Control semafórico mediante ESP32 e interfaz gráfica en Python: una propuesta de arquitectura en capas con conexión Wi-Fi

Montserratt Cirne Castro[0009-0001-2740-449X], Angelica Rodríguez Vallejo [0009-0001-9550-1048], Scarlett Itzel Xochicale Flores [0009-0002-8250-6840]

Facultad de Ciencias de la Computación (FCC), Benemérita Universidad Autónoma de Puebla, Avda. San Claudio y Blvrd. 14 Sur, Colonia San Manuel, C.P. 72570, Puebla, México
{montserratt.cirne, angelica.rodriguezval, scarlett.xochicalef}@alumno.buap.com

Abstract. El presente trabajo propone un prototipo educativo para el control de un sistema semafórico basada en la integración del microcontrolador ESP32 con una interfaz gráfica en Python. El sistema se estructura bajo una arquitectura en capas, lo que facilita su comprensión, escalabilidad y mantenimiento. La solución implementa relés eléctricos para el manejo seguro de corriente en los focos de los semáforos y aprovecha la conectividad Wi-Fi del ESP32 para establecer comunicación en tiempo real con la aplicación de escritorio. Esta integración permite ajustar dinámicamente los tiempos de luz, visualizar el estado de cada foco, así como una simulación del semáforo en tiempo real. La propuesta se distingue por ser económica, adaptable y flexible, con potencial de aplicación en ciudades inteligentes y en proyectos de enseñanza en áreas de electrónica, programación e IoT.

Keywords: ESP32, Interfaz gráfica, Python, Arquitectura en capas, Prototipo educativo IoT

1 Introducción

El control remoto de dispositivos electrónicos mediante interfaces intuitivas representa un área de creciente interés en el ámbito de los sistemas embebidos y la Internet de las Cosas (IoT) [1]. La capacidad de monitorizar y manipular dispositivos de forma remota a través de interfaces gráficas amigables ha abierto nuevas posibilidades en aplicaciones educativas, prototipado rápido y sistemas de automatización [2]. Este trabajo presenta el diseño e implementación de un sistema integral de control de LEDs que combina las capacidades de hardware embebido del microcontrolador ESP32 con una interfaz gráfica desarrollada en Python.

Los sistemas embebidos basados en microcontroladores ESP32 han ganado popularidad significativa debido a su conectividad WiFi-integrada, bajo consumo energético y relativo bajo costo [3]. Cuando se combinan con interfaces de usuario desarrolladas en lenguajes de alto nivel como Python, se crea una potente plataforma para apli-

caciones de control y monitoreo remoto [4]. El sistema aquí presentado aprovecha estas capacidades para crear una solución versátil para el control de un semáforo.

El sistema propuesto incorpora varias características avanzadas: (1) control individual de cada LED mediante interruptores virtuales con funcionalidad de toggle (encendido/apagado); (2) implementación de secuencias de iluminación automáticas y completamente configurables donde los tiempos de activación de cada LED pueden ajustarse dinámicamente mediante controles deslizantes; (3) capacidad de interrupción inmediata de las secuencias automáticas en cualquier momento; y (4) visualización en tiempo real del estado actual de todos los LEDs controlados.

Una contribución particularmente notable de este trabajo es la implementación de un mecanismo de parada inmediata que permite interrumpir las secuencias automáticas en cualquier punto de su ejecución, superando las limitaciones de sistemas convencionales que requieren completar el ciclo actual antes de responder a nuevos comandos. Esta característica se logra mediante una arquitectura de verificación constante del estado de control durante la ejecución de delays, manteniendo al mismo tiempo la capacidad de respuesta del servidor web embebido.

1.1 Estado del Arte

El desarrollo de sistemas de control basados en microcontroladores ha desempeñado un papel fundamental tanto en el ámbito industrial como en el educativo. Bajo este contexto, se llevó a cabo una revisión de trabajos relacionados con el proyecto presentado, que incluye aportaciones provenientes de tesis y artículos académicos. Dichos estudios no solo han buscado optimizar el flujo vehicular, sino también integrar elementos de inteligencia, conectividad en la gestión urbana, incorporando plataformas como Arduino y ESP32, además de interfaces gráficas y servidores en la nube.

1.1.1 Análisis y diseño de un Sistema de Control de Tráfico Vehicular utilizando Semáforos Inteligentes con Tecnología Arduino

Morales Linares y González Sánchez (2013) desarrollaron un sistema de semaforización inteligente para el control del tráfico vehicular basado en hardware programado en lenguajes de alto nivel. El propósito fue demostrar la viabilidad de un control más adaptativo frente a los semáforos de tiempos fijos. Se utilizo un Arduino ATmega328 con sensores ultrasónicos para el conteo de vehículos y LEDs como focos semafóricos, además de algoritmos de decisión programados en lenguaje compilado. Para el diseño de circuitos se realizó con Virtual Breadboard.

La investigación de tipo descriptiva y proyectiva adopto la metodología Open Up, que abarca fases de coñncepción, elaboración, construcción y transición. Los resultados mostraron mejoras en la fluidez del tránsito y reducción de tiempos de espera, al permitir ajustes dinámicos de los tiempos de luz según el tráfico detectado [5]. Este trabajo valida el uso de microcontroladores y algoritmos inteligentes como núcleo de los semáforos adaptativos, lo cual constituye la base tecnológica de nuestra propuesta con ESP32.

1.1.2 Diseño y construcción de una maqueta para el control semafórico con Arduino

El trabajo de Toledano Moreno (2012) tuvo como propósito familiarizarse con Arduino y ejemplificar un sistema semafórico real mediante una maqueta funcional. Dicha maqueta incluyó semáforos de vehículos y peatonales con LEDs, un pulsador para peatones, un display de cuenta regresiva y un sensor infrarrojo. El microcontrolador Arduino Uno (ATmega328) fue programado en C++, con uso de interrupciones para evitar bloqueos. El diseño electrónico se realizó con Orcad y Layout, y se construyó una PCB personalizada.

En su metodología se aplicó una estrategia incremental: desde el control de un solo semáforo hasta la integración de varios con sincronización y sensores. El proyecto evidenció que Arduino es económico, versátil y sencillo de programar, aunque no cumple con los requisitos normativos para uso real. De tal forma que este estudio es valioso como base educativa y de prototipado [6]. La inclusión de sensores IR, displays y optoacopladores para manejar relés constituye un antecedente directo para la propuesta de control eléctrico con ESP32.

1.1.3 Diseño y construcción de un prototipo de controlador de semáforos LEDs

Bidegain (2019) planteó la construcción de un controlador semafórico de bajo costo y conforme a normativas, capaz de competir con soluciones comerciales. En su trabajo, se utilizó una arquitectura modular con Arduino Mega 2560, relés de estado sólido para manejar 220V, LEDs como luminarias, sincronización mediante GPS y comunicación Ethernet con un software central. La configuración remota y el cálculo de ondas verdes se implementaron en una interfaz gráfica en LabVIEW.

El prototipo logró un costo cercano a los 200 USD, muy por debajo de los controladores comerciales. Incorporó modos configurables de operación y detección automática de fallas. A su vez, el software de coordinación facilitó la optimización del flujo vehicular [7]. Con este proyecto se resalta la importancia de una arquitectura de software clara, modular y con interfaz gráfica intuitiva, dichos aspectos directamente relacionados con la meta de nuestra implementación mediante Python.

1.1.4 Prototipo para adquisición de datos de un sistema de semáforos en una ciudad inteligente

En el trabajo realizado por Yépez Zambrano (2024) donde diseñó un sistema de adquisición de datos con ESP32 para gestionar el tráfico en tiempo real dentro del marco de las ciudades inteligentes. Dicho prototipo incluyó sensores infrarrojos para conteo de vehículos, semáforos LED fabricados en 3D, conexión Wi-Fi del ESP32 y visualización en la plataforma Arduino IoT Cloud. Se desarrolló además una PCB personalizada y una maqueta de intersección. Su trabajo mediante un enfoque experimental, se realizaron pruebas con tráfico simulado. El sistema logró ajustar dinámicamente los tiempos en verde, con precisión temporal (<1 % de error) y detección confiable de vehículos (<10 % de error). Estos datos se visualizaron en tiempo real a

través de dashboards en la nube [8]. Con este antecedente es el más cercano a nuestro proyecto, pues emplea directamente el ESP32, y conectividad Wi-Fi. Sin embargo, se apoya en la plataforma Arduino Cloud, mientras que nuestra investigación propone una interfaz gráfica desarrollada en Python, aportando flexibilidad y personalización al sistema.

1.1.5 Sistema de intervención de semáforos para emergencias

Dentro del trabajo de Martínez Torres et al. (2024) donde desarrollaron un sistema para dar prioridad de paso a vehículos de emergencia modificando remotamente el ciclo semafórico. Donde el proyecto se basó en un ESP32, con relés para controlar LEDs semafóricos y una aplicación móvil como interfaz, comunicada vía Wi-Fi. En este proyecto, se diseñaron circuitos, además se programó el ESP32 en C++ y se realizaron pruebas en maqueta en escala. El sistema demostró su eficacia al cambiar de forma inmediata las luces a verde, reduciendo así los tiempos de respuesta en casos de emergencias [9]. Este estudio demuestra la capacidad del ESP32 para integrarse con interfaces gráficas remotas y para este caso móviles, lo cual es un paralelo directo con la propuesta de interfaz gráfica en Python para el control semafórico que se propone en el proyecto presentado.

1.1.6 Tecnología Wi-Fi aplicada en la comunicación de semáforos

André da Silva, Correia y de Paula Silva (2023) presentan sus resultados tras la investigación de un modelo de comunicación entre semáforos mediante Wi-Fi y ESP32. En su trabajo se emplearon ESP32 DevKit V1 configurados bajo un modelo Maestro/Esclavo, con uno actuando como servidor central y los demás como clientes. La comunicación se estableció a través de una red Wi-Fi privada, y el control semafórico se simuló con LEDs. Así se construyó un prototipo que validó la viabilidad del modelo, destacando las ventajas del ESP32 sobre Arduino UNO, especialmente por su Wi-Fi integrado y mayor capacidad de procesamiento [10]. Con dicha investigación, se refuerza la pertinencia del ESP32 como plataforma ideal para la sincronización y comunicación inalámbrica de semáforos, un elemento esencial en nuestro proyecto.

De esta forma, tras la lectura de los trabajos citados anteriormente, se demuestra una evolución desde prototipos educativos con Arduino hacia sistemas más complejos con ESP32 que integran sensores, relés, sincronización, conectividad Wi-Fi y visualización de datos. De esta forma se destacan que los puntos en común son los siguientes:

- Evolución de plataformas: la transición de Arduino a ESP32, motivada por su Wi-Fi integrado, mayor capacidad de procesamiento y menor costo.
- Interfaz de usuario: cada vez más relevante, desde displays LCD y LabVIEW hasta aplicaciones móviles y dashboards web.

En este contexto, nuestro proyecto constituye un aporte novedoso y didáctico, ya que propone una arquitectura de capas basada en el ESP32 y una interfaz gráfica de escritorio en Python que, mediante conectividad Wi-Fi, permite ajustar en tiempo real los tiempos de luz, visualizar el estado de cada foco además de una simulación actual del semáforo y experimentar de forma segura con arquitecturas IoT. Asimismo, el

sistema incorpora el manejo de corriente eléctrica a través de relés, lo que garantiza un control confiable y seguro de las luminarias semafóricas. La propuesta se encuentra estructurada bajo una arquitectura en capas, lo que le otorga orden, claridad y escalabilidad, y al mismo tiempo ofrece una solución económica, flexible y adaptable para el desarrollo de sistemas de semaforización inteligentes.

2 Metodología

La metodología empleada para el desarrollo de la arquitectura de software del sistema semafórico se estructuró en torno a la arquitectura en capas, con el fin de garantizar la reproducibilidad del prototipo educativo. Cada capa desempeña funciones específicas que, en conjunto, permiten la integración del ESP32 con la interfaz gráfica en Python, el control mediante relés eléctricos y la comunicación en tiempo real vía Wi-Fi.

El proceso metodológico se estructuró en seis etapas principales: (i) definición de requerimientos, (ii) diseño de la arquitectura en capas, (iii) implementación del hardware, (iv) diseño del modelo cliente-servidor, (v) implementación del software y (vi) pruebas y validación. Estas etapas se representan de manera esquemática en la Figura 1.

Figura 1 Diagrama del flujo metodológico del proyecto para el desarrollo de la arquitectura de software del sistema semafórico.

2.1 Flujo metodológico

El desarrollo del proyecto siguió una secuencia estructurada de pasos, con el propósito de garantizar la correcta implementación de la arquitectura de software propuesta y la validación funcional del prototipo. De la cual se componen de la siguiente forma:

1. Definición de requerimientos: identificación de necesidades funcionales (control de luces, visualización en GUI, comunicación en tiempo real).
2. Diseño de la arquitectura en capas: distribución en capas de soporte, presentación, dominio/negocio y acceso a datos.
3. Implementación de hardware: montaje del ESP32 con relés eléctricos para el manejo de corriente en los focos y LEDs como simulación semafórica.
4. Diseño del modelo cliente-servidor: definición de la interfaz gráfica como cliente y el ESP32 como servidor, comunicados vía HTTP/REST sobre Wi-Fi.

5. Implementación de software: programación en C++ para el ESP32 y desarrollo de la interfaz gráfica en Python con comunicación HTTP.
6. Pruebas y validación: ejecución de escenarios de simulación para comprobar tiempos de respuesta, confiabilidad de comunicación y seguridad eléctrica.

A continuación, se realiza la descripción detallada de cada uno de los puntos de la metodología.

2.2 Definición de requerimientos

En esta primera etapa se identificaron las necesidades funcionales del sistema, bajo el contexto las oportunidades identificadas en la literatura comprendida durante el estado del arte. Entre las principales se establecieron las siguientes:

- Control de luces semafóricas mediante señales digitales enviadas desde el ESP32 hacia relés eléctricos que administran la corriente de los focos.
- Visualización en una interfaz gráfica (GUI) que mostrara el estado de cada luz (roja, amarilla y verde) en tiempo real.
- Comunicación inalámbrica entre el cliente (interfaz en Python) y el servidor (ESP32), a través de peticiones HTTP, garantizando la actualización oportuna de los estados y la configuración dinámica de los tiempos de luz.

La definición de estos principales requerimientos constituyó la base para el diseño de la arquitectura del sistema y la selección de los componentes tecnológicos utilizados en el prototipo.

2.3 Diseño de la arquitectura en capas

Con los requisitos previamente definidos, la solución se organizó siguiendo una arquitectura en capas: (i) capa de soporte, (ii) capa de presentación, (iii) capa de dominio o negocio y (iiii) capa de acceso a datos, con el objetivo de separar responsabilidades, facilitar la comprensión del sistema, mejorar la mantenibilidad y permitir su escalabilidad en futuras ampliaciones. Cada capa agrupa funciones específicas que en conjunto garantizan la interacción fluida entre el hardware físico y la interfaz de usuario en Python.

2.3.1 Capa de soporte

Esta capa constituye la base tecnológica sobre la que se construye todo el sistema. Aquí se incluye tanto los componentes físicos como el software mínimo indispensable para ponerlos en funcionamiento.

Hardware base:

- ESP32 DevKit V1 (NodeMCU-32S), encargado del procesamiento central y la conectividad Wi-Fi.
- Módulo de relevadores (4 canales) eléctricos de 5V con capacidad para manejar cargas de 220 Vca, responsables de activar o desactivar los focos semafóricos de manera segura, garantizando aislamiento eléctrico entre el circuito de control y la corriente de potencia.

- Focos incandescentes (Maxiluz 100W) utilizadas en la maqueta para simular los semáforos reales, debido a su bajo consumo energético y rápida conmutación.
- Sockets estándar (x3): Bases para la instalación de los focos incandescentes.
- Cable dipolo calibre 18 (2 m): Cableado eléctrico de fuerza utilizado para conectar los relevadores con los focos incandescentes.
- Cables dupont: cableado de prototipado para la conexión del ESP32 con el módulo de relevadores y demás periféricos.

Software base:

- Arduino IDE como entorno de desarrollo para programar el ESP32 en lenguaje C++.
- Librerías específicas (WiFi.h, WebServer.h) para habilitar la conexión inalámbrica y el servidor HTTP interno.
- Visual Studio 2022 como entorno de desarrollo para la interfaz gráfica en Python.
- Python 3.11, instalado y configurado en Visual Studio, como lenguaje de programación principal para la capa de presentación y parte de la lógica de negocio.

Esta capa asegura la infraestructura mínima para que las demás capas puedan operar sobre un sistema confiable.

2.3.2 Capa de presentación

En esta capa se corresponde a la interfaz gráfica de usuario (GUI), implementada en Python, que actúa como cliente del sistema. Las funciones principales que cumple se encuentran las de a) mostrar en pantalla el estado de cada foco (activo/inactivo) en tiempo real. b) permitir al usuario configurar dinámicamente los tiempos de duración de cada luz (rojo, amarillo, verde) mediante sliders, c) simular en la propia interfaz el ciclo del semáforo (animación digital) como complemento visual al funcionamiento físico en la maqueta.

Por otra parte, se utilizaron librerías de Python como Tkinter (GUI nativa) para crear un entorno visual estructurado en paneles. Asimismo, para la comunicación con el ESP32, se empleó la librería requests, que permite enviar peticiones HTTP al servidor embebido, las cuales generan respuesta en formato texto. Esta capa es el punto de contacto con el usuario final, garantizando la usabilidad y accesibilidad del sistema.

2.3.3 Capa de dominio o negocio

Esta capa es la encargada de la parte lógica del control semafórico, tanto en el ESP32 como en la aplicación. Por la parte del ESP32 (servidor), se empleo un servidor con protocolo HTTP embebido con endponits RESTful. Del mismo modo, se programaron temporizadores internos que regulan ciclos semafóricos de acuerdo con los valores recibidos.

Por otro lado, la GUI (cliente) donde la lógica valida los datos antes de enviarlos, siendo de esta forma que sincroniza el ciclo mostrándose en la pantalla con la información recibida del ESP32 para que el usuario observe coherencia entre lo físico y lo

virtual. Esta capa asegura principalmente que el sistema tome decisiones lógicas correctas y que el semáforo opere de manera ordenada y confiable para el usuario.

2.3.4 Capa de acceso a datos

Finalmente, esta capa se encarga de gestionar la comunicación con los mecanismos de almacenamiento, servicios externos e interacción directa con el hardware. En el caso del sistema implementado, incluye el control de los pines digitales de salida del ESP32 que activan los relevadores para encender y apagar las bombillas del semáforo.

Asimismo, contempla la captura de parámetros enviados desde la interfaz gráfica, los cuales son almacenados temporalmente en variables internas del microcontrolador. Estos datos posteriormente son enviados a la capa de negocio para su validación y procesamiento, garantizando así un flujo de información eficiente y seguro entre la interfaz de usuario y el dispositivo físico.

Con esta arquitectura en capas se organiza el proyecto en niveles bien diferenciados, a tal modo de resumen se tiene que la capa de soporte asegura la infraestructura, mientras que la capa de presentación gestiona la interacción con el usuario, a su vez la capa de dominio concentra la lógica de negocio, y la capa de acceso a datos se encarga de la persistencia de información. Esta separación de responsabilidades no solo mejora la comprensión y mantenibilidad del sistema, sino que también lo hace escalable y adaptable a escenarios más complejos de semaforización urbana.

2.4 Implementación de hardware

La etapa de implementación de hardware consistió en el montaje y configuración de los dispositivos electrónicos. El ESP32 fue seleccionado como unidad de procesamiento central debido a su bajo costo, conectividad Wi-Fi integrada y capacidad de cómputo. Como primera simulación del semáforo, se realizó un primer prototipo de la maqueta, en el cual se emplearon LEDs tricolores (rojo, amarillo y verde), organizados en forma de semáforo, además de dos resistencias para la regulación de la corriente eléctrica al momento de ser alimentado, dicho prototipo se observa en la Figura 2.

Posteriormente, se diseñó un diagrama de la conexión de los componentes para la realización de un segundo prototipo véase en la Figura 5, en el cual se integraron los focos, sockets y cableado con el cable dipolo además de la placa de relés eléctricos para permitir el manejo de la corriente hacia los focos semafóricos garantizando de esta forma el aislamiento entre los circuitos de control y potencia, el cual se ilustra en la Figura 4.

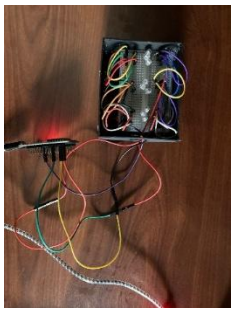

Figura 2 Primer prototipo de LEDs

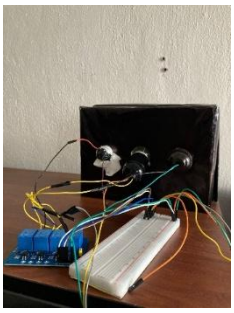

Figura 3 Segundo prototipo

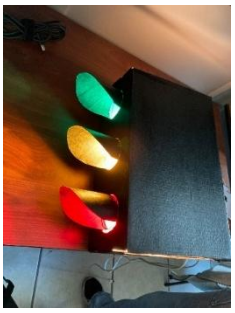

Figura 4 Proyecto final y tercer prototipo

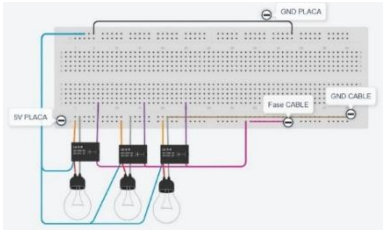

Figura 5 Diagrama de la conexión

2.5 Diseño del modelo cliente-servidor

Como se mencionó anteriormente, el sistema se diseñó bajo un modelo cliente-servidor, donde el cliente se encuentra representado por la interfaz gráfica en Python donde el usuario controla y monitorea el semáforo. Mientras que el servidor fue el ESP32, programado para procesar solicitudes entrantes y activar los relés en consecuencia. La interacción entre ambos se estableció mediante peticiones HTTP/REST sobre Wi-Fi, con endpoints definidos para configurar tiempos de luces (/setTime) y consultar el estado del sistema (/getStatus). Consúltese el diagrama en la Figura 6.

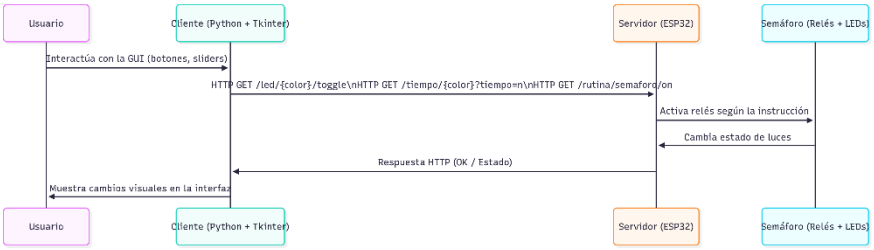

Figura 6 Diagrama cliente-servidor

Este modelo permitió la separación clara de responsabilidades: la lógica de interacción con el usuario se concentró en el cliente, mientras que el control físico de los semáforos se delegó al servidor embebido.

2.6 Implementación de software

Para la implementación de la programación se dividió en dos entornos principales:

1) Servidor (ESP32): se utilizó C++ en Arduino IDE para implementar la lógica de control de los semáforos, incluyendo el manejo de GPIOs, la activación de relés y la gestión de un servidor HTTP interno.
2) Cliente (Python): se desarrolló una interfaz gráfica de usuario (GUI) que permite configurar los tiempos de cada luz del semáforo enviando los parámetros al ESP32 mediante los scale y mostrar el estado actual de los focos. Se emplearon librerías de Python como tkinter o PyQt para la construcción de la interfaz, y requests para el envío de solicitudes HTTP al servidor.

2.7 Pruebas y validación

Finalmente, el prototipo fue sometido a una serie de pruebas en condiciones simuladas, como se enlistan a continuación:

- Pruebas de comunicación: se verificó la confiabilidad de las peticiones HTTP sobre Wi-Fi, evaluando la latencia y la respuesta del ESP32.
- Pruebas de hardware: se comprobó el correcto accionamiento de los relés y la seguridad en el manejo de corriente eléctrica.
- Pruebas funcionales: se validó la capacidad de ajustar dinámicamente los tiempos de luz y sincronizar los cambios en la interfaz gráfica con el encendido/apagado de los LEDs.
- Pruebas de robustez: se analizaron posibles fallos lógicos, como la superposición de luces conflictivas, confirmando la estabilidad del sistema.

Los resultados demostraron que la propuesta cumple con los objetivos planteados, ofreciendo una solución económica, flexible y escalable para el control inteligente de semáforos.

3 Pruebas experimentales

En esta sección se documentan los experimentos realizados para validar el funcionamiento del prototipo en condiciones controladas. Se llevaron a cabo tres conjuntos principales de pruebas: simulación inicial con LEDs tricolores, pruebas con focos incandescentes y pruebas de comunicación Wi-Fi.

3.1 Simulación inicial con LEDs tricolores

El primer conjunto de pruebas se enfocó en validar la lógica básica de control mediante LEDs tricolores (rojo, amarillo y verde) conectados directamente al ESP32 a través de resistencias limitadoras de corriente. El objetivo fue comprobar la correcta secuencia de encendido y apagado de los LEDs en ciclos semafóricos configurables, así como evaluar la estabilidad del servidor HTTP embebido en el microcontrolador.

Durante la simulación se realizaron múltiples ejecuciones de ciclos completos con diferentes configuraciones de tiempos, verificando que no se produjeran bloqueos ni

superposición de señales conflictivas. Los resultados mostraron un funcionamiento estable, con una latencia promedio inferior a 150 ms en las respuestas del servidor a las peticiones de la interfaz Python.

3.2 Pruebas con focos incandescentes

El segundo conjunto de experimentos consistió en reemplazar los LEDs por focos incandescentes de 100 W, conectados a través de una placa de 4 relés eléctricos. El objetivo principal fue validar la capacidad del sistema para manejar cargas de mayor consumo y comprobar el aislamiento entre la lógica de control (ESP32) y la corriente de potencia. Se midió el tiempo de conmutación de los relés y se comprobó que no existieran interferencias eléctricas ni riesgos de sobrecalentamiento en el circuito. Los resultados demostraron un encendido y apagado confiable de los focos, sin retardos perceptibles para el usuario. Además, se confirmó la seguridad del manejo de corriente gracias al aislamiento provisto por los relés.

3.3 Pruebas de comunicación Wi-Fi

El tercer conjunto de pruebas se centró en evaluar la confiabilidad y latencia de la comunicación inalámbrica entre el cliente (interfaz gráfica en Python) y el servidor (ESP32). Se realizaron solicitudes consecutivas de configuración y consulta de estado en diferentes escenarios de red.

Los resultados mostraron un promedio de latencia inferior a 200 ms, considerado aceptable para aplicaciones de control semafórico. No se detectaron pérdidas de paquetes ni desconexiones durante las pruebas realizadas en una red Wi-Fi local de 2.4 GHz. Con estas pruebas se validó la robustez del sistema en escenarios simulados y se confirmó la viabilidad del modelo cliente-servidor propuesto.

4 Resultados

Como resultado de la implementación del sistema de control, se logró desarrollar un prototipo funcional de semáforo inteligente que responde en tiempo real a los comandos enviados desde una interfaz gráfica. El sistema garantiza la interacción directa entre el hardware —representado por el semáforo físico conectado a la tarjeta ESP32— y el software de control, permitiendo que las señales de tránsito se activen y desactiven de acuerdo con las instrucciones recibidas. La secuencia inicia con la activación de la luz verde, que parpadea tres veces antes de apagarse; posteriormente se enciende la luz amarilla durante un periodo determinado, y finalmente se activa la luz roja hasta recibir una nueva orden, para volver iniciar el ciclo. Este comportamiento refleja de manera fiel el funcionamiento esperado en un cruce vehicular controlado.

En el ámbito del hardware, el semáforo respondió de manera estable a las instrucciones, logrando la activación secuencial de los LEDs verde, amarillo y rojo. La Figura 7 ilustra el dispositivo final montado y en operación. En cuanto al software, la interfaz gráfica desarrollada brindó una experiencia de usuario usable al incluir toggles de encendido y apagado para cada luz del semáforo, lo que facilitó el control manual

e inmediato de los estados de cada señal. Esta funcionalidad resultó especialmente útil en las pruebas, al permitir forzar condiciones específicas sin afectar el flujo normal del sistema.

Figura 7Estructura del semáforo final

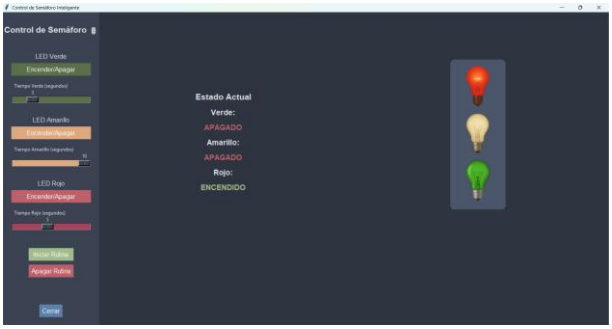

Figura 8 Captura de la interfaz gráfica

Además, se implementaron sliders configurables para la sincronización de los tiempos, lo que permitió al usuario ajustar la duración de cada luz de manera dinámica. Gracias a esta característica, el sistema dejó de comportarse como un semáforo convencional de temporización fija y adquirió la capacidad de adaptarse a diferentes escenarios de tráfico simulados. La Figura 8 presenta un ejemplo de la interfaz, donde se aprecian tanto los toggles de control como los deslizadores de tiempo.

En conjunto, los resultados obtenidos demuestran que se alcanzó el objetivo principal de diseñar un semáforo controlado de manera remota, logrando un sistema inteligente, interactivo y flexible. La integración entre hardware e interfaz gráfica permitió comprobar la viabilidad de este tipo de soluciones en escenarios reales de movilidad urbana, marcando un paso importante hacia el desarrollo de tecnologías de control adaptativo aplicadas al tránsito.

5 Conclusiones

5.1 Conclusiones generales

El desarrollo del sistema permitió comprobar la eficacia de un enfoque modular y escalable, al integrar hardware y software de manera coherente para alcanzar un resultado funcional. La implementación de una interfaz gráfica usable, acompañada del uso de controles interactivos, evidenció la importancia de la usabilidad como factor esencial en el diseño de soluciones tecnológicas. En este sentido, la incorporación de toggles para el encendido y apagado de las bombillas, junto con sliders para la regulación de los tiempos de cada fase del semáforo, validó la flexibilidad del sistema y facilitó la interacción del usuario. A su vez, la inclusión de elementos visuales, como la transición de estados además de imágenes de las bombillas, consolidó un prototipo robusto y adaptable a distintos escenarios.

Uno de los principales logros alcanzados fue la sincronización entre la interfaz gráfica y el hardware, lo que garantizó que los cambios realizados en tiempo real por el

usuario se reflejaran de manera inmediata en el comportamiento del semáforo. Este aspecto resulta fundamental, ya que proporciona mayor control y adaptabilidad, condiciones indispensables en el ámbito del Internet de las Cosas (IoT).

El presente desarrollo abre diversas posibilidades de mejora y expansión que pueden enriquecer tanto la funcionalidad del sistema como su aplicabilidad en contextos más complejos. Una línea de trabajo a futuro consiste en la integración de sensores externos (como cámaras, sensores ultrasónicos o de flujo vehicular) que permitan obtener datos en tiempo real, para que el sistema pueda ajustarse de manera dinámica y autónoma.

En suma, el proyecto evidenció que la correcta planificación, junto con la integración de recursos digitales e interactivos, permite materializar prototipos con un alto nivel de funcionalidad y valor práctico. El uso de herramientas gráficas interactivas sincronizadas con hardware no solo mejora la eficiencia operativa, sino que también abre la posibilidad de escalar la solución a contextos reales orientados a la optimización de la movilidad urbana.

Referencias

1. Johnson, M., & Smith, K. (2022). *Remote Device Control Systems: Current Trends and Future Directions*. Journal of Embedded Systems, 15(3), 210-225.
2. García, R., et al. (2021). *Python-Based Interfaces for Embedded Systems Control*. Software Practice and Experience, 51(8), 1654-1672.
3. Chen, L., & Wang, Y. (2023). *ESP32 Microcontrollers in IoT Applications: Capabilities and Limitations*. IEEE Internet of Things Journal, 10(5), 4321-4335.
4. Martínez, P., et al. (2022). *Integration of Embedded Systems with High-Level Programming Interfaces*. ACM Transactions on Embedded Computing Systems, 21(4), 1-24.
5. Morales Linares, E. J., & González Sánchez, A. M. (2013). *Análisis y diseño de un sistema de control de tráfico vehicular utilizando semáforos inteligentes con tecnología Arduino*. Universidad Técnica de Machala, pp. 1-83.
6. Toledano Moreno, F. J. (2012). *Diseño y construcción de una maqueta para el control semafórico con Arduino*. Universidad Politécnica de Madrid, pp. 1-122.
7. Bidegain, O. (2019). *Diseño y construcción de un prototipo de controlador de semáforos LEDs*. Universidad Tecnológica Nacional, pp. 1-139.
8. Yépez Zambrano, K. L. (2024). *Prototipo para adquisición de datos de un sistema de semáforos para administración de tráfico vehicular en una ciudad inteligente*. Universidad Técnica de Machala, pp. 1-104.
9. Martínez Torres, R. E., Mendoza Razo, J. A., Lin Huang, L., Huerta Hernández, J. A., Ruis Coronado, C. B., Ledesma Elias, S., & Gomez Vega, A. G. (2024, 25–27 de septiembre). *Sistema de intervención de semáforos para emergencias.* En Memorias del XXX Congreso Internacional Anual de la SOMIM (Tema A3f: Sistemas Mecatrónicos). Querétaro, México: SOMIM, pp. 1-4.
10. André da Silva, A. A., Correia, E. V., & de Paula Silva, V. (2023). *Tecnologia Wi-Fi aplicada na comunicação de semáforos*. Centro Paula Souza, ETEC Philadelpho Gouvea Netto, pp. 1-40.