TEC | Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Cruces Inteligentes con EDGE AI Embebido

Taller de Sistemas Embebidos

Integrantes:

Arias Ortiz Kendy Raquel

Bolaños Campos Ana Elena

Quiros Cisneros Christopher Alfonso

Profesor:

Ing. Johan Carvajal Godinez

21 de octubre de 2025

Índice

1. Introducción

El aumento constante del parque vehicular y la expansión de las ciudades han generado una problemática creciente en la gestión del tránsito y la seguridad vial. En los cruces más concurridos, donde convergen peatones, ciclistas, motocicletas y vehículos particulares, los accidentes y la congestión se presentan con mayor frecuencia debido a la limitada capacidad de los sistemas tradicionales para adaptarse a las condiciones dinámicas del entorno urbano.

En este contexto, la inteligencia artificial embebida (Edge AI) surge como una alternativa tecnológica capaz de ejecutar algoritmos de visión por computador y aprendizaje automático directamente en dispositivos de bajo consumo energético, como el Raspberry Pi. Esta capacidad de procesamiento local permite realizar tareas de detección, clasificación y seguimiento de objetos en tiempo real sin depender completamente de la conectividad a la nube, lo que reduce la latencia y mejora la privacidad de los datos.

El presente proyecto propone el desarrollo de un sistema embebido que funcione como nodo inteligente dentro de una red de monitoreo de tránsito urbano. Cada nodo estará basado en hardware de Raspberry Pi con una cámara periférica y sensores complementarios, ejecutando modelos optimizados de visión artificial con TensorFlow Lite y OpenCV. El objetivo es detectar y clasificar peatones, ciclistas, fauna o vehículos, y con ello ofrecer información útil para la toma de decisiones en la gestión del tránsito y el mejoramiento de la seguridad vial.

El desarrollo de este tipo de sistemas no solo promueve la aplicación práctica de los conocimientos adquiridos en la asignatura de Sistemas Embebidos, sino que también alinea la formación del estudiantado con las tendencias actuales de la industria electrónica, donde convergen la inteligencia artificial, el IoT y la computación de borde.

2. Justificación del Proyecto

Las intersecciones urbanas concentran buena parte de las fricciones de la movilidad: trayectorias impredecibles de peatones y ciclistas, picos de congestión y decisiones de conducción tomadas bajo presión. En este entorno cambiante, los esquemas tradicionales de control —basados en temporizaciones fijas o conteos manuales— resultan insuficientes para anticipar comportamientos y reaccionar con la rapidez que exige la seguridad vial. Incorporar inteligencia en el borde (Edge AI) permite llevar el análisis al lugar donde ocurre el fenómeno, reduciendo la latencia, disminuyendo la dependencia de la nube y resguardando la privacidad de quienes transitan.

Este proyecto propone nodos embebidos basados en Raspberry Pi que ejecutan, en tiempo real, modelos livianos de visión por computador para detectar, clasificar y seguir peatones, ciclistas, fauna y vehículos. Al observar el flujo local con granularidad fina (escena a escena), el sistema puede proveer evidencia cuantitativa para ajustar fases semafóricas, activar alertas preventivas o caracterizar riesgos específicos del cruce (por ejemplo, puntos ciegos peatonales en determinadas horas). Así, la solución trasciende el conteo básico y se orienta a decisiones operativas informadas que impactan directamente en la reducción de incidentes y la mejora de la fluidez.

Desde la perspectiva tecnológica, la iniciativa articula un ecosistema embebido moderno: construcción de una imagen de Linux con Yocto Project, integración de OpenCV y TensorFlow Lite, y despliegue en hardware accesible. Esta combinación habilita ciclos de iteración cortos (medición–ajuste–validación) y una escalabilidad pragmática, pues es factible replicar nodos en múltiples cruces con costos razonables y mantenimiento estandarizado.

El proyecto también posee un alto valor formativo, al poner al equipo frente a retos reales de ingeniería: levantamiento y priorización de requerimientos, diseño de arquitecturas de hardware y software, manejo de dependencias, pruebas en campo y validación contra casos de uso. Así, el estudiantado fortalece competencias clave —diseño, integración, validación y documentación— alineadas con las demandas actuales de la industria electrónica y de sistemas embebidos.

3. Descripción y Síntesis del Problema

El problema que aborda este proyecto no se limita solo al incremento del parque vehicular, sino a la complejidad con que conviven diferentes actores en una intersección urbana. Peatones, ciclistas, automovilistas y hasta fauna se mueven con patrones heterogéneos en espacios reducidos, generando riesgos de colisión e incertidumbre en el flujo de tránsito. Los sistemas actuales —temporizaciones fijas en semáforos y conteos manuales— no son capaces de anticipar comportamientos ni adaptarse a cambios repentinos, por lo que resultan insuficientes para garantizar seguridad y fluidez en estos cruces.

La síntesis de esta problemática señala la necesidad de dotar a los cruces de una visión integral y reactiva. Esto implica desarrollar nodos inteligentes de bajo coste, basados en plataformas como Raspberry Pi, que integren cámaras y sensores para observar la escena local, ejecutar modelos de visión por computador optimizados y generar métricas precisas en tiempo real. Al procesar los datos en el mismo sitio (edge computing), se reducen la latencia y la dependencia de la nube, y se preserva la privacidad de los usuarios. Sin embargo, esta solución exige afrontar desafíos técnicos: adaptar modelos de IA a recursos limitados, integrar hardware heterogéneo y gestionar el sistema operativo mediante herramientas como Yocto Project y TensorFlow Lite.

En esencia, el proyecto plantea diseñar un sistema replicable y escalable capaz de:

- **D**etectar y clasificar en tiempo real peatones, ciclistas, fauna y vehículos.
- **P**roveer datos útiles para ajustes semafóricos, activación de alertas preventivas y caracterización de riesgos específicos de cada cruce.
- **F**acilitar la toma de decisiones operativas y de política pública basadas en evidencias cuantitativas.

Este enfoque no solo responde a una necesidad real del entorno urbano, sino que también representa una oportunidad educativa. La implementación de un nodo embebido con Edge AI obliga a los estudiantes a enfrentarse a retos de selección de hardware, optimización de software y validación de sistemas en condiciones reales, alineándolos con las tendencias de la industria electrónica y de sistemas embebidos.

4. Gestión de Requerimientos

4.1. Requerimientos funcionales (RF)

- **RF1.** Capturar video en tiempo real desde cámara (CSI/USB, V4L2).
- **RF2.** Detectar y **clasificar** vehículos, peatones y fauna (TensorFlow Lite).
- **RF3. Seguimiento (tracking)** de objetos con IDs temporales.
- **RF4. Eventos** por cruce (conteos por clase, timestamps) y agregados (p. ej., por minuto).
- **RF5.** Exponer métricas localmente (CLI/log) y **publicar** a red (HTTP/MQTT).

- **RF6. Arranque autónomo**: servicio de la app al boot (systemd).
- **RF7. Registro de auditoría**: fallos, latencias, FPS y salud del nodo.

4.2. Requerimientos no funcionales (RNF)

- **RNF1.** Latencia "captura→detección→evento" $\leq$ **500 ms** (meta demo).
- **RNF2.** *Throughput* objetivo $\geq$ **10 FPS** sostenidos a 640×480.
- **RNF3.** Robustez: recuperación si la cámara se desconecta/reconecta sin reiniciar.
- **RNF4. Observabilidad**: logs con niveles y métricas (CPU/RAM/FPS/colas).
- **RNF5. Despliegue reproducible** con **Yocto** (OpenCV, TFLite, drivers incluidos).
- **RNF6. Seguridad mínima**: no exponer servicios sin autenticación fuera de la LAN del demo.
- **RNF7. Mantenibilidad**: código/recetas en GitHub con README de build/instalación.

4.3. Interfaces y dependencias

- **I1.** Cámara CSI/USB vía **V4L2**.
- **I2.** Red Ethernet/Wi-Fi para publicación de métricas/eventos.
- **I3.** GPIO opcional (p. ej., LED indicador).
- **D1. OpenCV + TensorFlow Lite**; **D2.** recetas Yocto; **D3.** servicio **systemd**.

4.4. Criterios de aceptación

- **CA1.** RF/RNF documentados y **rastreables** a objetivos.
- **CA2.** Casos de uso demostrados *end-to-end* en Raspberry Pi.
- **CA3.** Diagrama **HW/SW** con funciones → componentes e interfaces.
- **CA4. Build Yocto** reproducible con bitácora y árbol de dependencias.
- **CA5. Imagen bootea**, servicio corre, detección visible y métricas publicadas.
- **CA6. Plan** con hitos alineados a propuesta/demos.

5. Vista Operacional del Sistema

5.1. Administrador

1. **Configurar parámetros del sistema**
   Ajusta umbrales de detección (distancia, confianza de modelo, umbrales de cola), temporización de fases y permisos de red.
   Post: Sistema actualizado y registrado en logs.
2. **Reiniciar o actualizar software**
   Ejecuta reinicio seguro del nodo, actualiza imagen Yocto o paquete de la app Edge AI.
   Post: Nodo operativo con última versión.
3. **Consultar reportes de tráfico**
   Accede a históricos de conteos vehiculares y cruces peatonales.
   Post: Informes listos para análisis y toma de decisiones.

5.2. Peatón

1. **Solicitar paso peatonal**
   Presiona el botón o sensor capacitivo → el sistema agenda la fase segura de cruce.
   Post: Semáforo peatonal verde y temporizador visible.
2. **Alertar situación de riesgo**
   Botón de emergencia u orden manual si se detecta un vehículo invasor.
   Post: Evento registrado y alerta al administrador.
3. **Consultar tiempo restante de cruce**
   Pantalla o voz sintética indica cuánto falta para cerrar la fase.
   Post: Información accesible para personas con discapacidad.

5.3. Vehículo

1. **Detectar aproximación**
   El sistema de visión detecta el vehículo y clasifica tipo (auto, bus, moto, emergencia).
   Post: Demanda vehicular actualizada.
2. **Esperar cambio de color (fase)**
   Si el semáforo está en rojo, mantiene la columna de espera según la cola detectada.
   Post: Tiempo de espera monitoreado y reportado.
3. **Vehículo de emergencia prioritario**
   Detección por sirena/luces → se otorga fase preferente.
   Post: Tránsito priorizado y evento registrado.
4. **Infracción por cruce en rojo**
   Vehículo entra en rojo → el sistema genera evidencia (captura + *timestamp*).
   Post: Reporte enviado al administrador y almacenado en logs.

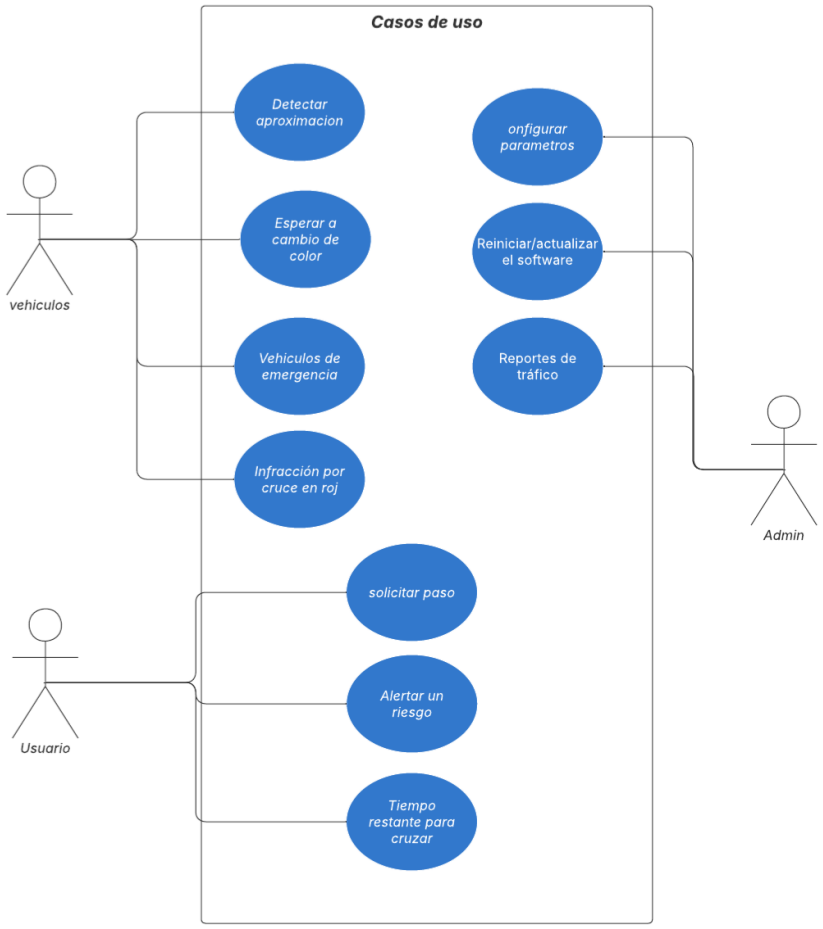

Figura 1: Diagrama de casos de uso.

6. Vista Funcional del Sistema

El sistema *Cruces Inteligentes con Edge AI* se compone de una arquitectura modular que integra componentes de **hardware embebido** y **software inteligente** para cumplir con los requerimientos funcionales del proyecto. Esta vista funcional representa la descomposición de las principales funciones del sistema y su relación con las interfaces e infraestructuras físicas. La arquitectura funcional se organiza en cinco módulos principales: *Captura y Preprocesamiento de Video*, *Procesamiento e Inferencia con Inteligencia Artificial*, *Control del Cruce Inteligente*, *Sistema Embebido y Software Base*, y *Registro y Monitoreo del Sistema*. Cada módulo desempeña un conjunto de funciones orientadas a la adquisición de datos, procesamiento inteligente, toma de decisiones y control en tiempo real del cruce.

6.1. Módulos Funcionales del Sistema

1. **Captura y Preprocesamiento de Video.** Este módulo adquiere el flujo de video en tiempo real mediante cámaras conectadas a la Raspberry Pi, a través de la interfaz USB. Los cuadros capturados son redimensionados, filtrados y normalizados para optimizar el rendimiento del modelo de IA. Además, se segmenta el entorno del cruce en zonas vehiculares, peatonales y de fauna, lo que favorece una clasificación contextual más precisa.
2. **Procesamiento e Inferencia con IA.** Este módulo ejecuta los modelos de detección y clasificación implementados en TensorFlow y OpenCV. Su objetivo es identificar vehículos, peatones y fauna. La toma de decisiones se realiza directamente en el dispositivo, priorizando la seguridad y la eficiencia del tránsito sin depender de conexión a la nube.
3. **Control del Cruce Inteligente.** A partir de las inferencias generadas por la IA, este módulo gestiona las señales del semáforo —simuladas— mediante la activación de salidas con interfaces gráficas. Su función consiste en traducir las decisiones del sistema en acciones concretas, asignando prioridad de paso según la densidad del tráfico y garantizando la respuesta adecuada de los actuadores (virtuales).
4. **Sistema Embebido y Software Base.** Implementado sobre una Raspberry Pi, este módulo proporciona el entorno operativo y de ejecución del sistema. Se basa en una distribución Linux personalizada mediante *Yocto Project*, que incluye las dependencias necesarias (OpenCV, TensorFlow). Adicionalmente, se encarga de la gestión de recursos, la inicialización automática de servicios y el mantenimiento de la operación continua del sistema.
5. **Registro y Monitoreo del Sistema.** Este componente documenta la operación general del sistema con fines de validación y depuración. Registra detecciones, decisiones y estados del cruce, ofreciendo una interfaz de monitoreo local o remoto. Asimismo, permite la exportación de datos hacia almacenamiento local o servicios en red para análisis posterior.

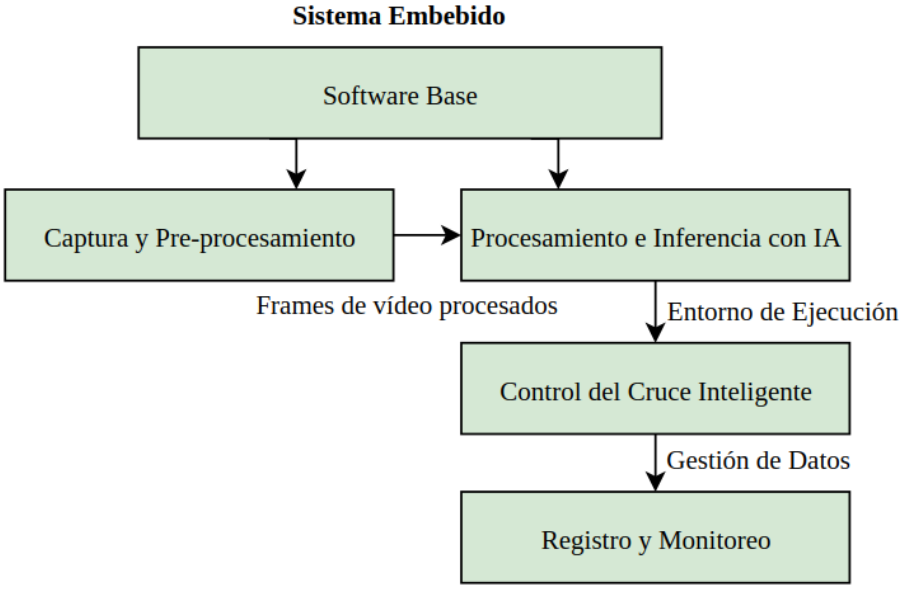

Figura 2: Diagrama funcional del sistema *Cruces Inteligentes con Edge AI*.

Cuadro 1: Descomposición Funcional del Sistema

Módulo Funcional	**Descripción General**	**Funciones Específicas**
Captura y Preprocesamiento de Video	Control del flujo de video proveniente de la cámara conectada a la Raspberry Pi, preparando las imágenes para el análisis de IA.	(1) Captura en tiempo real, (2) Preprocesamiento (redimensionamiento, filtrado, normalización), (3) Segmentación del entorno.
Procesamiento e Inferencia con IA	Detección y clasificación de entidades mediante TensorFlow Lite y OpenCV.	(1) Clasificación de vehículos, peatones y fauna, (2) Estimación de movimiento, (3) Toma de decisiones local.
Control del Cruce Inteligente	Gestión del semáforo según las inferencias de IA.	(1) Activación de señales, (2) Priorización por densidad, (3) Supervisión de actuadores (virtuales).
Sistema Embebido y Software Base	Administración del entorno Linux embebido y sus dependencias.	(1) Integración de dependencias con Yocto, (2) Gestión de recursos, (3) Inicialización y monitoreo de servicios.
Registro y Monitoreo del Sistema	Documentación y visualización de la operación del cruce.	(1) Registro de eventos, (2) Interfaz de diagnóstico, (3) Exportación de datos.

7. Arquitectura del Sistema Propuesto

7.1. Arquitectura de Hardware

El nodo de monitoreo se basa en una plataforma de bajo costo y alta flexibilidad, diseñada para procesamiento de video en tiempo real y ejecución de modelos de Edge AI:

- **Plataforma de Cómputo Embebido:** Raspberry Pi 4 Model B (4GB RAM), capaz de ejecutar TensorFlow Lite y procesar video en tiempo real.
- **Periféricos de Entrada (Visión):** Dos cámaras USB conectadas a la Raspberry Pi para capturar video de alta resolución desde distintos ángulos. Se utilizan puertos USB 3.0 para asegurar ancho de banda suficiente.
- **Periférico de Salida (Visualización):** Pantalla HDMI para mostrar el flujo de video en tiempo real, el estado del semáforo virtual y métricas relevantes del cruce.
- **Almacenamiento:** Tarjeta MicroSD de 32GB (Kingston Canvas Select Plus), suficiente para el sistema operativo, modelos de IA y datos temporales.
- **Conectividad:** Módulo Ethernet/WiFi integrado para comunicación de métricas y gestión remota del nodo dentro de la red de monitoreo.

7.2. Integración Hardware–Software

La Figura 3 muestra la relación entre los módulos de software y los componentes físicos del sistema. Las cámaras proporcionan el flujo de video al módulo de captura, cuyos datos son procesados por los algoritmos de IA en la Raspberry Pi. Los resultados se traducen en señales de control para el semáforo

(virtual), mientras que el sistema embebido coordina la ejecución continua y registra los eventos del sistema.

Cada función del sistema se encuentra asociada a uno o más componentes físicos y lógicos. La interfaz **de adquisición de datos** conecta las cámaras con el módulo de captura de video, transfiriendo cuadros de imagen hacia el sistema embebido para su procesamiento. La interfaz **de procesamiento e inferencia** ejecuta los modelos de IA dentro del entorno embebido, generando decisiones basadas en las detecciones de vehículos, peatones o fauna. Estas decisiones son transmitidas a la interfaz **de control virtual**, responsable de simular el comportamiento del semáforo mediante un entorno gráfico o una representación visual en pantalla. Finalmente, las interfaces **de registro y monitoreo** comunican los resultados y estados del sistema hacia la base de datos local o hacia una interfaz remota para diagnóstico y análisis. Cada una de estas interfaces responde a requerimientos funcionales específicos definidos en el documento de requerimientos, tales como: RF-01 (detección de vehículos), RF-02 (detección de peatones y fauna), RF-03 (control automatizado del semáforo simulado) y RF-04 (registro y supervisión del sistema).

Cuadro 2: Mapeo entre requerimientos funcionales y módulos del sistema

Requerimiento	**Descripción**	**Módulo asociado**
RF-01	Detección de vehículos	Procesamiento e Inferencia con IA
RF-02	Detección de peatones y fauna	Procesamiento e Inferencia con IA
RF-03	Control automatizado del semáforo simulado	Control del Cruce Inteligente (virtual)
RF-04	Registro y supervisión del sistema	Registro y Monitoreo del Sistema

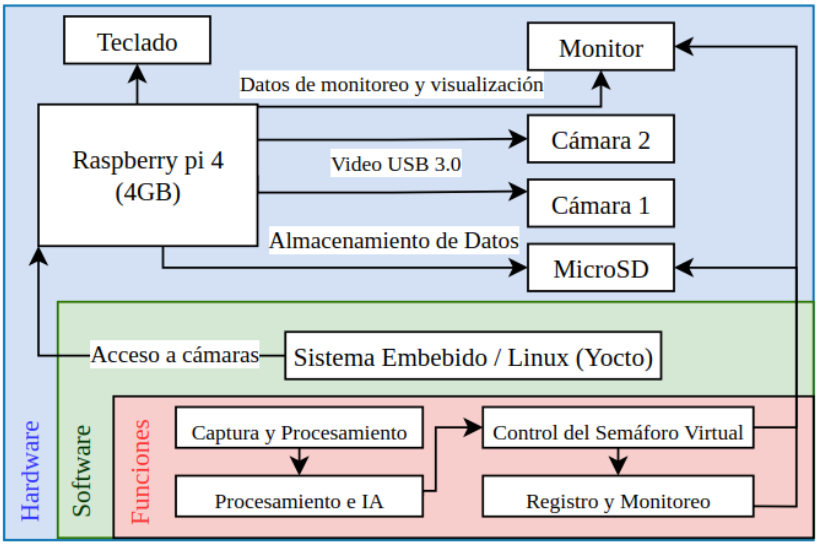

Figura 3: Arquitectura del sistema *Cruces Inteligentes con Edge AI.*

8. Análisis de Dependencias

Para la implementación de la imagen del sistema operativo con Yocto, es importante identificar y analizar los paquetes de software necesarios y sus relaciones. A continuación se presenta un análisis de dependencias simplificado junto con su descripción.

8.1. Tabla de Dependencias

Dependencia	Tipo	Propósito
TensorFlow	Runtime de ML	Ejecución de modelos de detección y clasificación en el Edge.
OpenCV	Librería de Visión	Manipulación de video y preprocesamiento de imágenes para el modelo de ML.
Python 3	Intérprete/Librerías	Lenguaje base para el desarrollo del código de la aplicación.
Drivers de Cámara	Kernel/Drivers	Interfaz para la comunicación con el módulo de cámara de la Raspberry Pi.

Cuadro 3: Paquetes de software y sus propósitos para la imagen Yocto.

8.2. Árbol de Dependencias

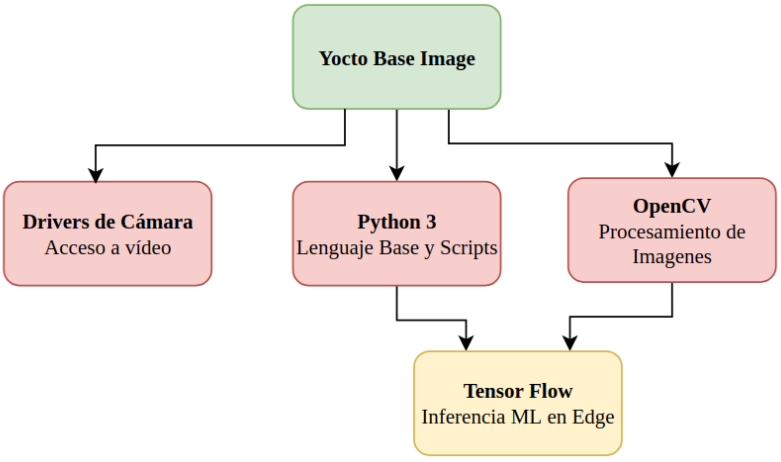

Figura 4: Árbol de dependencias preliminar de la imagen Yocto para el sistema

El siguiente esquema ilustra cómo se relacionan estas dependencias para la construcción y ejecución del sistema:

- **Yocto Base Image**
  - **Python 3** → necesario para la ejecución de scripts y la integración de librerías de ML.
  - **Drivers de Cámara (V4L2)** → necesarios para que la aplicación acceda al hardware de la cámara.

- **OpenCV** → depende de Python 3 para scripts y de los drivers de cámara para captura de video.
- **TensorFlow** → depende de Python 3 y de OpenCV para preprocesamiento de imágenes antes de la inferencia de ML.

Este análisis asegura que la imagen construida con Yocto incluya todos los paquetes necesarios y que sus relaciones estén correctamente definidas para la ejecución del sistema en el Edge.

9. Estrategia de Integración de la Solución

La estrategia de integración asegura que la aplicación de **Edge AI** corra de manera correcta sobre un sistema operativo optimizado y personalizado para la **Raspberry Pi 4 Model B**. Se busca empaquetar todo en una **imagen única**, lista para deploy y ejecución en el nodo de monitoreo.

9.1. 1. Identificación de Recetas

- Analizar las dependencias necesarias para el proyecto, incluyendo TensorFlow, OpenCV y librerías adicionales como `numpy`, `libjpeg` y `zlib`.
- Identificar o sintetizar las recetas (`.bb files`) que permitan compilar estas dependencias para la arquitectura ARM de la Raspberry Pi.

9.2. 2. Generación de Imagen Base

- Configurar Yocto Project para el target **Raspberry Pi 4 B**.
- Generar una imagen mínima de Linux embebido con soporte para CPU, GPU, USB 3.0, HDMI y WiFi/Ethernet.

9.3. 3. Inclusión de Dependencias

- Modificar la receta de la imagen final para incluir todas las librerías y frameworks identificados.
- Asegurar la compilación correcta para ARM y realizar pruebas de compatibilidad.

9.4. 4. Integración de la Aplicación Principal

- Crear la receta para la aplicación de monitoreo del cruce inteligente.
- Configurar la aplicación para que se ejecute automáticamente al iniciar el sistema operativo y controle las cámaras, semáforo virtual y envío de métricas.

9.5. 5. Síntesis de Imagen Final

- Ejecutar `bitbake` para compilar la imagen final, que incluirá Linux embebido, librerías y la aplicación de Edge AI.
- Validar que la imagen generada sea funcional y estable.

9.6. 6. Despliegue y Verificación

- Instalar la imagen en la MicroSD de la Raspberry Pi.
- Conectar las cámaras, la pantalla HDMI y verificar la operación completa del nodo (procesamiento de video en tiempo real, visualización del semáforo virtual, registro de métricas).

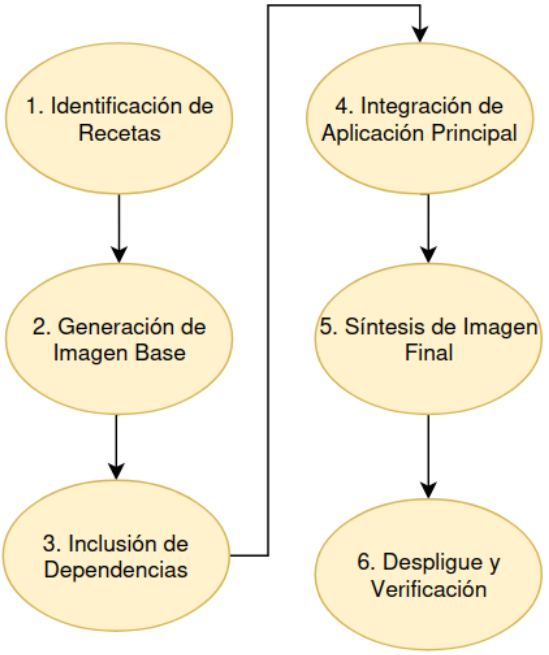

Figura 5: Flujo de integración de la solución *Edge AI* para el nodo de monitoreo.

10. Planeamiento de la Ejecución

Para el planeamiento de la ejecucion del proyecto, se realizo un diagrama de Gantt con los respectivos task a desarrollar. `https://estudianteccr-my.sharepoint.com/:x:/g/personal/acostchris_estudiantec_cr/ERtFBhxp_XxPtcSzFqkqFTgBtX6mbRqnva7ExeMKVMnOEw?e=FxJHmr`

11. Conclusiones

- **Diseño integral de nodo Edge AI:** Se definió una arquitectura funcional y modular para el sistema de cruces inteligentes, incluyendo los módulos de captura de video, procesamiento con IA, control del cruce, sistema embebido y registro/monitoreo.
- **Planificación de la integración:** Se estableció una estrategia clara para generar la imagen de Linux embebido con Yocto Project, incluyendo todas las dependencias necesarias y la aplicación de monitoreo, garantizando un despliegue reproducible en Raspberry Pi 4.
- **Cumplimiento de requerimientos preliminares:** La propuesta asegura que los requerimientos funcionales y no funcionales estén contemplados en el diseño, permitiendo un seguimiento y

trazabilidad hacia futuras implementaciones.

- **Documentación y visualización de flujos:** Se elaboraron diagramas funcionales, de casos de uso y de integración que permiten comprender la interacción entre módulos y la síntesis de la solución final, facilitando futuras etapas de desarrollo.
- **Base para escalabilidad y mejoras:** La arquitectura propuesta permite planificar la expansión del sistema a múltiples nodos y la incorporación de nuevas funciones de IA, sin comprometer la estructura general del diseño.

12. Bibliografía

Referencias

[1] Haofeng Wang y Yan Zhang. *Research on pedestrian detection algorithm in dense scenes*. En *Proceedings of the 4th International Conference on Computer, Artificial Intelligence and Control Engineering (CAICE '25)*, páginas 35–41, ACM, Nueva York, NY, Estados Unidos, enero de 2025. DOI: 10.1145/3727648.3727655.

[2] Chenxu Li y Chunmei Wang. *Analysis According to the Algorithm of Pedestrian Vehicle Target Detection in Hazy Weather Based on Improved YOLOv8*. En *Proceedings of the 2024 7th International Conference on Artificial Intelligence and Pattern Recognition (AIPR '24)*, páginas 35–39, ACM, Nueva York, NY, Estados Unidos, septiembre de 2024. DOI: 10.1145/3703935.3704068.

[3] Anilcan Bulut, Fatmanur Ozdemir, Yavuz Selim Bostanci y Mujdat Soyturk. *Performance Evaluation of Recent Object Detection Models for Traffic Safety Applications on Edge*. En *Proceedings of the 2023 5th International Conference on Image Processing and Machine Vision (IPMV '23)*, páginas 1–6, ACM, Nueva York, NY, Estados Unidos, enero de 2023. DOI: 10.1145/3582177.3582178.

[4] Woojae Kim e Inbum Jung. *Smart Parking Lot Based on Edge Cluster Computing for Full Self-Driving Vehicles*. *IEEE Access*, volumen 10, páginas 115271–115281, 2022. DOI: 10.1109/ACCESS.2022.3208356.

[5] Komal Saini y Sandeep Sharma. *Smart Road Traffic Monitoring: Unveiling the Synergy of IoT and AI for Enhanced Urban Mobility*. *ACM Computing Surveys*, volumen 57, número 11, artículo 276, páginas 1–45, junio de 2025. DOI: 10.1145/3729217.

[6] Kasra Aminiyeganeh y Rodolfo W. L. Coutinho. *Performance Evaluation of CNN-based Object Detectors on Embedded Devices*. En *Proceedings of the Int'l ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications (DIVANet '23)*, páginas 55–60, ACM, Nueva York, NY, Estados Unidos, octubre de 2023. DOI: 10.1145/3616392.3623417.

[7] Ching-Lung Su, Wen-Cheng Lai, Yu-Kai Zhang, Ting-Jia Guo, Yi-Jiun Hung y Hui-Chiao Chen. *Artificial Intelligence Design on Embedded Board with Edge Computing for Vehicle Applications*. En *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, páginas 130–133, IEEE, Laguna Hills, CA, EE. UU., diciembre de 2020. DOI: 10.1109/AIKE48582.2020.00026.

[8] Reenu Mohandas, Mangolika Bhattacharya, Mihai Penica, Karl Van Camp y Martin J. Hayes. *TensorFlow Enabled Deep Learning Model Optimization for Enhanced Realtime Person Detection Using Raspberry Pi Operating at the Edge*. En *Proceedings of the 28th Irish Conference on*

Artificial Intelligence and Cognitive Science (AICS 2020), vol. 2771 de *CEUR Workshop Proceedings*, páginas 157–168, Dublín, Irlanda, 2020. Disponible en: https://ceur-ws.org/Vol-2771/AICS2020_paper_61.pdf. (Consulta: 20 de octubre de 2025).

[9] Leevi Oranen. *Utilizing Deep Learning on Embedded Devices*. Tesis de máster, Tampere University, Faculty of Medicine and Health Technology, Tampere, Finlandia, 2021. Disponible en: https://trepo.tuni.fi/bitstream/handle/10024/133689/OranenLeevi.pdf?sequence=2.

[10] Lightsout. *Capturing video from two cameras in OpenCV at once*. Pregunta en Stack Overflow, 16 de abril de 2015. Disponible en: https://stackoverflow.com/questions/29664399/capturing-video-from-two-cameras-in-opencv-at-once. (Consulta: 20 de octubre de 2025).

[11] K. Mohamed Haris, N. Sabiyath Fatima y Syed Abdallah Albeez. *Advanced Vehicle Detection Heads-Up Display with TensorFlow Lite*. En S. Shakya, V. E. Balas y H. Wang (eds.), *Proceedings of the Third International Conference on Sustainable Expert Systems*, vol. 587 de *Lecture Notes in Networks and Systems*, páginas 631–647, Springer, Singapur, 2023. DOI: 10.1007/978-981-19-7874-6_47.