



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería Electrónica y Eléctrica

Escuela Profesional de Ingeniería Electrónica

**Sistema de semáforos autónomos con visión artificial
para una intersección vial en Lima: diseño y
prototipado**

TESIS

Para optar el Título Profesional de Ingeniero Electrónico

AUTOR

Rolando Segundo HUAHUALA CUYUBAMBA

ASESOR

Mg. Alex Alfonso GONZALES SUAREZ

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Huahuala, R. (2026). *Sistema de semáforos autónomos con visión artificial para una intersección vial en Lima: diseño y prototipado*. [Tesis de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería Electrónica y Eléctrica, Escuela Profesional de Ingeniería Electrónica]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Rolando Segundo Huahuala Cuyubamba
Tipo de documento de identidad	DNI
Número de documento de identidad	74130063
URL de ORCID	No Aplica
Datos de asesor	
Nombres y apellidos	Alex Alfonso Gonzales Suarez
Tipo de documento de identidad	DNI
Número de documento de identidad	07644682
URL de ORCID	https://orcid.org/0000-0002-4370-4951
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Celso Ysidro Gerónimo Huamán
Tipo de documento	DNI
Número de documento de identidad	09191500
Miembro del jurado 1	
Nombres y apellidos	Carlos Alberto Moreno Paredes
Tipo de documento	DNI
Número de documento de identidad	01292577
Miembro del jurado 2	
Nombres y apellidos	Jean Carlos Malca Fernández
Tipo de documento	DNI
Número de documento de identidad	45535181
Datos de investigación	
Línea de investigación	C.0.5.8. Inteligencia artificial
Grupo de investigación	No Aplica
Agencia de financiamiento	No Aplica

Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Callao Distrito: Bellavista Latitud: -12.06250 Longitud: -77.09709
Año o rango de años en que se realizó la investigación	2025
URL de disciplinas OCDE	Sistemas de automatización, Sistemas de control https://purl.org/pe-repo/ocde/ford#2.02.03



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
(Universidad del Perú, DECANA DE AMÉRICA)
FACULTAD DE INGENIERÍA ELECTRÓNICA Y ELÉCTRICA
Teléfono 619-7000 Anexo 4226
Calle Germán Amezaga 375 – Lima 1 – Perú



ACTA DE SUSTENTACIÓN

TESIS

ACTA N°05/FIEE-PPP/2026

Los suscritos Miembros del Jurado, docentes permanentes de las Escuelas Profesionales de la Facultad de Ingeniería Electrónica y Eléctrica, reunidos en la fecha 15 de enero del 2026, como presidente de Jurado el **DR. CELSO YSIDRO GERÓNIMO HUAMÁN**, integrado por el Miembro de Jurado el **DR. CARLOS ALBERTO MORENO PAREDES**, el Miembro de Jurado el **MG. JEAN CARLOS MALCA FERNÁNDEZ** y Miembro Asesor el **MG. ALEX ALFONSO GONZALES SUAREZ**.

Después de escuchar la Sustentación Presencial de la Tesis del Bachiller **ROLANDO SEGUNDO HUAHUALA CUYUBAMBA** con código 16190127 que para optar el Título Profesional de Ingeniero Electrónico sustentó la Tesis titulada **SISTEMA DE SEMÁFOROS AUTÓNOMOS CON VISIÓN ARTIFICIAL PARA UNA INTERSECCIÓN VIAL EN LIMA: DISEÑO Y PROTOTIPADO**.

El jurado examinador procedió a formular las preguntas reglamentarias y, luego de una deliberación en privado, decidió otorgarle el calificativo de 17 (Quisiete).

Ciudad Universitaria, 15 de enero del 2026

DR. CELSO YSIDRO GERÓNIMO HUAMÁN
Presidente Jurado

DR. CARLOS ALBERTO MORENO PAREDES
Miembro Jurado

MG. JEAN CARLOS MALCA FERNÁNDEZ
Miembro Jurado

MG. ALEX ALFONSO GONZALES SUAREZ
Miembro Asesor



UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, Decana de América

CERTIFICADO DE SIMILITUD

Yo , **ALEX ALFONSO GONZALES SUAREZ** en mi condición de asesor acreditado con el Dictamen N°02/FIEE-PPP/2026 de la Tesis titulada **SISTEMA DE SEMÁFOROS AUTÓNOMOS CON VISIÓN ARTIFICIAL PARA UNA INTERSECCIÓN VIAL EN LIMA: DISEÑO Y PROTOTIPADO**, presentado por el bachiller ROLANDO SEGUNDO HUAHUALA CUYUBAMBA, para optar el título profesional de Ingeniero Electrónico. CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 4% de similitud, nivel PERMITIDO para continuar con los trámites correspondientes y para su publicación en el repositorio institucional.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del título profesional de Ingeniero Electrónico.

Lima, 15 de Enero del 2026

Firma del Asesor

DNI: 07644682

Nombres y apellidos del asesor:

ALEX ALFONSO GONZALES SUAREZ

DEDICATORIA

A mi familia, por darme siempre la fuerza necesaria para continuar.

De manera muy especial, a la memoria de mi padrino Teobaldo Rojas Flores, quien con su apoyo, palabras y confianza fue la inspiración que me impulsó a no rendirme y a seguir adelante en este camino.

A mis amigos y a todas las personas que me rodean, por sus palabras de aliento y compañía en los momentos más difíciles y desafiantes.

Este logro también les pertenece a ustedes.

AGRADECIMIENTOS

Agradezco al Municipio de San Miguel, por facilitarme el acceso a los videos de sus cámaras de seguridad.

A la Iglesia de Dios Sociedad Misionera Mundial, por permitirme hacer grabaciones en sus instalaciones.

A la Benemérita y Sesquicentenario Compañía Italiana de Bomberos Italia N.º 5, por su apoyo en la obtención de tomas aéreas con dron.

Al Magíster Alex Alfonso Gonzales Suarez, por su orientación académica y apoyo en el desarrollo de este proyecto.

Finalmente, mi gratitud a todas las personas que, de una u otra manera, contribuyeron para que este proyecto pudiera hacerse realidad.

ÍNDICE GENERAL

DEDICATORIA	ii
AGRADECIMIENTOS	iii
TABLA DE CONTENIDO	iv
LISTA DE TABLAS	viii
LISTA DE FIGURAS	ix
RESUMEN	x
ABSTRACT	xi
I. INTRODUCCIÓN	1
I.1. Introducción	1
I.2. Planteamiento del problema	1
I.2.1. Determinación del problema	2
I.2.2. Formulación del problema.....	3
I.2.2.1. Problema general	4
I.2.2.2. Problemas específicos.....	4
I.3. Objetivos de la investigación	5
1.3.1. Objetivo general	5
1.3.2. Objetivo específico	5
I.4. Importancia y alcance de la investigación	6
I.5. Limitaciones de la investigación	6
II. REVISION DE LITERATURA	7
II.1. Marco teórico	7
II.1.1 Teoría de visión artificial	7
II.1.2 Algoritmo y lógica de cambio de luces.....	8
II.1.3. Congestión vehicular	9
II.1.3.1. Inicios del Problema.....	9
II.1.3.2. Estructura vial	9
II.1.3.3. Demanda Vehicular vs oferta Vial.....	10

II.1.3.4. Factores en el tránsito	10
II.1.3.5. Estudio del problema	10
II.1.3.6. Soluciones en el tiempo a la congestión vehicular.....	11
II.1.3.7. Resultados de las soluciones.....	11
II.1.4. Semaforización	14
II.1.4.1. Inicios Históricos	14
II.1.4.2. Tipos de Semáforos	14
II.1.4.3. Semáforos Accionados por el Transito.....	15
II.1.4.4. Detectores	15
II.1.4.5. Etapas esenciales.....	15
II.1.4.5.1. Etapa del microcontrolador	16
II.1.4.5.2. Etapa de potencia	17
II.1.4.5.3. Etapa de monitoreo	17
II.2. Antecedentes del estudio	18
II.2.1 Antecedentes a nivel nacional	19
II.2.2. Antecedentes a nivel internacional	21
II.2.3. Sistemas usados en otras ciudades.....	23
II.2.3.1. Sistema usado en Lima	24
II.3. Bases teóricas	25
II.3.1. Normativa vial	25
II.4. Definición de términos	26
II.4.1 Factores en un sistema de semáforos accionados por el tránsito	26
II.4.2. Sensores	26
III. HIPOTESIS Y VARIABLES	28
III.1. Hipótesis	28
III.1.1 Hipótesis general	28
III.1.2 Hipótesis específicas.....	28
III.2. Variables	28
III.3. Operacionalización de variables	29
IV. MATERIALES Y METODOS.....	30
IV.1. Área de estudio	30
IV.2. Diseño de investigación	30
IV.3. Población y muestra.....	31

IV.3.1. Población	31
IV.3.2. Muestra	31
IV.4. Procedimientos, técnicas e instrumentos de recolección de información	31
IV.4.1. Cronograma del proyecto	31
IV.4.2. Presupuesto referencial	32
IV.4.3. Tecnología a emplearse en la solución	33
IV.4.3.1 Hardware.....	34
IV.4.3.1.1. Cámara ESP32-CAM	34
IV.4.3.1.2. Procesador principal: Raspberry Pi 5	34
IV.4.3.1.3. Módulo de interfaz para control de luces	35
IV.4.3.1.4. Fuente de alimentación y accesorios	35
IV.4.3.2. Software	35
IV.4.3.2.1. Sistema Operativo: Raspberry Pi OS	35
IV.4.3.2.2. Lenguaje de programación: Python	36
IV.4.3.2.3. Librerías de visión artificial: Open CV y Ultralytics YOLOv8	36
IV.4.3.2.4. Protocolos de comunicación	36
IV.4.3.3. Justificación de la elección de tecnologías	36
IV.4.4. Procedimiento experimental / metodología de implementación	37
IV.4.4.1 Etapas de implementación	37
IV.4.4.1.1. Fase inicial: Enfoque tradicional con OpenCV y modelos matemáticos	38
IV.4.4.1.2. Fase final: Enfoque con redes neuronales (YOLOv8 + OpenCV).....	40
IV.4.4.1.3 Configuración del entorno de desarrollo	42
IV.4.4.1.4 Desarrollo del software de captura y transmisión de imágenes	44
IV.4.4.1.5 Implementación de procesamiento y detección vehicular	47
IV.4.4.1.6 Parámetros utilizados para el sistema	48
IV.4.4.1.7 Desarrollo de la lógica de control adaptativo	49
IV.4.4.1.8 Pruebas y calibración del sistema	50
IV.5. Análisis estadístico.....	52
V. RESULTADOS	53

V.1. Presentación y análisis de los resultados	53
V.1.1. Metodología de evaluación	53
V.1.1.1. Sistema actual (ciclo fijo convencional)	53
V.1.1.2. Sistema propuesto (ciclo adaptativo con visión artificial)	54
V.1.1.3. Escenario de análisis	55
V.1.2. Resultados experimentales	56
VI. DISCUSIÓN.....	58
VI.1. Discusión de resultados	58
VI.2. Propuestas de mejora	59
VI.3. Limitaciones del estudio.....	60
VI.4. Análisis comparativo con trabajos similares	60
VII. CONCLUSIONES	62
VIII. RECOMENDACIONES.....	64
IX. BIBLIOGRAFÍA	66
APENDICE A	70
APENDICE B.....	71
APENDICE C.....	73
APENDICE D	79
APENDICE E.....	85

LISTA DE TABLAS

Tabla 1: Volumen por hora de vehículos	12
Tabla 2: Datos del tráfico de una rotonda sin semaforización	13
Tabla 3: Datos del tráfico de una rotonda semaforizada con pulsadores e inteligente	13
Tabla 4: Variables del proyecto.....	29
Tabla 5: Cronograma de proyecto	32
Tabla 6: Presupuesto del prototipo	33
Tabla 7: Descripción de los parámetros utilizados por el sistema.....	49
Tabla 8: Resultados comparativos entre ciclo fijo y adaptativo	56
Tabla 9: Resultados comparativos entre ciclo fijo y adaptativo con tiempos diferentes en cada vía	57
Tabla 10: Comparación entre el sistema propuesto en Lima y la tesis de Ecuador ...	61

LISTA DE FIGURAS

Figura 1: Trafico Av. Venezuela	3
Figura 2: Intersección Av. Venezuela y Av. Faucett	3
Figura 3: Esquema general de un sistema	16
Figura 4: Estructura genérica de un microcontrolador	16
Figura 5: Tarjeta de Potencia para 2 semáforos	17
Figura 6: Esquema de la etapa de monitoreo	18
Figura 7: Prototipo Inicial implementado con OpenCV estado del semáforo, zona de interés, mascara de sustracción de fondo y bandas de demanda	39
Figura 8: Resultados de detección con YOLOv8 en el prototipo	41
Figura 9: Raspberry Pi5	43
Figura 10: Configuración del access point	43
Figura 11: Librerías del entorna virtual	44
Figura 12: Seudocódigo de Microcontrolador Esp32 Cam	45
Figura 13: Seudocódigo de Microcontrolador Raspberry Pi	46
Figura 14: Aplicación del cuadrante de interés	47
Figura 15: Uso del modelo entrenado YOLO	47
Figura 16: Aplicación de lógica de demanda por áreas	48
Figura 17: Parámetros tomados en cuenta para el sistema	48
Figura 18: Seudocódigo de la lógica del control adaptativo	50
Figura 19: Visualización del prototipo en operación de demanda vehicular en las cuatro vías	51
Figura 20: Videos del cruce	54
Figura 21: Videos de Faucett	54
Figura 22: Videos tomados por dron	55

RESUMEN

Los semáforos juegan un rol importante en la vida cotidiana de muchos ciudadanos, pero en Lima y Callao en la mayoría de intersecciones aun funcionan con ciclos fijos que no se pueden adaptar a una demanda de vehículos en tiempo real, este tipo de limitaciones genera congestión, pérdida de tiempo y dinero.

La iniciativa de este proyecto nace a partir de estas observaciones, donde se plantea el objetivo de diseñar el prototipo de un sistema de semáforos autónomos con visión artificial, aplicándolo en el cruce de avenidas Faucett y Venezuela, en el distrito de Bellavista, Callao. Todo esto bajo la metodología del procesamiento digital de imágenes mediante cámaras ESP32-CAM y un módulo central en Raspberry Pi 5, utilizando la librería OpenCV y el modelo YOLOv8 para la detección de vehículos y la lectura del nivel de demanda.

Los resultados de las simulaciones con videos reales demuestran que el sistema adaptativo mejora la gestión del tiempo de luz verde, así reduciendo la espera innecesaria dando prioridad a la vía con mayor flujo.

En conclusión, implementar este prototipo demuestra que es posible una solución a bajo costo y sobre todo escalable para mejorar su desempeño para la optimización de el transito urbano.

Palabras clave: Semáforo autónomo, visión artificial, YOLOv8, gestión de tráfico, Raspberry Pi.

ABSTRACT

Traffic lights play an important role in the daily lives of many citizens, but in Lima and Callao, most intersections still operate on fixed cycles that cannot adapt to real-time vehicle demand. This type of limitation generates congestion and wastes time and money.

The initiative for this project was born from these observations, with the objective of designing a prototype of an autonomous traffic light system with computer vision, applying it to the intersection of Faucett and Venezuela avenues, in the Bellavista district of Callao. All of this is based on the methodology of digital image processing using ESP32-CAM cameras and a central module on a Raspberry Pi 5, using the OpenCV library and the YOLOv8 model for vehicle detection and reading demand levels.

The results of simulations with real videos demonstrate that the adaptive system improves green light management, thus reducing unnecessary waiting times by giving priority to the road with the greatest traffic flow.

In conclusion, implementing this prototype demonstrates that a low-cost and, above all, scalable solution is possible to improve its performance for optimizing urban traffic.

Keywords: Autonomous traffic light, computer vision, YOLOv8, traffic management, Raspberry Pi.

CAPÍTULO I: INTRODUCCIÓN

I.1 Introducción

Según estudios, Lima continúa siendo la ciudad con la mayor congestión vehicular de América Latina y la quinta a nivel mundial. En 2023, recorrer 10 km en la capital peruana tomó en promedio 28 minutos con 30 segundos, lo cual representó un incremento de 1 minuto y 20 segundos respecto al año anterior (AAP, 2024), en otras palabras, se toma un 96% más de tiempo que en condiciones sin tráfico. Esta situación no solo afecta la calidad de vida de los ciudadanos, sino que también genera pérdidas económicas estimadas en aproximadamente 2 mil millones de soles anuales en productividad y un gasto adicional de S/ 652 por conductor al año en combustible (Infobae, 2024).

Entonces debido a la congestión vehicular, recorrer 10 km en Lima toma un 96% más de tiempo que lo estimado en condiciones sin tráfico, lo que genera además un gasto adicional promedio al año en combustible. Y a pesar que se anunció una inversión de S/ 300 millones por parte del Municipio de Lima integrar un sistema semafórico con inteligencia artificial, esto resalta lo costoso que resulta introducir una solución (Infobae, 2025). Por eso, esta propuesta plantea una alternativa más económica, funcional y localizada: el diseño de un sistema de semáforos autónomos con visión artificial, aplicado específicamente en el cruce de la Av. Faucett con Av. Venezuela, en el distrito de Bellavista, Callao.

I.2 Planteamiento del problema

Actualmente, el sistema semafórico no cumple con las condiciones adecuadas y necesarias para el entorno real de nuestra ciudad. A pesar de existir diferentes tipos de

semáforos instalados en Lima y Callao, la gran mayoría sigue operando con ciclos fijos que no se ajustan al flujo vehicular en tiempo real. Según ITS Perú (2024), aproximadamente 800 intersecciones en Lima se encuentran integradas a una red semafórica centralizada. No obstante, el total de intersecciones semaforizadas en Lima y Callao asciende a 1,790, de acuerdo con el Plan Regulador de Rutas 2024 (ATU, 2024), lo que indica que más del 55% de los semáforos aún funcionan sin conexión a un sistema de gestión inteligente. Esta falta de adaptación dinámica produce pérdidas de tiempo innecesarias, incluso en momentos donde las vías están despejadas, afectando de manera negativa la eficiencia del tránsito urbano.

En el presente trabajo se realiza una investigación cuantitativa, aplicada y experimental, ya que, a partir de la recopilación y análisis de datos, buscaremos solucionar un problema de la sociedad como es la adaptabilidad del semáforo en el respectivo cruce. Para ello, se aplicarán todos los conocimientos teóricos recopilados, se realizará la simulación y pruebas con un prototipo, y se tendrán en cuenta las recomendaciones de los antecedentes consultados para un correcto desarrollo.

I.2.1 Determinación del problema

Lo que se busca en este proyecto es analizar la problemática de la falta de un sistema que interprete el entorno mediante imágenes y actúe de forma autónoma. Es por ello que se plantea el diseño de un modelo funcional que trabaje con visión artificial, capaz de interpretar en tiempo real la cantidad de vehículos en circulación y definir automáticamente el tiempo y estado de las luces del semáforo. Este sistema estará compuesto por una cámara para la detección vehicular y un microcontrolador para el procesamiento local de imágenes mediante algoritmos programados. La propuesta se aplicará específicamente al cruce de las avenidas Faucett con Venezuela, en el distrito de Bellavista, Callao, una intersección de alto tráfico y desorden peatonal.

Al delimitar el proyecto a esta única intersección, se podrá evaluar con precisión el desempeño del sistema, logrando ajustar automáticamente los tiempos de las luces del semáforo en función del flujo vehicular detectado, y centrando la investigación en la adquisición, procesamiento y análisis de imágenes como eje principal del desarrollo técnico.

Figura 1

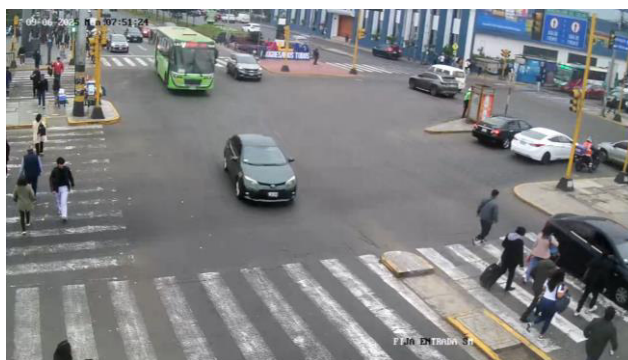
Tráfico en la Av. Venezuela



Nota. Adaptado de *Bellavista: gran congestión vehicular*, por *Prensa Chalaca* (2022). Todos los derechos reservados 2022 por Prensa Chalaca.

Figura 2

Intersección Av. Venezuela y Av. Faucett



Nota. Imagen proporcionada por el *Municipio de San Miguel* (2025). Todos los derechos reservados por el Municipio de San Miguel.

I.2.2 Formulación del problema

Luego del planteamiento expuesto, ahora la incógnita se centra en cómo diseñar un sistema que permita ajustar automáticamente los tiempos del semáforo en función del flujo vehicular y que responda técnicamente a las condiciones reales del cruce de las

avenidas Faucett con Venezuela, en el distrito de Bellavista, Callao. En este punto, los semáforos funcionan de manera pre sincronizada, sin capacidad de adaptación al flujo vehicular, lo cual genera congestión severa durante las horas punta, especialmente en la Av. Faucett. Además, los peatones enfrentan tiempos de espera excesivos para cruzar avenidas anchas como Faucett, lo que agrava la incomodidad y el desorden en el cruce. También se ha observado una distribución desigual del tráfico, con acumulaciones en ciertas direcciones mientras otras permanecen despejadas, y conflictos en los giros a la derecha, principalmente en Faucett. A pesar de la presencia ocasional de personal de tránsito, el sistema semafórico actual mantiene los mismos ciclos incluso durante la noche, cuando el flujo vehicular es distinto. Frente a estas limitaciones, el reto de esta investigación es plantear un sistema de bajo costo, basado en visión artificial, que permita detectar la cantidad de vehículos en tiempo real y ajustar automáticamente la duración de las luces, considerando que los sistemas actualmente instalados en dicha intersección funcionan con limitaciones operativas, y mejorando así la eficiencia del cruce sin depender de una red centralizada.

I.2.2.1 Problema General

Es debido a esto que nos preguntamos cómo problema general:

¿Cómo diseñar un sistema de semáforo autónomo, basado en visión artificial, que active sus luces en función de la detección en tiempo real de vehículos y peatones mediante una cámara de bajo costo, para el cruce de las avenidas Faucett con Venezuela, en el distrito de Bellavista, Callao?

I.2.2.2 Problema Específicos

El sistema debe ser evaluada tanto su eficiencia como su costo, es aquí cuando apuntamos al problema general, la de interpretación de imágenes, por ende, los problemas específicos serian:

¿Cómo diseñar un algoritmo de procesamiento de imágenes que permita detectar vehículos y peatones con una precisión utilizando una cámara de bajo costo?

¿Cómo implementar un sistema que, a partir de la detección en tiempo real, active automáticamente las luces del semáforo de acuerdo con el flujo vehicular y peatonal?

¿Cómo garantizar que el sistema opere de forma autónoma, con bajo consumo energético y capacidad de respuesta en tiempo real, sin depender de una red centralizada?

I.3 Objetivos de la investigación

Ya que este nuestro sistema precisa de un correcto desarrollo e investigación en el tratamiento visual del entorno vial, es necesario que a partir del problema que nos planteamos determinar nuestros objetivos así delimitar la llegada de nuestra investigación.

I.3.1 Objetivo general

A partir del problema general propuesto el objetivo esencial de nuestra investigación es el de:

Diseñar un sistema de semáforo autónomo con visión artificial, de forma económica, que active sus luces en función de la detección en tiempo real de vehículos y peatones mediante una cámara de bajo costo, para el cruce de las avenidas Faucett con Venezuela, en el distrito de Bellavista, Callao.

I.3.2 Objetivos específicos

Y para este desarrollo específicamente debemos proponernos el siguiente objetivo:

Diseñar un algoritmo de procesamiento de imágenes que permita detectar vehículos y peatones con una precisión usando una cámara de bajo costo.

Implementar un sistema embebido que active automáticamente las luces del semáforo en función de los datos obtenidos por el algoritmo de detección.

Desarrollar una arquitectura autónoma que garantice bajo consumo energético y una respuesta en tiempo real sin depender de una red semafórica centralizada.

I.4 Importancia y alcance de la investigación

La importancia de este proyecto reside en dar una respuesta practica a un problema que afecta a los ciudadanos de Lima y Callao, la congestión vehicular. El desarrollo del prototipo de un semáforo autónomo con visión artificial no solo es un aporte en el campo de la ingeniería electrónica, el impacto esperado es una mejora eficiente en el cruce mediante la gestión dinámica del tiempo de luz, con un diseño equivalente y de bajo costo frente a un sistema centralizado. Esta aplicación puede mejorar el flujo vehicular en intersecciones críticas.

Si bien existen múltiples factores que afectan el tráfico, esta investigación se enfoca únicamente en el diseño técnico de un sistema semafórico autónomo basado en visión artificial. El alcance se limita a una sola intersección, sin abordar temas de fiscalización o comportamiento vial.

I.5 Limitaciones de la investigación

Esta investigación tiene algunas limitaciones que es fundamental identificar. En primera instancia, la investigación se limita solo a la intersección de las avenidas Venezuela y Faucett en el distrito Bellavista de Callao, por lo que no es posible extrapolar los resultados directamente a todos los cruces de Callao y Lima. En segundo lugar, el prototipo se enfoca en la detección de vehículos a través de visión artificial y en el manejo dinámico del tiempo semafórico, sin tratar temas vinculados al control del tránsito, la conducta de los conductores o la seguridad vial integral. Y finalmente la validación del sistema se realiza con videos pregrabados y pruebas en entorno de simulación, lo que significa que su prueba en un entorno real aún debe ser evaluado en una futura implementación en campo.

CAPÍTULO II: REVISIÓN DE LA LITERATURA

II.1. Marco teórico

II.1.1. Teoría de visión artificial

La visión artificial es un campo de la inteligencia artificial y el procesamiento digital de imágenes donde su propósito es dar a las máquinas de la habilidad de interpretar y examinar datos visuales del entorno, simulando parcialmente el sistema visual humano. Como aclara Szeliski (2022), esta rama busca reproducir las habilidades como segmentar objetos, reconocer formas y hasta determinar propiedades o emociones a partir de patrones visuales, tareas naturales para el ser humano pero un reto significativo, técnico y científico (p.3).

Desde una perspectiva técnica la visión artificial se sustenta en procesar digitalmente de imágenes, el cual consiste en manipulación mediante computadora. Cada imagen está compuesta por un conjunto finito de elementos denominados píxeles, que poseen una ubicación y un valor específicos, siendo la unidad mínima de información visual sobre la cual se aplican las operaciones de análisis y procesamiento (González & Woods, 2018, p.18).

En múltiples sistemas este procesamiento se desarrolla siguiendo etapas importantes: adquisición de imagen, donde se captura y digitaliza la escena; preprocesamiento, orientado a mejorar la calidad de la imagen mediante reducción de ruido o corrección geométrica; segmentación, que separa la imagen en regiones u objetos de interés; representación y descripción, donde se obtienen atributos cuantitativos de los objetos segmentados; reconocimiento, que asigna etiquetas a los objetos detectados; e interpretación, que otorga un significado global en función de la aplicación (Gonzalez & Woods, 2018, pp.41-43).

YOLO (You Only Look Once) es uno de los algoritmos de detección más utilizados en este ámbito, es un modelo de aprendizaje basado en redes neuronales convolucionales, en el que se examina la imagen completa en una única ocasión, detectando y rastreando objetos en tiempo real de manera eficaz (Redmon, Divvala, Girshick, & Farhadi, 2016).

II.1.2. Algoritmo y lógica de cambio de luces

El sistema, utilizando la información suministrada por el modelo, calcula el número de vehículos que se encuentran en áreas de interés previamente definidas para determinar la demanda de paso en cada vía. Esta estimación de la movilidad o la detención de vehículos permite distinguir entre congestión, tráfico fluido y falta de demanda, por lo que este conteo se completa.

Los principios del control adaptativo son la base de los fundamentos lógicos para alternar luces:

1. La luz verde se queda encendida en una carretera con alta demanda y tráfico constante, hasta que se alcanza un límite máximo previamente definido.
2. Si se observa que un número de vehículos se acumula sin moverse sustancialmente durante un período de tiempo específico, se cambia a luz roja para permitir el paso en otra vía.
3. En la falta de vehículos en la vía actual, se da prioridad a cambiar de dirección con antelación hacia otras que tengan más demanda.

Este método concuerda con estudios anteriores de sistemas inteligentes de control semafórico por medio de visión artificial, en los cuales la duración del verde se adapta constantemente en función de la densidad vehicular que se detecta en tiempo real (Chong, Damanhuri y Othman, 2021). El sistema intenta mejorar la fluidez del tráfico y disminuir tiempos de espera innecesarios en el cruce al combinar una detección precisa con una lógica de control flexible.

II.1.3. Congestión vehicular

II.1.3.1. Inicios del problema

Con la invención del vehículo a motor su uso y su introducción como transporte público, prontamente iniciaría su tendencia a su uso privado, generando el inicio de los problemas del tránsito urbano contemporáneo.

Como ya lo constatamos, a fines del siglo XX el transporte experimento cambios influyendo en la población y buscando adaptarse a la economía creciente, a la globalización y al desarrollo de la informática, ahora en este siglo los nuevos sistemas deben ser seguros, eficientes, integrados y en especial ambientalmente sostenibles pero el automóvil como medio de transporte estará cuestionado, pues si se sigue con ese aumento exponencial, para atender a solo un pequeño sector, será necesario el aumento de calles y vías, que tarde o temprano terminaran aglomeradas y no abra espacio para más, debido a eso que las soluciones venideras son sistemas integrados al transporte público, el uso de bicicletas y otro tipo de transporte alternativo, por ende el uso de la electricidad, la electrónica y los sistemas satelitales, y posiblemente el transporte ya se optaría por vías elevadas y ya no de superficie (Cal y Mayor, 2018).

II.1.3.2. Estructura vial

La gran parte de las vías en el mundo están diseñadas inicialmente para los primeros medios de transporte que eran carruajes y diligencias tiradas por animales, es común observar que la velocidad del proyecto de ese entonces es muy menor a la que hoy en día estamos acostumbrados a viajar, y no solo eso también las distintas características de la vía como pendiente, curvatura y capacidad de carga, que estaban más ligados a medios de transporte lentos, pequeños y ligeros. Una carretera en los años 60 se estimaba que tuviera vehículos que podrían llegar a 60 km/h que en ese tiempo esto era avanzado, ahora una velocidad de 100 km/h se considera algo normal en vehículos contemporáneos (Cal y Mayor, 2018). Entonces los proyectos o diseños de las nuevas estructuras viales deben adaptarse a las nuevas velocidades de los vehículos modernos.

II.1.3.3. Demanda vehicular vs oferta vial

La demanda vehicular hace referencia a la cantidad de automóviles que requieren movilizarse dentro de un sistema vial, mientras que la oferta vial corresponde a la capacidad máxima que dicho sistema puede atender. En este sentido, la demanda incluye tanto a los vehículos que se encuentran circulando, aquellos que esperan en colas, como también los que optan por rutas alternas. De acuerdo con Cal y Mayor (2018), cuando la demanda es menor que la oferta, el flujo vehicular se mantiene estable y con niveles de servicio adecuados, si ambos valores son equivalentes, el sistema alcanza su capacidad máxima, lo que genera inestabilidad y riesgo de congestión, y cuando la demanda supera a la oferta, se produce un flujo forzado con detenciones y demoras significativas.

II.1.3.4. Factores en el tránsito

En los últimos años con el avance tecnológico se han logrado construir vías de transporte más acorde con la necesidad del entorno urbano y sus áreas aledañas, pero aun así los problemas de tránsito relacionados a diferentes factores aún existen, y estos son:

1. Diferentes tipos de vehículos en la misma vía.
2. Ausencia de un plan de tránsito.
3. El vehículo no reconocido como necesidad pública.
4. Falta de comprensión por parte del estado y los usuarios.

II.1.3.5. Estudio del problema

Para enfrentar esta problemática, Cárdenas (2018) propone una serie de etapas metodológicas que permiten analizarla y orientar posibles soluciones. Estas etapas incluyen: la observación inicial del problema, la formulación de hipótesis, la recolección y análisis de datos, la elaboración de propuestas específicas y la evaluación de los posibles resultados.

El autor enfatiza que la recopilación de información debe realizarse en el área afectada y a partir de fuentes estadísticas y oficiales, ya que las opiniones personales no

constituyen evidencia suficiente. Una vez procesados los datos por un especialista, se plantea un proyecto de solución que considere tres aspectos esenciales: el entorno físico y urbanístico (incluyendo el uso del suelo), la adaptación a las características de los vehículos y conductores, y finalmente la implementación de medidas de educación vial junto con reformas normativas y de control que garanticen su cumplimiento.

II.1.3.6. Soluciones en el tiempo a la congestión vehicular

Este problema genera pérdidas materiales y humanas, y se vincula directamente con la inseguridad ciudadana. Una manera de enfrentarlo es garantizar un tránsito más seguro y eficiente. En este sentido, Cárdenas (2018) propone tres enfoques de solución:

1. Solución integral: implica diseñar y construir nuevas vialidades adaptadas a las características de los vehículos modernos, lo que requiere una planificación urbana de mayor alcance.
2. Solución parcial de bajo costo: se basa en aprovechar al máximo la infraestructura existente, aplicando medidas de regulación, técnicas de gestión del tránsito y fomentando la disciplina y educación de los usuarios.
3. Solución parcial de alto costo: contempla intervenciones más complejas como la ampliación de vías, la modificación de intersecciones, la instalación de sistemas semafóricos automáticos o la construcción de estacionamientos públicos y privados, entre otras obras de infraestructura.

II.1.3.7. Resultados de las soluciones

Después de exponer los diferentes tipos de soluciones para la congestión vehicular, ahora expondremos resultados que apoyen la teoría sustentada.

En Estados Unidos específicamente en el estado de Kentucky una aplicación de señales de tránsito desarrollada por el Centro de Transporte de Kentucky (KTC), con la intención de demostrar que era viable el estudio e instalación de señales de tránsito para el correcto tránsito, teniendo en cuenta el factor de la cantidad de vehículos que transitaban en las vías estudiadas en el lapso de una hora, volumen por hora (vph),

donde la investigación indica que en la mayoría de intersecciones el volumen de tránsito se mantuvo y hubo una reducción de tráfico en la tercera parte de las intersecciones estudiadas, como podemos observar en la tabla.

Tabla 1

Volumen vehicular por hora antes y después de la instalación de señales

ID	Location	County	Warrant	Volume (vph)	
				Before	After
1*	US 127 B @Bellerive	Anderson	Yes	2473	1688
2	US 62 @ Robert B. Turner/Westwood	Boone	No	534	560
6	KY 237 @ Rogers Lane	Boyle	Yes	1040	1200
8	US 150B @ Daniel Drive	Carroll	Yes	1470	1443
12	KY 227 @ KY 36	Christian	Yes	1211	1324
14	US 41 @ Murray State Ext. Campus	Christian	No	1412	1372
15	US 41A @ Bradford Sqr. Mall Entr.	Daviess	Yes	2208	2214
19	KY 2698 @ Unifirst Drive	Daviess	No	1576	1780
20	KY 2698 @ Tamarck Road	Daviess	Yes	1073	1300
21*	US 431 @ Home Depot Entrance	Fayette	Yes	1506	1088
23	US 25 @ Sandersville Road	Fayette	Yes	1889	2012
24	US 27 @ Old Paris Pike	Fayette	Yes	2172	2386
25	US 421 @ Ruffian Way	Hardin	Yes	809	1236
34	KY 251 @ Panther Lane	Jefferson	Yes	1568	1918
40*	KY 1065 @ Vaughn Mill Road	Jefferson	Yes	2205	1602
41	KY 1230 @ KY 1934	Jefferson	Yes	1501	1560
43*	KY 1747 @ Home Depot/Target Entr.	Jefferson	Yes	2696	2690
45	KY 3084 @ Nelson Miller Pkwy.	Jefferson	Yes	1563	1644
46*	KY 61 @ Interchange Dr.	Jefferson	Yes	3784	3250
47*	KY 864 @ Jefferson Blvd.	Jefferson	Yes	NA	1444
48	KY 913 @ Plantside Drive	Jefferson	Yes	1611	2842
49	US 42 @ KY 329	Jefferson	Yes	1876	2162
52	KY 321 @ Federal Drive	Johnson	Yes	1374	1416
55*	KY 17 @ Old KY 17	Kenton	Yes	NA	2411
58	US 23 @ KY 2565	Lawrence	Yes	815	772
59	KY 52 @ KY 977	Madison	Yes	998	1186
60	US 25 @ KY 1986	Madison	Yes	1759	1834
61	US 25X @ Boggs Lane/Morrow Drive	Madison	Yes	1575	1582
62	US 68 @ Warehouse Drive	Marion	No	1894	1807
67	KY 686 @ Old Owingsville Rd	Montgomery	Yes	1039	1026
78	KY 55 @ KY 43/KY 2268	Shelby	Yes	888	1140
79	US 60 @ KY 1848	Shelby	Yes	1176	1144

Nota. Reproducido de *Operational and safety efficiency of traffic signal installations* (p. 8), por Stamatiadis, Jones y Agarwal (2008), Kentucky Transportation Center, University of Kentucky. Todos los derechos reservados 2008 por Kentucky Transportation Center.

Donde la investigación finalmente apunta que las señales de tránsito que no cumplan con las ordenes apropiadas tendrán efectos operativos negativos y tienen el potencial de crear riesgos de seguridad.

Otro estudio donde se analizan diferentes esquemas para la solución del tráfico vial es el de Acosta (2019), donde a partir de una rotonda se examinan dos posibles soluciones: semáforos accionados por pulsadores y semáforos inteligentes, mostrando los siguientes resultados.

Tabla 2

Datos del tráfico de una rotonda sin semaforización

	Escenario 1	
	Media	IC
Vehículos Creados	590,86	(306;1085,975)
Vehículos Salientes	344,50	(53;1067)
Vehículos Promedio en el sistema	76,50	(52,65;114,20)
Peatones Salientes	749,05	(695;803)

Nota. Reproducido de *Simulación de semáforo inteligente* (p. 26), por Acosta (2019), Tesis de grado, Universidad de La Laguna.

Tabla 3

Datos del tráfico de una rotonda semaforizada con pulsadores e inteligente

	Escenario 3			Escenario 4		
	Media	IC	% mejora Escenario1	Media	IC	% mejora Escenario1
Vehículos Creados	347,97	(276;446)	-41,11	982,48	(929;1036)	66,28
Vehículos Salientes	97,53	(44;192,98)	-71,69	977,84	(927;1031)	183,84
Vehículos Promedio en el sistema	145,46	(78,25;268,24)	90,14	137,53	(113,10;169,06)	79,78
Peatones Salientes	449,31	(403;496)	-40,02	749,45	(695;803)	0,05

Nota. Reproducido de *Simulación de semáforo inteligente* (p. 26), por Acosta (2019), Tesis de grado, Universidad de La Laguna.

Concluyendo finalmente que, una vez analizado los resultados, podemos decir que la inclusión de los semáforos inteligentes mejora la intensidad media diaria de los vehículos, reduciendo así las retenciones que se producen a día de hoy.

II.1.4. Semaforización

II.1.4.1. Inicios históricos

Los primeros semáforos surgieron en Londres en 1868 y eran operados manualmente como una extensión mecánica de las señales que realizaba el agente de tránsito. Años más tarde, en 1914, se instaló el primer semáforo eléctrico en Cleveland (Estados Unidos) y en 1917, en Salt Lake City, se implementó por primera vez un sistema de interconexión entre ellos.

Con el crecimiento urbano y el aumento tanto del tráfico como de los accidentes, el uso de semáforos se fue perfeccionando y hoy constituye un elemento esencial en las grandes ciudades. La evolución tecnológica ha permitido la incorporación de sistemas coordinados y computarizados, junto con detectores automáticos de vehículos que ajustan de forma dinámica los tiempos de cada fase según la demanda. Gracias a estos avances, se han desarrollado estrategias de control que adaptan la operación semafórica a las diferentes horas del día, diferenciando entre periodos de alta y baja circulación.

II.1.4.2. Tipos de semáforos

Existen distintos tipos de semáforos según las condiciones de tránsito en que se aplican. En primer lugar, se encuentran los semáforos de tiempo fijo, empleados en intersecciones con patrones de circulación relativamente estables o con variaciones que pueden ajustarse a un programa preestablecido, evitando así demoras y congestiones. En segundo lugar, están los semáforos accionados por el tránsito, recomendados en vías donde el flujo vehicular es irregular y se requiere dar prioridad a la circulación principal reduciendo al mínimo las interrupciones. Finalmente, se tienen los semáforos totalmente accionados, que cuentan con mecanismos de detección en todos los accesos de la intersección, permitiendo que el control del tránsito se realice en función directa de la demanda registrada según Cárdenas (2019).

II.1.4.3 Semáforos accionados por el tránsito

Los semáforos accionados por el tránsito se caracterizan porque la duración de sus ciclos depende directamente de la demanda vehicular registrada en tiempo real. Dicha información es captada mediante dispositivos detectores conectados al sistema de control, lo que permite ajustar de manera continua tanto el tiempo total del ciclo como la distribución interna de sus fases. En este tipo de control, la secuencia no es rígida: si alguna fase no presenta demanda, simplemente se omite.

Cárdenas (2019) plantea que para implementar este tipo de semáforos es necesario evaluar previamente ciertos factores clave, tales como el volumen vehicular, el movimiento transversal, los horarios de máxima afluencia, la presencia de peatones y la incidencia de accidentes.

II.1.4.4. Detectores

Los diversos tipos de detectores para registrar el paso de vehículos en las intersecciones con control accionado, son los siguientes:

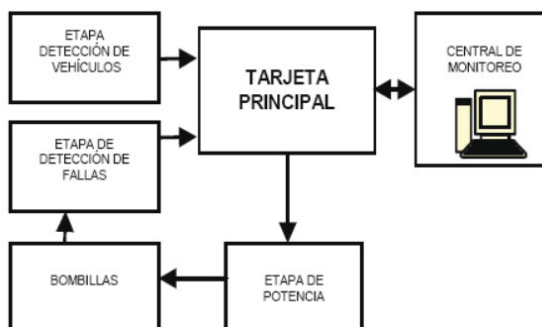
1. De presión.
2. Magnéticos.
3. De radar.
4. De video.

II.1.4.5. Etapas esenciales

Con el avance de la tecnología se ha logrado diseñar diferentes tipos de semáforos para un eficiente conteo de vehículos, comunicación entre sus dispositivos y autonomía, donde podemos determinar que todos los sistemas que se respaldan con una central de monitoreo, siguen el siguiente esquema

Figura 3

Esquema general de un sistema



Nota. Reproducido de *Sistema de comunicación TCP/IP para una intersección de tráfico vehicular*, por Pedraza (2012), Tesis de la Universidad Francisco José de Caldas. Todos los derechos reservados 2012 por Luis Pedraza Martínez.

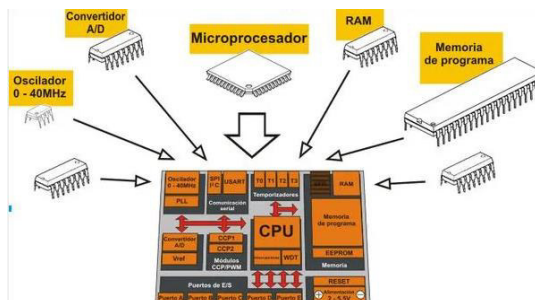
Donde la etapa de detección consta de los sensores anteriormente mencionados, esta información debidamente capturada es enviada a la tarjeta principal, este bloque generalmente es ocupado por un microcontrolador.

II.1.4.5.1. Etapa del microcontrolador

Un microcontrolador es un circuito integrado programable, capaz de ejecutar ordenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen con una tarea específica (Gridling, 2007).

Figura 4

Estructura genérica de un microcontrolador



Nota. Reproducido de *Introducción al mundo de los microcontroladores* (cap. 1), por MikroElektronika(2013). Todos los derechos reservados 2013 por MikroElektronika.

En esta tarjeta se programan los tiempos de duración de las luces de los semáforos de la intersección y enviados a la etapa de potencia para la ejecución de la función de los semáforos, aunque estos pueden ser modificados por la central de monitoreo a través de comunicación basada en los protocolos TCP/IP, mediante tarjetas seriales ethernet.

II.1.4.5.2. Etapa de potencia

Por medio de esta etapa se transforman los estados lógicos provenientes del microcontrolador, a los niveles de corriente alterna manejados por las luces de los semáforos.

Figura 5

Tarjeta de Potencia para 2 semáforos.



Nota. Reproducido de *Sistema de comunicación TCP/IP para el control de una intersección de tráfico vehicular*, por Pedraza (2012), Tesis de la Universidad Distrital Francisco José de Caldas. Todos los derechos reservados 2012 por Luis Pedraza Martínez.

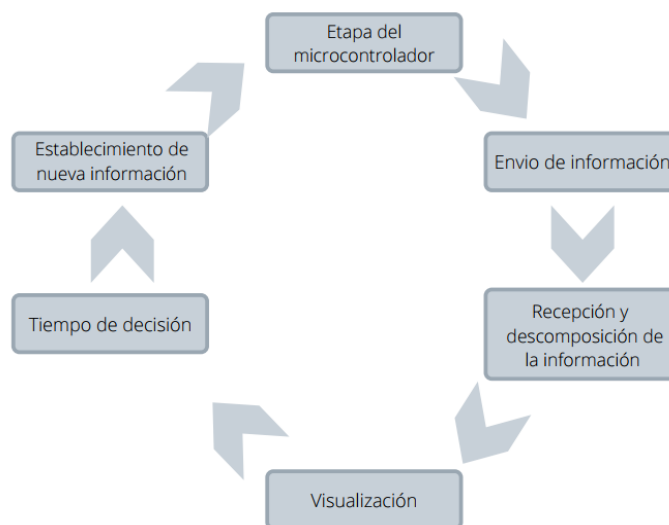
Esta tarjeta a su vez censa, mediante transformadores, las luces rojas fundidas y el encendido de los bombillos verdes, lo cual reporta al microcontrolador.

II.1.4.5.3. Etapa de monitoreo

Esta etapa permite visualizar el estado de las luces de la intersección de forma remota, así como la administración de los tiempos de las luces verdes y rojas de la intersección.

Figura 6

Esquema de la etapa de monitoreo.



Nota. Adaptado de *Sistema de comunicación TCP/IP para el control de una intersección de tráfico vehicular*, por Pedraza (2012), Tesis de la Universidad Distrital Francisco José de Caldas. Todos los derechos reservados 2012 por Luis Pedraza Martínez.

II.2. Antecedentes del estudio

La problemática que incentivo a la realización de esta investigación es sin duda algo que lleva en nuestra ciudad muchos años, es razonable pensar que otros investigadores plantearan formas de mejorar el sistema vial tanto en nuestro país como en el extranjero, que sirven de inspiración para las nuevas investigaciones que buscan mejorar estos planteamientos o usar las nuevas tecnologías para la estructuración de sistemas innovadores.

En el Perú investigaciones antecedentes frente a esta problemática existen, con distintos enfoques en la forma de abordar el problema, también es enriquecedor la manera de como estas investigaciones analizaron el contexto social y cultural que hay detrás de este problema. Y aunque nuestro análisis y enfoque en el diseño distan de muchas maneras, podemos decir que nuestro objetivo es el mismo, solucionar esta gran problemática.

Los antecedentes internacionales cuentan también como una gran fuente de ideas y análisis, sin embargo, debemos formularnos la siguiente pregunta, ¿el planteamiento del problema y el diseño es trasladable a nuestro contexto social? Aunque las mejores tecnologías se encuentran en países de primer mundo estas están acondicionadas a contextos culturales muy diferentes a los nuestros, se piensa que este tipo de antecedentes se debe tener en cuenta ya sea por su innovación tecnológica (si es que se cuenta con los recursos) o ingenio, sin embargo, debemos tener en cuenta los contrastes que poseen frente a nuestra sociedad.

II.2.1. Antecedentes a nivel nacional

Tenemos como primer antecedente la tesis de Arce (2017), fuente principal de esta investigación que se basa en la determinación de los tiempos de cambio a través de datos recopilados por un sistema de adquisición de parámetros de tráfico (SAPT), estos datos son enviados entre semáforos que establecerán su tiempo de cambio de manera autónoma, y a una base de datos que son analizados en una interfaz donde sus valores iniciales de cambio también podrán ser modificados, su diseño se basa en una interfaz de monitoreo, controlador y para la parte de comunicación rúters en cada semáforo.

En su estudio del tráfico de Lima Metropolitana y sus avenidas principales, se identificó a las que poseen mayor carga de vehículos y reconoció los factores que generan tráfico vehicular, con la herramienta de Google Maps logro observar eventos en las horas punta, determinando en el área que nos interesa San Miguel, tres cruces importantes donde se genera tráfico Av. Faucett con Venezuela, La Marina con Riva güero y Universitaria con La Mar. También detecto los principales componentes que generan el poco flujo vial; fases de semáforo, longitud de cola y el aforo vehicular, también se analizó los diferentes sistemas de control que existen tanto en el país como en el extranjero, como el de la ciudad de Barranquilla en Colombia, el cual registra la cantidad de vehículos detenidos por medio de cámaras de video y de acuerdo a eso se modifica el tiempo de pase, luego se menciona el sistema SCOOT que hace uso de sensores ubicados en las calzadas y hacen un recuento de la demanda vehicular y de acuerdo a eso se modifica el tiempo de cambio de fase. En el Perú, Lima posee en algunas de sus avenidas el sistema SICE 2016, haciendo uso de cámaras que son monitoreadas en una central se modifica el cambio de luz, este sistema presenta una

limitante debido a que deben ser operados por personas, ósea manualmente. Entre sus objetivos específicos el autor busco, presentar un diseño viable y desarrollar un algoritmo para el correcto funcionamiento del controlador. La presente tesis busca a partir del antecedente mencionado su máxima reducción de costo con las nuevas tecnologías que surgieron en este lapso de tiempo.

Con el paso del tiempo nuevas tecnologías emergen, una tesis innovadora a tener en cuenta es la tesis de De La Cruz, Risco y Paredes (2019), sistema que se basa en el uso de inteligencia artificial, a través de una cámara que enviara imágenes a una computadora que reconocerá el número de autos y a partir de eso el sistema estimará el tiempo necesario para que esa cantidad pase. En este antecedente formula su problema a partir de la cultura social, plantea el modelo Ishikawa el cual formula que las variables peatones, semáforos, conductores y vehículos determinan el tiempo de congestión vehicular.

Esta investigación presenta el problema específico haciendo observaciones del actual sistema de semáforos, el cual solo está programado para el cambio de estado, pero este no orienta el flujo vehicular. Su interfaz se basa en lenguaje de alto nivel, en este caso Python, usando las librerías Numpy y Matplotlib para las ecuaciones matemáticas, TensorFlow para el uso de redes neuronales y predecir patrones, y Labelling el cual es una herramienta para delimitar objetos en recuadros tomados en video, todas estas herramientas en conjunto crean una red neuronal, donde su objetivo es aprender modificándose así mismo, es así que antes de pasar a ser implementado e instalado en la zona ya designada, en este caso la Av. La Molina con Javier Prado, el sistema fue entrenado con múltiples fotos y videos para el correcto reconocimiento de un auto, todo esto es procesado por una computadora, obteniendo finalmente el próximo tiempo de cambio de luz del semáforo en base a la cantidad de vehículos, sin embargo, este sistema posee algunos inconvenientes, solo funciona de día y su sistema de reconocimiento, aunque con resultados satisfactorios no es el más preciso, debido a que la inteligencia artificial aún sigue siendo una tecnología emergente que aun desconocemos en su totalidad, pero con el tiempo no cabe duda que distintos investigadores nacionales tomaran esta investigación como referente, aplicando el avance que va a tener este campo de la tecnología con el tiempo.

En la Universidad Nacional del Santa, Apestegui y Caselli (2024) publicaron un artículo donde se desarrolló un sistema aplicable a intersecciones viales de alta congestión. El sistema emplea el algoritmo YOLOv4 para el conteo de vehículos y el método de detección de intrusiones (MDI) para la medición de velocidad, integrando un algoritmo heurístico que ajusta los tiempos de fase según la velocidad y densidad vehicular.

En sus pruebas experimentales y simulaciones (15,000 escenarios), alcanzaron una precisión del 95% en el conteo vehicular y entre 96% y 97% en la medición de velocidad, logrando una reducción del tiempo de espera en un 18.3% en comparación con un ciclo semafórico fijo. La propuesta introduce cinco niveles de congestión (muy baja, baja, media, alta y muy alta), permitiendo un ajuste más preciso de los tiempos de luz.

Este trabajo resulta relevante para la presente investigación, pues comparte el objetivo de optimizar los ciclos semafóricos mediante visión computacional y algoritmos adaptativos, aunque difiere en la técnica y tecnología empleada: mientras en el antecedente se utiliza YOLOv4 con medición de velocidad, la presente investigación emplea YOLOv8 con un enfoque en la ocupación por bandas y la estimación de demanda vehicular, orientado a la implementación en hardware embebido de bajo costo.

II.2.2. Antecedentes a nivel internacional

En este aspecto, tomamos como referencia la investigación de Sosa (2020), tesis colombiana que fue elegida a criterio propio no solo por el hecho de que compartimos la misma problemática, ya que posee la segunda ciudad con más tráfico (Moneybeam, 2020), sino también por el hecho de encontrar solución a uno de los inconvenientes del segundo antecedente nacional.

En su descripción del problema el investigador tomo en cuenta el pasado de su país frente a esta problemática, exponiendo la medida llamada “pico y placa” la cual se basa en el libre tránsito de ciertos vehículos por el número inicial de su placa, medida que al contrario de ayudar con el tiempo provoco una subida en el número del parque automotor, es debido a esta ultimo problema y también al incremento de la población

en Bogotá, es que el autor plantea que el sistema de semáforos deba adaptarse, dando su alcance en las partes más críticas de la ciudad.

En sus objetivos esta la implementación del software de comunicación entre el sistema de monitoreo y servidores, también el planteamiento de variables necesarias para procesar imágenes. Su sistema se basa en el reconocimiento por cámara de video de la cantidad de carros y así determinar por medio de un controlador en el semáforo el tiempo de cambio, el investigador reconoció alternativas, falencias y soluciones de su sistema, una vez optado por su sistema de reconocimiento, se estudió las diferentes formas de procesar la información dejando en claro que la variable de entrada sería el tráfico y la variable de salida el tiempo, en el aspecto de reconocimiento para su efectividad el investigador optó por los servicios de otra empresa, en este caso Amazon, la cual le garantizaba un 100% de precisión en el reconocimiento de vehículos. A lo que podemos deducir que es posible dar con soluciones alternativas, incluso si estas vienen de terceros, y así solo se aislaría el único problema que implica la iluminación de la noche para el correcto procesamiento de imágenes.

También podemos encontrar soluciones acondicionando partes de la vía en la tesis de Morales y González (2013), realizada en Venezuela. El autor plantea el problema a partir de las causas que genera el tráfico; accidentes automovilísticos, largos tiempos de espera para poder desplazarse en la ciudad y las altas cantidades de emisión de CO₂ por los vehículos parados, y a partir de esto proponiendo las incógnitas del cómo desarrollar el sistema, cuales serían los principales requerimientos y como estaría estructurado el sistema, planteándose como objetivos el análisis del funcionamiento actual de los semáforos, establecer los requerimientos técnicos del sistema y establecer una arquitectura definida para el sistema, este consta de sensores instalados en la berma que envían una señal al controlador Arduino de cuantos vehículos ya han pasado por la vía así determinando la cola vehicular, es a partir de esta información que el controlador define que tan prolongado debe ser el tiempo de luz verde, un sistema muy similar al modelo SCOOT mencionado anteriormente, con distintos diseños para cada tipo de vía, intersección sencilla, intersección con desvíos y entre muchas más.

A partir de su aplicación el autor destacó muchas diferencias entre el sistema predecesor y el que implementó, en caso del tiempo que anteriormente era establecido por el fabricante, en su sistema era variable y en la mayoría de casos reducidos, otra

diferencia es en las fases de cambio en su sistema a menos que ocurra un evento (el paso de un vehículo), el semáforo se mantendrá en rojo lo cual permite reducir accidentes, finalmente se detectó que como resultado de sus sistema es la prioridad que existe hacia las vías con mayor flujo de tráfico y una ventaja principal de este sistema es su adaptabilidad y mejoramiento, al que se le podría ir agregando más utilidades a su controlador, como cámaras para la seguridad o carteles led para en caso de siniestros o reparaciones de la vía, se pueda dar las correctas indicaciones de desviación.

Un referente importante a nivel internacional es la tesis de Castro (2022), este proyecto plantea un sistema conformado por un módulo de visión por computador y un módulo de control del semáforo, programados en Python con el uso de la librería OpenCV para la detección de vehículos en tiempo real. La propuesta implementa un conteo vehicular bidireccional en intersecciones, priorizando el flujo con mayor número de vehículos y ajustando dinámicamente el tiempo de cambio de luces. La data obtenida se aloja en una base de datos PostgreSQL para su estudio. La investigación tiene como objetivo solucionar el problema común de los semáforos convencionales con ciclos fijos que no consideran la densidad del tráfico, lo que ocasiona demoras y congestión. Para su desarrollo, se utilizó Raspberry Pi, cuidando aspectos como la ubicación de la cámara para optimizar la visibilidad y la iluminación, también la sección de hardware con suficiente capacidad de procesamiento.

Este antecedente es relevante para la investigación porque está enfocado en un contexto urbano que es Quito, demuestra la viabilidad de integrar visión artificial y procesamiento embebido para la gestión adaptativa del tráfico. Además, sus resultados permiten identificar retos similares a los de nuestro entorno, como la posición de la posición más óptima de la cámara, disminuir gastos y preservar la funcionalidad en tiempo real sin depender de una red semafórica centralizada.

II.2.3. Sistemas usados en otras ciudades

Guadalajara, México.

En esta ciudad se implementó una red integral de transporte público de gran capacidad, basada en ejes troncales que facilitan la movilidad de pasajeros a través de distintos

modos de transporte. Como parte de este sistema, se inauguró el Centro de Control Vial, que gestiona más de 700 intersecciones mediante una red computarizada de semaforización apoyada por cámaras de vigilancia (Gobierno del Estado de Jalisco, 2020).

Bogotá, Colombia.

La capital colombiana incorporó 25 puntos de semaforización inteligente, equipados con detectores ubicados en los accesos viales, alrededor de 30 metros antes de la línea de detención. Estos dispositivos permiten ajustar el tiempo del semáforo: si se registra un flujo alto de vehículos, se extiende la luz verde, y si no se detecta demanda, se otorga prioridad mínima al sentido contrario o a los peatones (Gobierno de Bogotá, 2016).

Medellín, Colombia.

En Medellín se consolidó un sistema de movilidad urbana inteligente, que articula cámaras de foto detección, sistemas de monitoreo, semáforos automatizados y sensores de velocidad, con el fin de optimizar la regulación del tránsito y la seguridad vial (Gobierno de Medellín, 2016).

II.2.3.1. Sistema usado en Lima

En Lima, la gestión del tránsito está a cargo de PROTRANSPORTE, que a través de su equipo de monitoreo y control administra la Red Semafórica Centralizada. Este sistema tiene como objetivo mejorar la regulación y supervisión del tráfico en toda el área metropolitana.

Para ello, dispone de 196 cámaras tipo DOMO, encargadas de vigilar las principales intersecciones, además de 1,520 cámaras de conteo vehicular que permiten estimar la demanda de los usuarios. Asimismo, cuenta con 13 estaciones de trabajo dedicadas al control del tránsito y una red de aproximadamente 200 km de fibra óptica, la cual enlaza los semáforos con el Centro de Control para su operación en tiempo real (Municipalidad de Lima, 2017).

II.3 Bases teóricas

II.3.1 Normativa vial

La implementación de un sistema de control semafórico, incluso en entornos de investigación y desarrollo, debe cumplir con los lineamientos técnicos establecidos en la normativa vigente para garantizar la seguridad vial y la compatibilidad con la infraestructura existente. En el Perú, el *Manual de Dispositivos de Control de Tránsito Automotor para Calles y Carreteras* (MTC, 2016) constituye la referencia oficial para el diseño, instalación y operación de semáforos.

Este manual establece que los semáforos deben ubicarse de manera que sean claramente visibles para los usuarios de la vía, con una altura y orientación que aseguren la percepción oportuna de las señales luminosas. Asimismo, define las características técnicas de las luces, como la intensidad luminosa, los colores estandarizados (rojo, ámbar y verde) y los tiempos mínimos recomendados para cada fase, con el fin de garantizar un tiempo adecuado de reacción para conductores y peatones.

En cuanto a la secuencia de operación, el manual especifica que el orden de cambio de luces debe seguir la lógica internacionalmente aceptada: verde → ámbar → rojo, y que el tiempo de luz ámbar debe permitir que los vehículos que se aproximan puedan detenerse de manera segura antes de la línea de pare. Además, indica que la duración del verde y del rojo debe ser calculada en función de factores como el volumen de tráfico, la velocidad de aproximación y el ancho de la intersección.

El sistema desarrollado en este proyecto se diseña respetando dichas disposiciones, asegurando que las fases semafóricas generadas por el algoritmo de control adaptativo se mantengan dentro de los parámetros establecidos en el manual, particularmente en lo referente a los tiempos mínimos de cada luz y a la secuencia de cambio. De esta manera, la propuesta combina el uso de visión artificial y control inteligente con el cumplimiento estricto de la normativa técnica vigente en el país.

II.4 Definición de términos

II.4.1 Factores en un sistema de semáforos accionados por el tránsito

Volumen de vehículos: en aquellas intersecciones donde el flujo no justifica la colocación de semáforos de tiempo fijo, se puede optar por los de control vehicular siempre que existan otras condiciones que sustenten la inversión, aunque suelen ser más costosos.

Movimiento transversal: cuando la vía principal concentra gran cantidad de vehículos y obstaculiza el paso de la vía secundaria, los semáforos accionados por el tránsito resultan adecuados. No obstante, si el tránsito en la vía secundaria también es elevado, debe limitarse el tiempo de luz verde asignado para evitar demoras excesivas.

Horas de máxima demanda: este tipo de semáforos es útil en horarios específicos de gran congestión, ya que permiten regular de forma eficiente las intersecciones, evitando los inconvenientes de los semáforos de tiempo fijo en periodos de menor afluencia.

Peatones: son recomendables en lugares donde el volumen de peatones es reducido, puesto que solo interrumpen el flujo vehicular cuando alguien solicita el cruce.

Accidentes: en intersecciones con antecedentes de siniestralidad, incluso si no alcanzan los niveles exigidos para instalar semáforos fijos, se puede considerar este tipo de control. Sin embargo, siempre debe realizarse un análisis detallado para garantizar su efectividad.

II.4.2 Sensores

Los sistemas de semaforización inteligente emplean diversos sensores para detectar la presencia y el flujo vehicular:

- Sensores de presión: se instalan bajo la superficie de la vía y funcionan cuando las llantas de un vehículo ejercen presión sobre ellos, cerrando un circuito que confirma su presencia. Existen dos tipos: los que registran vehículos en una sola dirección y los que lo hacen en ambos sentidos.

- Sensores magnéticos: se colocan también bajo el pavimento y pueden ser compensados o no compensados. Los primeros ofrecen mayor precisión y no se ven afectados por tranvías o trolebuses, mientras que los segundos tienen un área de influencia menos definida y son susceptibles a interferencias electromagnéticas.
- Sensores de radar: operan mediante un emisor de microondas, usualmente montado en un poste de alumbrado. Al pasar un vehículo, parte de la señal se refleja y es recibida por una antena que transmite la información al controlador, registrando así el tránsito.
- Sensores de video: utilizan cámaras que capturan imágenes en tiempo real sobre un área amplia de la intersección. Estas imágenes son procesadas y enviadas al sistema de control, lo que permite ajustar dinámicamente las fases del semáforo de acuerdo a la presencia y volumen de vehículos.

CAPÍTULO III: HIPÓTESIS Y VARIABLES

III.1 Hipótesis

III.1.1 Hipótesis general

La implementación de un sistema de semáforos autónomos con visión artificial y control adaptativo, basado en la detección en tiempo real del flujo vehicular, optimiza la fluidez del tránsito en la intersección de las avenidas Faucett con Venezuela, reduciendo los tiempos de espera y optimizando la asignación de fases semafóricas.

III.1.2 Hipótesis específicas

La detección de vehículos mediante visión artificial permite estimar con precisión la demanda de paso en cada vía.

El ajuste dinámico de los tiempos de luz verde y roja, en función de la demanda detectada, reduce el tiempo promedio de espera por vehículo.

El sistema se adapta a variaciones de flujo vehicular en diferentes horarios, manteniendo la eficiencia en la regulación del tránsito.

III.2. Variables

- Variable independiente: flujo vehicular detectado en tiempo real mediante visión artificial.
- Variable dependiente: tiempo de luz verde del semáforo.

- Variables de control: ubicación fija de la cámara, condiciones de simulación (videos pregrabados), algoritmo YOLOv8, Raspberry Pi 5.

III.3. Operacionalización de las variables

Tabla 4

Variables del proyecto

Variable	Dimensión	Indicador	Escala/Unidad	Técnica de dimensión
Flujo Vehicular	Niveles de demanda	N.º de vehículos detectados	Conteo / vehículos	Procesamiento de video
Tiempo de luz verde	Duración por fase	Segundos asignados por vía	Segundos (s)	Registro del algoritmo

CAPÍTULO IV: MATERIALES Y METODOS

Se procederá al diseño y simulación del sistema de semáforos autónomos con visión artificial propuesto. Se presentan las actividades planificadas para su ejecución, los recursos necesarios para su implementación y el procedimiento seguido para integrar y validar cada uno de sus componentes. El enfoque se centra en lograr un funcionamiento óptimo en tiempo real, manteniendo criterios de eficiencia, bajo costo y escalabilidad para su posible aplicación en otras intersecciones.

IV.1. Área de estudio

La zona de estudio se encuentra en el cruce entre la Avenida Venezuela y la Avenida Faucett, el cual pertenece a la provincia constitucional del Callao. Se considera que esta intersección es uno de los lugares más congestionados en la zona, debido a su elevado tránsito de vehículos y peatones, ya que es una vía directa de conexión entre el puerto del Callao, el aeropuerto internacional Jorge Chávez y Lima. Las coordenadas geográficas de la intersección son: latitud -12.062505644792141 y longitud -77.09709510742262.

Se eligió este lugar porque muestra circunstancias verdaderas de saturación vehicular y largos períodos de espera, lo cual lo hace un ambiente propicio para examinar la viabilidad y el funcionamiento del sistema propuesto de semáforos autónomos con visión artificial.

IV.2. Diseño de investigación

El diseño de la investigación es experimental, ya que se manipularán variables relacionadas con el control semafórico, como la duración de las luces, para observar

sus efectos sobre el flujo vehicular en un entorno controlado mediante simulación y pruebas con un prototipo. Se realizarán pruebas repetidas bajo diferentes configuraciones para asegurar la consistencia y confiabilidad de los resultados. Asimismo, es de carácter descriptivo, debido a que se documentará detalladamente el diseño, implementación y funcionamiento del sistema de semáforos autónomos, así como los hallazgos obtenidos a partir del análisis de los datos recolectados en las pruebas.

IV.3. Población y muestra

IV.3.1. Población

La población objetivo está constituida por el flujo vehicular que circula por la intersección de las avenidas Faucett con Venezuela, en el distrito de Bellavista, Callao. Esta población incluye todos los vehículos que transitan por las cuatro vías de acceso a la intersección durante diferentes horarios y condiciones de tráfico

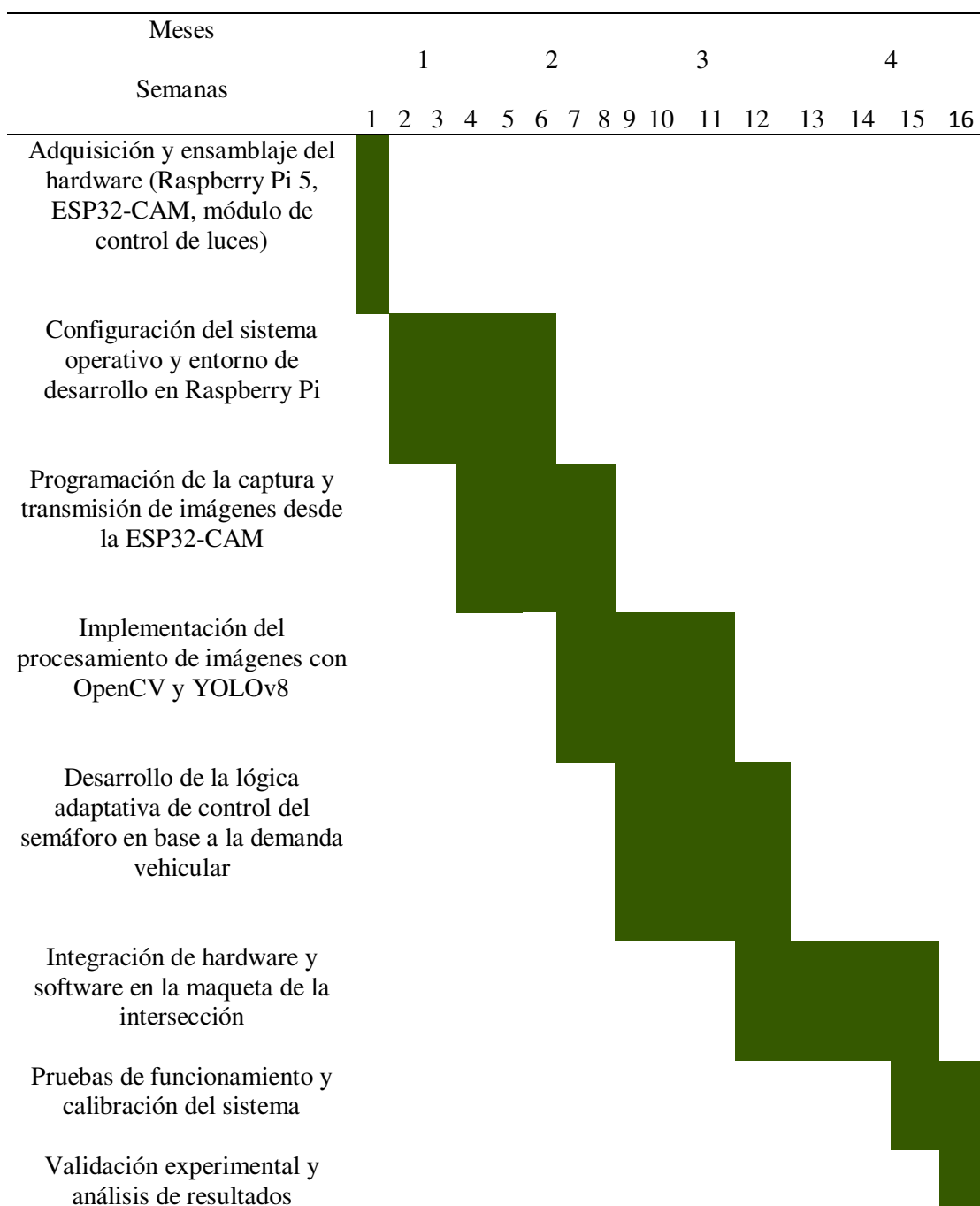
IV.3.2. Muestra

La muestra corresponde a los escenarios de tráfico simulados y las pruebas realizadas con el prototipo del sistema de semáforos autónomos. Estos escenarios se han diseñado en base a datos observados y registros de conteo vehicular, representando condiciones de alta, media y baja demanda de tránsito. La selección de la muestra es no probabilística por conveniencia, debido a que se eligen las condiciones de prueba en función de su relevancia para evaluar el desempeño del sistema y la factibilidad de su implementación.

IV.4. Procedimientos, técnicas e instrumentos de recolección de información

IV.4.1. Cronograma del proyecto

En la siguiente tabla se detalla el cronograma del diseño mediante el cual se llevó a cabo el trabajo de investigación, respetando todos los puntos correspondientes

Tabla 5*Cronograma de actividades del proyecto***IV.4.2. Presupuesto referencial**

El presupuesto estimado contempla los recursos necesarios para la implementación del sistema de semáforos autónomos con visión artificial en un entorno real. No se considera la adquisición de carcazas, estructuras o postes, ya que estos elementos ya están instalados en la intersección seleccionada. Los costos corresponden a precios

promedio del mercado local y se orientan a obtener una solución funcional y replicable.

Tabla 6

Presupuesto del prototipo

Descripción	Cantidad	Precio Unitario(s/)	Subtotal(s/)
Raspberry Pi 5 (4 GB RAM)	1	480.00	480.00
Módulo cámara ESP32-CAM	4	55.00	220.00
Módulo de relés de 4 canales (para control de luces)	1	35.00	35.00
Fuente de alimentación industrial 5V/3 ^a	1	100.00	100.00
Tarjeta microSD 64 GB Clase 10	1	35.00	35.00
Cableado eléctrico y conectores industriales	-	-	150.00
Mano de obra técnica (instalación e integración)	-	-	1000.00
Total			2020.00

IV.4.3. Tecnología a emplearse en la solución

Para la creación del prototipo del sistema de semáforos autónomos con visión artificial, se elegirán herramientas tecnológicas que aseguren una operación eficaz, segura y asequible en costos. Se ha hecho la selección del hardware y software teniendo en

cuenta aspectos como la habilidad para procesar en tiempo real, la sencillez de integración, la compatibilidad con librerías de visión artificial y el uso de energía.

En las secciones siguientes se describen los elementos físicos (hardware) y los programas lógicos (software) que constituyen la solución sugerida, además de su papel concreto en el sistema y la razón por la que se opta por ella en comparación con otras opciones.

IV.4.3.1. Hardware

El hardware seleccionado para la implementación del sistema propuesto debe cumplir con los requisitos de captura de imágenes en tiempo real, procesamiento eficiente de datos y control autónomo del semáforo.

Además, debe presentar un bajo consumo energético, alta disponibilidad en el mercado y un costo accesible para permitir su replicabilidad.

IV.4.3.1.1. Cámara ESP32-CAM

El módulo Esp32-Cam se estructura por un microcontrolador ESP32 y un sensor OV2640, el cual captura imágenes de hasta 1600x1200 píxeles (Espressif Systems, 2023).

Dado su bajo consumo de energía, tamaño pequeño y bajo costo, se vuelve una opción ideal para nuestro prototipo, el cual tomara imágenes del flujo vehicular para su procesamiento.

IV.4.3.1.2. Procesador Principal: Raspberry Pi 5

El microcontrolador Raspberry Pi 5 es una computadora de placa única equipada con un procesador ARM Cortex-A76 de cuatro núcleos de 2.4 GHz, GPU VideoCore VII y en este caso, con 4GB de memoria RAM (Raspberry Pi Foundation, 2023).

Esta capacidad de procesamiento permite ejecutar modelos de inteligencia artificial, como YOLOv8 en tiempo real, procesando imágenes recibidas desde la cámara y

gestionando los tiempos de las luces del semáforo. Su tamaño compacto y compatibilidad con periféricos la convierte en el componente corazón del prototipo.

IV.4.3.1.3. Módulo de interfaz para control de luces

Aunque el Raspberry Pi 5 puede generar las señales de control directamente, se requiere un módulo de interfaz, como un conjunto de relés para manejar el voltaje y corriente necesarios para las lámparas del semáforo. Este módulo actúa como puente entre las señales de bajo voltaje del Raspberry Pi y la alimentación de las luces, garantizando un funcionamiento seguro y fiable.

IV.4.3.1.4. Fuente de Alimentación y Accesorios

Todo el sistema se alimenta principalmente desde la Raspberry Pi 5, que a su vez provee energía a la ESP32-CAM y al módulo de control de luces (si es de bajo consumo). Se incluirán elementos de protección eléctrica, cables y soportes para la instalación segura de todos los componentes.

IV.4.3.2. Software

El software del sistema está compuesto por herramientas y librerías que permiten la captura, procesamiento y análisis de imágenes, así como la ejecución de la lógica de control adaptativo del semáforo.

La selección de cada recurso se ha realizado considerando criterios de compatibilidad con el hardware elegido, facilidad de desarrollo, rendimiento en tiempo real y disponibilidad de soporte.

IV.4.3.2.1. Sistema Operativo: Raspberry Pi OS

Se utiliza Raspberry Pi OS como sistema operativo principal por su compatibilidad nativa con la Raspberry Pi 5, estabilidad y soporte para entornos de desarrollo en Python.

Esta plataforma ofrece un entorno flexible y optimizado para aplicaciones de visión artificial y control de hardware mediante GPIO.

IV.4.3.2.2. Lenguaje de programación: Python

Este lenguaje fue seleccionado debido a su amplia documentación en visión artificial, uso comprensible y compatibilidad con librerías de control de hardware y comunicación.

IV.4.3.2.3. Librerías de Visión Artificial: OpenCV y Ultralytics YOLOv8

Open Source Computer Vision (OpenCV) es una librería usada para el procesamiento de imágenes, segmentación y manipulación de datos visuales. El modelo UltraLytics YOLOv8 se utiliza para la detección de vehículos en tiempo real, gracias a su precisión y eficiencia computacional

IV.4.3.2.4. Protocolos de Comunicación

Para el envío de imágenes entre la ESP32-CAM a la Raspberry Pi 5, este último usa la librería de Python llamada Flask para crear un servidor para la recepción de imágenes, entonces la ESP32-CAM usa protocolos HTTP que envía estos datos a través de la red local, para su procesamiento.

IV.4.3.3. Justificación de la elección de tecnologías

La selección del hardware y software del prototipo responde a criterios técnicos, económicos y viables operativamente, buscando un funcionamiento equilibrado. Viendo el lado del hardware se decidió la cámara ESP32-CAM como el dispositivo de captura de imágenes por su bajo costo, facilidad de implementar y conectividad wi-fi, lo que permite una instalación sencilla y escalable. En cuanto a la Raspberry Pi5 fue seleccionada por su capacidad de ejecutar modelos de inteligencia artificial en tiempo real, un bajo consumo de energía y compatibilidad con diferentes dispositivos. En el caso de software, se eligió Raspberry Pi OS por su estabilidad y optimización para la

arquitectura ARM, también se seleccionó Python por su extensa documentación en el campo del desarrollo. OpenCV y Ultralytics YOLOv8, por su rendimiento comprobado en aplicaciones de visión artificial en tiempo real, mientras paralelamente los servidores de Flask reciben imágenes de las cámaras.

Esta combinación tecnológica permite desarrollar un sistema autónomo, confiable y adaptable a diferentes entornos urbanos, cumpliendo con los requerimientos funcionales definidos en el diseño y respetando las restricciones de costo y consumo energético.

IV.4.4. Procedimiento experimental / metodología de implementación

El procedimiento de implementación del sistema de semáforos autónomos con visión artificial se desarrolló en una serie de etapas secuenciales, que abarcan desde la configuración inicial del hardware hasta la validación del sistema en condiciones controladas. Cada etapa fue planificada para garantizar un montaje seguro, un procesamiento de datos eficiente y un control confiable de las señales luminosas.

IV.4.4.1. Etapas de implementación

El desarrollo del prototipo se llevó a cabo en diferentes etapas, las cuales reflejan la evolución metodológica del proyecto. Inicialmente, se optó por un enfoque tradicional basado en modelos matemáticos y librerías clásicas de visión por computador (OpenCV). En esta primera fase se emplearon técnicas de sustracción de fondo (MOG2), rastreo por solapamiento de áreas (IoU) y análisis de bandas ponderadas para estimar la demanda vehicular. Este esquema permitió generar un control adaptativo básico, en el que la duración de la luz verde se ajustaba a tablas de demanda observada. Si bien el enfoque inicial mostró resultados aceptables en condiciones controladas, también evidenció limitaciones importantes frente a variaciones de iluminación, oclusiones entre vehículos y la imposibilidad de clasificar correctamente los distintos tipos de tránsito. Estos factores motivaron la migración hacia un segundo enfoque basado en redes neuronales convolucionales, específicamente con la implementación del modelo YOLOv8, el cual ofreció mayor robustez y precisión en la detección de vehículos y peatones.

De esta manera, el proyecto se consolidó en dos fases complementarias:

- una fase experimental inicial con algoritmos tradicionales, útil para validar la lógica del sistema y comprender la dinámica de control;
- y una fase final con visión artificial avanzada, que permitió integrar el prototipo de manera más estable en el hardware embebido (Raspberry Pi 5 y ESP32-CAM).

A continuación, se describen las etapas específicas de implementación, considerando tanto la evolución de los enfoques como los procedimientos técnicos realizados en cada una de ellas.

IV.4.4.1.1 Fase inicial: Enfoque tradicional con OpenCV y modelos matemáticos

En la etapa inicial del proyecto se implementó un sistema de control semafórico autónomo empleando exclusivamente técnicas de visión por computador basadas en OpenCV y algoritmos matemáticos de rastreo. El objetivo de esta fase era validar la lógica de adaptación de tiempos de luz sin recurrir aún a modelos de inteligencia artificial de alto costo computacional.

El procedimiento se organizó en tres componentes principales:

Segmentación de movimiento, se utilizó el algoritmo de sustracción de fondo MOG2, que permite distinguir objetos móviles respecto al entorno estático. Posteriormente, se aplicó una binarización y filtrado de contornos, reteniendo únicamente aquellos con un área mayor a un umbral (≈ 1130 píxeles) para descartar ruido, sombras y objetos irrelevantes.

Rastreo de vehículos mediante solapamiento (IoU), los contornos filtrados se asociaron frame a frame mediante el cálculo del índice de solapamiento (Intersection over Union – IoU). Cada vehículo detectado fue asignado a un identificador único (ID), actualizando su posición y dirección de movimiento. Este rastreo permitió contabilizar objetos de manera persistente y registrar su trayectoria dentro de la zona de interés definida.

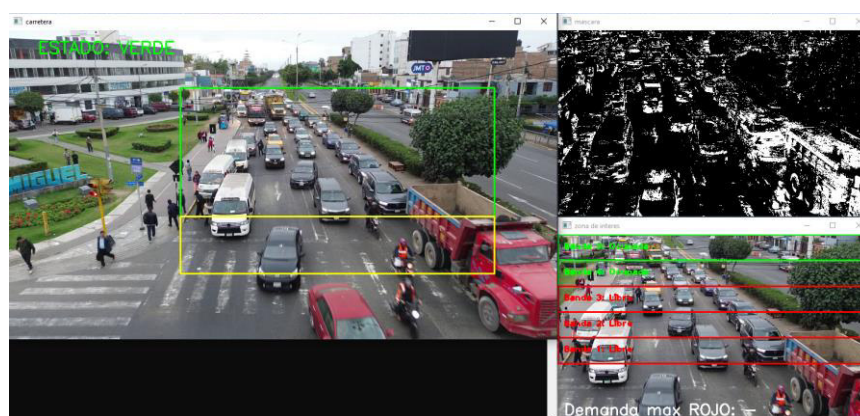
Estimación de demanda vehicular por bandas, la región de interés (ROI) se dividió en cinco bandas horizontales, cada una con un peso decreciente conforme se alejaba de la línea de detención. De este modo, la presencia de vehículos en las bandas inferiores (más cercanas al cruce) incrementaba la demanda ponderada en mayor medida. Durante la fase de luz roja, se almacenaba el valor máximo de demanda registrado, el cual posteriormente era utilizado para asignar el tiempo de luz verde.

Finalmente, la lógica de control se resolvió a través de una máquina de estados simple:

- Si un objeto en movimiento era detectado en la zona de cruce peatonal (zebra), el sistema ordenaba la transición de ROJO a VERDE.
- El tiempo de luz verde se definía según una tabla preestablecida de demanda (20, 30, 40 o 50 segundos).
- Concluido este intervalo, el sistema regresaba al estado de ROJO, reiniciando el proceso de medición.

Figura 7

Prototipo Inicial implementado con OpenCV: estado del semáforo, zona de interés, mascara de sustracción de fondo y bandas de demanda



Nota. Elaboración propia.

Este enfoque inicial resultó útil para comprobar la factibilidad de un control dinámico sin intervención humana. Sin embargo, durante las pruebas se identificaron limitaciones importantes:

- Sensibilidad a cambios de iluminación y condiciones ambientales.
- Pérdida de seguimiento en casos de oclusión o tránsito denso.
- Imposibilidad de diferenciar entre tipos de vehículos (autos, buses, motos).

Estos factores motivaron la transición hacia un enfoque más robusto basado en modelos de detección por aprendizaje profundo, que se describe en la siguiente sección.

IV.4.4.1.2. Fase final: Enfoque con redes neuronales (YOLOv8 + OpenCV)

Tras las limitaciones observadas en la fase inicial, se decidió migrar hacia un enfoque de visión artificial basado en redes neuronales convolucionales, con el objetivo de incrementar la robustez y precisión en la detección vehicular. Para ello se empleó el modelo YOLOv8 (You Only Look Once, versión 8), que destaca por su capacidad de realizar detecciones en tiempo real con alta exactitud y bajo costo computacional relativo frente a otros modelos de su categoría. a) Integración con OpenCV. El modelo YOLOv8 se integró a la lógica de procesamiento mediante la librería OpenCV, encargada de la captura de imágenes y la superposición de resultados en tiempo real. OpenCV también se empleó en esta fase para operaciones complementarias como el redimensionamiento de frames, delimitación de zonas de interés (ROI) y visualización del estado del semáforo.

Detección y clasificación de vehículos:

A diferencia del enfoque tradicional, el modelo YOLOv8 permite no solo identificar objetos en movimiento, sino clasificarlos según su categoría (automóviles, motocicletas, camiones, autobuses, peatones, etc.). Esta capacidad ofrece una visión más completa de la dinámica del cruce, diferenciando el tipo de tránsito y otorgando una base más sólida para la toma de decisiones del controlador semafórico.

Optimización para hardware embebido:

Dado que la implementación final se realizó en una Raspberry Pi 5 en conjunto con una cámara ESP32-CAM, se aplicaron diversas estrategias de optimización:

- Reducción de la resolución de entrada (inferencias en 640×480 px).

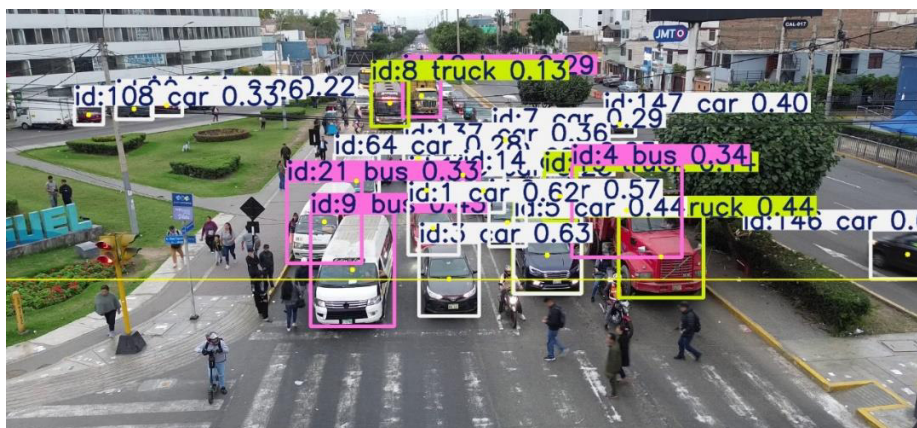
- Procesamiento en intervalos de frames, evitando correr detección en cada uno de ellos para ahorrar recursos.
- Configuración de hilos y uso eficiente de la CPU de la Raspberry Pi. Estas optimizaciones permitieron mantener un equilibrio entre velocidad de respuesta y precisión en las detecciones.

Lógica de control adaptativo con detecciones YOLO:

La salida del modelo YOLOv8 fue empleada como insumo para el cálculo de demanda vehicular en cada vía. En esta etapa se redefinió el criterio de ocupación de carriles, basándose en el conteo de vehículos detectados y su distribución en la zona de espera antes del cruce. El controlador adaptativo ajustó la duración del verde considerando no solo la cantidad de vehículos, sino también la tipología detectada (por ejemplo, otorgando mayor peso a la presencia de vehículos pesados).

Figura 8

Resultados de detección con YOLOv8 en el prototipo



Nota. Elaboración propia.

Ventajas frente al enfoque inicial:

La adopción de YOLOv8 trajo consigo beneficios significativos:

- Mayor precisión en la detección bajo diferentes condiciones de iluminación y clima.

- Reducción de falsos positivos frente a sombras y movimientos de peatones en zonas no relevantes.
- Capacidad de clasificación multicategoría, lo cual permite plantear reglas de control más avanzadas.
- Mejor adaptabilidad del sistema a escenarios reales, en contraste con el enfoque basado únicamente en contornos y análisis de bandas.

En resumen, esta fase representó la consolidación del proyecto, integrando un sistema de detección robusto y escalable que puede operar en condiciones más cercanas al entorno real del tráfico urbano limeño. La combinación de YOLOv8 y OpenCV permitió superar las limitaciones del enfoque tradicional, sentando las bases de un prototipo de semáforo inteligente con visión artificial plenamente funcional.

En conclusión, la transición desde un enfoque tradicional basado en OpenCV hacia un enfoque moderno con YOLOv8 representó un punto de inflexión en el desarrollo del sistema. Mientras que la primera fase permitió validar la factibilidad de un control adaptativo empleando algoritmos ligeros, la segunda ofreció la solidez y escalabilidad necesarias para un prototipo viable en condiciones reales de tráfico. De esta manera, la elección final de YOLOv8 no se limitó a sustituir técnicas previas, sino que constituyó un paso evolutivo sustentado en las pruebas experimentales y en la necesidad de contar con un sistema confiable y robusto.

IV.4.4.1.3. Configuración del entorno de desarrollo

Con el enfoque metodológico definido, la siguiente etapa consistió en preparar el entorno de desarrollo que serviría de base para la implementación del sistema. Esta configuración fue esencial para garantizar la compatibilidad entre los distintos componentes de hardware (Raspberry Pi 5 y cámara ESP32-CAM) y las librerías de software necesarias para la captura, procesamiento y análisis de imágenes. En esta sección se describen las herramientas utilizadas, la instalación de dependencias críticas (Python, OpenCV, Ultralytics YOLOv8) y la configuración del entorno virtual, así como la preparación de la Raspberry Pi como nodo central del sistema. Esta infraestructura permitió unificar tanto las pruebas iniciales con OpenCV como la

posterior integración con YOLOv8, asegurando reproducibilidad y estabilidad en las fases de experimentación.

- Instalación de Raspberry Pi OS en la tarjeta microSD.

Figura 9

Raspberry Pi5



Nota. Elaboracion propia.

- Configuración de red y acceso remoto a la Raspberry Pi.

Figura 10

Configuración del Access Point

```
gualas@semaforo-central: /usr/local/bin
File Edit Tabs Help
GNU nano 7.2          iniciar_ap.sh
#!/bin/bash

# Desbloquear WiFi
rfkill unblock wifi

# Activar wlan0 con IP
ifconfig wlan0 up
ifconfig wlan0 192.168.4.1 netmask 255.255.255.0

# Iniciar servicios
systemctl restart hostapd
systemctl restart dnsmasq

python3 /home/gualas/servidor.py &

exit 0

[ Read 16 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^N Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Nota. Elaboración propia.

- Instalación de librerías necesarias: OpenCV, Ultralytics YOLOv8, numpy, entre otras.

Figura 11

Librerías del entorno virtual

```
(yoloenv) gualas@semaforo-central:~/Videos $ pip3 list
Package            Version
-----
certifi            2025.7.14
charset-normalizer 3.4.2
contourpy          1.3.3
cyclor             0.12.1
filelock           3.18.0
fonttools          4.59.0
fsspec             2025.7.0
idna               3.10
Jinja2             3.1.6
kiwisolver         1.4.8
lap                0.5.12
MarkupSafe         3.0.2
matplotlib         3.10.5
mpmath             1.3.0
networkx           3.5
numpy              2.2.6
opencv-python     4.12.0.88
packaging          25.0
pandas            2.3.1
pillow            11.3.0
pip               25.2
psutil            7.0.0
py-cpuinfo        9.0.0
pyparsing         3.2.3
python-dateutil   2.9.0.post0
pytz              2025.2
PyYAML            6.0.2
requests          2.32.4
scipy             1.16.1
setuptools        66.1.1
six              1.17.0
sympy            1.14.0
torch            2.7.1
torchvision      0.22.1
tqdm             4.67.1
typing_extensions 4.14.1
tzdata           2025.2
ultralytics      8.3.171
ultralytics-thop 2.0.14
urllib3          2.5.0
```

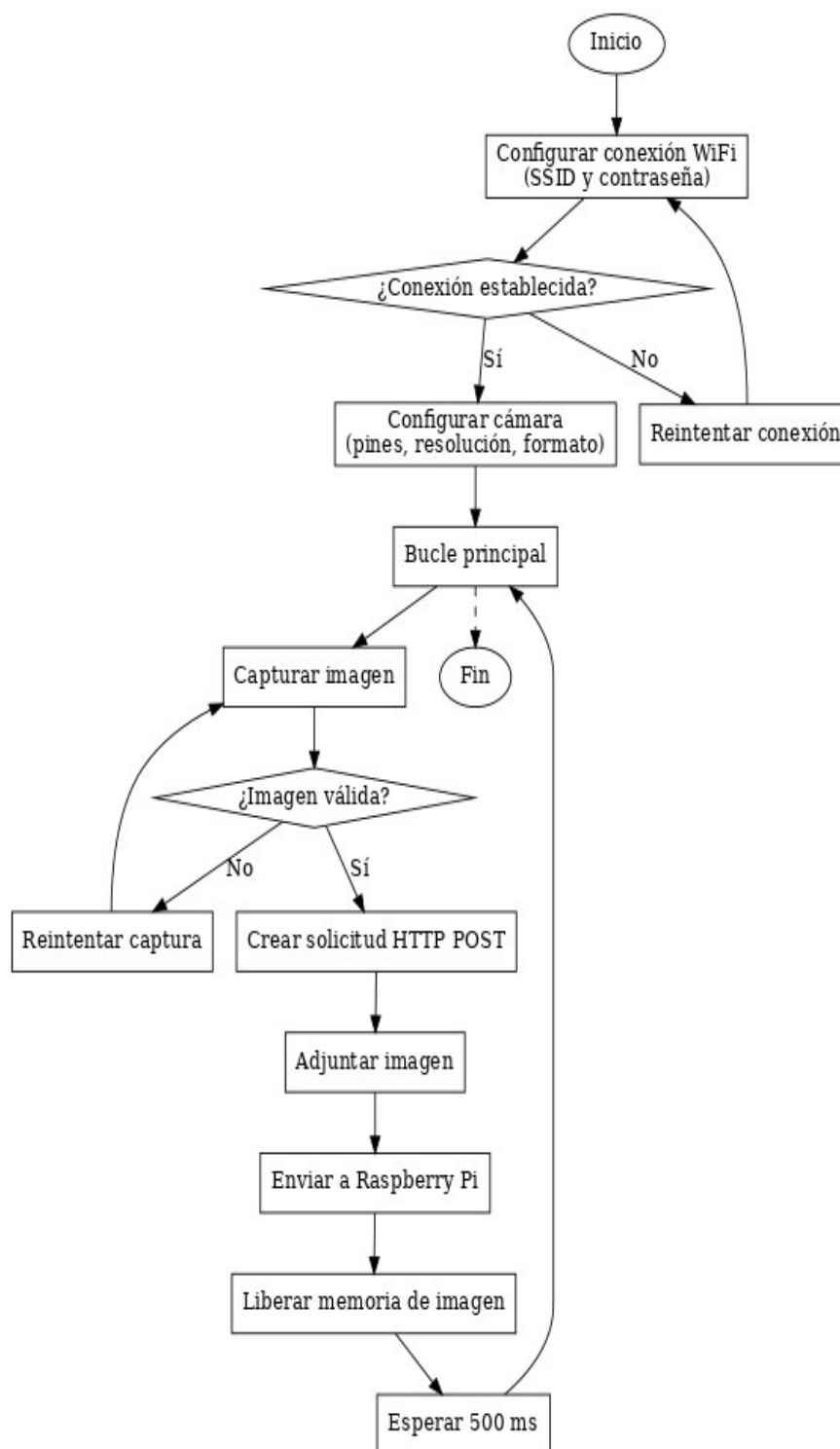
Nota. Elaboración propia.

IV.4.4.1.4. Desarrollo del software de captura y transmisión de imágenes

- Programación de las ESP32-CAM para capturar imágenes en tiempo real y transmitir las mediante protocolo HTTP a la Raspberry Pi.

Figura 12

Seudocódigo de Microcontrolador Esp32 Cam

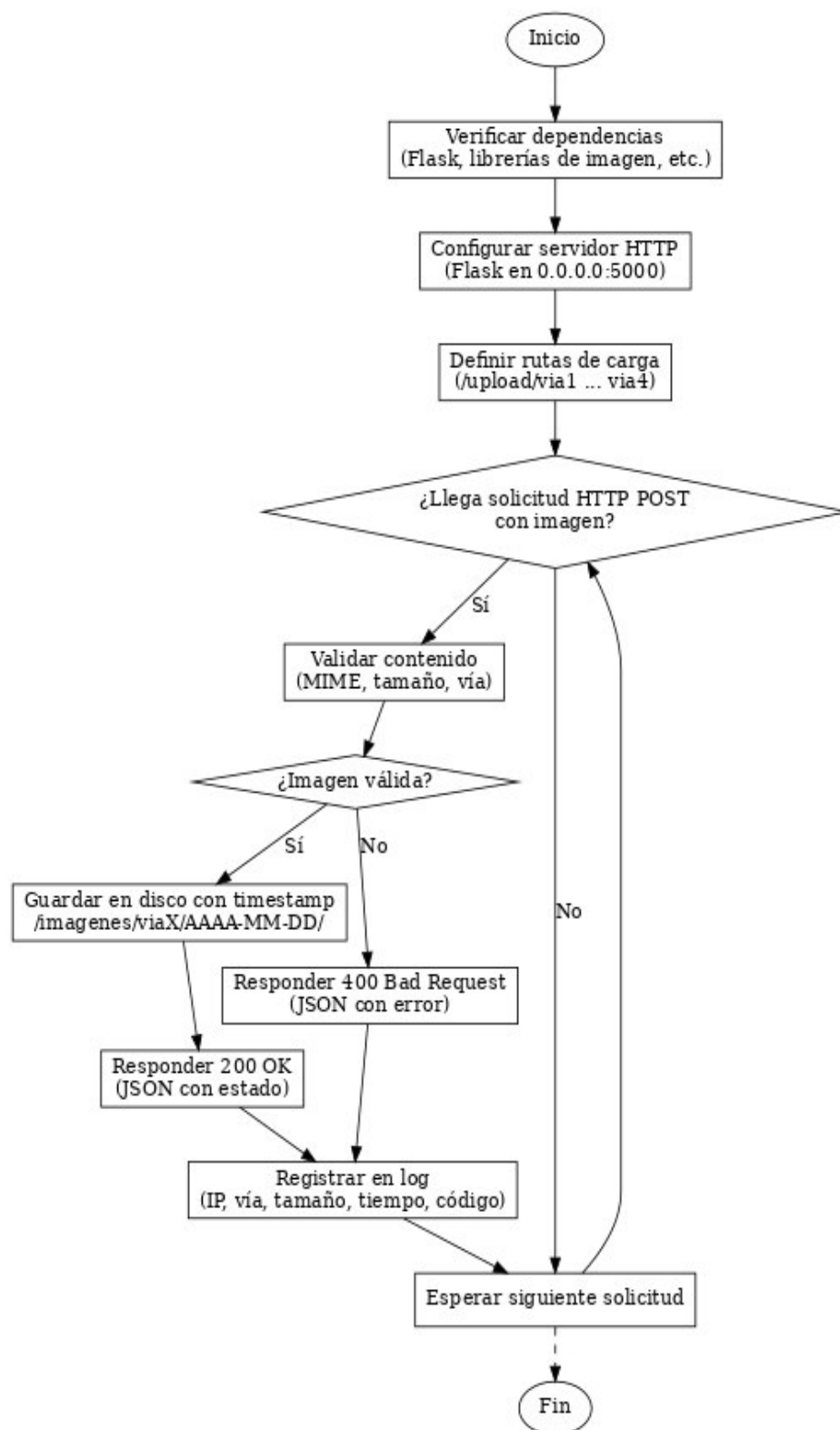


Nota. Elaboración propia.

- Configuración de rutas y servidores de recepción en la Raspberry Pi para almacenar o procesar directamente las imágenes recibidas.

Figura 13

Seudocódigo de Microcontrolador Raspberry Pi



Nota. Elaboración propia.

IV.4.4.1.5. Implementación del procesamiento y detección vehicular

- Integración de OpenCV para preprocesamiento de imágenes (redimensionado, corrección de color, recortes de región de interés).

Figura 14

Aplicación del cuadrante de interés

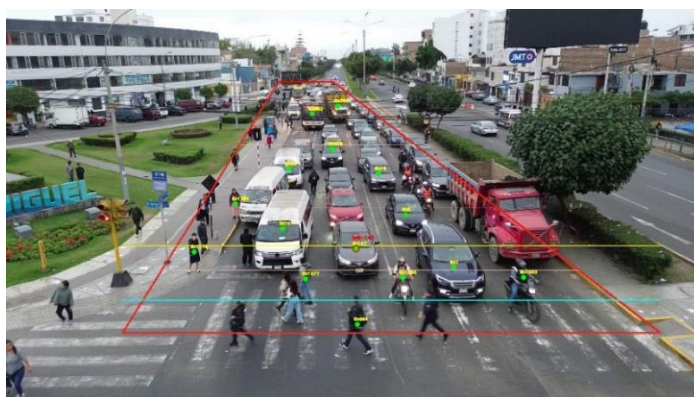


Nota. Elaboración propia.

- Uso de YOLOv8 para la detección y clasificación de vehículos en las imágenes capturadas.

Figura 15

Uso del modelo entrenado YOLO

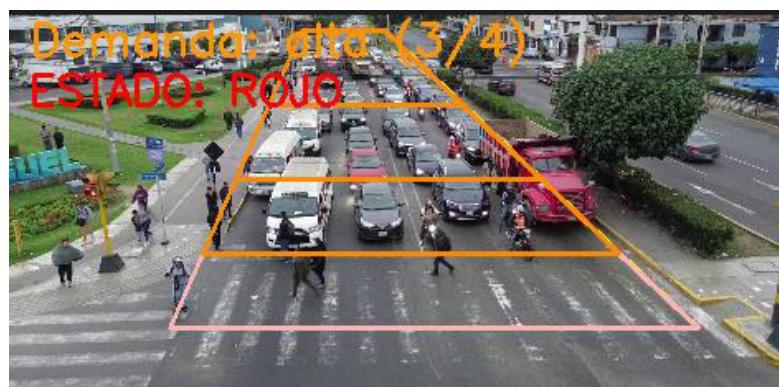


Nota. Elaboración propia.

- Cálculo de la demanda vehicular mediante análisis de áreas de ocupación en la vía.

Figura 16

Aplicación de lógica de demanda por áreas



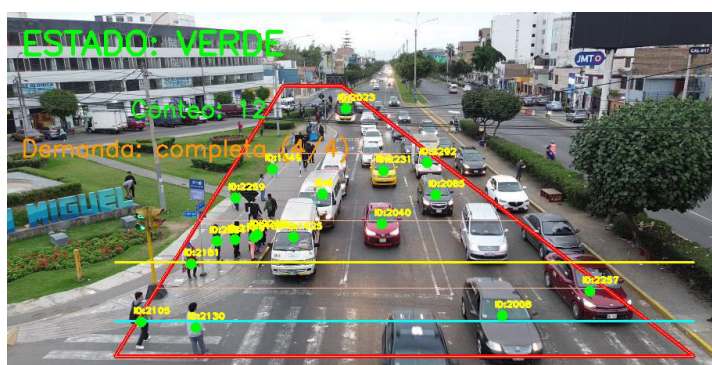
Nota. Elaboración propia. Demanda alta $\frac{3}{4}$ y Estado Rojo.

IV.4.4.1.6. Parámetros utilizados para el sistema

En el desarrollo del prototipo se tomó en cuenta el uso de distintos parámetros y pruebas, para verificar la certeza del modelo de aprendizaje YOLOv8 en nuestro sistema, como la identificación del estado activo, el conteo del registro de vehículos que cruzan la línea peatonal, la demanda que mide la proporción del área ocupada y los centroides que marcan la ubicación de cada vehículo o peatón. Estos parámetros permitirán que nuestro prototipo interprete correctamente la situación del tráfico.

Figura 17

Parámetros tomados en cuenta para el sistema (Autoría propia)



Nota. Elaboración propia. Estado Verde, Conteo 12 y Demanda $\frac{4}{4}$.

Tabla 7

Descripción de los parámetros utilizados por el sistema

Parámetro	Descripción
Estado	Luz en el que se encuentra el semáforo (Rojo o Verde)
Conteo	Número de vehículos que pasaron el paso de cebra
Demanda	Proporción del cuadrante de interés ocupado por vehículos
Centroides	Puntos de cada elemento, como vehículos o personas

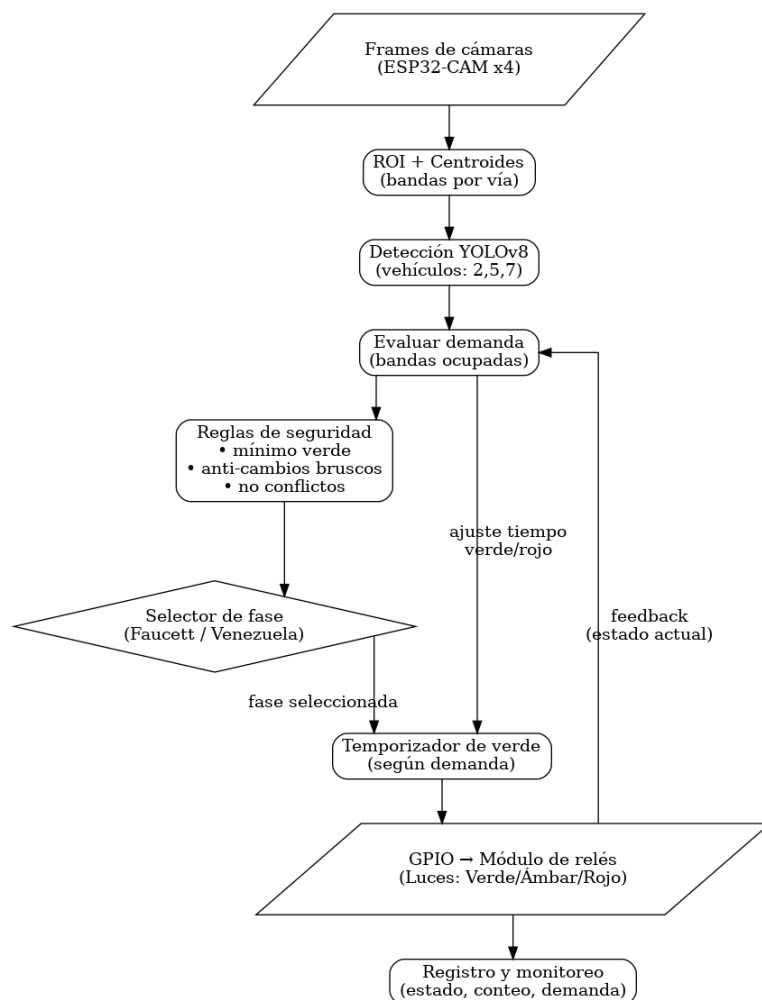
IV.4.4.1.7. Desarrollo de la lógica de control adaptativo

La lógica de control adaptativo implementada en el sistema se basa en el análisis continuo de imágenes capturadas por las cámaras ESP32-CAM y procesadas en la Raspberry Pi 5. Mediante el modelo YOLOv8, se identifican los vehículos presentes en áreas de interés (bandas de conteo) predefinidas para cada dirección de la intersección. El algoritmo calcula la demanda vehicular por vía a partir de:

- Conteo de vehículos detectados en cada banda activa.
- Estado de movimiento o detención (ocupación prolongada vs. paso fluido).
- Con esta información, el sistema ajusta dinámicamente los tiempos de luz verde y roja de cada fase semafórica.
- Si la demanda es alta y el flujo es continuo, se prolonga la luz verde hasta un máximo predefinido.
- Si se detecta congestión sin avance, se anticipa el cambio a rojo para dar paso a otra dirección.
- Si no hay demanda, se reduce el verde al mínimo y se prioriza otra vía.

Además, se incluyen reglas de seguridad para evitar cambios bruscos:

- Respeto de los tiempos mínimos de cada fase según normativa MTC.
- Evitar cambios consecutivos en menos de un umbral de segundos.
- Control coordinado de relés para impedir encendidos conflictivos.

Figura 18*Seudocódigo de la lógica del control adaptativo*

Nota. Elaboración propia.

IV.4.4.1.8. Pruebas y calibración del sistema

Las pruebas iniciales se realizaron en entorno controlado, utilizando videos reales de la intersección para simular el tráfico y afinar los parámetros:

- Ajuste de sensibilidad de detección en YOLOv8.
- Definición de coordenadas precisas para las bandas de conteo en cada cámara.
- Configuración de umbrales de demanda para cambio de fase.

Posteriormente, se realizaron pruebas con el prototipo en funcionamiento, conectando la Raspberry Pi 5 al módulo de relés para controlar el encendido de luces LED representativas de cada fase.

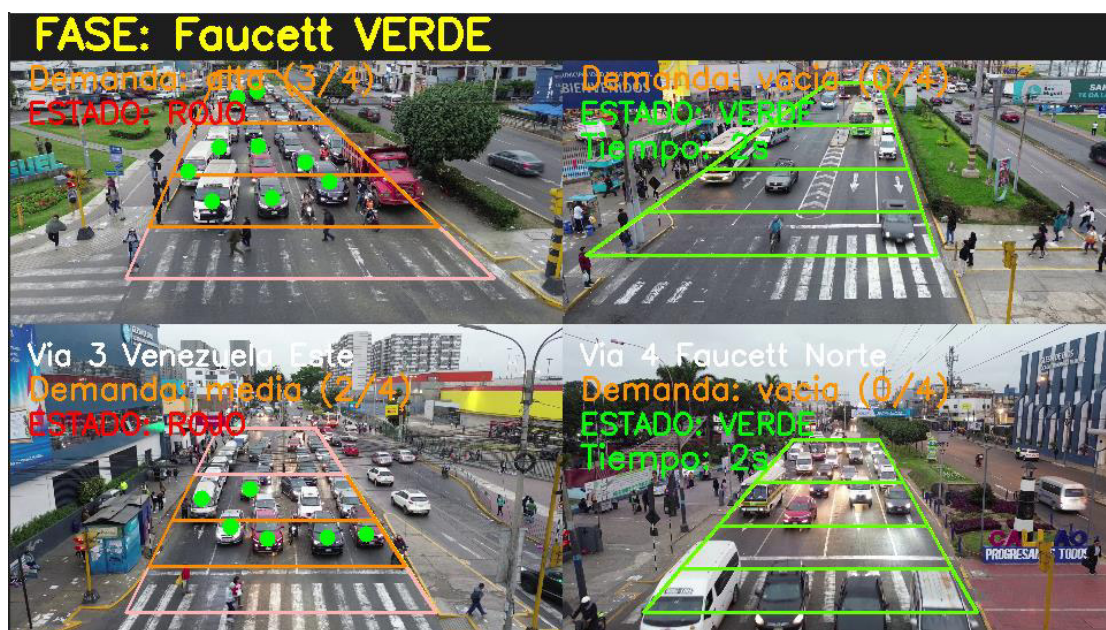
Durante esta etapa se verificó:

- Sincronización correcta entre detección y cambio de luces.
- Respuesta del sistema ante variaciones súbitas de demanda.
- Filtrado de falsas detecciones (vehículos fuera de la zona de conteo).

Los ajustes finales incluyeron la calibración fina de tiempos de ciclo y umbrales de cambio, buscando un equilibrio entre fluidez vehicular y seguridad en el cruce.

Figura 19

Visualización del prototipo en operación de demanda vehicular en las cuatro vías



Nota. Elaboración propia. Fase Faucett verde, Vía 1: demanda alta (3/4), estado rojo.

Vía 2: demanda vacía (0/4), estado verde (2 s). Vía 3: demanda media (2/4), estado rojo. Vía 4: demanda vacía (0/4), estado verde (2 s).

IV.5. Análisis estadístico

El análisis estadístico se realizará mediante técnicas descriptivas, comparando el desempeño del sistema semafórico actual de ciclo fijo con el sistema adaptativo propuesto. Para ello se tomarán como variables principales el tiempo promedio de espera por vehículo, el número de vehículos atendidos por ciclo y la relación entre demanda vehicular y tiempo de luz verde asignado.

Los datos se organizarán en tablas comparativas, el criterio de validación estará orientado a identificar la reducción de los tiempos de espera y la mejora en la eficiencia del flujo vehicular en la intersección estudiada.

CAPÍTULO V: RESULTADOS

V.1. Presentación y análisis de los resultados

Los resultados obtenidos durante la validación del prototipo de semáforo autónomo con visión artificial, así como la evaluación de su desempeño frente a un sistema de semafórico con un ciclo fijo, serán presentado para el análisis de sus métricas de funcionamiento y se discutirán el impacto del sistema adaptativo en la fluidez vehicular.

V.1.1. Metodología de evaluación

Las pruebas de evaluación se realizaron usando videos reales del tráfico de la intersección de las avenidas Faucett y Venezuela. Se trabajo con distintos tipos de material audiovisual, tanto como de autoría propia como dados por el municipio.

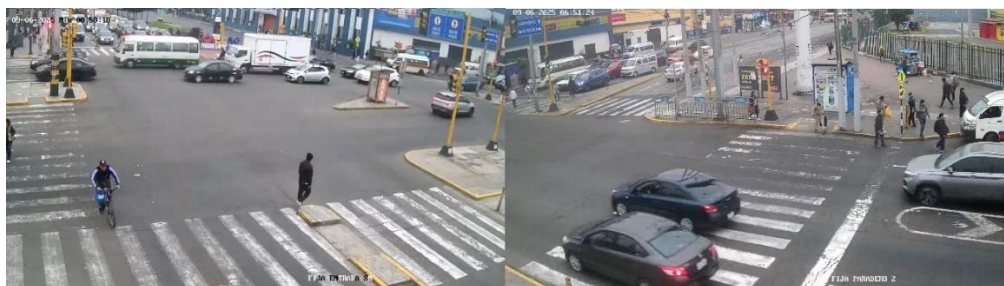
V.1.1.1. Sistema actual (ciclo fijo convencional)

Para obtener datos del semáforo en operación actualmente, se utilizaron:

- Videos de cámaras de vigilancia municipal proporcionados por la Municipalidad de San Miguel (Figura 20).
- Grabaciones propias realizadas desde ubicaciones elevadas, con autorización de entidades cercanas (Figura 21).

Figura 20

Videos del cruce



Nota. Imágenes proporcionadas por el *Municipio de San Miguel* (2025). Todos los derechos reservados por el Municipio de San Miguel.

Figura 21

Videos de Faucett



Nota. Elaboración propia.

Si bien estas grabaciones no corresponden al ángulo óptimo de observación para un sistema de visión artificial (montado en el propio semáforo), permitieron realizar un conteo manual de vehículos y registrar los tiempos efectivos de luz verde en cada fase (Figuras 20 y 21).

V.1.1.2. Sistema propuesto (ciclo adaptativo con visión artificial)

Para verificar el desempeño del prototipo, se elaboró un conjunto de 4 videos óptimos, cada uno correspondiente a una vía de la intersección. Estos videos fueron grabados con un dron, con el fin de obtener una perspectiva que simule la vista de una cámara instalada directamente sobre los semáforos, buscando la mejor vista del tráfico.

Figura 22

Videos tomados por dron



Nota. Elaboración propia.

Estas grabaciones permitieron al algoritmo de visión artificial detectar a los vehículos que ocupaban las diferentes bandas de detección, estimar la demanda y ajustar dinámicamente la duración de los tiempos en verde (Figura 22).

V.1.1.3. Escenario de análisis

Se analizaron segmentos de 5 minutos de duración bajo dos condiciones:

1. Ciclo fijo convencional (semáforo actual).
2. Ciclo adaptativo simulado (sistema propuesto).

En ambos escenarios se registraron:

- Número de vehículos que cruzaron en verde.
- Segundos totales de verde otorgados por vía.
- Flujo vehicular procesado (vehículos por segundo).
- Porcentaje de tiempo de verde sin demanda (cuando las vías estaban vacías).

V.1.2. Resultados experimentales

Los resultados obtenidos permiten comparar el desempeño del sistema convencional de ciclo fijo frente al sistema adaptativo propuesto con visión artificial. Para la evaluación se consideraron dos escenarios:

- Escenario 1: Sistema adaptativo con tiempos de luz verde iguales en todas las vías.
- Escenario 2: Sistema adaptativo con tiempos diferenciados según la vía, otorgando mayor duración a la avenida Faucett por su mayor capacidad y demanda.

En la Tabla 8 se muestran los resultados del Escenario 1. Se observa que el modelo adaptativo mejora el flujo vehicular en las vías correspondientes a la avenida Venezuela, donde con un tiempo verde más corto se logró un mayor aprovechamiento. Sin embargo, en las vías de la avenida Faucett, el desempeño se redujo, ya que el tiempo verde uniforme no compensó la mayor carga vehicular de estas arterias.

Tabla 8

Resultados comparativos entre ciclo fijo y adaptativo

Nota. Vía 1 Venezuela oeste, vía 2 Faucett sur, vía 3 Venezuela este y vía 4 Faucett norte.

Vía	Verde actual	Vehículos actuales	Flujo veh/s	Verde adaptativo	Vehículos adaptativos	Flujo veh/s	Variación (%)
1	65	47.5	0.73	50	46	0.92	+26%
2	140	79.5	0.57	50	43.5	0.87	+53%
3	65	40	0.62	50	46	0.92	+50%
4	140	85.5	0.61	60	51.5	0.86	+41%

Por otro lado, la Tabla 9 muestra los resultados del Escenario 2, en el cual se aplicaron tiempos diferenciados según la vía. En este caso, la avenida Faucett mejora significativamente su eficiencia, mientras que Venezuela mantiene las ganancias del

sistema adaptativo. Así, la asignación de tiempos diferenciados genera un mejor equilibrio global entre equidad y eficiencia en la intersección analizada.

Tabla 9

Resultados comparativos entre ciclo fijo y adaptativo con tiempos diferentes en cada vía

Nota. Vía 1 Venezuela oeste, vía 2 Faucett sur, vía 3 Venezuela este y vía 4 Faucett norte.

Vía	Verde actual	Vehículos actuales	Flujo veh/s	Verde adaptativo	Vehículos adaptativos	Flujo veh/s	Variación (%)
1	65	47.5	0.73	50	46	0.92	+26%
2	140	79.5	0.57	70	85	1.21	+114%
3	65	40	0.62	50	48	0.96	+56%
4	140	85.5	0.61	70	61	0.87	+43%

CAPÍTULO VI: DISCUSIÓN

VI.1. Discusión de resultados

Los datos obtenidos permiten evidenciar diferencias significativas entre el sistema convencional de ciclo fijo y el sistema adaptativo propuesto, tanto en el escenario de tiempos iguales como en el escenario con tiempos diferenciados por vía.

- Venezuela (vías 1 y 3): En ambos escenarios, el flujo vehicular por segundo se incrementó de manera consistente. Con tiempos iguales, el aumento fue del orden del +26% y +50%, mientras que con tiempos diferenciados la mejora se mantuvo en valores similares (+26% y +56%). Esto demuestra que, al reducir el tiempo verde sobrante, se aprovecha mejor la capacidad de las vías que en el sistema actual permanecían subutilizadas.
- Faucett (vías 2 y 4): En el escenario de tiempos iguales, el flujo por segundo mostró una reducción respecto al sistema actual, debido a que el verde uniforme no compensó la alta demanda de esta arteria. Sin embargo, al aplicar tiempos diferenciados, los resultados cambiaron significativamente: la vía 2 (Faucett Sur) alcanzó un incremento notable de +114%, mientras que la vía 4 (Faucett Norte) mostró una mejora moderada de +43%. Esto confirma que la asignación de tiempos según la geometría y volumen vehicular es fundamental para no sacrificar eficiencia en vías principales.
- Equidad: El sistema convencional evidenció una variabilidad marcada entre vías (de hasta un 60%), mientras que el sistema adaptativo logró reducir esta diferencia. Con tiempos iguales, la equidad mejora, pero penaliza a Faucett. Con tiempos diferenciados, se alcanza un mejor balance global, manteniendo mejoras en Venezuela y recuperando eficiencia en Faucett.
- Comportamiento fuera de hora punta: Si bien las grabaciones utilizadas corresponden al periodo de mayor congestión (17:30 horas), es importante destacar que el sistema adaptativo ofrece beneficios adicionales en horarios de menor demanda. En estas condiciones, donde las vías presentan menor

afluencia, el algoritmo puede asignar tiempos de verde más cortos sin generar colas significativas. Esto evita que los semáforos permanezcan en verde cuando no hay vehículos esperando, reduciendo el tiempo perdido y favoreciendo la fluidez global en la intersección durante el resto del día.

En resumen, el sistema adaptativo demuestra ser más eficiente que el ciclo fijo al redistribuir los tiempos de luz verde de acuerdo con la demanda.

No obstante, el análisis confirma que el mejor desempeño se obtiene al aplicar tiempos diferenciados por vía, ya que este enfoque combina la reducción de desperdicio en avenidas secundarias (Venezuela) con la atención adecuada de la alta demanda en avenidas principales (Faucett). Asimismo, en horarios no pico, el modelo contribuye a minimizar tiempos muertos y mejorar la fluidez vehicular de manera sostenida.

VI.2. Propuestas de mejora

En base a los resultados, se plantean los siguientes ajustes:

1. Extensión del verde por demanda residual: prolongar el tiempo de verde durante 5 – 10 segundos, si permanecen más de dos bandas ocupadas al final de la fase
2. Tiempo diferenciado por avenida: dar un rango mayor de tiempo (90 segundos en caso de demanda completa) a Faucett, debido a las dimensiones de la vía.
3. Histéresis en el cambio de fase: evitar transacciones bruscas si la fase activa aún mantiene alta demanda.
4. Flujo mínimo de vehículos viables: Determinar que durante cada fase verde se logre despachar una cantidad suficiente de autos, evitando que el semáforo cambie antes que realmente se halla aprovechado el tiempo.

VI.3. Limitaciones del estudio

- El conteo manual del sistema actual puede prestarse a errores humanos y a la calidad del video.

- El análisis se realizó con fragmentos de videos (con un total de 5 minutos cada video), lo que no representa la conducta del tráfico real.
- En ciertos casos puntuales los videos no fueron óptimos debido a sus movimiento o ángulos inadecuados.
- El prototipo fue validado media la simulación con videos pregrabados, y si bien llego a implementar con una Raspberry Pi y cámaras ESP32-CAM, todavía no se ha probado en un entorno real de tráfico vehicular.

VI.4. Análisis comparativo con trabajos similares

Además de los resultados propios, resulta pertinente contrastar el sistema desarrollado en Lima con experiencias similares en la región. En particular, se revisa la tesis presentada por Castro (2022). En dicho trabajo se probó inicialmente el uso de la red YOLO para la detección vehicular; sin embargo, debido a limitaciones de hardware y a la necesidad de entrenar modelos específicos para la realidad local, el autor optó finalmente por implementar técnicas de procesamiento de imágenes clásicas con OpenCV (sustracción de fondo, filtrado morfológico y conteo básico). El sistema se aplicó únicamente a dos vías, con almacenamiento de datos en PostgreSQL, mostrando una reducción de congestión respecto al ciclo fijo. En contraste, la presente investigación en Lima integra YOLOv8 como base de detección, incorporando clasificación de la demanda vehicular en cinco niveles, así como la diferenciación de tiempos por avenida principal y secundaria, lo cual incrementa la adaptabilidad del sistema en un cruce complejo como el de Faucett con Venezuela.

Tabla 10

Comparación entre el sistema propuesto en Lima y la tesis de Ecuador

Criterio	Tesis Perú	Tesis Ecuador
Lugar de aplicación	Cruce Av. Faucett con Av. Venezuela (Callao, Perú)	Intersección genérica en Quito, Ecuador
Tecnología principal	YOLOv8 (detección por redes neuronales profundas)	OpenCV clásico (sustracción de fondo, filtrado morfológico)
Uso de YOLO	Integrado como base del sistema adaptativo	Probado, pero descartado por limitaciones de hardware y entrenamiento
Procesamiento	Raspberry Pi 5 + ESP32-CAM, optimizado para hardware embebido	Raspberry Pi 4 con cámaras web, pruebas en prototipo a escala
Método de detección	Ocupación por bandas (5 niveles: vacía, baja, media, alta, completa)	Conteo básico de vehículos en dos vías
Gestión de fases	Tiempo dinámico diferenciado: 10–60 s (más para Faucett que Venezuela)	Tiempo dinámico solo entre dos vías, sin diferenciación por jerarquía vial
Almacenamiento de datos	Local en Raspberry (sin BD central)	PostgreSQL para registro de conteos
Resultados	Reducción significativa del tiempo de espera; mayor adaptabilidad en vías principales	Reducción de congestión en prototipo, pero limitado a 2 vías
Limitaciones	No validado aún en tráfico en tiempo real (solo videos pregrabados)	Detección poco precisa en condiciones nocturnas; descartó YOLO por rendimiento

En conclusión, ambos trabajos validan la eficacia de la visión artificial para la gestión dinámica del tránsito, pero con enfoques distintos: mientras el prototipo ecuatoriano representa una aproximación inicial basada en herramientas de procesamiento clásicas, la presente investigación incorpora modelos de inteligencia artificial más avanzados, ofreciendo un mayor potencial de escalabilidad y aplicación en escenarios urbanos críticos.

CAPÍTULO VII: CONCLUSIONES

Con los resultados obtenidos durante la evaluación del prototipo de semáforo con visión artificial en la intersección de Faucett con Venezuela, se establecen las siguientes conclusiones:

1. Viabilidad técnica del prototipo: Se demostró que la visión artificial es una herramienta viable para determinar la demanda vehicular en tiempo real, utilizando algoritmos de segmentación por bandas y detección automática de vehículos para medir la demanda de cada vía.
2. Uso completo del tiempo en verde: Gracias al ciclo adaptativo, se redujo el tiempo de luz verde sin demanda (especialmente en la avenida Venezuela), incrementando el flujo vehicular procesado en más de un 50% respecto al sistema convencional.
3. Tiempo entre vías equilibrado: El algoritmo otorgó a las fases de verde un tiempo más equitativo entre las vías, disminuyendo su variabilidad de un 60% a cerca de un 3%. Sin embargo, se evidenció que en vías de mayor carga como Faucett este esquema uniforme reduce la eficiencia, lo que confirma la necesidad de un ajuste diferenciado.
4. Desempeño en vías de alta demanda: En el escenario con tiempos diferenciados, las vías de la avenida Faucett incrementaron notablemente su rendimiento, con mejoras de hasta +114% en el flujo vehicular, mientras que la avenida Venezuela mantuvo sus beneficios. Esto demuestra que la calibración según la geometría y demanda de cada vía es esencial para no sacrificar arterias principales.
5. Beneficios fuera de la hora punta: Aunque el análisis se centró en la hora de mayor congestión (17:30 h), el sistema adaptativo ofrece ventajas adicionales en horarios de menor demanda, evitando ciclos de verde innecesarios y favoreciendo una circulación más fluida durante todo el día.

6. Verificación del método empleado: La comparación entre los videos del municipio, grabaciones propias y videos óptimos logró demostrar que, aun considerando las limitaciones de ángulo de las grabaciones, el sistema puede operar de manera confiable y constituirse en una base sólida para futuros prototipos implementados en tiempo real.
7. Sistema escalable: El prototipo ofrece una base firme para la implementación de soluciones de gestión del tráfico aún más avanzadas, dado que, al combinar la comunicación entre semáforos y sensores, se elimina la necesidad de una red central.

CAPÍTULO VIII: RECOMENDACIONES

Finalmente, una vez desarrolladas las conclusiones, se procederá a formular recomendaciones para las mejoras, implementaciones e investigaciones futuras que tendrá el sistema:

1. Ajustes de parámetros diferenciados por vía: Incluir en el código tiempos de luz verde adaptados a la geometría y demanda de cada avenida. En vías principales como la avenida Faucett, se recomienda asignar en condiciones de alta demanda tiempos de 80 a 90 segundos, mientras que en vías secundarias se puede mantener un rango menor.
2. Prolongación dinámica del tiempo: Añadir al código la posibilidad si segundos antes de terminar el tiempo de un semáforo en verde, si aún existe demanda en más de dos bandas, siempre y cuando la demanda de las otras vías es baja.
3. Evaluación del sistema en el campo y optimización del código: Si bien el prototipo del sistema demostró con videos pregrabados un buen desempeño, se requiere llevarlo a un entorno de condiciones reales de tráfico, donde ya no se trabajará con secuencias de video, sino con toma de imágenes (frames) de las cámaras esp32, a partir de ello se requerirá la optimización del código para procesar exclusivamente estos frames en tiempo real.
4. Integración de cámaras y otros sensores: Complementar el sistema con cámaras ESP32-CAM adjuntas en posiciones laterales de la vía para poder confirmar la demanda que no se puede observar por la cámara principal, sobre todo los vehículos más alejados en el tramo. También se puede complementar con sensores magnéticos o de lazo inductivo, a fin de mejorar la detección en condiciones de baja visibilidad (noche, lluvia).
5. Pruebas en horarios diversos y a largo plazo: Realizar estudios más extensos (≥ 30 minutos por fase) en distintos horarios del día. Esto permitirá validar no solo el desempeño en hora punta, sino también en horarios de baja demanda, donde el sistema puede reducir tiempos muertos y favorecer la fluidez general.

6. Escalabilidad para diferentes tipos de intersecciones: Diseñar una arquitectura modular enfocada a distintos tipos de cruces viales, desde un cruce simple donde solo se requieran 2 o 3 semáforos, hasta intersecciones más complejas con múltiples accesos. Esto garantiza que el sistema se pueda adaptar a diversos escenarios urbanos.

CAPÍTULO IX: BIBLIOGRAFÍA

- AAP. (2024). Lima continúa siendo la ciudad con mayor tráfico vehicular de américa latina. <https://aap.org.pe/lima-continua-siendo-la-ciudad-con-mayor-trafico-vehicular-de-america-latina-agencia-nacional-de-transito-tomtom/>
- Infobae. (2024). Lima es la quinta ciudad con más tráfico vehicular del mundo: pérdidas de horas, dinero y contaminación ambiental. <https://www.infobae.com/peru/2024/02/08/lima-es-la-quinta-ciudad-con-mas-trafico-vehicular-del-mundo-perdida-de-horas-dinero-y-contaminacion-ambiental/>
- ATU Perú. (2024). ¿Quién o qué decide el ritmo de los semáforos en Lima? <https://itsperu.org/articulos/quien-o-que-decide-el-ritmo-de-los-semaforos-en-lima/>
- ITS Perú. (2024). Plan regulador de rutas. <https://cdn.www.gob.pe/uploads/document/file/7433554/6332854-plan-regulador-de-rutas-2024.pdf>
- Infobae. (2025). Plan para modernizar semáforos con IA en lima costara 300 millones. <https://www.infobae.com/peru/2025/02/21/rafael-lopez-aliaga-presenta-plan-para-modernizar-semaforos-con-ia-en-lima-destinaran-300-millones-de-soles/>
- Prensa Chalaca. (2022). Bellavista: gran congestión vehicular en la Av. Faucett con Venezuela. <https://prensachalaca.com/https-prensachalaca-com-web2022-bellavista-gran-congestion-vehicular-en-la-av-faucett-con-venezuela/>
- Municipio de San Miguel. (2025). Imagen de la intersección Av. Venezuela y Av. Faucett [Material no publicado]. Archivo proporcionado al autor.

- D. Arce Cigüeñas. (2017). Sistema Autónomo de Control de Tráfico Vehicular para Intersecciones de Avenidas (Tesis, Pontificia Universidad Católica del Perú).
- M. De La Cruz, G. Risco, R. Paredes. (2019). El Neuro Semáforo: Alternativa para la Reducción del Tráfico a Través de un Sistema Basado en IA. (Tesis, Universidad San Ignacio de Loyola, Perú).
- Yim Isaias Apestegui Florentino y Hugo Esteban Caselli Gismondi. (2024). Sistema Inteligente de Gestión de Semáforos en Tiempo Real. (Tesis, Universidad Nacional del Santa)
- C. Sosa Ramírez. (2020). Diseño de un Sistema de Semaforización Inteligente para Controlar el Flujo Vehicular a partir del Procesamiento de Imágenes. (Tesis, Universidad Piloto de Colombia)
- Moneybarn. (2020). Cuáles son las ciudades más lentas para conducir.
<https://www.moneybarn.com/slowest-cities/>
- R. Morales, J. González. (2013). Control de Tráfico Vehicular por Medio de Semáforos Inteligentes. (Tesis, Universidad Bolivariana de Venezuela)
- Marcelo Omar Castro Quishpe. (2022). Desarrollo de un prototipo de semáforo inteligente con visión por computador. (Tesis, Universidad Politécnica Salesiana de Ecuador).
- Szeliski, R. (2010). Computer vision: Algorithms and applications (2nd ed.). Springer.
- Gonzalez, R. C., & Woods, R. E. (2018). Digital image processing (4th ed.). Pearson.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016) You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

- Belinda Chong Chiew Meng, Damanhuri, N. S., & Othman, N. A. (2021). Smart traffic light control system using image processing.
- Ministerio de Transportes y Comunicaciones. (2016). Manual de dispositivos de control de tránsito automotor para calles y carreteras.
https://portal.mtc.gob.pe/transportes/caminos/normas_carreteras/MTC%20NORMAS/ARCH_PDF/MAN_6%20DCT-2016.pdf.
- Rafael Cal y Mayor R y James Cárdenas. (2018). Ingeniería de Tránsito. (9th ed.). Alfaomega.
- Kentucky Transportation Center. (2008). Eficiencia operacional de las instalaciones semaforicas.
https://uknowledge.uky.edu/cgi/viewcontent.cgi?article=1072&context=ktc_researchreports#:~
- D. Dorta Acosta. (2019). Simulación de semáforo inteligente. (Tesis, Universidad de la Laguna)
- Gunther Gridling. (2007) Introducción a microcontroladores. Versión 1.4.
- MikroElectronica (2013) Introducción al mundo de los microcontroladores.
<https://www.mikroe.com/ebooks/microcontroladores-pic-programacion-en-c-con-ejemplos/introduccion-al-mundo-de-los-microcontroladores>
- Mg. Luis Pedraza Martinez. (2012). Sistema de comunicación TCP/IP para el control de una intersección de tráfico vehicular. (Tesis Universidad Distrital Francisco José de Caldas).
- Gobierno del estado de Jalisco. (2020). Se renueva la red de semáforos.
<https://www.jalisco.gob.mx/es/gobierno/comunicados/con-siga-se-renueva-la-red-de-semaforos-y-la-movilidad-en-la-ciudad>

- Gobierno de Bogotá (2016). Semaforización inteligente en Bogotá.
<https://bogota.gov.co/yo-participo/blogs/semaforizacion-inteligente-en-bogota\>.
- Gobierno de Medellín Colombia (2016). Nueva optimización de semáforos,
<https://www.medellin.gov.co/SIMM/optimizacion-semaforos>.
- Municipalidad de Lima. (2017). Gestión y control.
<http://protransito.munlima.gob.pe/index.php/es-es/monitoreo/ccgt>
- Espressif Systems. (2023). ESP32-CAM datasheet. <https://www.espressif.com/>
- Raspberry Pi Foundation. (2023). Raspberry Pi 5 product brief.
<https://www.raspberrypi.com/>

APENDICE A

Código del servidor Flask:

```
from flask import Flask, request
import os
from datetime import datetime

app = Flask(__name__)
UPLOAD_FOLDER = "imagenes"
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

@app.route('/upload/<via>', methods=['POST'])
def upload(via):
    now = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"{via}_{now}.jpg"
    via_folder = os.path.join(UPLOAD_FOLDER, via)
    os.makedirs(via_folder, exist_ok=True)
    filepath = os.path.join(via_folder, filename)

    with open(filepath, 'wb') as f:
        f.write(request.data)

    print(f"[{via}] Imagen recibida: {filename}")
    return "OK", 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

APENDICE B

Código del ESP32-CAM:

```
#include "esp_camera.h"
#include <WiFi.h>
#include <HTTPClient.h>

// Datos del WiFi del Access Point
const char* ssid = "SemaforoAI";
const char* password = "clave123456";

// URL del servidor Flask
const char* serverName = "http://192.168.4.1:5000/upload/via1"; // puedes cambiar 'via1'

void setup() {
  Serial.begin(115200);

  // Configuración de la cámara (ESP32-CAM AI Thinker)
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = 5;
  config.pin_d1 = 18;
  config.pin_d2 = 19;
  config.pin_d3 = 21;
  config.pin_d4 = 36;
  config.pin_d5 = 39;
  config.pin_d6 = 34;
  config.pin_d7 = 35;
  config.pin_xclk = 0;
  config.pin_pclk = 22;
  config.pin_vsync = 25;
  config.pin_href = 23;
  config.pin_sscb_sda = 26;
  config.pin_sscb_scl = 27;
  config.pin_pwdn = 32;
  config.pin_reset = -1;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_QVGA; // Puedes probar con FRAMESIZE_VGA
  config.jpeg_quality = 12;
  config.fb_count = 1;

  // Inicializar la cámara
  if (esp_camera_init(&config) != ESP_OK) {
```

```

    Serial.println(" Error al inicializar la cámara");
    return;
}
// Conexión al WiFi
WiFi.begin(ssid, password);
Serial.print("Conectando a WiFi");

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("\n Conectado a WiFi");
}

void loop() {
    if (WiFi.status() == WL_CONNECTED) {
        camera_fb_t *fb = esp_camera_fb_get();
        if (!fb) {
            Serial.println(" ✕ Error al capturar imagen");
            return;
        }

        HTTPClient http;
        http.begin(serverName);
        http.addHeader("Content-Type", "image/jpeg");

        int httpResponseCode = http.POST(fb->buf, fb->len);

        if (httpResponseCode > 0) {
            Serial.printf(" Imagen enviada. Código: %d\n", httpResponseCode);
        } else {
            Serial.printf(" Error al enviar: %s\n",
http.errorToString(httpResponseCode).c_str());
        }

        http.end();
        esp_camera_fb_return(fb);
    } else {
        Serial.println("WiFi desconectado");
    }

    delay(3000); // Esperar 3 segundos antes de tomar otra foto
}

```

APENDICE C

Código de detección y control para una sola vía:

```
import cv2
import numpy as np
from ultralytics import YOLO
import os

# --- Función para guardar frame (comentada) ---
def guardar_frame_cambio_estado(frame, estado, frame_count, ids_activadores,
demanda, conteo, carpeta="detecciones"):
    # if not os.path.exists(carpeta):
    #     os.makedirs(carpeta)
    # ids_str = '-'.join(str(i) for i in ids_activadores) if ids_activadores else "none"
    # filename =
    f"{carpeta}/cambio_{estado}_frame{frame_count}_id{ids_str}_dem{demanda}_cnt
{conteo}.jpg"
    # cv2.imwrite(filename, frame)
    # print(f"[DEBUG] Cambio a {estado} en frame {frame_count} | ID(s): {ids_str} |
Demanda: {demanda} | Conteo: {conteo}")
    pass

def boxes_iou(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])
    interW = max(0, xB - xA)
    interH = max(0, yB - yA)
    interArea = interW * interH
    boxAArea = (boxA[2] - boxA[0]) * (boxA[3] - boxA[1])
    boxBArea = (boxB[2] - boxB[0]) * (boxB[3] - boxB[1])
    unionArea = boxAArea + boxBArea - interArea
    return interArea / unionArea if unionArea > 0 else 0

model = YOLO("yolov8n.pt")

polygon = np.array([
    [160, 445],
    [900, 445],
    [450, 100],
    [375, 100],
], np.int32)

PRE_LINE_Y = 325
COUNT_LINE_Y = 400
```

```

MEMORIA_FRAMES = 60
DETENIDO_FRAMES = 40
UMBRAL_MOVIMIENTO = 60
UMBRAL_PERSONA_CERCA = 125
IOU_PERSONA_CARRO = 0.12

```

```

NUM_BANDAS = 4

```

```

def calcular_estado_semaforo(vehiculos_en_zona, vehiculos_avanzando,
frames_desde_ultimo_avance):

```

```

    if len(vehiculos_en_zona) > 0 and len(vehiculos_avanzando) > 0:
        estado_actual = "VERDE"
        frames_desde_ultimo_avance = 0
        color_estado = (0, 255, 0)

```

```

    else:

```

```

        if frames_desde_ultimo_avance < MEMORIA_FRAMES:

```

```

            estado_actual = "VERDE"
            color_estado = (0, 255, 0)
            frames_desde_ultimo_avance += 1

```

```

        else:

```

```

            estado_actual = "ROJO"
            color_estado = (0, 0, 255)
            frames_desde_ultimo_avance += 1

```

```

    return estado_actual, color_estado, frames_desde_ultimo_avance

```

```

def actualizar_conteo(id, cx, cy, polygon, COUNT_LINE_Y, tracked_ids,
car_count):

```

```

    if (id not in tracked_ids and
        cy > COUNT_LINE_Y - 5 and cy < COUNT_LINE_Y + 5 and
        cv2.pointPolygonTest(polygon, (cx, COUNT_LINE_Y), False) >= 0):
        car_count += 1
        tracked_ids.add(id)
    return car_count, tracked_ids

```

```

def calcular_band_limits(polygon, y_start, y_end, num_bands):

```

```

    bands = []

```

```

    for i in range(num_bands):

```

```

        frac0 = i / num_bands
        frac1 = (i + 1) / num_bands
        x_left0 = int(polygon[0][0] + frac0 * (polygon[3][0] - polygon[0][0]))
        x_left1 = int(polygon[0][0] + frac1 * (polygon[3][0] - polygon[0][0]))
        y0 = int(y_end + frac0 * (y_start - y_end))
        y1 = int(y_end + frac1 * (y_start - y_end))
        x_right0 = int(polygon[1][0] + frac0 * (polygon[2][0] - polygon[1][0]))
        x_right1 = int(polygon[1][0] + frac1 * (polygon[2][0] - polygon[1][0]))
        bands.append((
            (x_left0, y0, x_right0, y0),
            (x_left1, y1, x_right1, y1),
            y0, y1
        ))

```

```

return bands

def evaluar_demanda(centroides, bands):
    bandas_ocupadas = [False]*len(bands)
    for idx, band in enumerate(bands):
        y_sup, y_inf = min(band[2], band[3]), max(band[2], band[3])
        for (cx, cy) in centroides:
            if y_sup <= cy < y_inf:
                bandas_ocupadas[idx] = True
                break
    num_ocupadas = sum(bandas_ocupadas)
    if num_ocupadas == 0:
        nivel = "vacía"
    elif num_ocupadas == 1:
        nivel = "baja"
    elif num_ocupadas == 2:
        nivel = "media"
    elif num_ocupadas == 3:
        nivel = "alta"
    else:
        nivel = "completa"
    return nivel, num_ocupadas, bandas_ocupadas

cap = cv2.VideoCapture("video_via1.MP4")
car_count = 0
tracked_ids = set()

estado_actual = "ROJO"
frames_desde_ultimo_avance = MEMORIA_FRAMES + 1

historial_centroides = dict()
BANDS = calcular_band_limits(polygon, polygon[2][1], polygon[0][1],
NUM_BANDAS)
ultimo_estado = estado_actual
frame_count = 0

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Escala el frame a 960x540 (si tu video no ya está así)
    frame = cv2.resize(frame, (960, 540))
    frame_count += 1

    cv2.polylines(frame, [polygon], isClosed=True, color=(0,0,255), thickness=3)
    for band in BANDS:
        pts = np.array([
            [band[0][0], band[0][1]],
            [band[0][2], band[0][3]],

```

```

        [band[1][2], band[1][3]],
        [band[1][0], band[1][1]]
    ], np.int32)
    cv2.polylines(frame, [pts], isClosed=True, color=(150,180,255), thickness=1)
    cv2.line(frame, (polygon[0][0], PRE_LINE_Y), (polygon[1][0], PRE_LINE_Y),
(0, 255, 255), 2)
    cv2.line(frame, (polygon[0][0], COUNT_LINE_Y), (polygon[1][0],
COUNT_LINE_Y), (255, 255, 0), 2)

```

```

# Procesa solo 1 de cada 2 frames para más velocidad
if frame_count % 2 != 0:
    continue

```

```

results = model.track(frame, persist=True, classes=[0,2,5,7],
tracker="bytetrack.yaml")

```

```

vehiculos_en_zona = []
vehiculos_avanzando = []
centroides_demanda = []
personas_centroides = []
personas_boxes = []
carros_boxes = []
carros_boxes_info = []

```

```

if results[0].boxes.id is not None:
    for box, id, cls in zip(
        results[0].boxes.xyxy.cpu().numpy(),
        results[0].boxes.id.cpu().numpy(),
        results[0].boxes.cls.cpu().numpy(),
    ):
        x1, y1, x2, y2 = map(int, box)
        cx = int((x1 + x2) // 2)
        cy = int((y1 + y2) // 2)
        point = (cx, cy)
        box_tuple = (x1, y1, x2, y2)
        if cv2.pointPolygonTest(polygon, point, False) >= 0:
            cv2.circle(frame, point, 7, (0, 255, 0), -1)
            cv2.putText(frame, f"ID:{int(id)}", (cx-10, cy-10),
cv2.FONT_HERSHEY_PLAIN, 0.8, (0,255,255), 2)
            if int(cls) in [2,5,7] and cv2.pointPolygonTest(polygon, point, False) >= 0:
                carros_boxes.append(box_tuple)
                carros_boxes_info.append((box_tuple, point, id))
            if int(cls) == 0 and cv2.pointPolygonTest(polygon, point, False) >= 0:
                personas_centroides.append((cx, cy))
                personas_boxes.append(box_tuple)
            if int(cls) in [2,5,7] and cv2.pointPolygonTest(polygon, point, False) >= 0
and PRE_LINE_Y < cy < COUNT_LINE_Y:
                vehiculos_en_zona.append(int(id))
                if id not in historial_centroides:
                    historial_centroides[id] = []

```

```

        historial_centroides[id].append((cx, cy))
        if len(historial_centroides[id]) > DETENIDO_FRAMES:
            historial_centroides[id] = historial_centroides[id][-
DETENIDO_FRAMES:]
        if len(historial_centroides[id]) >= 2:
            dists = [np.linalg.norm(np.array(historial_centroides[id][i]) -
np.array(historial_centroides[id][i-1]))
                    for i in range(1, len(historial_centroides[id]))]
            total_dist = sum(dists)
            persona_cerca = False
            for (pcx, pcy), pbox in zip(personas_centroides, personas_boxes):
                if np.linalg.norm(np.array([cx, cy]) - np.array([pcx, pcy])) <
UMBRAL_PERSONA_CERCA:
                    persona_cerca = True
                    if boxes_iou(box_tuple, pbox) > IOU_PERSONA_CARRO:
                        persona_cerca = True
                    if total_dist > UMBRAL_MOVIMIENTO and not persona_cerca:
                        vehiculos_avanzando.append(int(id))
                    elif persona_cerca:
                        cv2.circle(frame, point, 9, (0,0,255), 2)
                        cv2.putText(frame, f"IGN-{int(id)}", (cx-10, cy-25),
cv2.FONT_HERSHEY_PLAIN, 0.9, (0,0,255), 2)
                        car_count, tracked_ids = actualizar_conteo(
                            id, cx, cy, polygon, COUNT_LINE_Y, tracked_ids, car_count
                        )
                    if int(cls) in [2,5,7] and cv2.pointPolygonTest(polygon, point, False) >= 0:
                        centroides_demanda.append((cx, cy))
                    else:
                        if id in historial_centroides:
                            del historial_centroides[id]

        estado_actual, color_estado, frames_desde_ultimo_avance =
calcular_estado_semaforo(
    vehiculos_en_zona, vehiculos_avanzando, frames_desde_ultimo_avance
)

        nivel_demanda, bandas_ocupadas, ocupadas_bool =
evaluar_demanda(centroides_demanda, BANDS)

        # ids_activadores = vehiculos_avanzando if estado_actual == "VERDE" else []
        # demanda_str = f"{nivel_demanda}_{bandas_ocupadas}de{NUM_BANDAS}"
        # if estado_actual != ultimo_estado:
        #     guardar_frame_cambio_estado(
        #         frame, estado_actual, frame_count, ids_activadores, demanda_str,
car_count
        #     )
        #     ultimo_estado = estado_actual

        cv2.putText(frame, f"ESTADO: {estado_actual}", (40, 60),
cv2.FONT_HERSHEY_SIMPLEX, 1.4, color_estado, 3)

```

```

    cv2.putText(frame, f"Conteo: {car_count}", (180, 140),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    cv2.putText(
        frame,
        f"Demanda: {nivel_demanda} ({bandas_ocupadas}/{NUM_BANDAS})",
        (40, 190),
        cv2.FONT_HERSHEY_SIMPLEX,
        1,
        (0, 150, 255),
        2
    )

    cv2.imshow("YOLOv8 - Zona de Cruce y Demanda", frame)
    if cv2.waitKey(67) == 27:
        break

cap.release()
cv2.destroyAllWindows()

```

APENDICE D

Código de gestión de las 4 vías (mismo tiempo en ambas vías):

```
import cv2
import numpy as np
import time
from ultralytics import YOLO

# ----- Ajustes de rendimiento -----
cv2.setNumThreads(2)          # limita threads de OpenCV (evita contención)
cv2.useOptimized()            # activa optimizaciones internas
SHOW_FPS = True
WAIT_MS = 1                   # menor latencia en imshow
DETECT_INTERVAL = 1.0         # s, ya lo usabas: detección cada X s (ROJO)
DETECT_IMGSZ = 384            # reducción de tamaño para YOLO (más rápido)
RESIZE_W, RESIZE_H = 480, 270 # panel 2x2 = 960x540 (liviano)
# -----

# =====
# Configuración de videos
# =====
videos_rojo =
["via1_ROJO.mp4", "via2_ROJO.mp4", "via3_ROJO.mp4", "via4_ROJO.mp4"]
videos_verde =
["via1_VERDE.mp4", "via2_VERDE.mp4", "via3_VERDE.mp4", "via4_VERDE.mp4"]
nombres_vias = ["Via 1 Venezuela Oeste", "Via 2 Faucett Sur", "Via 3 Venezuela
Este", "Via 4 Faucett Norte"]

def abrir_cap(path):
    cap = cv2.VideoCapture(path)
    # Buffer chico para bajar la latencia de lectura
    cap.set(cv2.CAP_PROP_BUFFERSIZE, 2)
    # Si conoces el FPS del archivo, puedes fijarlo:
    # cap.set(cv2.CAP_PROP_FPS, 30)
    return cap

caps_rojo = [abrir_cap(v) for v in videos_rojo]
caps_verde = [abrir_cap(v) for v in videos_verde]

# ROI fijos
rois = [
    np.array([[100,230],[420,230],[230,50],[180,50]], np.int32),
    np.array([[20,210],[325,210],[275,60],[230,60]], np.int32),
    np.array([[100,250],[370,250],[265,90],[180,90]], np.int32),
    np.array([[70,250],[360,250],[275,100],[200,100]], np.int32)
]
```

```

NUM_BANDAS = 4
TIEMPOS_VERDE = {"vacia":10,"baja":20,"media":40,"alta":50,"completa":60}
TIEMPO_MINIMO_VERDE = 15

# ----- PRE-COMPUTE: bandas por vía (evita recalculer cada frame) -----
def calcular_band_limits(polygon, num_bands):
    bands = []
    pts = polygon.reshape(-1, 2)
    for i in range(num_bands):
        frac0 = i / num_bands; frac1 = (i + 1) / num_bands
        x_left0 = int(pts[0][0] + frac0 * (pts[3][0] - pts[0][0]))
        x_left1 = int(pts[0][0] + frac1 * (pts[3][0] - pts[0][0]))
        y0 = int(pts[0][1] + frac0 * (pts[3][1] - pts[0][1]))
        y1 = int(pts[0][1] + frac1 * (pts[3][1] - pts[0][1]))
        x_right0 = int(pts[1][0] + frac0 * (pts[2][0] - pts[1][0]))
        x_right1 = int(pts[1][0] + frac1 * (pts[2][0] - pts[1][0]))
        bands.append(np.array([x_left0,y0],[x_right0,y0],[x_right1,y1],[x_left1,y1],
np.int32))
    return bands

bands_por_via = [calcular_band_limits(roi, NUM_BANDAS) for roi in rois]

def evaluar_demanda(centroides, bands):
    bandas_ocupadas = [False]*len(bands)
    for idx, band in enumerate(bands):
        for (cx, cy) in centroides:
            if cv2.pointPolygonTest(band, (cx, cy), False) >= 0:
                bandas_ocupadas[idx] = True; break
    n = sum(bandas_ocupadas)
    nivel = ["vacia","baja","media","alta","completa"][min(n,4)]
    return nivel, n, bandas_ocupadas

# YOLO (input reducido para acelerar). Usa solo clases 2,5,7
model = YOLO("yolov8n.pt")
# model.fuse() # (opcional) en GPU/CPU puede ayudar levemente en inferencia

ultimos_resultados =
[{"demanda":"vacia","bandas":0,"bandas_ocupadas":[False]*NUM_BANDAS} for
_ in range(4)]
ultimo_analisis = [0.0]*4

FASE_FACUETT = [1, 3]
FASE_VENEZUELA = [0, 2]
fase_actual = "faucett"
temporizador_verde = 15
t_inicio_fase = time.time()

# ----- helpers de lectura rápida -----
def fast_read(cap, w, h):
    # Usa grab/retrieve para poder saltar frames si se atrasa

```

```

grabbed = cap.grab()
if not grabbed:
    cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
    grabbed = cap.grab()
    if not grabbed:
        return False, None
ret, frame = cap.retrieve()
if not ret:
    return False, None
return True, cv2.resize(frame, (w, h))

# Ventana
cv2.namedWindow("Sistema Semaforo Inteligente - Fluido",
cv2.WINDOW_NORMAL | cv2.WINDOW_KEEPRATIO)

fps_tick = time.time(); frames_vis = 0; fps_val = 0.0

while True:
    frames = []
    ahora = time.time()

    vias_verde = FASE_FACUETT if fase_actual == "faucett" else
FASE_VENEZUELA
    vias_rojo = FASE_VENEZUELA if fase_actual == "faucett" else
FASE_FACUETT

    # ---- ROJO: actualizar demanda SOLO cada DETECT_INTERVAL (y con
tamaño de inferencia menor) ----
    for i in vias_rojo:
        ret, frame = fast_read(caps_rojo[i], RESIZE_W, RESIZE_H)
        if not ret: continue

        cv2.polylines(frame, [rois[i]], True, (0,255,255), 2)
        for idx, band in enumerate(bands_por_via[i]):
            color_band = (180,180,255) if not
ultimos_resultados[i]["bandas_ocupadas"][idx] else (0,140,255)
            cv2.polylines(frame, [band], True, color_band, 2)

        if ahora - ultimo_analisis[i] >= DETECT_INTERVAL:
            # Inferencia sobre un frame reducido (solo para detección, no cambia lo que
mostramos)
            small = cv2.resize(frame, (DETECT_IMGSZ,
int(DETECT_IMGSZ*RESIZE_H/RESIZE_W)))
            results = model(small, classes=[2,5,7], conf=0.25, verbose=False)
            centroides = []
            if results and results[0].boxes is not None and results[0].boxes.xyxy is not
None:
                scale_x = RESIZE_W / small.shape[1]; scale_y = RESIZE_H /
small.shape[0]
                for box in results[0].boxes.xyxy.cpu().numpy():

```

```

        x1, y1, x2, y2 = box
        # reescala al frame de display
        x1, y1, x2, y2 = int(x1*scale_x), int(y1*scale_y), int(x2*scale_x),
int(y2*scale_y)
        cx, cy = (x1 + x2)//2, (y1 + y2)//2
        if cv2.pointPolygonTest(rois[i], (cx, cy), False) >= 0:
            centroides.append((cx, cy))
            cv2.circle(frame, (cx, cy), 6, (0,255,0), -1)

    nivel_demanda, num_ocupadas, bandas_ocupadas =
evaluar_demanda(centroides, bands_por_via[i])
    ultimos_resultados[i]["demanda"] = nivel_demanda
    ultimos_resultados[i]["bandas"] = num_ocupadas
    ultimos_resultados[i]["bandas_ocupadas"] = bandas_ocupadas
    ultimo_analisis[i] = ahora

frames.append((i, frame, False))

# ---- VERDE: solo overlays/temporizador (sin YOLO) para máxima fluidez ----
tiempo_fase = temporizador_verde - int(ahora - t_inicio_fase)
for i in vias_verde:
    ret, frame = fast_read(caps_verde[i], RESIZE_W, RESIZE_H)
    if not ret: continue
    cv2.polylines(frame, [rois[i]], True, (0,255,255), 2)
    for band in bands_por_via[i]:
        cv2.polylines(frame, [band], True, (0,255,100), 2)
    frames.append((i, frame, True))

frames.sort()
panel_frames = []
for idx, frame, es_verde in frames:
    nivel_demanda = ultimos_resultados[idx]["demanda"]
    num_ocupadas = ultimos_resultados[idx]["bandas"]
    color_estado = (0,255,0) if es_verde else (0,0,255)
    texto_estado = "VERDE" if es_verde else "ROJO"
    cv2.putText(frame, nombres_vias[idx], (15, 35),
cv2.FONT_HERSHEY_SIMPLEX, 0.85, (255,255,255), 2)
    cv2.putText(frame, f"Demanda: {nivel_demanda}
({num_ocupadas}/{NUM_BANDAS})", (15, 65),
cv2.FONT_HERSHEY_SIMPLEX, 0.90, (0,150,255), 2)
    if es_verde:
        cv2.putText(frame, f"ESTADO: {texto_estado}", (15, 95),
cv2.FONT_HERSHEY_SIMPLEX, 0.85, color_estado, 2)
        cv2.putText(frame, f"Tiempo: {max(0, tiempo_fase)}s", (15, 125),
cv2.FONT_HERSHEY_SIMPLEX, 0.95, (0,255,0), 2)
    else:
        cv2.putText(frame, f"ESTADO: {texto_estado}", (15, 95),
cv2.FONT_HERSHEY_SIMPLEX, 0.85, color_estado, 2)
    panel_frames.append(frame)

```

```

if len(panel_frames) == 4:
    fila1 = np.hstack([panel_frames[0], panel_frames[1]])
    fila2 = np.hstack([panel_frames[2], panel_frames[3]])
    panel = np.vstack([fila1, fila2])
else:
    # fallback si algún cap devolvió ret=False
    panel = np.zeros((RESIZE_H*2, RESIZE_W*2, 3), dtype=np.uint8)

nombre_fase = "FASE: Faucett VERDE" if fase_actual == "faucett" else "FASE:
Venezuela VERDE"
cv2.rectangle(panel, (0, 0), (960, 40), (30,30,30), -1)
cv2.putText(panel, nombre_fase, (20, 30), cv2.FONT_HERSHEY_SIMPLEX,
1.2, (0,255,255), 3)

# FPS indicador (opcional)
frames_vis += 1
if SHOW_FPS and (time.time() - fps_tick) >= 1.0:
    fps_val = frames_vis / (time.time() - fps_tick)
    fps_tick = time.time(); frames_vis = 0
if SHOW_FPS:
    cv2.putText(panel, f'FPS: {fps_val:.1f}', (800, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255,255,255), 2)

cv2.imshow("Sistema Semaforo Inteligente - Fluido", panel)

# ---- Cambio de fase (idéntico) ----
if tiempo_fase <= 0:
    orden = ["vacía", "baja", "media", "alta", "completa"]
    demanda_faucett = max(ultimos_resultados[1]["demanda"],
ultimos_resultados[3]["demanda"], key=lambda x: orden.index(x))
    demanda_venezuela = max(ultimos_resultados[0]["demanda"],
ultimos_resultados[2]["demanda"], key=lambda x: orden.index(x))
    idx_f, idx_v = orden.index(demanda_faucett), orden.index(demanda_venezuela)

    if idx_f > idx_v:
        siguiente_fase, demanda_fase = "faucett", demanda_faucett
    elif idx_f < idx_v:
        siguiente_fase, demanda_fase = "venezuela", demanda_venezuela
    else:
        siguiente_fase = "venezuela" if fase_actual == "faucett" else "faucett"
        demanda_fase = demanda_faucett

    temporizador_verde = max(TIEMPOS_VERDE[demanda_fase],
TIEMPO_MINIMO_VERDE)
    fase_actual = siguiente_fase
    t_inicio_fase = time.time()

if cv2.waitKey(WAIT_MS) == 27:
    break

```

```
for cap in caps_rojo + caps_verde:  
    cap.release()  
cv2.destroyAllWindows()
```

APENDICE E

Código de gestión de las 4 vías (vía Faucett con más tiempos):

```
import cv2
import numpy as np
import time
from ultralytics import YOLO

# ----- Ajustes de rendimiento -----
cv2.setNumThreads(2)          # limita threads de OpenCV (evita contención)
cv2.useOptimized()            # activa optimizaciones internas
SHOW_FPS = True
WAIT_MS = 1                   # menor latencia en imshow
DETECT_INTERVAL = 1.0         # s, detección cada X s (vías en ROJO)
DETECT_IMGSZ = 384            # reducción de tamaño para YOLO (más rápido)
RESIZE_W, RESIZE_H = 480, 270 # panel 2x2 = 960x540 (liviano)
# -----

# =====
# Configuración de videos
# =====
videos_rojo =
["via1_ROJO.mp4", "via2_ROJO.mp4", "via3_ROJO.mp4", "via4_ROJO.mp4"]
videos_verde =
["via1_VERDE.mp4", "via2_VERDE.mp4", "via3_VERDE.mp4", "via4_VERDE.mp4"]
nombres_vias = ["Via 1 Venezuela Oeste", "Via 2 Faucett Sur", "Via 3 Venezuela
Este", "Via 4 Faucett Norte"]

def abrir_cap(path):
    cap = cv2.VideoCapture(path)
    cap.set(cv2.CAP_PROP_BUFFERSIZE, 2) # buffer chico para bajar latencia
    # cap.set(cv2.CAP_PROP_FPS, 30)    # opcional si conoces FPS
    return cap

caps_rojo = [abrir_cap(v) for v in videos_rojo]
caps_verde = [abrir_cap(v) for v in videos_verde]

# ROI fijos
rois = [
    np.array([[100,230],[420,230],[230,50],[180,50]], np.int32),
    np.array([[20,210],[325,210],[275,60],[230,60]], np.int32),
    np.array([[100,250],[370,250],[265,90],[180,90]], np.int32),
    np.array([[70,250],[360,250],[275,100],[200,100]], np.int32)
]
NUM_BANDAS = 4
```

```

# ===== Tiempos diferenciados por corredor =====
TIEMPOS_FACUETT =
{"vacía":10,"baja":50,"media":70,"alta":80,"completa":90}
TIEMPOS_VENEZUELA =
{"vacía":10,"baja":20,"media":40,"alta":50,"completa":60}
TIEMPO_MINIMO_VERDE = 15
# =====

# ----- PRE-COMPUTE: bandas por vía -----
def calcular_band_limits(polygon, num_bands):
    bands = []
    pts = polygon.reshape(-1, 2)
    for i in range(num_bands):
        frac0 = i / num_bands; frac1 = (i + 1) / num_bands
        x_left0 = int(pts[0][0] + frac0 * (pts[3][0] - pts[0][0]))
        x_left1 = int(pts[0][0] + frac1 * (pts[3][0] - pts[0][0]))
        y0 = int(pts[0][1] + frac0 * (pts[3][1] - pts[0][1]))
        y1 = int(pts[0][1] + frac1 * (pts[3][1] - pts[0][1]))
        x_right0 = int(pts[1][0] + frac0 * (pts[2][0] - pts[1][0]))
        x_right1 = int(pts[1][0] + frac1 * (pts[2][0] - pts[1][0]))
        bands.append(np.array([x_left0,y0,x_right0,y0,x_right1,y1,x_left1,y1],
np.int32))
    return bands

bands_por_via = [calcular_band_limits(roi, NUM_BANDAS) for roi in rois]

def evaluar_demanda(centroides, bands):
    bandas_ocupadas = [False]*len(bands)
    for idx, band in enumerate(bands):
        for (cx, cy) in centroides:
            if cv2.pointPolygonTest(band, (cx, cy), False) >= 0:
                bandas_ocupadas[idx] = True
                break
    n = sum(bandas_ocupadas)
    nivel = ["vacía","baja","media","alta","completa"][min(n,4)]
    return nivel, n, bandas_ocupadas

# YOLO (input reducido para acelerar). Usa solo clases 2,5,7
model = YOLO("yolov8n.pt")
# model.fuse() # (opcional)

ultimos_resultados =
[{"demanda":"vacía","bandas":0,"bandas_ocupadas":[False]*NUM_BANDAS} for
_ in range(4)]
ultimo_analisis = [0.0]*4

# Índices de vías por corredor
FASE_FACUETT = [1, 3]
FASE_VENEZUELA = [0, 2]

```

```

# Estado inicial
fase_actual = "faucett"
temporizador_verde = 15
t_inicio_fase = time.time()

# ----- helpers de lectura -----
def fast_read(cap, w, h):
    grabbed = cap.grab()
    if not grabbed:
        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
        grabbed = cap.grab()
        if not grabbed:
            return False, None
    ret, frame = cap.retrieve()
    if not ret:
        return False, None
    return True, cv2.resize(frame, (w, h))

# Ventana
cv2.namedWindow("Sistema Semaforo Inteligente - Fluido",
cv2.WINDOW_NORMAL | cv2.WINDOW_KEEPRATIO)

fps_tick = time.time(); frames_vis = 0; fps_val = 0.0

while True:
    frames = []
    ahora = time.time()

    vias_verde = FASE_FACUETT if fase_actual == "faucett" else
FASE_VENEZUELA
    vias_rojo = FASE_VENEZUELA if fase_actual == "faucett" else
FASE_FACUETT

    # ---- ROJO: actualizar demanda SOLO cada DETECT_INTERVAL ----
    for i in vias_rojo:
        ret, frame = fast_read(caps_rojo[i], RESIZE_W, RESIZE_H)
        if not ret:
            continue

        cv2.polylines(frame, [rois[i]], True, (0,255,255), 2)
        for idx, band in enumerate(bands_por_via[i]):
            color_band = (180,180,255) if not
ultimos_resultados[i]["bandas_ocupadas"][idx] else (0,140,255)
            cv2.polylines(frame, [band], True, color_band, 2)

        if ahora - ultimo_analisis[i] >= DETECT_INTERVAL:
            small = cv2.resize(frame, (DETECT_IMGSZ,
int(DETECT_IMGSZ*RESIZE_H/RESIZE_W)))
            results = model(small, classes=[2,5,7], conf=0.25, verbose=False)
            centroides = []

```

if results and results[0].boxes is not None and results[0].boxes.xyxy is not None:

```

    scale_x = RESIZE_W / small.shape[1]; scale_y = RESIZE_H /
small.shape[0]
    for box in results[0].boxes.xyxy.cpu().numpy():
        x1, y1, x2, y2 = box
        x1, y1, x2, y2 = int(x1*scale_x), int(y1*scale_y), int(x2*scale_x),
int(y2*scale_y)
        cx, cy = (x1 + x2)//2, (y1 + y2)//2
        if cv2.pointPolygonTest(rois[i], (cx, cy), False) >= 0:
            centroides.append((cx, cy))
            cv2.circle(frame, (cx, cy), 6, (0,255,0), -1)

```

```

    nivel_demanda, num_ocupadas, bandas_ocupadas =
evaluar_demanda(centroides, bands_por_via[i])
    ultimos_resultados[i]["demanda"] = nivel_demanda
    ultimos_resultados[i]["bandas"] = num_ocupadas
    ultimos_resultados[i]["bandas_ocupadas"] = bandas_ocupadas
    ultimo_analisis[i] = ahora

```

```
frames.append((i, frame, False))
```

```

# ---- VERDE: solo overlays/temporizador (sin YOLO) ----
tiempo_fase = temporizador_verde - int(ahora - t_inicio_fase)
for i in vias_verde:
    ret, frame = fast_read(caps_verde[i], RESIZE_W, RESIZE_H)
    if not ret:
        continue
    cv2.polylines(frame, [rois[i]], True, (0,255,255), 2)
    for band in bands_por_via[i]:
        cv2.polylines(frame, [band], True, (0,255,100), 2)
    frames.append((i, frame, True))

```

```

frames.sort()
panel_frames = []
for idx, frame, es_verde in frames:
    nivel_demanda = ultimos_resultados[idx]["demanda"]
    num_ocupadas = ultimos_resultados[idx]["bandas"]
    color_estado = (0,255,0) if es_verde else (0,0,255)
    texto_estado = "VERDE" if es_verde else "ROJO"
    cv2.putText(frame, nombres_vias[idx], (15, 35),
cv2.FONT_HERSHEY_SIMPLEX, 0.85, (255,255,255), 2)
    cv2.putText(frame, f"Demanda: {nivel_demanda}
({num_ocupadas}/{NUM_BANDAS})", (15, 65),
cv2.FONT_HERSHEY_SIMPLEX, 0.90, (0,150,255), 2)
    if es_verde:
        cv2.putText(frame, f"ESTADO: {texto_estado}", (15, 95),
cv2.FONT_HERSHEY_SIMPLEX, 0.85, color_estado, 2)
        cv2.putText(frame, f"Tiempo: {max(0, tiempo_fase)}s", (15, 125),
cv2.FONT_HERSHEY_SIMPLEX, 0.95, (0,255,0), 2)

```

```

else:
    cv2.putText(frame, f"ESTADO: {texto_estado}", (15, 95),
                 cv2.FONT_HERSHEY_SIMPLEX, 0.85, color_estado, 2)
    panel_frames.append(frame)

if len(panel_frames) == 4:
    fila1 = np.hstack([panel_frames[0], panel_frames[1]])
    fila2 = np.hstack([panel_frames[2], panel_frames[3]])
    panel = np.vstack([fila1, fila2])
else:
    panel = np.zeros((RESIZE_H*2, RESIZE_W*2, 3), dtype=np.uint8)

nombre_fase = "FASE: Faucett VERDE" if fase_actual == "faucett" else "FASE:
Venezuela VERDE"
cv2.rectangle(panel, (0, 0), (960, 40), (30,30,30), -1)
cv2.putText(panel, nombre_fase, (20, 30), cv2.FONT_HERSHEY_SIMPLEX,
1.2, (0,255,255), 3)

# FPS indicador (opcional)
frames_vis += 1
if SHOW_FPS and (time.time() - fps_tick) >= 1.0:
    fps_val = frames_vis / (time.time() - fps_tick)
    fps_tick = time.time(); frames_vis = 0
if SHOW_FPS:
    cv2.putText(panel, f"FPS: {fps_val:.1f}", (800, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255,255,255), 2)

cv2.imshow("Sistema Semaforo Inteligente - Fluido", panel)

# ---- Cambio de fase: alternancia fija (sin "ganador") ----
if tiempo_fase <= 0:
    orden = ["vacia", "baja", "media", "alta", "completa"]

# Alterna corredor
siguiente_fase = "venezuela" if fase_actual == "faucett" else "faucett"

# Usa la demanda FRESCA del corredor que estaba en rojo para fijar su tiempo
if siguiente_fase == "faucett":
    demanda_fase = max(
        ultimos_resultados[1]["demanda"],
        ultimos_resultados[3]["demanda"],
        key=lambda x: orden.index(x)
    )
    tiempos = TIEMPOS_FACUETT
else:
    demanda_fase = max(
        ultimos_resultados[0]["demanda"],
        ultimos_resultados[2]["demanda"],
        key=lambda x: orden.index(x)
    )

```

```

    tiempos = TIEMPOS_VENEZUELA

    temporizador_verde = max(tiempos[demanda_fase],
    TIEMPO_MINIMO_VERDE)

    if fase_actual == "faucett":
        for idx in FASE_FACUETT:
            caps_verde[idx].set(cv2.CAP_PROP_POS_FRAMES, 0)
    else:
        for idx in FASE_VENEZUELA:
            caps_verde[idx].set(cv2.CAP_PROP_POS_FRAMES, 0)

    fase_actual = siguiente_fase
    t_inicio_fase = time.time()

    if cv2.waitKey(WAIT_MS) == 27:
        break

for cap in caps_rojo + caps_verde:
    cap.release()
cv2.destroyAllWindows()

```