Documento de Arquitectura  
**Baliza de Integración Continua**

**Materia:** Diseño y Arquitectura de Sistemas de Computación

**Alumnos:** Franco Sgro, David Wasyluk

**Profesores:** Jorge Fossati, Joaquín Alejo Fernandez

# 

[**Introducción: 3**](#_srxnelj5tsnv)

[Objetivo: 3](#_dde2jw9va2qd)

[Audiencia: 3](#_bilaq55i1ars)

[**Vista del Diagrama de Despliegue 4**](#_3se6y2z9gbfo)

[Descripción de la Vista: 4](#_k7fooeoxcua8)

[Elementos: 4](#_kqm76jxpz1ok)

[ESP32 (Componente de Hardware): 4](#_76t7k97zswe1)

[LEDs (Componentes de Hardware): 4](#_vptkuk6pizep)

[WiFi (Librería): 4](#_s9ai85w6fotb)

[HTTPClient (Librería): 4](#_30iin7m0ueuf)

[ArduinoJson (Librería): 4](#_wu5ozhp89ge4)

[parseJson (Función): 4](#_l747bv1k0fya)

[blinkYellowLED (Función): 4](#_v8r2jdiolx38)

[DatosDeConexion (Función) [Pendiente de implementación]: 4](#_3qxlndf4hvx5)

[Relaciones: 5](#_u9csjaiqedhc)

[Interfaces: 5](#_djk4bk7qnnmq)

[Comportamiento: 5](#_ddqgvgkno7pt)

[Variabilidad: 5](#_w9qmypu1mdcz)

[Diagrama: 6](#_q284vea2zqw0)

[Contexto Arquitectónico: 7](#_j9s6s893fiwp)

[Decisiones/Razones de Diseño: 7](#_kn7p1ahazk0y)

[Supuestos: 7](#_bgoyg0qg3koz)

[**Vista de Diagrama del Circuito Electrónico 8**](#_546y2sgsiq6x)

[Descripción de la Vista: 8](#_4dkty07qiyfg)

[Elementos: 8](#_ie7zw6pdw858)

[Relaciones: 8](#_gaa7j1xtw1ft)

[Interfaces: 8](#_xxui08xhaqpq)

[Comportamiento: 8](#_kkivkbhli0zd)

[Diagrama: 9](#_sz16yq82bunn)

[Variabilidad: 10](#_37gilqf1fp4o)

[Contexto Arquitectónico: 10](#_jym4jaya6wpa)

[Decisiones/Razones de Diseño: 10](#_3z0n7q4lpaxy)

[Análisis de los Resultados: 10](#_nvi0qwk02p70)

[Supuestos: 10](#_p8oyfd4g0yux)

[Información Adicional: 10](#_lou9civ1u47h)

[**Descripción General del Sistema: 11**](#_mnyhwl3ysa3n)

[Relación (Mapeo) entre Vistas: 11](#_eufm2hk3hg36)

[Elementos Comunes: 11](#_jsnqaxa1ohf)

[**Glosario: 11**](#_vlf8idc18hze)

[**Contexto General: 12**](#_75dufw4yv4op)

[Decisiones Generales: 12](#_qlwd4l8hti55)

[Restricciones: 12](#_p7kzjhjs0pz5)

[Razones: 12](#_d3uzzzm0fyj)

# Introducción:

El proyecto consiste en la implementación de un semáforo de integración continua basado en un microcontrolador ESP32. Este sistema interactúa con una API en GitHub para obtener información sobre las ejecuciones de un flujo de trabajo. Además, se ha incorporado un componente de gestión de datos de conexión para facilitar la configuración de la red WiFi y el repositorio de GitHub.

El proyecto se basa en la creación de un semáforo de integración continua que utiliza un ESP32 como plataforma de hardware. El ESP32 controla tres LEDs (rojo, amarillo y verde) que indican el estado del flujo de trabajo. Se comunica con una API en GitHub para obtener información sobre las ejecuciones. Además, se ha añadido un componente de gestión de datos de conexión para facilitar la configuración.

## Objetivo:

El objetivo de este proyecto es proporcionar una solución efectiva para visualizar el estado de las ejecuciones de un flujo de trabajo de integración continua. Se busca utilizar un ESP32 para controlar LEDs que reflejen el resultado de las ejecuciones y permitan identificar rápidamente cualquier problema.

## Audiencia:

Este documento de arquitectura está dirigido a:

Desarrolladores de software interesados en implementar un semáforo de integración continua utilizando un ESP32 y GitHub API.

Ingenieros electrónicos o de sistemas interesados en comprender la configuración física y la interacción de los componentes electrónicos en el circuito.

Profesionales de IT que deseen implementar un sistema de monitoreo visual para sus flujos de trabajo de integración continua.

Cualquier persona autodidacta e interesada en aprender sobre la integración de hardware y software para proyectos de automatización y visualización de datos.

Este documento proporcionará una visión completa de la arquitectura del sistema, incluyendo tanto la vista de despliegue que muestra la interacción entre los componentes, como el diagrama del circuito electrónico que representa la conexión física entre el ESP32, los LEDs y las resistencias.

# Vista del Diagrama de Despliegue

## Descripción de la Vista:

La vista presenta un sistema que implementa un semáforo de integración continua utilizando un microcontrolador ESP32. El semáforo se comunica con el servicio de GitHub a través de una conexión WiFi para obtener información sobre la última ejecución de acciones en un repositorio específico. El estado de esta ejecución se refleja mediante LEDs de colores (rojo, amarillo y verde).

## Elementos:

### ESP32 (Componente de Hardware):

Microcontrolador que ejecuta el código para controlar el semáforo y realizar conexiones WiFi.

### LEDs (Componentes de Hardware):

Indicadores visuales que representan el estado de la ejecución. Incluyen LED Rojo, Amarillo y Verde.

### WiFi (Librería):

Librería que proporciona las funciones necesarias para conectarse a una red WiFi.

### HTTPClient (Librería):

Librería que facilita las solicitudes HTTP para interactuar con servicios web.

### ArduinoJson (Librería):

Librería que permite analizar y manipular datos en formato JSON.

### parseJson (Función):

Procesa el JSON obtenido de la API de GitHub para determinar el estado de la última ejecución y controlar los LEDs en consecuencia.

### blinkYellowLED (Función):

Hace parpadear el LED amarillo cuando hay errores de conexión.

### DatosDeConexion (Función) [Pendiente de implementación]:

Función planificada para recibir y almacenar datos de conexión a WiFi y al repositorio de GitHub, permitiendo configuraciones dinámicas.

## 

## Relaciones:

El ESP32 interactúa con los LEDs para controlar su estado.

El ESP32 utiliza la librería WiFi para conectarse a la red.

La librería HTTPClient se emplea para realizar solicitudes a la API de GitHub.

La librería ArduinoJson se utiliza para analizar el JSON obtenido de la API.

Las funciones parseJson y blinkYellowLED se utilizan en el ciclo principal de ejecución.

## Interfaces:

Interfaz de Hardware (ESP32, LEDs)

Interfaz de Red (WiFi)

Interfaz de Servicio Web (API de GitHub)

## Comportamiento:

* El ESP32 se inicia y configura la conexión WiFi.
* Se intenta la conexión a la red WiFi definida.
* Si la conexión es exitosa, se realiza una solicitud a la API de GitHub para obtener información sobre la última ejecución.
* Se analiza el JSON recibido y se determina el estado de la ejecución.
* Se controlan los LEDs según el estado de la ejecución.
* Si hay errores de conexión, se activa la función blinkYellowLED para indicar el problema.

## 

## Variabilidad:

Para permitir el manejo de información proveniente de múltiples repositorios simultáneamente, se podria implementar una configuración dinámica de repositorios en el sistema. En lugar de tener la información del repositorio fija en el código, se busca habilitar la adición, eliminación o modificación de repositorios de manera dinámica a través de la interfaz web proporcionada por el Access Point. Esta funcionalidad permitiría una adaptabilidad más amplia del sistema a diferentes repositorios de GitHub, posibilitando su gestión de manera flexible y simplificada a través de la interfaz de configuración web.

## Diagrama:

## 

## Contexto Arquitectónico:

El sistema opera en un entorno que dispone de una red WiFi y acceso a la API de GitHub. La comunicación se establece a través de protocolos estándar de red y web.

**Decisiones/Razones de Diseño:**

"La elección del microcontrolador ESP32 se basó en su capacidad de conectividad WiFi y su extensa compatibilidad con librerías de desarrollo. Se añadieron funciones clave aprovechando las siguientes librerías:

* WiFi: Se utilizó para configurar el ESP32 como Access Point (AP) y para conectarlo a redes Wi-Fi externas.
* ESPAsyncWebSrv: Permitió la implementación de un servidor web asíncrono para gestionar solicitudes HTTP, posibilitando la configuración del ESP32 mediante una interfaz web.
* HTTPClient: Facilitó la realización de solicitudes HTTP para interactuar con la API de GitHub, obteniendo información sobre las ejecuciones de workflows.
* ArduinoJson: Posibilitó el análisis y procesamiento eficiente de datos en formato JSON, extraídos de las respuestas de la API de GitHub.
* La función blinkYellowLED se agregó para proporcionar una indicación visual de problemas de conexión.

**Análisis de los Resultados:**

El sistema ha demostrado eficacia al obtener información sobre la última ejecución de acciones en un repositorio de GitHub y representar su estado mediante LEDs. La inclusión de un modo Access Point (AP) permite la configuración dinámica de la conexión WiFi, facilitando la interacción con dispositivos a través de una interfaz web.

### Supuestos:

Se asume que el hardware y las conexiones físicas están configurados correctamente.

Se supone que la red WiFi y la API de GitHub están disponibles y accesibles.

# Vista de Diagrama del Circuito Electrónico

## Descripción de la Vista:

La nueva vista proporcionada es un diagrama del circuito electrónico que representa la conexión física entre los componentes de un sistema basado en un ESP32. Este sistema implementa un semáforo de integración continua utilizando tres LEDs (rojo, amarillo y verde), cada uno con una resistencia asociada.

## Elementos:

ESP32: Es el microcontrolador utilizado como plataforma de hardware para el sistema. Controla los LEDs y la comunicación con la API de GitHub.

LEDs (rojo, amarillo, verde): Representan los indicadores visuales del estado del flujo de trabajo de integración continua. Se encienden y apagan según el resultado de la ejecución.

Resistencias: Componentes que limitan el flujo de corriente a través de los LEDs para evitar daños por sobre corriente.

## Relaciones:

Los LEDs y las resistencias están conectados físicamente al ESP32.

## Interfaces:

Las interfaces son las conexiones físicas entre los pines del ESP32 y los componentes electrónicos (LEDs y resistencias).

## Comportamiento:

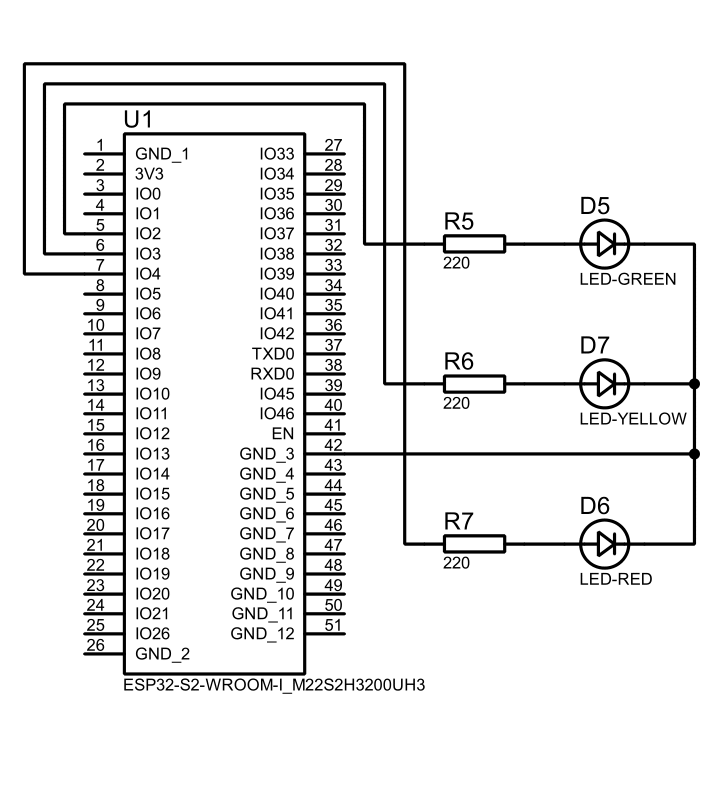
El ESP32 controla el estado de los LEDs a través de los pines a los que están conectados.

Dependiendo de la ejecución del flujo de trabajo, los LEDs se encienden o apagan para indicar el resultado.

## 

## 

## Diagrama:



Se proporciona un esquema gráfico que muestra la disposición física de los componentes en el circuito electrónico. Cada LED está conectado en serie con una resistencia a un pin específico del ESP32.

## Variabilidad:

La configuración de los pines del ESP32 puede ser modificada para adaptarse a diferentes configuraciones de hardware.

## Contexto Arquitectónico:

### Decisiones/Razones de Diseño:

Se utilizan resistencias para limitar la corriente que pasa a través de los LEDs y evitar daños por sobre corriente.

La disposición de los componentes se elige para facilitar el control y la visualización de los estados del flujo de trabajo.

### Análisis de los Resultados:

El circuito electrónico permite al ESP32 controlar el estado de los LEDs y proporcionar una indicación visual del resultado de la ejecución del flujo de trabajo.

### Supuestos:

Se asume que los LEDs y resistencias seleccionados son compatibles con el ESP32 y están correctamente conectados.

## Información Adicional:

Se recomienda verificar la hoja de datos de los componentes utilizados para asegurar su correcto funcionamiento y evitar daños.

# Descripción General del Sistema:

El sistema consiste en un semáforo de integración continua implementado en un ESP32. Este dispositivo se comunica con una API en GitHub para obtener información sobre las ejecuciones de un flujo de trabajo. El semáforo visualiza el estado de las ejecuciones a través de tres LEDs (rojo, amarillo y verde). Además, se ha incorporado un componente de gestión de datos de conexión para facilitar la configuración de la red WiFi y el repositorio de GitHub.

## Relación (Mapeo) entre Vistas:

La vista de despliegue muestra la interacción y comunicación entre los componentes del sistema, incluyendo el ESP32, los LEDs, el módulo WiFi y el HTTP Client.

El diagrama del circuito electrónico representa la conexión física entre el ESP32, los LEDs y las resistencias.

## Elementos Comunes:

ESP32: Es el núcleo del sistema en ambas vistas, encargado de controlar los LEDs y comunicarse con la API de GitHub.

LEDs: Presentes en ambas vistas, visualizan el estado del flujo de trabajo.

# Glosario:

ESP32: Microcontrolador utilizado como plataforma de hardware.

LEDs (rojo, amarillo, verde): Indicadores visuales del estado del flujo de trabajo.

WiFi Module: Componente que proporciona conectividad WiFi al ESP32.

HTTP Client: Permite realizar solicitudes HTTP a la API de GitHub.

ArduinoJSON: Biblioteca para el procesamiento de datos en formato JSON.

DatosDeConexion: Componente para la gestión de configuración de la conexión WiFi y el repositorio de GitHub.

Resistencias: Componentes que limitan la corriente a través de los LEDs.

# Contexto General:

El sistema opera en un entorno donde se dispone de conectividad WiFi y acceso a la API de GitHub. El ESP32 se alimenta a través de una fuente de corriente proporcionada por una notebook, y se comunica con la API de GitHub para obtener información sobre las ejecuciones de un flujo de trabajo.

## Decisiones Generales:

Se eligió el ESP32 como plataforma de hardware debido a su capacidad para conectarse a WiFi y su soporte para comunicaciones HTTP.

Se utilizan LEDs y resistencias para proporcionar una retroalimentación visual clara sobre el estado del flujo de trabajo.

## Restricciones:

El ESP32 debe tener acceso a una red WiFi y la API de GitHub debe estar disponible para el funcionamiento correcto del sistema.

Se debe garantizar que las credenciales de red y el token de autorización sean válidos.

## Razones:

La elección del ESP32 se basa en su versatilidad y capacidad de comunicación inalámbrica.

El uso de LEDs proporciona una forma efectiva y visualmente clara de indicar el estado del flujo de trabajo.

Esta documentación proporciona una visión completa del sistema, desde la interacción de los componentes hasta la configuración física y electrónica. Permitiendo a los desarrolladores y profesionales de TI entender y desplegar eficazmente el semáforo de integración continua.