# Semáforo Inteligente

Rojas Pérez José Gabriel [1][0009-0005-1227-7335], Lozano Cruz José Ángel [2][0009-0009-3034-924X] y Jimenez Bonilla Giovani [3][0009-0009-7064-477X]

[1] facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, Puebla 72570, México.

**Abstract.** Este proyecto se centra en el diseño y desarrollo de una interfaz gráfica que facilite la interacción entre un usuario y una computadora, mediante la simulación y control de un semáforo inteligente implementado en el aula. El propósito principal es integrar conceptos de la arquitectura por capas dentro de un sistema que permita observar tanto la estructura como el comportamiento del dispositivo. Para lograrlo, se emplearon diversas herramientas tecnológicas: por un lado, el editor de código Arduino IDE, utilizado para programar y cargar las instrucciones en la placa ESP32, la cual actúa como un mecanismo de control del semáforo; y, por otro lado, con el lenguaje Python se construyó la interfaz gráfica que posibilita la comunicación con los componentes del semáforo. Dicha conexión se estableció a través de redes inalámbricas mediante Wi-Fi, lo que brinda flexibilidad y escalabilidad al sistema, constituyendo un ejemplo práctico de integración entre hardware y software.

**Palabras clave:** Inteligente, semáforo, cliente, computadora, arquitectura, servidor, capas, conexión.

## 1 Introducción

En la actualidad se han empleado sistemas inteligentes que permiten la interacción con el ser humano durante toda su vida cotidiana. Estos sistemas facilitan tareas previamente programadas con el objetivo de simplificar las actividades humanas y optimizar el tiempo. Un ejemplo claro de ello es la apertura automática de puertas mediante sensores de proximidad o el uso de comandos de voz para controlar dispositivos inteligentes como Alexa de Amazon. Gracias a este tipo de tecnologías, el ser humano ha logrado una evolución en la forma de comunicarse con las máquinas, logrando que estas realicen tareas específicas sin la necesidad de la presencia física de un operador. Sin embargo, la mayoría de los usuarios utiliza estas tecnologías sin tener un conocimiento real de cómo funcionan internamente, ya que suelen limitarse a comprender únicamente la función superficial sin conocer los componentes y procesos que las hacen posibles.

La importancia de la automatización en la actualidad radica en su capacidad para optimizar procesos, reducir errores humanos y aumentar la eficiencia en diversos ámbitos. Desde la industria manufacturera hasta los sistemas de transporte, la automatización se ha convertido en un pilar esencial para el desarrollo tecnológico de las sociedades

modernas. En el caso particular de las grandes ciudades, una de las problemáticas más comunes es el tráfico vehicular, el cual no solo afecta la movilidad de los ciudadanos, sino que también incrementa el riesgo de accidentes, en especial de tipo peatonal. Ante esta situación, los semáforos inteligentes se presentan como una solución viable y efectiva, ya que permiten regular los tiempos de circulación de manera dinámica y remota, adaptándose a las necesidades del entorno.

Este proyecto busca acercarnos al entendimiento y aplicación de estos conceptos a través del diseño y construcción de un semáforo inteligente. Dicho sistema fue implementado con relevadores, focos de bombilla y una placa ESP32, la cual actúa como el controlador central del dispositivo. Además, se desarrolló una interfaz gráfica en Python que permite controlar los tiempos de encendido y apagado de las luces del semáforo, simulando una rutina de circulación vial. Gracias a la conexión inalámbrica mediante Wi-Fi, este proyecto no solo demuestra la viabilidad técnica de un sistema automatizado, sino que también resalta la importancia de integrar diferentes áreas del conocimiento como la electrónica, la programación y el diseño de interfaces.

Finalmente, cabe destacar que este trabajo no se limita únicamente a replicar el funcionamiento visible de un semáforo, sino que también busca profundizar en los conocimientos sobre los componentes y la lógica interna que hacen posible su operación. La implementación de estilos arquitectónicos, como la arquitectura por capas, permite organizar de manera eficiente las distintas funciones del sistema y comprender cómo se relacionan entre sí. En este sentido, el proyecto constituye un ejemplo práctico de la integración entre hardware y software, así como una oportunidad para analizar cómo la automatización puede ser aplicada en entornos reales, contribuyendo al aprendizaje y a la innovación tecnológica.

## 2. Metodología

Para realizar este proyecto fue necesario construir un semáforo con componentes físicos, estos componentes nos ayudaron a construir la estructura del semáforo que nos ayuda a simular un semáforo real para entender cuál es la lógica o los procesos involucrados para hacer que un semáforo funcione en la vida real, como se establecen los tiempos o como es que pueden sincronizarse en caso de que estuviera en un crucero.

### 2.2. Componentes físicos

Para este proyecto se ocuparon los componentes:

- ESP32: es una placa de desarrollo con un microcontrolador de bajo costo y alto rendimiento fabricado por Espressif. Cuenta con conectividad Wi-Fi y Bluetooth integradas, lo que lo hace ideal para proyectos de Internet de las Cosas (IoT).
- Módulos de relevadores de 5V: es un dispositivo electromecánico que funciona como un interruptor controlado eléctricamente.

- Bombillas: Son el elemento de salida visual del sistema, permitiendo observar las rutinas de encendido y apagado programadas.
- Sockets: son los soportes en los que se colocan las bombillas. Su función es proporcionar una conexión segura entre la bombilla y la fuente de energía.
- Cables macho y hembra: Los cables jumper macho y hembra son utilizados para realizar conexiones rápidas y seguras en protoboards o directamente entre módulos electrónicos.
- Shield: es una placa adicional que se coloca sobre un microcontrolador, puede ser empleado como interfaz de expansión para facilitar la conexión de los relevadores, simplificar el cableado y proteger la placa principal durante las pruebas.

### 2.3. Construcción del semáforo

Para realizar esta labor nos basamos en un diagrama propuesto en clase por lo que seguimos paso a paso como conectar los cables al relevador y a las bombillas para que nos quedara igual al siguiente diagrama:

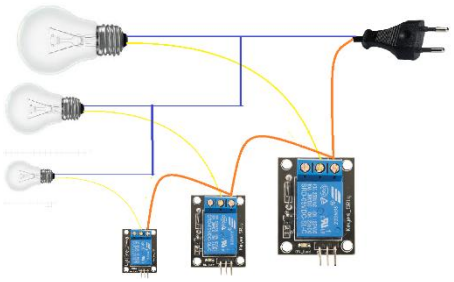

**Fig. 1.** Diagrama de conexión de con los relevadores.

Una vez teniendo la conexión de los cables con el relevador procedimos a instalar las bombillas donde elegimos los colores del semáforo, cada relevador lo conectamos a un puerto en especial por ejemplo el 21 es para el color verde, el 22 para el color amarillo y el 23 para el color rojo.

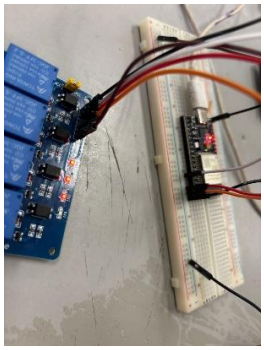

**Fig. 2.** Conexión del relevador con la placa ESP32.

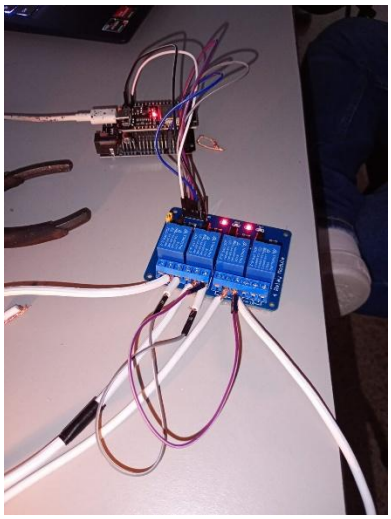

**Fig. 3.** Puentes y conexiones con el relevador.

Ya una vez teniendo todo conectado ahora procedimos a probar que la placa pudiera ejecutar de manera eficaz la tarea por lo que subimos el programa y esperamos a que los focos comenzaran a prender en el siguiente orden de verde, amarillo y rojo, al ver que si funcionaba ya solo era colocar los foquitos en la estructura.

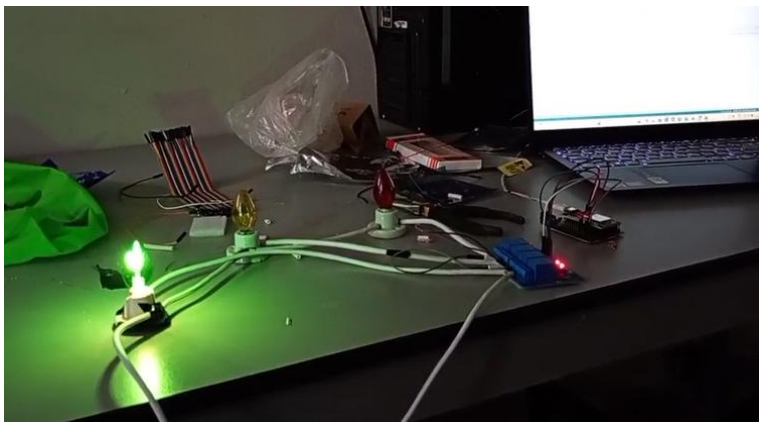

**Fig. 4.** Implementación del sistema de control con ESP32, módulo de relés y bombillas de colores que simulan un semáforo. La interfaz gráfica en la laptop envía peticiones HTTP al ESP32, el cual activa el relé correspondiente para encender o apagar cada bombilla.

Esta estructura fue nuestro inicio de partida donde ya teníamos una idea clara de cómo iba a quedar nuestro semáforo, los materiales que utilizamos fue cartón y una placa de madera, donde tenemos tres orificios que son los lugares que ocupan las bombillas por lo que el primer diseño fue este.

**Fig. 5.** Primer diseño del semáforo.

En la figura 6 se aprecia el montaje con los focos instalados en sus bases, cada uno con un color característico: verde, amarillo y rojo. Los focos están conectados al módulo de relés, que a su vez obedece las señales enviadas por el ESP32. En este momento el foco verde permanece encendido, lo que simboliza el correcto funcionamiento del sistema y la comunicación establecida entre la computadora, que envía las órdenes mediante la interfaz gráfica, y el microcontrolador que las ejecuta. El conjunto refleja de manera visual y tangible la idea central del proyecto: un semáforo controlado a distancia, donde el hardware y el software se enlazan para dar vida a una simulación realista.

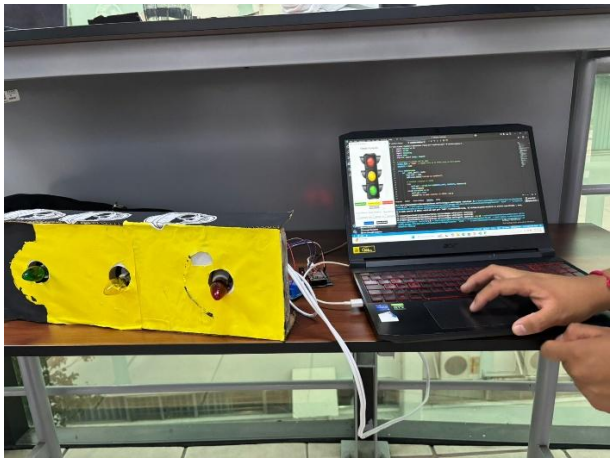

**Fig. 6.** Semáforo inteligente con interfaz gráfica.

### 2.4. Cliente y servidor

Para establecer la conexión del cliente con el servidor mostramos el siguiente diagrama que explica los pasos que realizamos para establecer la comunicación con el usuario.

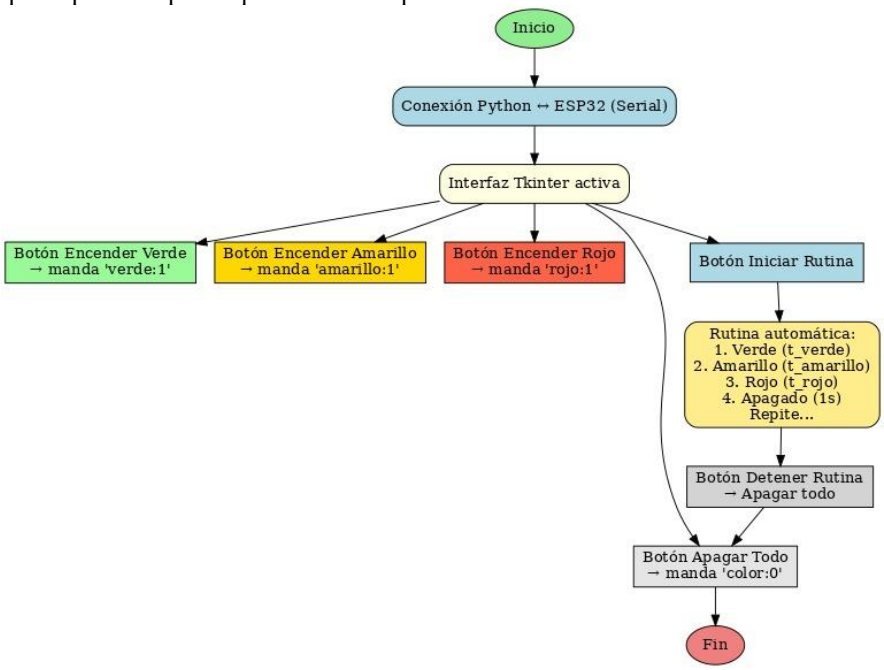


**Fig. 7.** Diagrama de flujo del semáforo

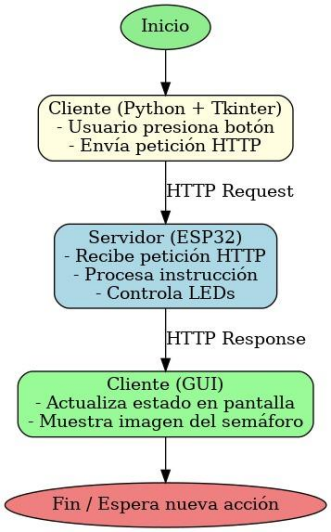


**Fig.8.** Diagrama de flujo de comunicación entre el cliente y el servidor a través de peticiones HTPP.

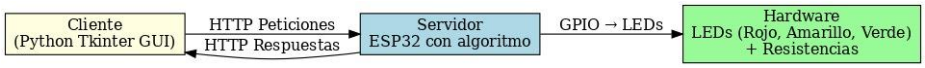


**Fig. 9.** Diagrama de flujo del proyecto.

## 2.5. Descripción de la implementación de hardware y software.

2.5.1. Cliente

Lenguaje de programación: Python

Librerías que se utilizaron:

- Tkinter: Esta librería facilita el posicionamiento y desarrollo de una interfaz gráfica de escritorio con Python. Tkinter es el paquete estándar de Python para interactuar con Tk.

- Requests: te permite enviar solicitudes HTTP/1.1 con extrema facilidad. No necesitas agregar manualmente cadenas de consulta a tus URL ni codificar tus datos en formato PUT& POST.

- PIL: es una librería gratuita que permite la edición de imágenes directamente desde Python. Soporta una variedad de formatos, incluidos los más utilizados como GIF, JPEG y PNG. Una gran parte del código está escrito en C, por cuestiones de rendimiento.

Funciones principales:

- Encender y apagar luces manualmente.

- Configurar tiempos con sliders.

- Iniciar/detener rutina automática.

- Mostrar imagen actual del semáforo en pantalla.

2.5.2. Servidor

Software (Servidor en ESP32)

Lenguaje de programación: C++ (Arduino IDE / PlatformIO)

Componentes principales:

- Servidor HTTP embebido el cual nos conectamos a través de la tarjeta de red de ESP32.

- Manejo de endpoints: Definimos estos endpoints para que a través de peticiones HTTP tengamos acceso a las rutas que contentan la activación de encendido y apagado de los leds, como, por ejemplo: /verde, /amarillo, /rojo, /apagar, /rutina.

- Control de pines GPIO para activar LEDs.

Hardware

- Placa: ESP32.

Componentes:

- LED Rojo, Amarillo, Verde.

- Resistencias limitadoras de corriente.

- Protoboard y cables de conexión.

Conexión:

- Cada LED conectado a un pin digital del ESP32.

- Pines GND compartidos.

### 2.6. Interfaz grafica

Para lograr que el cliente pudiera mandar instrucciones al semáforo, desarrollamos la siguiente interfaz gráfica donde utilizamos las librerías previamente definidas en la descripción de implementación del software.

Para esta interfaz propusimos el siguiente diseño:

1. Parte superior: Definimos una etiqueta con el estado en donde Muestra en tiempo real si el semáforo está apagado o si tiene alguna luz encendida.

2. Imagen del semáforo: Representa un semáforo de manera virtual con las tres luces ya sea verde, amarilla o roja, al presionar un botón de control, las imágenes del semáforo cambiaran para simular el funcionamiento de un semáforo real.

3. Botones de control manual: nos permiten activar de forma manual el estado de encendido o apagado del semáforo.

4. Configuración de tiempos: Son cajas que nos permiten definir la duración en que las luces estarán encendidas, los cuales nos permitan definir una rutina automática.

5. Rutina automática: consta de dos estados, el primero comienza la secuencia automática del semáforo en el siguiente orden: verde, amarillo y rojo en tiempos definidos.

6. Detener rutina: Interrumpe la rutina automática y detiene el ciclo del semáforo.

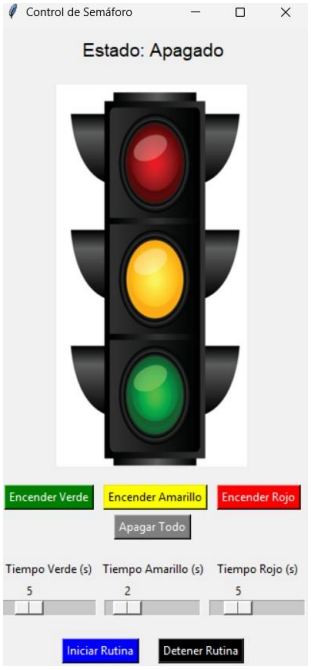


**Fig.10.** Interfaz gráfica del semáforo desarrollado en Python con botones que manejan su estado de activación, imágenes graficas del estado del semáforo y configuración de tiempos para establecer rutinas.

Con esta interfaz a través de los botones de control podemos mandar señales de ejecución a los endpoints que definimos para cada color y poder tener acceso a ellos a través de una red wifi que establece correctamente conexión con el servidor de nuestro ESP32.

# 3. Pruebas

## 3.1. Prueba de conexión con ESP32

Para comenzar a trabajar con este proyecto primero nos garantizamos de conectar nuestra placa a un dispositivo que contenga Arduino IDE que nos ayuda a cargar los programas que contienen la programación lógica para que sea capaz de ejecutar tareas donde posteriormente hicimos la prueba de conexión donde la conectamos a una red wifi para que pueda recibir instrucciones.

## 3.2. Prueba de conexión con relevador

Comenzamos a ensamblar nuestro semáforo en la cual primero hicimos conexiones con la placa ESP32, para ello cargamos un programa donde establecimos los puertos 21 para el color verde, 22 para el color amarillo y 23 para el color rojo, además de que con un protoboard y foquitos leds de colores rojo, amarillo y verde. Posteriormente, una vez identificado los pines, procedimos a hacer la prueba con el relevador de 5V.

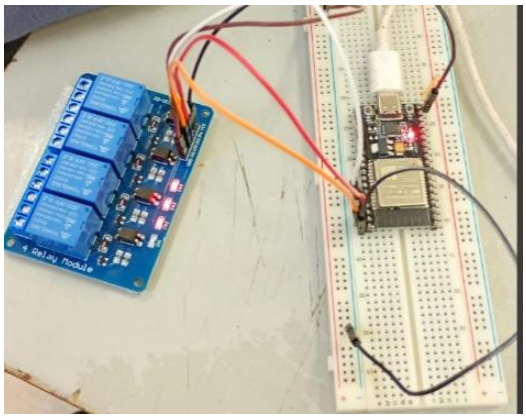

**Fig. 11.** Pruebas de conexión de la placa ESP32 con el relevador.

# 4. Resultados

## 4.1. Semáforo inteligente

El semáforo opera de manera adecuada. Al presionar el botón de control correspondiente a la luz LED verde, el sistema muestra un mensaje indicando que la luz verde ha sido encendida. De forma simultánea, la bombilla física de color verde también se activa, lo que confirma la correcta comunicación entre el programa y el hardware. Esto es posible porque el botón, al ser ejecutado manualmente, realiza una petición al servidor, lo que permite que el cambio de estado se vea reflejado tanto en la aplicación como

en el dispositivo físico. Este proceso se repite constantemente con las bombillas de color amarillo y roja por lo que muestra que la practica se ha realizado correctamente.


**Fig.12.** Interfaz gráfica del semáforo con la luz verde encendida.

**Fig.13.** Bombilla verde encendido a través de la interfaz

También presentamos los resultados de la luz de color amarilla que es la luz que va en medio antes de que el semáforo pase al color rojo.


**Fig. 14.** Interfaz gráfica del semáforo con la luz amarilla encendida.

**Fig. 15.** Bombilla amarilla encendido a través de la interfaz.

Finalmente tenemos el resultado de la luz roja que en términos de transito es cuando el semáforo esta en alto y los vehículos se detienen.

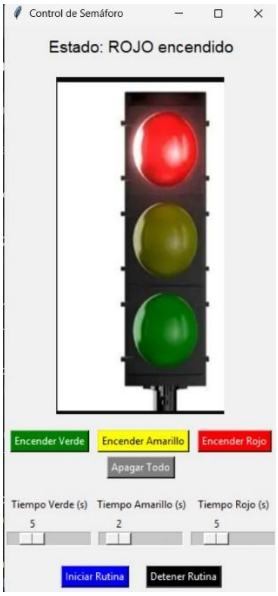


**Fig. 16.** Interfaz gráfica del semáforo con la luz roja encendida.

**Fig. 17.** Bombilla verde encendida a través de la interfaz.

Asimismo, se observó que el sistema mantiene una respuesta estable en cada uno de los casos de prueba realizados. La activación de las bombillas se ejecutó de manera inmediata tras la acción en la interfaz, lo cual evidencia una correcta sincronización entre la comunicación cliente con el servidor y la interacción con el hardware. Estos resultados permiten validar que la arquitectura implementada es funcional y que el prototipo desarrollado cumple con las condiciones necesarias para replicar el comportamiento de un semáforo real dentro de un entorno controlado.

## 5. Conclusiones

El proyecto del Semáforo Inteligente desarrollado por el equipo demuestra de manera efectiva la integración de hardware y software mediante el uso de tecnologías accesibles y modernas. Se logró construir un sistema automatizado que simula el funcionamiento de un semáforo real, controlado de forma remota a través de una interfaz gráfica desarrollada en Python y comunicada vía Wi-Fi con una placa ESP32.

Entre los aspectos más destacados se encuentran:

- La aplicación práctica de conceptos de arquitectura por capas, separando claramente la lógica del cliente y del servidor.

- El uso de componentes económicos y ampliamente para construir un prototipo funcional y escalable.

- La implementación de una comunicación inalámbrica robusta mediante HTTP, permitiendo el control remoto y la configuración flexible de los tiempos del semáforo.

- El desarrollo de una interfaz intuitiva con Tkinter que permite tanto el control manual como automático, facilitando la interacción usuario-máquina.

Este proyecto no solo cumple con el objetivo de simular un semáforo, sino que también sirve como ejemplo educativo para entender la integración entre electrónica, programación y diseño de interfaces. Además, sienta las bases para futuras mejoras, como la incorporación de sensores o el uso de algoritmos de inteligencia artificial para adaptarse dinámicamente al entorno. En resumen, el semáforo inteligente es un modelo funcional y didáctico que ilustra cómo la automatización puede aplicarse en contextos cotidianos, contribuyendo al aprendizaje multidisciplinario y fomentando la innovación tecnológica.

## Referencias

1. Maldonado, R. (2025, April 9). ¿Qué es Tkinter?: Todo sobre esta librería de Python - 2025. *KeepCoding Bootcamps*. https://keepcoding.io/blog/que-es-tkinter/

2. requests. (n.d.). PyPI. https://pypi.org/project/requests/

3. Python, R. (2022, September 19). Instalar PIL / Pillow y aplicar efectos visuales - Recursos Python. Recursos Python. https://recursospython.com/guias-y-manuales/instalar-pil-pillow-efectos/