

UNIVERSIDAD CATÓLICA DE CUENCA

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

FACULTAD DE INGENIERÍA E INDUSTRIA

CARRERA DE ELECTRICIDAD

SISTEMA DE SUPERVISIÓN AUTOMATIZADA PARA EL RECONOCIMIENTO Y CONTEO TAXONÓMICO DE VEHÍCULOS MEDIANTE ALGORITMOS DE VISIÓN ARTIFICIAL

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELÉCTRICO

AUTOR: JOSE ANDRES AGREDA ORDOÑEZ

DIRECTOR: PABLO DANILO ARIAS REYES

CO-DIRECTOR: JUAN CARLOS COBOS TORRES

CUENCA – ECUADOR

2026

DIOS, PATRIA, CULTURA Y DESARROLLO



UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

FACULTAD DE INGENIERÍA E INDUSTRIA

CARRERA DE ELECTRICIDAD

SISTEMA DE SUPERVISIÓN AUTOMATIZADA PARA EL RECONOCIMIENTO Y CONTEO TAXONÓMICO DE VEHÍCULOS MEDIANTE ALGORITMOS DE VISIÓN ARTIFICIAL

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELÉCTRICO

AUTOR: JOSÉ ANDRES AGREDA ORDOÑEZ

DIRECTOR: PABLO DANILO ARIAS REYES

CO-DIRECTOR: JUAN CARLOS COBOS TORRES

CUENCA - ECUADOR

2026

DIOS, PATRIA, CULTURA Y DESARROLLO

DECLARATORIA DE AUTORIA Y RESPONSABILIDAD

Jose Andres Agreda Ordoñez portador de la cédula de ciudadanía Nº1900752435. Declaro ser el autor de la obra: "Sistema de supervisión automatizada para el reconocimiento y conteo taxonómico de vehículos mediante algoritmos de visión artificial", sobre la cual me hago responsable sobre las opiniones, versiones e ideas expresadas. Declaro que la misma ha sido elaborada respetando los derechos de propiedad intelectual de terceros y eximo a la Universidad Católica de Cuenca sobre cualquier reclamación que pudiera existir al respecto. Declaro finalmente que mi obra ha sido realizada cumpliendo con todos los requisitos legales, éticos y bioéticos de investigación, que la misma no incumple con la normativa nacional e internacional en el área específica de investigación, sobre la que también me responsabilizo y eximo a la Universidad Católica de Cuenca de toda reclamación al respecto.

Cuenca, 20 de octubre de 2026



F: ……..

Jose Andres Agreda Ordoñez

1900752435

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por José Andres Agreda Ordoñez, bajo mi supervisión.



……

Ing. Pablo Danilo Arias, Mgts.

DIRECTOR

DEDICATORIA

Dedico este trabajo, en primer lugar, a Dios, por brindarme la fortaleza, sabiduría y perseverancia necesarias para superar cada desafío y culminar esta importante etapa de mi vida.

A mis padres y a mi familia, por su amor, apoyo y confianza incondicional. Gracias por acompañarme en cada momento, por motivarme a seguir adelante y por ser una fuente constante de inspiración para alcanzar mis metas.

A todas las personas que estuvieron presentes durante este camino y que, con su apoyo y palabras de aliento, contribuyeron de alguna manera a la culminación de este trabajo.

AGRADECIMIENTOS

Expreso mi más sincero agradecimiento a Dios, por brindarme la fortaleza, la salud y la sabiduría necesarias para culminar satisfactoriamente esta etapa de formación profesional.

A mi familia, especialmente a mis padres, por su amor, apoyo y confianza incondicional. Gracias por acompañarme durante todo este proceso, por motivarme a seguir adelante y por ser un pilar fundamental para alcanzar esta meta.

A la Universidad Católica de Cuenca, por brindarme la oportunidad de formarme profesionalmente y por los conocimientos adquiridos a lo largo de mi carrera, los cuales han sido fundamentales para mi crecimiento académico y profesional.

De manera especial, agradezco a mi director de tesis, Ing. Pablo Danilo Arias, Mgts., por su orientación, acompañamiento y valiosas observaciones durante el desarrollo de este trabajo.

Asimismo, expreso mi agradecimiento a mi co-director, Ing. Juan Carlos Cobos, Ph.D., por sus aportes, orientación y apoyo durante el desarrollo de la investigación.

A todas las personas que, de una u otra manera, contribuyeron con sus conocimientos, colaboración y palabras de aliento durante este proceso, mi más sincero agradecimiento.

Finalmente, quiero expresar mi agradecimiento al Laboratorio de Robótica, Automatización, Sistemas Inteligentes y Embebidos (RobLab) y al proyecto de investigación “Sistema Inteligente de Monitoreo de la Calidad del Aire en Iglesias Patrimoniales”, código PICIDN25-20 por brindarme el espacio, los recursos y el apoyo necesarios para poder desarrollar esta investigación.

RESUMEN

El crecimiento del parque automotor y la necesidad de contar con sistemas confiables de supervisión vehicular contrastan con los altos costos de las soluciones comerciales, lo que motivó a esta investigación a desarrollar e implementar un sistema automatizado de bajo costo para el reconocimiento y conteo taxonómico de vehículos mediante algoritmos de visión artificial, para lo cual se diseñó una arquitectura IoT basada en un nodo ESP32-CAM con módulo W5500 para la transmisión de video por Ethernet hacia un servidor local, donde un modelo YOLO con técnicas de seguimiento y una línea virtual clasificó y contabilizó automóviles, buses, camiones y motocicletas, evaluando el sistema durante cuatro semanas en tres puntos del campus matriz de la Universidad Católica de Cuenca, dos de ellos con alimentación fotovoltaica autónoma; la validación, contrastando 11 998 vehículos contabilizados automáticamente frente a 13 966 vehículos de aforo manual, mostró una concordancia global del 85,91 % y F1-score entre 75,96 % y 86,08 %, con el mejor desempeño en automóviles (12,48 % de error) y el menor en motocicletas (22,38 %), mientras que la perspectiva lateral de la cámara resultó determinante en el error del sector Basílica, el costo total fue de $412,62 USD, equivalente a $137,54 USD por punto, significativamente menor al de los sistemas comerciales radares, se concluye que la integración de nodos embebidos de bajo costo, procesamiento centralizado y energía solar constituye una alternativa técnica y económicamente viable para la supervisión del tráfico urbano.

Palabras clave: visión artificial; conteo vehicular; YOLO; Internet de las Cosas; sistemas fotovoltaicos.

ABSTRACT

The growth of the vehicle fleet and the need for reliable vehicle monitoring systems contrast with the high costs of commercial solutions. This motivated the present study to develop and implement a low-cost automated system for vehicle recognition and classification and counting by vehicle type using computer vision algorithms. To this end, an IoT architecture was designed based on an ESP32-CAM node with a W5500 module for transmitting video over Ethernet to a local server, where a YOLO (You Only Look Once) model incorporating tracking techniques and a virtual line was used to classify and count cars, buses, trucks, and motorcycles. The system was evaluated over four weeks at three locations on the main campus of the Catholic University of Cuenca, two of which were powered by standalone photovoltaic systems. Validation, based on a comparison of 11,998 automatically counted vehicles with 13,966 vehicles recorded through manual counting, showed an overall agreement of 85.91% and F1-scores ranging from 75.96% to 86.08%. The best performance was achieved for cars, with an error rate of 12.48%, while motorcycles showed the highest error rate, at 22.38%. Meanwhile, the camera's lateral viewing angle proved to be a determining factor in the error observed in the Basílica sector. The total cost was USD 412.62, equivalent to USD 137.54 per monitoring point, significantly lower than the cost of commercial radar-based systems. It is concluded that integrating low-cost embedded nodes, centralized processing, and solar power constitutes a technically and economically viable alternative for urban traffic monitoring.

Keywords: computer vision, vehicle counting, YOLO, Internet of Things, photovoltaic systems.

ÍNDICE GENERAL

DECLARATORIA DE AUTORIA Y RESPONSABILIDAD I
CERTIFICACIÓN II
DEDICATORIA III
AGRADECIMIENTOS IV
RESUMEN V
ABSTRACT VI
CAPÍTULO I 12
1.1 Introducción 12
1.2 Problema 14
1.2.1 Formulación del problema 14
1.2.2 Delimitación del problema 15
1.3 Justificación 16
1.4 Objetivos 17
1.4.1 Objetivo General 17
1.4.2 Objetivos Específicos 17
CAPÍTULO II 19
2. Fundamentación Teórica 19
2.1 Conceptos de tránsito y flujo vehicular 19
2.1.1 Parámetros para la medición de flujo vehicular 19
2.1.2 Sistemas de conteo vehicular: métodos manuales y automáticos 20
2.1.3 Sistemas Inteligentes de Transporte (ITS) 21
2.2 Visión artificial 22
2.2.1 Fundamentos de visión artificial 22
2.2.2 Reconocimiento y detección de imágenes 23
2.2.3 Aprendizaje automático y aprendizaje profundo 24
2.2.4 Tecnologías y bibliotecas para procesamiento de imágenes 25
2.2.5 Modelos para detección vehicular 26
2.2.6 Seguimiento y conteo multiobjeto (tracking) 28
2.3 Internet de las Cosas (IoT) 29
2.3.1 Integración de IoT y Aprendizaje Profundo para el Monitoreo Vehicular 30
2.3.2 Arquitecturas IoT 30
2.3.3 ESP32-CAM 31
2.3.4 Redes de comunicación para sistemas IoT 32
2.4 Eficiencia energética en sistemas IoT y Edge AI 33
2.4.1 Consumo energético en sistemas de embebidos de visión artificial 34

2.4.2 Estrategias de eficiencia energética en sistemas embebidos basados en ESP32-CAM 34
2.4.3 Eficiencia y sostenibilidad energética en sistemas IoT 35
2.5 Sistemas fotovoltaicos autónomos 36
2.5.1 Radiación solar 36
2.5.2 Sistemas fotovoltaicos fuera de red 37
2.5.3 Componentes de un sistema fotovoltaico autónomo 37
CAPÍTULO III 39
3. Metodología 39
3.1 Enfoque de la investigación 39
3.2 Desarrollo de la propuesta 39
CAPÍTULO IV 42
4. Desarrollo 42
4.1 Componentes electrónicos y de comunicación del sistema 43
4.2 Dimensionamiento fotovoltaico 46
4.2.1 Estimación del consumo eléctrico 46
4.2.2 Demanda energética diaria 48
4.2.3 Dimensionamiento del panel fotovoltaico 50
4.2.4 Dimensionamiento del banco de baterías 52
4.3 Diseño de la placa PCB de interconexión ESP32 CAM– Módulo W5500 54
4.3.1 Diseño Esquemático 54
4.3.2 Diseño PCB 56
4.4 Diseño del compartimiento para los componentes electrónicos 56
4.4.1 Herramientas de software utilizadas 58
4.5 Implementación del Sistema 59
4.5.1 Fabricación y ensamblaje de los componentes 59
4.5.2 Fabricación de las estructuras de soporte 61
4.5.3 Integración y pruebas del sistema fotovoltaico 63
4.5.4 Funcionamiento del sistema 64
4.5.5 Instalación en los puntos de monitoreo 68
CAPÍTULO V 71
5. Análisis de Resultados 71
5.1 Conteo mediante modelo YOLO 71
5.1.1 Resultados consolidados del conteo vehicular mediante YOLO 72
5.2 Conteo manual de referencia 73
5.2.1 Resultados consolidados del conteo vehicular manual 74

5.3 Comparación de los resultados del conteo vehicular 74
5.4 Variación temporal del error de conteo según semana y día. 78
5.5 Evaluación mediante métricas de detección. 82
5.6 Estimación de Costos 85
5.7 Discusión 87
CAPÍTULO VI 90
6. Conclusiones y Recomendaciones 90
6.1 Conclusiones 90
6.2 Recomendaciones 92
7. Referencias 94
AUTORIZACION DE PUBLICACION EN EL REPOSITORIO INSTITUCIONAL 132

ÍNDICE DE TABLAS

Tabla 1 Comparación de métodos utilizados para conteo vehicular 20
Tabla 2 Principales bibliotecas y herramientas para el procesamiento de imágenes y detección de objetos 26
Tabla 3 Principales versiones de YOLO aplicadas a la detección vehicular. 27
Tabla 4 Comparación del desempeño de diferentes modelos YOLO. 28
Tabla 5 Especificaciones de componentes eléctricos 44
Tabla 6 Estimación del consumo de corriente y potencia del nodo según la condición de operación 48
Tabla 7 Demanda energética diaria del nodo 49
Tabla 8 Parámetros y producción estimada mediante PVGIS 50
Tabla 9 Parámetros y producción estimada mediante Global Solar Atlas 50
Tabla 10 Verificación del panel fotovoltaico seleccionado 52
Tabla 11 Conexiones entre el ESP32-CAM y el módulo W5500. 55
Tabla 12 Herramientas de software utilizadas 58
Tabla 13 Características de los puntos de monitoreo 68
Tabla 14 Resultados del conteo vehicular mediante el algoritmo YOLO. 72
Tabla 15 Total de vehículos contabilizados por clasificación mediante YOLO. 72
Tabla 16 Resultados del conteo manual de vehículos 73
Tabla 17 Conteo manual por clasificación y ubicación. 74
Tabla 18 Comparación del conteo vehicular por ubicación. 75
Tabla 19 Comparación del conteo vehicular por clasificación. 76
Tabla 20 Error porcentual del conteo automático por semana y ubicación. 78
Tabla 21 Error porcentual del conteo automático según semana, día y ubicación 80
Tabla 22 Valores de TP, FP y FN por punto de monitoreo. 84
Tabla 23 Métricas de evaluación del sistema de detección. 84
Tabla 24 Presupuesto para la ejecución del trabajo de titulación 86

ÍNDICE DE FIGURAS

Figura 1 Sistemas ITS 22
Figura 2 Detección de objetos mediante visión por computadora 24
Figura 3 Relación entre inteligencia artificial, aprendizaje automático, aprendizaje profundo, redes neuronales convolucionales y YOLO. 25
Figura 4 Proceso de seguimiento de vehículos entre fotogramas mediante detección de objetos 29
Figura 5 Comparación de las arquitecturas Cloud, Edge y Fog Computing en sistemas IoT. 31
Figura 6 ESP32CAM: (a) Vista frontal; (b) Vista posterior 32
Figura 7 Características de las tecnologías de comunicación utilizadas en sistemas IoT. 33
Figura 8 Estrategias de eficiencia energética en ESP32-CAM. 34
Figura 9 Eficiencia y sostenibilidad energética en sistemas IoT embebidos. 35
Figura 10 Diagrama de flujo de la metodología empleada. 41
Figura 11 Diagrama de conexiones eléctricas y de comunicación del nodo. 45
Figura12 Esquema electrónico de interconexión entre el ESP32-CAM y el módulo W5500. 55
Figura 13 Diseño de la placa PCB: (a) trazado de la placa PCB, (b) esquema eléctrico de los componentes, (c) Ruteado de PCB 56
Figura 14 Diseño 3D de la carcasa: (a) Despiece de la carcasa y tapa frontal; (b) Disposición interna del controlador de carga solar y módulos electrónicos. 57
Figura 15 Diseño de las piezas de sujeción para el tubo de soporte 57
Figura 16 Modelo 3D del sistema ensamblado: (a) vista frontal y (b) vista posterior. 58
Figura 17 Placa PCB fabricada: (a) placa PCB, (b) montaje de la ESP32-CAM y el W5500. 60
Figura 18 Componentes de la carcasa fabricados en ABS. 60
Figura 19 Prototipo ensamblado. 60
Figura 20 Diseño del soporte para el panel fotovoltaico. 61
Figura 21 Diseño del tubo en forma de L para la fijación de la carcasa. 62
Figura 22 Estructuras metálicas fabricadas para el montaje del panel fotovoltaico y (b) la carcasa 63
Figura 23 (a) Integración y (b) pruebas del sistema fotovoltaico con el prototipo. 64
Figura 24 Esquema general del sistema 65
Figura 25 Diagrama de flujo del programa implementado en la ESP32-CAM y el módulo W5500. 66
Figura 26 Diagrama de flujo del programa de procesamiento y conteo vehicular desarrollado en Python. 67
Figura 27 Instalación de los sistemas de monitoreo en los tres puntos establecidos: (a) Basílica, (b) Facultad de Ingeniería y (c) Posgrados. 69
Figura 28. Funcionamiento del sistema de conteo vehicular en los puntos de monitoreo: (a) Basílica, (b) Facultad de Ingeniería, Industria y Construcción y (c) Dirección de Posgrado. 69
Figura 29 Error porcentual del conteo vehicular según clasificación y punto de monitoreo. 77
Figura 30 Evolución semanal del error porcentual según punto de monitoreo. 79
Figura 31 Variación diaria del error porcentual por semana, día y punto de monitoreo. 81

CAPÍTULO I

1.1 Introducción

El crecimiento acelerado de las ciudades y el aumento constante del parque automotor han generado nuevos desafíos relacionados con la movilidad urbana, la congestión vehicular y la gestión eficiente de la infraestructura vial. A nivel mundial, la cantidad de vehículos en circulación superó los 1400 millones en 2021, evidenciando una tendencia creciente, que incrementa la demanda de infraestructura y de herramientas para el monitoreo del tránsito (IEA, 2022). En América Latina, este fenómeno ha mostrado una expansión sostenida durante las últimas décadas, posicionándose entre las regiones con mayor crecimiento de la motorización (Viscidi & O'Connor, 2017). De manera similar, en Ecuador el parque automotor presenta una tendencia creciente, registrando una tasa media anual cercana al 7,5 % durante el período 2014-2023 (Hidalgo, 2024). Particularmente en la ciudad de Cuenca, la concentración de actividades académicas, comerciales y de servicios genera una elevada demanda de desplazamientos urbanos, produciendo variaciones significativas en el flujo vehicular (Parra & Torres, 2017).

En este contexto, la disponibilidad de información sobre el comportamiento del flujo vehicular constituye un elemento fundamental para apoyar la planificación de la movilidad urbana y la gestión eficiente de la infraestructura vial (Caiza, 2019). Como respuesta a esta necesidad, los Sistemas Inteligentes de Transporte (ITS) han incorporado tecnologías de información y comunicación capaces de recopilar, procesar y analizar datos de tráfico en tiempo real, contribuyendo a optimizar la operación de la red vial y fortalecer los procesos de toma de decisiones en entornos urbanos.

Además, los avances en visión artificial y aprendizaje profundo han favorecido el desarrollo de sistemas capaces de detectar y clasificar vehículos de manera automática a partir de imágenes y secuencias de video (Lashkov et al., 2025). Bajo esta premisa, los

modelos de la familia YOLO se han consolidado como una de las principales herramientas para aplicaciones de monitoreo vehicular debido a su capacidad para realizar detección, seguimiento y conteo en tiempo real. Investigaciones recientes reportan precisiones superiores al 94 % en tareas de clasificación vehicular y valores cercanos al 96 % en procesos de conteo automático mediante arquitecturas basadas en YOLOv8, evidenciando su potencial para aplicaciones de movilidad inteligente y análisis del tráfico urbano (Pudaruth et al., 2024). De manera similar, estudios comparativos han mostrado que estos modelos pueden superar el 97 % de precisión en tareas de detección y seguimiento vehicular, consolidándose como una alternativa eficiente para sistemas de supervisión automatizada del tránsito (Borse et al., 2024).

A partir de ello, la implementación de sistemas de monitoreo inteligentes ha incrementado considerablemente el interés por adquirir arquitecturas capaces de operar con bajos requerimientos energéticos y elevada autonomía. De esta manera, la integración de tecnologías IoT con sistemas de generación fotovoltaica ha permitido desarrollar soluciones de monitoreo de bajo costo y supervisión remota, capaces de adquirir y visualizar en tiempo real variables asociadas al funcionamiento del sistema energético (Hamied et al., 2023). Por lo tanto, los sistemas fotovoltaicos autónomos constituyen una alternativa viable para alimentar dispositivos electrónicos distribuidos, reduciendo la dependencia de la red eléctrica convencional y facilitando su despliegue en ubicaciones con acceso limitado a infraestructura energética (Sangprasert & Inthavisas, 2025). Estudios recientes evidencian que la combinación de plataformas embebidas de bajo consumo y generación fotovoltaica favorece la implementación de sistemas de monitoreo de bajo costo con supervisión en tiempo real y mínima intervención humana (Alhasnawi et al., 2024; Hamied et al., 2023). En consecuencia, la convergencia entre visión artificial, Internet de las Cosas y energía solar fotovoltaica constituye una alternativa prometedora

para el desarrollo de sistemas de monitoreo vehicular autónomos y energéticamente eficientes.

Con base en lo expuesto, la presente investigación planteó el desarrollo de una solución tecnológica orientada a la obtención automatizada de información vehicular mediante técnicas de visión artificial y procesamiento inteligente de imágenes. La propuesta buscó generar registros confiables, clasificación y conteo de vehículos, contribuyendo a la disponibilidad de datos útiles para el análisis de la movilidad urbana. Paralelamente, se incorporó una arquitectura IoT de bajo consumo energético alimentada mediante energía solar fotovoltaica, con el propósito de evaluar su capacidad de operación autónoma y su viabilidad como alternativa sostenible para aplicaciones de monitoreo inteligente en entornos urbanos.

1.2 Problema

1.2.1 Formulación del problema

La disponibilidad de información precisa sobre el flujo vehicular constituye un elemento fundamental para la gestión de la movilidad urbana y la planificación de intervenciones en infraestructura vial. El conteo vehicular permite conocer el volumen y la composición del tránsito, proporcionando datos necesarios para analizar las condiciones de circulación y apoyar la toma de decisiones relacionadas con el transporte (Caiza, 2019). Sin embargo, la obtención de esta información continúa dependiendo en gran medida de procedimientos manuales y campañas de aforo puntuales, lo que limita la disponibilidad de registros continuos y actualizados del comportamiento vehicular (Basantes & Gaibor, 2025).

Aunque existen sistemas de monitoreo basados en sensores y otras tecnologías de detección, su implementación suele requerir infraestructura específica, costos de

instalación y mantenimiento elevados, así como condiciones operativas que dificultan ampliar la cobertura del monitoreo en múltiples puntos de observación (Febres-Loaiza et al., 2024). De manera similar, los métodos convencionales presentan restricciones para obtener información permanente del tránsito sin una intervención constante de recursos humanos y técnicos (Rani & Dalal, 2024).

En este escenario, persiste la necesidad de disponer de soluciones capaces de automatizar la adquisición de información vehicular y operar de manera continua en condiciones reales de funcionamiento. No obstante, la implementación de sistemas de monitoreo distribuidos requiere considerar aspectos relacionados con la comunicación de datos, la disponibilidad energética y la sostenibilidad operativa del sistema, particularmente cuando se busca reducir la dependencia de infraestructura convencional y favorecer esquemas de operación autónoma.

1.2.2 Delimitación del problema

La presente investigación se desarrolló en el campus matriz de la Universidad Católica de Cuenca, ubicado en la ciudad de Cuenca, provincia del Azuay. La evaluación del sistema se realizó en tres puntos de monitoreo vehicular correspondientes a la Basílica de la Santísima Trinidad (Av. de las Américas y Humboldt), la Facultad de Ingeniería, Industria y Construcción (Av. de las Américas y General Torres) y la Dirección de Posgrados (vía a Patamarca y Cojímes), donde se llevó a cabo la captura de información vehicular para las pruebas de funcionamiento de la arquitectura propuesta.

La investigación comprendió las etapas de diseño, implementación y evaluación del sistema durante el periodo de ejecución del proyecto. La recolección de información vehicular se realizó durante un periodo de cuatro semanas, mediante registros efectuados entre las 06h00 y las 18h00 en los tres puntos de monitoreo establecidos. A partir de las

grabaciones obtenidas se seleccionaron posteriormente los fragmentos utilizados para la evaluación del sistema.

Desde el punto de vista tecnológico, el estudio se enfocó en el desarrollo de un sistema de supervisión automatizada para el reconocimiento, clasificación y conteo taxonómico de vehículos mediante algoritmos de visión artificial. La arquitectura integra un nodo IoT basado en un módulo ESP32-CAM con comunicación Ethernet y alimentación mediante un sistema fotovoltaico, destinado a la captura y transmisión de video hacia un servidor; el cual, mediante una plataforma de procesamiento, realizó la detección y el conteo de vehículos. El alcance de la investigación se limitó a la adquisición de imágenes, reconocimiento, clasificación taxonómica y conteo automático de vehículos, así como a la evaluación del desempeño del sistema.

1.3 Justificación

El desarrollo de sistemas automatizados basados en visión artificial representa una alternativa para la obtención de información cuantificable sobre el flujo vehicular, al permitir integrar procesos de reconocimiento, clasificación y conteo sin requerir una intervención permanente durante la adquisición de datos. La incorporación de estas técnicas en plataformas embebidas de bajo costo amplía las posibilidades de implementar soluciones de supervisión vehicular con menores requerimientos de infraestructura y capacidad de operación durante periodos prolongados. En este contexto, la integración de plataformas IoT, hardware de bajo costo y herramientas de software de código abierto representa una alternativa viable para el desarrollo de sistemas automatizados orientados a la adquisición y procesamiento de información sobre el flujo vehicular. Desde el ámbito tecnológico, los avances en visión artificial y aprendizaje profundo permiten implementar modelos capaces de detectar y clasificar diferentes tipos de vehículos a partir de imágenes y secuencias de video. Su integración con plataformas embebidas favorece su

implementación en varios puntos de monitoreo. En consecuencia, el desarrollo de una solución basada en ESP32-CAM para la adquisición de imágenes y algoritmos de visión artificial para su procesamiento permitió explorar una alternativa tecnológica orientada al reconocimiento y conteo taxonómico automatizado de vehículos.

Asimismo, la incorporación de una arquitectura alimentada mediante energía solar fotovoltaica otorga relevancia al área de Ingeniería Eléctrica, al analizar el consumo energético y la autonomía operativa del sistema como criterio para valorar la viabilidad de emplear fuentes renovables en nodos IoT de monitoreo. En conjunto, los resultados de esta investigación ofrecen una referencia para futuros trabajos que combinen visión artificial, IoT y sistemas fotovoltaicos en aplicaciones de monitoreo vehicular automatizado.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un sistema automático de conteo vehicular, basado en técnicas de visión artificial y hardware de bajo costo (DiY), que permita la obtención de datos cuantificables sobre el flujo vehicular en tres zonas específicas del campus matriz de la Universidad Católica de Cuenca, considerando sus diferentes edificios distribuidos en la ciudad.

1.4.2 Objetivos Específicos

- Reforzar los conocimientos teóricos y técnicos relacionados con visión artificial, detección de objetos y conteo vehicular, mediante una revisión sistemática de literatura científica especializada, sustentando el diseño del sistema propuesto en información actual, relevante y de calidad.

- Diseñar un prototipo funcional de sistema automático de conteo vehicular, utilizando software de diseño 3D, hardware de bajo costo (DiY) e integración de módulos de captura y transmisión de datos, con el propósito de implementarlo en zonas estratégicas del campus matriz de la Universidad Católica de Cuenca.

- Implementar el prototipo en puntos estratégicos del entorno urbano cercano a los edificios del campus matriz de la Universidad Católica de Cuenca, mediante la instalación de módulos de captura con ESP32-CAM y sistemas autónomos de alimentación, evaluando su funcionamiento en condiciones reales de tránsito vehicular.

- Validar la precisión del sistema de clasificación y conteo vehicular automatizado mediante un análisis comparativo frente a un muestreo de conteo manual basado en análisis de grabaciones de video, determinando el índice de fiabilidad y el error relativo de la instrumentación propuesta.

CAPÍTULO II

2. Fundamentación Teórica

2.1 Conceptos de tránsito y flujo vehicular

El tránsito vehicular comprende el desplazamiento de vehículos a través de una infraestructura vial, cuyo comportamiento puede analizarse mediante diferentes variables y niveles de representación. Entre las principales variables utilizadas para caracterizar el flujo vehicular se encuentran la velocidad, la densidad y el flujo, las cuales permiten describir las condiciones generales de circulación en una vía (Rosini, 2013).

2.1.1 Parámetros para la medición de flujo vehicular

La medición del tráfico es un aspecto fundamental en la ingeniería de tránsito, permitiendo evaluar el flujo de distintos componentes de la movilidad, como vehículos, ciclistas, pasajeros y peatones (Cabrera & Cárdenas, 2023). Entre los principales parámetros utilizados para caracterizar el flujo vehicular se encuentran la intensidad, la velocidad y la densidad, los cuales permiten describir el comportamiento del tránsito mediante medidas macroscópicas que representan las condiciones generales de circulación en una vía. Estas variables proporcionan información sobre la cantidad de vehículos que circulan, la velocidad a la que se desplazan y el nivel de ocupación de la infraestructura vial, constituyendo elementos fundamentales para el análisis del comportamiento vehicular y la comprensión de la dinámica del tránsito (José & Romero, 2024).

La recopilación de estos parámetros resulta de gran importancia debido a que constituye la base para la generación de información vehicular confiable y actualizada. Los datos obtenidos permiten identificar patrones de circulación, variaciones en la

demanda vehicular y condiciones de congestión, proporcionando evidencia objetiva para el monitoreo y análisis del comportamiento del tránsito.

2.1.2 Sistemas de conteo vehicular: métodos manuales y automáticos

El conteo vehicular es una técnica esencial para cuantificar los volúmenes de tránsito, evaluar los niveles de servicio y proyectar el comportamiento del tráfico a lo largo del tiempo. Esta herramienta permite analizar flujos vehiculares considerando tanto el tránsito base como el generado por el crecimiento demográfico y socioeconómico, facilitando de esta manera la toma de decisiones para optimizar la infraestructura vial (Ramón & Suquinagua, 2023). En este contexto, el conteo de vehículos constituye una actividad fundamental dentro de los estudios de aforo de tránsito, siendo utilizado tanto para el diseño de nuevas infraestructuras viales como para el control y optimización del tráfico urbano. Para su implementación existen diversas técnicas de adquisición de datos, entre las que destacan el conteo manual, los sistemas basados en sensores neumáticos, sensores magnéticos y los sistemas de visión artificial, cada uno con características, ventajas y limitaciones específicas según el entorno de aplicación y los requerimientos de precisión del estudio (Amarilla et al., 2016). En la **Tabla 1** se presenta una comparación entre los diferentes métodos utilizados en el conteo vehicular mostrando sus características, ventajas y limitaciones.

Tabla 1

Comparación de métodos utilizados para conteo vehicular

Método	Características	Ventajas	Desventajas
Manual	Realizado por observadores que registran el paso de los vehículos en campo.	Bajo costo inicial, fácil implementación y verificación visual directa.	Requiere personal permanente, presenta fatiga del observador y posibles errores humanos.
Sensores neumáticos	Utilizan tubos instalados sobre la calzada para	Económicos, portátiles y	Desgaste de los sensores y limitada capacidad de

	detectar el paso de los ejes de los vehículos.	adecuados para estudios temporales.	clasificación vehicular.
Sensores magnéticos	Detectan alteraciones del campo magnético producidas por los vehículos.	Registro continuo y automatizado del flujo vehicular.	Requieren instalación en la infraestructura vial y mantenimiento periódico.
Fibra óptica	Emplean sensores ópticos para registrar el tránsito vehicular.	Permiten obtener volumen, clasificación y dirección del tráfico.	Costos elevados de instalación y mantenimiento.
Visión artificial	Utiliza cámaras y algoritmos de procesamiento de imágenes para detectar, clasificar y contar vehículos.	Permite conteo automático, clasificación vehicular, análisis de velocidad y monitoreo en tiempo real sin alteración de la infraestructura vial.	Sensible a condiciones climáticas e iluminación; requiere capacidad de procesamiento computacional.

Nota: Elaboración propia.

2.1.3 Sistemas Inteligentes de Transporte (ITS)

Los Sistemas Inteligentes de Transporte (ITS) representan la integración de tecnologías de información, comunicación, censado y procesamiento de datos orientadas a mejorar la eficiencia, seguridad y sostenibilidad de los sistemas de movilidad (ver **Figura 1**). Según (Shubneet & Yadav, 2026), los ITS emplean tecnologías avanzadas de detección, análisis de datos y comunicación para optimizar la gestión del transporte y responder a las crecientes demandas de movilidad urbana. En este sentido, dichos sistemas permiten recopilar y procesar información proveniente de diversos elementos de la infraestructura vial, facilitando el monitoreo del tránsito y la toma de decisiones en tiempo real. De manera similar, (Adeniran et al., 2026) destacan que los ITS constituyen una infraestructura fundamental para alcanzar sistemas de transporte más seguros, eficientes y sostenibles, especialmente en entornos urbanos caracterizados por un crecimiento constante de la demanda vehicular.

Figura 1

Sistemas ITS



Nota: (Masita et al., 2020), (Liu & Rodelas, 2024).

Por otra parte, el avance de la inteligencia artificial (IA), el aprendizaje automático (Machine Learning) y el aprendizaje profundo (Deep Learning) ha impulsado una nueva generación de ITS. En la revisión realizada por (Saki & Soori, 2026), se destaca que estas tecnologías están transformando la gestión del tráfico al proporcionar herramientas capaces de analizar grandes volúmenes de datos, identificar patrones complejos y apoyar procesos de toma de decisiones.

2.2 Visión artificial

2.2.1 Fundamentos de visión artificial

La visión artificial es una rama de la inteligencia artificial que permite a los sistemas computacionales adquirir, procesar e interpretar información visual proveniente de imágenes y videos. Su objetivo es dotar a las máquinas de capacidades de percepción similares a las humanas, facilitando la identificación de objetos, patrones y eventos dentro de una escena (Cinnamon & Dai, 2026; De Magistris & Del Bimbo, 2023).

Una de las principales aplicaciones de esta tecnología se encuentra en los ITS, donde se emplea para el monitoreo del tránsito, el conteo de vehículos y la estimación del flujo vehicular. Estas funciones contribuyen a una gestión más eficiente de la movilidad;

sin embargo, aún persiste el desafío de equilibrar la precisión de detección con la velocidad de procesamiento en tiempo real (Yang et al., 2021).

2.2.2 Reconocimiento y detección de imágenes

Los avances en visión por computador impulsados por el aprendizaje profundo han favorecido el desarrollo de sistemas cada vez más eficientes para tareas como la clasificación de imágenes, la detección de objetos, la segmentación y el seguimiento de objetos. Gracias a la disponibilidad de grandes conjuntos de datos y al incremento de la capacidad de procesamiento, estos modelos han alcanzado un rendimiento cercano al tiempo real, incluso en dispositivos con recursos limitados. En este contexto, la detección de objetos desempeña un papel fundamental, ya que constituye la base para aplicaciones más complejas de análisis visual, tales como la comprensión de escenas, el seguimiento de objetos, la detección de eventos y el reconocimiento de actividades (K. Li, 2022).

2.2.2.1 Detección de Objetos

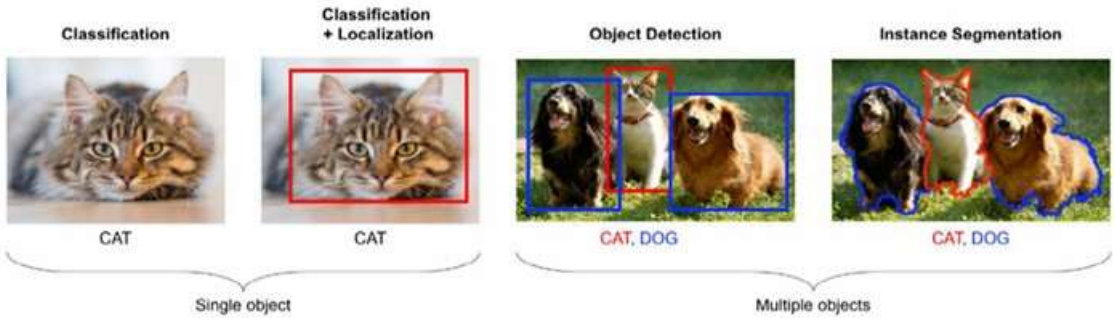
La detección de objetos es una técnica fundamental de la visión por computadora que permite identificar y localizar automáticamente los objetos presentes en imágenes o videos, permitiendo determinar tanto la categoría del objeto como su ubicación dentro de la escena (Kaur & Singh, 2022). De acuerdo con (Pathak et al., 2018), este proceso debe afrontar desafíos como variaciones de iluminación, cambios de escala y oclusiones que pueden afectar el desempeño de los modelos de detección. Asimismo, la detección de objetos integra las tareas de clasificación y localización, permitiendo determinar tanto la categoría del objeto como su ubicación dentro de la escena (Kaur & Singh, 2022).

Por otra parte, los métodos modernos de detección de objetos se agrupan principalmente en arquitecturas de dos etapas y de una sola etapa. Mientras que modelos como R-CNN y Faster R-CNN destacan por su precisión al generar regiones de interés antes de la clasificación, algoritmos como SSD y YOLO realizan la detección en una

única etapa, logrando una mayor velocidad de procesamiento y un mejor desempeño en aplicaciones en tiempo real (Lin et al., 2026). En la **Figura 2** se muestra el proceso general de detección de objetos, donde una imagen es analizada para reconocer los elementos presentes y determinar su posición mediante cuadros delimitadores.

Figura 2

Detección de objetos mediante visión por computadora



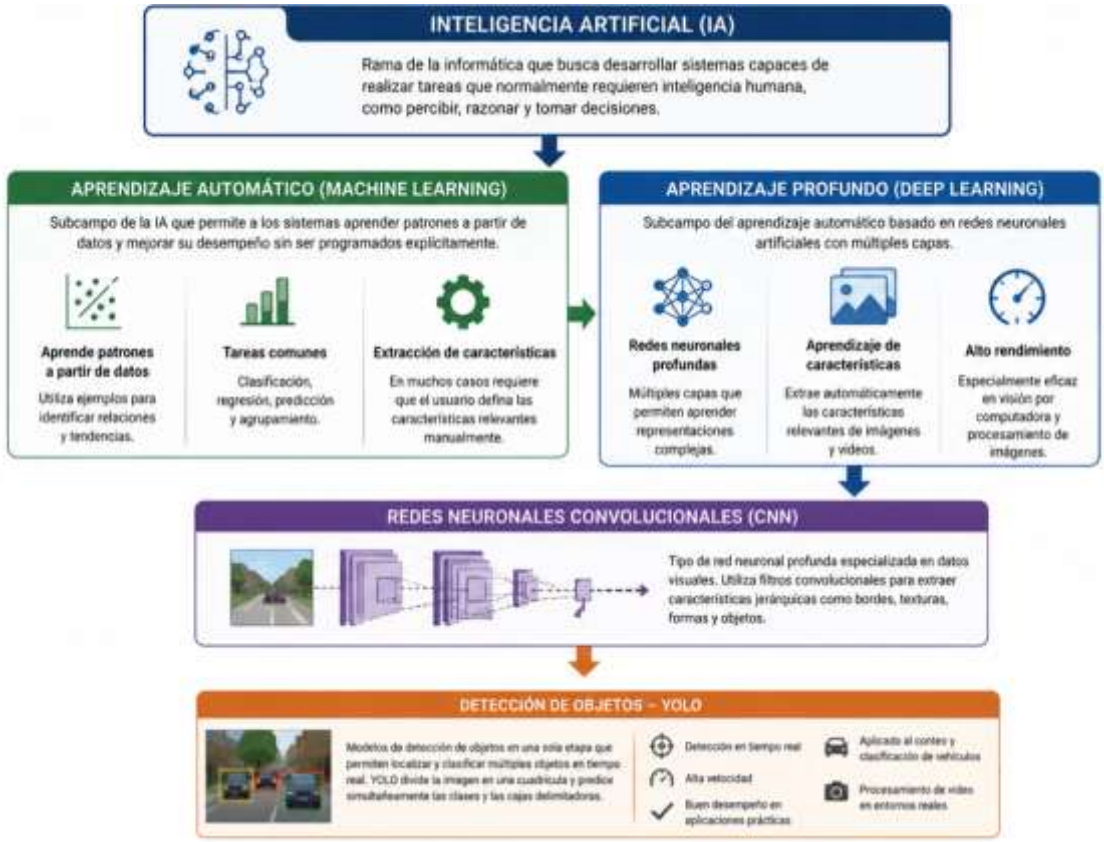
Nota: Imagen adaptada de (Pathak et al., 2018)

2.2.3 Aprendizaje automático y aprendizaje profundo

La inteligencia artificial, el aprendizaje automático y el aprendizaje profundo constituyen la base de las técnicas empleadas en visión por computadora, en particular, las redes neuronales convolucionales permiten procesar imágenes y videos, sobre las cuales se desarrollan modelos de detección de objetos como YOLO. La relación entre estos conceptos y su evolución hacia los modelos de detección de objetos se presenta en la **Figura 3**.

Figura 3

Relación entre inteligencia artificial, aprendizaje automático, aprendizaje profundo, redes neuronales convolucionales y YOLO.



Nota: (Janiesch et al., 2021), (Dimililer et al., 2020), (Masita et al., 2020), (Liu & Rodelas, 2024), (Rezgui et al., 2019), (Farid et al., 2025).

2.2.4 Tecnologías y bibliotecas para procesamiento de imágenes

Los sistemas modernos de detección vehicular basados en visión artificial integran diversas herramientas de software para el procesamiento de imágenes, el desarrollo de modelos de aprendizaje profundo y la ejecución eficiente de algoritmos de detección de objetos. Entre las tecnologías más utilizadas se encuentran OpenCV para el

procesamiento de imágenes y video, TensorFlow y PyTorch para la implementación de redes neuronales profundas, y NumPy para el manejo de operaciones matemáticas y matriciales En la **Tabla 2** se presentan las principales bibliotecas y herramientas utilizadas en este tipo de sistemas y aplicación dentro del proceso de detección vehicular. Estas herramientas permiten desarrollar modelos de detección y clasificación de vehículos basados en CNN y YOLO, ampliamente utilizados en aplicaciones de monitoreo de tráfico y Sistemas Inteligentes de Transporte (ITS) (Alvarado et al., 2025; Majumder & Wilmot, 2023).

Tabla 2

Principales bibliotecas y herramientas para el procesamiento de imágenes y detección de objetos.

Biblioteca / herramienta	Aplicación en el sistema	Ventaja principal
Python	Desarrollo e integración del sistema de detección y conteo	Amplio ecosistema
OpenCV	Captura y procesamiento de video e imágenes	Código abierto y alto rendimiento
NumPy	Operaciones numéricas y manipulación de datos	Procesamiento eficiente
TensorFlow	Entrenamiento e inferencia de modelos	Optimización para GPU
PyTorch	Desarrollo y ejecución de modelos de Deep Learning	Flexibilidad
CUDA	Aceleración del procesamiento mediante GPU	Reduce tiempos de procesamiento
cuDNN	Optimización de operaciones de redes neuronales	Mayor velocidad

Nota: Elaboración Propia.

2.2.5 Modelos para detección vehicular

Los modelos de detección de objetos constituyen una herramienta para identificar y localizar vehículos a partir de imágenes o secuencias de video, permitiendo automatizar tareas relacionadas con el monitoreo del tránsito. Su aplicación en escenarios vehiculares

comprende diferentes arquitecturas y configuraciones, cuyo desempeño puede variar de acuerdo con el tipo de vehículo, los datos utilizados para el entrenamiento y las condiciones de detección (X. Li et al., 2025).

2.2.5.1 YOLO aplicada al monitoreo vehicular

En aplicaciones de transporte inteligente, YOLO ha sido utilizado para la detección, seguimiento y conteo de vehículos a partir de secuencias de video, permitiendo estimar la densidad vehicular y analizar las condiciones de circulación en tiempo real. La integración de este modelo con técnicas de seguimiento facilita la identificación continua de los vehículos entre fotogramas, proporcionando información útil para la gestión del tráfico y la optimización de los tiempos (Jayapradha et al., 2024).

Las diferentes versiones de YOLO han incorporado mejoras orientadas a optimizar su desempeño en la detección de objetos. En la **Tabla 3** se presentan las principales versiones empleadas en la detección vehicular, junto con sus mejoras y dificultades.

Tabla 3

Principales versiones de YOLO aplicadas a la detección vehicular.

Versión	Principales mejoras para la detección vehicular	Incidencia en el conteo de vehículos	Principales dificultades o limitaciones
YOLOv3	Detección a distintas escalas.	Mejora la detección de vehículos de diferentes tamaños.	Dificultades con vehículos pequeños y ocluidos.
YOLOv4	Mayor precisión y eficiencia.	Favorece la detección en video.	Sensible a oclusiones y escenas complejas.
YOLOv5	Optimización de velocidad y precisión.	Adecuado para conteo en tiempo real.	Compromiso entre precisión y velocidad según el modelo.
YOLOv8	Mejoras en arquitectura y detección.	Mayor precisión para detección y seguimiento.	Puede presentar dificultades con objetos pequeños y oclusiones.

YOLOv11	Mayor eficiencia y precisión.	Favorece el procesamiento en tiempo real.	Dependencia de los datos y condiciones de la escena.

Nota: Elaboración Propia.

Dentro de la evolución de la familia YOLO, YOLOv8 constituye una evolución orientada a la detección de objetos, incorporando mejoras en la extracción y fusión de características mediante su arquitectura compuesta por backbone, neck y head (Zhang et al., 2026). Dentro de sus diferentes variantes se encuentra YOLOv8m, empleada para la detección y clasificación de vehículos mediante imágenes CCTV, así como para el análisis del volumen y flujo vehicular incorpora mejoras en su arquitectura orientadas a la detección de objetos y a la extracción de características a diferentes escalas.

En aplicaciones vehiculares, (Neamah & Karim, 2023) reportaron que con YOLOv8 tienen una precisión media de detección del 96,58 % y una precisión media de conteo del 97,54 %. Como referencia adicional, la **Tabla 4** recopila resultados reportados en diferentes estudios sobre el desempeño de variantes de YOLOv8 en la detección y clasificación vehicular.

Tabla 4

Comparación del desempeño de diferentes modelos YOLO.

Modelo	**Precisión**	**Recall**	**mAP@0.50**
YOLOv5m	86,34 %	85,28 %	87,48 %
YOLOv8m	85,99 %	77,51 %	86,20 %
YOLOv12m	90,27 %	77,92 %	87,50 %

Nota: Elaboración Propia.

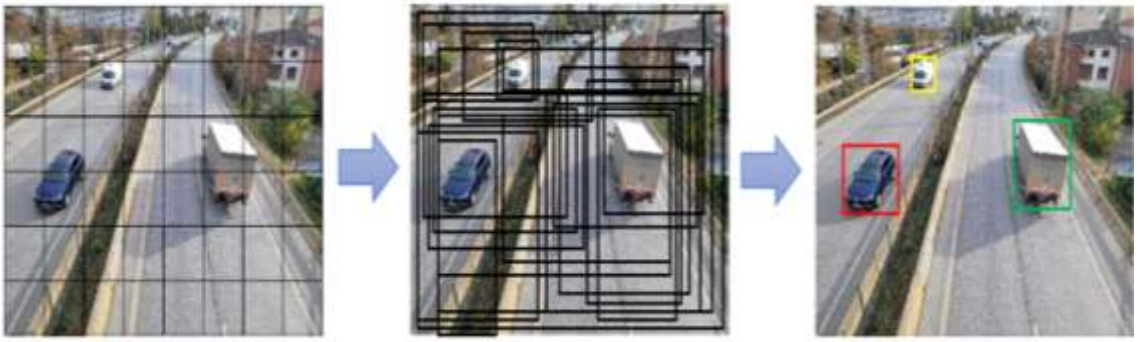
2.2.6 Seguimiento y conteo multiobjeto (tracking)

Después de detectar los vehículos en cada fotograma, es necesario asociar las detecciones correspondientes a un mismo objeto en cuadros consecutivos para generar trayectorias de movimiento. Este proceso permite mantener la identidad de cada vehículo

durante su desplazamiento y evitar registros repetidos durante el conteo (Huang et al., 2025). Los algoritmos de seguimiento pueden asignar un identificador a cada detección para mantener la correspondencia del vehículo entre fotogramas consecutivos (D. Xu et al., 2022). El proceso de seguimiento permite asociar las detecciones de un mismo objeto entre fotogramas consecutivos, manteniendo su correspondencia durante el desplazamiento (ver **Figura 4**). En sistemas que emplean diferentes cámaras, estas técnicas pueden complementarse con métodos de reidentificación de vehículos para determinar la correspondencia de un mismo vehículo entre diferentes puntos de observación (Alahdal et al., 2024). Para establecer esta asociación se emplean técnicas utilizadas en aplicaciones de monitoreo vehicular como KCF, DeepSORT y ByteTrack, las cuales permiten mantener la continuidad de las trayectorias incluso ante oclusiones parciales, cambios de escala o variaciones de iluminación, contribuyendo a mejorar la precisión del conteo y del análisis del flujo vehicular (Pudaruth et al., 2024).

Figura 4

Proceso de seguimiento de vehículos entre fotogramas mediante detección de objetos.



Nota: Imagen adaptada de (Estevez et al., 2025)

2.3 Internet de las Cosas (IoT)

El IoT se define como una red de dispositivos, sensores y sistemas físicos interconectados que se comunican a través de Internet para recopilar, procesar y compartir información, permitiendo la interacción entre el mundo físico y digital (Hashemian &

Derakhshanfard, 2025). En el ámbito del transporte, los dispositivos IoT ubicados en carreteras o vehículos permiten recopilar datos de tráfico en tiempo real para apoyar el desarrollo de ITS (Mrudula et al., 2024).

2.3.1 Integración de IoT y Aprendizaje Profundo para el Monitoreo Vehicular

En los sistemas inteligentes de transporte, la combinación del IoT y el aprendizaje profundo ha fortalecido las capacidades de monitoreo al integrar la recopilación de información en tiempo real mediante sensores y cámaras con técnicas avanzadas de análisis automático. En este contexto, los datos capturados por dispositivos conectados pueden ser procesados mediante algoritmos de aprendizaje profundo para apoyar las tareas de monitoreo, detección y supervisión del tráfico (Tang et al., 2024).

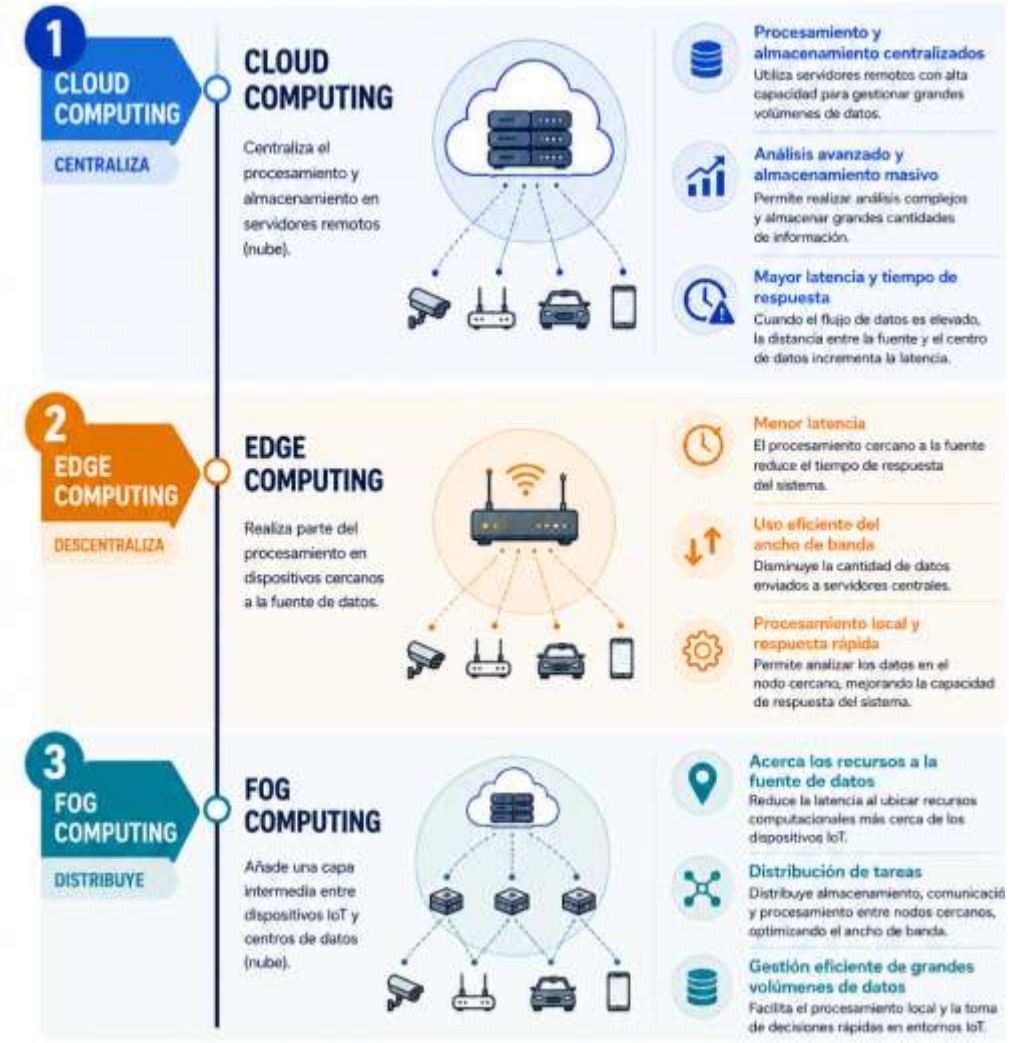
Asimismo, la incorporación de modelos de detección de objetos en infraestructuras basadas en IoT permite analizar secuencias de video en tiempo real, favoreciendo una respuesta más rápida y precisa ante eventos de interés dentro del entorno vial (Suhana et al., 2025)

2.3.2 Arquitecturas IoT

El crecimiento de los dispositivos conectados y la generación masiva de datos han impulsado el desarrollo de diferentes arquitecturas de procesamiento dentro del Internet de las Cosas (IoT). En este contexto, las arquitecturas Cloud Computing, Edge Computing y Fog Computing representadas en la **Figura 5**, han surgido como alternativas para mejorar el tratamiento de la información, reducir la latencia y optimizar el uso de los recursos computacionales y de comunicación. Estas arquitecturas distribuyen las tareas de procesamiento en distintos niveles de la red, permitiendo responder a los requerimientos de aplicaciones que demandan análisis eficiente y disponibilidad de datos en tiempo real (X. Xu et al., 2024).

Figura 5

Comparación de las arquitecturas Cloud, Edge y Fog Computing en sistemas IoT.



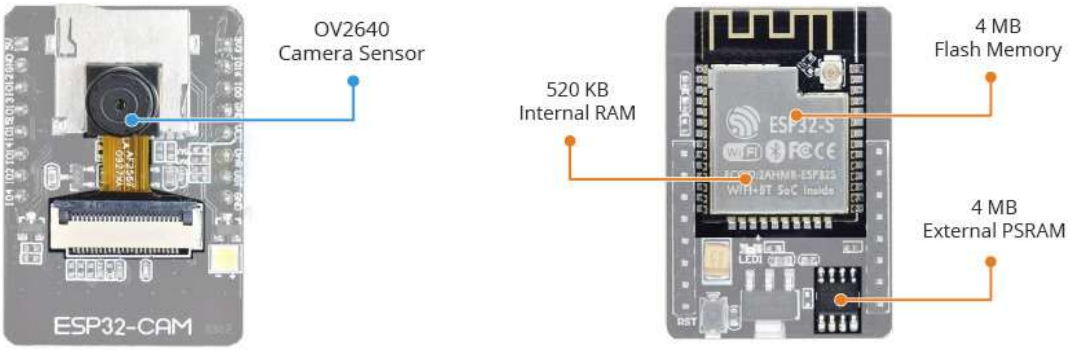
Nota: Elaboración Propia.

2.3.3 ESP32-CAM

El ESP32 representado en la **Figura 6**, es una plataforma ampliamente utilizada en aplicaciones de IoT debido a su capacidad de procesamiento y a la integración de conectividad Wi-Fi y Bluetooth, características que permiten la adquisición, procesamiento y transmisión de datos en tiempo real. para aplicaciones de visión artificial el ESP32-CAM representa una solución de bajo costo y alto rendimiento para el desarrollo de sistemas de detección de objetos en tiempo real (Cangrejo & Ovalle, 2024).

Figura 6

ESP32CAM: (a) Vista frontal; (b) Vista posterior



(a) (b)

Nota: Elaboración Propia.

2.3.4 Redes de comunicación para sistemas IoT

En los sistemas IoT se emplean diferentes tecnologías y protocolos de comunicación para permitir el intercambio de información entre dispositivos y plataformas de procesamiento. Entre las alternativas más utilizadas se encuentran Wi-Fi, MQTT y Ethernet, cuyas principales características se resumen en la **Figura 7**.

Figura 7

Características de las tecnologías de comunicación utilizadas en sistemas IoT.



Nota: Elaboración Propia.

2.4 Eficiencia energética en sistemas IoT y Edge AI

La eficiencia energética constituye un aspecto fundamental en los sistemas IoT debido a que una parte importante del consumo energético está asociada a la transmisión de datos hacia infraestructuras externas. En este sentido, el procesamiento en el borde permite ejecutar tareas localmente y reducir la energía requerida para las comunicaciones (Rojek et al., 2026). A nivel computacional, la implementación de modelos ligeros favorece la ejecución de algoritmos de inteligencia artificial en dispositivos con recursos

limitados, disminuyendo la carga de procesamiento y el consumo energético (Annamalai et al., 2024). De manera complementaria, las técnicas de compresión de modelos permiten reducir los requerimientos de memoria y cálculo, contribuyendo a mejorar la eficiencia energética de las aplicaciones Edge AI (Moudoud et al., 2023).

2.4.1 Consumo energético en sistemas de embebidos de visión artificial

Los sistemas de visión artificial embebidos demandan energía para la adquisición, procesamiento y transmisión de imágenes, por lo que su desempeño energético depende tanto del hardware como de la carga computacional ejecutada (Ning et al., 2024). En este contexto, los microcontroladores representan una alternativa adecuada para aplicaciones IoT debido a su bajo consumo energético y costo reducido (Porreca et al., 2025).

2.4.2 Estrategias de eficiencia energética en sistemas embebidos basados en ESP32-CAM

La eficiencia energética en sistemas embebidos basados en ESP32-CAM puede mejorarse mediante estrategias orientadas a reducir el consumo durante los periodos de actividad e inactividad del dispositivo. En la **Figura 8**, se resumen las tres principales estrategias y sus beneficios.

Figura 8

Estrategias de eficiencia energética en ESP32-CAM.



Nota: Elaboración Propia.

2.4.3 Eficiencia y sostenibilidad energética en sistemas IoT

La eficiencia y sostenibilidad energética constituyen aspectos relevantes en los sistemas IoT embebidos, donde se busca optimizar el consumo y aprovechar responsablemente los recursos disponibles. En la **Figura 9**, se resumen los principales conceptos y estrategias relacionados con la eficiencia energética y el enfoque Green IoT.

Figura 9

Eficiencia y sostenibilidad energética en sistemas IoT embebidos.



Nota: Elaboración Propia.

2.5 Sistemas fotovoltaicos autónomos

Los sistemas fotovoltaicos autónomos constituyen una alternativa para el suministro de energía eléctrica en aplicaciones que requieren operar de manera independiente de la red convencional, especialmente en zonas rurales para lo cual integran módulos fotovoltaicos, unidades de almacenamiento y elementos de conversión que permiten gestionar la energía disponible y mantener un suministro estable y confiable bajo diferentes condiciones de operación (Marin & Gaiceanu, 2025).

2.5.1 Radiación solar

La radiación solar corresponde a la energía emitida por el Sol que llega a la superficie terrestre y constituye el recurso energético aprovechado por los sistemas fotovoltaicos. La irradiación solar representa la cantidad de energía recibida sobre una superficie durante un periodo determinado y se expresa normalmente en kWh/m² mientras que la irradiancia corresponde a la potencia instantánea de la radiación solar sobre una superficie y se expresa en W/m² (Estefanía & Crespo, 2026).

2.5.2 Sistemas fotovoltaicos fuera de red

Los sistemas fotovoltaicos fuera de red (off-grid) permiten suministrar energía eléctrica de manera independiente de la red convencional y son adecuados para lugares donde el acceso al suministro eléctrico es limitado o inexistente. Su funcionamiento se basa en la generación de electricidad mediante módulos fotovoltaicos y el almacenamiento de energía para mantener el suministro cuando no existe generación solar (Tuba & Jimenez, 2025).

2.5.3 Componentes de un sistema fotovoltaico autónomo

Los sistemas fotovoltaicos autónomos requieren la integración de diferentes elementos para aprovechar la energía solar y disponer de ella de acuerdo con las necesidades de consumo. Entre sus componentes principales se encuentran el panel fotovoltaico que transforma la radiación solar en energía eléctrica, las baterías que almacenan la energía generada durante los periodos de disponibilidad solar y el regulador de carga que controla el proceso de carga y protege el almacenamiento frente a sobrecargas y descargas excesivas (Pupiales, 2024).

2.5.3.1 Panel solar monocristalino

Los paneles solares monocristalinos se fabrican a partir de silicio de alta pureza y se caracterizan por su buen rendimiento y larga vida útil además presentan un comportamiento favorable ante condiciones de baja iluminación, aunque su eficiencia puede verse afectada por el aumento de temperatura. Como principal desventaja presentan un mayor costo de fabricación debido al proceso de obtención del silicio (Buele & Chazi, 2019).

2.5.3.2 Baterías

Las baterías permiten almacenar la energía eléctrica generada durante las horas de radiación solar para utilizarla posteriormente cuando la generación fotovoltaica

disminuye o no está disponible, por lo que su capacidad de almacenamiento, eficiencia, autodescarga, vida útil y requerimientos de mantenimiento son características importantes para garantizar un funcionamiento adecuado del sistema autónomo (Pupiales, 2024).

2.5.3.3 Controlador de Carga

El controlador de carga regula la energía proveniente de los paneles hacia las baterías y evita condiciones que puedan afectar su funcionamiento como la sobrecarga o la descarga excesiva, entre sus principales tecnologías se encuentran los controladores PWM y MPPT que permiten gestionar de diferente manera la energía generada (Buele & Chazi, 2019).

CAPÍTULO III

3. Metodología

3.1 Enfoque de la investigación

La investigación presentada, conto con un enfoque aplicativo y descriptivo, orientado al diseño y desarrollo de un sistema automatizado para el monitoreo y conteo vehicular mediante técnicas de visión artificial e IoT. La propuesta integro un nodo de adquisición de video con alimentación fotovoltaica autónoma y un sistema de procesamiento de imágenes para la obtención automática de datos del flujo vehicular.

3.2 Desarrollo de la propuesta

El desarrollo de la propuesta se llevó a cabo mediante una metodología orientada al diseño, implementación y evaluación de un sistema automatizado de conteo vehicular mediante visión artificial. En primer lugar, se realizó una revisión de la literatura científica relacionada con el monitoreo vehicular, la visión artificial, los modelos de detección de objetos, las arquitecturas IoT y los sistemas de alimentación autónoma, mediante la consulta de bases de datos digitales, con el propósito de establecer los fundamentos teóricos y técnicos necesarios para el desarrollo del sistema. La secuencia general de estas actividades, se presenta en la **Figura 10**.

Con base en la información recopilada, se definió la arquitectura general del proyecto, considerando la plataforma destinada a la captura de video y transmisión de datos, así como los requerimientos energéticos asociados a su funcionamiento. Con base en estos parámetros, se efectuaron los cálculos correspondientes para determinar la demanda energética y realizar el dimensionamiento del sistema fotovoltaico y del banco de baterías.

Una vez definida la infraestructura de adquisición de información y el sistema de alimentación, se desarrollaron los algoritmos necesarios para el procesamiento del video, utilizando las librerías y modelos de visión artificial descritos en la sección anterior. Se pusieron a prueba diferentes modelos preentrenados en la detección de vehículos y, de esta manera, se seleccionó el modelo más preciso en la detección y clasificación de vehículos.

Paralelamente, se realizó el diseño tridimensional del prototipo utilizando software de diseño 3D Autodesk Fusión 360. Esta etapa comprendió el diseño y modelado 3D de la caja, así como las estructuras necesarias para el ensamblaje, protección e integración de los componentes encargados de la captura de video, comunicación y alimentación del sistema. Asimismo, permitió definir la distribución física de los diferentes subsistemas, verificar las condiciones de montaje y facilitar el ensamblaje del prototipo final antes de su implementación en campo.

A continuación, se integraron los distintos subsistemas desarrollados en un prototipo funcional que permitió su instalación y operación en campo. Durante esta fase se realizaron pruebas preliminares en entornos controlados, para verificar su funcionamiento y efectuar los ajustes requeridos antes de su despliegue.

Una vez verificada la operatividad del sistema, se realizó la instalación de los dispositivos en cada uno de los puntos de monitoreo seleccionados dentro del campus universitario, correspondientes a la Basílica, la Facultad de Ingeniería, Industria y Construcción, así como la Dirección de Posgrado, donde se desarrollaron jornadas de funcionamiento comprendidas entre las 06h00 y las 18h00 para la adquisición de información vehicular. Durante este periodo, se efectuaron pruebas de transmisión de datos mediante la red Ethernet y se verificó la recepción y almacenamiento de los videos en los servidores del Laboratorio RobLab para su procesamiento mediante los algoritmos

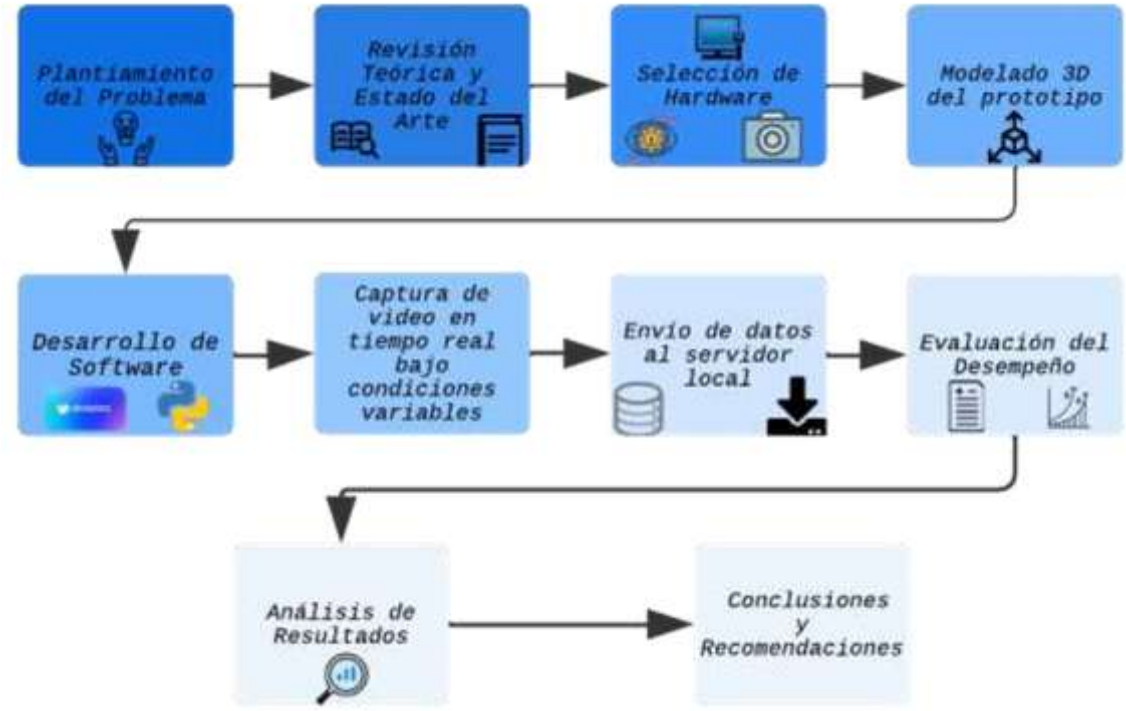
de detección y conteo vehicular previamente desarrollados. Asimismo, se evaluó el comportamiento del sistema de alimentación fotovoltaica durante las jornadas de monitoreo, verificando su capacidad para mantener la operación de los dispositivos.

Para la evaluación del desempeño del sistema de conteo vehicular se empleó un muestreo mixto compuesto por estratificación criterial y selección aleatoria intra-estrato, (selección estratificada de los días de observación y la selección aleatoria de fragmentos de video). Durante las cuatro semanas de monitoreo se realizaron registros en los tres puntos establecidos, en el horario de 06h00 a 18h00. De estos registros se seleccionaron los días lunes, jueves y domingo de cada semana y fragmentos aleatorios de 15 minutos para realizar el conteo manual y compararlo con el conteo automáti co mediante YOLO. En total, se analizaron 36 fragmentos de video, correspondientes a cuatro semanas, tres días por semana y tres puntos de monitoreo.

Como etapa final, se realizó un análisis de resultados de fragmentos representativos de las grabaciones obtenidas para realizar conteos manuales de referencia. Los resultados fueron comparados con los registros generados automáticamente por el sistema, permitiendo evaluar su desempeño mediante indicadores asociados a los aciertos y errores de detección y conteo vehicular.

Figura 10

Diagrama de flujo de la metodología empleada.



Nota: Elaboración Propia.

CAPÍTULO IV

4. Desarrollo

En este capítulo se presenta el diseño y modelado del sistema de monitoreo vehicular considerando los componentes electrónicos, de comunicación, energéticos y mecánicos necesarios para su implementación. Se describe la selección de los elementos para la captura y transmisión de video y se desarrolla el dimensionamiento energético de los puntos que emplean sistemas fotovoltaicos. Además, se presenta el diseño de la placa de interconexión entre el ESP32-CAM y el W5500 junto con el diseño del compartimiento y los elementos de montaje desarrollados mediante Autodesk Fusion 360.

Finalmente, se estableció la arquitectura lógica del sistema desde la captura y transmisión del video hasta la detección, seguimiento y conteo de los vehículos.

4.1 Componentes electrónicos y de comunicación del sistema

El sistema de monitoreo vehicular está conformado por componentes electrónicos y de comunicación que trabajan de manera integrada para realizar la captura y transmisión del video, así como su posterior procesamiento para el conteo vehicular. La alimentación de los dispositivos se adapta a las condiciones de cada punto de monitoreo, mediante sistemas fotovoltaicos en la Facultad de Ingeniería, Industria y Construcción, y la Basílica y mediante conexión a la red eléctrica en la Dirección de Posgrado. En la **Tabla 5**, se presentan los principales componentes empleados en el desarrollo del prototipo y la función que desempeñan dentro del sistema.

Tabla 5

Especificaciones de componentes eléctricos

Componente	Parámetros	Descripción
ESP32-CAM 	Alimentación: 5 V DC CPU: Xtensa LX6, 240 MHz SRAM: 520 KB PSRAM: 4 MB Cámara: OV2640, 2 MP Wi-Fi / Bluetooth 4.2	Módulo compacto que incorpora cámara OV2640, memoria PSRAM, ranura microSD y conectividad inalámbrica.
W5500 	Alimentación: 5 V DC Ethernet: 10/100 Mbps Interfaz: SPI, hasta 80 MHz Sockets: 8 Protocolos: TCP/IP	Controlador Ethernet con pila TCP/IP integrada y PHY 10/100Base-T. Permite establecer comunicaciones de red mediante una interfaz SPI.
Controlador de carga solar 	Tensión: 12/24 V DC Corriente: 30 A Tipo: PWM Pantalla: LCD USB: 2 × 5 V Detección automática: 12/24 V	Regulador solar para sistemas fotovoltaicos de 12 y 24 V. Incorpora control PWM, pantalla LCD, puertos USB y funciones de protección para la gestión de la carga.
Panel solar 	Modelo: SZYL-M40-18C Tipo: monocristalino Potencia: 40 Wp Eficiencia: 17,5 % Vmp: 18,24 V Imp: 2,19 A	Panel fotovoltaico monocristalino compuesto por 32 celdas de silicio, diseñado para convertir la radiación solar en energía eléctrica. Cuenta con caja de

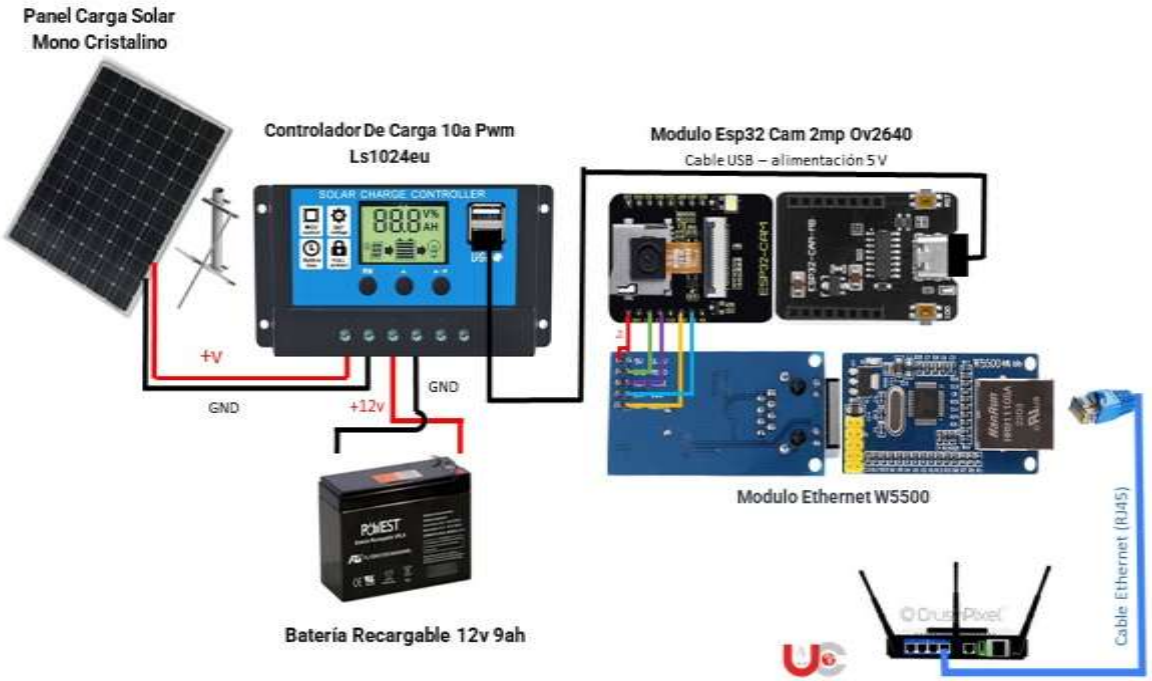
Batería		conexiones con protección IP67.
	Modelo: FL1290GS Tipo: VRLA-AGM Voltaje: 12 V Capacidad: 9 Ah Terminales: F2	Batería sellada de plomo-ácido regulada por válvula y tecnología AGM, diseñada para aplicaciones de respaldo y almacenamiento de energía sin requerir mantenimiento periódico.

Nota: Elaboración propia.

A partir de los componentes seleccionados, se estableció la configuración eléctrica y de comunicación del nodo, definiendo las conexiones entre los subsistemas de adquisición, comunicación y alimentación. La **Figura 11** presenta el esquema general de conexiones empleado para la integración de los diferentes elementos del sistema.

Figura 11

Diagrama de conexiones eléctricas y de comunicación del nodo.



Nota: Elaboración propia.

4.2 Dimensionamiento fotovoltaico

El sistema de monitoreo requirió una fuente de alimentación capaz de mantener la operación de sus componentes durante las jornadas de monitoreo establecidas. Para determinar los requerimientos del sistema fotovoltaico, se analizó el consumo eléctrico de los dispositivos y la demanda energética diaria, a partir de la cual se establecieron las capacidades necesarias del panel fotovoltaico y del banco de baterías. Finalmente, se verificó la autonomía disponible para las condiciones de operación previstas.

4.2.1 Estimación del consumo eléctrico

Para estimar el consumo eléctrico del nodo de monitoreo, se consideraron las especificaciones de alimentación y consumo proporcionadas por los fabricantes de los componentes encargados de la captura y transmisión de video. En esta etapa, se estableció una estimación teórica de la demanda energética del sistema, sin considerar mediciones experimentales realizadas sobre el prototipo.

El ESP32-CAM, utilizado para la adquisición y procesamiento inicial de las imágenes, presenta un consumo especificado de 300 mA a 5 V con el sistema de iluminación apagado, mientras que en modo Deep Sleep el fabricante establece un consumo mínimo de 20 mA a 5 V.

Por su parte, el controlador Ethernet W5500 opera con una tensión de alimentación de 5 V y presenta una corriente de suministro de 200 mA en condiciones normales de operación, de acuerdo con las especificaciones del fabricante.

A partir de los valores registrados, la potencia eléctrica consumida se determinó mediante:

$$P = V \times I \tag{1}$$

Donde (P) representa la potencia eléctrica en vatios (W), (V) la tensión de alimentación en voltios (V) e (I) la corriente de operación en amperios (A).

Para el ESP32-CAM durante el funcionamiento activo se obtuvo:

$$P_{\mathrm{ESP32CAM}} = 5\ V \times 0{,}3\ A = 1{,}5\ W$$

Mientras que para el módulo W5500:

$$P_{\mathrm{W5500}} = 5\ V \times 0{,}2\ A = 1{,}0\ W$$

Por tanto, la potencia nominal conjunta de los componentes durante el funcionamiento activo se estimó en:

$$P_{\mathrm{W5500+ESP32CAM}} = 1{,}5\ W + 1{,}0\ W = 2{,}5\ W$$

Para el periodo de bajo consumo del ESP32-CAM, se consideró el valor especificado por el fabricante para el modo Deep Sleep:

$$P_{\mathrm{Slepp}} = 5\ V \times 0{,}02\ A = 0{,}1\ W$$

Durante el periodo de bajo consumo, el ESP32-CAM permanece en modo Deep Sleep mientras el módulo W5500 continúa activo para mantener la comunicación del nodo. Considerando el consumo especificado por los fabricantes, la potencia estimada para esta condición se determinó mediante:

$$P_{\mathrm{W5500+Sleep}} = 0{,}1\ W + 1{,}0\ W\ = 1{,}1\ W$$

Con el propósito de sintetizar la demanda energética teórica del nodo en sus diferentes estados de funcionamiento, en la **Tabla 6** se detallan las condiciones operativas evaluadas, indicando la tensión nominal, el consumo de corriente resultante y la potencia eléctrica calculada para cada régimen.

Tabla 6

Estimación del consumo de corriente y potencia del nodo según la condición de operación

Condición de operación	Tensión (V)	Corriente (A)	Potencia (W)
ESP32-CAM activo	5,00	0,3	1,5
ESP32-CAM + W5500 activos	5,00	0,5	2,5
ESP32-CAM en Deep Sleep + W5500 activo	5,00	0,22	1,1

Nota: Elaboración propia

4.2.2 Demanda energética diaria

La demanda energética diaria del nodo se determinó considerando el horario de operación establecido para el sistema, comprendido entre las 06:00 y las 18:00 horas correspondientes a un periodo de actividad de 12 horas diarias. El ESP32-CAM permanece en funcionamiento normal para la captura y envío de los datos de video a través del módulo W5500. Mientras que en el periodo comprendido entre las 18:00 horas y las 06:00 horas del día siguiente, el ESP32-CAM entra en modo Deep Sleep disminuyendo su consumo energético, pero el módulo W5500 continúa activo manteniendo la conexión Ethernet activa durante todo el día. El modo Deep Sleep del módulo ESP32 ayuda a reducir el consumo energético del nodo durante las horas en las que no se realiza la adquisición de video.

La energía consumida durante cada periodo se determinó mediante:

$$E = P \times t \tag{2}$$

donde E representa la energía consumida en vatios-hora (Wh), P la potencia eléctrica en vatios (W) y t el tiempo de operación en horas (h).

Durante el periodo de operación normal, se consideró la potencia medida de 2,5 W sacada en la sección 4.2.1 para el conjunto formado por el ESP32-CAM y el módulo W5500. Al mantenerse esta condición durante 12 horas, la energía requerida es:

$$E_{normal} = 2{,}5W \times 12h$$

$$E_{normal} = 30\ Wh$$

Durante las 12 horas restantes, el ESP32-CAM permanece en modo Deep Sleep y el W5500 continúa activo. Considerando la potencia medida de 1,1 W para esta condición, se obtiene:

$$E_{Sleep} = 1{,}1\ W \times 12h$$

$$E_{Sleep} = 13{,}2\ Wh$$

Por tanto, la demanda energética diaria del nodo se obtiene sumando la energía requerida durante ambos periodos:

$$E_d = E_{normal} + E_{Sleep} \quad (3)$$

$$E_d = 30\ Wh + 13{,}2\ Wh$$

$$E_d = 43{,}2\ Wh/dia$$

Los resultados obtenidos se resumen en la **Tabla 7**.

Tabla 7

Demanda energética diaria del nodo

Estado de operación	**Potencia (W)**	**Tiempo de operación (h)**	**Energía (Wh)**
ESP32-CAM + W5500	2,5	12	30
Deep Sleep: ESP32-CAM + W5500	1,1	12	13,2
Demanda energética diaria		24	43,2 Wh/día

Nota: Elaboración propia.

4.2.3 Dimensionamiento del panel fotovoltaico

Para el dimensionamiento del sistema fotovoltaico se analizaron las condiciones solares correspondientes a los dos puntos donde se implementó alimentación fotovoltaica: Basílica y Facultad de Ingeniería. Para el dimensionamiento del panel se analizaron las condiciones solares de los tres puntos considerados para la alimentación fotovoltaica. Se utilizaron las plataformas PVGIS y Global Solar Atlas, configurando una potencia instalada de 0,04 kWp, equivalente al panel de 40 Wp seleccionado. La **Tabla 8** presenta los parámetros obtenidos mediante PVGIS para las ubicaciones en las que se implementó el sistema de alimentación fotovoltaica.

Tabla 8

Parámetros y producción estimada mediante PVGIS

Punto de instalación	Potencia instalada (kWp)	Pérdidas	Irradiación anual (Wh/m^2)	Producción anual (kWh)
Basílica	0,04	14 %	1392,62	44,04
Facultad de Ingeniería	0,04	14 %	1392,62	44,04

Nota: Elaboración propia.

Como se observa en la **Tabla 8**, los resultados obtenidos mediante PVGIS muestran valores similares de irradiación y producción entre las dos ubicaciones. Como referencia adicional, se consultó Global Solar Atlas para las mismas ubicaciones. Los valores obtenidos se utilizaron como complemento de los resultados de PVGIS y se resumen en la **Tabla 9**.

Tabla 9

Parámetros y producción estimada mediante Global Solar Atlas

Punto de instalación	GTI anual (kWh/m^2)	Producción anual con 40 Wp (kWh)
Basílica	1800,1	58,933
Facultad de Ingeniería	1805,3	58,933

Nota: Elaboración propia.

La producción media diaria correspondiente a las ubicaciones especificadas en la **Tabla 9** se obtienen mediante la siguiente ecuación:

$$E_{FV,dia} = \frac{E_{FV,anual}}{365} \quad (4)$$

$$E_{FV,dia} = \frac{58{,}933}{365} = 0{,}1615 kWh/dia$$

$$E_{FV,dia} = 161{,}46 Wh/dia$$

Considerando la demanda energética diaria obtenida en el apartado 4.2.2:

$$\mathrm{E_d} = 43{,}2 Wh/dia$$

se establece la relación entre la energía generada y la demanda:

$$\mathrm{R} = \frac{E_{FV,dia}}{E_d} \quad (5)$$

$$\mathrm{R} = \frac{161{,}46}{43{,}2} = 3{,}73$$

Por tanto, la producción media estimada del panel de 40 Wp supera la demanda energética diaria del nodo. Si bien el dimensionamiento permitió determinar que una potencia menor podría cubrir la demanda actual, se seleccionó el panel de 40 Wp debido a su disponibilidad comercial en el mercado ecuatoriano y a que su costo fue similar al de alternativas de menor potencia. Esta capacidad adicional proporciona un margen de generación que permitirá atender futuras ampliaciones o mejoras del sistema que puedan incrementar su consumo energético.

Para el montaje físico se considera una inclinación de 12°, correspondiente al soporte desarrollado para el panel. La producción obtenida de las plataformas se utilizó como referencia para la evaluación del recurso solar, mientras que la configuración

definitiva del soporte será considerada en el diseño físico del sistema. Los principales parámetros utilizados para esta verificación se presentan en la **Tabla 10**.

Tabla 10

Verificación del panel fotovoltaico seleccionado

Punto de instalación	**Valor**
Demanda energética diaria	43,2 Wh/día
Potencia del panel	40 Wp
Producción media estimada	161,46 Wh/día
Relación generación/demanda	3,73
Inclinación del soporte	12°

Nota: Elaboración propia.

4.2.4 Dimensionamiento del banco de baterías

El dimensionamiento del banco de baterías se realizó a partir de la demanda energética diaria del nodo y del periodo de autonomía requerido. Primero, se determinó la capacidad de almacenamiento necesaria para mantener el funcionamiento del sistema durante un periodo sin aporte de generación fotovoltaica. Posteriormente, este requerimiento se comparó con la capacidad de la batería seleccionada para el prototipo.

La demanda energética diaria determinada en el apartado 4.2.2 es:

$$\mathrm{E_d} = 43{,}2 Wh/dia$$

Para establecer inicialmente la capacidad energética requerida para un día de autonomía, se considera:

$$E_{\mathrm{bat,req}} = E_d \times d \qquad (6)$$

Donde ($E_{\mathrm{bat,req}}$) corresponde a la energía mínima requerida por la batería y (d) al número de días de autonomía.

Considerando inicialmente un día de autonomía:

$$E_{\text{bat,req}} = 43{,}2 \times 1$$

$$E_{\text{bat,req}} = 43{,}2$$

Para expresar esta capacidad en amperios-hora, considerando una tensión nominal de 12 V:

$$C_{\text{bat,req}} = \frac{E_{\text{bat,req}}}{V_{\text{bat}}} \quad (7)$$

$$C_{\text{bat,req}} = \frac{43{,}2}{12}$$

$$C_{\text{bat,req}} = 3{,}6\ Ah$$

Sin embargo, esta capacidad representa únicamente la energía necesaria sin considerar una profundidad de descarga limitada. Si se adopta una profundidad de descarga del 50 %, la capacidad nominal requerida se determina mediante:

$$C_{\text{bat,nrmal}} = \frac{C_{\text{bat,req}}}{DOD} \quad (8)$$

$$C_{\text{bat,nrmal}} = \frac{3{,}6}{0{,}50}$$

$$C_{\text{bat,nrmal}} = 7{,}2\ Ah$$

Por tanto, para proporcionar un día de autonomía utilizando hasta el 50 % de la capacidad de una batería de 12 V, se requiere una capacidad nominal mínima aproximada de 7,2 Ah.

4.2.4.1 Selección de la batería

Para el prototipo se seleccionó una batería POWEST FL1290GS, de tecnología VRLA, con una tensión nominal de 12 V y una capacidad de 9 Ah a 20 horas.

Su capacidad energética nominal es:

$$E_{\text{bat}} = 12 \times 9 \quad (9)$$

$$E_{\text{bat}} = 108Wh$$

Considerando una profundidad de descarga del 50 %, la energía utilizable corresponde a:

$$E_{\text{util}} = 108 \times 0{,}50 \quad (10)$$

$$E_{\text{bat}} = 54Wh$$

La autonomía proporcionada por la batería seleccionada es:

$$A = \frac{54}{43{,}2} \quad (11)$$

$$A = 1{,}25$$

Por tanto, la batería seleccionada supera la capacidad mínima requerida de 7,2 Ah y proporciona una autonomía aproximada de 1,25 días bajo la condición de descarga adoptada.

4.3 Diseño de la placa PCB de interconexión ESP32 CAM– Módulo W5500

Para integrar el ESP32-CAM con el módulo W5500 se diseñó una placa de interconexión que permite organizar las conexiones eléctricas y de comunicación entre ambos dispositivos. El diseño se desarrolló mediante herramientas de diseño electrónico, considerando la distribución de las conexiones y la disposición física de los elementos para obtener una placa compacta y adecuada para su integración en el nodo.

4.3.1 Diseño Esquemático

Para establecer la comunicación entre el ESP32-CAM y el módulo W5500 se definieron las conexiones eléctricas y de comunicación que serán consideradas para el diseño de la placa de interconexión. La comunicación entre ambos dispositivos se realiza

mediante la interfaz SPI, cuyas conexiones se detallan en la **Tabla 11** y adicionalmente su diagrama de conexiones se representan gráficamente en la **Figura12**.

Tabla 11

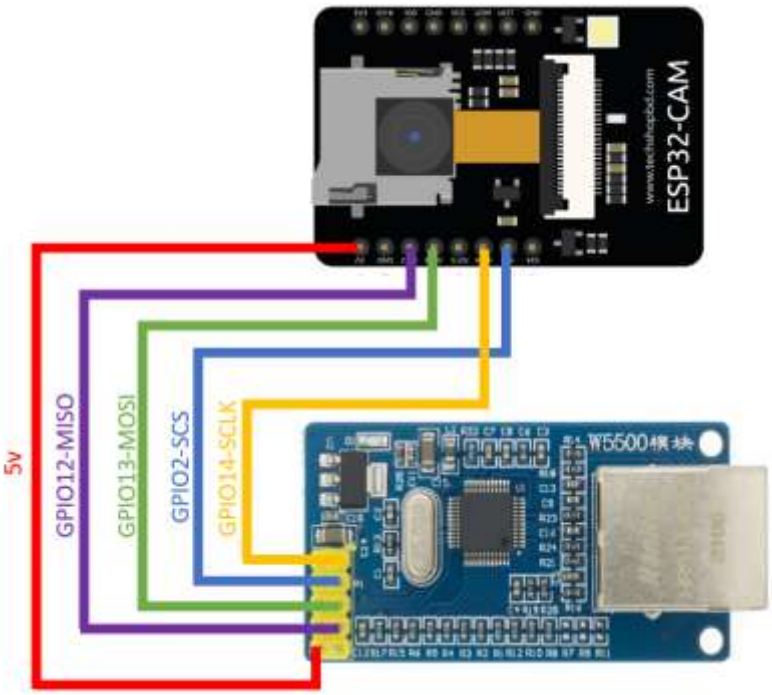
Conexiones entre el ESP32-CAM y el módulo W5500.

ESP32-CAM	W5500	Función
GPIO12	MISO	Comunicación SPI
GPIO13	MOSI	Comunicación SPI
GPIO14	SCLK	Reloj SPI
GPIO 2	SCS	Selección del dispositivo
5V	5V	Alimentación

Nota: Elaboración propia.

Figura12

Esquema electrónico de interconexión entre el ESP32-CAM y el módulo W5500.



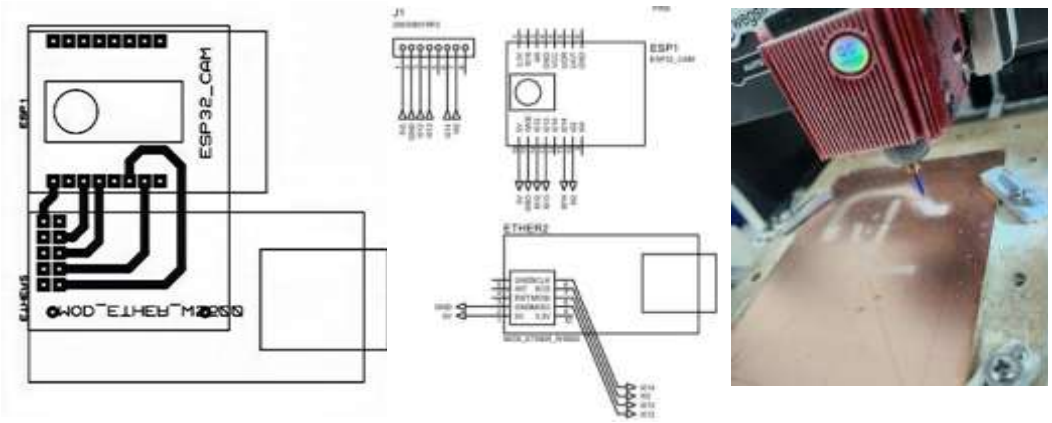
Nota: Elaboración propia.

4.3.2 Diseño PCB

A partir del esquema eléctrico desarrollado previamente, se realizó el diseño de la placa PCB en el software Proteus (ver **Figura 13**), definiendo la distribución física de los componentes y el trazado de las pistas de conexión. En esta etapa, se establecieron las rutas correspondientes a la alimentación y a las señales de comunicación entre el ESP32-CAM y el módulo W5500, procurando una distribución ordenada de las conexiones. Posteriormente se realizó el ruteado de las pistas utilizando la máquina CNC WEGSTR (ver **Figura 13** (c)).

Figura 13

Diseño de la placa PCB: (a) trazado de la placa PCB, (b) esquema eléctrico de los componentes, (c) Ruteado de PCB



(a) (b) (c)

Nota: Elaboración propia.

4.4 Diseño del compartimiento para los componentes electrónicos

El modelado de la carcasa se realizó mediante Autodesk Fusion 360, considerando el espacio necesario para alojar los componentes del sistema, las aberturas para sus conexiones y los elementos de acceso. En la **Figura 14** (a) y (b) se observa el modelo tridimensional de la carcasa, incluyendo la distribución interna de los componentes.

Figura 14

Diseño 3D de la carcasa: (a) Despiece de la carcasa y tapa frontal; (b) Disposición interna del controlador de carga solar y módulos electrónicos.



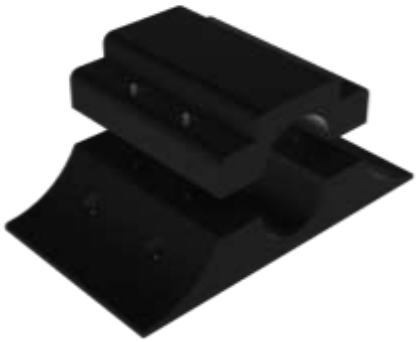
(a) (b)

Nota: Elaboración propia.

En la parte posterior de la carcasa se incorporaron dos piezas destinadas a su fijación sobre el tubo de soporte. Estas funcionan como abrazaderas y cuentan con las perforaciones necesarias para los elementos de sujeción. En la **Figura 15**, se muestra el diseño de estas piezas y su disposición en la parte posterior de la carcasa, y en la **Figura 16** (a) y (b) se presenta el modelo 3D del sistema, en donde se observa la vista frontal y trasera.

Figura 15

Diseño de las piezas de sujeción para el tubo de soporte



Nota: Elaboración propia

Figura 16

Modelo 3D del sistema ensamblado: (a) vista frontal y (b) vista posterior.



(a) (b)

Nota: Elaboración propia

4.4.1 Herramientas de software utilizadas

El desarrollo del sistema requiere la integración de diferentes herramientas de software que intervienen en las etapas de procesamiento y análisis del video; cada una de ellas cumple una función específica dentro del programa y permite integrar la detección, clasificación, seguimiento y gestión de los datos obtenidos durante el monitoreo. En la **Tabla 12**, se resumen las principales herramientas utilizadas y su función dentro del sistema.

Tabla 12

Herramientas de software utilizadas

Herramientas	Función dentro del sistema
Python	Desarrollo e integración del programa de procesamiento.
OpenCV	Captura y procesamiento de los fotogramas del video.
Ultralytics YOLO	Detección y clasificación de los vehículos.
NumPy	Gestión de coordenadas y configuración de las zonas de análisis.
Pandas	Organización y manejo de los datos generados durante las pruebas.
Matplotlib	Generación de gráficos para la verificación del procesamiento.

Nota: Elaboración propia.

4.5 Implementación del Sistema

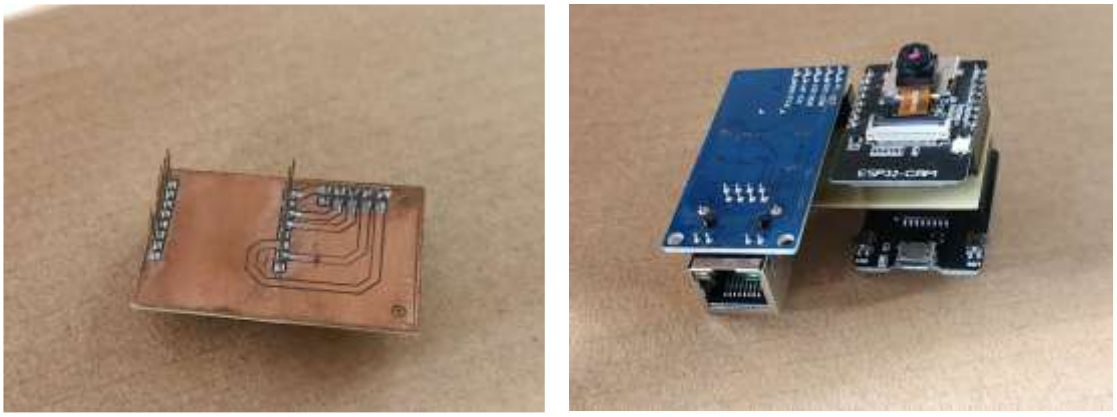
En la implementación del sistema de monitoreo vehicular a partir de los diseños desarrollados previamente, considerando la fabricación y ensamblaje de los componentes, la integración de las estructuras de soporte y los sistemas de alimentación. También se describe la programación, configuración y realización de las pruebas iniciales del sistema, para finalmente abordar su instalación y puesta en funcionamiento en los tres puntos de monitoreo establecidos.

4.5.1 Fabricación y ensamblaje de los componentes

Para la construcción del prototipo se fabricó la placa PCB de acuerdo con el diseño previamente desarrollado en Proteus para la interconexión entre la ESP32-CAM y el módulo W5500 y se elaboró el compartimiento en material ABS para alojar y proteger los componentes electrónicos, posteriormente se realizó el montaje de la ESP32-CAM y el W5500 sobre la placa y se integró el conjunto dentro del compartimiento. En la **Figura 17** y la **Figura 18**, se muestra el proceso de fabricación de la caja y ensamblaje del prototipo, donde se presenta la placa PCB y el montaje de la ESP32-CAM y el módulo W5500 sobre ella, la fabricación del compartimiento en material ABS y la **Figura 19** el montaje final del prototipo.

Figura 17

Placa PCB fabricada: (a) placa PCB, (b) montaje de la ESP32-CAM y el W5500.



(a) (b)

Nota: Elaboración propia.

Figura 18

Componentes de la carcasa fabricados en ABS.



Nota: Elaboración propia.

Figura 19

Prototipo ensamblado.



Nota: Elaboración propia.

4.5.2 Fabricación de las estructuras de soporte

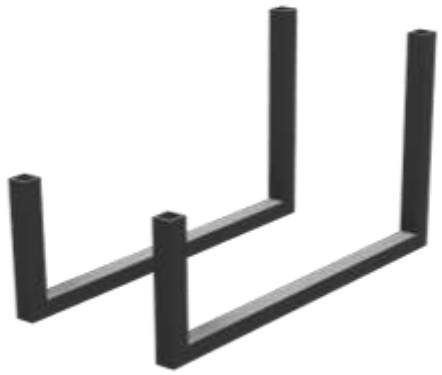
4.5.2.1 Diseño de las estructuras de soporte

Para la instalación de los diferentes elementos del sistema se diseñaron estructuras de soporte independientes, considerando las características y condiciones de montaje de cada componente. El soporte destinado al panel fotovoltaico fue modelado en Autodesk Fusion 360, definiendo su geometría para mantener una inclinación de 12° durante la instalación. Sus dimensiones son de 14 × 36 × 24 cm.

Por otra parte, se diseñó un tubo en forma de L destinado exclusivamente a la fijación de la carcasa del sistema. Este elemento presenta dimensiones de 13 × 25 cm y fue modelado considerando los puntos de unión necesarios para su instalación. En la **Figura 20**, se presenta el diseño del soporte para el panel fotovoltaico, mientras que la **Figura 21** muestra el modelo del tubo en forma de L destinado a la fijación de la carcasa.

Figura 20

Diseño del soporte para el panel fotovoltaico.



Nota: Elaboración propia.

Figura 21

Diseño del tubo en forma de L para la fijación de la carcasa.



Nota: Elaboración propia.

4.5.2.2 Fabricación

A partir de los modelos desarrollados, se fabricaron en metal las dos estructuras de soporte. El soporte del panel conserva la geometría establecida para proporcionar la inclinación de 12°, mientras que el tubo en forma de L fue fabricado de acuerdo con las dimensiones definidas para permitir la fijación independiente de la carcasa.

Los elementos obtenidos después del proceso de fabricación se presentan en la **Figura 22** (a) y (b), donde se distinguen el soporte destinado al panel fotovoltaico y el tubo en forma de L utilizado para la instalación de la carcasa.

Figura 22

Estructuras metálicas fabricadas para el montaje del panel fotovoltaico y (b) la carcasa.



(a) (b)

Nota: Elaboración propia.

4.5.3 Integración y pruebas del sistema fotovoltaico

La integración del sistema fotovoltaico comprendió la conexión del panel solar con el controlador de carga y la batería para suministrar energía al prototipo de acuerdo con los requerimientos establecidos previamente, durante esta etapa, se realizaron pruebas de funcionamiento con el prototipo ensamblado para verificar la alimentación del sistema y el comportamiento de sus componentes, como se muestra en la **Figura 23** (a) y (b), donde se observa la integración del sistema y la medición de tensión realizada durante la prueba, obteniéndose un valor aproximado de 13 V registrado en el controlador de carga y comprobado mediante el multímetro.

Figura 23

(a) Integración y (b) pruebas del sistema fotovoltaico con el prototipo.



(a) (b)

Nota: Elaboración propia.

Los valores obtenidos se contrastaron con los requerimientos energéticos determinados previamente para el sistema. Considerando el consumo calculado del ESP32-CAM y del módulo W5500, así como la capacidad de almacenamiento de la batería, se verificó que la energía disponible permite mantener el funcionamiento del prototipo y proporcionar las condiciones necesarias para la carga de la batería durante el período de operación establecido.

4.5.4 Funcionamiento del sistema

Las pruebas preliminares de funcionamiento se realizaron inicialmente en el punto de Posgrados donde se verificó el comportamiento general del sistema antes de su implementación definitiva en los demás puntos de monitoreo.

El funcionamiento del sistema comprende una secuencia de etapas que permiten realizar el monitoreo vehicular desde la captura del video hasta la obtención de los datos

procesados, en el siguiente esquema se presentan las fases que conforman el funcionamiento general del sistema (ver **Figura 24**)

Figura 24

Esquema general del sistema



Nota: Elaboración propia.

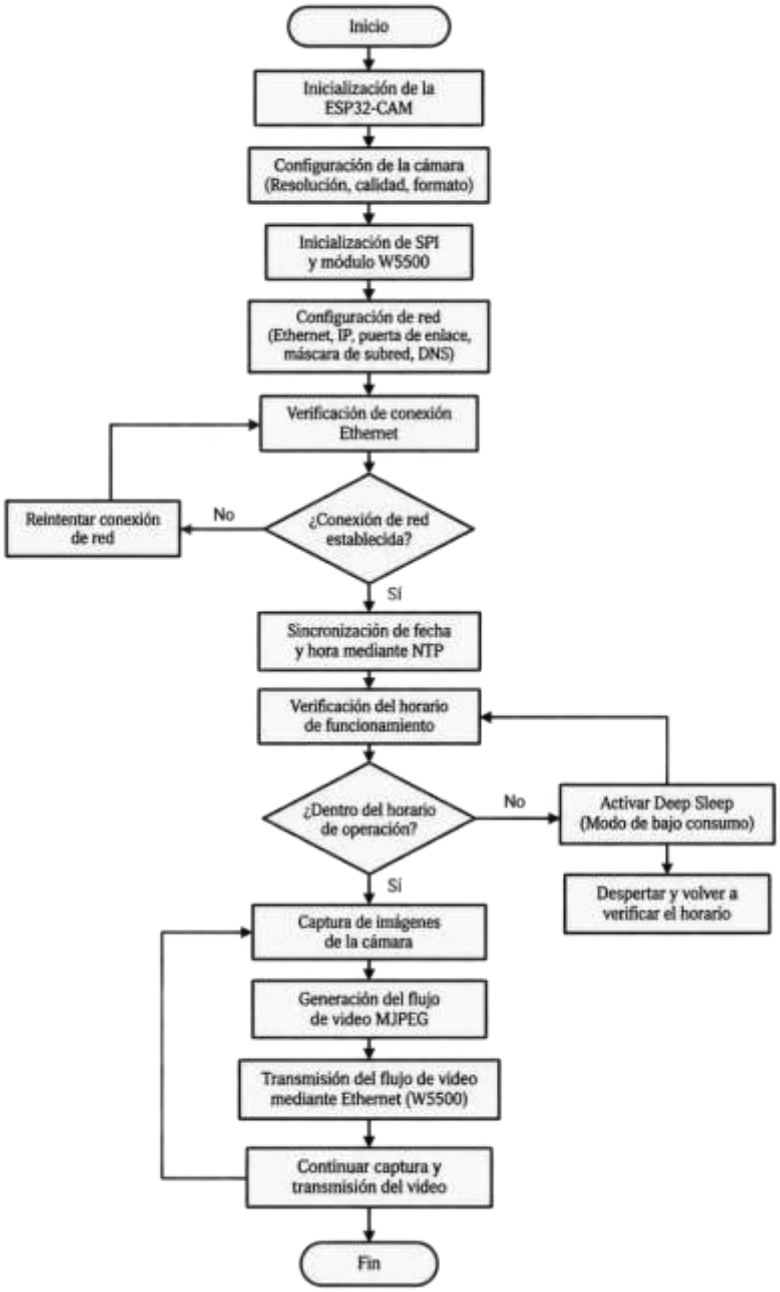
4.5.4.1 Programación del sistema

Con el propósito de representar de manera estructurada la lógica de funcionamiento de los programas desarrollados, se elaboraron los diagramas de flujo correspondientes a las etapas de captura, transmisión y procesamiento del video. Estos permiten visualizar la secuencia de operaciones, las condiciones de decisión y los procesos que intervienen en la ejecución del sistema, facilitando la comprensión de la

interacción entre el código de la ESP32-CAM con el módulo W5500 (ver **Figura 25**)y el programa de procesamiento desarrollado en Python (ver **Figura 26**).

Figura 25

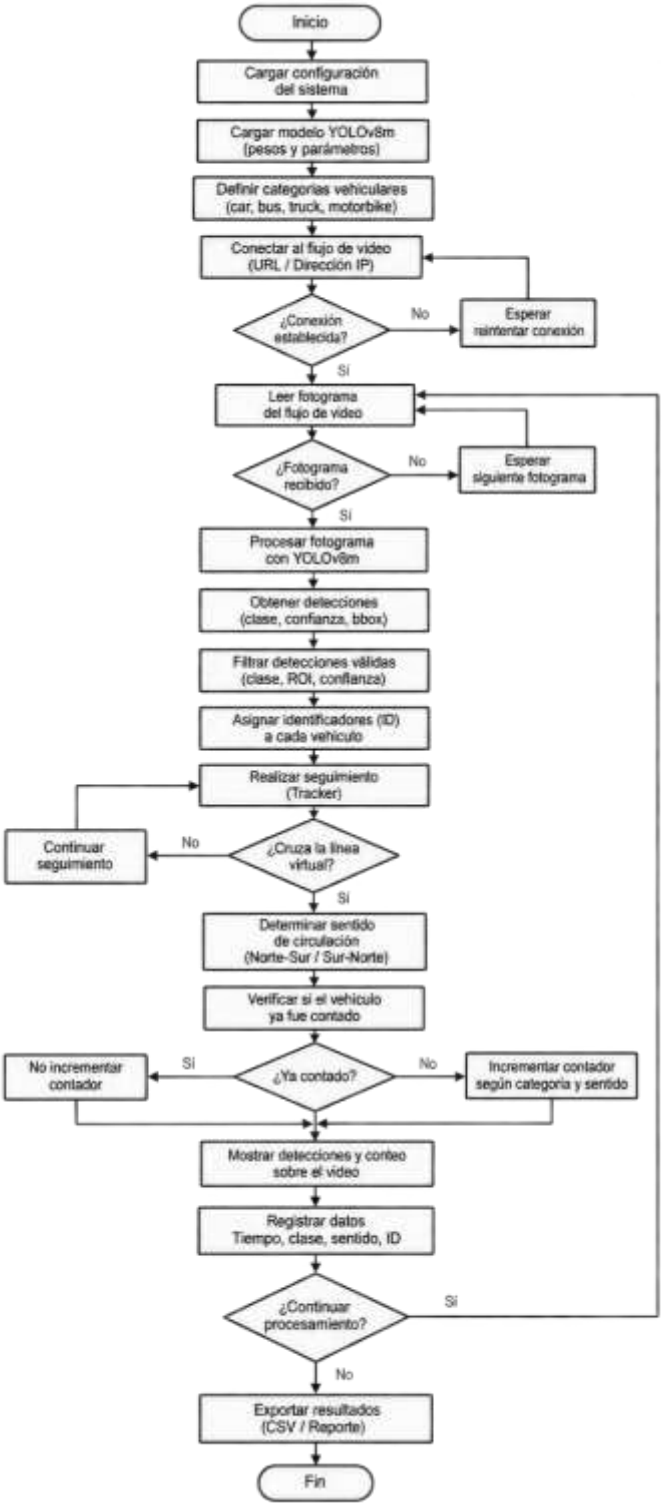
Diagrama de flujo del programa implementado en la ESP32-CAM y el módulo W5500.



Nota: Elaboración propia.

Figura 26

Diagrama de flujo del programa de procesamiento y conteo vehicular desarrollado en Python.



Nota: Elaboración propia.

4.5.5 Instalación en los puntos de monitoreo

La instalación definitiva del sistema se realizó en tres puntos de monitoreo dentro del campus universitario, correspondientes a la Basílica, la Facultad de Ingeniería, Industria y Construcción y la Dirección de Posgrado. En la **Tabla 13**, se detallan las principales características de cada ubicación, considerando las condiciones disponibles para el montaje de la cámara, la alimentación eléctrica y la orientación hacia la vía de estudio.

Tabla 13

Características de los puntos de monitoreo.

Punto de monitoreo	Ubicación	Alimentación	Característica de circulación
Basílica	Av. Las Américas y Humboldt	Sistema fotovoltaico	Unidireccional
Facultad de Ingeniería, Industria y Construcción	Av. Las Américas y General Torres	Sistema fotovoltaico	Unidireccional
Dirección de Posgrado	Cam. A Patamarca	Red eléctrica	Bidireccional

Nota: Elaboración propia

En la Basílica y la Facultad de Ingeniería, Industria y Construcción se implementó el sistema fotovoltaico debido a la ubicación de estos puntos y a la necesidad de disponer de una fuente de alimentación independiente de la red eléctrica, mientras que en la Dirección de Posgrado se utilizó directamente la red eléctrica debido a la cercanía de un punto de conexión disponible.

En la **Figura 27** (c), (b) y (c) se presenta la instalación final de los equipos en los tres puntos de monitoreo, donde se observa la ubicación de las cámaras y sus respectivas estructuras de soporte.

Figura 27

Instalación de los sistemas de monitoreo en los tres puntos establecidos: (a) Basílica, (b) Facultad de Ingeniería y (c) Posgrados.



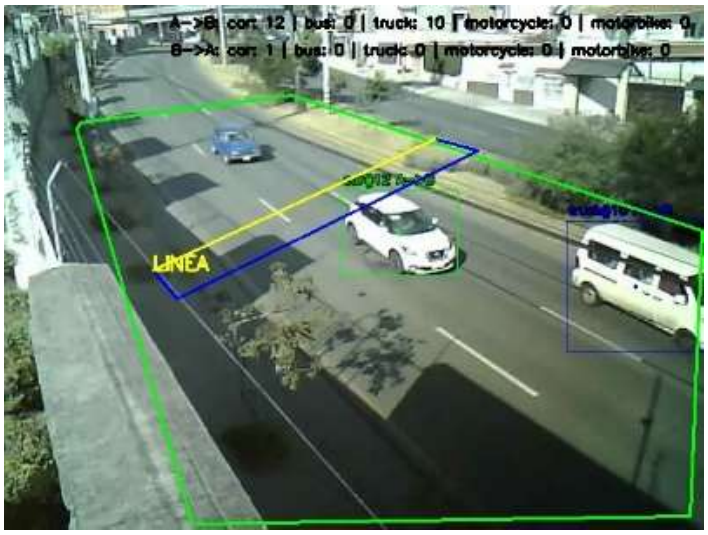
(a) (b) (c)

Nota: Elaboración propia

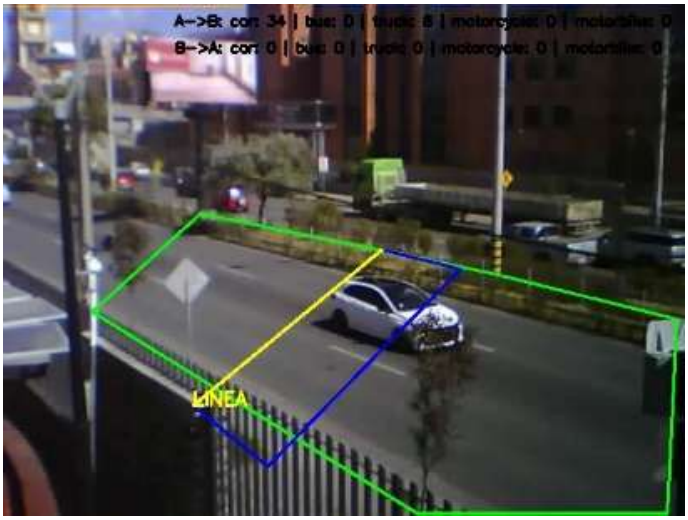
Una vez instalados los equipos, se realizaron pruebas de funcionamiento en los tres puntos de monitoreo, verificando la transmisión del video y la generación del conteo vehicular en cada ubicación, como se muestra en la **Figura 28**.

Figura 28.

Funcionamiento del sistema de conteo vehicular en los puntos de monitoreo: (a) Basílica, (b) Facultad de Ingeniería, Industria y Construcción y (c) Dirección de Posgrado.



(a)



(b)



(c)

Nota: Elaboración propia

CAPÍTULO V

5. Análisis de Resultados

Para la evaluación del sistema, durante cuatro semanas se realizaron registros de video en los tres puntos de monitoreo, dentro del horario comprendido entre las 06h00 y las 18h00. De estos registros se seleccionaron los días lunes, jueves y domingo de cada semana, utilizando fragmentos aleatorios de 15 minutos correspondientes a cada punto de monitoreo. En total, se analizaron 36 fragmentos de video, equivalentes a 9 horas de evaluación. En cada jornada, se procesaron los videos correspondientes a los tres puntos de monitoreo: Basílica, la Facultad de Ingeniería, Industria y Construcción, y la Dirección de Posgrado. El modelo YOLO permitió identificar y realizar el conteo taxonómico de los vehículos de acuerdo con las cuatro categorías establecidas: automóviles, buses, camiones y motocicletas.

5.1 Conteo mediante modelo YOLO

Para registrar el comportamiento del sistema automatizado en condiciones operativas reales, se procesaron los fragmentos de video correspondientes a los tres puntos de monitoreo a lo largo de las cuatro semanas. La **Tabla 14**, se elaboró con el propósito de organizar cronológicamente las salidas del algoritmo YOLO, permitiendo observar la capacidad del modelo para clasificar y contabilizar de forma continua las cuatro categorías vehiculares bajo diferentes densidades de tránsito.

Tabla 14

Resultados del conteo vehicular mediante el algoritmo YOLO.

Semana	Día	Sectores											
		Basílica				Facultad de Ingeniería, Industria y Construcción				Dirección de Posgrado			
		Auto	Bus	Camión	Moto	Auto	Bus	Camión	Moto	Auto	Bus	Camión	Moto
S1	Lunes	289	2	64	38	316	2	58	43	257	12	48	16
	Jueves	352	2	164	239	367	1	64	48	241	14	53	13
	Domingo	161	1	31	32	188	3	49	34	155	8	37	12
S2	Lunes	237	2	67	35	301	2	55	41	269	11	51	18
	Jueves	271	3	61	40	283	3	61	46	232	13	47	15
	Domingo	170	1	28	35	157	1	22	27	116	11	29	8
S3	Lunes	285	1	69	37	324	1	66	50	254	15	56	12
	Jueves	262	2	57	43	295	2	57	42	276	10	44	17
	Domingo	153	1	33	36	164	2	30	31	126	9	17	11
S4	Lunes	268	1	63	34	261	3	62	47	248	13	52	14
	Jueves	246	2	66	41	280	1	60	45	263	12	49	18
	Domingo	149	1	30	38	171	2	27	24	284	14	26	10
Total, por sector		**4243**				**4219**				**3536**			

Nota: Elaboración Propia

5.1.1 Resultados consolidados del conteo vehicular mediante YOLO

Con el fin de sintetizar el volumen global procesado por el software y caracterizar la proporción de cada clase detectada, se agruparon las salidas del modelo en la **Tabla 15**. Esta tabla permite establecer la distribución taxonómica capturada por el sistema automático.

Tabla 15

Total de vehículos contabilizados por clasificación mediante YOLO.

Ubicación	Auto	Bus	Camión	Moto
Basílica	2 843	19	733	648
Facultad de Ingeniería, Industria y Construcción	3 107	23	611	478

Dirección de Posgrado	2 721	142	509	164
Total, por tipo de vehículo	**8 671**	**184**	**1 853**	**1 290**

Nota: Elaboración Propia

5.2 Conteo manual de referencia

Para evaluar cuantitativamente el desempeño de un sistema de visión artificial, es indispensable contrastar sus resultados contra un registro de referencia real. El conteo manual se realizó sobre los mismos fragmentos de video utilizados para la evaluación del sistema automático, con el propósito de disponer de un registro de referencia para su posterior validación. La **Tabla 16** refleja los resultados obtenidos durante las cuatro semanas de evaluación.

Tabla 16

Resultados del conteo manual de vehículos

Semana	**Día**	**Sectores**											
		Basílica				**Facultad de Ingeniería, Industria y Construcción**				**Dirección de Posgrado**			
		Auto	Bus	Camión	Moto	Auto	Bus	Camión	Moto	Auto	Bus	Camión	Moto
S1	Lunes	326	3	79	46	357	3	71	54	297	14	53	20
	Jueves	404	3	198	298	410	2	78	61	279	16	58	16
	Domingo	193	2	37	42	208	4	60	43	173	9	40	16
S2	Lunes	237	292	3	82	44	337	3	67	51	301	12	56
	Jueves	319	4	76	51	315	4	75	62	262	14	51	19
	Domingo	198	2	35	47	174	2	26	38	135	13	32	11
S3	Lunes	285	334	2	80	48	357	2	78	63	290	17	62
	Jueves	294	3	65	57	327	3	67	58	314	11	47	22
	Domingo	191	2	38	48	180	3	36	39	148	10	19	15
S4	Lunes	268	325	2	72	43	290	4	72	59	276	15	58
	Jueves	306	3	80	54	314	2	73	58	308	13	53	24
	Domingo	167	2	34	51	189	3	33	35	318	16	29	13
Total, por sector		**5085**				**4850**				**4031**			

Nota: Elaboración Propia

5.2.1 Resultados consolidados del conteo vehicular manual

Con el propósito de analizar la distribución de los vehículos según su clasificación, se consolidaron los datos correspondientes al conteo manual en la **Tabla 17**. En esta tabla, se presentan los valores acumulados por categoría y punto de monitoreo, los cuales serán utilizados posteriormente como referencia para evaluar el desempeño del sistema.

Tabla 17

Conteo manual por clasificación y ubicación.

Ubicación	Auto	Bus	Camión	Moto
Basílica	3 349	31	876	829
Facultad de Ingeniería, Industria y Construcción	3 458	35	736	621
Dirección de Posgrado	3 101	160	558	212
Total, por tipo de vehículo	**9 908**	**226**	**2 170**	**1 662**

Nota: Elaboración Propia

5.3 Comparación de los resultados del conteo vehicular

La comparación entre los resultados del modelo YOLO y el conteo manual permite evaluar el desempeño del sistema automático en condiciones reales. Para cuantificar las diferencias obtenidas en cada escenario, se utilizaron dos métricas: la diferencia absoluta y el error porcentual.

En primer lugar, se calculó la diferencia absoluta (D) que representa la cantidad numérica de vehículos omitidos o contabilizados de más por el sistema, calculada mediante la ecuación (12) donde C_{YOLO} es el conteo automático obtenido mediante el modelo de Yolo y C_{manual} corresponde a los resultados adquiridos a través del conteo manual de cada uno de los fragmentos de video.

$$D = | C_{YOLO} - C_{manual} | \quad (12)$$

Por su parte, el error porcentual (E) que determina la magnitud relativa de dicha diferencia con respecto al conteo real, calculado mediante la ecuación (13).

$$E(\%) = \frac{| C_{YOLO} - C_{manual} |}{C_{manual}} \times 100 \quad (13)$$

Estas expresiones se aplican a los resultados obtenidos de las tablas de la sección **5.1** y **5.2** para cada ubicación y clasificación vehicular. Para analizar el comportamiento del sistema desde dos perspectivas distintas, la **Tabla 18** presenta la comparación consolidada según el punto de monitoreo, mientras que la **Tabla 19** desglosa las discrepancias en función de la categoría vehicular.

Tabla 18

Comparación del conteo vehicular por ubicación.

Ubicación	Conteo YOLO	Conteo manual	Diferencia absoluta	Error (%)
Basílica	4 243	5 085	842	16,56
Facultad de Ingeniería, Industria y Construcción	4 219	4 850	631	13,01
Dirección de Posgrado	3 536	4 031	495	12,28
Total, general	**11 998**	**13 966**	**1 968**	**14,09**

Nota: Elaboración Propia

Como se observa en la **Tabla 18**, el error del conteo automático alcanzó su mayor valor en la Basílica (16,56 %), seguido por la Facultad de Ingeniería (13,01 %), mientras que el sector de la Dirección de Posgrado presentó el menor error (12,28 %). A nivel global, el sistema contabilizó 11 998 unidades frente a los 13 966 reales, lo que representa una diferencia absoluta acumulada de 1 968 vehículos, equivalente a un error relativo del 14,09 % y una concordancia general del 85,91 %. La diferencia entre Basílica y Facultad de Ingeniería, Industria y Construcción resulta relevante, ya que ambos puntos se encuentran sobre la misma avenida y presentan circulación unidireccional. Sin embargo,

las condiciones de observación y el ángulo de la cámara generaron mayores dificultades en la Basílica. Su perspectiva elevada y lateral favoreció el ocultamiento entre vehículos, afectando el seguimiento durante el cruce de la línea virtual de conteo. Posgrado obtuvo la menor discrepancia, aunque también presentó casos de oclusión asociados con vehículos de mayor tamaño.

Tabla 19

Comparación del conteo vehicular por clasificación.

Clasificación	Conteo YOLO	Conteo manual	Diferencia absoluta	Error (%)
Automóvil	8 671	9 908	1 237	12,48
Bus	184	226	42	18,58
Camión	1 853	2 170	317	14,61
Motocicleta	1 290	1 662	372	22,38
Total, general	11 998	13 966	1 968	14,09

Nota: Elaboración Propia

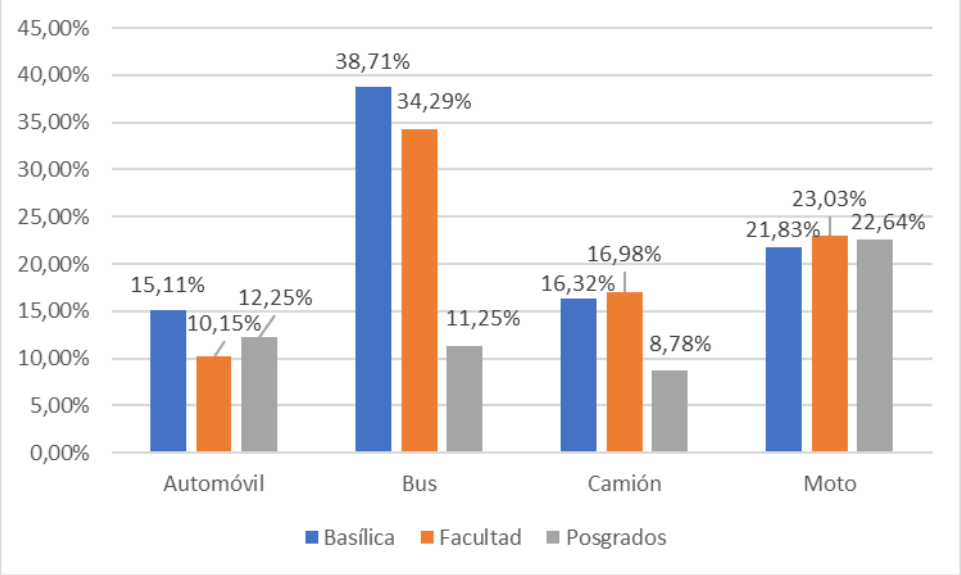
La desagregación por clases presentada en la **Tabla 19** evidencia que la exactitud del conteo automatizado guarda una relación directa con la escala visual del objeto y su dinámica de circulación dentro de la calzada. Los automóviles obtuvieron el menor error porcentual con un 12,48 % debido a que su forma definida facilitó la detección; sin embargo, al ser el tipo de vehículo más común, representaron la mayor cantidad de pérdidas reales con 1 237 unidades no registradas, principalmente en momentos de tráfico denso donde circulaban muy pegados entre sí. En el caso de los camiones, el error fue del 14,61 % (317 omisiones), logrando identificarse con facilidad por su gran tamaño, aunque en varias ocasiones su presencia tapó la visibilidad de otros autos cercanos. Los buses registraron un error del 18,58 %, pero con solo 42 unidades omitidas; este porcentaje alto se debe a que pasaron muy pocos buses por la vía (226 en total), por lo que perder unos cuantos elevó bastante el resultado porcentual. Por último, las motocicletas presentaron la mayor dificultad en la detección con un error del 22,38 % (372 omisiones), lo cual se

explica porque ocupan muy poco espacio en la imagen y a la limitada información visual disponible para su identificación ya que suelen cambiar de carril o esquivar otros autos, complicando su seguimiento continuo al momento de pasar por la línea de conteo.

La comparación por clasificación vehicular muestra que la discrepancia del conteo varió según el tipo de vehículo. Para complementar estos resultados, se relacionó el error porcentual de cada categoría con el punto de monitoreo, permitiendo identificar las variaciones del desempeño del sistema entre los diferentes sectores, como se muestra en la **Figura 29**.

Figura 29

Error porcentual del conteo vehicular según clasificación y punto de monitoreo.



Nota: Elaboración Propia

La distribución presentada en la **Figura 29** permite apreciar que el comportamiento del error no fue uniforme entre las categorías y los puntos de monitoreo. Los buses mostraron la mayor variación entre sectores, mientras que las motocicletas mantuvieron valores elevados y relativamente similares en los tres puntos. Los automóviles y camiones presentaron diferencias más moderadas entre ubicaciones. En

conjunto, estos resultados evidencian que las condiciones de observación tuvieron una influencia diferente según la categoría vehicular.

5.4 Variación temporal del error de conteo según semana y día.

Los resultados muestran el comportamiento del sistema de conteo a lo largo de las cuatro semanas y evidencian las variaciones del error porcentual entre los puntos de monitoreo. Para analizar la estabilidad temporal del sistema de visión artificial, se evaluó la evolución del error porcentual relativo determinado mediante la Ecuación **(13)**, presentada en la sección **5.3**. La **Tabla 20** presenta los valores correspondientes a cada semana y punto de monitoreo.

Tabla 20

Error porcentual del conteo automático por semana y ubicación.

Semana	Basílica (%)	Facultad de Ingeniería, Industria y Construcción (%)	Dirección de Posgrado (%)
S1	15,7	13,18	12,61
S2	17,61	13,43	11,73
S3	15,75	12,28	12,68
S4	17,56	13,16	12,09

Nota: Elaboración Propia

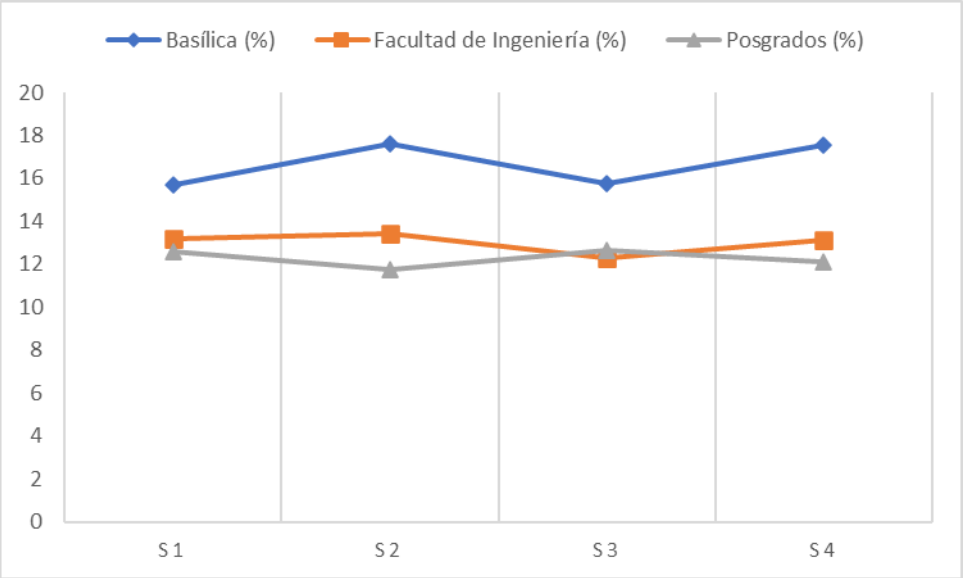
El análisis de **Tabla 20** permite identificar diferencias en el comportamiento semanal del error entre los tres puntos evaluados. La Facultad de Ingeniería, Industria y Construcción, y la Dirección de Posgrado mantuvieron una notable regularidad a lo largo del mes, con rangos de variación estrechos de 12,28 % a 13,43 % y de 11,73 % a 12,68 %, respectivamente. Por su parte, la Basílica presentó una mayor oscilación en sus mediciones, situándose entre 15,70 % y 17,61 %, alcanzando su valor más alto en la segunda semana y repitiendo un incremento en la cuarta. En términos generales, no se observa una tendencia continua de incremento o descenso progresivo del error con el paso

del tiempo, lo que confirma que el modelo mantuvo un rendimiento operativo estable durante todo el mes.

La evolución semanal del error permite apreciar visualmente las variaciones registradas entre los puntos de monitoreo, como se presenta en la **Figura 30**.

Figura 30

Evolución semanal del error porcentual según punto de monitoreo.



Nota: Elaboración Propia

La tendencia presentada en la **Figura 30** permite distinguir las curvas correspondientes a la Facultad de Ingeniería, Industria y Construcción y la Dirección de Posgrado exhiben trayectorias próximas y sensiblemente horizontales durante las cuatro semanas, confirmando la consistencia del algoritmo ante flujos vehiculares estables. En contraste, la trayectoria de la Basílica muestra fluctuaciones más marcadas entre periodos; este comportamiento se vincula directamente con las variaciones en el volumen de camiones y motocicletas que transitaron por dicho sector en cada semana, cuya presencia acentuó temporalmente los eventos de oclusión bajo la perspectiva lateral de la cámara.

Por otra parte, para analizar con mayor detalle el comportamiento del error durante el periodo de evaluación, se desagregaron los resultados por semana, día y punto de monitoreo. La **Tabla 21** permite comparar el comportamiento de cada sector entre las jornadas evaluadas e identificar las diferencias existentes dentro de cada semana.

Tabla 21

Error porcentual del conteo automático según semana, día y ubicación.

Semana	Día	Basílica (%)	Facultad de Ingeniería, Industria y Construcción (%)	Dirección de Posgrado (%)
S1	Lunes	13,44	13,61	13,28
	Jueves	16,17	12,89	13,01
	Domingo	17,88	13,02	10,92
S2	Lunes	19	12,88	10,97
	Jueves	16,67	13,82	11,27
	Domingo	17,02	13,75	14,14
S3	Lunes	15,52	11,8	12,24
	Jueves	13,13	12,97	11,93
	Domingo	20,07	12,02	15,1
S4	Lunes	17,19	12,24	10,9
	Jueves	19,86	13,65	14,07
	Domingo	14,17	13,85	11,17

Nota: Elaboración Propia

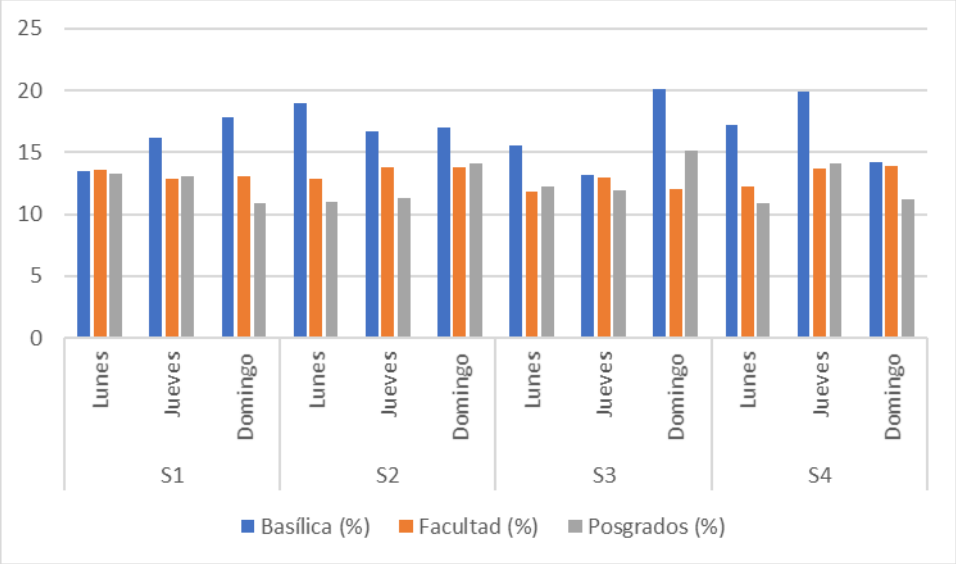
La **Tabla 21** evidencia diferencias en el comportamiento del error según el día y el punto de monitoreo. La Basílica registró los valores más elevados en la gran mayoría de sesiones (alcanzando un máximo de 20,07 % el domingo de la Semana 3), mientras que la Facultad de Ingeniería. Industria y Construcción, y la Dirección de Posgrado conservaron cifras más moderadas y cercanas entre sí en casi todas las fechas. Al comparar las jornadas, se advierte que los domingos suelen reportar márgenes de error porcentual ligeramente mayores, debido a que el volumen total de vehículos en día de

descanso es menor, haciendo que la omisión de unas pocas unidades incida con más peso en la fórmula del error. Por el contrario, los lunes y jueves mostraron en su mayoría registros más equilibrados, asociados a un flujo vehicular continuo y predecible.

La variación diaria del error permite examinar la respuesta detallada del comportamiento del sistema entre las jornadas evaluadas, como se muestra en la **Figura 31**.

Figura 31

Variación diaria del error porcentual por semana, día y punto de monitoreo.



Nota: Elaboración Propia

La **Figura 31** evidencia que el comportamiento del error no fue uniforme durante las cuatro semanas en todas las ubicaciones. La Basílica concentra las variaciones más pronunciadas entre jornadas continuas (por ejemplo, entre el jueves y el domingo de las semanas 3 y 4), reflejando una mayor sensibilidad a las condiciones particulares de tráfico de cada sesión. En cambio, la Facultad de Ingeniería, Industria y Construcción conservó un comportamiento sumamente uniforme, con márgenes contenidos en una franja estrecha (entre 11,80 % y 13,85 %) a lo largo de las doce sesiones. Por su parte, la

Dirección de Posgrado presentó un desempeño intermedio, registrando errores reducidos en las jornadas laborales (en torno al 11 % y 12 %) y repuntes aislados en domingos específicos. En síntesis, estos contrastes confirman que las condiciones puntuales de cada día influyeron con mayor notoriedad en los sectores con mayor variedad de clases vehiculares (como la Basílica), mientras que las zonas con predominio de automóviles ligeros mantuvieron una precisión altamente constante.

5.5 Evaluación mediante métricas de detección.

Dado el elevado volumen de vehículos registrados durante las cuatro semanas de evaluación (13 966 unidades en total), la verificación manual cuadro a cuadro de cada detección sobre la totalidad de los videos resultaba impracticable en términos de tiempo. Por esta razón, se seleccionó una muestra estratificada de 40 vehículos de referencia por cada punto de monitoreo, conformada de manera equitativa por 10 unidades de cada una de las cuatro categorías analizadas. Este procedimiento garantizó la representatividad estadística de todas las clases, especialmente de aquellas con menor presencia natural en el tránsito, como los buses. Cada vehículo de referencia fue auditado manualmente y contrastado con las detecciones generadas por el sistema, clasificándose bajo los siguientes criterios:

- Verdadero positivo (TP): Vehículo de prueba que fue correctamente detectado, clasificado y contabilizado al cruzar la línea virtual.
- Falso positivo (FP): Detección espuria generada sin correspondencia con un vehículo real, o bien un conteo duplicado producto de la fragmentación del identificador durante el seguimiento.

- Falso negativo (FN): Vehículo real de la muestra que no fue detectado o que no se contabilizó al cruzar la línea virtual.

A partir de esta clasificación se calcularon las métricas de Precisión, Recall y F1-score

La Precisión mide la proporción de detecciones que correspondieron efectivamente a vehículos reales, determinándose mediante la Ecuación **(14)**:

$$Precision = \frac{TP}{TP + FP} \times 100 \tag{14}$$

El Recall (o sensibilidad) representa la capacidad del modelo para identificar la totalidad de los vehículos reales que circularon por la zona de aforo, calculándose mediante la Ecuación **(15)**:

$$Recall = \frac{TP}{TP + FN} \times 100 \tag{15}$$

El F1-score constituye la media armónica entre la Precisión y el Recall, proporcionando un indicador balanceado del desempeño del sistema mediante la Ecuación **(16)**:

$$F1 = 2\left(\frac{Precision \cdot Recall}{Precision + Recall}\right) \tag{16}$$

Los valores de TP, FP y FN se organizan por punto de monitoreo para establecer el comportamiento de las detecciones. La **Tabla 22** presenta los valores correspondientes a cada ubicación.

Tabla 22

Valores de TP, FP y FN por punto de monitoreo.

Punto de monitoreo	TP	FP	FN
Posgrados	34	5	6
Facultad de Ingeniería	32	7	8
Basílica	30	9	10

Nota: Elaboración Propia

La **Tabla 22**, evidencian que en los tres puntos evaluados los falsos negativos superaron a los falsos positivos. Esta relación ratifica la tendencia al subconteo analizada en las secciones anteriores, demostrando que el sistema tiende a omitir vehículos antes que a generar registros falsos o duplicados. La mayor concentración de falsos negativos se produjo en la Basílica (FN = 10), atribuible al ángulo lateral y elevado de la cámara, el cual favoreció que vehículos de mayor altura cubrieran temporalmente a unidades más pequeñas al momento de atravesar la línea virtual. Por el contrario, la Facultad de Ingeniería y Posgrado presentaron menores pérdidas (FN = 8 y FN = 6, respectivamente), beneficiados por perspectivas de captura de imagen que redujeron las zonas de ocultamiento. Estas diferencias en las detecciones se reflejan posteriormente en las métricas de desempeño calculadas para cada punto de monitoreo. La **Tabla 23** permite valorar cómo los verdaderos positivos, falsos positivos y falsos negativos registrados se traducen en los valores de Precisión, Recall y F1-score.

Tabla 23

Métricas de evaluación del sistema de detección.

Punto de monitoreo	Precisión (%)	Recall (%)	F1-score (%)
Dirección de Posgrado	87,18	85	86,08
Facultad de Ingeniería, Industria y Construcción	82,05	80	81,01
Basílica	76,92	75	75,96

Nota: Elaboración Propia

Los resultados de la **Tabla 23** muestran que las métricas de detección guardan una correspondencia directa con la calidad del aforo de cada ubicación. El sector de la Dirección de Posgrado obtuvo el rendimiento más alto en las tres métricas, alcanzando un F1-score del 86,08 %, favorecido por un encuadre visual despejado que permitió mantener el seguimiento continuo de las trayectorias vehiculares. La Facultad de Ingeniería, Industria y Construcción exhibió un comportamiento intermedio con un F1-score del 81,01 %, donde la perspectiva de la cámara facilitó la detección, pero no eliminó del todo los solapamientos en momentos de tráfico agrupado. Finalmente, la Basílica registró los valores más bajos con un F1-score del 75,96 %. La diferencia de 10,12 puntos porcentuales en el F1-score entre Posgrado y la Basílica evidencia que las variaciones en el desempeño del sistema no se deben a fallas en la red neuronal YOLO, sino a las limitaciones geométricas de la perspectiva de captura en cada estación de monitoreo.

5.6 Estimación de Costos

La estimación de los costos asociados a la implementación del sistema de monitoreo y conteo vehicular, considerando los componentes electrónicos, elementos del sistema fotovoltaico, materiales de instalación, recursos utilizados durante el desarrollo y demás rubros requeridos para la ejecución del proyecto. En la **Tabla 24** se presenta el presupuesto correspondiente a la ejecución del trabajo de titulación

Tabla 24

Presupuesto para la ejecución del trabajo de titulación.

PRESUPUESTO PARA LA EJECUCIÓN DEL TRABAJO DE TITULACIÓN				
PRESUPUESTO				
Descripción	**Unidad**	**Cantidad**	**P. Unitario**	**P. Total**
Módulo ESP32-CAM OV2640 2MP, Wifi, Bluetooth, incluye módulo grabador micro USB	U	3	13,20	39,6
Módulo W5500 Ethernet Arduino	U	3	7,99	23,97
Cable USB A micro USB	U	3	2,20	6,6
Batería Powest 12v 9ah Ups Recargable Alarma Cerco Seca	U	3	25,00	75
Cable Concéntrico de 2 x 10 THHN / THWN	U	10	2,55	25,5
Rollo De Cable Utp Cambo Cat6 305m Exterior Doble Chaqueta	U	1	64,99	64,99
Conectores Plug Rj45 Cat 6 Delta Funda 50 Unidades Para Kit	U	1	6,99	6,99
Controlador Regulador De Carga Panel Solar Pwm 20a	U	3	15,00	45
Panel Solar Monocristalino Powest 40w 2.19a, Ip67 57 X 40cm	U	3	24,99	74,97
Elementos Varios	Global	1	50,00	50
			Total (USD)	412,62

Nota: Elaboración Propia

Como parte de la evaluación económica se comparó el costo de implementación del sistema desarrollado con el costo registrado para un equipo comercial destinado al conteo vehicular en Ecuador. El sistema desarrollado presenta un costo total de USD 412,62 para las tres estaciones de monitoreo lo que corresponde a USD 137,54 por estación. Por otra parte, se identificó un valor de USD 24 050,00 para un equipo contador de tráfico registrado en procesos de adquisición pública en Ecuador. En términos de costo

de adquisición del equipamiento el valor por estación del sistema desarrollado representa aproximadamente el 0,57 % del valor del equipo comercial. Esto corresponde a una reducción del 99,43 % en el costo de implementación por estación y evidencia una menor inversión inicial para el sistema desarrollado.

5.7 Discusión

La integración de una arquitectura distribuida basada en nodos de captura embebidos y el modelo YOLOv8m demostró viabilidad operativa para el aforo continuo en entornos urbanos reales, alcanzando una concordancia global del 85,91 % (error porcentual de 14,09 %) frente a un universo manual de referencia de 13 966 vehículos. El hallazgo metodológico más relevante es que las discrepancias exhibieron una tendencia constante al subconteo en todas las sesiones analizadas, descartando la presencia de falsos positivos provocados por ruido visual o artefactos ambientales y situando las limitaciones del sistema en eventos específicos de omisión de paso.

Al contrastar estos hallazgos con investigaciones previas, se identifican puntos de convergencia en la sensibilidad del algoritmo según el entorno de captura. Worachairungreung et al. (2026), quienes utilizaron YOLOv8m, la misma variante empleada en esta investigación, mediante cámaras CCTV para la detección y clasificación vehicular. Los autores señalan que factores como la iluminación, la resolución, la densidad del tránsito y el ocultamiento entre vehículos pueden afectar el desempeño. Estas dificultades también se presentaron durante las pruebas realizadas, principalmente cuando varios vehículos se aproximaban entre sí o cruzaban simultáneamente la línea virtual, provocando pérdidas durante el seguimiento. El efecto fue más evidente en la Basílica, donde la perspectiva elevada y lateral de la cámara redujo la visibilidad de algunos vehículos.

Por su parte Bermeo Cabrera y Cárdenas Loja (2025), quienes emplearon YOLOv8n determinaron que los vehículos de dos ruedas (Motos) registran las tasas de error más críticas. Aunque, sus métricas no son directamente comparables con las de esta investigación, ambos trabajos evidencian que el tipo de vehículo influye en el desempeño del sistema. Este fenómeno se corrobora en la presente investigación, donde las motocicletas alcanzaron la mayor discrepancia (22,38 %).

Asimismo, Luna Díaz (2025) empleó YOLO para el conteo vehicular mediante imágenes aéreas obtenidas con UAV, utilizando una perspectiva cenital diferente a las cámaras fijas de este estudio, lograron precisiones superiores al eliminar las oclusiones intervehiculares gracias a planos perpendiculares al suelo, ventaja ausente en estaciones fijas a ras de calzada. Esta diferencia permite destacar que el desempeño de un sistema de visión artificial también está condicionado por la forma en que se obtiene la imagen. Por ello, los errores encontrados no pueden atribuirse únicamente al YOLOv8m, sino también a la posición de las cámaras, la interacción entre vehículos y el seguimiento necesario para realizar el conteo.

Desde el punto de vista tecnológico, la propuesta integra ESP32-CAM, W5500, comunicación Ethernet, YOLOv8m, seguimiento vehicular y línea virtual de conteo, junto con alimentación fotovoltaica en los nodos correspondientes. Esto permite combinar captura, transmisión y procesamiento automatizado en una solución orientada al monitoreo continuo, con una demanda estimada de 43,2 Wh/día cubierto mediante el sistema fotovoltaico diseñado.

Finalmente, las principales limitaciones estuvieron relacionadas con la perspectiva de las cámaras, la interferencia entre vehículos y el seguimiento de motocicletas. Como mejoras futuras se plantea optimizar la ubicación de las cámaras y de la línea virtual, incorporar algoritmos de seguimiento con memoria de apariencia

(como BoT-SORT o ByteTrack) para retener la identidad del vehículo tras oclusiones temporales, así como entrenar submodelos dedicados a la detección de objetos pequeños que fortalezcan el aforo de motocicletas en entornos urbanos de tráfico denso. En conjunto, los resultados permiten considerar el sistema como una alternativa viable para automatizar el conteo vehicular en condiciones reales, con potencial para apoyar el monitoreo vial mediante una solución de bajo costo y operación autónoma.

CAPÍTULO VI

6. Conclusiones y Recomendaciones

6.1 Conclusiones

Se sustentó el diseño e implementación del sistema mediante una revisión sistemática de la literatura científica especializada en visión artificial, detección de objetos y conteo vehicular, lo que permitió seleccionar la combinación de modelos de aprendizaje profundo (YOLO) y hardware de bajo costo como la alternativa más adecuada y actualizada.

El sistema desarrollado alcanzó un costo total de implementación de USD 412,62 para las tres estaciones de monitoreo, equivalente a un costo promedio de USD 137,54 por estación. El procesamiento centralizado mediante YOLO se realizó utilizando infraestructura informática disponible para el desarrollo de la investigación, por lo que el costo de adquisición del computador utilizado para este proceso no se incorporó al presupuesto de implementación de las estaciones.

Se validó la viabilidad técnica del sistema de conteo vehicular basado en YOLO y línea virtual de conteo, procesando un total de 36 fragmentos de video (9 horas de metraje, distribuidas en tres sectores durante cuatro semanas) y registrando un universo automático de 11 998 vehículos con una concordancia global del 85,91 % frente al aforo manual 13 966 vehículos.

La perspectiva y el ángulo de inclinación de la cámara demostraron ser las variables de mayor impacto en la precisión de conteo del sistema. El sector Posgrado obtuvo el mejor rendimiento (12,28 % de error y 86,08 % de F1-score) estos debido a una vista despejada y frontal; en contraste, la perspectiva lateral en la Basílica incrementó el

error al 16,56 % (F1-score de 75,96 %) debido al ocultamiento mutuo entre vehículos contiguos durante el cruce de la línea de conteo.

La precisión taxonómica es directamente proporcional a las dimensiones aparentes del vehículo. Mientras los automóviles alcanzaron un error moderado del 12,48 %, las motocicletas presentaron la mayor vulnerabilidad con un error del 22,38 %, evidenciando que la detección de objetos de menor tamaño exige mayor resolución de captura.

El sistema evidenció una tendencia sistemática hacia el subconteo en todas las sesiones y sectores analizados (diferencia acumulada de 1 968 vehículos omitidos). La auditoría de la muestra estratificada ratificó que las fallas correspondieron de forma predominante a falsos negativos causados por oclusiones momentáneas (FN > FP en las tres estaciones), Basílica: 10 vs. 9 falsos positivos; Facultad de Ingeniería: 8 vs. 7; Posgrados: 6 vs. 5, descartando la presencia de falsos positivos masivos por ruido visual.

En definitiva, la investigación desarrollada demuestra que la combinación de hardware de bajo costo (DiY), alimentación fotovoltaica autónoma y modelos avanzados de visión artificial como YOLOv8m constituye una alternativa técnica y económicamente viable frente a los sistemas comerciales de aforo vehicular. Con una concordancia global del 85,91 % y un costo por estación significativamente menor al de las soluciones tradicionales, el prototipo validado en los sectores del campus matriz de la Universidad Católica de Cuenca no solo cumple con los requerimientos operativos de monitoreo continuo en entornos urbanos reales, sino que sienta una base escalable para futuros proyectos institucionales de gestión vial y movilidad inteligente.

6.2 Recomendaciones

Se recomienda reemplazar la línea única transversal por polígonos virtuales o líneas segmentadas para cada carril de circulación, de modo que el sistema pueda contabilizar de forma individual a los vehículos que cruzan la zona de detección de manera simultánea.

Incorporar algoritmos de seguimiento multiobjetivo con persistencia temporal y memoria de apariencia (como ByteTrack o BoT-SORT), de modo que el sistema mantenga la identidad del vehículo aun si este desaparece temporalmente tras una oclusión breve, evitando la pérdida del vector de movimiento antes de cruzar la línea de conteo.

Es aconsejable aumentar la resolución de captura de la cámara para mejorar la distinción entre clases vehiculares con siluetas similares, en particular las motocicletas y las camionetas con balde tapado, un modelo común en la ciudad que el algoritmo tiende a confundir con la clase "camión" debido a la similitud en las características visuales de las camionetas y a la forma en que el modelo procesa la información geométrica y contextual.

Procurar la instalación de los sensores de captura bajo ángulos semi-cenitales o perpendiculares a la calzada, aprovechando infraestructuras viales elevadas como puentes peatonales o pórticos de señalización. Esta orientación vertical reduce al mínimo las proyecciones oblicuas que ocasionaron las oclusiones críticas registradas en la estación Basílica, eliminando así el ángulo lateral que redujo el desempeño del sistema en el dicho sector.

Se sugiere construir una base de datos local con ejemplos de vehículos parcialmente ocluidos casos en los que el cruce simultáneo con otro vehículo de mayor

tamaño deja visible solo una parte de la carrocería, así como de camionetas con balde tapado, y reentrenar el algoritmo con este conjunto de datos para mejorar su precisión de clasificación ante estas condiciones particulares del tránsito local.

7. Referencias

Adeniran, A. O., Adeniran, A. A., Ogieva, M. O., & Ogwuche, G. (2026). Adopción de Sistemas de Transporte Inteligentes (ITS) en la planificación del transporte urbano. *Descubre Global*, *4*(1). https://doi.org/10.1007/s44282-025-00312-3

Alahdal, N. M., Abukhodair, F., Meftah, L. H., & Cherif, A. (2024). Real-time Object Detection in Autonomous Vehicles with YOLO. *Procedia Computer Science*, *246*(C), 2792–2801. https://doi.org/10.1016/J.PROCS.2024.09.392

Alhasnawi, B. N., Almutoki, S. M. M., Hussain, F. F. K., Harrison, A., Bazooyar, B., Zanker, M., & Bureš, V. (2024). A new methodology for reducing carbon emissions using multi-renewable energy systems and artificial intelligence. *Sustainable Cities and Society*, *114*, 105721. https://doi.org/10.1016/J.SCS.2024.105721

Alvarado, J. A., Galarza, K. G., Asanza, W. R., & Olivo, B. M. (2025). Comparison of Computer Vision and Convolutional Neural Networks for Vehicle Parking Control. *Journal of Automation, Mobile Robotics and Intelligent Systems*, *19*(2), 26–33. https://doi.org/10.14313/JAMRIS-2025-011

Amarilla, M. F., Devincenzi, G. H., & Ramos, R. C. (2016). *Conteo vehicular utilizando visión artificial. XVII Congreso Argentino de Vialidad y Tránsito, Buenos Aires, Argentina.* https://congresodevialidad.org.ar/congreso2016/TRA/TRA-064.pdf

Annamalai, P., Sivasakthi, Pandi, V. S., Stanley Raja, T. A., Vasudhevan, V., & Saravanan, S. (2024). Técnicas de aprendizaje automático eficientes energéticamente: Avanzando en las capacidades de computación en el borde para operaciones sostenibles de redes IoT. *2024 1ª Conferencia Internacional Sobre Sostenibilidad y Avances Tecnológicos En El Ámbito de La Ingeniería, SUSTAINED 2024*, 149–154. https://doi.org/10.1109/SUSTAINED63638.2024.11073959

Basantes, M. E., & Gaibor, Y. S. (2025). *Desarrollo e implementación de un sistema de conteo vehicular basado en visión por computadora.* https://dspace.ups.edu.ec/handle/123456789/31929

Borse, R., Bhattacharyya, A., Sarkar, A., & Bhattacharjee, S. (2024). Employing YOLO model for traffic monitoring on roadways. *2024 International Conference on Intelligent Computing and Sustainable Innovations in Technology, IC-SIT 2024*. https://doi.org/10.1109/IC-SIT63503.2024.10862128

Buele, C. O., & Chazi, A. M. (2019). *Modelación del sistema fotovoltaico conectado a la red eléctrica en la Universidad Politécnica Salesiana sede Cuenca*. https://dspace.ups.edu.ec/handle/123456789/18210

Cabrera, J. X., & Cárdenas, P. X. (2023). *Análisis y propuesta de solución para el tráfico existente en la intersección de las calles Paseo de los Cañaris y Yana Urco*. http://dspace.ups.edu.ec/handle/123456789/25545

Caiza, L. J. (2019). *Desarrollo e implementación de un analizador de tráfico vehicular basado en la obtención y tratamiento de los datos contenidos en mapas geográficos con información de flujo vehicular georreferenciados*. http://bibdigital.epn.edu.ec/handle/15000/20029

Cangrejo, C. D., & Ovalle, J. E. (2024). *Implementación de sistema de detección de objetos mediante ESP32-CAM para el control de calidad de la estación FMS-210*. http://hdl.handle.net/11349/92931

Cinnamon, J., & Dai, J. (2026). Unseen things: Computer vision and the uneven geographies of AI failure. *Digital Geography and Society*, *10*. https://doi.org/10.1016/J.DIGGEO.2026.100173

De Magistris, S., & Del Bimbo, A. (2023). SDG-Based AI Ethics: An Analysis of Recent Computer Vision Research. *2023 IEEE International Conference on Metrology for EXtended Reality, Artificial Intelligence and Neural Engineering, MetroXRAINE 2023 - Proceedings*, 560–564. https://doi.org/10.1109/METROXRAINE58569.2023.10405796

Estefanía, J., & Crespo, A. (2026). *Diseño de un sistema fotovoltaico ON GRID para la disminución del consumo de energía de la red convencional en el Aeropuerto Mariscal La Mar de la ciudad de Cuenca*. https://dspace.ups.edu.ec/handle/123456789/33230

Estevez, C. J., Dang, M., Bass, R., Estevez, C., Jose, ;, Dang, M. ;, José, C., & Cabrera, E. (2025). Mobile Computer Vision Application for Agricultural Disease Detection of Pepper Diseases using Two-Stage Deep Learning System. *SMU Data Science Review*, *9*(2), 7. https://scholar.smu.edu/datasciencereview/vol9/iss2/7

Farid, Das, P. K., Islam, M., & Sina, E. (2025). Bangladeshi Vehicle Classification and Detection Using Deep Convolutional Neural Networks With Transfer Learning. *IEEE Access*, *13*, 26429–26455. https://doi.org/10.1109/ACCESS.2025.3539713

Febres-Loaiza, D. A., Jácome-Galarza, L. R., Jaramillo-Sangurima, W. E., & Jaramillo-Luzuriaga, S. A. (2024). Determinación de la densidad vehicular en ciudades intermedias por medio de visión por computador. *Revista Tecnológica - ESPOL*, *36*(E1), 68–79. https://doi.org/10.37815/RTE.V36NE1.1216

Hamied, A., Mellit, A., Benghanem, M., & Boubaker, S. (2023). IoT-Based Low-Cost Photovoltaic Monitoring for a Greenhouse Farm in an Arid Region. *Energies 2023, Vol. 16, Page 3860*, *16*(9), 3860. https://doi.org/10.3390/EN16093860

Hashemian, A., & Derakhshanfard, N. (2025). Target tracking in Internet of Things using reinforcement learning. *Array*, *28*, 100551. https://doi.org/10.1016/J.ARRAY.2025.100551

Hidalgo, G. (2024). *Anuario de Estadísticas de Transporte, 2023*.

Huang, F., Yao, J., Su, C., & Xu, S. (2025). An online multi-camera multi-vehicle tracking using lightweight YOLO11 and improved association strategies. *Journal of King Saud University - Computer and Information Sciences*, *37*(7). https://doi.org/10.1007/S44443-025-00190-4

IEA. (2022). *Global EV Outlook 2022*. https://www.iea.org/reports/global-ev-outlook-2022

Jayapradha, J., Sathishkumar, R., Swathi, R., Tejaswini, S., & Rinjima, J. R. (2024). Computer Vision Based Enhanced Traffic Management System Using YOLO. *2024 International Conference on System, Computation, Automation and Networking, ICSCAN 2024*. https://doi.org/10.1109/ICSCAN62807.2024.10894375

José, R., & Romero, F. (2024). *Estimación de las relaciones velocidad-densidad, densidad-flujo y velocidad-flujo del tránsito vehicular en las vías urbanas de Bogotá y Medellín basado en técnicas de estadística inferencial para múltiples regímenes de flujo*. Universidad Nacional de Colombia. https://repositorio.unal.edu.co/handle/unal/87497

Kaur, J., & Singh, W. (2022). Tools, techniques, datasets and application areas for object detection in an image: a review. *Multimedia Tools and Applications 2022 81:27*, *81*(27), 38297–38351. https://doi.org/10.1007/S11042-022-13153-Y

Lashkov, I., Yuan, R., & Zhang, G. (2025). Machine learning-based vehicle detection and tracking based on headlight extraction and GMM clustering under low illumination conditions. *Expert Systems with Applications*, *267*, 126240. https://doi.org/10.1016/J.ESWA.2024.126240

Li, K. (2022). Applications of Deep Learning in Object Detection. *Proceedings - 2022 International Conference on Computers, Information Processing and Advanced Education, CIPAE 2022*, 436–442. https://doi.org/10.1109/CIPAE55637.2022.00097

Li, X., Qian, M., Zhang, Y., Guo, J., & Wang, R. (2025). *Evaluation and optimization of YOLO algorithms for multitype vehicle detection*. 126. https://doi.org/10.1117/12.3090165

Lin, C., Yang, W., Wu, G., & Li, W. (2026). Real-time object detection via enhanced YOLOv5s with attention mechanisms and multi-scale feature fusion. *Discover Artificial Intelligence 2026 6:1*, *6*(1), 390-. https://doi.org/10.1007/S44163-026-01107-4

Majumder, M., & Wilmot, C. (2023). Automated Vehicle Counting from Pre-Recorded Video Using You Only Look Once (YOLO) Object Detection Model. *Journal of Imaging*, *9*(7). https://doi.org/10.3390/JIMAGING9070131

Marin, G. A., & Gaiceanu, M. (2025). Sistema Autónomo de Energía Fotovoltaica. *SIELMEN 2025 - Actas de La 15ª Conferencia Internacional Sobre Sistemas Electromecánicos y Energéticos*, 77–82. https://doi.org/10.1109/SIELMEN67352.2025.11260797

Masita, K. L., Hasan, A. N., & Shongwe, T. (2020). Deep learning in object detection: A review. *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems, IcABCD 2020 - Proceedings*. https://doi.org/10.1109/ICABCD49160.2020.9183866

Moudoud, H., Mlika, Z., Cherkaoui, S., & Khoukhi, L. (2023). Green machine learning for Internet-of-Things: Current solutions and future challenges. *Green Machine Learning Protocols for Future Communication Networks*, 161–175. https://doi.org/10.1201/9781003230427-6

Mrudula, S. T., Meenakshi, Ritonga, M., Sivakumar, S., Jawarneh, M., F, S., Keerthika, T., Rane, K. P., & Roy, B. (2024). Internet of things and optimized knn based intelligent transportation system for traffic flow prediction in smart cities. *Measurement: Sensors*, *35*, 101297. https://doi.org/10.1016/J.MEASEN.2024.101297

Neamah, S. B., & Karim, A. A. (2023). Real-time Traffic Monitoring System Based on Deep Learning and YOLOv8. *ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY*, *11*(2), 137–150. https://doi.org/10.14500/aro.11327

Ning, Z., Vandersteegen, M., Van Beeck, K., Goedemé, T., & Vandewalle, P. (2024). BENCHMARK DE CONSUMO ENERGÉTICO PARA INFERENCIA DE IA EMBEBIDA. *Actas de Las Conferencias Internacionales Sobre Computación Aplicada y WWW/Internet 2024*. https://www-scopus-com.vpn.ucacue.edu.ec/pages/publications/85214130241?origin=scopusAI

Parra, J. A., & Torres, R. L. (2017). *Estimación de los sobrecostos por congestionamiento vehicular en rutas de la ciudad de Cuenca*. http://dspace.uazuay.edu.ec/handle/datos/7502

Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of Deep Learning for Object Detection. *Procedia Computer Science*, *132*, 1706–1717. https://doi.org/10.1016/J.PROCS.2018.05.144

Porreca, F., Frustaci, F., & Gravina, R. (2025). A Codesign Framework for the Development of Next Generation Wearable Computing Systems. *Sensors*, *25*(21). https://doi.org/10.3390/S25216624

Pudaruth, S., Boodhun, I. M., & Onn, C. W. (2024). Reducing Traffic Congestion Using Real-Time Traffic Monitoring with YOLOv8. *International Journal of Advanced Computer Science and Applications*, *15*(10), 1068–1079. https://doi.org/10.14569/IJACSA.2024.01510109

Pupiales, L. M. (2024). *Diseño de sistemas fotovoltaicos autónomos para viviendas en zonas aisladas de la red eléctrica : diseño de un prototipo de sistema de generación fotovoltaico residencial autónomo para consumo nivel 3.* https://bibdigital.epn.edu.ec/handle/15000/26212

Ramón, G. B., & Suquinagua, L. F. (2023). *Análisis y solución al congestionamiento vehicular producido en la intersección Av. 24 de Mayo y Max Uhle de la ciudad de Cuenca*. http://dspace.ups.edu.ec/handle/123456789/25708

Rani, S., & Dalal, S. (2024). Comparative analysis of machine learning techniques for enhanced vehicle tracking and analysis. *Transportation Engineering*, *18*, 100271. https://doi.org/10.1016/J.TRENG.2024.100271

Rezgui, J., Daze, S., & Mehenni, G. (2019). Vehicles Classification and Brand Recognition Using Convolution Network and Neural Networks. *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*. https://doi.org/10.1109/SMARTNETS48225.2019.9069784

Rojek, I., Prokopowicz, P., Piechowiak, M., Kotlarz, P., Náprstková, N., & Mikołajewski, D. (2026). El impacto del análisis de datos basado en Internet de las Cosas, computación en el borde e inteligencia artificial en la eficiencia energética en entornos inteligentes. *Applied Sciences (Suiza), 16*(1). https://doi.org/10.3390/app16010225

Rosini, M. D. (2013). Tráfico vehicular. *Entendiendo Sistemas Complejos*, 123–138. https://doi.org/10.1007/978-3-319-00155-5_9

Saki, S., & Soori, M. (2026). Artificial intelligence, machine learning and deep learning in advanced transportation systems, a review. *Multimodal Transportation*, *5*(1). https://doi.org/10.1016/j.multra.2025.100242

Sangprasert, N., & Inthavisas, K. (2025). A Smart Marine Debris Management System for a sustainable coastal city: An IoT-based application. *Cleaner Waste Systems*, *11*, 100262. https://doi.org/10.1016/J.CLWAS.2025.100262

Shubneet, & Yadav, A. R. (2026). Sistemas de transporte inteligentes: arquitecturas, aplicaciones y direcciones futuras. *Inteligencia Artificial y Lógica Difusa Para Sistemas de Transporte Inteligentes de Próxima Generación*, 1–18. https://doi.org/10.1201/9781003649670-1

Suhana, S., Prathap, B. R., Narang, K., & Anto, I. (2025). IoT-Based Emergency Vehicle Detection Using YOLOv8. *Journal of Automation, Mobile Robotics and Intelligent Systems*, *19*(2), 79–88. https://doi.org/10.14313/jamris-2025-018

Tang, J., Ye, C., Zhou, X., & Xu, L. (2024). YOLO-Fusion and Internet of Things: Advancing object detection in smart transportation. *Alexandria Engineering Journal*, *107*, 1–12. https://doi.org/10.1016/J.AEJ.2024.09.012

Tuba, J. J., & Jimenez, S. S. (2025). *Análisis y diseño de un sistema fotovoltaico en la Universidad Politécnica Salesiana sede Cuenca, aprovechando la infraestructura existente*. https://dspace.ups.edu.ec/handle/123456789/31767

Viscidi, L., & O'Connor, R. (2017). *La energía del transporte: Un enfoque en el transporte urbano en América Latina – Inter-American Dialogue*. https://thedialogue.org/analysis/the-energy-of-transportation-a-focus-on-latin-american-urban-transportation-2/?lang=es

Xu, D., Yang, Y., Zhu, L., Dai, C., Chen, T., & Zhao, J. (2022). Vehicle Re-Identification System Based on Appearance Features. *Security and Communication Networks, 2022*. https://doi.org/10.1155/2022/1833362

Xu, X., Zang, S., Bilal, M., Xu, X., & Dou, W. (2024). Intelligent architecture and platforms for private edge cloud systems: A review. *Future Generation Computer Systems*, *160*, 457–471. https://doi.org/10.1016/J.FUTURE.2024.06.024

Yang, H., Zhang, Y., Zhang, Y., Meng, H., Li, S., & Dai, X. (2021). A Fast Vehicle Counting and Traffic Volume Estimation Method Based on Convolutional Neural Network. *IEEE Access*, *9*, 150522–150531. https://doi.org/10.1109/ACCESS.2021.3124675

Zhang, W., Li, Y., Hu, J., & Wang, N. (2026). A study on front vehicle collision warning method based on lightweight YOLOv8 and DeepSort. *Egyptian Informatics Journal*, *33*, 100861. https://doi.org/10.1016/J.EIJ.2025.100861

Anexos (Scripts, Planos, Articulo)

Anexo 1 Programación del sistema de conteo vehicular

```
import cv2
import numpy as np
import pandas as pd
from datetime import datetime
from ultralytics import YOLO
from config import (
    model_path, target_classes, class_colors, min_confidence_global
)
from tracker import VehicleTracker

# URL del stream RTSP/HTTP proporcionado por el usuario
VIDEO_URL = 'http://172.16.102.47/stream'  # Cambia a
'http://172.16.100.126' si es HTTP MJPEG

# Cargar zonas y línea desde archivo existente (ajusta la ruta si es
necesario)
data = np.load(r"C:\Users\Andres\Documents\goge\Deteccion de
vehiculos\config_zonas.npz", allow_pickle=True)
roi = data["polygon_points"]
zona_control = data["zona_control"]
linea = [tuple(pt) for pt in data["linea_entrada"]]

def bbox_in_roi(x1, y1, x2, y2, roi):
    puntos = [(x1, y1), (x2, y2), ((x1 + x2) // 2, (y1 + y2) // 2)]
    return any(cv2.pointPolygonTest(roi, pt, False) >= 0 for pt in
puntos)

model = YOLO(model_path)
cap = cv2.VideoCapture(VIDEO_URL)

fps = cap.get(cv2.CAP_PROP_FPS) or 25
frame_size = (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),
int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))
now_str = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
output_video_path = fr"C:\Users\Andres\Documents\Videos de conteo
vehicular\conteo_webcam_{now_str}.avi"
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter(output_video_path, fourcc, fps, frame_size)

tracker = VehicleTracker(roi, zona_control, linea, target_classes)
excel_data = []

print("[INFO] Procesando video en tiempo real desde:", VIDEO_URL)
print("Presiona 'q' para detener y exportar resultados.")

while cap.isOpened():
```

```
    ret, frame = cap.read()
    if not ret:
        print("[ADVERTENCIA] No se pudo leer frame del stream.
Esperando...")
        cv2.waitKey(100)
        continue

    results = model.track(frame, persist=True)[0]
    detecciones = []
    if results and results.boxes is not None:
        for box in results.boxes:
            conf = float(box.conf[0])
            cls_id = int(box.cls[0])
            cls_nm = model.names[cls_id]
            if not hasattr(box, 'xyxy') or box.xyxy.shape[-1] < 4:
                continue
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            cx, cy = (x1 + x2) // 2, (y1 + y2) // 2
            if cls_nm not in target_classes:
                continue
            if roi is not None and len(roi) >= 3 and not bbox_in_roi(x1,
y1, x2, y2, roi):
                continue
            if conf < min_confidence_global:
                continue
            box_id = None
            if hasattr(box, 'id') and box.id is not None and len(box.id)
> 0 and box.id[0] is not None:
                box_id = int(box.id[0])
            detecciones.append({
                'bbox': (x1, y1, x2, y2),
                'centro': (cx, cy),
                'clase': cls_nm,
                'conf': conf,
                'id': box_id
            })

    drawables = tracker.update(detecciones)
    for x1, y1, x2, y2, cx, cy, cls, number, _, _, _, _, sentido in
drawables:
        if number is not None:
            excel_data.append({
                'Frame': tracker.frame,
                'ID': number,
                'Clase': cls,
                'Sentido': sentido,
                'BBox_x1': x1,
                'BBox_y1': y1,
                'BBox_x2': x2,
```

```
                'BBox_y2': y2,
                'Centro_x': cx,
                'Centro_y': cy
            })

    # Visualización
    font_scale = 0.38
    thickness = 1
    ids_vistos = set()
    for x1, y1, x2, y2, cx, cy, cls, number, _, _, _, _, sentido in
drawables:
        key = (number, cls, sentido, x1, y1, x2, y2)
        if key in ids_vistos or number is None:
            continue
        ids_vistos.add(key)
        color = class_colors.get(cls, (120, 120, 120))
        label = f"{cls}#{number if number else ''} {sentido}"
        y_txt = max(16, y1 - 6)
        x_txt = x1
        cv2.rectangle(frame, (x1, y1), (x2, y2), color, 1)
        cv2.putText(frame, label, (x_txt + 2, y_txt),
cv2.FONT_HERSHEY_SIMPLEX, font_scale, (0, 0, 0), 2, cv2.LINE_AA)
        cv2.putText(frame, label, (x_txt + 2, y_txt),
cv2.FONT_HERSHEY_SIMPLEX, font_scale, color, thickness, cv2.LINE_AA)
        cv2.circle(frame, (cx, cy), 2, color, -1)

    resumen = tracker.resumen()
    h_txt = 20
    margin = -20
    resumen_A_B = "A->B: " + " | ".join(f"{cls}: {resumen[('A->B',
cls)]}" for cls in target_classes)
    resumen_B_A = "B->A: " + " | ".join(f"{cls}: {resumen[('B->A',
cls)]}" for cls in target_classes)
    (w_A_B, _), _ = cv2.getTextSize(resumen_A_B,
cv2.FONT_HERSHEY_SIMPLEX, 0.48, 2)
    (w_B_A, _), _ = cv2.getTextSize(resumen_B_A,
cv2.FONT_HERSHEY_SIMPLEX, 0.48, 2)
    w_txt = max(w_A_B, w_B_A)
    x_txt = frame.shape[1] - w_txt - margin
    y_txt1 = h_txt
    y_txt2 = h_txt + 25
    cv2.putText(frame, resumen_A_B, (x_txt, y_txt1),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 0), 2)
    cv2.putText(frame, resumen_B_A, (x_txt, y_txt2),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 0), 2)
    cv2.polylines(frame, [roi], True, (0, 255, 0), 2)
    cv2.polylines(frame, [zona_control], True, (255, 0, 0), 2)
    cv2.line(frame, *linea, (0, 255, 255), 2)
```

```
    cv2.putText(frame, "LINEA", linea[0], cv2.FONT_HERSHEY_SIMPLEX, 0.6,
(0, 255, 255), 2)

    out.write(frame)
    cv2.imshow("Conteo Vehicular Webcam", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q") or key == 27:  # 27 es el código ASCII de ESC
        print("[INFO] Proceso detenido por el usuario. Exportando
resultados...")
        break

cap.release()
out.release()
cv2.destroyAllWindows()

# --- Exportar resumen y gráficas igual que en analizar_video.py ---
excel_path = fr"C:\Users\Andres\Documents\Videos de conteo
vehicular\resumen_webcam_{now_str}.xlsx"
csv_path = fr"C:\Users\Andres\Documents\Videos de conteo
vehicular\resumen_webcam_{now_str}.csv"

if not excel_data:
    print("[ADVERTENCIA] No hay datos de conteo para exportar.")
else:
    print(f"[INFO] Procesando {len(excel_data)} registros para
exportar...")
    df = pd.DataFrame(excel_data)
    # Agrupar por hora
    df['Hora'] = (df['Frame'] / fps / 3600).astype(int)
    df['Reloj'] = df['Hora'].apply(lambda h: f"{int(h):02d}:00")
    resumen_completo = []
    for hora in sorted(df['Hora'].unique()):
        datos_hora = df[df['Hora'] == hora]
        reloj = f"{int(hora):02d}:00"
        fila = {'Hora': reloj}
        total_hora = 0
        for sentido in ['A->B', 'B->A']:
            subtotal_sentido = 0
            for clase in target_classes:
                total = len(datos_hora[(datos_hora['Clase'] == clase) &
(datos_hora['Sentido'] == sentido)])
                fila[f"{clase}_{sentido}"] = total
                subtotal_sentido += total
            fila[f"Total_{sentido}"] = subtotal_sentido
            total_hora += subtotal_sentido
        fila['Total_Hora'] = total_hora
        resumen_completo.append(fila)
    df_resumen = pd.DataFrame(resumen_completo)
    totales = {'Hora': 'TOTAL'}
```

```
    for col in df_resumen.columns:
        if col != 'Hora':
            totales[col] = df_resumen[col].sum()
    df_resumen = pd.concat([df_resumen, pd.DataFrame([totales])],
ignore_index=True)
    try:
        print("[INFO] Guardando Excel...")
        with pd.ExcelWriter(excel_path, engine='xlsxwriter') as writer:
            df_resumen.to_excel(writer, sheet_name='Resumen_Por_Hora',
index=False)
            colores = ['#4472C4', '#ED7D31', '#A5A5A5', '#FFC000',
'#70AD47', '#5B9BD5', '#264478', '#9E480E']
            workbook  = writer.book
            worksheet = writer.sheets['Resumen_Por_Hora']
            worksheet.set_column('A:Z', 15)
            worksheet_charts = workbook.add_worksheet('Gráficas')
            chart_barras = workbook.add_chart({'type': 'column',
'subtype': 'stacked'})
            col_idx = 1
            color_idx = 0
            for sentido in ['A->B', 'B->A']:
                for clase in target_classes:
                    col_name = f"{clase}_{sentido}"
                    if col_name in df_resumen.columns:
                        chart_barras.add_series({
                            'name': col_name,
                            'categories': ['Resumen_Por_Hora', 1, 0,
len(df_resumen)-2, 0],
                            'values': ['Resumen_Por_Hora', 1, col_idx,
len(df_resumen)-2, col_idx],
                            'fill':   {'color': colores[color_idx %
len(colores)]},
                            'border': {'color': colores[color_idx %
len(colores)]}
                        })
                        col_idx += 1
                        color_idx += 1
            chart_barras.set_title({'name': 'Conteo Vehicular por Hora y
Sentido'})
            chart_barras.set_x_axis({'name': 'Hora', 'num_font':
{'rotation': -45}})
            chart_barras.set_y_axis({'name': 'Cantidad de Vehículos'})
            chart_barras.set_legend({'position': 'bottom'})
            chart_barras.set_size({'width': 800, 'height': 350})
            worksheet_charts.insert_chart('B2', chart_barras)
            chart_lineas = workbook.add_chart({'type': 'line'})
            color_idx = 0
            for clase in target_classes:
```

```
                col_ab = df_resumen.columns.get_loc(f"{clase}_A->B") if
f"{clase}_A->B" in df_resumen.columns else None
                col_ba = df_resumen.columns.get_loc(f"{clase}_B->A") if
f"{clase}_B->A" in df_resumen.columns else None
                if col_ab is not None and col_ba is not None:
                    valores = [df_resumen.iloc[i, col_ab] +
df_resumen.iloc[i, col_ba] for i in range(len(df_resumen)-1)]
                    col_sum = len(df_resumen.columns) + color_idx
                    worksheet.write_column(1, col_sum, valores)
                    chart_lineas.add_series({
                        'name': f"Total {clase}",
                        'categories': ['Resumen_Por_Hora', 1, 0,
len(df_resumen)-2, 0],
                        'values':     ['Resumen_Por_Hora', 1, col_sum,
len(df_resumen)-2, col_sum],
                        'line': {'color': colores[color_idx %
len(colores)], 'width': 2.5},
                        'marker': {'type': 'circle', 'size': 7, 'border':
{'color': colores[color_idx % len(colores)]}, 'fill': {'color':
colores[color_idx % len(colores)]}}
                    })
                    color_idx += 1
            chart_lineas.set_title({'name': 'Tendencia por Clase (Total
por Hora)'})
            chart_lineas.set_x_axis({'name': 'Hora', 'num_font':
{'rotation': -45}})
            chart_lineas.set_y_axis({'name': 'Total Vehículos'})
            chart_lineas.set_legend({'position': 'bottom'})
            chart_lineas.set_size({'width': 800, 'height': 350})
            worksheet_charts.insert_chart('B22', chart_lineas)
            if 'Total_Hora' in df_resumen.columns:
                chart_total = workbook.add_chart({'type': 'line'})
                col_total = df_resumen.columns.get_loc('Total_Hora')
                chart_total.add_series({
                    'name': 'Total Vehículos',
                    'categories': ['Resumen_Por_Hora', 1, 0,
len(df_resumen)-2, 0],
                    'values':     ['Resumen_Por_Hora', 1, col_total,
len(df_resumen)-2, col_total],
                    'line': {'color': '#264478', 'width': 3},
                    'marker': {'type': 'diamond', 'size': 8, 'border':
{'color': '#264478'}, 'fill': {'color': '#264478'}}
                })
                chart_total.set_title({'name': 'Total Vehículos por
Hora'})
                chart_total.set_x_axis({'name': 'Hora', 'num_font':
{'rotation': -45}})
                chart_total.set_y_axis({'name': 'Total Vehículos'})
                chart_total.set_legend({'none': True})
```

```
                chart_total.set_size({'width': 800, 'height': 350})
                worksheet_charts.insert_chart('B42', chart_total)
        print(f"[INFO] Excel guardado exitosamente con formato en:
{excel_path}")
    except Exception as e:
        print(f"[ERROR] No se pudo guardar Excel con formato: {str(e)}")
        try:
            print("[INFO] Intentando guardar Excel sin formato...")
            df_resumen.to_excel(excel_path, index=False)
            print(f"[INFO] Excel guardado sin formato en: {excel_path}")
        except Exception as e:
            print(f"[ERROR] No se pudo guardar Excel sin formato:
{str(e)}")
            try:
                print("[INFO] Intentando guardar como CSV...")
                df_resumen.to_csv(csv_path, index=False)
                print(f"[INFO] Datos guardados como CSV en: {csv_path}")
            except Exception as e:
                print(f"[ERROR] No se pudo guardar ni siquiera como CSV:
{str(e)}")



# Tamaño mínimo y máximo del bounding box por clase (en píxeles)
min_box_size = {
    "car": 10,
    "bus": 20,
    "motorcycle": 3
}

max_box_size = {
    "bus": 1000,
    "motorcycle": 500
}

# Tolerancia especial para activación de conteo (en píxeles)
activation_tolerance = {
    "default": 12,
    "motorcycle": 40,
    "bus": 18,
    "truck": 16
}

video_path = r"C:\Users\Andres\Documents\Videos de conteo
vehicular\POSGRADOS_Fecha_2026-04-22_Inicio_12-32-46.avi"
model_path = r"C:\Users\Andres\Documents\goge\Deteccion de
vehiculos\yolo11m.pt"

# 🔥 incluir motorbike
```

```
target_classes = ["car", "bus", "truck", "motorcycle", "motorbike"]

class_colors = {
    "car": (0, 255, 0),
    "bus": (0, 0, 255),
    "truck": (255, 0, 0),
    "motorcycle": (255, 255, 0),
    "motorbike": (255, 255, 0)
}

confidence_threshold = 0.12

visual_thresholds = {
    "default": 0.10,
    "motorcycle": 0.015,
    "motorbike": 0.015,
    "bus": 0.10,
    "car": 0.10,
    "truck": 0.50
}

min_confidence_global = 0.015
```

```
import numpy as np

class VehicleTracker:
    def __init__(self, zona_A, zona_B, linea, target_classes):
        self.zona_A = zona_A
        self.zona_B = zona_B
        self.linea = linea
        self.target_classes = target_classes

        self.counters = {('A->B', cls): 0 for cls in target_classes}
        self.counters.update({('B->A', cls): 0 for cls in
target_classes})

        self.vehicles = {}
        self.frame = 0

    def update(self, detecciones):
        drawables = []
        self.frame += 1

        (x1, y1), (x2, y2) = self.linea
```

```
def lado(p):
    return (x2 - x1)*(p[1] - y1) - (y2 - y1)*(p[0] - x1)

for det in detecciones:
    x1b, y1b, x2b, y2b = det['bbox']
    cx, cy = det['centro']
    cls_nm = det['clase']
    tid = det['id']

    if tid not in self.vehicles:
        self.vehicles[tid] = {
            "last": (cx, cy),
            "frames": 0,
            "contado": False,
            "number": None,
            "sentido": None,
            "frames_post": 0  # 🔥 control visual
        }

    v = self.vehicles[tid]
    anterior = v["last"]
    actual = (cx, cy)

    v["last"] = actual
    v["frames"] += 1

    # ignorar ruido inicial
    if v["frames"] < 3:
        continue

    lado_anterior = lado(anterior)
    lado_actual = lado(actual)

    # 🔥 DETECCIÓN ROBUSTA DE CRUCE
    if not v["contado"] and (lado_anterior * lado_actual < 0):

        if lado_actual > 0:
            sentido = "A->B"
        else:
            sentido = "B->A"

        v["contado"] = True
        v["sentido"] = sentido

        self.counters[(sentido, cls_nm)] += 1
        v["number"] = self.counters[(sentido, cls_nm)]

    # 🔥 VISUALIZACIÓN CONTROLADA (NO SATURA)
    if v["contado"]:
```

```
                v["frames_post"] += 1

                # SOLO mostrar después de cruzar y por tiempo limitado
                if 5 < v["frames_post"] < 35:
                    drawables.append(
                        (x1b, y1b, x2b, y2b, cx, cy, cls_nm, v["number"],
"", "", "", "", v["sentido"])
                    )

        return drawables

    def resumen(self):
        return self.counters




import cv2
import numpy as np
import json

# 1) Cargar primer fotograma
video_path = r"C:\Users\Andres\Documents\Videos de conteo
vehicular\POSGRADOS.avi"
cap = cv2.VideoCapture(video_path)
ret, frame = cap.read()
cap.release()
if not ret:
    raise RuntimeError("No se pudo leer el primer fotograma.")
h, w = frame.shape[:2]

# 2) Dibujar ROI
polygon_points = []
def draw_polygon(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        polygon_points.append((x, y))
    elif event == cv2.EVENT_RBUTTONDOWN and polygon_points:
        polygon_points.pop()

cv2.namedWindow("1) ROI (izq=agregar, der=deshacer, c=confirmar)")
cv2.setMouseCallback("1) ROI (izq=agregar, der=deshacer, c=confirmar)",
draw_polygon)
while True:
    vis = frame.copy()
    for pt in polygon_points:
        cv2.circle(vis, pt, 4, (0, 0, 255), -1)
    if len(polygon_points) > 1:
```

```
        cv2.polylines(vis, [np.array(polygon_points)], False, (0, 255,
0), 2)
    cv2.putText(vis, f"Puntos: {len(polygon_points)}", (10,25),
                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255,255,255), 2)
    cv2.imshow("1) ROI (izq=agregar, der=deshacer, c=confirmar)", vis)
    if cv2.waitKey(1) & 0xFF == ord("c") and len(polygon_points) >= 3:
        break
cv2.destroyAllWindows()

# 3) Líneas de entrada/salida
linea_entrada = [(int(w*0.2), int(h*0.3)), (int(w*0.8), int(h*0.35))]
linea_salida  = [(int(w*0.2), int(h*0.7)), (int(w*0.8), int(h*0.75))]
moving, idx, mode = False, None, None

def move_line(event, x, y, flags, param):
    global moving, idx, mode
    tol = 10
    pts = linea_entrada + linea_salida
    if event == cv2.EVENT_LBUTTONDOWN:
        for i, p in enumerate(pts):
            if abs(p[0]-x)<tol and abs(p[1]-y)<tol:
                moving = True
                idx  = i % 2
                mode = "in" if i < 2 else "out"
    elif event == cv2.EVENT_MOUSEMOVE and moving:
        if mode == "in":  linea_entrada[idx] = (x, y)
        else:             linea_salida[idx]  = (x, y)
    elif event == cv2.EVENT_LBUTTONUP:
        moving = False

cv2.namedWindow("2) Líneas (arrastra, c=confirmar)")
cv2.setMouseCallback("2) Líneas (arrastra, c=confirmar)", move_line)
while True:
    vis = frame.copy()
    cv2.line(vis, *linea_entrada, (0,255,255), 2)
    cv2.line(vis, *linea_salida,  (255,0,255), 2)
    for p in linea_entrada + linea_salida:
        cv2.circle(vis, p, 6, (255,255,255), -1)
    cv2.putText(vis, "'c'=confirmar", (10,25),
                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255,255,255), 2)
    cv2.imshow("2) Líneas (arrastra, c=confirmar)", vis)
    if cv2.waitKey(1) & 0xFF == ord("c"):
        break
cv2.destroyAllWindows()

# 4) Máscara y zona de control
mask = np.zeros((h, w), dtype=np.uint8)
cv2.fillPoly(mask, [np.array(polygon_points, np.int32)], 255)
zona_control = np.array([
```

```
    linea_entrada[0],
    linea_entrada[1],
    linea_salida[1],
    linea_salida[0]
], dtype=np.int32)

# 5) Guardar zonas
np.savez("config_zonas.npz",
         mask=mask,
         polygon_points=np.array(polygon_points, np.int32),
         linea_entrada=np.array(linea_entrada, np.int32),
         linea_salida=np.array(linea_salida, np.int32),
         zona_control=zona_control)

# 6) Guardar configuración general
settings = {
    "video_path":      video_path,
    "model_path":      "yolo11n.pt",
    "target_classes":  ["car", "bus", "truck", "motorcycle"],
    "class_colors": {
        "car": [0,255,0], "bus": [0,0,255],
        "truck": [255,0,0], "motorcycle": [255,255,0]
    },
    "conf_threshold":        0.5,
    "vis_threshold":         0.6,
    "vis_thresholds_per_class": {
        "default": 0.6,
        "motorcycle": 0.25
    }
}
with open("config.json", "w") as f:
    json.dump(settings, f, indent=2)

print("☑ Configuración guardada: config_zonas.npz y config.json")
```

Anexo 2. Programación y configuración del nodo ESP32-CAM

```
#include <Ethernet.h>
#include <SPI.h>
#include <EthernetUdp.h>
#include "esp_camera.h"
#include "esp_timer.h"
#include "soc/soc.h"
```

```
#include "soc/rtc_cntl_reg.h"

// ================ CONFIGURACIÓN RED Y TIEMPO ================
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = -18000; // Ecuador UTC-5
EthernetUDP Udp;
byte mac[] = { 0x02, 0xAB, 0xCD, 0xEF, 0x00, 0x01 };
EthernetServer server(80);

// ================ PINES CÁMARA (AI-THINKER) ================
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

const int camPins[] = {0, 26, 27, 35, 34, 39, 36, 21, 19, 18, 5, 25, 23,
22};

// ================ VARIABLES DE CONTROL ================
#define LED 33
unsigned long lastFrameTime = 0;
unsigned long epochAlArranque = 0;
unsigned long millisAlArranque = 0;

// ================ LOGGING PROFESIONAL ================
void logHeader() {
  Serial.println("\n**************************************************
******");
  Serial.println("* UNIVERSIDAD CATÓLICA DE CUENCA - MONITOREO VIAL
AI     *");
  Serial.println("* FIRMWARE DE PRODUCCIÓN (RED ROBUSTA + VIDEO
ESTABLE)   *");
  Serial.println("**************************************************
****");
}

void logMsg(String level, String msg) {
```

```
  Serial.print("[" + level + "] [" + String(millis()) + "ms] ");
  Serial.println(msg);
}

// ================ RESPUESTAS HTTP ================
void responderPing(EthernetClient* client) {
  client->println("HTTP/1.1 200 OK\r\nContent-Type:
text/plain\r\nConnection: close\r\n\r\nOK");
}

void responderId(EthernetClient* client) {
  client->println("HTTP/1.1 200 OK\r\nContent-Type:
text/plain\r\nConnection: close\r\n\r\nESP32_Vial_Produccion");
}

void responder404(EthernetClient* client) {
  client->println("HTTP/1.1 404 Not Found\r\nContent-Type:
text/plain\r\nConnection: close\r\n\r\nNot Found");
}

// ================ PROTOCOLO DE RED CORREGIDO ================
bool inicializarRed() {
  logMsg("INFO", "Despertando modulo W5500...");

  // Pequeña pausa para que el bus SPI se estabilice
  delay(500);

  // 1. Intentamos obtener IP (Esto hace el soft-reset al W5500)
  if (Ethernet.begin(mac) == 0) {
    logMsg("WARN", "El router no asigno IP por DHCP. Diagnosticando...");

    // 2. Ahora que intentó encender, verificamos si físicamente existe
    if (Ethernet.hardwareStatus() == EthernetNoHardware) {
      logMsg("ERROR", "Modulo W5500 NO detectado. Revise cables SPI.");
      return false;
    }

    // 3. Verificamos el cable
    if (Ethernet.linkStatus() == LinkOFF) {
      logMsg("ERROR", "Cable Ethernet NO conectado al router.");
      return false;
    }

    // 4. Si el hardware está bien pero el DHCP falló, forzamos IP
estática
    logMsg("WARN", "Forzando IP estatica de respaldo
(192.168.1.177)...");
    Ethernet.begin(mac, IPAddress(192, 168, 1, 177));
  }
```

```
  // 5. Validamos que la IP no sea basura (0.0.0.0)
  IPAddress ip = Ethernet.localIP();
  if (ip[0] == 0) {
    logMsg("ERROR", "IP Invalida detectada (0.0.0.0).");
    return false;
  }

  logMsg("SYSTEM", "Conexion Exitosa. IP: " + ip.toString());
  return true;
}

// ================ TIEMPO, INTERNET Y SUEÑO ================
unsigned long sincronizarNTP() {
  logMsg("INFO", "Comprobando salida a Internet (Servidor NTP)...");
  Udp.begin(8888);
  byte packetBuffer[48];
  memset(packetBuffer, 0, 48);
  packetBuffer[0] = 0b11100011;
  Udp.beginPacket(ntpServer, 123);
  Udp.write(packetBuffer, 48);
  Udp.endPacket();

  delay(1500);
  if (Udp.parsePacket()) {
    Udp.read(packetBuffer, 48);
    unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
    unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
    epochAlArranque = (highWord << 16 | lowWord) - 2208988800UL +
gmtOffset_sec;
    millisAlArranque = millis();
    logMsg("SYSTEM", "Internet OK. Reloj sincronizado.");
    return epochAlArranque;
  }
  logMsg("ERROR", "Sin acceso a Internet (NTP falló).");
  return 0;
}

unsigned long obtenerHoraActual() {
  if (epochAlArranque == 0) return 0;
  return epochAlArranque + ((millis() - millisAlArranque) / 1000);
}

void irADormir(unsigned long segundos) {
  logMsg("SYSTEM", "Horario nocturno. Hibernando por " + String(segundos)
+ "s.");
  esp_camera_deinit();
  delay(100);
  for(int i=0; i<14; i++) {
```

```
    pinMode(camPins[i], OUTPUT);
    digitalWrite(camPins[i], LOW);
  }
  pinMode(PWDN_GPIO_NUM, OUTPUT);
  digitalWrite(PWDN_GPIO_NUM, HIGH);
  Serial.flush();
  esp_sleep_enable_timer_wakeup((uint64_t)segundos * 1000000ULL);
  esp_deep_sleep_start();
}

// ================ STREAMING ULTRA-ESTABLE ================
void stream_h(EthernetClient* ec) {
  logMsg("INFO", ">>> CLIENTE CONECTADO. INICIANDO STREAM <<<");
  camera_fb_t* fb = NULL;

  ec->println("HTTP/1.1 200 OK");
  ec->println("Content-Type: multipart/x-mixed-replace;boundary=frame");
  ec->println("Access-Control-Allow-Origin: *");
  ec->println("Connection: close\r\n");

  while (ec->connected()) {
    unsigned long currentMillis = millis();
    if (currentMillis - lastFrameTime < 80) {
        delay(1); continue;
    }
    lastFrameTime = currentMillis;

    fb = esp_camera_fb_get();
    if (!fb) {
      logMsg("ERROR", "Fallo captura de sensor.");
      break;
    }

    if (fb->len < 1500) {
      esp_camera_fb_return(fb); continue;
    }

    ec->printf("--frame\r\nContent-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n", fb->len);

    uint8_t* buf = fb->buf;
    size_t size = fb->len;
    while (size > 0 && ec->connected()) {
      size_t chunk = size > 1024 ? 1024 : size;
      if (ec->write(buf, chunk) != chunk) break;
      buf += chunk;
      size -= chunk;
      delayMicroseconds(150);
    }
```

```
    ec->print("\r\n");
    ec->flush();
    esp_camera_fb_return(fb);

    if (ESP.getFreeHeap() < 25000) {
      logMsg("WARN", "Memoria RAM critica. Forzando limpieza.");
      break;
    }
  }
  logMsg("INFO", ">>> CLIENTE DESCONECTADO. STREAM CERRADO <<<");
}

// ================ SETUP ================
void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  pinMode(LED, OUTPUT);
  Serial.begin(115200);
  delay(1000);
  logHeader();

  // Iniciar SPI
  SPI.begin(14, 12, 13, 2);
  SPI.setFrequency(16000000);
  Ethernet.init(2);

  // Bucle de conexion corregido
  while (!inicializarRed()) {
    logMsg("WARN", "Reintentando conexion en 5 segundos...");
    delay(5000);
  }

  // Sincronización y Horarios
  unsigned long ahora = sincronizarNTP();
  if (ahora > 0) {
    unsigned long segHoy = ahora % 86400L;
    const unsigned long INICIO = 21000; // 05:50 AM
    const unsigned long FIN = 64800;    // 18:00 PM
    if (segHoy < INICIO || segHoy >= FIN) {
      unsigned long dormirSegs = (segHoy >= FIN) ? (86400 - segHoy +
INICIO) : (INICIO - segHoy);
      irADormir(dormirSegs);
    }
  }

  // Cámara OV3660 - Configuración Anti-Corrupción
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0; config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM; config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM; config.pin_d3 = Y5_GPIO_NUM;
```

```
  config.pin_d4 = Y6_GPIO_NUM; config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM; config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM; config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM; config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM; config.pin_sscb_scl =
SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM; config.pin_reset = RESET_GPIO_NUM;

  config.xclk_freq_hz = 8000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 20;
  config.fb_count = 2;

  if (esp_camera_init(&config) != ESP_OK) {
    logMsg("ERROR", "Fallo inicializacion de sensor optico.");
ESP.restart();
  }

  // Calibración Anti-Saturación
  sensor_t * s = esp_camera_sensor_get();
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); s->set_hmirror(s, 1);
    s->set_brightness(s, -2); s->set_contrast(s, 2);
    s->set_saturation(s, 1);
    s->set_whitebal(s, 1); s->set_wb_mode(s, 0);
    s->set_exposure_ctrl(s, 1); s->set_ae_level(s, -2);
  }

  server.begin();
  logMsg("SYSTEM", ">>> SISTEMA DE MONITOREO VIAL 100% OPERATIVO <<<");
}

// ================ LOOP (LÓGICA HTTP ROBUSTA) ================
void loop() {
  // 1. Vigilante de Red Constante
  if (Ethernet.linkStatus() == LinkOFF) {
    logMsg("ERROR", "Cable de red desconectado. Esperando
reconexion...");
    while (Ethernet.linkStatus() == LinkOFF) { delay(1000); }
    logMsg("SYSTEM", "Cable reconectado. Reiniciando nodo para limpiar
buffers...");
    ESP.restart();
  }

  // 2. Procesamiento de Clientes Web
  EthernetClient client = server.available();
  if (client) {
    String currentLine = "";
```

```
    String request = "";
    bool requestReceived = false;

    while (client.connected() && !requestReceived) {
      if (client.available()) {
        char c = client.read();
        if (c == '\n') {
          if (currentLine.length() == 0) break;
          if (currentLine.indexOf("GET ") != -1) request = currentLine;
          currentLine = "";
        } else if (c != '\r') {
          currentLine += c;
        }
      }
    }

    if (request.length() > 0) {
      if (request.indexOf("GET /stream") >= 0) {
        stream_h(&client);
      } else if (request.indexOf("GET /ping") >= 0) {
        responderPing(&client);
      } else if (request.indexOf("GET /id") >= 0) {
        responderId(&client);
      } else {
        responder404(&client);
      }
    }
    delay(1);
    client.stop();
  }

  // 3. Vigilante de Horario Nocturno
  static unsigned long lastCheck = 0;
  if (millis() - lastCheck > 60000) {
    lastCheck = millis();
    unsigned long ahora = obtenerHoraActual();
    if (ahora > 0) {
      unsigned long segHoy = ahora % 86400L;
      if (segHoy >= 64800 || segHoy < 21000) {
        unsigned long dormir = (segHoy >= 64800) ? (86400 - segHoy +
21000) : (21000 - segHoy);
        irADormir(dormir);
      }
    }
  }
}
```

Anexo 3. Ruteado de PCB

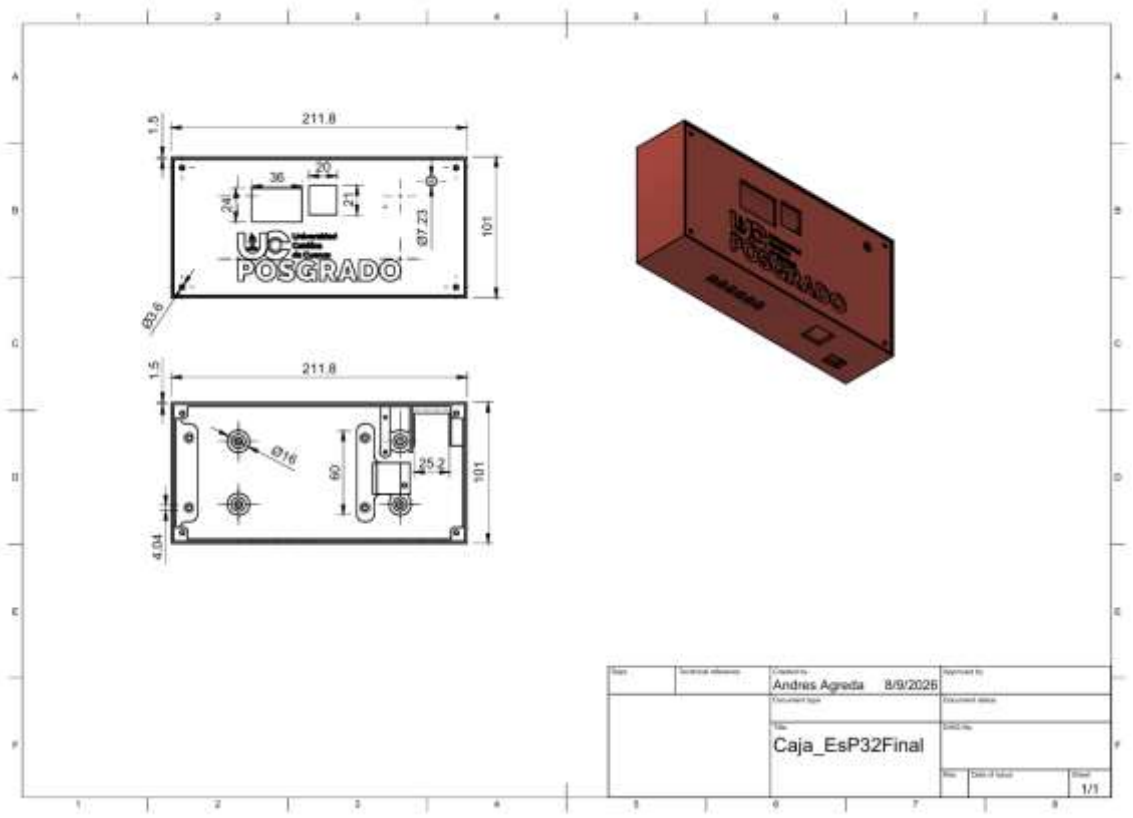


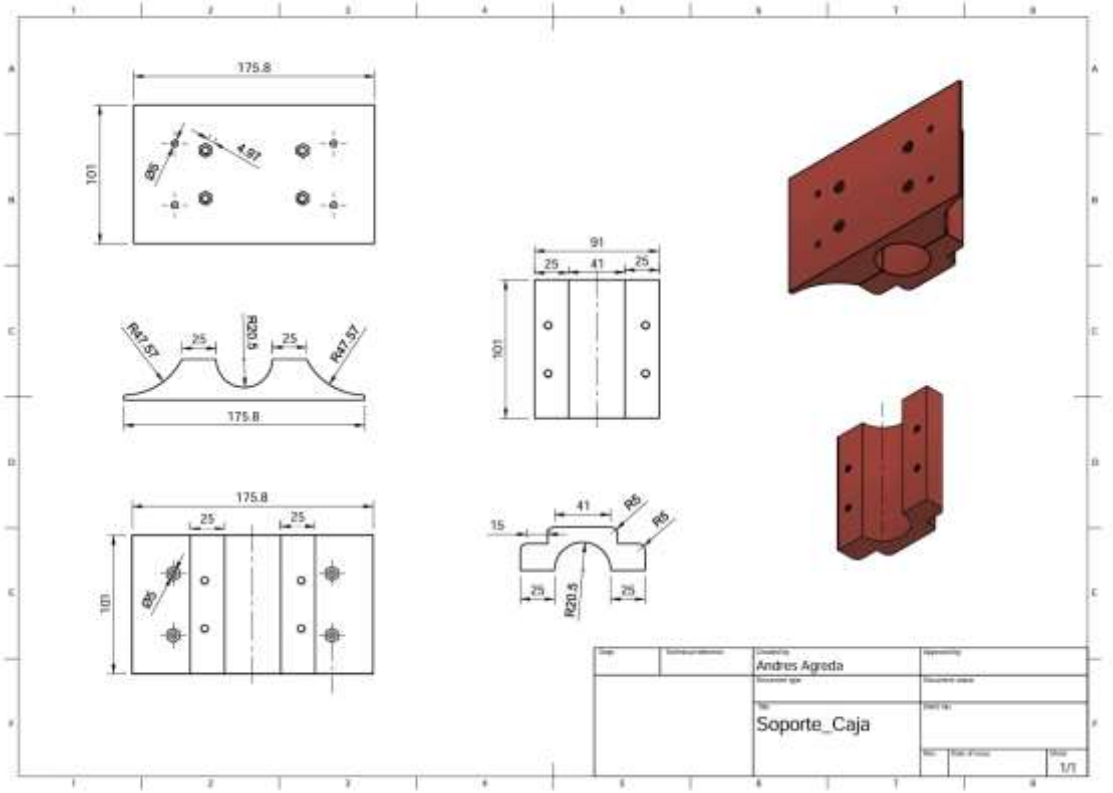
Anexo 4. Instalación de equipos de monitoreo

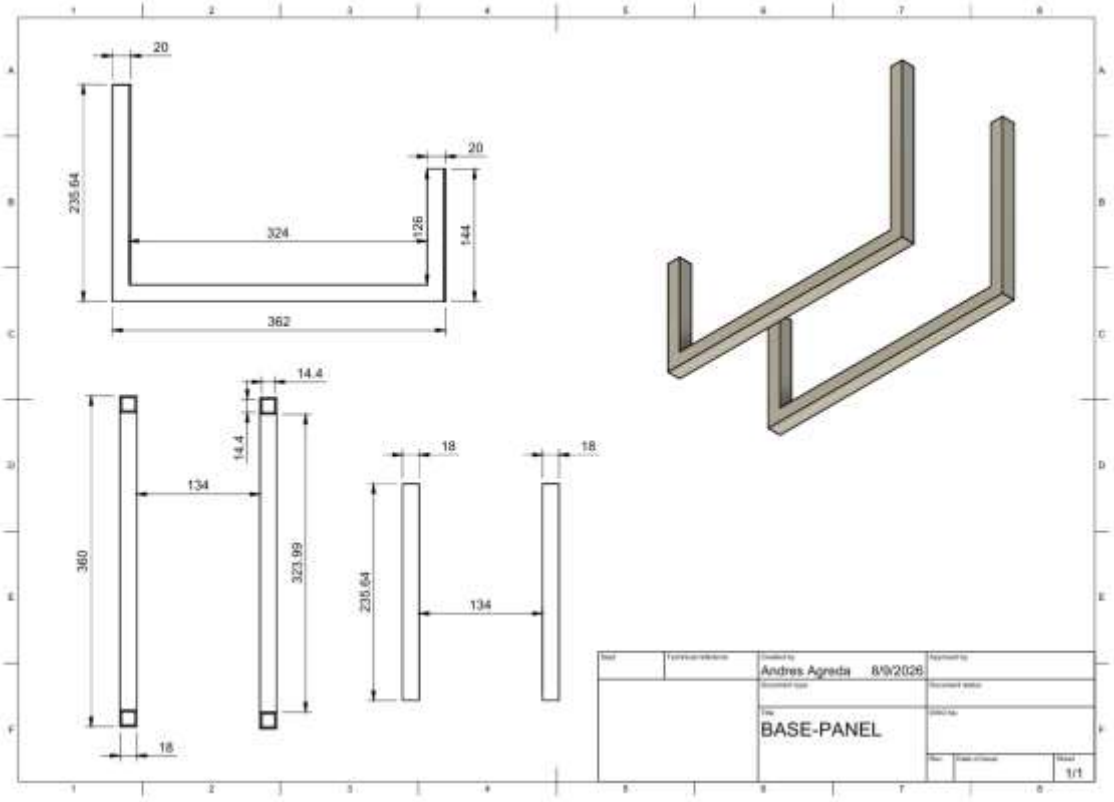


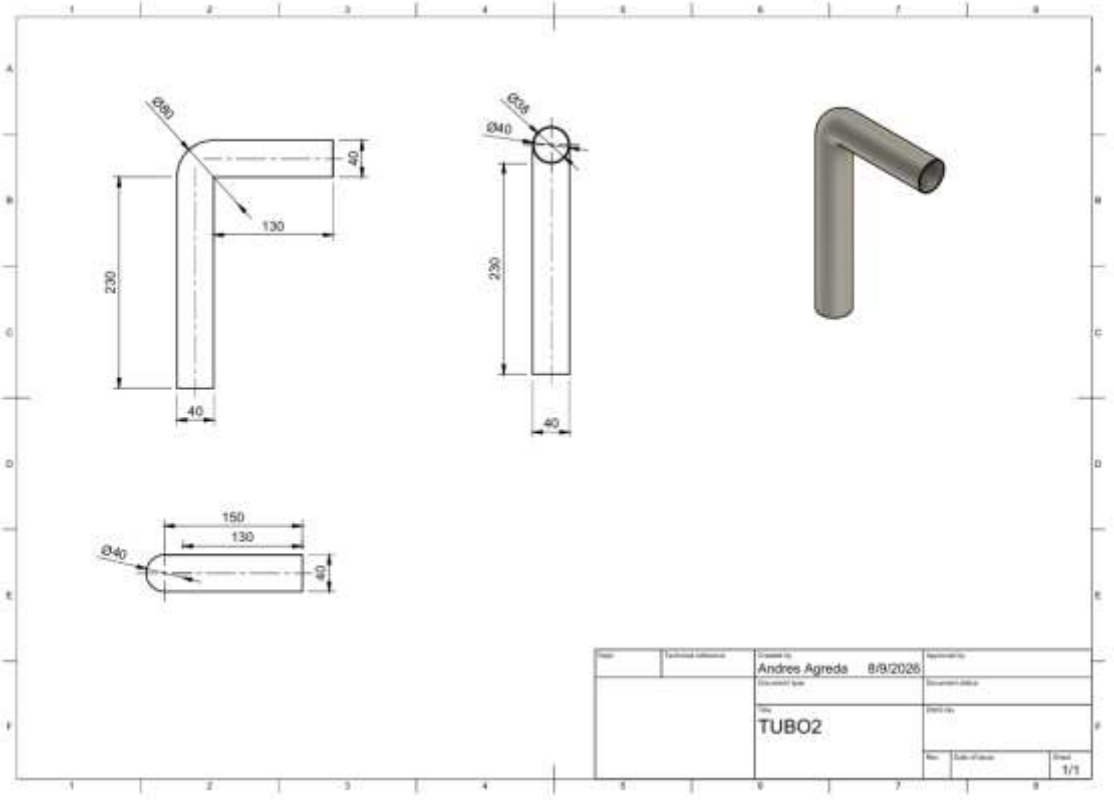


Anexo 5. Diseño y planos de la carcasa y soporte



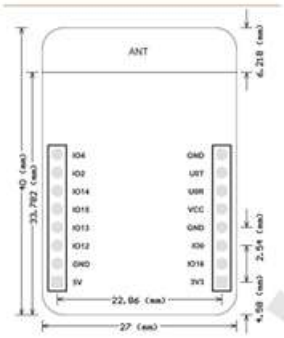



20
235.64
20
324
126
144
362
14.4
14.4
134
360
323.99
18
18
235.64
134
18
Andres Agreda 8/9/2026
BASE-PANEL
1/1


Ø90
40
130
230
40
Ø58
Ø40
230
40
150
130
Ø40
40
Andres Agreda 8/9/2026
TUBO2
1/1

Anexo 6. Hoja de especificaciones técnicas del ESP32-CAM

ESP32-CAM Module





Features

- The smallest 802.11b/g/n Wi-Fi BT SoC Module
- Low power 32-bit CPU,can also serve the application processor
- Up to 160MHz clock speed，Summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras,Built-in Flash lamp.
- Support image WiFI upload
- Support TF card
- Supports multiple sleep modes.
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)

Overview

The ESP32-CAM has a very competitive small-size camera module that can operate independently as a minimum system with a footprint of only 27*40.5*4.5mm and a deep sleep current of up to 6mA.

ESP-32CAM can be widely used in various IoT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals and other IoT applications. It is an ideal solution for IoT applications.

ESP-32CAM adopts DIP package and can be directly inserted into the backplane to realize rapid production of products, providing customers with high-reliability connection mode, which is convenient for application in various IoT hardware terminals.

Copyright © 2017 Shenzhen Ai-Thinker Technology Co., Ltd All Rights Reserved

Product Specifications

Module Model	ESP32-CAM
Package	DIP-16
Size	27*40.5*4.5（±0.2）mm
SPI Flash	Default 32Mbit
RAM	520KB SRAM +4M PSRAM
Bluetooth	Bluetooth 4.2 BR/EDR and BLE standards
Wi-Fi	802.11 b/g/n/
Support interface	UART、SPI、I2C、PWM
Support TF card	Maximum support 4G
IO port	9
UART Baudrate	Default 115200 bps
Image Output Format	JPEG(OV2640 support only),BMP,GRAYSCALE
Spectrum Range	2412 ~2484MHz
Antenna	Onboard PCB antenna, gain 2dBi
Transmit Power	802.11b: 17±2 dBm (@11Mbps) 802.11g: 14±2 dBm (@54Mbps) 802.11n: 13±2 dBm (@MCS7)
Receiving Sensitivity	CCK, 1 Mbps : -90dBm CCK, 11 Mbps: -85dBm 6 Mbps (1/2 BPSK): -88dBm 54 Mbps (3/4 64-QAM): -70dBm MCS7 (65 Mbps, 72.2 Mbps): -67dBm
Power Dissipation	Turn off the flash lamp:180mA@5V Turn on the flash lamp and turn on the brightness to the maximum:310mA@5V Deep-sleep: Minimum power consumption can be achieved 6mA@5V Moderm-sleep: Minimum up to 20mA@5V Light-sleep: Minimum up to 6.7mA@5V
Security	WPA/WPA2/WPA2-Enterprise/WPS
Power Supply Range	5V
Operating Temperature	-20 ℃ ~ 85 ℃
Storage Environment	-40 ℃ ~ 90 ℃ , < 90%RH

Copyright © 2017 Shenzhen Ai-Thinker Technology Co., Ltd All Rights Reserved

Anexo 7. Hoja de especificaciones técnicas del W5500

WIN-MD-W5500-V1.0

Technical Datasheet W5500-Based Ethernet Module

The Ethernet module communicates with the host controller via a high-speed SPI interface while internally handling the full TCP/IP stack using the W5500 chipset. It supports 10/100 Mbps Ethernet connectivity, enabling reliable and efficient data transmission over standard LAN networks.

W5500 Module: -



The module is designed in a compact PCB form factor and features an onboard RJ45 connector with integrated magnetics and status indication LEDs. It offers a simple and robust SPI interface for seamless integration with a wide range of microcontrollers, including Arduino, ESP32, STM32, and industrial control systems.

Configuration and communication management are simplified through the hardware TCP/IP stack, significantly reducing firmware complexity and improving overall system performance.

WIN-MD-W5500-V1.0

Specifications

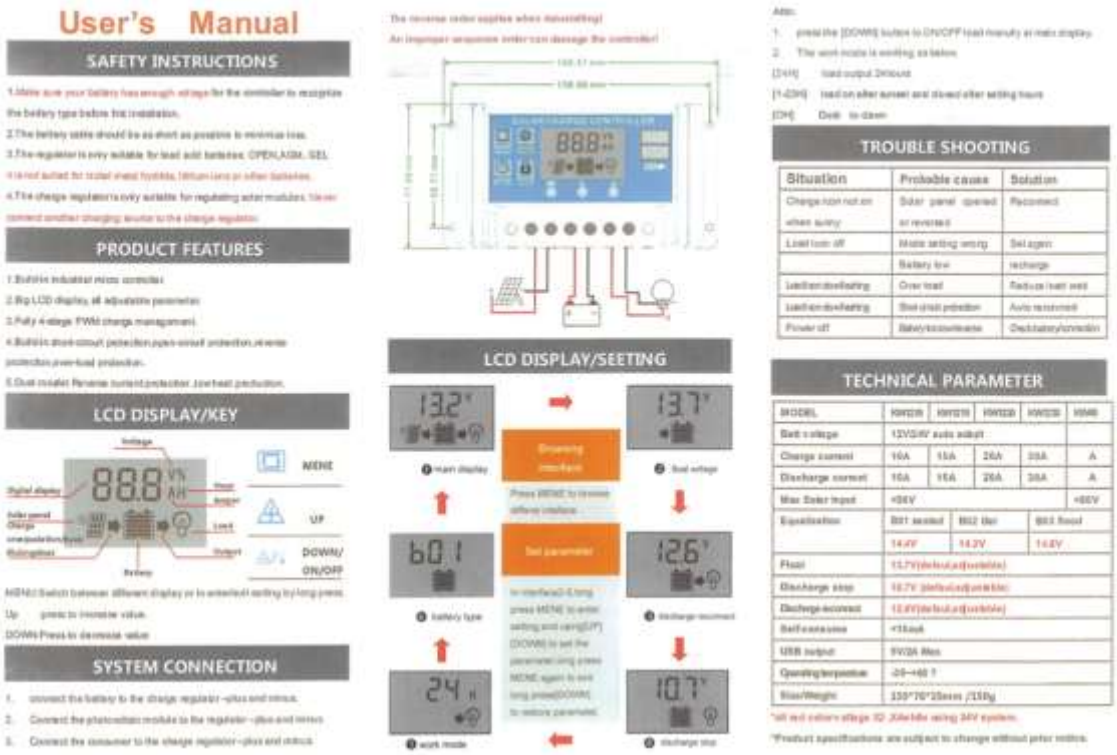
General –

Interface	SPI (Serial Peripheral Interface)
Ethernet	10/100 Mbps, RJ45 with Integrated Magnetics
I/O Connector	2 × 5 Pin Header, 2.54 mm Pitch
Dimensions	28.3 mm (L) × 55.4 mm (W)
Power Supply	5V DC / 3.3V DC Input
Certifications	RoHS, CE APPROVED

Additional Features: -

Integrated hardware TCP/IP stack (W5500)
Supports up to 8 simultaneous socket connections
Onboard RJ45 with integrated magnetics for improved signal integrity
Low firmware overhead due to hardware networking offload
Reliable operation with optimized PCB layout and power decoupling

Anexo 8. Hoja de especificaciones técnicas del controlador PWM



User's Manual

SAFETY INSTRUCTIONS

PRODUCT FEATURES

LCD DISPLAY/KEY

MENU

UP

DOWN/ ON/OFF

SYSTEM CONNECTION

LCD DISPLAY/SEETING

TROUBLE SHOOTING

Situation	Probable cause	Solution
Charge icon not on when sunny	Solar panel opened or reversed	Reconnect
Load icon off	Mode setting wrong	Set again
	Battery low	recharge
Led/lcm/overheating	Over load	Reduce load wat
Led/lcm/overheating	Short circuit protection	Auto reconnect
Power off	Battery/circuit/device	Check battery/circuit/d

TECHNICAL PARAMETER

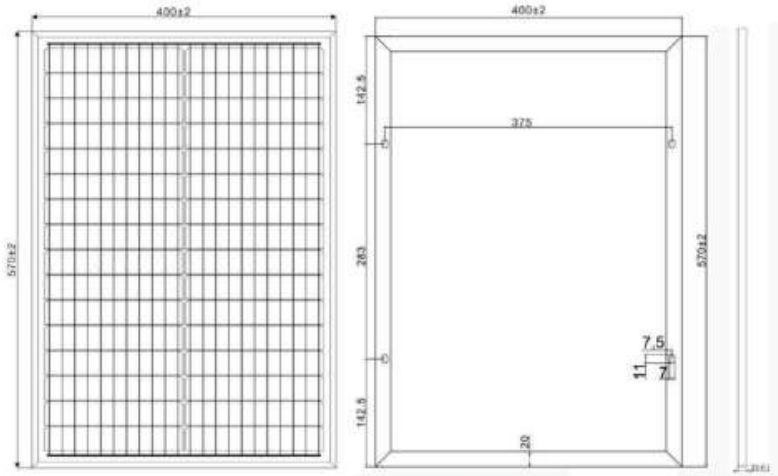
MODEL					
Batt voltage	12V/24V auto adapt				
Charge current	10A	15A	20A	30A	A
Discharge current	10A	15A	20A	30A	A
Max Solar Input	<50V				<80V
Equalization	B01 sealed	B02 Gel	B03 flood		
	14.4V	14.2V	14.6V		
Float	13.7V(default,adjustable)				
Discharge stop	10.7V (default,adjustable)				
Discharge reconnect	12.6V(default,adjustable)				
Self-consume	<15mA				
USB output	5V/2A Max				
Operating/temperature	-20~+60 ℃				
Size/Weight	230*70*25mm /150g				

*All red colors allege 3D ,Available using 24V system.

*Product specifications are subject to change without prior notice.

Anexo 9. Hoja de especificaciones técnicas del panel fotovoltaico

POWEST®



RENDIMIENTO ELÉCTRICO

Parámetros eléctricos en Condiciones de prueba estándar (STC)			
Tipo de módulo			Monocristalino
Potencia de salida	P_{max}	W	40
Tolerancia de salida de potencia	ΔP_{max}	W	±3%
Eficiencia del módulo	η_m	%	17.50
Voltaje en Pmax	V_{mpp}	V	18.24
Corriente en Pmax	I_{mpp}	A	2.19
Voltaje de circuito abierto	V_{oc}	V	21.80
Corriente de cortocircuito	I_{sc}	A	2.39

CONDICIONES DE OPERACIÓN

Max. voltaje del sistema	1000Vdc
Max. clasificación de fusibles en serie	10A
Rango de temperatura de funcionamiento	-40ºC a 85ºC
Máx. carga estática, frontal (p. ej., nieve)	5400Pa
Max. carga estática, hacia atrás (p. ej., viento)	2400Pa
Max. impacto de granizo (diámetro / velocidad)	25mm / 23m/s

* No conecte fusibles en la caja combinadora con dos o más cadenas en conexión paralela

CARACTERÍSTICAS TÉRMICAS

Temperatura nominal de la celda de operación	NOCT	ºC	45±2
Coeficiente de temperatura de Pmax	γ	%/ºC	-0.35
Coeficiente de Variación de Temperatura de Voc	βV_{DC}	%/ºC	-0.27
Coeficiente de temperatura de Isc	α Isc	%/ºC	0.05

MATERIALES DE CONSTRUCCIÓN

Cubierta frontal (material / espesor)	vidrio templado con bajo contenido de hierro / 3.2 mm
Celda (cantidad / material)	32 / silicio monocristalino
Marco (material)	Aleación de aluminio anodizado
Caja de conexiones (grado de protección)	IP67
Cable (longitud / área de sección transversal)	500mm/2.5mm²

CARACTERÍSTICAS GENERALES

Dimensiones (Largo / Ancho / Alto)	570mm×400mm×25mm
Peso	2.80kg

Anexo 10. Hoja de especificaciones técnicas de la batería FL1290GS



BATERIA SELLADA 12V- 9 AH REF.FL1290GS

FuliBattery

Especificaciones

MH48936

Voltaje Nominal	12V
Capacidad Nominal (20 Hr)	9.0 Ah
Dimensiones	Longitud 151±2mm (5.95 ") Ancho 65±1mm (2.56 ") Altura 94±1mm (3.70 ") Altura Total 102±2mm (4.02 ")
Peso Aproximado	Aprox. 2.7 kg (5.87 lbs)
Terminal	T1/T2-F1/F5
Material del Envase	ABS UL 94-HB (retardante de llama UL94V-0)
Rango de Operación	8.60 A(I/0.43A (20hr, 1.80V/Celda,25° C/77°F) 7.56 A(I/1.53A (5hr, 1.75V/Celda,25° C/77°F) 5.70 A(I/9.00A (1C, 1.60V/Celda,25° C/77°F) 3.60 A(I/27.0A (3C, 1.60V/Celda,25° C/77°F) 36W/Celda (15Min rate, 1.60V/Celda,25° C/77°F)
Máxima Corriente de Descarga	135A(5s)
Resistencia Interna	Carga Completa 100%. Aprox. 22mΩ
Rango de Temperatura de Operación	Descarga -15~50°C (5~122°F) Carga -15~40°C (5~104°F) Almacenamiento -15~40°C (5~104°F)
Rango Nominal de Temp. de Operación	25±3°C (77±5°F)
Ciclo de Uso	Corriente de carga inicial de menos de 2.7O. Voltage. 14.4V ~ 15.0V a 25°C (77° F) de temperatura. Coeficiente -30mV/ °C
Modo Espera	No hay límite en la carga inicial de tensión actual. 13.5V ~ 13.8V a 25°C (77° F) Temp. Coeficiente -20mV /°OC
Capacidad Afectada por la temperatura	40°C (104°F) 103% 25°C (77°F) 100% 0°C (32°F) 86%
Auto- descarga	Las Baterías de la serie Fulibattery GS se pueden almacenar durante un máximo de 6 meses a 25°C (77°F) y luego se requiere una carga de reposición.Para temperaturas más altas del intervalo de tiempo será más corto.

Aplicaciones

- De uso múltiple.
- Fuente de alimentación. Ininterrumpida (UPS).
- Sistema de potencia Eléctrica (EPS).
- Alimentación de reserva de emergencia.
- Luz de emergencia.
- Señal ferroviaria.
- Señal de aeronaves.
- Alarma y sistema de seguridad.
- Aparatos y equipos electrónicos.
- Fuente de alimentación comunicación.
- Fuente de alimentación DC.
- Sistema de control automático.

Corriente Constante de Descarga (Amperios) a 25 °C (77 °F)

V/Tiempo	5min	10min	15min	20min	30min	45min	1h	2h	3h	4h	5h	6h	8h	10h	20h
1.85V/Celda	28.6	19.1	15.0	12.4	9.24	6.71	5.24	2.84	2.03	1.61	1.34	1.16	0.927	0.776	0.421
1.80V/Celda	32.1	20.9	16.2	13.2	9.65	6.95	5.43	2.95	2.09	1.64	1.42	1.19	0.956	0.798	0.430
1.75V/Celda	35.5	22.1	17.2	13.9	10.0	7.20	5.61	3.03	2.15	1.70	1.48	1.25	0.994	0.819	0.439
1.70V/Celda	37.6	23.3	18.0	14.3	10.4	7.42	5.75	3.12	2.22	1.75	1.53	1.28	1.00	0.834	0.444
1.67V/Celda	39.3	24.1	18.5	14.8	10.7	7.60	5.86	3.17	2.26	1.79	1.57	1.31	1.02	0.843	0.447
1.60V/Celda	40.5	25.0	18.9	15.1	10.9	7.72	5.96	3.22	2.28	1.81	1.58	1.30	1.03	0.850	0.450

Potencia Constante de Descarga (Vatios/celda) a 25 °C (77°F)

V/Tiempo	5min	10min	15min	20min	30min	45min	1h	2h	3h	4h	5h	6h	8h	10h	20h
1.85V/Celda	52.8	35.6	28.2	23.5	17.6	12.9	10.1	5.50	3.95	3.14	2.63	2.28	1.85	1.54	0.836
1.80V/Celda	58.4	38.4	30.1	24.7	18.2	13.2	10.4	5.66	4.06	3.20	2.69	2.34	1.88	1.58	0.851
1.75V/Celda	63.4	40.1	32.0	25.7	18.8	13.6	10.7	5.82	4.15	3.30	2.78	2.41	1.95	1.61	0.867
1.70V/Celda	66.5	41.8	33.6	26.5	19.4	14.0	10.9	5.97	4.27	3.39	2.84	2.46	1.97	1.64	0.877
1.67V/Celda	68.4	42.6	34.5	27.0	19.8	14.2	11.1	6.05	4.33	3.43	2.89	2.49	2.00	1.65	0.881
1.60V/Celda	69.5	43.6	36.0	27.1	20.0	14.3	11.2	6.12	4.37	3.47	2.92	2.52	2.01	1.67	0.885

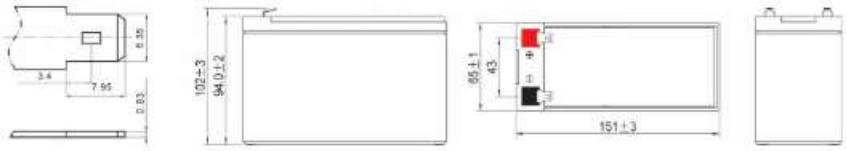
www.fulibattery.co

BATERIA SELLADA 12V- 9 AH REF.FL1290GS

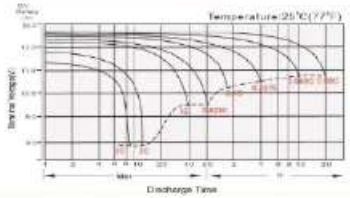
FuliBattery

Dimensiones (mm)

- Terminal



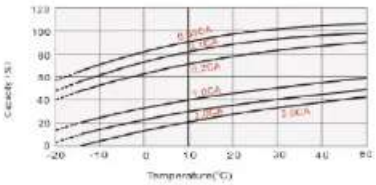
Características de Descarga



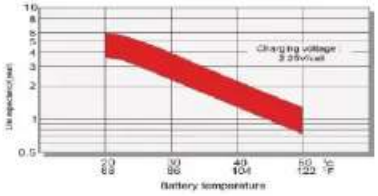
Características de Carga en flotación



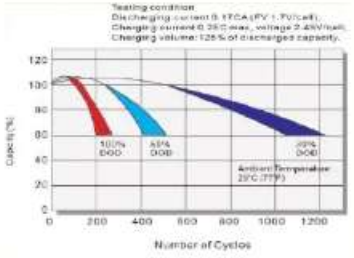
Efectos de la temperatura en relación con la capacidad de la batería



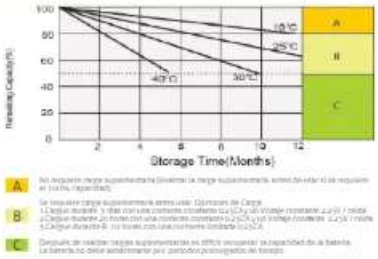
Efecto de la temperatura sobre la vida útil de la batería en flotación



Ciclo de vida en relación a la profundidad de descarga



Características de Auto-descarga



www.fulibattery.co

AUTORIZACION DE PUBLICACION EN EL REPOSITORIO INSTITUCIONAL

Yo, Jose Andres Agreda Ordoñez portador de la cédula de ciudadanía N.º 1900752435. En calidad de autor/a y titular de los derechos patrimoniales del trabajo de titulación "Sistema de supervisión automatizada para el reconocimiento y conteo taxonómico de vehículos mediante algoritmos de visión artificial" de conformidad a lo establecido en el artículo 114 Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación, reconozco a favor de la Universidad Católica de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos, Así mismo; autorizo a la Universidad para que realice la publicación de este trabajo de titulación en el Repositorio Institucional de conformidad a lo dispuesto en el artículo 144 de la Ley Orgánica de Educación Superior.

Cuenca, 20 de octubre de 2026



F: ……..
Jose Andres Agreda Ordoñez
1900752435