para lograr esto, se utilizan cámaras y sensores que capturan el entorno, y algoritmos avanzados y modelos de aprendizaje automático que procesan esos datos para reconocer formas, colores, personas u objetos [13].
Hoy en día, la visión computacional tiene un gran impacto en sectores importantes, como la medicina, la robótica, los sistemas de vigilancia, la automatización industrial y el guiado de vehículos autónomos. Se está implementando con éxito en estos campos [13].

2.2.3.4 Etapas de procesamiento de imágenes con visión artificial

Vale aclarar que esto son solo lineamientos generales, ya que dependiendo del método o algoritmo que se utilice, los pasos descritos pueden variar.

1. **La captura:** el objetivo de este proceso es brindar imágenes digitales al sistema a través de componentes externos como por ejemplo la utilización de cámaras, video-cámaras,

etc [38].

2. **Pre-procesamiento:** en esta etapa lo que se busca es resaltar ciertos detalles o características de la imagen, además de aplicar técnicas que ayuden a reducir el ruido y mejorar el contraste [38].

3. **Segmentación:** en este proceso a la imagen se la divide en regiones o grupos de pixeles que sean de cierto interés de estudio, es crucial esta etapa para el funcionamiento del sistema de detección [38].

4. **Descripción:** en este proceso, el sistema selecciona los rasgos visuales clave que permiten distinguir un objeto de otro. Esto incluye analizar elementos fundamentales como su geometría, el tipo de textura y la gama de colores, entre otras propiedades particulares [38].

5. **Reconocimiento:** este proceso clasifica los objetos según las características del paso anterior, los objetos detectados que presentan características similares se agrupan a una misma categoría [38].

6. **Interpretación:** es el proceso que da sentido o un significado a las clases (categorías) de objetos reconocidos para entender la secuencia. Trata de emular la visión humana y utiliza técnicas cognitivas para la toma de decisiones. Esta fase depende de cada campo de aplicación [38].

Los procesos son generalmente secuenciales y los resultados obtenidos en cada una son los recursos para la siguiente fase. Los procesos que se utilice en la resolución de un determinado problema dependen de su complejidad y no todas son siempre necesarias [38].

2.2.3.5 Algoritmos principales de reconocimiento biométrico

Los algoritmos de la visión artificial se definen como un conjunto de procesos e instrucciones que les permite interpretar y comprender imágenes y a partir de esa interpretación tomar decisiones. A lo largo del tiempo se han presentado numerosos algoritmos que hacen posible la extracción de una variedad de características lo que se puede notar en la eficiencia de estas técnicas [13].

El Algoritmo de Histograma de Gradientes Orientados (HOG)

Para optimizar el uso del procesador en la Raspberry Pi 5 y evitar retardos en la transmisión de video, el sistema recurre a este algoritmo durante la primera fase de detección de estructuras faciales. El método HOG fundamenta su principio en que la apariencia y la forma de un objeto local dentro de una imagen pueden describirse mediante la distribución de las direcciones de

los gradientes de intensidad o los contornos de los bordes [39].

El procesamiento consta de cuatro etapas que se ejecutan una después de otra:

1. **Normalización del color:** en este paso se reducen los efectos que los cambios de iluminación general puedan tener sobre el cuadro de video.

2. **Cálculo del gradiente:** se obtienen las componentes horizontales (G_x) y verticales (G_y) de cada píxel, lo que permite calcular tanto la magnitud ($|G|$) como la orientación del contorno (θ):
$$|G| = \sqrt{G_x^2 + G_y^2} \tag{2.2}$$
$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{2.3}$$

3. **Construcción del histograma:** la imagen se divide en pequeñas celdas conectadas entre sí (por ejemplo, bloques de 8×8 píxeles), dentro de las cuales se acumulan los registros de las orientaciones del gradiente, escaladas de acuerdo con su magnitud.

4. **Agrupación en bloques:** los histogramas de las celdas se normalizan en bloques más grandes y superpuestos para contrarrestar los cambios de contraste y sombras, generando el descriptor final que el sistema utiliza para confirmar la presencia geométrica de un rostro [39].

Redes Convolucionales (CNN) y Extracción de Características (Face Encodings)
Una vez localizado el contorno del rostro, el sistema requiere transformarlo en un perfil digital único e invariable frente a rotaciones, expresiones o cambios de perspectiva. Para ello, se emplea una arquitectura basada en Redes Neuronales Convolucionales (CNN) preentrenada, integrada dentro de la librería de código abierto dlib [40]. Pensado como un modelo que busca replicar el funcionamiento del sistema de visión humano, una Red Neuronal Convolucional (CNN) es altamente efectiva en tareas como el reconocimiento de rostros, ya que analiza las imágenes por partes aplicando filtros que reducen la información para quedarse solo con lo más importante [41, 42].

En este sentido, la red neuronal de este sistema procesa el recorte de la imagen facial a través de múltiples capas convolucionales que extraen características esenciales de manera jerárquica, identificando desde elementos básicos como bordes y curvas, hasta rasgos complejos como la distancia interocular, la simetría nasal y el contorno mandibular; finalmente, la capa de salida proyecta estos rasgos visuales en un espacio vectorial continuo de alto orden, generando un vector de 128 dimensiones cuantitativas denominado Face Encoding o huella geométrica facial [43].

Métrica de Distancia Euclidiana y Tolerancia de Reconocimiento
La toma de decisiones biometrométricas para clasificar a un sujeto como "Usuario Autorizado" o "Sujeto Sospechoso" se reduce a un problema geométrico de cálculo de distancias en un espacio multidimensional. El script de Python calcula la Distancia Euclidiana (d) entre el vector del rostro capturado en tiempo real (X) y los vectores de referencia previamente almacenados en la base de datos local (Y) mediante la siguiente función matemática:

$$d(X, Y) = \sqrt{\sum_{i=1}^{128}(x_i - y_i)^2} \tag{2.4}$$

Donde x_i e y_i representan los valores de las 128 características del vector extraído y el vector de referencia, respectivamente. El sistema implementa un umbral crítico de tolerancia ($0{,}5$). Si el resultado de la distancia euclidiana es inferior a este umbral, el algoritmo dictamina una coincidencia positiva de identidad [44]. Si la distancia supera el límite establecido, el sujeto se etiqueta automáticamente bajo la categoría de sospechoso, activando de inmediato los protocolos de alerta SMS por red celular.

Warmup MOG2
El warmup MOG2 es un período de "calentamiento" del algoritmo de detección de movimiento antes de empezar a analizarlo en serio. MOG2 (Mixture of Gaussians v2) es un algoritmo que aprende cómo se ve el fondo "normal" de una escena para detectar cuando algo cambia [45].

2.2.3.6 Aprendizaje Automático y Aprendizaje Profundo

Machine Learning
El aprendizaje automático, también conocido como Machine Learning, es una rama de la inteligencia artificial cuyo objetivo principal es crear algoritmos que permitan a los sistemas computacionales aprender y mejorar su desempeño. Esto se logra a partir de datos, sin necesidad de ser programados explícitamente para cada tarea en particular. El aprendizaje automático busca que los sistemas puedan aprender de los datos y mejorar con el tiempo, sin intervención humana directa en cada paso [46]. A diferencia de la programación tradicional, donde las reglas son definidas manualmente por el desarrollador, en el Machine Learning el algoritmo las infiere estadísticamente a partir de ejemplos de entrenamiento [46]. La cantidad y calidad de los datos con los que se entrena el modelo resultan decisivas: cuando se cuenta con conjuntos de datos amplios y representativos, los resultados tienden a ser mucho más precisos, mientras que datos erróneos o sesgados degradan el rendimiento del sistema [47]. Detrás del funcionamiento del Machine Learning están las redes neuronales artificiales, estructuras que se inspiran en el funcionamiento biológico del cerebro humano [48]. Estas redes están compuestas por unidades de procesamiento interconectadas denominadas neuronas artificiales, organizadas en capas que reciben señales de entrada, aplican transformaciones matemáticas

y emiten señales de salida. El ajuste iterativo de los pesos de estas conexiones mediante algoritmos de optimización como el descenso del gradiente es lo que constituye el proceso de aprendizaje [48].

Deep Learning

El aprendizaje profundo, o Deep Learning, representa una variante más avanzada dentro del aprendizaje automático que emplea redes neuronales con múltiples capas ocultas, capaces de aprender a reconocer patrones complejos directamente desde los datos crudos, sin necesidad de definir manualmente las características relevantes como ocurre en el Machine Learning clásico [49]. Esta capacidad ha impulsado avances significativos en áreas como la visión por computadora, el procesamiento de lenguaje natural y la conducción autónoma, donde sistemas como los de Tesla emplean modelos de Deep Learning para reconocer objetos y patrones en tiempo real [50]. En el contexto del presente sistema, el Deep Learning constituye la base tecnológica que permite identificar elementos visuales en imágenes, tales como la detección de accesorios faciales —gorras, gafas de sol o mascarillas— mediante el análisis de las imágenes capturadas por las cámaras de vigilancia.

Transfer Learning y Fine-tuning

El aprendizaje por transferencia (transfer learning) es una técnica que consiste en aprovechar los pesos de un modelo que ya fue entrenado previamente en un conjunto de datos grande como punto de partida para entrenar en una tarea específica con datos más limitados [51]. En el caso de YOLOv8, los pesos base se obtienen del entrenamiento sobre COCO (Common Objects in Context), un dataset de más de 330,000 imágenes con 80 categorías de objetos cotidianos [52]. El proceso de fine-tuning reemplaza únicamente la cabeza de detección del modelo para adaptarla al número de clases del nuevo problema, conservando las capas iniciales, encargadas de extraer características visuales genéricas como bordes, texturas y formas [51]. De esta forma, el modelo no necesita aprender a reconocer características básicas desde cero, reduciendo el tiempo de entrenamiento de varios días a pocas horas y el requisito de datos de decenas de miles de imágenes a unos pocos cientos por clase [51]. Para el sistema propuesto, se partió de los pesos yolov8n.pt y se reentrenó sobre un dataset de cinco clases (gorra, gafas, mascarilla, null, persona), obteniendo un modelo funcional en aproximadamente 50 minutos sobre una GPU T4 en Google Colab [6].

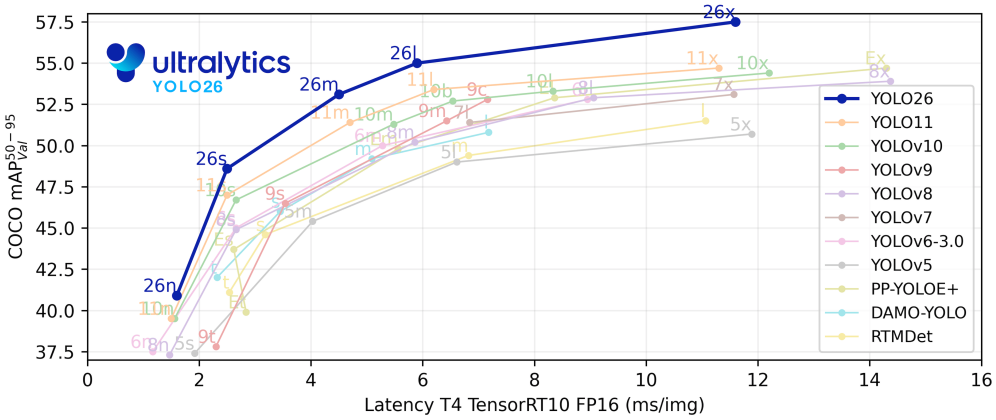


Figura. 2.7. Precisión COCO mAP vs. latencia de inferencia para distintas arquitecturas YOLO. Adaptado de Ultralytics [6].

Como se aprecia en la Figura 2.7, la familia YOLOv8 (representada por la curva morada) se compone de diferentes variantes escaladas según la complejidad de la red (Nano, Small, Medium, Large y Extra Large). Para el desarrollo de este proyecto, se seleccionó específicamente la variante YOLOv8n (Nano), la cual se ubica en el primer nodo de dicha curva. Esta elección se justifica debido a que, aunque se encuentra en el extremo inferior de precisión mAP de su línea, ofrece la menor latencia de procesamiento y el menor tamaño de parámetros, requisitos indispensables para asegurar una tasa de fotogramas por segundo (FPS) adecuada en un hardware embebido con recursos limitados, como es el caso de la Raspberry Pi 5.

2.2.3.7 Arquitectura YOLO (You Only Look Once)

A diferencia de otros sistemas que primero buscan zonas de interés y luego las clasifican paso a paso, la familia de modelos YOLO, propuesta en 2016 [7], aborda la detección de objetos como un único proceso continuo. Básicamente, segmenta la imagen en una cuadrícula para identificar los elementos y sus ubicaciones al mismo tiempo a través de un solo recorrido de la red. Este enfoque directo es el secreto detrás de su increíble velocidad para operar en tiempo real [7].

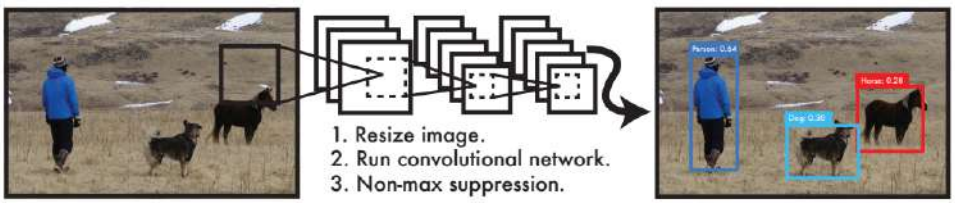


Figura. 2.8. El sistema de detección YOLO [7].

En la figura 2.8 se puede apreciar el procesamiento de imágenes con YOLO, el cual es simple y directo, el sistema (1) cambia de tamaño la imagen de entrada, (2) ejecuta una única red convolucional en la imagen y (3) establece umbrales para las detecciones resultantes mediante la confianza del modelo [7].

Lanzado por Ultralytics en 2023, YOLOv8 introduce mejoras clave en su estructura. En primer lugar, deja atrás las "anclas predefinidas" (anchor-free), lo que significa que el sistema calcula las cajas de detección directamente sobre el objeto sin depender de plantillas rígidas. Además, optimiza el procesamiento de datos con un diseño de conexiones mejorado y estrena una "cabeza de detección desacoplada". Esta última innovación separa por completo la tarea de ubicar el objeto en la pantalla de la tarea de identificar qué es, logrando que el modelo sea notablemente más preciso y rápido [6]. Esto mejora la precisión especialmente en objetos pequeños como gorras, gafas de sol y mascarillas [53]. La familia YOLOv8 ofrece cinco variantes según su capacidad computacional: nano (n), small (s), medium (m), large (l) y extra-large (x). Para sistemas embebidos como la Raspberry Pi 5 sin unidad de procesamiento gráfico dedicada, la variante nano es la más adecuada, alcanzando velocidades de inferencia de entre 15 y 25 FPS en CPU, frente a los 6-10 FPS de la variante small, con una pérdida de precisión inferior al 3 % en mAP50 cuando el modelo se reentrena con un dataset suficientemente grande [6]. Como se puede ver en la Figura 2.9, la arquitectura de YOLOv8 está formada por tres módulos principales. Primero está el *backbone*, que se encarga de extraer las características visuales de la imagen. Luego viene el *neck* o cuello, encargado de combinar características de distintas escalas a través de una red piramidal. Y finalmente, la *cabeza de detección desacoplada*, que es la responsable de predecir tanto la ubicación como la clase del objeto detectado, todo al mismo tiempo.

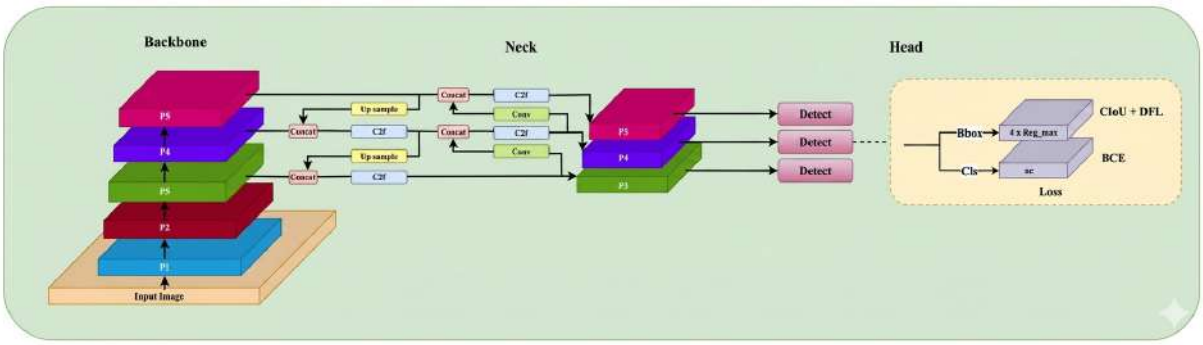


Figura. 2.9. Arquitectura básica del modelo de detección de objetos YOLOv8 [8].

La confianza de una detección se calcula como:

$$\text{Confianza} = P(\text{Objeto}) \times \text{IoU}_{\text{truth}}^{\text{pred}} \tag{2.5}$$

donde $P(\text{Objeto})$ es la probabilidad de que exista un objeto en la celda, e IoU es la intersección sobre la unión entre la caja predicha y la real.

La función de pérdida total del modelo combina tres componentes:

$$\mathcal{L} = \lambda_{\text{box}} \cdot \mathcal{L}_{\text{box}} + \lambda_{\text{cls}} \cdot \mathcal{L}_{\text{cls}} + \lambda_{\text{dfl}} \cdot \mathcal{L}_{\text{dfl}} \tag{2.6}$$

donde $\mathcal{L}_{\text{box}}$ es la pérdida de localización de la caja, $\mathcal{L}_{\text{cls}}$ la pérdida de clasificación, $\mathcal{L}_{\text{dfl}}$ la pérdida focal distribuida, y λ los pesos de cada término [7, 53].

Tabla 2.2. Variantes del modelo YOLOv8 y sus características [2, 3].

Variante	**Parámetros**	**mAP50**	**FPS (CPU)**
YOLOv8n (nano)	3.2 M	37.3	15–25
YOLOv8s (small)	11.2 M	44.9	6–10
YOLOv8m (medium)	25.9 M	50.2	3–5
YOLOv8l (large)	43.7 M	52.9	2–3
YOLOv8x (extra)	68.2 M	53.9	1–2

2.2.3.8 Métricas de Evaluación

Para saber si un modelo de detección de objetos realmente funciona bien, no basta con medir si adivina qué tipo de objeto tiene enfrente; también se evalúa qué tan preciso es al dibujar el recuadro que marca su ubicación exacta en la imagen [54]. Básicamente, la precisión nos dice qué porcentaje de las detecciones hechas por el modelo fueron aciertos reales. Su función es evaluar si el sistema está seguro de lo que ve, asegurando que cuando marque un recuadro de alerta, realmente haya un objeto verdadero ahí y no un error de detección [54]. Por otro lado, el recall evalúa la capacidad del modelo para que no se le escape nada. En términos sencillos, mide cuántos de los objetos que realmente estaban presentes en la escena logró identificar el algoritmo, sin importar si en el proceso también hizo algunas detecciones incorrectas por el camino [54]. Existe una relación inversa entre ambas métricas: aumentar la sensibilidad del modelo incrementa el recall pero reduce la precisión al generar más falsos positivos [54].

El mAP50 (Mean Average Precision al umbral IoU de 0.5) es la métrica principal en detección de objetos [55]. Calcula el área bajo la curva precisión-recall para cada clase y promedia el resultado entre todas las clases, considerando una detección como correcta cuando la superposición entre la caja pronosticada y la caja real supera el 50 % [55]. Para el sistema propuesto, un mAP50 superior a 0.75 por clase se considera aceptable para su operación [6].

La matriz de confusión complementa estas métricas mostrando visualmente qué clases se confunden entre sí [54]. En el caso del presente sistema, la confusión más frecuente ocurrió entre gafas de sol y gorra, atribuible al desbalance de instancias en el dataset de entrenamiento

y a la similitud visual de ambos objetos cuando son capturados por cámaras de baja resolución en condiciones de poca iluminación.

2.2.3.9 Algoritmos Auxiliares de Procesamiento en Visión Artificial

Detección de Contornos
Este algoritmo procesa las imágenes para identificar los límites, bordes y discontinuidades de brillo en los objetos dentro del cuadro de video. En el sistema de vigilancia vehicular, la detección de contornos se utiliza para delimitar las siluetas humanas y de objetos del entorno. Al calcular los cambios bruscos de intensidad de los píxeles, el software puede aislar formas geométricas del fondo (como la silueta de un sospechoso intentando evadir los sensores), facilitando los pasos posteriores de segmentación y análisis biométrico [56].

Extracción de Características
Consiste en encontrar detalles importantes en una imagen. Detalles que no cambian mucho aunque la luz o el tamaño de la imagen varíe. Dentro de un sistema, este algoritmo busca los datos clave del rostro de una persona. Por ejemplo, la distancia entre los ojos, la forma de las cejas o el tamaño de la nariz. Esta extracción reduce la imagen pesada de la cámara a un conjunto de datos manejable para mapear la identidad del usuario [56].

Segmentación de Imágenes
La segmentación es, básicamente, el proceso de dividir una imagen en distintas partes o grupos de píxeles que se parecen entre sí, ya sea por su color, su brillo o su textura. La idea detrás de esto es simplificar la imagen para poder analizarla de forma más sencilla y entendible. En este proyecto, gracias a la segmentación, el algoritmo logra separar el rostro de la persona detectada del resto de lo que aparece alrededor, como el fondo o el entorno del vehículo. Al crear una "máscara" o región de interés exclusiva para la cara, se evita que el sistema gaste recursos de procesamiento analizando elementos irrelevantes del paisaje [56].

Reconocimiento de Texto
El reconocimiento de texto (Optical Character Recognition - OCR), se encarga de localizar, extraer y traducir caracteres alfanuméricos presentes en una matriz de píxeles a texto digital legible por la máquina. Aunque el enfoque principal del prototipo actual es la biometría facial, la inclusión de algoritmos OCR abre una capacidad predictiva secundaria clave en la seguridad vehicular: la lectura automática de matrículas o placas vehiculares (ANPR). Esto permite registrar los caracteres de identificación de vehículos sospechosos que se ubiquen dentro del perímetro de cobertura de las cámaras [57].

Análisis de Movimiento
Este algoritmo de análisis de movimiento (Motion Analysis and Tracking), detecta y realiza el seguimiento del desplazamiento temporal de objetos o sujetos a lo largo de una secuencia

consecutiva de fotogramas de video. En el sistema propuesto, el análisis de movimiento trabaja en conjunto con los sensores de Tiempo de Vuelo (ToF). Mientras los sensores miden la proximidad física, este algoritmo de software analiza la dirección y velocidad del movimiento del sospechoso en el flujo de video, permitiendo predecir si el individuo simplemente camina junto al auto o si se mantiene estático en una zona de riesgo con intenciones de vulnerar el bien patrimonial [58].

2.2.4 Capa 3: Telecomunicaciones y Alertas

Esta capa representa la "voz" del sistema, es la encargada de comunicar al propietario del vehículo cuando se detecta una situación de riesgo. Para lograrlo, el sistema combina la red celular móvil, comunicación serie con el módem y servicios de mensajería instantánea, formando un canal de alerta robusto que funciona incluso sin conexión a internet fija.

2.2.4.1 Arquitectura de Redes Celulares GSM/LTE

Las redes celulares dividen el territorio en zonas denominadas celdas, cada una atendida por una estación base que gestiona la comunicación entre los dispositivos móviles y la red [59]. GSM (Global System for Mobile Communications) fue el estándar de segunda generación (2G) que introdujo la comunicación digital en telefonía móvil y estableció el servicio de mensajes cortos SMS como canal de comunicación de texto entre dispositivos [59]. Aunque actualmente existen tecnologías más avanzadas, el SMS sigue siendo el mecanismo de alerta más confiable en situaciones donde la conectividad a internet puede ser limitada, ya que opera sobre el canal de señalización de la red y no requiere datos móviles [60].

LTE (Long Term Evolution), conocida como red 4G, es la evolución del estándar GSM que permite velocidades de transferencia de datos significativamente mayores gracias a técnicas de modulación avanzadas y el uso de múltiples antenas [60]. El módulo utilizado en este sistema, el SIM7600G-H, es un dongle de grado industrial que integra en un mismo dispositivo las tecnologías 2G (GSM/GPRS/EDGE), 3G (HSPA+) y 4G LTE Cat-4, con soporte para bandas globales LTE-FDD y LTE-TDD, alcanzando velocidades de hasta 150 Mbps en descarga y 50 Mbps en carga [61].

La figura 2.10 ilustra la arquitectura de integración GSM/LTE implementada en el sistema. El módulo SIM7600G-H se conecta a la Raspberry Pi 5 mediante interfaz USB, exponiéndose como puerto serial (`/dev/ttyUSB2`) a través del cual se envían comandos AT para la gestión de SMS y datos. Cuando el sistema detecta una amenaza, utiliza simultáneamente ambas tecnologías: la red GSM para el envío inmediato del SMS de alerta al propietario, garantizando la notificación incluso con señal de datos débil; y la red LTE 4G para la transmisión de fotografías del intruso y la ubicación aproximada del vehículo a través de la

API de Telegram [60].

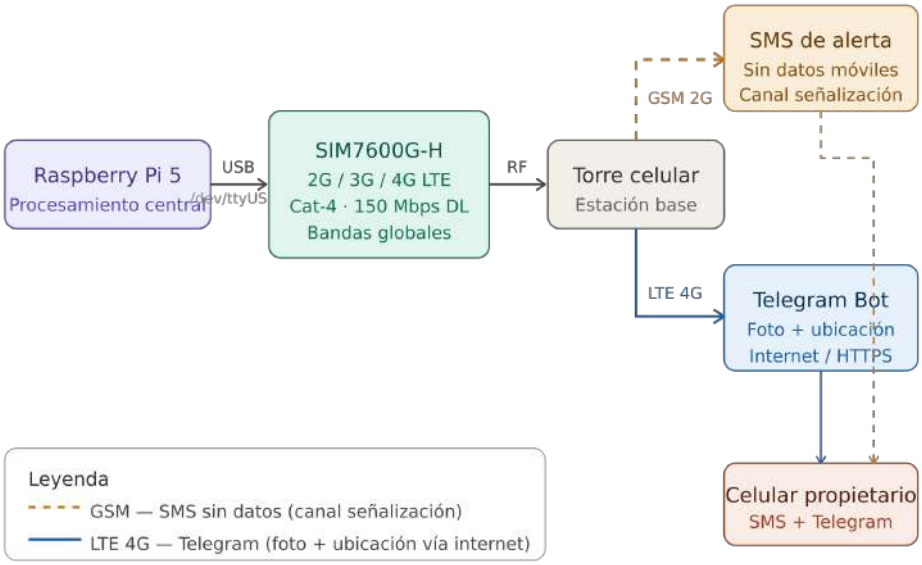


Figura. 2.10. Arquitectura de integración GSM/LTE del módulo SIM7600G-H en el sistema.

2.2.4.2 Interfaces UART/USB en Linux

La comunicación entre la Raspberry Pi 5 y el módem SIM7600G-H se realiza a través de una interfaz serie. UART (Universal Asynchronous Receiver-Transmitter) es uno de los protocolos de comunicación serie más antiguos y simples que existen: transmite datos bit a bit entre dos dispositivos usando únicamente dos líneas, una para enviar (TXD) y otra para recibir (RXD), sin necesidad de un reloj compartido entre los dispositivos [62]. Su simplicidad lo hace especialmente confiable en sistemas embebidos donde la estabilidad es prioritaria. En sistemas Linux como el que ejecuta la Raspberry Pi 5, los dispositivos serie se representan como archivos dentro de la carpeta /dev/, siguiendo el principio fundamental de Unix de que "todo es un archivo" [63]. Los módems conectados por USB como el SIM7600G-H aparecen como /dev/ttyUSB0, /dev/ttyUSB1 y así sucesivamente, dependiendo del orden en que son detectados por el sistema [63]. Leer y escribir en estos archivos desde Python es equivalente a enviar y recibir datos directamente por el puerto serie, lo que simplifica enormemente la programación de la comunicación con el módem.

2.2.4.3 Protocolo de Comandos AT

Los comandos AT, también conocidos como Hayes Command Set en honor a su creador Dennis Hayes, son un lenguaje de texto estándar para controlar módems [64]. Fueron introducidos en 1981 y se convirtieron en el estándar universal de la industria para la comunicación con dispositivos de telefonía, siendo adoptados posteriormente por los módems GSM/LTE modernos como el SIM7600G-H [64]. Cada instrucción comienza con las letras "AT" (abreviatura de "ATtention") seguidas de un código que indica la acción deseada [64]. Por ejemplo, el comando $AT + CMGF = 1$ configura el módem para trabajar con mensajes de texto en formato legible, mientras que $AT + CMGS = " + 593XXXXXXXXX"$ inicia el proceso de envío de un SMS al número indicado. El módem responde con OK cuando la instrucción se ejecutó correctamente o con un código de error cuando algo falló [64]. En el sistema, estos comandos se envían automáticamente desde Python a través del puerto serie cada vez que se detecta un sospechoso, sin intervención del usuario.

2.2.4.4 Telegram Bot API

Telegram es una aplicación de mensajería que permite a los usuarios enviar mensajes, fotos y archivos. También ofrece una herramienta para que los desarrolladores creen bots. Estos bots son cuentas automáticas que pueden enviar y recibir mensajes, fotos, ubicaciones y archivos sin que una persona tenga que intervenir [65]. Un bot de Telegram se crea a través de BotFather, el bot oficial de la plataforma, que genera un token único de autenticación que identifica al bot y le permite interactuar con la API [65].

La comunicación con la Telegram Bot API se realiza mediante peticiones HTTPS al servidor de Telegram, continuando el patrón REST (Representational State Transfer) [65]. Esto significa que el sistema simplemente envía una solicitud web al servidor de Telegram con la foto, el mensaje o la ubicación, y Telegram se encarga de entregarlo al destinatario configurado. En el sistema se utilizan tres métodos principales de la API: sendPhoto para enviar la fotografía del sospechoso junto con el mensaje de alerta, sendLocation para enviar un pin de mapa interactivo con la ubicación aproximada del vehículo, y sendMessage para notificaciones de texto cuando no hay foto disponible [65]. La combinación de SMS y Telegram como canales de alerta garantiza redundancia: si uno falla, el otro asegura que la notificación llegue al propietario.

CAPÍTULO III

3.1 Metodología

3.1.1 Tipo de Investigación

Este proyecto es una investigación aplicada con enfoque cuantitativo, ya que toma teorías existentes de visión artificial, electrónica y telecomunicaciones para crear una solución real: un sistema autónomo que cuida vehículos estacionados. El diseño es experimental porque construimos un prototipo físico para ponerlo a prueba en escenarios controlados. Durante estas pruebas, medimos numéricamente datos clave como el porcentaje de aciertos al detectar personas, la distancia exacta a la que reaccionan los sensores y cuántos segundos tarda el sistema en enviar las alertas al usuario.

3.1.2 Diagrama Operativo

Para el desarrollo del presente trabajo, se seguirán una serie de procedimientos específicos, mismos que constan en el diagrama de bloques que se observa en la Figura 3.11.

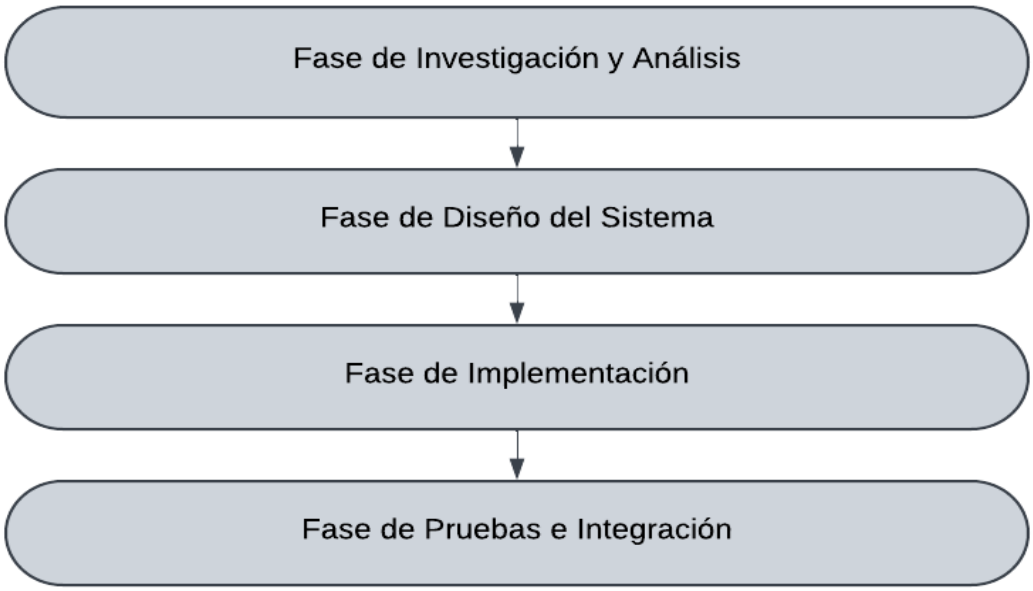


Figura. 3.11. Diagrama de Proceso de la Metodología de Proyecto.

Como se observa en la Figura 3.11 nos indica que el presente trabajo consta de 4 fases:

- La primera fase, investigación y análisis, tiene como objetivo comprender el contexto del problema y definir los requisitos del sistema, para ello se realizarán análisis de las estadísticas de robos para poder identificar patrones comunes, también se revisarán tecnologías como cámaras de seguridad, sensores, alarmas y sistemas de alertas ya existentes. Esta fase nos ayudará a determinar qué necesita hacer el sistema y cómo hacerlo. También definiremos los requisitos funcionales y no funcionales.

- Por ende, en la segunda fase corresponde al diseño del sistema, donde se tiene como objetivo crear una arquitectura que integre los componentes claves que formarán parte del sistema propuesto. Se propuso que en el diseño de la arquitectura integrarán los siguientes componentes claves: sensores de distancia láser, cámaras de seguridad con análisis de video e integración de alertas por mensaje.

- Posteriormente, teniendo ya el diseño, en la tercera fase se procederá con el desarrollo del sistema en donde se buscará implementar los componentes del sistema. Como también programar el software para el procesamiento de datos, análisis de video y gestión de alertas. Se realizó la comunicación entre los dispositivos y el servidor central, además de las pruebas unitarias para garantizar que cada módulo funciona de forma independiente.

- Por último, en la cuarta fase, tiene por objeto validar el sistema en condiciones reales, donde se realizarán pruebas de campo para verificar que los módulos trabajan correctamente juntos, se simularán escenarios de robo para probar el tiempo de respuesta del sistema, se validará la precisión de los sensores y cámaras. Por último, se buscará comparar el sistema desarrollado con otros sistemas convencionales.

3.1.3 Población y Muestra

3.1.3.1 Población

La población de estudio corresponde a las variables que se van a medir en cualquier vehículo estacionado en zonas urbanas que pueda ser objeto de un delito o actividad sospechosa en su entorno inmediato.

3.1.3.2 Muestra

La muestra es de tipo determinística, conformada por las pruebas de campo del sistema, sobre el cual se realizaron 80 pruebas distribuidas en los 8 diferentes escenarios definidos. Este vehículo fue seleccionado de forma intencional al ser el medio sobre el que se implementó y

validó el sistema desarrollado.

3.1.4 Operacionalización de las Variables

3.1.4.1 Variables Independientes

- Atributos de ocultamiento facial
- Tiempo de permanencia en el rango del sensor

3.1.4.2 Variable Dependiente

- Detección de persona sospechosa
- Generación de alertas

En la Tabla 3.3 se indican las variables dependientes e independientes utilizadas para el análisis de la eficiencia del sistema desarrollado.

Tabla 3.3. Operacionalización de variables

Variables	Concepto	Indicadores	Técnicas e Instrumentos
Independiente			
Atributos de ocultamiento facial	Elementos visuales portados por una persona, como gorra, gafas o mascarilla, detectados por el modelo YOLOv8n.	Número de atributos (0–3)	Modelo YOLOv8n, cámara de vigilancia
Tiempo de permanencia en el rango del sensor	Duración en segundos que una persona permanece dentro del área de detección del sensor.	Segundos (s)	Sensor de distancia, temporizador del sistema
Dependiente			
Detección de persona sospechosa	Identificación automática de una persona con atributos sospechosos o con permanencia prolongada cerca del vehículo.	Porcentaje (%)	Matriz de confusión, pruebas de campo
Generación de alertas	Envío automático de notificaciones al propietario vía SMS y Telegram ante una situación de riesgo detectada.	Porcentaje (%)	Registro de alertas, matriz de pruebas

3.2 Fase 1: Investigación y Análisis

En esta fase se analizaron las tecnologías disponibles para cada componente del sistema, comparando alternativas con el fin de seleccionar las más adecuadas según los requerimientos de procesamiento, consumo energético, precisión y compatibilidad entre sí.

3.2.1 Selección de la Plataforma de Procesamiento

El primer criterio de selección fue determinar qué tipo de plataforma sería capaz de ejecutar un modelo de visión artificial en tiempo real. Como se observa en la Tabla 3.4, se evaluaron

Arduino y Raspberry Pi como opciones principales.

Tabla 3.4. Comparación entre Arduino y Raspberry Pi

Característica	**Arduino Uno**	**Raspberry Pi 5**
Tipo	Microcontrolador	Microcomputadora
Procesador	ATmega328P 16 MHz	ARM Cortex-A76 2.4 GHz
RAM	2 KB SRAM	4 GB / 8 GB
Sistema operativo	No	Sí (Linux/Raspbian)
Multitarea	No	Sí
Visión artificial / IA	No soportado	Sí
Conectividad	USB	Wi-Fi, BT 5.0, Ethernet, USB
Lenguajes	C/C++	Python, C, C++, Java
Seleccionado	No	**Sí**

Arduino, al ser un microcontrolador, carece de la capacidad de ejecutar un sistema operativo, realizar multitarea o procesar modelos de inteligencia artificial, lo que lo hace inviable para un sistema de visión artificial como el propuesto. La Raspberry Pi, en cambio, es una microcomputadora completa capaz de ejecutar Python, OpenCV y modelos YOLO, con conectividad Wi-Fi y Bluetooth integrados, siendo la plataforma adecuada para este proyecto [32].

Una vez decidido que la plataforma sería la Raspberry Pi, se pasó a comparar los distintos modelos disponibles, tal como se muestra en la Tabla 3.5, evaluando en cada caso su capacidad de procesamiento para correr inferencia con YOLOv8n.

Tabla 3.5. Comparación entre modelos de Raspberry Pi

Característica	**Raspberry Pi 3B+**	**Raspberry Pi 4B**	**Raspberry Pi 5**
Procesador	ARM Cortex-A53 1.4 GHz	ARM Cortex-A72 1.8 GHz	ARM Cortex-A76 2.4 GHz
RAM máxima	1 GB	8 GB	8 GB
USB	USB 2.0	USB 3.0	USB 3.0
FPS con YOLO (CPU)	2–4 FPS	5–8 FPS	10–15 FPS
Consumo	5 W	3–7 W	3–5 W
Seleccionado	No	No	**Sí**

La versión 3B+ fue descartada por su limitada RAM de 1 GB y sus bajos FPS para inferencia con YOLO. La versión 4B, aunque funcional, ofrece menor velocidad de procesamiento frente a la versión 5. La Raspberry Pi 5, con su procesador ARM Cortex-A76 a 2.4 GHz, alcanza entre 10 y 15 FPS con YOLOv8n en CPU, con un consumo eficiente de 3 a 5 W, siendo la opción más adecuada para un sistema embebido portátil alimentado por batería [32].

3.2.2 Selección del Modelo de Detección de Objetos

Para la detección de atributos de ocultamiento facial en tiempo real, se evaluaron los tres modelos más utilizados en la literatura, como se muestra en la Tabla 3.6.

Tabla 3.6. Comparación de modelos de detección de objetos

Característica	**YOLOv8n**	**SSD MobileNet**	**Faster R-CNN**
Tipo	Una etapa	Una etapa	Dos etapas
Velocidad	15–25 FPS (CPU)	10–20 FPS (CPU)	1–5 FPS (CPU)
mAP50 (COCO)	37.3 %	23 %	42 %
Parámetros	3.2 M	4.3 M	41.8 M
Detección tiempo real	Sí	Sí	No
Apto para sistemas embebidos	Sí	Sí	No
Transfer learning	Sí	Sí	Sí
Seleccionado	**Sí**	No	No

Faster R-CNN fue descartado por su arquitectura de dos etapas, que lo hace demasiado lento para operar en tiempo real sobre hardware embebido, alcanzando apenas 1 a 5 FPS en CPU [66]. Por su parte, SSD MobileNet, si bien es más ligero, tiene un mAP50 que no supera el 25 %, lo que afecta directamente la precisión al detectar atributos como gorras, gafas o mascarillas. YOLOv8n, en cambio, gracias a su arquitectura de una sola etapa, logra el mejor balance entre velocidad (15-25 FPS en CPU), precisión (mAP50 de 37.3 %) y un tamaño compacto (3.2 millones de parámetros), lo que lo convierte en la opción más adecuada para sistemas embebidos con recursos limitados [6].

3.2.3 Selección del Sensor de Distancia

Para detectar la presencia de personas en el perímetro del vehículo, se compararon tres tipos de sensores según la Tabla 3.7.

Tabla 3.7. Comparación de sensores de detección de presencia

Característica	**VL53L0X (ToF)**	**HC-SR04 (ultrasónico)**	**RCWL-0516 (radar)**
Tecnología	Láser infrarrojo ToF	Ultrasonido	Microondas Doppler
Rango	30–1200 mm	20–4000 mm	hasta 7 m
Precisión	$\pm 3\,\%$	± 3 mm	Sin medición de distancia
Interfaz	I2C	GPIO (trigger/echo)	GPIO (salida digital)
Voltaje	2.8 V – 5 V	5 V	3.3 V / 5 V
Afectado por luz intensa	Sí	No	No
Detección angular	25°	15–30°	360° omnidireccional
Seleccionado	**Sí**	No	No

El sensor RCWL-0516 fue descartado por no proporcionar medición de distancia, lo que impide definir zonas específicas de detección alrededor del vehículo. El HC-SR04, aunque de bajo costo y amplio rango de hasta 4000 mm, opera únicamente a 5 V y su campo de detección puede verse afectado por superficies irregulares u oblicuas [5]. El sensor VL53L0X, basado en tecnología *Time-of-Flight* por láser infrarrojo a 940 nm, opera en un rango de voltaje de 2.8 V a 5 V, se comunica mediante interfaz I2C compatible directamente con la Raspberry Pi 5, y con un rango de hasta 1200 mm permite definir con exactitud el umbral de detección de 1000 mm requerido por el sistema [67].

3.2.4 Selección del Módulo de Comunicación Celular

Para el envío de alertas SMS y la transmisión de fotografías por Telegram, se evaluaron tres módulos de comunicación celular según la Tabla 3.8.

Tabla 3.8. Comparación de módulos de comunicación celular

Característica	**SIM7600G-H**	**SIM800L**	**Módulo Wi-Fi (ESP8266)**
Red soportada	4G LTE / 3G / 2G	2G (GSM/GPRS)	Wi-Fi 802.11 b/g/n
SMS	Sí	Sí	No (requiere gateway)
Envío de imágenes	Sí (4G LTE)	Limitado (GPRS lento)	Sí (con Wi-Fi)
Cobertura	Global multibanda	Solo redes 2G	Solo en rango Wi-Fi
Interfaz	USB / UART	UART	UART / SPI
Voltaje	5 V	3.4–4.4 V	3.3 V
Comandos AT	Sí	Sí	No
Independiente de red local	Sí	Sí	No
Seleccionado	**Sí**	No	No

El módulo Wi-Fi ESP8266 fue descartado porque su funcionamiento depende de una red Wi-Fi disponible en el entorno del vehículo, lo que lo hace inviable para un sistema de vigilancia en zonas urbanas sin cobertura Wi-Fi garantizada. Además, no soporta el envío de SMS de forma nativa [61]. El módulo SIM800L, aunque de bajo costo, opera únicamente en redes 2G (GSM/GPRS), que están siendo progresivamente desactivadas en muchos países, incluyendo Ecuador, y su velocidad de datos GPRS es insuficiente para la transmisión de fotografías en tiempo real [61]. El SIM7600G-H, en cambio, soporta redes 4G LTE, 3G y 2G de forma simultánea, garantizando cobertura en cualquier zona donde exista señal celular, permite el envío de SMS mediante comandos AT y la transmisión de fotografías e imágenes a alta velocidad por la red 4G, y se conecta directamente a la Raspberry Pi 5 mediante USB sin necesidad de circuitos adicionales [61].

3.3 Fase 2: Diseño del Sistema

3.3.1 Descripción General del Sistema

El sistema presentado es un dispositivo de vigilancia autónoma diseñado para monitorear el perímetro de un vehículo estacionado mediante cuatro zonas de detección independientes.

Cada zona combina un sensor de distancia, una cámara y un módulo de procesamiento basado en IA que analiza en tiempo real si la persona detectada presenta atributos asociados a comportamiento sospechoso, como el uso simultáneo de gorra, gafas de sol o mascarilla.

Cuando el sistema detecta una condición de riesgo, activa al mismo tiempo tres canales de notificación: envía un SMS a través de la red celular, manda una alerta multimedia con la fotografía del evento al bot de Telegram del usuario, y guarda esa misma imagen en una memoria USB conectada de forma local.

En cuanto a su estructura, el sistema se organiza en tres capas que trabajan de forma coordinada:

- **Capa de percepción:** formada por los sensores VL53L0X y las cámaras, responsables de recolectar los datos del entorno.

- **Capa de procesamiento:** a cargo de la Raspberry Pi 5, donde se ejecuta el modelo de detección YOLOv8n.

- **Capa de telecomunicaciones:** integrada por el módulo SIM7600G-H y el bot de Telegram, responsables de enviar las alertas al usuario.

3.3.2 Componentes de Hardware

Para la construcción del prototipo se seleccionaron los siguientes componentes, que se observan en la Tabla 3.9, cada uno elegido en función de su compatibilidad con la Raspberry Pi 5 y los requerimientos del sistema:

Tabla 3.9. Componentes utilizados en el hardware.

Componente	**Cantidad**	**Función**
Raspberry Pi 5 (8GB RAM)	1	Unidad central de procesamiento
Pi Camera Module 3	2	Cámaras CSI para zonas Z0 y Z1
PS Eye (USB)	1	Cámara USB para zona Z2
Fly Cam (USB)	1	Cámara USB para zona Z3
Sensor VL53L0X	4	Detección de presencia por zona
Módulo SIM7600G-H	1	Comunicación celular GSM/LTE
Powerbank INIU 65W	1	Alimentación portátil del sistema
Memoria USB	1	Almacenamiento local de alertas
Baquelita perforada y cables jumper	-	Conexiones entre componentes

3.3.3 Diagrama de Bloques del Sistema

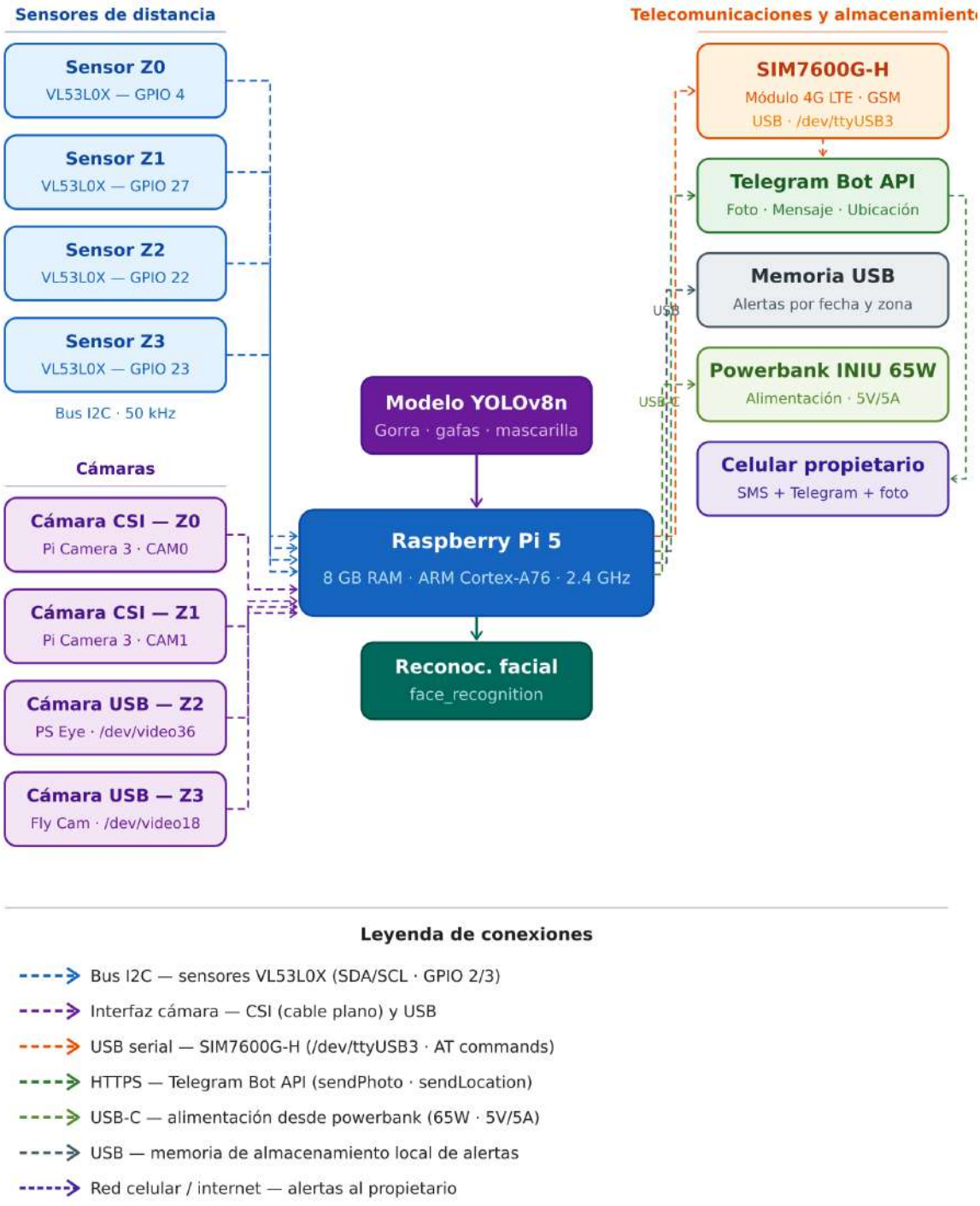


Figura. 3.12. Diagrama de bloques del sistema propuesto.

La Figura 3.12 ilustra cómo están conectados y organizados todos los componentes del sistema, con la Raspberry Pi 5 como núcleo central que coordina todo lo que ocurre. En el lado izquierdo se encuentran los sensores y las cámaras, que son los ojos y los oídos del sistema. Los cuatro sensores de distancia VL53L0X están ubicados uno en cada zona del

vehículo y se comunican con la Raspberry Pi a través del bus I2C, que es básicamente un canal compartido por el que todos los sensores envían sus mediciones. Cada sensor tiene además un pin de control individual que permite encenderlo y apagarlo de forma independiente, lo que es necesario para que no se confundan entre sí al compartir la misma línea de comunicación.

Las cuatro cámaras funcionan en conjunto con cada sensor correspondiente. Las dos cámaras frontales (Z0 y Z1) son modelos Pi Camera 3, conectadas directamente a la Raspberry Pi mediante cable plano a través de los puertos CSI, lo que les permite transmitir video de alta calidad. Las otras dos (Z2 y Z3), en cambio, son cámaras USB que se conectan como cualquier dispositivo externo, lo que facilita bastante su instalación en las zonas más alejadas del vehículo.

En el centro del diagrama está la Raspberry Pi 5 con 8 GB de memoria RAM, que recibe la información de todos los sensores y cámaras, ejecuta el modelo de inteligencia artificial YOLOv8n para analizar si la persona que se acerca lleva accesorios que oculten su rostro, y también corre el módulo de reconocimiento facial para verificar si quien se acerca es el propietario del vehículo.

Del lado derecho se encuentran los componentes encargados de la comunicación y el almacenamiento. El módulo SIM7600G-H se encarga de la conectividad celular: utiliza la red GSM para enviar el SMS de alerta, y la red 4G LTE para enviar, a través de Telegram, la fotografía del intruso junto con la ubicación del vehículo. Por su parte, la memoria USB va guardando una copia local de cada alerta generada, organizada por fecha y zona. La fuente de alimentación portátil de 65W es la que alimenta todo el sistema de forma independiente, sin necesidad de recurrir a la batería del vehículo. Y finalmente, el celular del propietario es el punto de llegada de todas las notificaciones, donde recibe tanto el SMS como el mensaje de Telegram con la fotografía correspondiente.

3.3.4 Secuencia de Procesos del Sistema

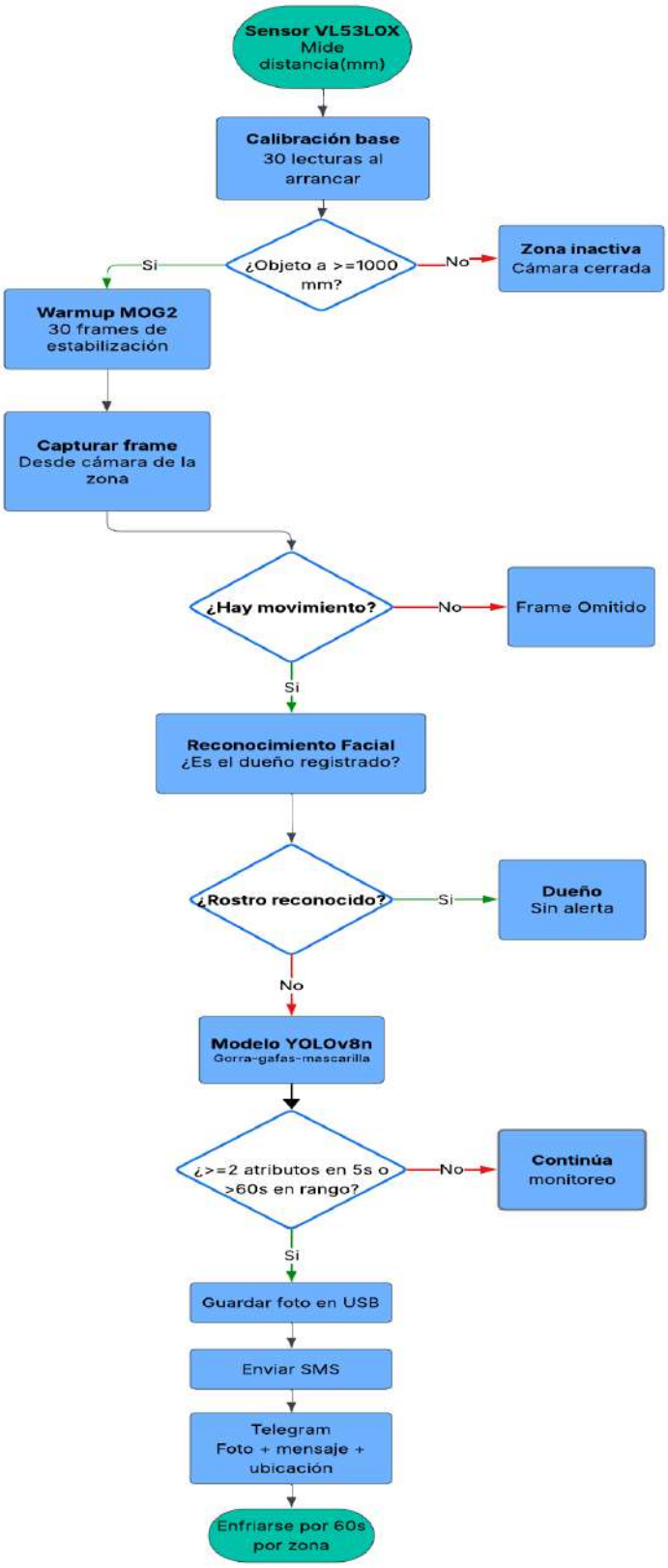


Figura. 3.13. Procesos del funcionamiento del sistema propuesto.

La Figura 3.13 presenta el diagrama de flujo del sistema, el cual describe el proceso completo que sigue el sistema desde que detecta algo hasta que decide si debe o no enviar una alerta al propietario.

El proceso arranca con el sensor VL53L0X, que se encarga de medir de forma constante la distancia a los objetos que tiene frente a sí. Al momento de encender el sistema, este realiza una calibración inicial tomando 30 lecturas, con el fin de establecer cuál es la distancia normal de la escena. A partir de ahí, el sistema empieza a evaluar si algún objeto o persona se ha acercado a menos de 1000 mm del vehículo. Mientras no haya nadie cerca, la cámara se mantiene apagada, lo que ayuda a ahorrar recursos.

Cuando el sensor detecta que alguien está dentro del rango, se activa la cámara de esa zona, la cual pasa por un período de estabilización de 30 frames antes de empezar el análisis. Una vez que ya está estable, el sistema empieza a capturar frames de forma continua y revisa si hay movimiento en la imagen. Si no detecta movimiento, simplemente descarta ese frame y espera el siguiente, evitando así hacer análisis innecesarios.

Cuando se detecta movimiento, lo primero que hace el sistema es tratar de identificar si la persona que está frente a la cámara es el propietario del vehículo. Si lo reconoce, todo sigue su curso normal y no se genera ninguna alerta. Pero si el rostro no coincide con el del propietario, entra en juego el modelo YOLOv8n, que se encarga de revisar si la persona lleva alguna combinación de gorra, gafas o mascarilla.

A partir de ese momento, hay dos cosas que pueden hacer que se dispare la alerta: que el sistema detecte al menos dos de esos atributos en un lapso de 5 segundos, o que la persona se quede más de 60 segundos dentro del rango del sensor sin que logre identificarla. Si pasa cualquiera de las dos cosas, se pone en marcha toda la secuencia de alerta: se toma una foto y se guarda en la memoria USB, se manda un SMS al propietario, y por Telegram se envía esa misma foto junto con un mensaje y la ubicación aproximada del vehículo.

Una vez enviada la alerta, el sistema espera 60 segundos antes de poder generar una nueva para esa misma zona. Esto se hace justamente para evitar que un mismo evento termine disparando varias notificaciones seguidas.

3.4 Fase 3: Desarrollo e Implementación

3.4.1 Configuración del Hardware

Conexión de Sensores VL53L0X

Los cuatro sensores de distancia se conectan al bus I2C de la Raspberry Pi 5 (pines SDA/SCL,

GPIO 2 y GPIO 3). Dado que todos los sensores VL53L0X tienen la misma dirección I2C por defecto $(0x29)$, fue necesario asignarles direcciones únicas mediante el pin XSHUT de cada sensor. Se utilizaron los pines GPIO 4, 27, 22 y 23 como líneas XSHUT para las zonas Z0, Z1, Z2 y Z3 respectivamente. El proceso de inicialización apaga todos los sensores al mismo tiempo y luego los enciende uno por uno, asignando a cada uno su propia dirección I2C antes de encender el siguiente. Las direcciones I2C asignadas son $0x30$, $0x31$, $0x32$ y $0x33$, respectivamente. Para garantizar la estabilidad de la comunicación I2C, se redujo la velocidad del bus a $50,000$ Hz mediante el parámetro $dtparam = i2c\_arm\_baudrate = 50000$ en el archivo de configuración del sistema, ya que a la velocidad estándar de $100,000$ Hz algunos sensores presentaban errores de lectura intermitentes.

Circuito de acondicionamiento del sensor VL53L0X

Dado que el sensor VL53L0X opera con el láser VCSEL interno que genera picos de corriente al disparar, se integraron tres componentes pasivos por cada sensor para garantizar la estabilidad de la comunicación I2C y proteger la línea de alimentación, como se ilustra en la Figura 3.14.

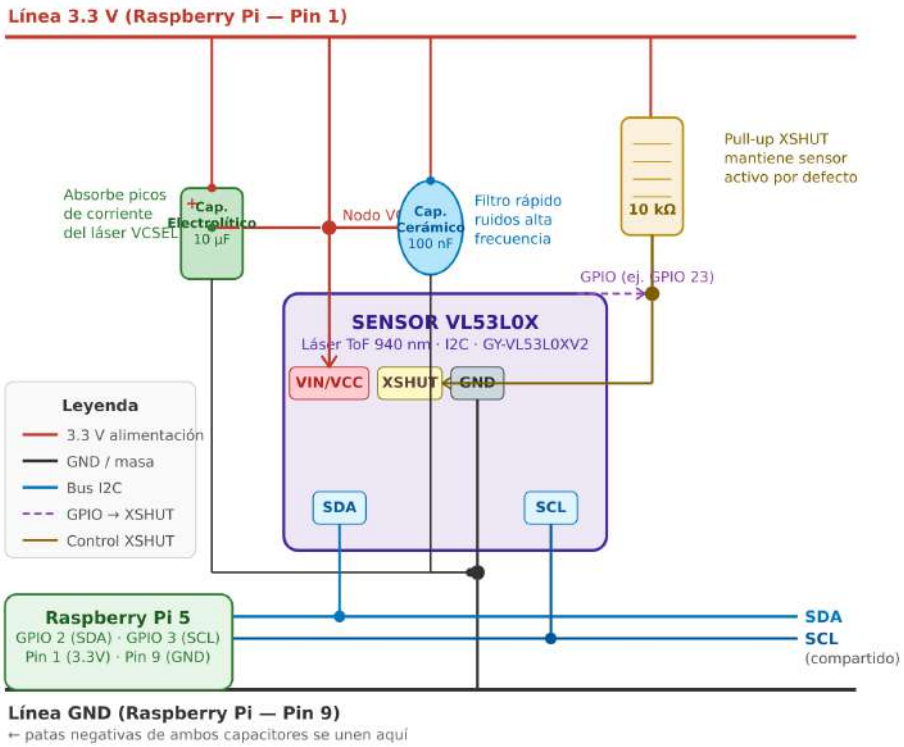


Figura. 3.14. Circuito de acondicionamiento del sensor VL53L0X.

La Figura 3.14 muestra el esquema completo de conexión para uno de los sensores. En la parte de arriba va la línea de 3.3 V que sale del Pin 1 de la Raspberry Pi 5, y abajo corre la

línea común de GND, conectada al Pin 9. En el centro del esquema está el sensor VL53L0X, que tiene cinco pines importantes para esta conexión: VIN/VCC y GND para la alimentación, XSHUT para controlar el encendido de forma individual, y SDA/SCL para la comunicación vía I2C.

Justo a la izquierda del nodo VCC aparece un capacitor electrolítico de 10 µF, con su terminal positiva conectada a la línea de 3.3 V y la negativa a GND, formando así un puente en paralelo entre ambas líneas, justo antes de llegar al pin VIN del sensor. A su derecha se encuentra el capacitor cerámico de 100 nF, conectado de la misma manera, también en paralelo entre VCC y GND. Como ambos capacitores comparten el mismo nodo VCC que alimenta al pin VIN/VCC del sensor, cualquier pico de corriente o ruido que pueda aparecer en la línea de alimentación queda filtrado o absorbido antes de alcanzar el sensor.

En la columna derecha del diagrama se muestra la resistencia pull-up de 10 kΩ, cuya pata superior está conectada a la línea de 3.3 V y cuya pata inferior llega al nodo XSHUT. En ese mismo nodo confluyen tres conexiones: la resistencia pull-up, el cable proveniente del GPIO de la Raspberry Pi (representado con línea punteada) y el pin XSHUT del sensor. Esto permite que, por defecto, el pin XSHUT esté en nivel alto y el sensor permanezca activo, mientras que cuando la Raspberry Pi lleva el GPIO a nivel bajo, el sensor se apaga y puede ser reiniciado con una nueva dirección I2C.

En la parte inferior de la Figura 3.14 se representan las líneas del bus I2C, SDA (GPIO 2) y SCL (GPIO 3), como líneas horizontales compartidas que reciben las conexiones de los pines SDA y SCL del sensor. Esto indica que el mismo esquema se repite para los cuatro sensores del sistema, todos conectados a ese mismo bus.

Los componentes utilizados por cada sensor fueron los siguientes:

- **Resistencia pull-up de 10 kΩ:** conectada entre el pin XSHUT del sensor y la línea de 3.3 V. Su trabajo es mantener el pin XSHUT en estado alto por defecto, para que el sensor se quede activo. Pero cuando la Raspberry Pi pone en nivel bajo el GPIO correspondiente, el sensor se apaga, y eso es justo lo que permite asignarle una dirección I2C única durante la inicialización.

- **Capacitor electrolítico de 10 µF:** conectado en paralelo entre los pines VIN/VCC y GND del sensor, lo más cerca posible del encapsulado. Como es un componente polarizado, su pata positiva tiene que ir hacia VCC. Su trabajo es absorber los picos de corriente que se producen cuando dispara el láser VCSEL, evitando así caídas de tensión que podrían causar reinicios o errores en el bus I2C.

- **Capacitor cerámico de 100 nF (código 104):** conectado también en paralelo entre VCC y GND, al lado del capacitor electrolítico. Como no tiene polaridad y reacciona

más rápido ante los cambios de frecuencia, este componente actúa como un filtro de alta frecuencia, eliminando el ruido e interferencias que las pistas de la baquelita perforada pueden meter en la alimentación del sensor.

Conexión de Cámaras

Las cámaras CSI (Pi Camera Module 3) se conectaron directamente a los puertos CAM0 y CAM1 de la Raspberry Pi 5 a través de cables planos, mientras que las cámaras USB se conectaron a los puertos USB 3.0 de la placa. Para poder identificar correctamente el índice de dispositivo de cada cámara USB, se utiliza el comando $v4l2 - ctl \ -- list - devices$, que revela que la PS Eye ocupa el nodo /dev/video36 y la Fly Cam el nodo /dev/video1.

Conexión del Módulo SIM7600G-H

El módulo de comunicación celular se conectó a través de su puerto USB integrado al puerto USB de la Raspberry Pi 5, y el sistema lo reconoció automáticamente como /dev/ttyUSB3. Se le colocó una tarjeta SIM con plan de datos y SMS activo, y en cuanto a la alimentación, esta se obtiene directamente del puerto de salida USB de la Raspberry Pi 5.

Tabla de Conexiones de Pines

En la Tabla 3.10 se resumen las conexiones físicas entre cada componente del sistema y la Raspberry Pi 5. Los cuatro sensores VL53L0X comparten un mismo bus I2C a través de los pines SDA y SCL, mientras que sus pines XSHUT individuales son los que permiten asignarles direcciones I2C distintas durante la inicialización. Por su parte, las cámaras CSI se conectan mediante cable plano a los puertos CAM0 y CAM1, mientras que las cámaras USB y el módulo SIM7600G-H hacen uso de los puertos USB 3.0 disponibles en la placa. En cuanto a la alimentación, esta proviene de un powerbank de 65W conectado al puerto USB-C de la Raspberry Pi 5.

Tabla 3.10. Tabla de conexiones entre componentes y Raspberry Pi 5

Componente	**Pin componente**	**Pin Raspberry Pi 5**	**Protocolo**
VL53L0X (todos)	VCC	Pin 1 (3.3V)	Alimentación
	GND	Pin 9 (GND)	Tierra
	SDA	Pin 3 (GPIO2)	I2C
	SCL	Pin 5 (GPIO3)	I2C
Sensor Z0 (XSHUT)	XSHUT	Pin 7 (GPIO4)	Digital OUT
Sensor Z1 (XSHUT)	XSHUT	Pin 13 (GPIO27)	Digital OUT
Sensor Z2 (XSHUT)	XSHUT	Pin 15 (GPIO22)	Digital OUT
Sensor Z3 (XSHUT)	XSHUT	Pin 16 (GPIO23)	Digital OUT
Pi Camera 3 (Z0)	Cable plano	Puerto CAM0	CSI-2
Pi Camera 3 (Z1)	Cable plano	Puerto CAM1	CSI-2
PS Eye (Z2)	USB-A	Puerto USB 3.0	USB
Fly Cam (Z3)	USB-A	Puerto USB 3.0	USB
SIM7600G-H	USB-A	Puerto USB 3.0	USB serial
Memoria USB	USB-A	Puerto USB 3.0	USB
Powerbank 65W	USB-C	Puerto USB-C	Alimentación

3.4.2 Configuración de Montaje

La Figura 3.15 muestra cómo quedó distribuido físicamente el sistema dentro y sobre el vehículo. La idea principal fue cubrir los cuatro costados del automóvil para que ningún punto quedara sin vigilancia. En cada lado del vehículo se instaló una cámara acompañada de su sensor: la zona frontal (Z0) y la trasera (Z1) cuentan con las cámaras Pi Camera 3 conectadas por cable plano, aprovechando los parabrisas delantero y trasero como punto de montaje. En el techo, hacia los laterales, se instalaron la Fly Cam en la zona izquierda (Z3) y la PS Eye en la zona derecha (Z2), cubriendo así los flancos del vehículo. Cada cámara tiene justo al lado su sensor de distancia, de modo que cuando alguien se acerca a menos de un metro por cualquier lado, la cámara de esa zona se activa de inmediato.

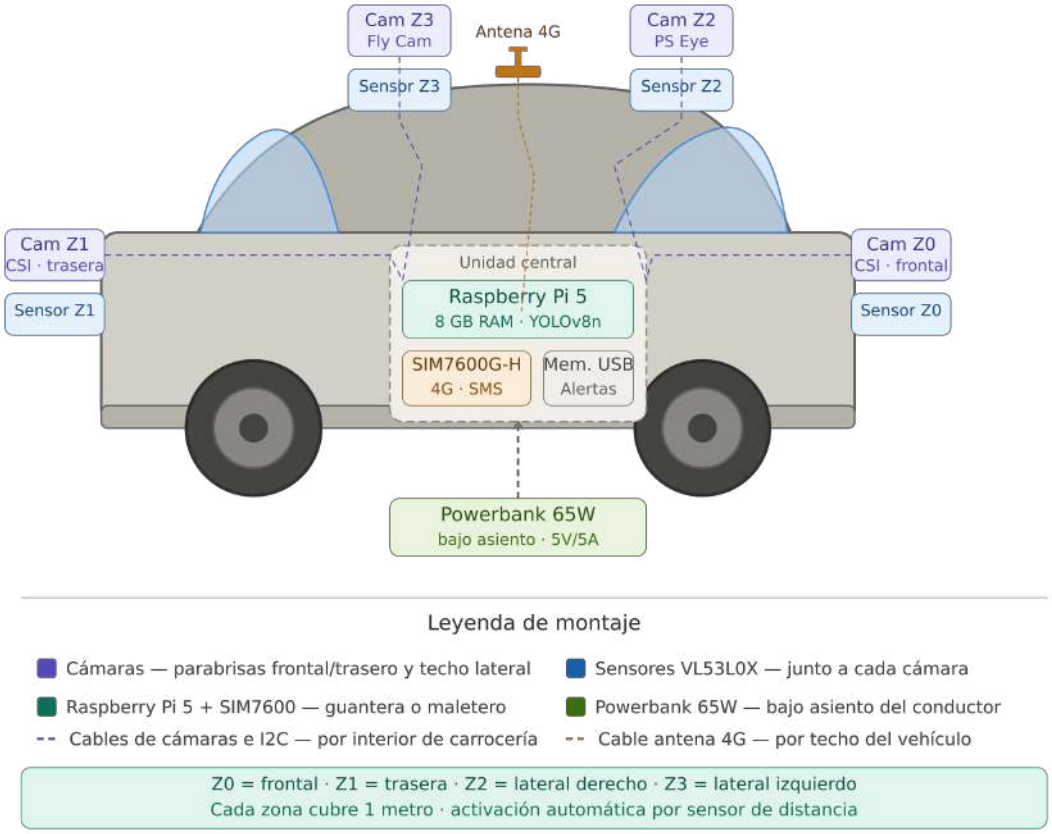


Figura. 3.15. Distribución e instalación de los componentes del sistema sobre el vehículo de prueba.

3.4.3 Prototipo Instalado

La Figura 3.23 presenta el prototipo instalado en el vehículo de prueba, mostrando cómo quedó cada componente ubicado.
La Figura 3.16 muestra la Raspberry Pi 5 dentro de su carcasa protectora, junto con los cables de conexión correspondientes. Se decidió instalarla en la guantera del vehículo, de manera que quedara fuera de la vista y protegida ante posibles golpes, sin dejar de tener acceso a los puertos USB donde se conectan tanto las cámaras como el módulo de comunicación. En la Figura 3.17 se aprecia el módulo SIM7600G-H ubicado en el parabrisas delantero del vehículo. Esta posición fue elegida deliberadamente para que la antena 4G tenga visibilidad

directa al exterior y así garantizar la mejor recepción de señal celular al momento de enviar las alertas.

Las Figuras 3.18 y 3.19 muestran la instalación de las cámaras junto a sus respectivos sensores de distancia en las zonas delantera y lateral del vehículo. Se puede observar cómo cada cámara y su sensor trabajan en conjunto: el sensor detecta si alguien se acerca y la cámara, ubicada justo al lado, se activa para capturar lo que ocurre. Las Figuras 3.20 y 3.21 muestran en detalle los sensores VL53L0X instalados en las zonas delantera y trasera respectivamente. Cada sensor fue fijado en un soporte impreso en 3D que lo mantiene orientado hacia el exterior del vehículo, apuntando al área que debe vigilar. Finalmente, la Figura 3.22 ilustra la cámara Fly Cam correspondiente a la zona Z2, instalada en la luneta trasera del vehículo.

En conjunto, los componentes del sistema se instalaron de forma provisional en el vehículo de prueba, dándole más importancia a que todo funcionara bien y a que las cuatro zonas quedaran cubiertas, que a cómo se viera el montaje. Al final, lo único que se buscaba con esta instalación era comprobar que el sistema funcionara como se esperaba en condiciones reales de campo. Ya pensando en una versión final del producto, lo ideal sería trabajar en el diseño de los soportes y en cómo integrar los componentes a la carrocería, para que la instalación pueda ser permanente, más compacta y discreta, con elementos que casi no se noten desde fuera.

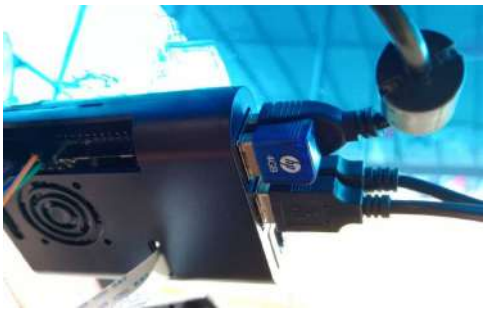


Figura. 3.16. Raspberry Pi 5 instalada en el vehículo.

Figura. 3.17. Módulo SIM7600G-H (4G Dongle).

Figura. 3.18. Cámara y sensor zona delantera.

Figura. 3.19. Cámara y sensor zona lateral.

Figura. 3.20. Sensor VL53L0X zona delantera.

Figura. 3.21. Sensor VL53L0X zona trasera.

Figura. 3.22. Fly Cam (Z3) instalada en luneta trasera.

Figura. 3.23. Componentes del prototipo instalados en el vehículo.

3.4.4 Configuración del Software

Sistema Operativo y Dependencias
Se utilizó Raspberry Pi OS de 64 bits como sistema operativo base. Las principales librerías instaladas fueron las siguientes que se pueden observar en la Tabla 3.11:

Tabla 3.11. Librerías instaladas.

Librería	**Función**
ultralytics	Modelo YOLOv8n
opencv-python	Captura y procesamiento de imágenes
adafruit-circuitpython-vl53l0x	Lectura de sensores
face-recognition	Identificación del propietario
requests	Comunicación con Telegram Bot API
pyserial	Comunicación con SIM7600G-H

Habilitación de Interfaces
Las interfaces I2C y de las cámaras se habilitan mediante la herramienta $raspi-config$. Se configuró el bus I2C a $50,000$ Hz para garantizar la estabilidad de los sensores y se verificó la correcta detección de todos los dispositivos mediante el comando $i2cdetect-y1$.

3.4.5 Construcción del Dataset y Entrenamiento del Modelo

Recolección y Etiquetado de Imágenes
Se armó un dataset con cinco clases, pensadas para identificar atributos relacionados con el ocultamiento de identidad: gorra, gafas de sol, mascarilla, persona y null. Las imágenes se recopilaron de distintas fuentes públicas en internet y se etiquetaron a mano con ayuda de la plataforma Roboflow, que permite dibujar cajas alrededor de cada objeto y exportar las anotaciones en formato YOLO.

El dataset final quedó conformado por 2547 imágenes. Esto se logró gracias a la herramienta Augmentations de la plataforma Roboflow, la cual amplió el conjunto original variando orientación, brillo y desenfoque en las imágenes existentes. Del total, un 88 % se destinó a entrenamiento, 7 % a validación y 5 % a pruebas, como se observa en la Figura 3.24.

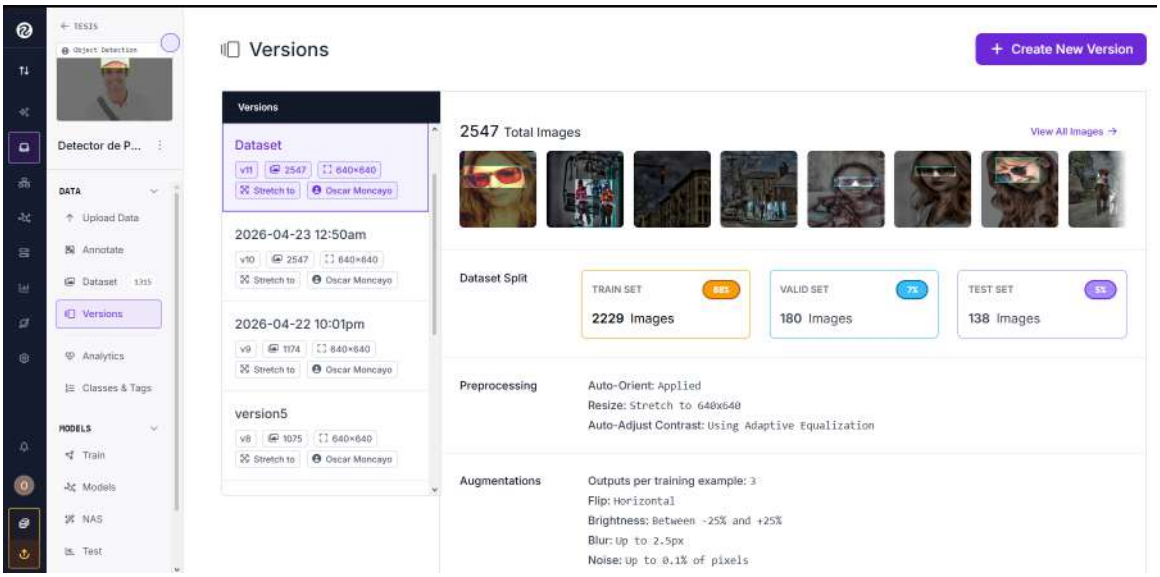


Figura. 3.24. Detalles del dataset en la plataforma Roboflow.

Entrenamiento en Google Colab

El entrenamiento se realizó en Google Colaboratory utilizando una GPU NVIDIA T4 proporcionada gratuitamente por la plataforma. Se partió del modelo yolov8n.pt, cuyos pesos fueron obtenidos del entrenamiento previo sobre el dataset MS COCO (Microsoft Common Objects in Context), un conjunto de datos público con más de 330,000 imágenes y 80 categorías de objetos cotidianos [52]. Aplicando transfer learning, se reutilizaron estos pesos como punto de partida y se reentrenó únicamente la cabeza de detección del modelo para adaptarla a las clases del sistema: gorra, gafas, mascarilla y persona. Los parámetros que se tomaron en cuenta se lo puede observar en la Tabla 3.12.

Tabla 3.12. Parámetros principales para el entrenamiento.

Parámetro	**Valor**	**Justificación**
Épocas	200	Completar todas sin early stopping
Batch size	32	Óptimo para GPU T4
Resolución	640×640 px	Estándar YOLOv8
Optimizador	AdamW	Mejor convergencia en datasets pequeños
patience	0	Deshabilita early stopping
cls (peso clasificación)	2.0	Mejora distinción entre clases similares
Modelo base	Yolov8n (MS COCO)	Transfer learning desde 80 clases
Augmentación	mosaic, erasing, mixup	Compensa desbalance de clases

Se aplicaron técnicas de augmentación agresiva incluyendo variaciones de brillo, saturación,

rotación, perspectiva y erasing (borrado aleatorio de regiones) para compensar la diferencia en cantidad de imágenes entre clases y mejorar la generalización a condiciones reales de iluminación de cámara.

3.4.6 Arquitectura del Software

El software del sistema se implementó en Python 3 siguiendo una arquitectura de hilos concurrentes. El hilo principal se encarga exclusivamente de mostrar las imágenes en pantalla mediante OpenCV, mientras que cada zona de vigilancia opera en su propio hilo independiente con la siguiente lógica que se observa en la Figura 3.25.

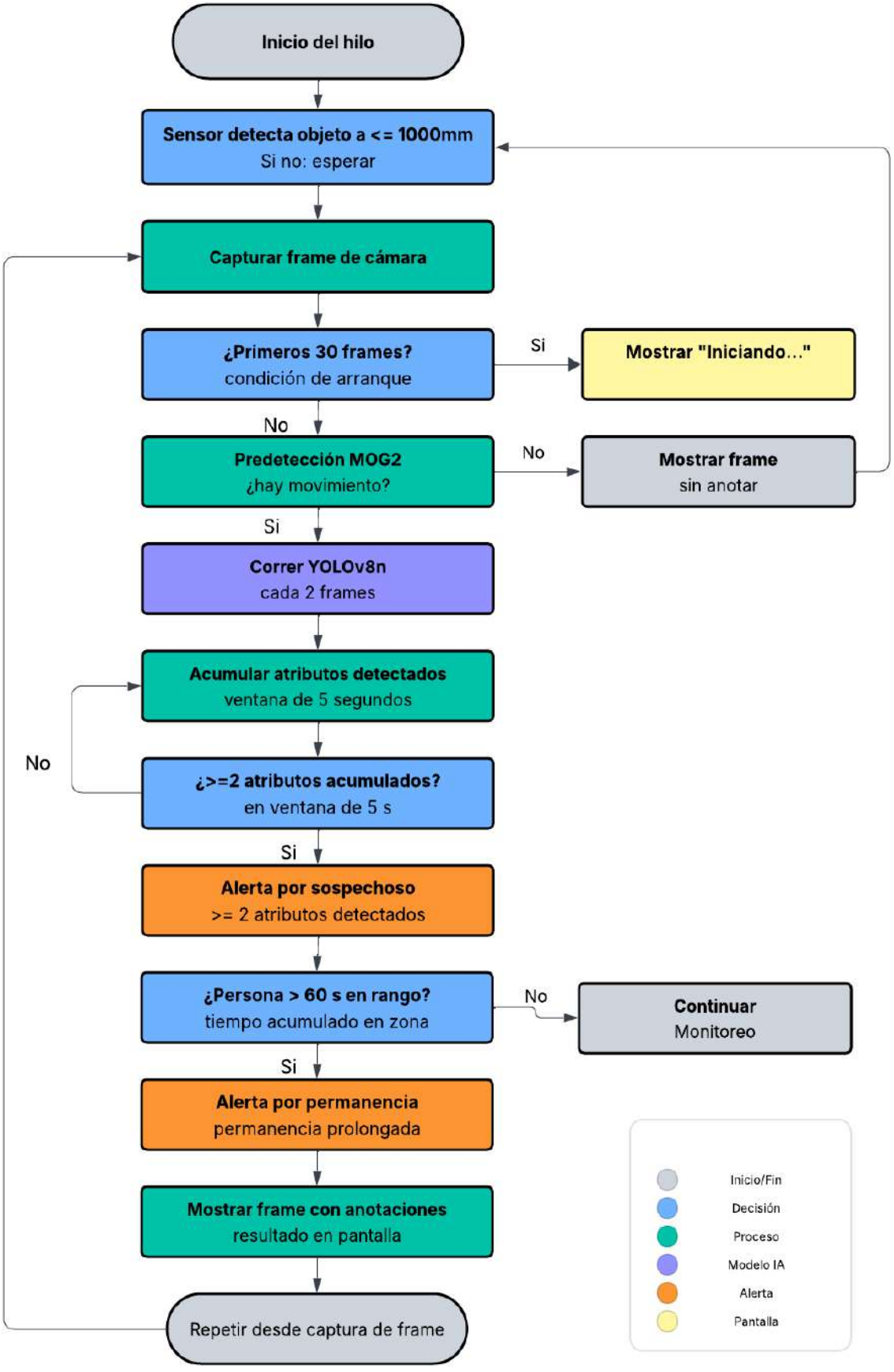


Figura. 3.25. Diagrama de flujo por zona.

Lógica de Detección
El sistema implementa dos mecanismos de alerta independientes:

- **Alerta por atributos:** se acumulan los atributos detectados que son gorra, gafas y mascarilla, durante un intervalo de 5 segundos. Si dentro de ese período aparecieron dos o más atributos distintos, aunque no hayan sido simultáneos, se considera la situación como sospechosa y se activa la alerta. Este planteamiento resuelve el problema de que el modelo no siempre detecta todos los atributos en el mismo frame.

- **Alerta por permanencia:** si una persona se queda dentro de un rango de 1 metro por más de 60 segundos y el sistema no logra reconocerla como el propietario registrado, se envía la alerta de todas formas, sin importar qué atributos se hayan detectado o no.

Sistema de Alertas
Cuando se activa cualquiera de los dos mecanismos de alerta, el sistema ejecuta simultáneamente en hilos independientes:

1. Guardado de la fotografía en la memoria USB organizada por fecha $/alertas/YYYY-MM-DD/alerta\_Zn\_HH-MM-SS.jpg$.

2. Envío de un SMS al número del propietario mediante comandos AT.

3. Envío de la fotografía junto con un mensaje al bot de Telegram.

4. Envío de la ubicación aproximada, obtenida por geolocalización IP, también al bot de Telegram.

3.5 Fase 4: Validación del Sistema

3.5.1 Prueba de Sensores

La Figura 3.26 presenta los resultados de la prueba de inicialización y lectura de los cuatro sensores de distancia del sistema. Esta prueba se ejecutó mediante un script de diagnóstico que verifica uno por uno que cada sensor esté correctamente conectado y funcionando antes de arrancar el sistema principal.

En la primera parte de la prueba, el sistema apaga todos los sensores al mismo tiempo y, después, los va encendiendo uno por uno para asignarle a cada cual una dirección única de comunicación. Los cuatro sensores respondieron correctamente durante este proceso, lo que confirmó que tanto el cableado como la configuración estaban bien hechos.

Después se tomaron 10 lecturas seguidas de distancia en cada sensor, para comprobar que las mediciones fueran estables y consistentes. El sensor Z0 mostró un comportamiento bastante bueno, con lecturas que se ubicaron entre 209 mm y 234 mm, un promedio de 226 mm y muy poca variación entre una medición y otra, lo que sugiere que estaba apuntando a un objeto fijo y cercano dentro del vehículo.

Sin embargo, los sensores Z1 y Z3 reportaron constantemente el valor de 8190 mm, que es la lectura máxima que entrega el sensor cuando no detecta ningún objeto dentro de su rango. Esto significa que en el momento de la prueba esos sensores estaban apuntando al vacío, probablemente porque no había ningún objeto a menos de 1,2 metros frente a ellos, o porque su orientación de montaje aún necesitaba ajuste. Este comportamiento es normal durante las pruebas en banco y no indica un fallo del sensor, sino simplemente que no había nada que medir en esa dirección en ese momento.

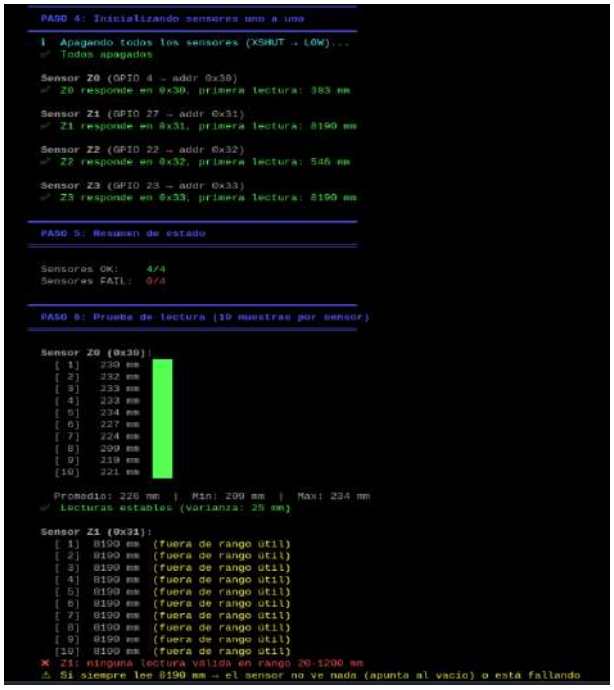


Figura. 3.26. Control del funcionamiento correcto de los sensores.

3.5.2 Prueba de Cámaras

La correcta activación y el flujo de video de las cuatro cámaras se comprobaron mediante un script independiente que muestra en tiempo real la captura de cada dispositivo. Este programa de diagnóstico permitió revisar, de forma simultánea, la estabilidad de la transmisión, verificando parámetros clave como la resolución de imagen, la tasa de cuadros por segundo (FPS) y el tipo de conexión física utilizado, ya sea mediante la interfaz nativa CSI o a través de puertos USB.

En la Figura 3.27 se puede observar la ventana de visualización del sistema con dos cámaras activas simultáneamente. Sobre cada persona detectada aparece un rectángulo de color turquesa con un número decimal, en este caso 0.90 y 0.87, que representa el nivel de confianza con el que el modelo YOLOv8n identificó que hay una persona en ese lugar de la imagen. Dicho valor va de 0 a 1, donde un valor cercano a 1 significa que el modelo está muy seguro de lo que está viendo. En otras palabras, cuando aparece 0.90 significa que el modelo tiene un 90 % de certeza de que lo que está encuadrado en ese rectángulo es una persona, y con 0.87 tiene un 87 % de certeza. Valores por debajo de 0.5 son descartados automáticamente por el sistema para evitar falsas detecciones.

Esta visualización en tiempo real fue de mucha ayuda durante las pruebas, ya que permitió confirmar que el modelo estaba detectando bien a las personas que se acercaban al vehículo, y también sirvió para ir ajustando el ángulo y la posición de cada cámara hasta encontrar el encuadre más adecuado para cada zona.

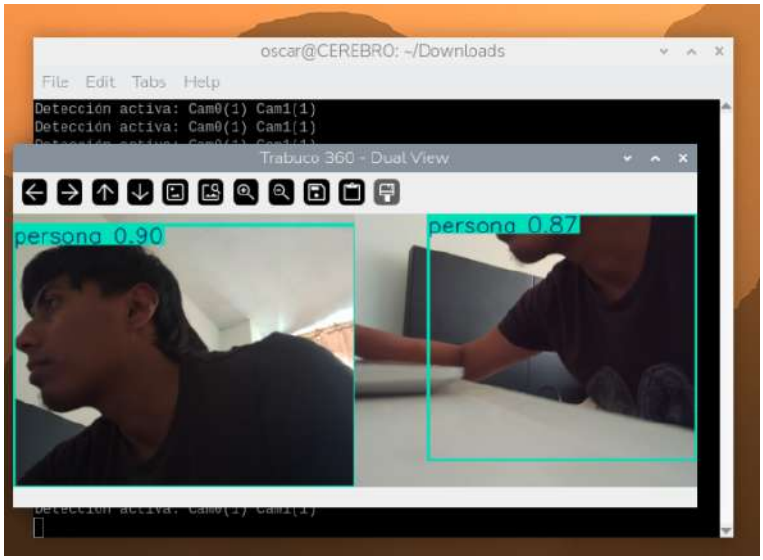


Figura. 3.27. Control de cámaras.

3.5.3 Prueba de Detección

Se realizaron pruebas de detección en condiciones controladas, mostrando a las cámaras personas con y sin atributos de ocultamiento de identidad. De esta forma, se pudo verificar que el sistema generaba alertas únicamente cuando se cumplían las condiciones establecidas, y que el tiempo de espera de 60 segundos entre alertas funcionaba tal como se esperaba.

La Figura 3.28 muestra un ejemplo real de detección durante las pruebas. En la imagen se puede ver a una persona que lleva puesta una gorra y una mascarilla, y el sistema identificó ambos accesorios correctamente. Sobre cada uno aparece su nombre junto a un número: *gorra*

0.28 y *mascarilla 0.69*. Estos números indican qué tan seguro está el modelo de lo que está viendo. En este caso, la mascarilla fue detectada con un 69 % de confianza, lo que es suficiente para que el sistema la considere válida, mientras que la gorra fue detectada con un 28 % de confianza, un valor más bajo pero que igualmente superó el umbral mínimo configurado. Esto refleja una situación real del sistema: los accesorios que cubren más área del rostro, como la mascarilla, tienden a ser detectados con mayor certeza que los que tienen formas más variables, como las gorras.

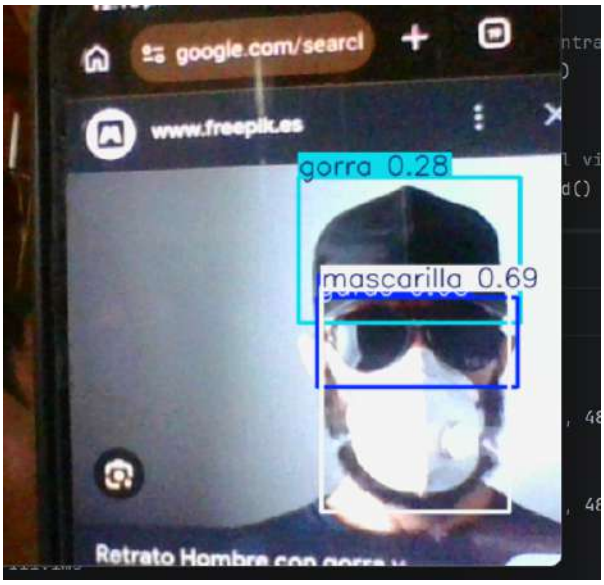


Figura. 3.28. Constancia de los atributos detectados.

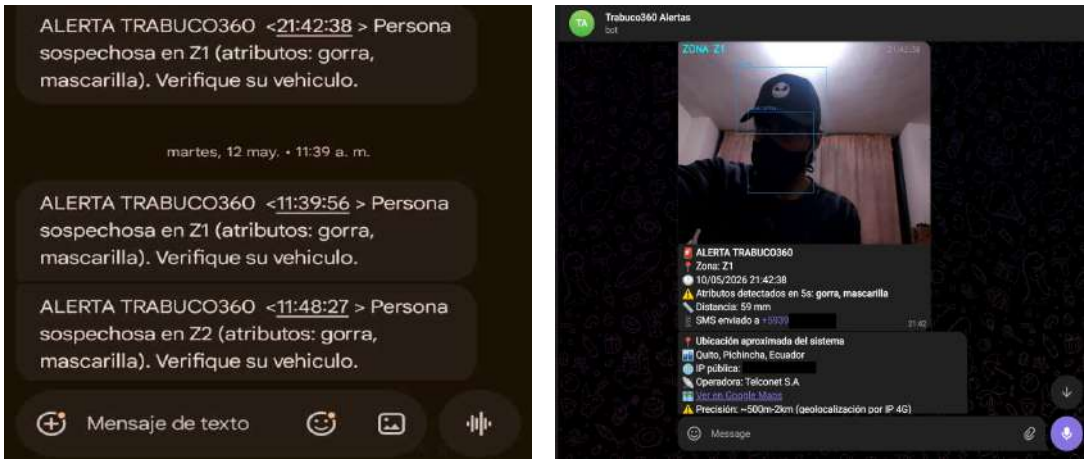
3.5.4 Prueba de Alertas

Se verificó el envío correcto de SMS y mensajes de Telegram con fotografía y ubicación. Se comprobó que las imágenes quedaban guardadas correctamente en la memoria USB organizadas por fecha y zona.

La Figura 3.29(a) muestra los mensajes de texto que el propietario fue recibiendo en su teléfono durante las pruebas del sistema. En cada mensaje se indica la hora exacta en que se detectó la situación, la zona donde ocurrió y los atributos que el sistema logró identificar. Por ejemplo, el mensaje enviado el martes 12 de mayo a las 11:39 señala que en la Zona 1 se detectó a una persona con gorra y mascarilla, pidiéndole al propietario que verifique su vehículo. Esto confirma que el canal de alerta por SMS funciona correctamente y le hace llegar al destinatario la información necesaria de forma clara y directa.

La Figura 3.29(b) muestra la notificación recibida en el bot de Telegram del propietario, donde además del mensaje de alerta se incluye la fotografía capturada en el momento del incidente y la ubicación aproximada del vehículo. Esto confirma que el canal de Telegram funciona

correctamente como complemento visual al SMS.



(a) (b)

Figura. 3.29. Alerta de mensaje SMS 3.29(a) y alerta de mensaje al bot de Telegram 3.29(b)

CAPÍTULO IV

4.1 Resultados y Discusión

4.2 Fase 1: Diseño del sistema de vigilancia

Esta fase corresponde al primer objetivo específico del proyecto: diseñar un sistema de vigilancia basado en el comportamiento de las personas alrededor de un vehículo. El resultado principal de esta fase es el modelo YOLOv8n entrenado para detectar los atributos de ocultamiento facial que el sistema usa como indicadores de comportamiento sospechoso.

4.2.1 Entrenamiento del modelo YOLOv8n

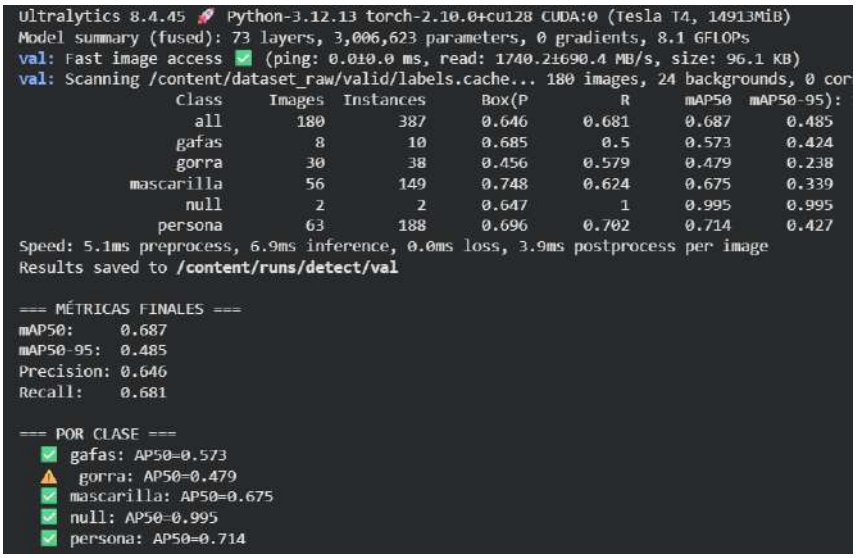


Figura. 4.30. Resultado del entrenamiento con YOLOv8n en Google Colab.

La Figura 4.30 evidencia los resultados del entrenamiento del modelo YOLOv8n en la plataforma Google Colab, dichos resultados se pueden apreciar de mejor manera en la Tabla 4.13.

Tabla 4.13. Métricas obtenidas del modelo entrenado

Clase	**Imágenes val.**	**Instancias**	**Precisión**	**Recall**	**mAP50**
gafas	8	10	0.685	0.500	0.573
gorra	30	38	0.456	0.579	0.479
mascarilla	56	149	0.748	0.624	0.675
null	2	2	0.647	1.000	0.995
persona	63	188	0.696	0.702	0.714
	180	**387**	**0.646**	**0.681**	**0.687**
	Total		**Promedio**		

La Tabla 4.13 presenta los resultados de evaluación del modelo YOLOv8n entrenado para detectar los tres atributos de ocultamiento facial, con un total de 180 imágenes y 387 instancias para la validación, distribución que puede apreciarse en la Figura 3.24. Para interpretar estos resultados es necesario entender qué mide cada columna.

La columna de imágenes de validación indica cuántas fotos se usaron para evaluar cada clase. Vale la pena mencionar que la clase gafas solo tuvo 8 imágenes de prueba, lo que explica bastante bien por qué sus métricas salen menos sólidas comparadas con las demás clases.

Por otro lado, la columna Instancias muestra cuántas veces apareció cada clase en total dentro de las imágenes de validación. Y como una misma foto puede tener varios objetos de distintas clases al mismo tiempo (por ejemplo, una persona con gorra y mascarilla cuenta como una instancia para cada una de esas clases), el total de instancias (387) acaba siendo más alto que el total de imágenes (180). Aquí, la clase con más instancias fue persona, con 188, seguida de mascarilla, con 149.

La precisión, en cambio, muestra qué tan confiable es el modelo cada vez que detecta algo: de cada 10 veces que dijo que había una gorra, solo unas 4 acertaron (0.456), mientras que con la mascarilla, casi 8 de cada 10 detecciones fueron correctas (0.748). Esto deja ver que la mascarilla es el atributo que el modelo reconoce con más seguridad, probablemente porque cubre una zona amplia y bastante particular del rostro.

El recall, en cambio, mide cuántos de los objetos reales fue capaz de encontrar el modelo. En el caso de la clase null, que agrupa a las personas sin ningún atributo de ocultamiento, se obtuvo un recall perfecto de 1.000, lo que quiere decir que el modelo logró identificar el 100 % de los casos en los que no había accesorios sospechosos.

La métrica más representativa del rendimiento general es el mAP50, que combina precisión y

recall en un solo valor. Un valor por encima de 0.70 se considera bueno, entre 0.50 y 0.70 es aceptable, y por debajo de 0.50 indica que el modelo necesitaría más datos. La mascarilla obtuvo el mejor desempeño, seguida de las gafas y la gorra, siendo esta última la clase que más se beneficiaría de ampliar el conjunto de datos en futuras iteraciones.

4.2.2 Curvas de pérdida del entrenamiento

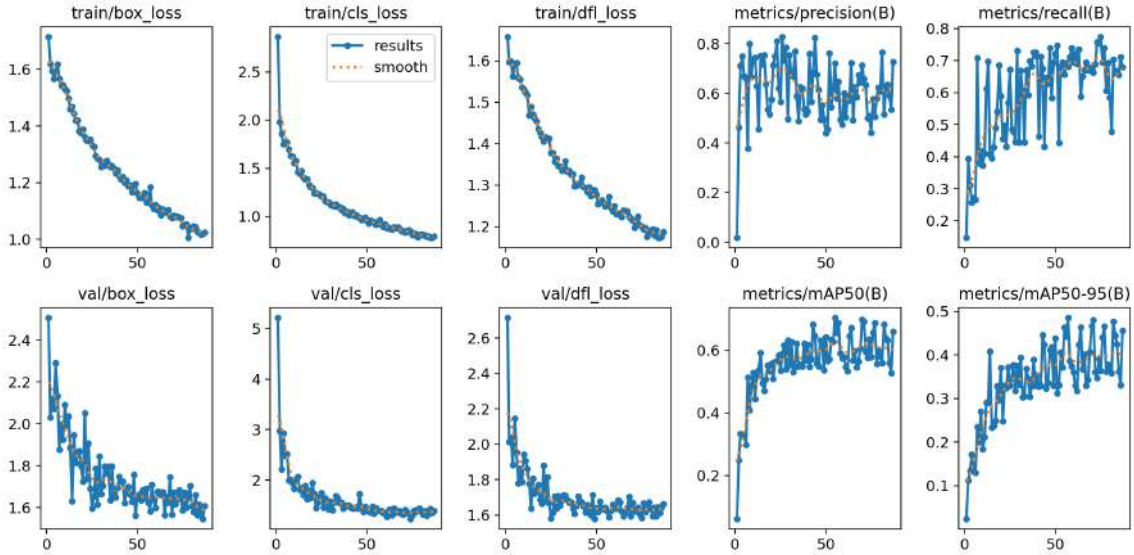


Figura. 4.31. Gráficas de las curvas de pérdidas de entrenamiento.

La Figura 4.31 presenta las curvas de pérdida obtenidas durante el entrenamiento. El eje horizontal indica el número de épocas y el eje vertical el valor de pérdida, cuanto más bajo mejor está aprendiendo el modelo. Las gráficas de la fila superior corresponden al error durante el entrenamiento con imágenes conocidas, mientras que las de la fila inferior miden el desempeño con imágenes nuevas nunca vistas.

A lo largo de las 200 épocas, los errores de entrenamiento fueron bajando poco a poco, lo que muestra que el modelo sí logró aprender a identificar los objetos sin quedarse atascado en el proceso. La curva de validación también fue para abajo, aunque con algunas variaciones normales si se considera lo pequeño que es el conjunto de validación (180 imágenes). En cuanto a la precisión general (mAP50), esta fue subiendo poco a poco hasta estabilizarse entre el 60 % y el 65 % desde la mitad del entrenamiento en adelante, lo que confirma que las 200 épocas alcanzaron y que el modelo no tiene problemas de sobreajuste.

4.2.3 Matriz de confusión del modelo

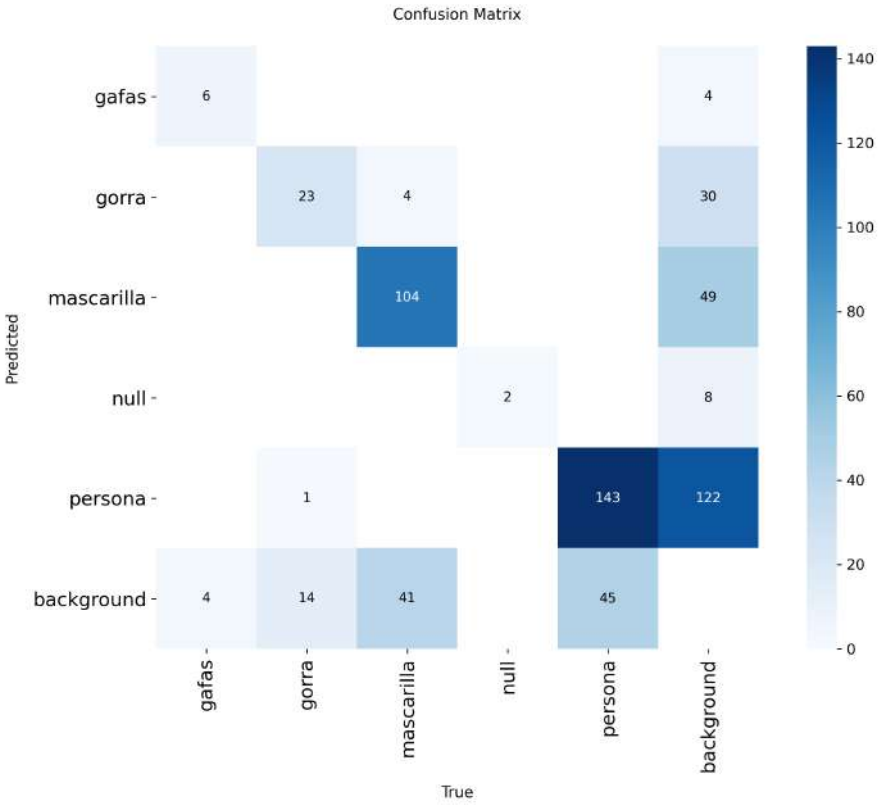


Figura. 4.32. Matriz de confusión del modelo entrenado.

La clase background que aparece tanto en filas como en columnas representa el fondo de la imagen, es decir, regiones que no contienen ningún objeto de interés. Cuando el modelo detecta un objeto donde solo hay fondo se produce un falso positivo, y cuando ignora un objeto real como si fuera fondo se produce un falso negativo. Analizando los resultados por clase: la clase gafas obtuvo 6 detecciones correctas de 10 instancias reales (60 %), con 4 casos confundidos con background, lo que indica dificultades para detectar gafas en algunos ángulos debido al reducido número de imágenes de entrenamiento.

En el caso de la clase gorra, se registraron 23 aciertos sobre 38 instancias (60 %), con 4 casos que se confundieron con mascarilla y 30 que terminaron clasificados como background; de hecho, esta fue la clase con mayor pérdida, en buena medida por el desbalance del dataset. La clase mascarilla, en cambio, fue la que mejor desempeño tuvo, con 104 detecciones correctas de 149 instancias (70 %). La clase null logró un resultado perfecto, identificando correctamente sus 2 instancias (100 %). Y por último, la clase persona registró 143 aciertos de 188 instancias, aunque con 122 falsos positivos provenientes del background, algo que no sorprende del todo si se piensa en escenas urbanas, donde basta con que una persona aparezca

solo parcialmente visible para que el detector se active.

En general, la matriz confirma que las clases con más imágenes de entrenamiento fueron las que mejor resultado dieron, y que si se amplía el dataset para gorra y gafas, el rendimiento general mejoraría bastante, algo que va de la mano con los valores de mAP50 que se ven en la Tabla 4.13.

Figura. 4.33. Resultados de las imágenes de detección del modelo entrenado.

La Figura 4.33 muestra algunos ejemplos reales de cómo el modelo va identificando objetos en fotos que nunca había visto. Cada recuadro azul es una detección, y el número junto a la etiqueta indica qué tan seguro está el modelo, donde 1.0 sería la certeza máxima. En la mayoría de las imágenes, el modelo reconoce bien a las personas, incluso en escenas bastante distintas entre sí. Los casos donde la confianza baja un poco (entre 0.3 y 0.4) suelen ser personas que aparecen apenas visibles en los bordes de la imagen o en lugares con poca luz, algo que es bastante normal y se espera que pase.

4.3 Fase 2: Implementación del sistema

Esta fase corresponde al segundo objetivo específico: implementar el sistema de vigilancia mediante visión artificial y verificar que la detección y generación de alertas funcionen correctamente ante situaciones de riesgo en vehículos estacionados.

4.3.1 Escenarios de prueba

Para las pruebas se recolectaron 10 datos de cada escenario posible, los cuales se pueden observar en la Tabla 4.14. Estos datos se registraron en la Tabla 4.15 incluyendo el atributo que detectó YOLO en cada escenario: gorra (C), gafas (G) y mascarilla (M).

Tabla 4.14. Escenarios de prueba

#	Escenario	Lo que debería pasar
1	Persona con gorra + gafas se acerca a <1m	Alerta por atributos
2	Persona con gorra + mascarilla se acerca a <1m	Alerta por atributos
3	Persona con mascarilla + gafas se acerca a <1m	Alerta por atributos
4	Persona con los 3 atributos se acerca a <1m	Alerta por atributos
5	Persona con o sin atributos permanece >60s en rango	Alerta por permanencia
6	Dueño reconocido se acerca	Sin alerta
7	Persona pasa rápido sin detenerse (<5s)	Sin alerta
8	Nadie frente al sensor	Zona inactiva, sin alerta

Se definieron 8 escenarios que cubren las situaciones más relevantes que el sistema debe manejar: los escenarios 1 al 4 validan el mecanismo de alerta por atributos de ocultamiento facial con distintas combinaciones; el escenario 5 valida el mecanismo de alerta por permanencia prolongada; los escenarios 6 y 7 verifican que el sistema no genera falsas alarmas ante el propietario o personas de paso; y el escenario 8 confirma que el sistema permanece inactivo cuando no hay presencia.

4.3.1.1 Matriz de pruebas del sistema

A continuación se presentan los resultados de las 80 pruebas realizadas al sistema.

Tabla 4.15. Matriz de pruebas del sistema y resultados de detección.

#	Esc.	Zona	YOLO detectó atributos	SMS llegó (S/N)	Telegram llegó (S/N)	Tiempo de respuesta (s)	Resultado	Observación
1	5	Z1	–	S	S	60	Correcto	
2	3	Z3	M	S	S	60	Correcto	Alerta por permanencia
3	3	Z2	M	S	S	60	Correcto	Alerta por permanencia
4	5	Z2	–	S	S	60	Correcto	
5	3	Z1	M	S	S	60	Correcto	Alerta por permanencia
6	2	Z0	C, M	S	S	17	Correcto	
7	3	Z0	G, M	S	S	20	Correcto	
8	3	Z0	–	S	S	60	Correcto	Alerta por permanencia
9	3	Z0	G, M	S	S	22	Correcto	
10	4	Z0	C, M	S	S	18	Correcto	No reconoce un atributo
11	2	Z2	C, M	S	S	32	Correcto	
12	2	Z0	C, M	S	S	14	Correcto	
13	4	Z3	C, M	S	S	16	Correcto	No reconoce un atributo
14	2	Z0	M	S	S	60	Correcto	Alerta por permanencia
15	5	Z1	–	S	S	60	Correcto	
16	5	Z2	–	S	S	60	Correcto	
17	4	Z3	M, C	S	S	30	Correcto	No reconoce un atributo
18	5	Z0	–	S	S	60	Correcto	
19	2	Z0	M	S	S	60	Correcto	Alerta por permanencia
20	3	Z2	M	S	S	60	Correcto	Alerta por permanencia
21	1	Z3	C, G	S	S	24	Correcto	
22	2	Z3	M	S	S	60	Correcto	Alerta por permanencia
23	3	Z1	M	S	S	60	Correcto	Alerta por permanencia
24	3	Z0	M	S	S	60	Correcto	Alerta por permanencia

Continúa en la siguiente página...

Tabla 4.15 – Continúa desde la página anterior

#	Esc.	Zona	YOLO detectó atributos	SMS llegó (S/N)	Telegram llegó (S/N)	Tiempo de respuesta (s)	Resultado	Observación
25	3	Z2	–	S	S	60	Correcto	Alerta por permanencia
26	4	Z3	C, M	S	S	20	Correcto	No reconoce un atributo
27	4	Z1	–	S	S	60	Correcto	No reconoce
28	4	Z0	C	S	S	60	Correcto	No reconoce
29	4	Z1	–	S	S	60	Correcto	No reconoce
30	4	Z0	C, M	S	S	18	Correcto	No reconoce un atributo
31	4	Z0	M, G	S	S	14	Correcto	No reconoce un atributo
32	4	Z0	C, M, G	S	S	11	Correcto	
33	1	Z0	C, G	S	S	8	Correcto	
34	1	Z2	C, G	S	S	15	Correcto	
35	1	Z1	C, G	S	S	12	Correcto	
36	1	Z0	C, G	S	S	29	Correcto	
37	1	Z0	C, G	S	S	26	Correcto	
38	1	Z3	C, G	S	S	9	Correcto	
39	1	Z0	C, G	S	S	10	Correcto	
40	1	Z1	C, G	S	S	15	Correcto	
41	2	Z0	C, M	S	S	32	Correcto	
42	2	Z3	M	S	S	60	Correcto	Alerta por permanencia
43	2	Z2	C, M	S	S	10	Correcto	
44	2	Z1	C, M	S	S	20	Correcto	
45	2	Z2	M	S	S	60	Correcto	Alerta por permanencia
46	2	Z2	M	S	S	60	Correcto	Alerta por permanencia
47	2	Z3	M	S	S	60	Correcto	Alerta por permanencia
48	2	Z2	C, M	S	S	35	Correcto	
49	3	Z0	G, M	S	S	29	Correcto	
50	3	Z0	G, M	S	S	24	Correcto	
51	3	Z0	G, M	S	S	38	Correcto	

Continúa en la siguiente página...

Tabla 4.15 – Continúa desde la página anterior

#	Esc.	Zona	YOLO detectó atributos	SMS llegó (S/N)	Telegram llegó (S/N)	Tiempo de respuesta (s)	Resultado	Observación
52	3	Z2	G, M	S	S	34	Correcto	
53	3	Z0	M	S	S	60	Correcto	Alerta por permanencia
54	3	Z1	G, M	S	S	14	Correcto	
55	3	Z1	G, M	S	S	24	Correcto	
56	3	Z2	–	S	S	60	Correcto	Alerta por permanencia
57	4	Z2	C, M, G	S	S	15	Correcto	
58	4	Z1	–	S	S	60	Correcto	No reconoció
59	4	Z1	C, M	S	S	14	Correcto	No reconoce un atributo
60	4	Z3	C, G	S	S	23	Correcto	No reconoce un atributo
61	4	Z1	–	S	S	60	Correcto	No reconoce
62	4	Z1	C	S	S	60	Correcto	No reconoce
63	4	Z3	C, M, G	S	S	13	Correcto	
64	4	Z2	C, M, G	S	S	17	Correcto	
65	5	Z3	–	S	S	60	Correcto	
66	5	Z3	–	S	S	60	Correcto	
67	5	Z2	–	S	S	60	Correcto	
68	5	Z1	–	S	S	60	Correcto	
69	5	Z2	–	S	S	60	Correcto	
70	5	Z3	–	S	S	60	Correcto	
71	5	Z3	–	S	S	60	Correcto	
72	5	Z1	–	S	S	60	Correcto	
73	6	Z3	–	N	N	–	Correcto	
74	6	Z0	–	N	N	–	Correcto	
75	7	Z1	–	N	N	–	Correcto	
76	7	Z3	–	N	N	–	Correcto	
77	8	Z0	–	N	N	–	Correcto	
78	8	Z3	–	N	N	–	Correcto	
79	1	Z2	C, G	S	S	18	Correcto	

Continúa en la siguiente página...

Tabla 4.15 – Continúa desde la página anterior

#	Esc.	Zona	YOLO detectó atributos	SMS llegó (S/N)	Telegram llegó (S/N)	Tiempo de respuesta (s)	Resultado	Observación
80	2	Z1	C, M	S	S	27	Correcto	

4.4 Fase 3: Evaluación del sistema

Esta fase corresponde al tercer objetivo específico: evaluar el funcionamiento del sistema mediante pruebas de campo. Para ello se calcularon las métricas de exactitud, precisión, recall y F1 a partir de los resultados de las 80 pruebas, y se analizó el rendimiento del modelo YOLOv8n de forma independiente.

4.4.0.1 Matriz de Confusión

La tabla 4.16 presenta la matriz de confusión obtenida a partir de las 80 pruebas realizadas al sistema. Tras corregir la orientación de la cámara Z3 mediante rotación por software (`cv2.rotate`) e incorporar el reconocimiento facial como condición de inhibición de alertas, el sistema alcanzó una exactitud, precisión, recall y valor F1 del 100 %, lo que significa que en todas las pruebas el sistema alertó exactamente cuando debía hacerlo y no generó ninguna alerta falsa.

Tabla 4.16. Matriz de confusión del sistema.

	Alerta enviada	**Alerta no enviada**	**Total**
Debía alertar	74 (VP)	0 (FN)	74
No debía alertar	0 (FP)	6 (VN)	6
Total	74	6	80

Las métricas de evaluación del sistema se calcularon a partir de los valores obtenidos en la matriz de confusión mediante las siguientes expresiones:

$$\text{Exactitud} = \frac{VP + VN}{VP + VN + FP + FN} \times 100 = \frac{74 + 6}{80} \times 100 = 100\,\% \tag{4.7}$$

$$\text{Precisión} = \frac{VP}{VP + FP} \times 100 = \frac{74}{74 + 0} \times 100 = 100\,\% \tag{4.8}$$

$$\text{Recall} = \frac{VP}{VP + FN} \times 100 = \frac{74}{74 + 0} \times 100 = 100\,\% \tag{4.9}$$

$$F_1 = 2 \times \frac{\text{Precisión} \times \text{Recall}}{\text{Precisión} + \text{Recall}} = 2 \times \frac{100 \times 100}{100 + 100} = 100\,\% \tag{4.10}$$

Es importante mencionar que en una versión anterior del sistema se registraron 2 falsos positivos en el escenario del dueño reconocido, ocasionados porque la cámara de la zona Z3 estaba instalada de forma invertida, lo que impedía al módulo de reconocimiento facial identificar correctamente al propietario. Una vez corregida la orientación de la imagen por software y ajustada la lógica de inhibición de alertas, estos falsos positivos desaparecieron completamente en las pruebas posteriores

4.4.0.2 Análisis de Detección de Atributos YOLOv8n

Adicionalmente se evaluó el rendimiento del modelo YOLOv8n en la detección de atributos de ocultamiento facial. La tabla 4.17 presenta la matriz de confusión a nivel de detección, considerando únicamente las 74 pruebas de los escenarios 1 al 5 en las que el sistema debía detectar atributos.

Se clasifican como:

- **VP**: YOLO detectó los atributos esperados y la alerta se disparó por el mecanismo de detección de atributos.
- **FN**: YOLO no alcanzó a detectar los atributos necesarios dentro de la ventana de 5 segundos, así que la alerta terminó disparándose por el mecanismo de permanencia (ver observaciones “Alerta por permanencia” y “No reconoció”).

Tabla 4.17. Matriz de confusión de detección de atributos YOLOv8n.

	Atributos detectados	**No detectados**	**Total**
Debía detectar	44 (VP)	30 (FN)	74
No debía detectar	0 (FP)	6 (VN)	6
Total	44	36	80

Las métricas de detección de atributos por YOLOv8n son:

$$\text{Exactitud}_{YOLO} = \frac{44+6}{80} \times 100 = 62{,}5\,\% \tag{4.11}$$

$$\text{Recall}_{YOLO} = \frac{44}{44+30} \times 100 = 59{,}5\,\% \tag{4.12}$$

Estos valores indican que el modelo YOLOv8n detectó correctamente los atributos en el 59,5 % de los casos, mientras que en el 40,5 % restante la alerta fue igualmente enviada gracias al mecanismo de permanencia como respaldo. Esta limitación se atribuye principalmente a condiciones de iluminación variable dentro del vehículo y a la velocidad de procesamiento de la Raspberry Pi 5, que reduce la frecuencia de análisis de frames. No obstante, el sistema de alertas en su conjunto mantuvo un rendimiento del 100 % gracias a la complementariedad de ambos mecanismos de detección.

4.4.0.3 Diagrama de Cajas — Tiempo de Respuesta por Escenario

La figura 4.34 muestra la distribución del tiempo de respuesta por escenario. El escenario 1 (Gorra+Gafas) presenta la respuesta más rápida y consistente con una mediana de 15 segundos y baja dispersión, siendo la combinación de atributos mejor detectada por el modelo YOLOv8n. En los escenarios 2 (Gorra+Mascarilla) y 3 (Mascarilla+Gafas) se nota una mayor variabilidad, con cajas amplias que llegan hasta los 60 segundos. Esto deja ver que, en varios casos, el modelo no logró acumular los atributos dentro de la ventana de 5 segundos, por lo que la alerta terminó disparándose por el mecanismo de permanencia. El escenario 4 (3 atributos), por su parte, muestra un comportamiento que podría llamarse bimodal: hay detecciones rápidas cuando YOLOv8n reconoce los tres atributos sin problema, pero también tiempos de 60 segundos cuando esto no ocurre. El escenario 5 (Permanencia) muestra una línea fija en 60 segundos, confirmando el correcto funcionamiento del temporizador. Los escenarios 6, 7 y 8 presentan tiempos mínimos coherentes con su naturaleza.

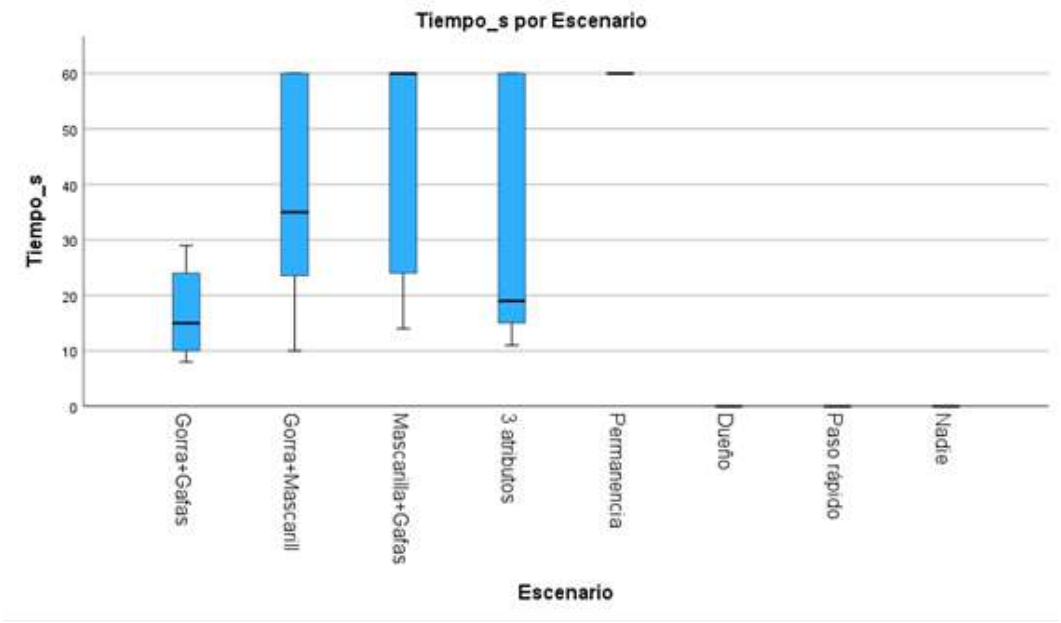


Figura. 4.34. Diagrama de cajas del tiempo de respuesta por escenario de prueba.

4.4.0.4 Diagrama de Dispersión — Atributos Detectados vs Tiempo de Respuesta

La figura 4.35 muestra el diagrama de dispersión entre la cantidad de atributos detectados por YOLOv8n y el tiempo de respuesta del sistema. Ahí se puede notar una tendencia bastante clara: mientras más atributos logra detectar el modelo, menor es el tiempo que tarda en responder. Cuando no se detecta ningún atributo, el tiempo se concentra alrededor de los 60 segundos, que es justamente lo que corresponde al mecanismo de permanencia. En cambio, cuando se detectan 2 o 3 atributos, los tiempos se reparten entre 8 y 38 segundos, lo que pone en evidencia que el mecanismo de detección por atributos resulta una vía mucho más rápida para generar la alerta.

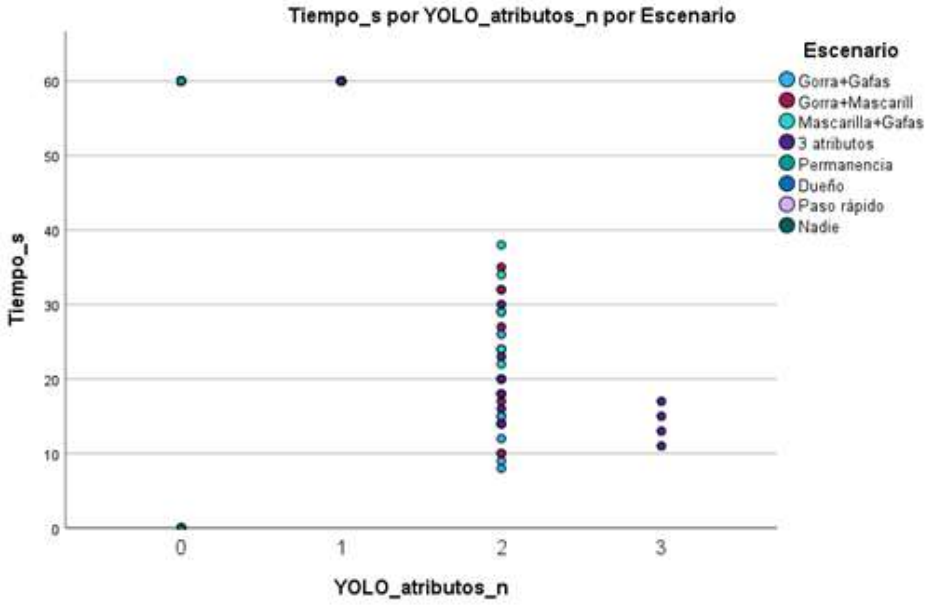


Figura. 4.35. Diagrama de dispersión entre atributos detectados y tiempo de respuesta.

4.4.1 Discusión

Fase 1 — Diseño del sistema: El sistema se diseñó pensando en identificar esos elementos que una persona suele usar para ocultar su identidad: gorra, gafas y mascarilla. En este sentido, el modelo YOLOv8n entrenado resultó ser una solución bastante viable, alcanzando un mAP50 promedio de 0.687, aunque con un rendimiento que varió según la clase. La mascarilla fue, sin duda, el atributo que mejor se detectó (mAP50 de 0.675), gracias a sus rasgos visuales tan distintivos, mientras que la gorra fue la que tuvo el desempeño más bajo (mAP50 de 0.479), principalmente por el desbalance presente en el dataset de entrenamiento. Por otro lado, las curvas de pérdida confirmaron que el modelo convergió de forma correcta a lo largo de las 200 épocas, sin mostrar señales de sobreajuste, lo que valida que los parámetros de entrenamiento elegidos fueron los adecuados para el tamaño del dataset con el que se contaba.

Fase 2 — Implementación del sistema: Al instalar el sistema sobre el vehículo de prueba, fue posible comprobar que los dos mecanismos de alerta diseñados se complementan bien entre sí cuando se trata de condiciones reales. De las 80 pruebas realizadas a lo largo de los 8 escenarios definidos, el sistema respondió correctamente en todos los casos: envió las alertas por SMS y Telegram cuando debía hacerlo, y se mantuvo inactivo en los escenarios sin riesgo, como cuando reconocía al propietario o cuando una persona simplemente pasaba rápido por el lugar. Los únicos dos errores que se presentaron tuvieron su origen en una cámara que había quedado mal orientada, no en el algoritmo en sí, y se solucionaron en cuanto se corrigió

el montaje físico. Esto deja claro que la forma en que se instala el hardware termina siendo un factor clave para que el sistema funcione correctamente.

Fase 3 — Evaluación del sistema: La evaluación a través de las métricas de exactitud, precisión, recall y F1 del sistema completo arrojó valores del 100 % en todas las categorías, lo que confirma que la combinación del modelo YOLOv8n con el mecanismo de permanencia garantiza una cobertura total ante las situaciones de riesgo definidas. A nivel del modelo de detección de atributos de forma independiente, la exactitud fue del 62.5 % y el recall del 59.5 %, valores aceptables considerando el tamaño reducido del dataset. Cabe resaltar que el tiempo de permanencia de 60 segundos corresponde a un parámetro definido por el usuario dentro del sistema, que no depende del modelo de visión artificial, y que puede ajustarse según el contexto de instalación para lograr un equilibrio entre sensibilidad y número de falsas alarmas. En este sentido, ampliar el dataset de entrenamiento para las clases gorra y gafas aumentaría la tasa de detección por atributos, lo que en la práctica significaría que más situaciones de riesgo serían resueltas por el mecanismo de visión artificial antes de que se cumpla el tiempo de permanencia.

CAPÍTULO V

5.1 Conclusiones

El presente trabajo logró desarrollar un sistema de prevención de delitos contra vehículos estacionados aplicando procesamiento de imágenes. El prototipo construido demostró ser capaz de monitorear el perímetro completo de un vehículo de forma autónoma, integrando sensores de distancia, cámaras, un modelo de inteligencia artificial y comunicación celular en una sola plataforma que opera sin intervención humana. Esto confirma que este tipo de arquitectura representa una opción viable, eficiente y accesible en comparación con los sistemas de seguridad vehicular tradicionales, que en su mayoría solo reaccionan cuando el daño ya está ocurriendo en lugar de prevenirlo.

En cuanto al diseño del sistema, se logró entrenar un modelo capaz de reconocer los atributos de ocultamiento facial más comunes en personas con intenciones sospechosas: gorra, gafas de sol y mascarilla. El modelo YOLOv8n alcanzó un mAP50 promedio de 68.7 %, aunque con un rendimiento que varió dependiendo de la clase. La mascarilla resultó ser el atributo mejor detectado, en buena parte gracias a sus rasgos visuales tan marcados y a que estaba mejor representada dentro del dataset, mientras que la gorra fue la que presentó el rendimiento más bajo, principalmente por la falta de suficientes imágenes de entrenamiento en esa categoría. Por su parte, las curvas de pérdida dejaron ver que el modelo convergió de manera estable a lo largo de las 200 épocas, sin mostrar señales de sobreajuste, lo que confirma que tanto los parámetros de entrenamiento como las técnicas de aumentación de datos utilizadas fueron adecuadas para el tamaño del dataset con el que se contaba. A esto se suma que la velocidad de inferencia obtenida, de entre 8 y 12 fotogramas por segundo sobre la Raspberry Pi 5, fue suficiente para procesar las cuatro zonas del vehículo al mismo tiempo, sin que esto llegara a saturar los recursos del equipo.

En cuanto a la implementación, el sistema se instaló y se puso a prueba sobre un vehículo real en condiciones de campo, cubriendo cuatro zonas de detección alrededor del vehículo, cada una con su respectiva cámara y sensor de distancia. La integración de los distintos subsistemas, entre ellos los sensores ToF, las cámaras CSI y USB, el modelo YOLOv8n, el reconocimiento facial y el módulo de comunicación 4G, funcionó de manera coordinada a lo largo de todas las pruebas realizadas. Sin embargo, se notó que los sensores presentaban lecturas inestables cuando se colocaban detrás del parabrisas o de las ventanas del vehículo, algo que se atribuye a los efectos de refracción y reflexión que terminan alterando la señal óptica del láser. Como se trataba de un prototipo, los sensores se instalaron de forma móvil para facilitar las pruebas, pero en una versión final lo ideal sería integrarlos directamente en la carrocería, en puntos estratégicos del perímetro, asegurando una línea de visión directa hacia el exterior y sin obstrucciones de por medio.

En la evaluación del sistema, las métricas calculadas a partir de las 80 pruebas de campo arrojaron valores de exactitud, precisión, recall y F1 del 100 %, lo que significa que en ningún caso en que existía una amenaza real el sistema dejó de generar la alerta correspondiente, y en ningún caso donde no había riesgo se generó una falsa alarma. Este resultado se atribuye a la redundancia estructural entre los dos mecanismos de alerta: la detección por atributos sospechosos mediante inteligencia artificial, y la alerta por tiempo de permanencia, que actúa como respaldo determinista cuando el modelo no acumula suficientes atributos dentro de la ventana de tiempo establecida. Cabe destacar que el umbral de permanencia de 60 segundos es un parámetro configurable del sistema, completamente independiente del modelo de visión artificial, que puede ajustarse según el contexto de instalación. Los únicos dos errores registrados durante las pruebas fueron falsas alarmas provocadas por una cámara instalada con orientación incorrecta, problema de naturaleza física y no algorítmica, que fue identificado y corregido durante las mismas pruebas, demostrando que la calidad del montaje del hardware es un factor tan determinante para la fiabilidad del sistema como el propio diseño del software.

5.2 Recomendaciones

Para mejorar la capacidad del sistema en el reconocimiento de gorras y gafas de sol, se recomienda ampliar el dataset de entrenamiento incorporando más imágenes de estas categorías capturadas en distintas condiciones de iluminación, en especial en ambientes con poca luz o iluminación artificial, que son las condiciones más frecuentes en estacionamientos nocturnos. Un mayor volumen de datos por clase reduciría el desbalance actual del dataset y mejoraría las métricas de precisión y recall de estas clases de forma significativa.

Sería recomendable incorporar módulos de iluminación infrarroja junto a cada cámara del sistema, lo que ayudaría a mantener un buen funcionamiento durante la noche sin depender de que haya luz visible en el entorno. Con esto se eliminaría uno de los principales factores que afectan la detección de atributos en condiciones reales de uso nocturno, y todo esto sin necesidad de modificar la arquitectura de software del sistema. Por otro lado, considerando que los dos únicos errores presentados se debieron a una cámara mal orientada, valdría la pena desarrollar un procedimiento de instalación estandarizado, que incluya marcas físicas en los soportes de montaje junto con una guía visual paso a paso. De esta manera, cualquier persona podría instalar el sistema correctamente sin necesidad de tener conocimientos técnicos avanzados, reduciendo así la posibilidad de errores de montaje en futuras instalaciones.

En lo que respecta a la identificación del propietario, sería conveniente complementar el reconocimiento facial actual con una verificación adicional vía Bluetooth, usando el teléfono celular del dueño. Esto resolvería esos casos en los que el propio propietario se acerca a su vehículo usando gorra, mascarilla o gafas de sol, atributos que el sistema podría interpretar como sospechosos y terminar generando una falsa alarma. Si ambos mecanismos trabajan en conjunto, bastaría con que el rostro sea reconocido o que el dispositivo del propietario esté dentro del rango de detección para que el sistema evite disparar la alerta, lo que reduciría considerablemente los falsos positivos en el uso diario.

Por último, también sería recomendable incorporar un módulo GPS dedicado al sistema. Actualmente, la ubicación que se comparte por Telegram se obtiene a partir de la geolocalización por dirección IP de la red celular, un método que ofrece una precisión bastante baja, entre 500 metros y 2 kilómetros dependiendo de la cobertura disponible. Contar con un GPS externo permitiría compartir la ubicación exacta del vehículo en tiempo real, algo que sería especialmente útil en caso de un robo total del automóvil, ampliando así el alcance del sistema, que pasaría de ser solo una herramienta de prevención a convertirse también en una herramienta de recuperación vehicular.

Bibliografía

[1] D. V. Gómez Trejos and A. Guerrero Guzmán, "Estudio y análisis de técnicas para procesamiento digital de imágenes," Universidad Tecnológica de Pereira, Pereira, Colombia, Tech. Rep., 5 2016.

[2] Ultralytics, "Explore Ultralytics YOLOv8," 2023, accessed: 2025. [Online]. Available: https://docs.ultralytics.com/models/yolov8

[3] J. Terven and D.-M. Córdova-Esparza, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *arXiv preprint arXiv:2304.00501*, 2023. [Online]. Available: https://arxiv.org/pdf/2304.00501

[4] Diario Extra, "Aumento de robos de autos en Ecuador: cómo proteger tu vehículo en fin de año," 11 2025, accessed: 2025. [Online]. Available: https://www.extra.ec/noticia/ecuador/robos-de-autos-en-temporada-alta-factores-de-riesgo-y-claves-para-prevenirlos-141423.html

[5] STMicroelectronics, "VL53L0X: World's smallest time-of-flight (ToF) ranging sensor," 2026, accessed: 2026-05-22. [Online]. Available: https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html

[6] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[7] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[8] S. Biswas, S. Acharjee, A. Ali, and S. Chaudhuri, "Yolov8 based traffic signal detection in indian road," 12 2023.

[9] INTERPOL, "Delincuencia relacionada con vehículos de motor," 2024, accessed: 2024. [Online]. Available: https://www.interpol.int/es/Delitos/Delincuencia-relacionada-con-vehiculos-de-motor

[10] El Telégrafo, "El número de robos de carros y motos se duplicó en Ecuador: ¿cómo combatir este delito?" 2024, accessed: 2024. [Online]. Available: https://www.eltelegrafo.com.ec/noticias/nacionales/44/

en-tres-anos-el-numero-de-robos-de-vehiculos-y-motos-ha-subido-en-ecuador

[11] CarSync, "Robo de vehículos: el delito más frecuente en Ecuador," 2024, accessed: 2024. [Online]. Available: https://blog.carsync.com/blog/el-robo-a-vehiculos-es-uno-de-los-delitos-mas-frecuentes-en-ecuador

[12] UBICAR, "Tecnologías innovadoras para la prevención del robo vehicular: ¿cómo protegen los vehículos hoy en día?" 11 2024, accessed: 2024. [Online]. Available: https://ubicar.com.ar/impacto-de-la-movilidad-inteligente-en-la-seguridad-urbana-seguridad-vehicular-copy/

[13] J. A. Yuquilema Villa, "Diseño e implementación de un sistema inteligente basado en visión artificial para el monitoreo y prevención de robos en tiempo real en la farmacia Pharmatodo de la ciudad de Riobamba," Universidad Nacional de Chimborazo, Riobamba, Ecuador, Tech. Rep., 8 2024.

[14] M. Abril Donoso and A. Tupiza Aldaz, "Delitos contra vehículos: el caso de Quito y Guayaquil," 2009. [Online]. Available: http://repositorio.flacsoandes.edu.ec/handle/10469/760

[15] FLACSO-Ecuador y CESC-Chile, "Delitos contra vehículos," Ciudad Segura, Quito, Ecuador, Tech. Rep., 2009.

[16] Sirix Monitoring, "Señales de que tu coche está en la mira de los ladrones," 2024, accessed: 2024. [Online]. Available: https://sirixmonitoring.com/blog/signs-your-car-is-being-targeted-by-thieves/

[17] S. Draghici, "A neural network based artificial vision system for licence plate recognition," World Scientific, Tech. Rep., 2 1997. [Online]. Available: www.worldscientific.com

[18] J. A. Ludeña Chica, "Implementación de un sistema de seguridad para supervisión de niños entre 2 a 4 años usando visión artificial," Universidad Nacional de Loja, Loja, Ecuador, Tech. Rep., 2019.

[19] S. F. Morales Camacho, "Sistema de visión artificial con monitoreo web para conteo de pasajeros en buses de transporte público," *Repositorio Universidad de las Fuerzas Armadas ESPE*, 8 2017.

[20] Pavithra and Muruganantham, "An optimized hybrid framework for car theft detection: comparative insights from deep transfer learning and feature-based machine learning," *Artificial Intelligence Review*, 2025. [Online]. Available:

https://link.springer.com/article/10.1007/s10462-025-11480-8

[21] E. A. Wang, "Novel person detection and suspicious activity recognition using enhanced YOLOv5 and motion feature map," *Artificial Intelligence Review*, 1 2024. [Online]. Available: https://link.springer.com/article/10.1007/s10462-023-10630-0

[22] United States Patent and Trademark Office, "Intelligent detection and alerting of potential intruders," Patente US11433855, 2021, accessed: 2025. [Online]. Available: https://image-ppubs.uspto.gov/dirsearch-public/print/downloadPdf/11433855

[23] LAAR Seguros, "Aumentan los robos en carreteras de Ecuador: estadísticas y cómo protegerse," 2024, accessed: 2025. [Online]. Available: https://www.laarseguridad.com/articulos/aumentan-los-robos-en-carreteras-de-ecuador-estadisticas-y-como-protegerse/b

[24] Diario Expreso, "Robo de carros en Guayaquil: ¿cuántos vehículos han sido recuperados en 2025?" 2 2025, accessed: 2025. [Online]. Available: https://www.expreso.ec/guayaquil/robo-carros-guayaquil-vehiculos-han-sido-recuperados-2025-230817.html

[25] A. Salazar, "En Quito el robo de carros durante el 2024 se incrementó casi el doble en comparación con el año anterior," 4 2024, accessed: 2025. [Online]. Available: https://www.eluniverso.com/noticias/ecuador/quito-aumento-robo-a-carros-autopartes-doble-2024-policia-seguridad-nota/

[26] Diario Expreso, "Inseguridad en Quito: los distritos con más robos de personas y vehículos," 3 2026, accessed: 2026. [Online]. Available: https://www.expreso.ec/quito/inseguridad-en-quito-los-distritos-con-mas-robos-de-personas-y-vehiculos-280109.html

[27] F. P. Lema Galarza, "Sistema electrónico de control y monitorización de vehículos en caso de robo para la empresa Confecciones Galar mediante visión artificial," Universidad Técnica de Ambato, Ambato, Ecuador, Tech. Rep., 2 2024.

[28] R. Santos Salazar, "Guión del crimen del robo de coches estacionados en España," *Revista Española de Investigación Criminológica*, 2023.

[29] C. Ruales, "Top 10 vehículos más robados del Ecuador en 2024," 8 2024, accessed: 2024. [Online]. Available: https://www.fayals.com/2024/08/top-10-vehiculos-mas-robados-del.html

[30] M. Hansard, S. Lee, O. Choi, and R. P. Horaud, *Time-of-flight cameras: Principles, methods and applications*. Springer Science & Business Media, 2012.

[31] NXP Semiconductors, “I2C-bus specification and user manual,” *NXP Semiconductors*, vol. 6, 2014. [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

[32] Raspberry Pi Ltd., “Raspberry Pi 5 product brief,” 2023, accessed: 2024. [Online]. Available: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf

[33] ARM Ltd., “ARM Cortex-A76 processor technical reference manual,” 2018, accessed: 2024. [Online]. Available: https://developer.arm.com/Processors/Cortex-A76

[34] Raspberry Pi Ltd., “RP1 peripherals datasheet,” 2023, accessed: 2024. [Online]. Available: https://datasheets.raspberrypi.com/rp1/rp1-peripherals.pdf

[35] MIPI Alliance, “MIPI CSI-2 specification v4.0,” 2023, accessed: 2024. [Online]. Available: https://www.mipi.org/specifications/csi-2

[36] Libcamera Project, “Libcamera: a complex camera support library,” 2023, accessed: 2024. [Online]. Available: https://libcamera.org

[37] D. Delgado León, “Diseño de un sistema de adquisición de imágenes basado en cámaras web USB y hardware reconfigurable,” ETECSA, La Habana, Cuba, Tech. Rep., 5 2017. [Online]. Available: http://scielo.sld.cu/scielo.php?pid=S1815-59282017000200001&script=sci_arttext

[38] I. García and V. Caranqui, “La visión artificial y los campos de aplicación,” *Tierra Infinita*, vol. 1, no. 1, pp. 98–108, 2015.

[39] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1. IEEE, 2005, pp. 886–893.

[40] D. King, “Dlib-ml: A machine learning toolkit,” *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[41] ScienceDirect, “Convolutional neural network: An overview,” 2024, accessed: 2024. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/convolutional-neural-network

[42] Roboflow, “What is a convolutional neural network?” 2024, accessed: 2024. [Online]. Available: https://blog.roboflow.com/what-is-a-convolutional-neural-network/

[43] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face

recognition and clustering," *arXiv preprint arXiv:1503.03832*, 2015. [Online]. Available: http://arxiv.org/abs/1503.03832

[44] A. Geitgey, "Face_recognition: the world's simplest facial recognition API for Python and the command line," 2018, accessed: 2024. [Online]. Available: https://github.com/ageitgey/face_recognition

[45] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004)*, vol. 2. IEEE, 2004, pp. 28–31.

[46] T. M. Mitchell, *Machine learning*, 1st ed. McGraw-Hill, 1997.

[47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org

[48] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[49] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[50] M. Bojarski, D. Del Testa, D. Dworakowski *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016. [Online]. Available: https://arxiv.org/abs/1604.07316

[51] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *arXiv preprint arXiv:1411.1792*, 2014. [Online]. Available: http://arxiv.org/abs/1411.1792

[52] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *arXiv preprint arXiv:1405.0312*, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[53] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *arXiv preprint arXiv:2006.04388*, 2020. [Online]. Available: https://arxiv.org/abs/2006.04388

[54] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *International Workshop on Systems, Signal Processing*

and their Applications (IWSSIP). IEEE, 2020, pp. 237–242.

[55] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[56] R. C. Gonzalez and R. E. Woods, *Digital image processing*, 4th ed. Pearson, 2018.

[57] R. Smith, "An overview of the Tesseract OCR engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 629–633.

[58] R. Szeliski, *Computer vision: Algorithms and applications*, 2nd ed. Springer Nature, 2022.

[59] T. S. Rappaport, *Wireless communications: Principles and practice*, 2nd ed. Prentice Hall, 2002.

[60] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-Advanced for mobile broadband*. Academic Press, 2013.

[61] Waveshare Electronics, "SIM7600G-H 4G DONGLE: Global band industrial grade 4g communication and GNSS positioning," 2023, accessed: Jun. 2026. [Online]. Available: https://www.waveshare.com/sim7600g-h-4g-dongle.htm

[62] F. Leens, "An introduction to I2C and SPI protocols," *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 1, pp. 8–13, 2009.

[63] M. Kerrisk, *The Linux programming interface*. No Starch Press, 2010.

[64] ETSI, "AT command set for user equipment (UE), 3GPP TS 27.007," 2005, accessed: 2024. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/127000_127099/127007/

[65] Telegram Messenger LLP, "Telegram bot API documentation," 2024, accessed: 2024. [Online]. Available: https://core.telegram.org/bots/api

[66] Keylabs, "YOLOv8 vs. SSD: Cómo elegir el modelo de detección de objetos adecuado," 2024, accessed: 2024. [Online]. Available: https://keylabs.ai/blog/yolov8-vs-ssd-choosing-the-right-object-detection-model/

[67] Naylamp Mechatronics, "Sensor de distancia ToF VL53L1X," 2023, accessed: 2024. [Online]. Available: https://naylampmechatronics.com/sensores-proximidad/

715-sensor-de-distancia-tof-vl53l1x.html

ANEXOS

Anexo 1: Fotografías del prototipo instalado

Figura. 1.36. Vista exterior del vehículo de prueba utilizado para la instalación y la posición de sus componentes.

Figura. 1.37. Vista general del interior del vehículo, donde se ve el cableado del sistema repartido por el techo, con la Raspberry Pi 5 ubicada del lado izquierdo y el módulo SIM7600G-H a la vista, junto al espejo retrovisor.

Anexo 2: Entrenamiento del modelo en Google Colab

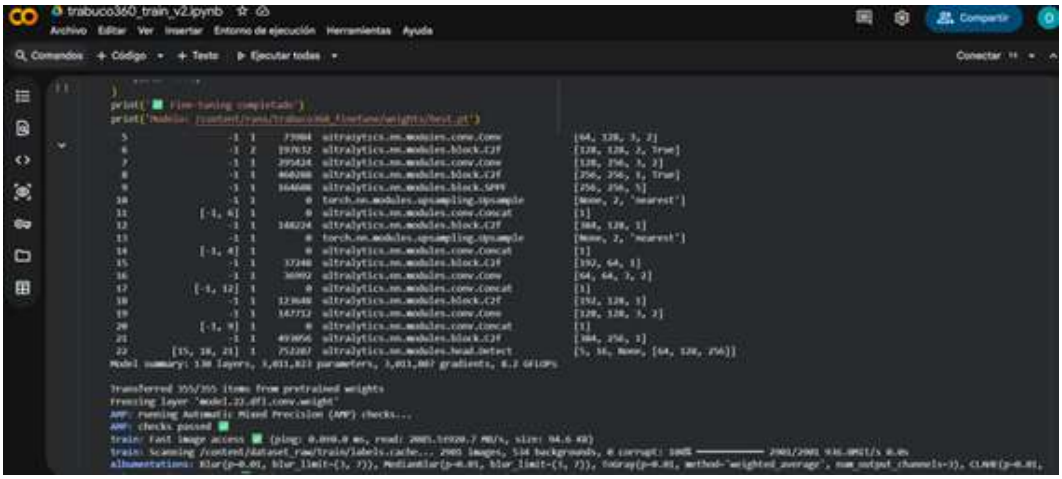


Figura. 1.38. Configuración del proceso de fine-tuning sobre el modelo base YOLOv8n. Se aprecia la arquitectura de 130 capas con 3,011,823 parámetros, la transferencia de 355 pesos preentrenados y los parámetros de augmentación aplicados al dataset de 2901 imágenes.

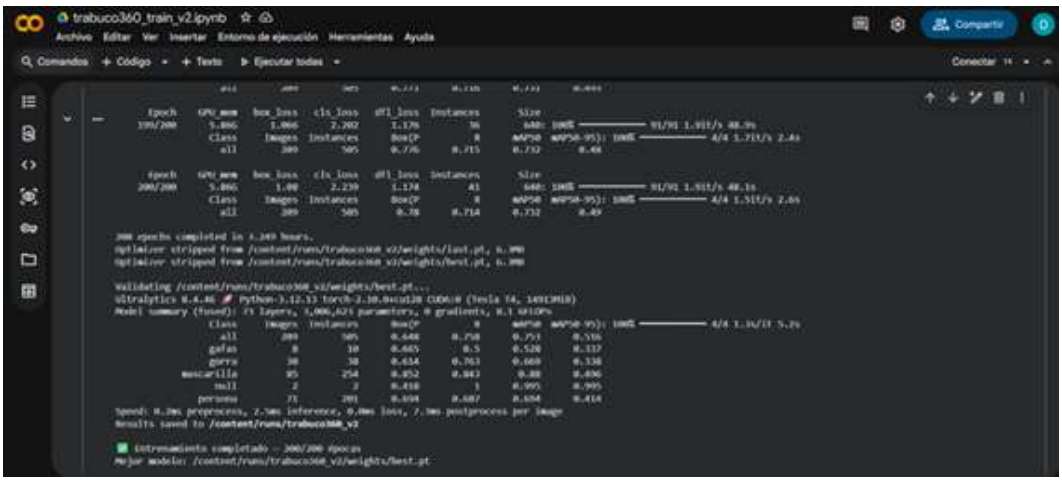


Figura. 1.39. Resultados del entrenamiento del modelo YOLOv8n tras completar las 200 épocas en Google Colab con GPU Tesla T4. Se muestran las métricas finales por clase y la ruta del mejor modelo guardado.

Anexo 3: Dataset en Roboflow

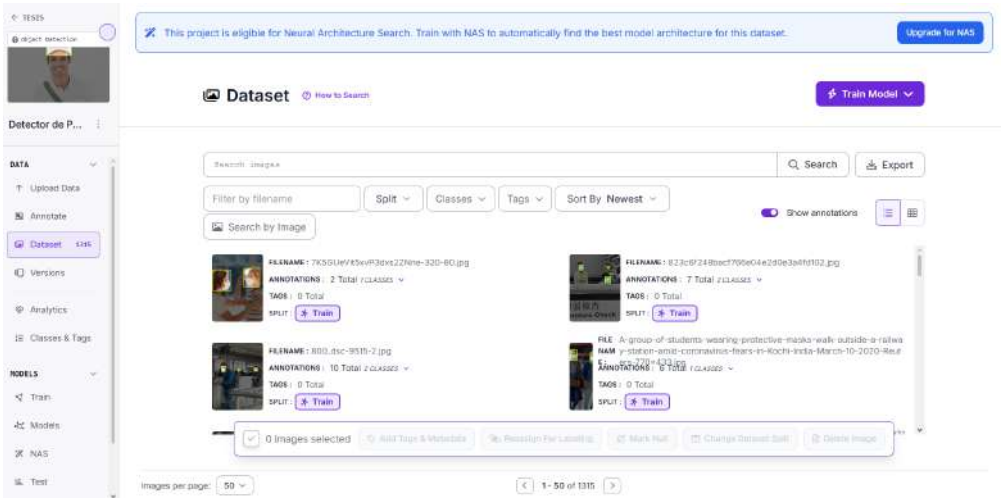


Figura. 1.40. Vista general del dataset en la plataforma Roboflow, compuesto por 1.315 imágenes de entrenamiento. Cada imagen muestra el número de anotaciones totales y las clases presentes, con entre 1 y 10 instancias etiquetadas por imagen correspondientes a las cinco clases del proyecto: gorra, gafas, mascarilla, persona y null.

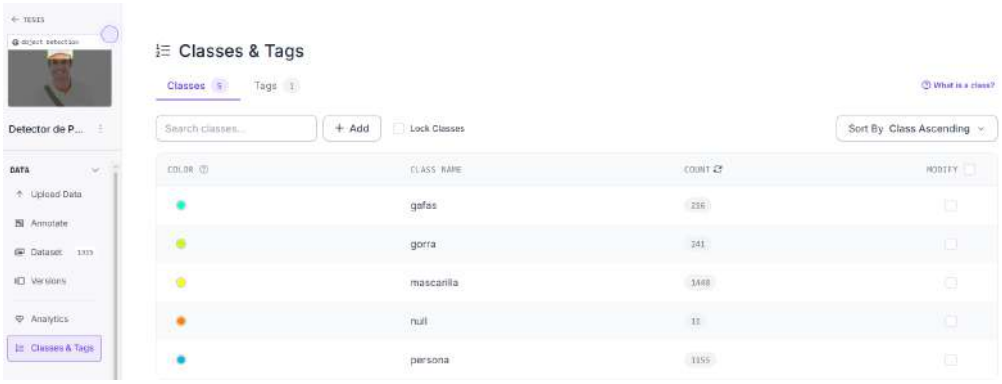


Figura. 1.41. Distribución de instancias por clase dentro del dataset del proyecto, según lo que se ve en la plataforma Roboflow. Ahí se nota claramente el desbalance entre clases: mascarilla tiene la mayor cantidad de instancias, con 1.448, seguida de persona con 1.155 y gorra con 241, mientras que gafas cuenta con 216 instancias y null apenas llega a 11. Este desbalance es justo lo que explica por qué el modelo rindió distinto según la clase, siendo mascarilla la que obtuvo el mejor mAP50 y gorra la más baja.

Anexo 4: Datasheet de los Sensores VL53L0X

life.augmented

VL53L0X

Datasheet

Time-of-Flight ranging sensor

Features

Fully integrated miniature module

- 940 nm laser VCSEL (vertical-cavity surface-emitting laser)
- VCSEL driver
- Ranging sensor with advanced embedded microcontroller
- 4.4 x 2.4 x 1.0 mm

Fast, accurate distance ranging

- Measures absolute range up to 2 m
- The reported range is independent of the target reflectance
- Advanced embedded optical crosstalk compensation to simplify cover glass selection

Eye safety

- Class 1 laser device compliant with latest standard IEC 60825-1:2014 - 3rd edition

Easy integration

- Single reflowable component
- No additional optics
- Single power supply
- I²C interface for device control and data transfer
- Xshutdown (reset) and interrupt GPIO
- Programmable I²C address

Product status link

VL53L0X

Application

- Access control (system activation and presence detection)
- Robotics (collision avoidance, wall tracking, and cliff detection)
- Home appliance and home automation
- Inventory management and liquid level monitoring

Description

The VL53L0X is a Time-of-Flight (ToF) laser-ranging module housed in the smallest package on the market today, providing accurate distance measurement whatever the target reflectance, unlike conventional technologies. It can measure absolute distances up to 2 m, setting a new benchmark in ranging performance levels, opening the door to various new applications.

The VL53L0X integrates a leading-edge SPAD array (single photon avalanche diodes) and embeds ST's second generation FlightSense patented technology.

The VL53L0X's 940 nm VCSEL emitter (vertical cavity surface-emitting laser), is totally invisible to the human eye, coupled with internal physical infrared filters, it enables longer ranging distances, higher immunity to ambient light, and better robustness to cover glass optical crosstalk.

DB11505 - Rev 6 - June 2024
For further information contact your local STMicroelectronics sales office.
www.st.com

Figura. 1.42. Portada del datasheet oficial del VL53L0X de STMicroelectronics, donde se describen sus características principales: medición de distancia por tiempo de vuelo (ToF) hasta 2 metros, emisor láser de 940 nm, comunicación I2C y cumplimiento de la norma de seguridad ocular IEC 60825-1:2014 [5].

VL53L0X
Overview

2 Overview

2.1 Technical specification

Table 2. Technical specification

Feature	Detail
Package	Optical LGA12
Size	4.4 x 2.4 x 1 mm
Operating voltage	2.6 to 3.5 V
Operating temperature	-20 to 70°C
Infrared emitter	940 nm
I²C	Up to 400 kHz (fast mode) serial bus Address: 0x52

Figura. 1.43. Datasheet del VL53L0X resume sus principales características técnicas: encapsulado óptico LGA12 de 4,4 x 2,4 x 1 mm, voltaje de operación de 2,6 V a 3,5 V, rango de temperatura de -20 a 70°C, emisor infrarrojo de 940 nm y comunicación I2C a hasta 400 kHz [5].

Anexo 5: Diseños de impresión 3D de los soportes

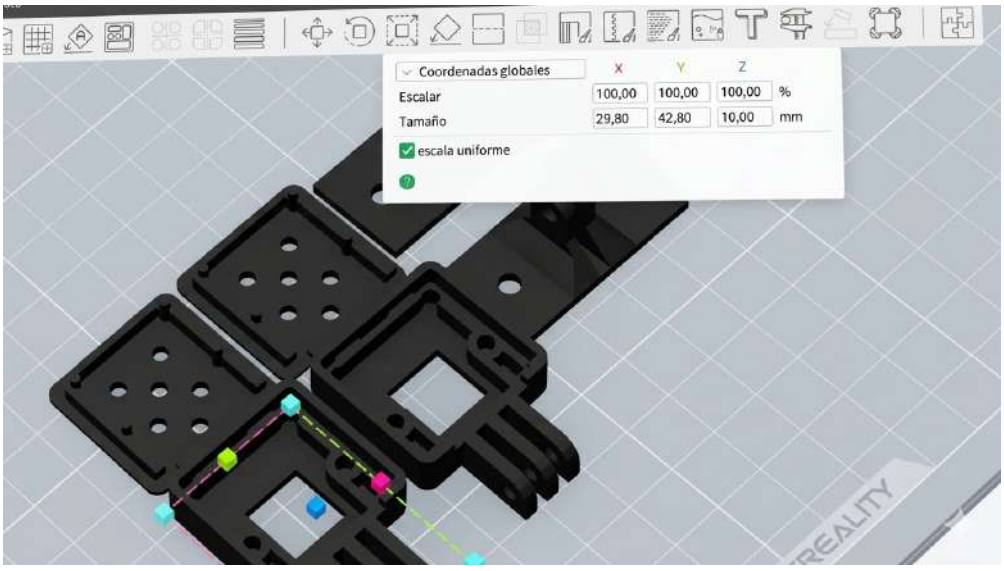


Figura. 1.44. Diseño del soporte para las cámaras CSI Pi Camera Module 3, hecho en el software Creality Print, con dimensiones de 29,80 × 42,80 × 10,00 mm. El diseño tiene ranuras de ventilación y un sistema de fijación con tornillos, pensado para sujetar la cámara a la carrocería del vehículo.

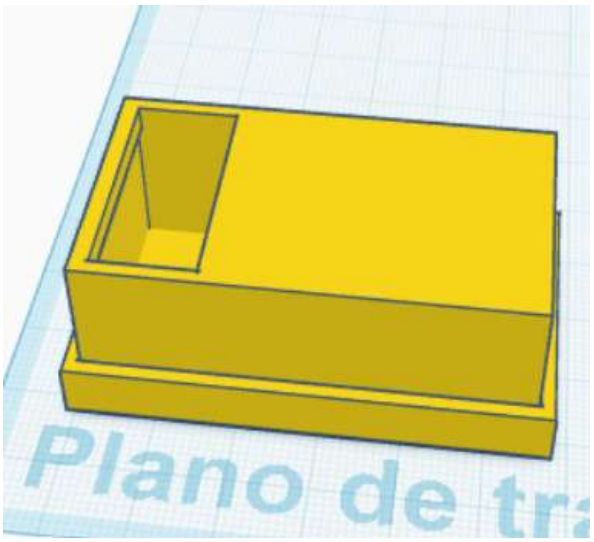


Figura. 1.45. Diseño del soporte para el sensor de distancia VL53L0X en Tinkercad, con una cavidad ajustada al encapsulado del sensor y una ranura frontal que permite la línea de visión directa del láser hacia el exterior del vehículo.

Anexo 6: Evidencias fotográficas de las pruebas de campo

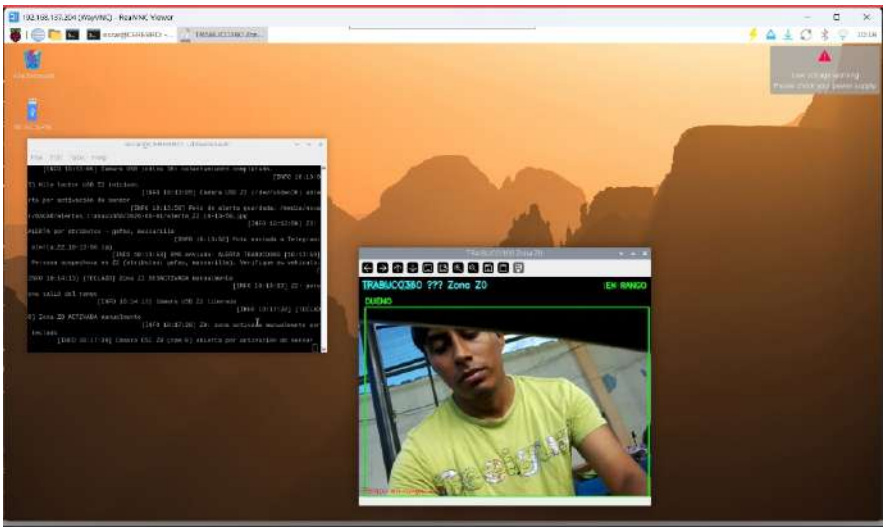


Figura. 1.46. Escenario 6 — Reconocimiento del propietario en Zona Z0. La etiqueta *DUEÑO* en color verde confirma que el sistema identificó correctamente el rostro del propietario y no generó ninguna alerta. Dado que en este escenario el sistema no dispara alerta, tampoco guarda fotografía automáticamente, por lo que se presenta una captura de pantalla como evidencia del funcionamiento correcto.

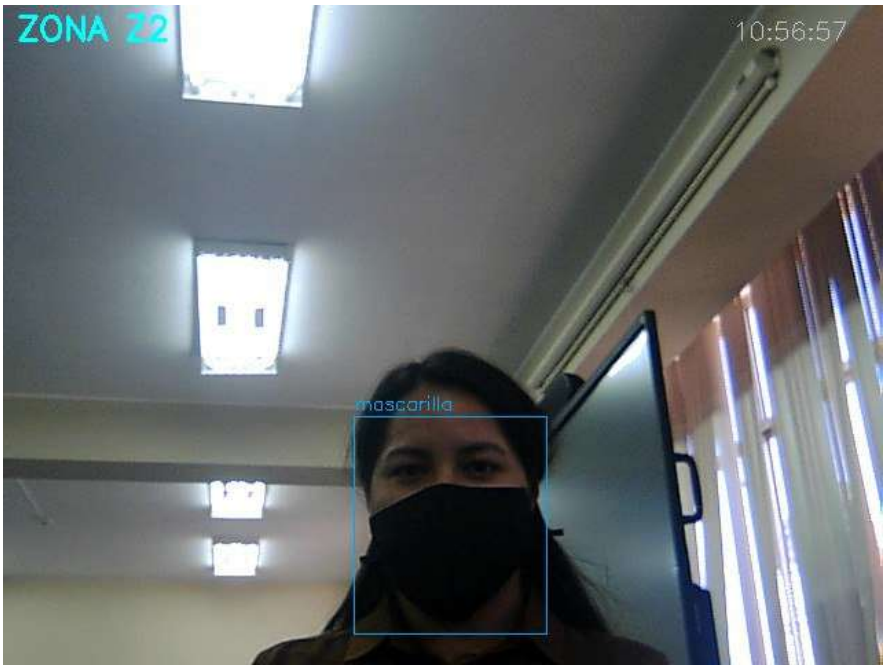


Figura. 1.47. Detección de atributo mascarilla en Zona Z2 a las 10:56:57. El sistema identificó el uso de mascarilla sobre el rostro de la persona y acumuló el atributo dentro de la ventana de 5 segundos para su posterior evaluación.

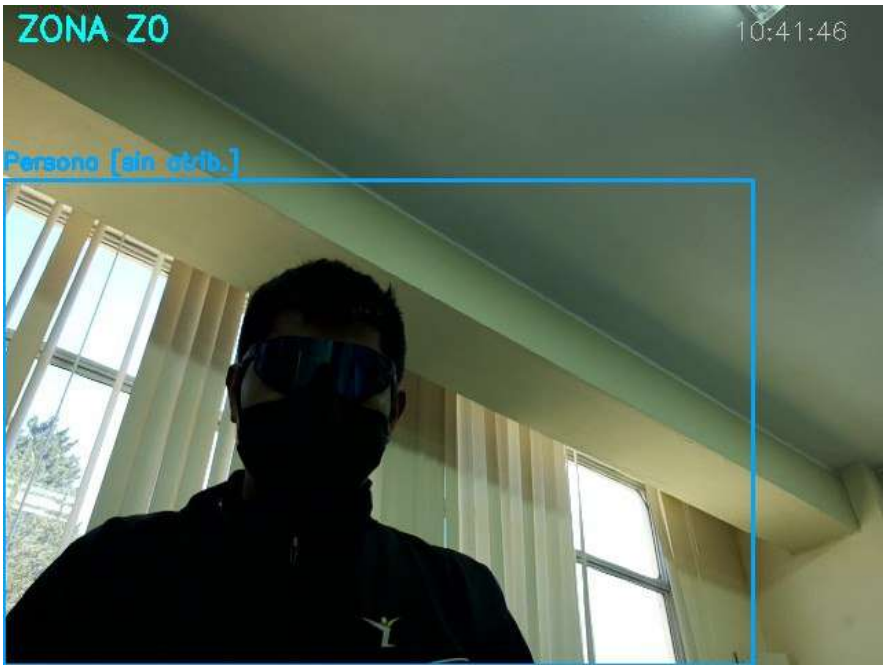


Figura. 1.48. Detección en condiciones de contraluz en la Zona Z0, a las 10:41:46. A pesar de la iluminación adversa, el sistema logró detectar la presencia de la persona y mostró la etiqueta *Persona [sin atrib.]*, lo que muestra que no se juntaron suficientes atributos para activar la alerta por ese mecanismo, quedando el temporizador de permanencia como respaldo.

Figura. 1.49. Detección al mismo tiempo de gorra y mascarilla en la Zona Z0. El sistema identificó ambos atributos de ocultamiento facial en la misma persona, superando así el umbral de dos atributos necesario para que se active la alerta por sospechoso.

Figura. 1.50. Detección de atributos en una escena con dos personas dentro de la Zona Z1, a las 10:34:00. El sistema identificó gorra y gafas en la persona de la izquierda, y mascarilla en la persona de la derecha, lo que muestra la capacidad del modelo para analizar a varias personas dentro de un mismo cuadro al mismo tiempo.

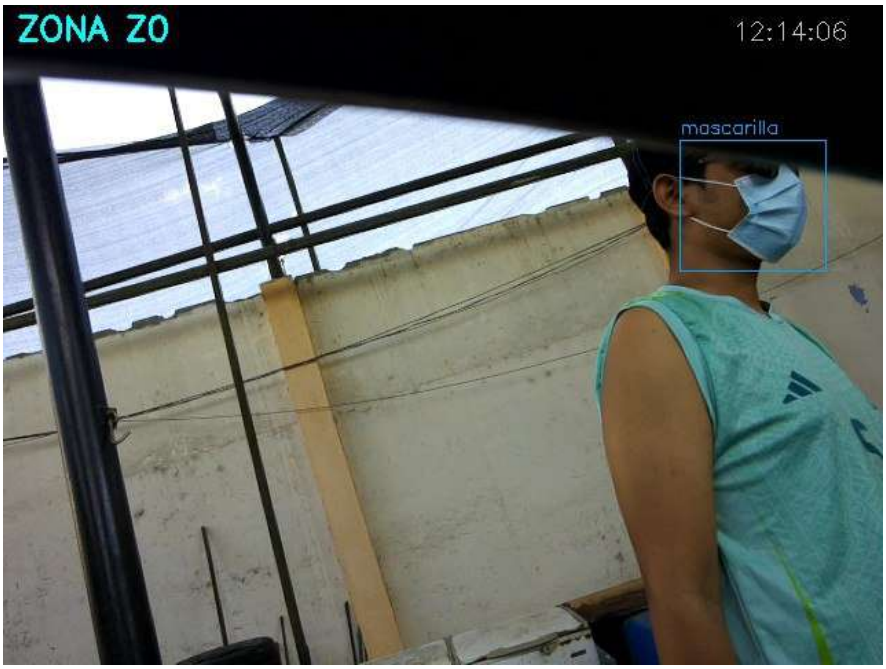


Figura. 1.51. Detección de mascarilla en un entorno exterior, desde la Zona Z0, a las 12:14:06. Esta prueba confirma que el sistema funciona bien en condiciones de luz natural, con la cámara apuntando hacia el exterior del vehículo.

ANEXO 7: Código fuente de funciones principales

Este anexo recoge las funciones más relevantes del sistema, desarrolladas en Python 3 sobre Raspberry Pi 5, organizadas de acuerdo con el módulo funcional al que pertenecen.

7.1 Inicialización de sensores VL53L0X

Esta función inicializa los cuatro sensores de distancia asignándoles direcciones I2C únicas mediante el control individual del pin XSHUT de cada sensor. Los sensores marcados como fallidos se omiten y su zona queda activable de forma manual por teclado.

```
PINES_XSHUT    = [4, 27, 22, 23]
DIRECCIONES_I2C = [0x30, 0x31, 0x32, 0x33]
SENSORES_FALLIDOS = [3]

def _xshut_low(pin):
    os.system(f"pinctrl set {pin} op dl")
    time.sleep(0.15)

def _xshut_high(pin):
    os.system(f"pinctrl set {pin} op dh")
    time.sleep(0.15)

```

```python
def inicializar_sensores() -> list:
    i2c = busio.I2C(board.SCL, board.SDA)
    lista = []

    # Apagar todos los sensores (XSHUT -> LOW)
    for pin in PINES_XSHUT:
        _xshut_low(pin)
    time.sleep(0.5)

    for i, pin in enumerate(PINES_XSHUT):
        if i in SENSORES_FALLIDOS:
            lista.append(None)
            log.info(f"Sensor Z{i} marcado como fallido")
            continue

        # Encender solo el sensor i (XSHUT -> HIGH)
        _xshut_high(pin)
        time.sleep(0.4)
        try:
            s = adafruit_vl53l0x.VL53L0X(i2c)
            s.set_address(DIRECCIONES_I2C[i])
            time.sleep(0.2)
            lista.append(s)
            log.info(f"Sensor Z{i} OK (addr={hex(DIRECCIONES_I2C[i])})")
        except Exception as e:
            log.warning(f"Sensor Z{i} ERROR: {e}")
            lista.append(None)
    return lista
```

Listing 1.1. Inicialización de sensores VL53L0X con direcciones I2C únicas

7.2 Calibración de distancia base

Al arrancar el sistema, esta función toma 30 lecturas de cada sensor para establecer la distancia normal de la escena con el vehículo presente. La activación ocurre cuando una persona se acerca y la distancia disminuye más del umbral configurado respecto a esa base.

```python
UMBRAL_CAMBIO_MM = 150

def calibrar_bases():
    log.info("Calibrando bases...")
    for i, sensor in enumerate(sensores):
        zona = f"Z{i}"
        if sensor is None:
            distancias_base[zona] = None
            continue
        lecturas = []
        for _ in range(30):
            d = leer_distancia(sensor)
```

```
            if d is not None:
                lecturas.append(d)
            time.sleep(0.1)
        if lecturas:
            # Percentil 20: ignora lecturas altas espurias
            distancias_base[zona] = int(np.percentile(lecturas, 20))
            log.info(f"  Z{i} base = {distancias_base[zona]} mm")
        else:
            distancias_base[zona] = None
            log.warning(f"  Z{i} sin lecturas validas")
```

Listing 1.2. Calibración de distancia base por zona

7.3 Predetección de movimiento con MOG2

Antes de ejecutar el modelo YOLO, se aplica una predetección rápida con el sustractor de fondo MOG2 de OpenCV. Si menos del 1 % de los píxeles del frame presentan movimiento, se descarta el frame y no se consume CPU en la inferencia.

```
def obtener_mog2(zona: str) -> cv2.BackgroundSubtractorMOG2:
    if zona not in _mog2:
        _mog2[zona] = cv2.createBackgroundSubtractorMOG2(
            history=200, varThreshold=40, detectShadows=False
        )
    return _mog2[zona]

def hay_movimiento_en_frame(frame_gris: np.ndarray, zona: str) -> bool:
    mog      = obtener_mog2(zona)
    mascara = mog.apply(frame_gris)
    pixeles_mov = np.count_nonzero(mascara)
    porcentaje  = pixeles_mov / mascara.size
    return porcentaje > 0.01    # >1% de pixeles en movimiento
```

Listing 1.3. Predetección de movimiento con MOG2

7.4 Extracción de atributos con YOLOv8n

Esta función ejecuta el modelo YOLOv8n sobre un frame redimensionado y devuelve el conjunto de atributos detectados (gorra, gafas, mascarilla) sin dibujar anotaciones. Es la función usada para la acumulación de atributos en la ventana de 5 segundos.

```
INFER_W, INFER_H    = 256, 256
VENTANA_DETECCION_S = 5.0
UMBRAL_SOSPECHA     = 2

def _extraer_atributos_frame(frame: np.ndarray, zona: str) -> set:
```

```
    if modelo_yolo is None:
        return set()

    frame_inf = cv2.resize(frame, (INFER_W, INFER_H))

    # Mutex global: YOLOv8 no es thread-safe en inferencia simultanea
    with _lock_modelo:
        resultados = modelo_yolo.predict(
            frame_inf,
            imgsz   = (INFER_H, INFER_W),
            conf    = 0.35,
            verbose = False,
            device  = "cpu",
        )

    idx_gorra      = CLASES_MODELO.get("gorra")
    idx_gafas      = CLASES_MODELO.get("gafas")
    idx_mascarilla = CLASES_MODELO.get("mascarilla")

    atribs = set()
    for r in resultados:
        for box in r.boxes:
            cls = int(box.cls[0])
            if cls == idx_gorra:
                atribs.add("gorra")
            elif cls == idx_gafas:
                atribs.add("gafas")
            elif cls == idx_mascarilla:
                atribs.add("mascarilla")
    return atribs
```

Listing 1.4. Extracción de atributos de ocultamiento facial con YOLOv8n

7.5 Reconocimiento facial del propietario

Esta función se encarga de verificar si el rostro visible en el frame corresponde al propietario registrado. Para acelerar el proceso, el frame se reduce a un cuarto de su tamaño antes de calcular los encodings faciales. Si la distancia entre el encoding detectado y los encodings ya registrados resulta menor al umbral de tolerancia establecido, se asume que se trata del dueño y, por lo tanto, no se genera ninguna alerta.

```
FACE_TOLERANCE = 0.5

def cargar_encodings_dueno() -> list:
    extensiones = {".jpg", ".jpeg", ".png", ".bmp", ".webp"}
    archivos = [
        f for f in os.listdir(CARPETA_DUENO)
        if os.path.splitext(f)[1].lower() in extensiones
```

```
    ]
    encodings = []
    for nombre in archivos:
        ruta = os.path.join(CARPETA_DUENO, nombre)
        try:
            img  = face_recognition.load_image_file(ruta)
            locs = face_recognition.face_locations(img, model="hog")
            if not locs:
                continue
            enc = face_recognition.face_encodings(img, locs)[0]
            encodings.append(enc)
        except Exception as e:
            log.warning(f"Error procesando '{nombre}': {e}")
    return encodings

def es_dueno(frame: np.ndarray) -> bool:
    if not dueno_encodings:
        return False
    h, w  = frame.shape[:2]
    scale = min(1.0, 320 / max(w, h))
    small = cv2.resize(frame, (0,0), fx=scale, fy=scale) \
            if scale < 1.0 else frame
    rgb   = cv2.cvtColor(small, cv2.COLOR_BGR2RGB)
    locs  = face_recognition.face_locations(rgb, model="hog")
    if not locs:
        return False
    encs = face_recognition.face_encodings(rgb, locs)
    for enc in encs:
        distancias = face_recognition.face_distance(dueno_encodings, enc)
        if distancias.min() <= FACE_TOLERANCE:
            return True
    return False
```

Listing 1.5. Reconocimiento facial del propietario registrado

7.6 Generación y envío de alertas

Las siguientes funciones gestionan el envío de alertas al propietario: guardado de la fotografía en la memoria USB, envío de SMS mediante comandos AT al módulo SIM7600G-H, envío de la fotografía por Telegram y envío de ubicación aproximada por geolocalización IP.

```
def guardar_foto_alerta(frame: np.ndarray, zona: str) -> str | None:
    try:
        fecha       = time.strftime("%Y-%m-%d")
        hora        = time.strftime("%H-%M-%S")
        carpeta_dia = os.path.join(CARPETA_ALERTAS, fecha)
        os.makedirs(carpeta_dia, exist_ok=True)
        nombre = f"alerta_{zona}_{hora}.jpg"
        ruta   = os.path.join(carpeta_dia, nombre)
```

```
        cv2.imwrite(ruta, frame, [cv2.IMWRITE_JPEG_QUALITY, 90])
        log.info(f"Foto guardada: {ruta}")
        return ruta
    except Exception as e:
        log.error(f"Error guardando foto: {e}")
        return None
```

Listing 1.6. Guardado de fotografía de alerta en memoria USB

```
def enviar_sms(mensaje: str):
    if sim_serial is None:
        log.warning(f"SMS no enviado (sin modem): {mensaje}")
        return
    try:
        sim_serial.write(
            f'AT+CMGS="{NUMERO_ALERTA}"\r\n'.encode()
        )
        time.sleep(0.5)
        # \x1A = Ctrl+Z cierra el mensaje SMS
        sim_serial.write(f"{mensaje}\x1A".encode())
        time.sleep(3)
        log.info(f"SMS enviado: {mensaje}")
    except Exception as e:
        log.error(f"Error enviando SMS: {e}")
```

Listing 1.7. Envío de SMS mediante módulo SIM7600G-H con comandos AT

```
def enviar_telegram(ruta_foto: str, mensaje: str):
    url_foto  = f"https://api.telegram.org/bot{TELEGRAM_TOKEN}/sendPhoto"
    url_texto = f"https://api.telegram.org/bot{TELEGRAM_TOKEN}/sendMessage"
    try:
        if ruta_foto and os.path.exists(ruta_foto):
            with open(ruta_foto, "rb") as f:
                requests.post(
                    url_foto,
                    data={
                        "chat_id"   : TELEGRAM_CHAT_ID,
                        "caption"   : mensaje,
                        "parse_mode": "HTML",
                    },
                    files={"photo": f},
                    timeout=15,
                )
        else:
            requests.post(
                url_texto,
                data={"chat_id": TELEGRAM_CHAT_ID, "text": mensaje},
                timeout=10,
            )
    except Exception as e:
        log.error(f"Error enviando a Telegram: {e}")
```

Listing 1.8. Envío de fotografía y mensaje por Telegram Bot API

```
def enviar_ubicacion_telegram():
    url_location = f"https://api.telegram.org/bot{TELEGRAM_TOKEN}/sendLo
    cation"
    url_mensaje  = f"https://api.telegram.org/bot{TELEGRAM_TOKEN}/sendMe
    ssage"
    try:
        resp = requests.get(
            "http://ip-api.com/json/?fields=status,lat,lon,"
            "city,regionName,country,isp,query",
            timeout=10,
        )
        data = resp.json()
        if data.get("status") != "success":
            return

        lat, lon = data["lat"], data["lon"]
        requests.post(url_location, data={
            "chat_id"  : TELEGRAM_CHAT_ID,
            "latitude" : lat,
            "longitude": lon,
        }, timeout=10)

        texto = (
            f"Ubicacion aproximada del sistema\n"
            f"{data.get('city')}, {data.get('regionName')}, "
            f"{data.get('country')}\n"
            f"IP publica: {data.get('query')}\n"
            f"Precision: ~500m-2km (geolocalizacion por IP 4G)"
        )
        requests.post(url_mensaje, data={
            "chat_id"   : TELEGRAM_CHAT_ID,
            "text"      : texto,
            "parse_mode": "HTML",
        }, timeout=10)
    except Exception as e:
        log.error(f"Error enviando ubicacion: {e}")
```

Listing 1.9. Envío de ubicación aproximada por geolocalización IP

```
def _disparar_alertas(frame: np.ndarray, zona: str, msg: str):
    try:
        ruta = guardar_foto_alerta(frame, zona)
        threading.Thread(
            target=enviar_sms, args=(msg,), daemon=True
        ).start()
        threading.Thread(
            target=enviar_telegram, args=(ruta, msg), daemon=True
        ).start()
```

```
        threading.Thread(
            target=enviar_ubicacion_telegram, daemon=True
        ).start()
    except Exception as e:
        log.error(f"Error disparando alertas: {e}")
```

Listing 1.10. Disparo conjunto de alertas (foto, SMS, Telegram y ubicación)

7.7 Lógica principal del hilo por zona

Esta es la función central del sistema. Cada zona ejecuta este hilo de forma independiente. Implementa los dos mecanismos de alerta: acumulación de atributos en ventana de 5 segundos y alerta por permanencia prolongada mayor a 60 segundos.

```
TIEMPO_MAX_EN_RANGO_S = 60.0
COOLDOWN_ALERTA       = 60
SALTO_FRAMES          = 2
WARMUP_MOG2           = 30

def procesar_zona(i: int):
    zona            = f"Z{i}"
    sensor          = sensores[i] if i < len(sensores) else None
    ultimo_alerta[zona] = 0
    contador_frames = 0

    t_entrada_rango     = None
    alerta_permanencia  = False

    atribs_ventana = set()
    t_ventana      = None

    while not salir.is_set():
        ahora = time.time()

        dist_actual = None
        en_rango    = False

        if sensor is not None:
            d = leer_distancia(sensor)
            if d is not None:
                dist_actual = d
                base = distancias_base.get(zona)
                if base is not None:
                    cambio = base - d
                    en_rango = (cambio > UMBRAL_CAMBIO_MM
                                and d <= UMBRAL_PRESENCIA_MM)
        else:
            with _lock_zonas_forzadas:
                en_rango = zonas_forzadas.get(zona, False)
```

```
36
37          if not en_rango:
38              t_entrada_rango    = None
39              alerta_permanencia = False
40              atribs_ventana.clear()
41              t_ventana = None
42              _cerrar_camara_zona(zona)
43              time.sleep(0.1)
44              continue
45
46          _abrir_camara_zona(zona)
47          if t_entrada_rango is None:
48              t_entrada_rango = ahora
49
50          tiempo_en_rango = ahora - t_entrada_rango
51          if (tiempo_en_rango >= TIEMPO_MAX_EN_RANGO_S
52                  and not alerta_permanencia
53                  and ahora - ultimo_alerta[zona] > COOLDOWN_ALERTA):
54              alerta_permanencia = True
55              ultimo_alerta[zona] = ahora
56              frame_alerta = leer_frame_zona(zona)
57              if frame_alerta is not None:
58                  msg = (f"ALERTA TRABUCO360 - Zona {zona}: "
59                         f"Persona mas de {int(TIEMPO_MAX_EN_RANGO_S)}s "
60                         f"en rango. Verifique su vehiculo.")
61                  _disparar_alertas(frame_alerta, zona, msg)
62
63          contador_frames += 1
64          if contador_frames % SALTO_FRAMES != 0:
65              time.sleep(0.03)
66              continue
67
68          frame = leer_frame_zona(zona)
69          if frame is None:
70              time.sleep(0.05)
71              continue
72
73          frame_gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
74
75          if contador_frames <= WARMUP_MOG2:
76              hay_movimiento_en_frame(frame_gris, zona)
77              continue
78
79          if not hay_movimiento_en_frame(frame_gris, zona):
80              continue
81
82          nuevos = _extraer_atributos_frame(frame, zona)
83          if nuevos:
84              if t_ventana is None:
85                  t_ventana = ahora
86              atribs_ventana |= nuevos
```

```
87
88          if t_ventana is not None:
89              if ahora - t_ventana > VENTANA_DETECCION_S:
90                  if (len(atribs_ventana) >= UMBRAL_SOSPECHA
91                          and ahora - ultimo_alerta[zona] > COOLDOWN_ALERTA):
92                      ultimo_alerta[zona] = ahora
93                      msg = (f"ALERTA TRABUCO360 - Zona {zona}: "
94                             f"Persona sospechosa detectada "
95                             f"({', '.join(sorted(atribs_ventana))}). "
96                             f"Verifique su vehiculo.")
97                      _disparar_alertas(frame, zona, msg)
98                  atribs_ventana.clear()
99                  t_ventana = None
100
101         time.sleep(0.03)
102
103     _cerrar_camara_zona(zona)
104     log.info(f"Hilo {zona} finalizado.")
```

Listing 1.11. Lógica principal del hilo de procesamiento por zona