Manual técnico

Sistema de movilidad y transporte inteligente

Kevin Evaristo Estrada Curiel

Dylan Fabricio Merlos Sánchez

Miguel Ángel Velasco Álvarez

Marco Antonio Velásquez González

TOLUCA, ESTADO DE MEXICO - JUNIO 2025

2. Índice

3. Introducción

Este manual proporciona las directrices necesarias para implementar y operar un sistema de semáforos inteligentes basado en inteligencia artificial. El sistema está diseñado para optimizar el flujo vehicular y mejorar la seguridad peatonal en zonas urbanas, mediante el uso de sensores, cámaras y algoritmos de aprendizaje automático. Además, incluye procedimientos para el mantenimiento preventivo y correctivo, asegurando la continuidad y eficiencia operativa del sistema. Se establecen pautas para la detección temprana de fallos, la reparación oportuna y la actualización de software y hardware, con el fin de maximizar la vida útil y confiabilidad del sistema.

Asimismo, se contemplan medidas rigurosas de seguridad y protección de datos, garantizando la privacidad de la información recopilada, la integridad de los datos y el cumplimiento de normativas legales vigentes. Esto incluye protocolos de encriptación, control de accesos y auditorías regulares para prevenir accesos no autorizados o vulnerabilidades que puedan comprometer la operación del sistema o la confidencialidad de los usuarios.

4. Descripción del Sistema

El sistema se compone de:

- Semáforos con control dinámico.
- Sensores ultrasónicos y cámaras LIDAR.
- Módulo de procesamiento con Raspberry Pi o servidores.
- Aplicación web para monitoreo y configuración.

- Red de comunicación Mesh.
- Algoritmo de aprendizaje por refuerzo para optimización en tiempo real.

Funciones principales:

- Ajuste automático de luces según flujo vehicular.
- Detección de peatones y ciclistas.
- Priorización del transporte público.
- Generación de reportes y análisis de tráfico.
- Integración con sistemas de control urbano.

5. Requisitos del Sistema

a) Requisitos de Hardware

- Raspberry Pi 4 o superior
- Módulo de cámara (compatible con visión artificial)
- Sensor ultrasónico HC-SR04
- Módulo de relé para control de luces
- Router con soporte Mesh
- Servidores locales o en la nube (opcional)

b) Requisitos de Software

- Raspbian OS / Linux
- Python 3.9+

- OpenCV
- TensorFlow o PyTorch
- YOLOv5
- Pandas
- Flask (para dashboard)
- Base de datos PostgreSQL / MySQL

c) Infraestructura

- Fuente de energía continua (UPS opcional)
- Red 5G / WiFi estable
- Poste o estructura para montar sensores y cámaras

6. Instalación del Sistema

6.1. Preparación

- Verifica que todos los dispositivos funcionen correctamente.
- Asegúrate de que la red Mesh esté activa y con señal estable.

6.2. Instalación de Hardware

1. Fija los semáforos y sensores en su posición física.
2. Conecta los sensores al microcontrolador (ej. Raspberry Pi).
3. Instala las cámaras con visión frontal al cruce.
4. Conecta el módulo de relé al sistema de luces.

6.3. Instalación de Software

1. Instala los paquetes requeridos (OpenCV, TensorFlow, etc.).
2. Configura el algoritmo de IA.
3. Conecta el servidor al sistema de bases de datos.
4. Ejecuta la interfaz web y valida la conexión con semáforos.

7. Funcionamiento

7.1. Modo Automático

- Detecta en tiempo real el volumen de vehículos.
- Ajusta el tiempo del semáforo dinámicamente.
- Prioriza peatones, ciclistas o transporte público cuando se detectan.

7.2. Modo Manual

- Acceso desde el dashboard por parte de autoridades.
- Permite ajustes en tiempo real o apagado del sistema.

7.3. Reportes

- Genera gráficos de congestión, eficiencia y emisiones.
- Exporta reportes en PDF/CSV.

8. Mantenimiento

8.1. Mantenimiento Preventivo

- Limpieza de sensores y cámaras: semanal.
- Revisión de conexiones eléctricas: mensual.
- Verificación del servidor y red: quincenal.

8.2 Mantenimiento Correctivo

El mantenimiento correctivo tiene como propósito **restaurar el funcionamiento del sistema** ante una falla, y se basa en la **inspección de voltajes, señales de control, comunicación y funcionamiento de sensores y actuadores**.

🔌 8.2.1 Revisión de Voltajes

a) Componentes alimentados a 5V

- **Dispositivos**: sensores ultrasónicos (HC-SR04), relés, algunos módulos de comunicación.
- **Rango normal**: **4.8V a 5.2V**
- **Instrumento**: multímetro digital en escala DC (VDC)
- **Cómo medir**:
    1. Colocar la punta negra en el GND del módulo.
    2. Colocar la punta roja en el pin VCC.

- **Si está fuera del rango**:
    - < 4.8V: posible caída de voltaje, resistencia en cables, alimentación débil.
    - 5.2V: fuente mal calibrada o falla en regulador → puede dañar componentes.

b) Componentes alimentados a 3.3V

- **Dispositivos**: Raspberry Pi, módulos de comunicación GPIO, algunos sensores.
- **Rango normal**: **3.2V a 3.4V**
- **Cómo medir**:
    - Igual que el procedimiento anterior, usando multímetro.
- **Fuera de rango**:
    - < 3.2V: puede causar errores en lectura/detección.
    - 3.4V: riesgo de sobrevoltaje → quemado de GPIO.

📶 8.2.2. Verificación de Señales Digitales y PWM

a) Señales digitales (ON/OFF)

- **Ejemplo**: pin de activación de un relé, salida de sensor.
- **Instrumento**: **multímetro** (modo voltaje) o preferentemente **osciloscopio**.
- **Rango esperado**:
    - Nivel alto (HIGH): **> 3.0V** (3.3V o 5V según el sistema)

- Nivel bajo (LOW): **< 0.5V**

- **Interpretación**:
  - Si se mantiene en estado fijo (siempre HIGH o LOW): posible bloqueo o mal código.
  - Señal fluctuante: confirmar si es esperada o es ruido.

b) Señales PWM (modulación de ancho de pulso)

- **Ejemplo**: control de servos, atenuación de LEDs.
- **Instrumento**: **osciloscopio**
- **Parámetros clave**:
  - **Frecuencia típica**: 1 kHz a 10 kHz (según aplicación)
  - **Ciclo de trabajo (Duty cycle)**: 0–100%
- **Rango esperado**: depende de la programación. Debería observarse una onda cuadrada estable.
- **Falla común**:
  - Señal distorsionada o con ruido → problema de conexión a tierra o interferencia.

📡 8.2.3. Verificación de Señales de Comunicación

a) UART / Comunicación Serie

- **Usado en**: comunicación Raspberry–Arduino o sensores.
- **Instrumento**: osciloscopio o analizador lógico.

- **Parámetros esperados**:
    - Voltaje típico: 3.3V o 5V (según dispositivo)
    - Baudrate común: 9600, 115200 bps
- **Falla común**:
    - No hay señal: cable roto, fallo en TX/RX, error de configuración.
    - Datos corruptos: diferencia de baudrate o mala conexión GND.

b) I2C / SPI

- **Instrumento recomendado**: **analizador lógico**
- **Indicadores de falla**:
    - Ausencia de señal en SDA/SCL (I2C) o MOSI/MISO/SCK (SPI).
    - Frecuencia y sincronía no acorde con protocolo.

🔦 8.3.4. Diagnóstico de Sensores

a) Sensor ultrasónico HC-SR04

- **Voltaje alimentación**: 5V
- **Salida de pulso (ECHO)**: se mide con osciloscopio
- **Parámetro esperado**:
    - Pulso proporcional a la distancia → 58 µs por cada cm.

- **Fuera de rango**:
    - Sin respuesta → sensor dañado o no recibe TRIG.
    - Pulso irregular o errático → interferencia o falla de hardware.

b) Cámara

- **Verificar imagen en software** (OpenCV, VLC)
- **Falla típica**:
    - Imagen congelada o negra → problema de alimentación o drivers.
    - Imagen distorsionada → interferencia o lente sucio.

🔄 8.2.5. Interpretación y Acciones Correctivas

Parámetro	**Valor Esperado**	**Diagnóstico si está fuera del rango**
Tensión 5V	4.8V – 5.2V	Alimentación insuficiente o daño en regulador
Tensión 3.3V	3.2V – 3.4V	Riesgo de sobrevoltaje o bajo rendimiento
Señal digital (HIGH/LOW)	>3V / <0.5V	Código fallando, cortocircuito, GPIO dañado
Señal PWM	Onda cuadrada variable	Fallo en configuración, hardware, o interferencia

Parámetro	Valor Esperado	Diagnóstico si está fuera del rango
UART / TX-RX	Trama estable	Fallas de comunicación o sincronización
Echo HC-SR04	Pulso proporcional	No hay respuesta = sensor dañado o sin trigger
Cámara	Imagen clara	Imagen negra o errores = problema de energía o lente

🧰 8.2.6. Herramientas Recomendadas

- **Multímetro digital**
- **Osciloscopio digital (100 MHz mínimo recomendado)**
- **Analizador lógico (opcional para I2C/SPI/UART)**
- **Pinzas, cautín, soldadura, alcohol isopropílico**
- **Software de diagnóstico (Putty, Arduino IDE, Python)**

9. Seguridad

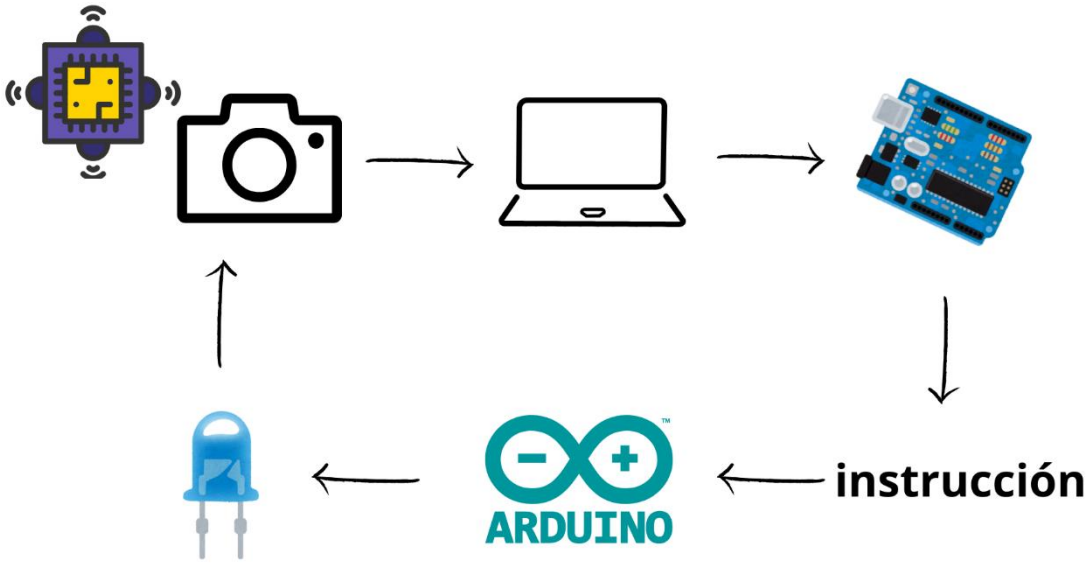
- El sistema incluye protocolos de seguridad de red (TLS/HTTPS).
- El acceso al dashboard requiere autenticación de usuario.
- Los datos son cifrados antes de ser almacenados.
- En caso de fallas graves, el sistema retorna al modo seguro (ciclo fijo de semáforo).

10. Especificaciones Técnicas

Componente	Especificación
Procesador Principal	Raspberry Pi 4 / Servidor Intel Xeon
Conectividad	WiFi, Ethernet, Mesh
Algoritmos	YOLOv5, Reinforcement Learning (DQN)
Precisión IA	≥ 85% detección de objetos en tiempo real
Latencia máxima	≤ 1 segundo
Autonomía UPS	30 minutos
Sensores	HC-SR04, LIDAR, cámaras HD

11. Diagramas

- Diagrama de flujo del sistema



- Diagrama de secuencia
- Diagrama de arquitectura
- Diagrama entidad-relación

12. Glosario

IA: Inteligencia Artificial

YOLOv5: Algoritmo de detección de objetos en tiempo real

Raspberry Pi: Microcomputadora de bajo consumo

Relé: Dispositivo para controlar circuitos de alto voltaje

LIDAR: Sensor óptico para medición de distancias

Dashboard: Interfaz gráfica de monitoreo

13. Anexos

- Manual de instalación del software
- Capturas del dashboard
- Resultados de pruebas de validación
- Reportes de tráfico simulados