

# Cruces inteligentes con EDGE AI embebido

Murillo Vega Milton, *Estudiante, ITCR*, Marcos Josue Marin Vargas, *Estudiante, ITCR*,  
Rodriguez Rocha Fernando Jose, *Estudiante, ITCR*

## Resumen

Este proyecto plantea el diseño e implementación de un sistema embebido inteligente orientado a la gestión segura y adaptativa de cruces peatonales mediante visión a través de cámaras e inteligencia artificial (Edge AI). Como elemento central de procesamiento se incorpora una Raspberry Pi, equipada con cámara y módulos de control físico, que permite detectar la presencia de personas, vehículos y animales para ajustar de manera dinámica las fases del semáforo, agilizando y mejorando la seguridad de peatón a transitar.

A través del uso de librerías como OpenCV, GStreamer y TensorFlow Lite, se realiza la adquisición, preprocesamiento y análisis de las imágenes en tiempo real, ejecutando localmente los modelos entrenados previamente para evitar dependencia de conectividad y reducir la latencia. La lógica de control, desarrollada en Python, interpreta los resultados de detección y aplica políticas y reglas adaptativas, como la reducción progresiva de tiempos de espera peatonal o la prioridad inmediata ante individuos de calidad animal.

La propuesta contribuye a la seguridad vial, la eficiencia del tránsito y la sostenibilidad, integrando tecnologías accesibles dentro de un enfoque de ciudad inteligente.

## Index Terms

Raspberry Pi, RTSP, GStreamer, TensorFlow Lite, OpenCV.

## I. Introducción

La seguridad de usuarios vulnerables (peatones y ciclistas) en intersecciones urbanas sigue siendo un reto prioritario para la gestión del tránsito. La normativa vigente en señales y tiempos semafóricos, como el *MUTCD* y guías estatales complementarias, orienta a que la operación peatonal disponga de tiempos suficientes y criterios de actuación basados en condiciones reales de la vía [1], [2]. En este marco, nuestro proyecto propone un sistema embebido de percepción y control que detecta personas, animales y vehículos para forzar transiciones semafóricas seguras y eficientes, alineadas con mejores prácticas regulatorias.

La literatura reciente sostiene la viabilidad y conveniencia de usar visión por computador para detección, seguimiento y análisis de conflictos en intersecciones, incluso en configuraciones multi-cámara que reducen oclusiones y mejoran la lectura de la escena [3]. Además, la predicción de intención de cruce de usuarios vulnerables (VRU) permite anticipar cambios de fase y reducir esperas innecesarias, con impactos positivos en seguridad y cumplimiento [4]. Estas estrategias se enmarcan dentro de la gestión adaptativa de semáforos para ciudades inteligentes, donde el control sensible a la demanda contribuye a la movilidad y a la accesibilidad [5]. A nivel institucional, se reconoce la prioridad de contar con detección fiable de VRU en infraestructura vial [6].

La factibilidad técnica se potencia con *edge AI*: ejecutar la inferencia en el dispositivo minimiza latencia, dependencia de conectividad y exposición de datos personales. En particular,

*LiteRT* (antes TensorFlow Lite) facilita modelos optimizados para hardware de bajo consumo, como Raspberry Pi con webcam USB. Para monitoreo en tiempo real, la transmisión mediante RTSP (*gst-rtsp-server*) integra fácilmente el video con clientes estándar (p.ej., VLC) [8]. En entornos urbanos con presencia de fauna, enfoques de detección de animales basados en aprendizaje profundo aportan precisión para activar políticas de resguardo específicas [10].

En síntesis, la combinación de normativa y evidencia científica justifica un sistema que: (i) detecte personas y active una transición cronometrada a rojo con reducción dinámica del tiempo conforme llegan más peatones; (ii) fuerce transición inmediata ante animales; y (iii) adopte un modo *rest-in-red* cuando no hay vehículos, restableciendo paso vehicular al detectarlos. Con soporte en *edge AI*, RTSP y lineamientos regulatorios, el proyecto busca disminuir conflictos, tiempos de espera injustificados y riesgo de siniestros, manteniendo la operación dentro de parámetros seguros y verificables [1]–[10].

## II. Justificación

La movilidad de usuarios vulnerables (peatones y ciclistas) en intersecciones urbanas requiere medidas de control que reduzcan conflictos con vehículos y fortalezcan la seguridad vial. Las normas vigentes (p.ej., el *MUTCD* 11.<sup>a</sup> edición) establecen criterios para señales y tiempos peatonales, lo que respalda que los cambios de fase consideren la presencia efectiva de peatones [1], [2]. En paralelo, la literatura reciente en visión por computador muestra que la detección basada en cámaras —incluidos esquemas multi-cámara— permite estimar trayectorias de peatones y vehículos y analizar conflictos en cruces con precisión suficiente para activar lógicas adaptativas de semáforos [3]–[5].

Frente a los pulsadores peatonales tradicionales, la detección automática reduce tiempos de espera innecesarios y habilita señalización sensible a la demanda. Estudios de intención de cruce (VRU) confirman que la anticipación del comportamiento peatonal/ciclista puede alimentar algoritmos de control más seguros y eficientes [4]. A nivel de infraestructura inteligente, la detección, clasificación, seguimiento y predicción de trayectorias para VRU en intersecciones es una prioridad de investigación aplicada [6].

Desde la perspectiva de implementación, ejecutar la inteligencia artificial en el borde (*edge*) sobre hardware accesible (p.ej., Raspberry Pi con Cámara) y modelos optimizados (*LiteRT*, antes *TensorFlow Lite*) mitiga latencia, dependencia de conectividad, consumo energético y riesgos de privacidad, favoreciendo inferencia local en tiempo real [7], [9]. Para la exposición de video y telemetría, el estándar RTSP y el servidor *gst-rtsp-server* de GStreamer proporcionan un canal estable de transmisión de baja latencia, integrable con VLC u otros clientes [8]. Finalmente, si el entorno incluye fauna urbana, existen enfoques por ejemplo basados en YOLO y variantes que han demostrado mejoras sustanciales en precisión/recobrado para detección de animales, útiles para políticas de prioridad o seguridad [10].

### III. Descripción y síntesis del problema

Se requiere un sistema embebido con (n) cámaras para detectar personas, ciclistas, vehículos y fauna en una calle y forzar transiciones semafóricas bajo políticas específicas que son mencionadas nuevamente en la sección de Requerimientos:

#### Políticas:

- **Personas:** al detectar, iniciar temporizador para transición a rojo; por cada nueva persona detectada reducir el tiempo de cambio en 30 s, con límite inferior igual o mayor al tiempo peatonal mínimo normativo.
- **Animales:** transición *inmediata* a amarillo→rojo (prioridad de seguridad).
- **Sin vehículos:** modo *rest-in-red* para habilitar cruce seguro; retorno a fase vehicular cuando se detecten autos.
- **Ventanas:** paso peatonal 1 min; paso de animales 2 min; reanudación vehicular en  $\leq 1$  min tras detección.

### IV. Recursos

#### IV-A. Hardware

- Raspberry Pi 4/5 ( $\geq 4$  GB), microSD/SSD.
- Cámara (PiCam).
- LEDs R/A/V + resistencias.
- Protoboard/caja, fuente.
- Punto de acceso Wi-Fi.

#### IV-B. Software

- Linux embebido (Yocto) + systemd.
- GStreamer (+ `gst-rtsp-server`) para RTSP.
- OpenCV + TensorFlow Lite.
- Tracker (SORT/OC-SORT), lógica de políticas del semáforo, servicio GPIO.
- Telemetría básica VLC/FFmpeg con un servidor RTSP.
- Integración de flujo con Python.

### V. Requerimientos

- Detección/seguimiento de personas/animales/vehículos.

#### Políticas

- **Animal** → Amarillo-rojo. Transición inmediata (LED A, R).
- **Personas** → Amarillo-rojo. Transición dinámica; tiempo inicial 3 min, reducción dinámica por persona 30 s.
- **Sin vehículos** → Rojo. Tránsito de personas y animales.
- Salida a LEDs (R/A/V) y exposición del RTSP con video en tiempo real.

- Paso de personas 1 min.
- Detectar carros nuevamente → dar paso dentro de 1 min.
- Paso de animales 2 min.

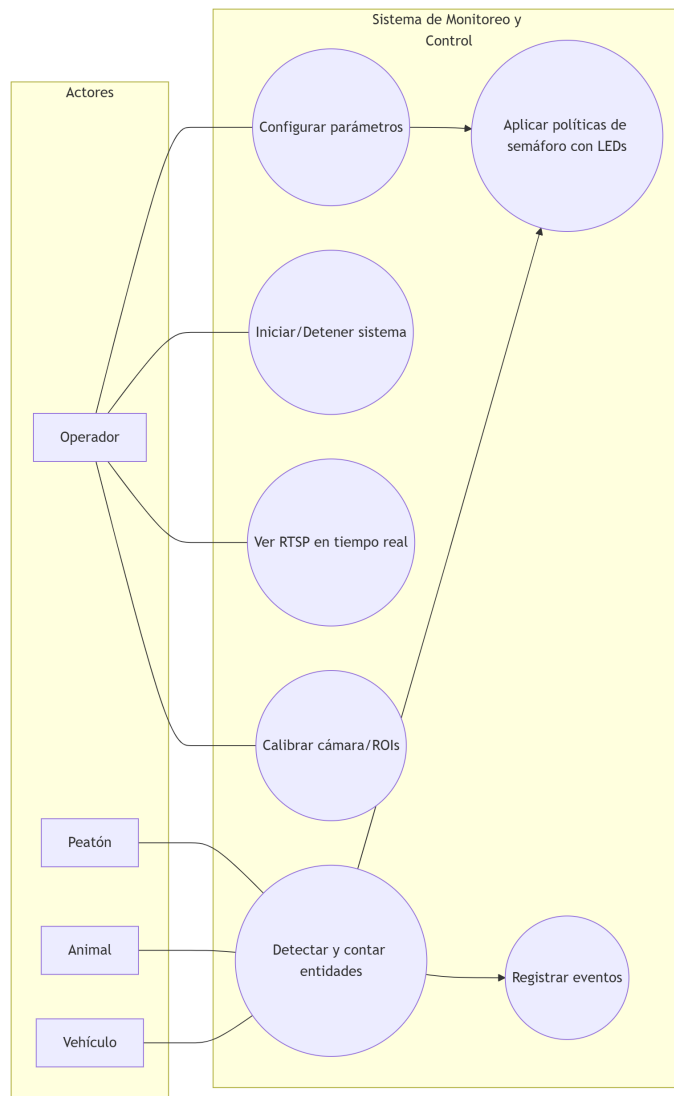


Figura 1: Diagrama de Concepto de Operaciones.

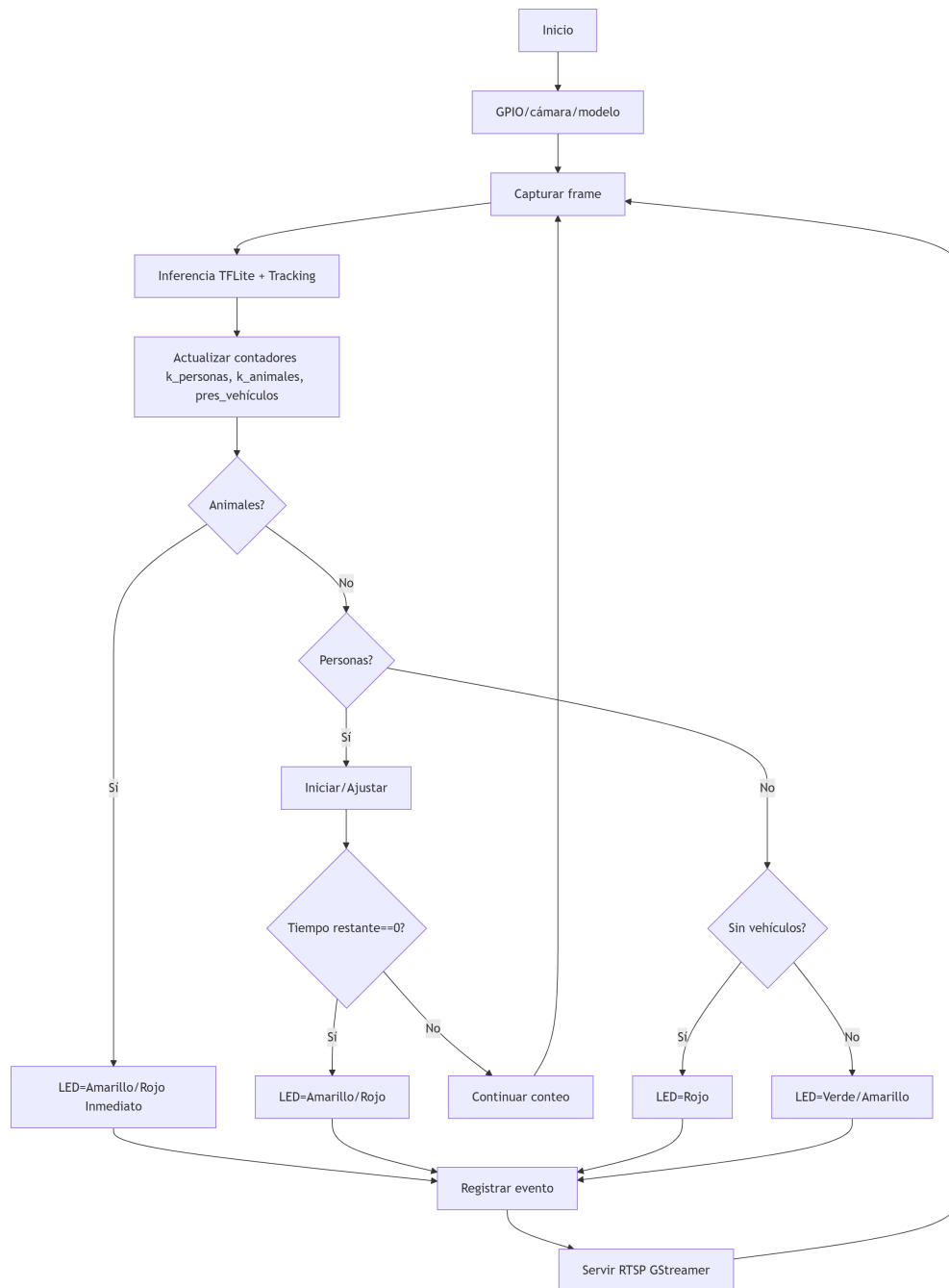


Figura 2: Flujo de trabajo del sistema.

El conjunto de funciones general del sistema contempla:

- El sistema **observa** su entorno (captura).
- **Prepara** los datos visuales para su análisis (procesamiento).
- **Reconoce** lo que ocurre en la escena (detección).
- **Decide** la acción más apropiada según sus políticas (lógica).
- **Actúa** físicamente sobre el semáforo (control).

### **V-A. Función de adquisición y supervisión (Observa)**

El sistema inicia su ciclo funcional con la captura continua de imágenes o video proveniente de las cámaras instaladas en el cruce. Estas cámaras constituyen el principal sensor del sistema, encargada de registrar la dinámica del entorno en tiempo real: tránsito vehicular, peatones y eventualmente fauna.

La Raspberry Pi se configura para gestionar esta adquisición mediante librerías estándar como OpenCV y servicios de transmisión RTSP, que permiten tanto el procesamiento local como la visualización remota del flujo de video. Las cámaras, como el medio captor de datos convierte el escenario de análisis (intersección) en información que será procesada en posteriores etapas, las cámaras son elementos de hardware que tienen conexión directa con raspberry.

### **V-B. Función de procesamiento visual (Prepara)**

Una vez capturadas las imágenes, el sistema ejecuta una etapa de procesamiento visual básico cuyo objetivo es preparar la información para ser interpretada. En este punto, se aplican transformaciones simples como la reducción de resolución, normalización del color o filtrado de ruido, que facilitan la eficiencia del procesamiento posterior.

Esta función constituye un puente entre la adquisición y el análisis: traduce las imágenes del entorno a un formato y estructura que puedan ser comprendidos por los módulos de inferencia, optimizando el uso de los recursos del hardware embebido.

La librería que podría emplearse en esta etapa sería OpenCV y como complemento Gstreamer, donde OpenCv se encargaría de funciones como la captura de los fotogramas, reajuste de escala o la aplicación de filtros. Por otro lado, Gstreamer gestionaría el flujo del video comunicando con etapas posteriores la información preprocesada.

### **V-C. Función de detección y análisis de presencia (Reconoce)**

El siguiente bloque funcional corresponde a la detección y clasificación de los agentes presentes en el cruce.

Aquí el sistema aplica modelos de inteligencia artificial livianos previamente entrenados que identifique personas, vehículos o animales dentro del flujo de video. Se pretende adaptar e integrar soluciones existentes, seleccionadas por su compatibilidad con entornos embebidos y su bajo consumo computacional.

En esta etapa se utilizan las capacidades del modelo para obtener sus resultados: el sistema debe saber cuándo y qué tipo de objeto está siendo detectado para tomar decisiones coherentes. En esta fase, la inferencia se ejecuta localmente gracias a TensorFlow Lite u otras bibliotecas equivalentes que permiten realizar detecciones con rapidez y sin depender de conectividad externa.

### **V-D. Función de interpretación y toma de decisiones (decide)**

Con la etapa anterior, se obtienen una manera de monitorear personas, vehículos o mascotas, ahora se requiere la etapa que utiliza esa información para darle un sentido, un propósito, que es hacer que los resultados de detección sean interpretados por una lógica de control que traduzca la información visual en acciones concretas sobre el semáforo. Esta función constituye el núcleo decisional del sistema, donde se implementan las políticas establecidas:

- Si se detecta la presencia de una o más personas, se activa una fase peatonal con tiempo base, ajustable dinámicamente según la cantidad de peatones.
- Si el sistema identifica animales, se prioriza su paso mediante una transición inmediata a la fase roja, con un margen temporal de seguridad.
- Si no se detectan vehículos durante un período prolongado, el sistema entra en un estado de “reposo en rojo” para evitar cruces peligrosos.

La lógica se diseña de forma modular, de modo que las condiciones y tiempos puedan modificarse fácilmente en el software sin alterar la estructura general. Esta función puede entenderse como la instancia que convierte la información en decisión, manteniendo un equilibrio entre seguridad, fluidez vehicular y sostenibilidad.

La columna vertebral de esta lógica pretende ser desarrollada en lenguaje python. El modelo empleado para detección una herramienta desarrollada por terceros, por lo que no se tiene un control o conocimiento profundo de su funcionamiento, mientras el código de python es completamente desarrollado desde cero para el proyecto. De esta manera el equipo de desarrollo tiene total libertad de modificación y ajuste de la lógica, para cumplir las expectativas de funcionabilidad y adaptabilidad, lo que correspondería a acciones y ordenes.

### **V-E. Función de control físico y retroalimentación (Actúa)**

Una vez determinada la acción correspondiente, el sistema ejecuta la orden a través de la interfaz de control físico, conectada a los LEDs del semáforo mediante los pines GPIO de la Raspberry Pi. Aquí se concretan las decisiones tomadas en las etapas anteriores, transformándolas en señales eléctricas que encienden o apagan los colores rojo, amarillo o verde según corresponda. Para la implementación de estas señales, es precisa la utilización de apatadores de potencia para el control de luces y algunos otros dispositivos electrónicos y eléctricos

Paralelamente, el sistema mantiene un proceso de retroalimentación que registra los eventos (activaciones, tiempos, detecciones) para su posterior análisis o depuración. Esta información puede transmitirse o almacenarse localmente, permitiendo evaluar el desempeño del sistema en diferentes condiciones de operación.

## **VI. Arquitectura del sistema propuesto**

En este proyecto está compuesto por un único nodo embebido formado por la raspberry pi, esta es la unidad central que integra todos los subsistemas, adquiere, procesa, controla y transmite, asegura el flujo coherente desde la apreciación del entorno hasta ejecutar acciones físicas que impactan para dar una solución.

### **VI-A. Flujo interno de información**

El flujo de información dentro de la arquitectura sigue una secuencia lineal con retroalimentación, en la cual cada bloque funcional se comunica con el siguiente mediante interfaces bien definidas:

- La cámara envía un flujo continuo de imágenes al subsistema de adquisición, gestionado por OpenCV.
- Los fotogramas procesados pasan a la etapa de inferencia administrada por TensorFlow Lite, que devuelve etiquetas y coordenadas de detección.
- Los resultados de detección son interpretados por el módulo de decisión implementado en Python, que evalúa bajo las políticas planteadas, políticas como los tiempos de espera o prioridad debido a un accion condicional del entorno observado.
- La salida de control se transmite a través de GPIO, activando las fases del semáforo que corresponden.

### **VI-B. Funcionameinto de la estructura general**

#### **VI-B1. 1. Nivel de percepción (entrada de datos)**

Compuesto por la cámara PiCam conectada a la Raspberry Pi mediante interfaz CSI o USB. Este nivel proporciona el flujo de video en tiempo real que representa el entorno de la intersección. La cámara actúa como sensor óptico y se comunica directamente con el sistema operativo mediante controladores nativos de libcamera o Video4Linux (V4L2).

#### **VI-B2. 2. Nivel de procesamiento e inteligencia (núcleo del sistema)**

: En este nivel se concentra la lógica del sistema embebido. Se ejecuta sobre una distribución de Linux embebido generada con Yocto Project, que integra las dependencias necesarias para la operación autónoma. Los componentes de software principales son:

- GStreamer: funciona como canal interno de comunicación multimedia entre los módulos del sistema. Su tarea es gestionar el flujo de video desde la cámara hacia las librerías de procesamiento e inferencia, garantizando sincronización y estabilidad en la transmisión interna de datos.
- OpenCV: encargada del procesamiento visual básico y del pretratamiento de las imágenes.
- TensorFlow Lite (LiteRT): realiza la inferencia sobre los fotogramas procesados, permitiendo la detección y clasificación de objetos (peatones, animales y vehículos).



- Python: lenguaje base en el que se desarrolla la aplicación principal, encargada de coordinar los módulos de adquisición, inferencia y control.

Este nivel constituye el “cerebro” del sistema, donde la información visual se convierte en decisiones de control conforme a las políticas predefinidas.

### **VI-B3. 3. Nivel de actuación y comunicación (salida de datos)**

En este nivel se materializan las decisiones tomadas por la lógica del sistema. Los pines GPIO de la Raspberry Pi se emplean como interfaz de control directo hacia los LEDs del semáforo (rojo, amarillo y verde). Cada salida digital activa los circuitos de potencia correspondientes, asegurando aislamiento y protección eléctrica. Paralelamente, se mantiene un canal de retroalimentación y registro local, que documenta las detecciones y los eventos de cambio de fase para evaluación posterior.

## **VII. Conclusiones**

### **Integración efectiva de hardware y software embebido**

Como se plantea la arquitectura propuesta, es posible integrar adquisición, procesamiento y detección en tiempo real dentro de un único nodo Raspberry Pi, logrando autonomía operativa y respuesta en tiempo real sin depender de servidores externos.

### **Viabilidad del enfoque Edge AI en control de tránsito**

El uso de infraestructura local reduce la latencia y los riesgos de privacidad asociados a “envío de video”, validando la aplicabilidad del edge computing para entornos urbanos sensibles y de infraestructura pública.

### **Contribución a la seguridad vial**

Las políticas de detección y control definidas prioriza a los peatones y animales mejorando la protección de usuarios vulnerables

## Referencias

- [1] U.S. Federal Highway Administration (FHWA), *Manual on Uniform Traffic Control Devices for Streets and Highways (MUTCD)*, 11th ed., Part 4: Highway Traffic Signals, Dec. 2023. [Online]. Available: [https://mutcd.fhwa.dot.gov/pdfs/11th\\_Edition/part4.pdf](https://mutcd.fhwa.dot.gov/pdfs/11th_Edition/part4.pdf)
- [2] Colorado Department of Transportation, “Chapter 10: Pedestrian Traffic Control,” *Colorado Supplement / Guidelines for Pedestrian Signal Timing*, 2024. [Online]. Available: <https://www.codot.gov/safety/traffic-safety/assets/documents/guidelines-for-pedestrian-signal-timing-v2.pdf>
- [3] A. Mohamed and M. M. Ahmed, “Multi-Camera Machine Vision for Detecting and Analyzing Vehicle–Pedestrian Conflicts at Signalized Intersections: Deep Neural-Based Pose Estimation Algorithms,” *Applied Sciences*, vol. 15, no. 19, art. 10413, 2025. doi: <https://doi.org/10.3390/app151910413> [Online]. Available: <https://www.mdpi.com/2076-3417/15/19/10413>
- [4] A. S. Abdelrahman, Z. Islam, and M. Abdel-Aty, “VRUCrossSafe: Real-Time Prediction of Vulnerable Road User Crossing Intention at Signalized Intersections,” *npj Sustainable Mobility and Transport*, vol. 1, art. 20, 2025. doi: <https://doi.org/10.1038/s44333-025-00037-5> [Online]. Available: <https://www.nature.com/articles/s44333-025-00037-5>
- [5] M. Almaliki, A. Bamaqa, M. Badawy, T. A. Farrag, H. M. Balaha, and M. A. Elhosseini, “Adaptive Traffic Light Management for Mobility and Accessibility in Smart Cities,” *Sustainability*, vol. 17, no. 14, 6462, 2025. doi: <https://doi.org/10.3390/su17146462> [Online]. Available: <https://www.mdpi.com/2071-1050/17/14/6462>
- [6] California Department of Transportation (Caltrans), “Vulnerable Road User: Accurate and Reliable Detection Using Roadside-Assisted Cooperative Driving,” *Research Notes*, May 2025. [Online]. Available: <https://dot.ca.gov/-/media/dot-media/programs/research-innovation-system-information/documents/research-notes/task4067-rns-05-25-a11y.pdf>
- [7] Google AI Edge, “LiteRT overview (formerly TensorFlow Lite),” Documentation, last updated May 19, 2025. [Online]. Available: <https://ai.google.dev/edge/litert>
- [8] GStreamer, “gst-rtsp-server Reference Manual,” 1.24.x. [Online]. Available: <https://gstreamer.freedesktop.org/documentation/gst-rtsp-server/index.html>
- [9] Raspberry Pi Documentation, “AI Camera (Sony IMX500) & libcamera,” 2024–2025. [Online]. Available: <https://www.raspberrypi.com/documentation/accessories/ai-camera.html>
- [10] Y. Cai, K. Tian, L. Ji, *et al.*, “A novel target-oriented enhanced infrared camera trap data screening method,” *Scientific Reports*, vol. 15, 16323, 2025. doi: <https://doi.org/10.1038/s41598-025-00042-1> [Online]. Available: <https://www.nature.com/articles/s41598-025-00042-1>