**Propuesta del proyecto de práctica**

**Modalidad de práctica:**

|  |  |
| --- | --- |
|  | Proyecto de investigación |
|  | Trabajo en la asignatura de: Sistemas Operativos y laboratorio |

|  |  |
| --- | --- |
| Nombres y apellidos: | Omar Torres  Yuly Yesenia Alvear Romo  Leidy Castaño |
| Documento de identidad: | 91220873 |
| Teléfono: | 3043440112 |
| Semestre académico: | Noveno semestre |
| Correo electrónico: | Omara.torres@udea.edu.co |
| Fecha de inicio de actividades: | 3/o3/2024 |
| Fecha de entrega de la propuesta: | 30/05/2024 |

**TÍTULO Y SUBTÍTULO DE LA PROPUESTA:**

Ecosistema del microcontrolador ESP32 análisis y aplicación práctica.

**RESUMEN**

La presente investigación se enfoca en realizar un análisis exhaustivo de la literatura oficial existente del microcontrolador ESP32. El objetivo  principal es conocer y entender el microcontrolador ESP32, aprender sobre sus características eléctricas, sus funcionalidades, el mapeo de sus pines, la manera en que se programa, se investigara el IDE que se requiere para su programación. Si bien no es un chip que requiera de un sistema operativo tradicional como el que tiene una computadora personal, si tiene un sistema operativo llamado ROT (Es un “Sistema operativo en tiempo real”) que permite la multitarea, pues bien, nos proponemos entender cómo funciona el RTO durante la corrida de un programa. Finalmente haremos un sencillo programa que nos permita manipular sus pines en modo salida y o entrada para el manejo de luces secuenciales.

**Palabras clave:** ESP32**,** Microcontrolador**,** ROT, CHIP, IDE**.**

**INTRODUCCIÓN**

El mundo actual experimenta cambios constantes y revoluciones tecnológicas aparentemente interminables. Apenas si se comprende una revolución cuando ya están en marcha otras transformaciones en el ámbito de la ciencia y la tecnología. Estas revoluciones se entrelazan y se complementan para dar lugar a nuevos productos, servicios, tecnologías y avances continuos en la ciencia. La revolución de la electrónica se vincula con el desarrollo del software, y ambas convergen con la inteligencia artificial. En la actualidad, la creciente demanda en el campo de la Internet de las cosas (IoT) refleja la automatización de diversos aspectos de nuestro entorno.

En el ámbito de la IoT y la industria, los microcontroladores, como dispositivos programables, son fundamentales para diseñar sistemas que permitan por ejemplo el control de la temperatura en el hogar, el ahorro de energía eléctrica, la creación de equipos de propósito general para la industria de la automatización, la electromedicina y diversas aplicaciones.

En Colombia, el entendimiento y el uso práctico de los microcontroladores están limitados principalmente al ámbito educativo, durante los procesos de formación académica. Sin embargo, el desarrollo y el uso de los microcontroladores en nuestro país permanecen en gran medida inexplorados, esperando la iniciativa de los emprendedores.

El propósito de este trabajo es establecer un primer contacto con la tecnología electrónica, centrándose particularmente en el microcontrolador ESP32. Se busca elaborar un documento que puedan funcionar como herramienta para el aprendizaje en lo relativo al ecosistema electrónico del ESP32, su sistema operativo y su integración con otros dispositivos. Además, se llevará a cabo una práctica sencilla para programar el chip y controlar un dispositivo externo, en este caso particular , un control secuencial de luces. Se espera también lograr la conexión del chip a un sistema wifi.

Este trabajo adquiere relevancia desde la perspectiva de la carrera de ingeniería de sistemas, ya que proporciona a los ingenieros una visión más amplia y menos genérica del mundo de los sistemas. Implica comprender que el concepto de '1' o '0' lógicos no es simplemente una abstracción, sino que detrás de ello existe un ente real, donde un '1' lógico representa un motor en pleno funcionamiento y un '0' lógico es una máquina apagada.

El estudio del microcontrolador y su dominio abre las puertas a un mundo de oportunidades cercanas y accesibles. Este conocimiento no solo enriquece la formación académica, sino que también proporciona las habilidades necesarias para explorar las diversas posibilidades que ofrece el mundo de la tecnología y la electrónica.

**MARCO TEORICO**

El presente trabajo se enmarca en la exploración integral del microcontrolador ESP32 y sus aplicaciones prácticas en la internet de las cosas (IoT).

Con el objetivo de comprender y entender plenamente la relevancia de este trabajo es necesario abordar diversos elementos teóricos como:

1. Diferencias entre el microcontrolador y el microprocesador.
2. El microcontrolador ESP32.
3. Arquitectura del ESP32.
4. Caracteristicas eléctricas.
5. Funcionalidades.
6. FreeRto y el sistema operativo.
7. Ecosistema de desarrollo para el ESP32.
8. Protocolos de comunicaciones del ESP32.

**OBJETIVOS**

**Objetivo general**

Investigar y analizar de manera amplia el microcontrolador ESP32 y su integración en aplicaciones prácticas en la IoT, para comprender sus funcionalidades, capacidades y limitaciones.

**Objetivos específicos**

1. Analizar las caracteristicas técnicas del ESP32
2. Examinar las capacidades de conectividad del ESP32.
3. Conocer los protocolos de comunicación del ESP32.
4. Explorar la programación del ESP32.
5. Funcionalidades.
6. Realizar un experimento práctico.

**IMPLEMENTACIÓN**

Desarrollar un programa sencillo que permita controlar secuencias de luces (Usando diodos led). Se ha de programar diversas secuencias.

**METODOLOGÍA**

En el desarrollo de este trabajo se realizo una búsqueda de literatura técnica relacionada con el ESP32, Se obtuvo manuales y “Datasheet” de los fabricantes. Se estudio la información se siguieron trabajos de programación elaborados por diferentes autores y se llevo a la práctica lo aprendido, programando el ES32P.

**DIFERENCIAS ENTRE EL CONCEPTO DE MICROCONTROLADOR Y MICROPROCESADOR.**

Las diferencias entre un microcontrolador y un microprocesador radican principalmente en su diseño, funciones y aplicaciones:

**Diseño:**

**Microcontrolador:** Es un dispositivo integrado que combina un procesador central, memoria y periféricos de entrada/salida en un solo chip.

**Microprocesador**: Es un circuito integrado que incluye únicamente la unidad central de procesamiento (CPU) y depende de otros componentes externos, como memoria y periféricos, para funcionar.

**Funciones:**

**Microcontrolador:** Está diseñado para ejecutar tareas específicas en sistemas embebidos. Tiene periféricos integrados que permiten la interacción directa con el entorno externo, como sensores, actuadores, puertos de comunicación, etc.

**Microprocesador:** Se utiliza en sistemas de propósito general, como computadoras personales, donde la CPU se encarga principalmente de procesar instrucciones y datos. Depende de otros dispositivos para la entrada/salida de datos y la gestión de periféricos.

**Aplicaciones:**

**Microcontrolador:** Se encuentra en una variedad de dispositivos embebidos y sistemas automatizados, como electrodomésticos, automóviles, dispositivos médicos, sistemas de control industrial, etc.

**Microprocesador:** Se utiliza en computadoras personales, servidores, estaciones de trabajo y otros sistemas informáticos que requieren capacidades de procesamiento generales.

**Consumo de Energía:**

**Microcontrolador:** Está diseñado para operar eficientemente en términos de consumo de energía, lo que lo hace adecuado para dispositivos alimentados por batería o sistemas que requieren eficiencia energética.

**Microprocesador:** Por lo general, consume más energía que un microcontrolador debido a su diseño de propósito general y a menudo se utiliza en sistemas que tienen acceso a una fuente de energía constante, como la red eléctrica.

**EL MICROCONTROLADOR ESP32.**

El ESP32, desde la perspectiva de la electrónica, se presenta como un microcontrolador integral con un diseño eficiente y potente. Al integrar un procesador dual-core, memoria, módulos WiFi y Bluetooth, así como diversos periféricos, este microcontrolador simplifica el diseño de circuitos y proporciona un rendimiento sólido a 240 MHz. Su conectividad inalámbrica facilita la creación de dispositivos conectados y la transferencia de datos sin cables, mientras que su eficiencia energética lo hace idóneo para aplicaciones alimentadas por batería. Con su versatilidad en periféricos, como puertos SPI, I2C y UART, y su compatibilidad con Arduino, el ESP32 se convierte en una opción atractiva para proyectos de IoT, ofreciendo a los diseñadores electrónicos una herramienta versátil respaldada por una activa comunidad de desarrolladores y un amplio soporte en línea.

**ARQUITECTUTA DEL ESP32**

El ESP32-WROOM-32 es un potente y versátil módulo MCU Wi-Fi+BT+BLE diseñado para una amplia variedad de aplicaciones, que van desde redes de sensores de bajo consumo hasta tareas más exigentes, como la codificación de voz, transmisión de música y decodificación de MP3.

En el núcleo de este módulo se encuentra el chip ESP32-D0WDQ6\*. Este chip incorporado está diseñado para ser escalable y adaptable. Cuenta con dos núcleos de CPU que pueden controlarse de forma independiente, y la frecuencia del reloj de la CPU es ajustable desde 80 MHz hasta 240 MHz. El usuario también puede apagar la CPU y utilizar el coprocesador de bajo consumo para monitorear constantemente los periféricos en busca de cambios o cruces de umbrales. El ESP32 integra un conjunto completo de periféricos, que incluyen sensores táctiles capacitivos, sensores Hall, interfaz de tarjeta SD, Ethernet, SPI de alta velocidad, UART, I²S e I²C. (Manual del fabricante ESP32)

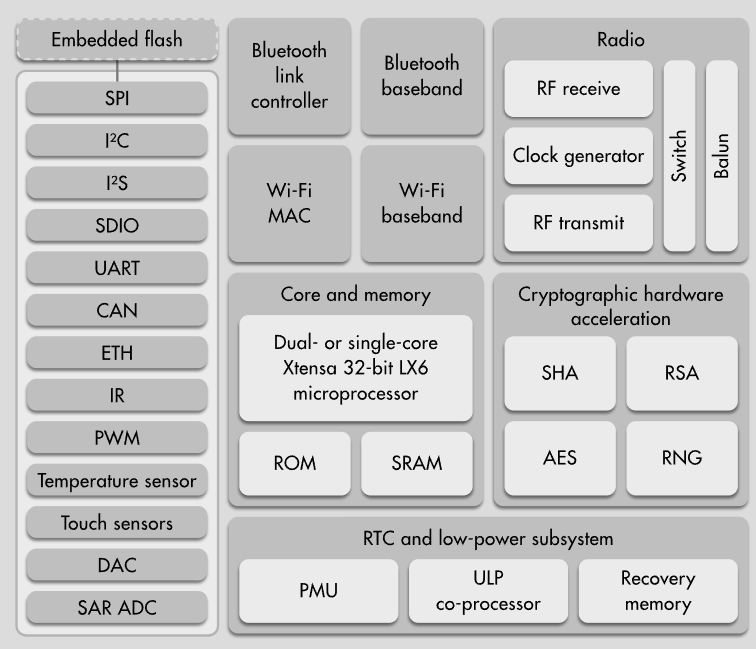


Imagen 2: arquitectura ESP32 (http://esp32.net/, ver bibliografía)

**Núcleos CPU**

El ESP32 tiene dos núcleos de procesamiento Xtensa LX6, los cuales son de arquitectura de 32 bits y funcionan a una velocidad de hasta 240 MHz.

Estos núcleos pueden ejecutar instrucciones independientemente y son útiles para realizar tareas en paralelo, como manejar la conectividad Wi-Fi y Bluetooth mientras realiza otras operaciones.

**Memoria RAM**

Cuenta con memoria RAM integrada, siendo comúnmente de 520 KB. Además, tiene una memoria flash de 4 MB para almacenar el programa y los datos.

La memoria flash es no volátil y se utiliza para almacenar el programa que se ejecuta en el ESP32.

**Memoria ROM**

El módulo ESP32P WROOM-32 se basa en el chip ESP32, y en este contexto, la "memoria ROM" generalmente se refiere a la memoria de solo lectura (Read-Only Memory, ROM) que está integrada en el propio chip ESP32. Sin embargo, vale la pena señalar que, en el caso del ESP32, la terminología utilizada es más específica.

En el ESP32, se hace una distinción entre la "ROM" y la "Flash". La ROM se refiere a una parte del firmware que se carga durante el arranque del chip y es de solo lectura, mientras que la "Flash" se refiere a la memoria flash donde se almacena el firmware del usuario y otros datos.

Aquí hay una descripción de estas dos áreas de memoria en el ESP32:

ROM:

**La ROM en el ESP32** contiene el firmware de arranque (bootloader) y el código de inicialización del sistema.

Este firmware es de solo lectura y se carga desde una memoria no volátil durante el arranque del dispositivo.

La ROM proporciona las funciones básicas necesarias para inicializar el sistema y cargar el firmware principal desde la memoria flash.

**Memoria Flash:**

El ESP32 utiliza memoria flash para almacenar el firmware del usuario, que incluye el código de la aplicación y los datos necesarios para el funcionamiento del dispositivo.

La memoria flash es no volátil y permite retener la información incluso cuando el dispositivo está apagado.

Además del firmware, la memoria flash también puede utilizarse para almacenar datos del usuario y configuraciones.

En resumen, la "memoria ROM" del ESP32P WROOM-32 se refiere principalmente a la ROM integrada en el chip ESP32, que contiene el firmware de arranque y el código de inicialización del sistema. La memoria flash se utiliza para almacenar el firmware del usuario y otros datos no volátiles.

**Wi-fi y Bluetooth**

El ESP32 integra conectividad Wi-Fi 802.11 b/g/n, lo que permite la conexión a redes inalámbricas.

También incorpora Bluetooth clásico y Bluetooth de baja energía (BLE), lo que lo hace versátil para diversas aplicaciones de IoT.

Diagrama, Esquemático

Descripción generada automáticamente

Imagen 1: arquitectura ESP32 (Web Electronics, ver bibliografía)

**Memoria**

Ofrece una amplia variedad de periféricos, incluyendo GPIO (Entrada/Salida de Propósito General), UART (Universal Asíncronos Receiver/Transmitter), SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), PWM (Modulación de Ancho de Pulso), entre otros.

Incluye un convertidor analógico a digital (ADC) que permite medir señales analógicas.

**Seguridad**

Ofrece una amplia variedad de periféricos, incluyendo GPIO (Entrada/Salida de Propósito General), UART (Universal Asynchronous Receiver/Transmitter), SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), PWM (Modulación de Ancho de Pulso), entre otros.

Incluye un convertidor analógico a digital (ADC) que permite medir señales analógicas.

**Sistema operativo**

Puede ejecutar un sistema operativo en tiempo real (RTOS), como FreeRTOS, que facilita la multitarea y la gestión eficiente de recursos.

**Consumo de Energía**

El ESP32 está diseñado para ser eficiente en términos de consumo de energía, lo que lo hace adecuado para dispositivos alimentados por batería.

**Desarrollo**

Puedes programar el ESP32 utilizando lenguajes como C o C++ utilizando el entorno de desarrollo integrado (IDE) de Arduino o el Espressif IDF (IoT Development Framework).

**Especificaciones técnicas del ESP32**

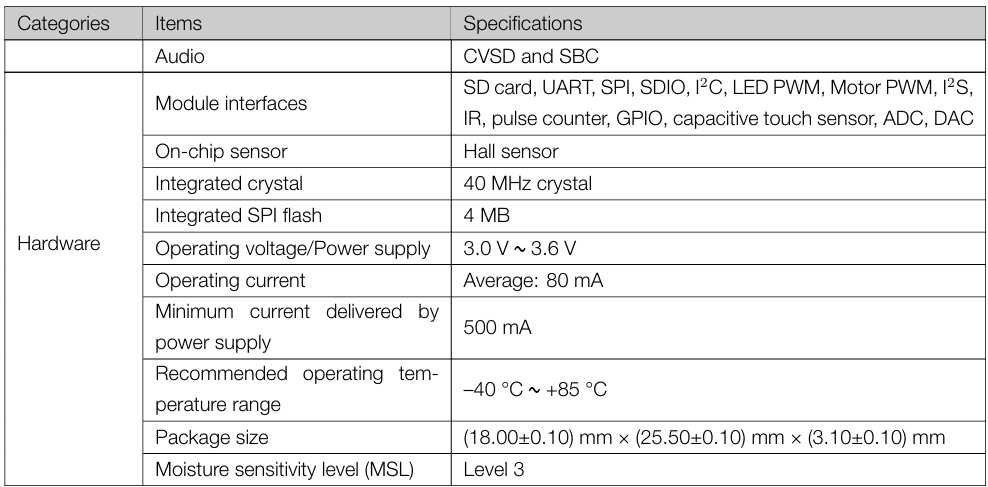
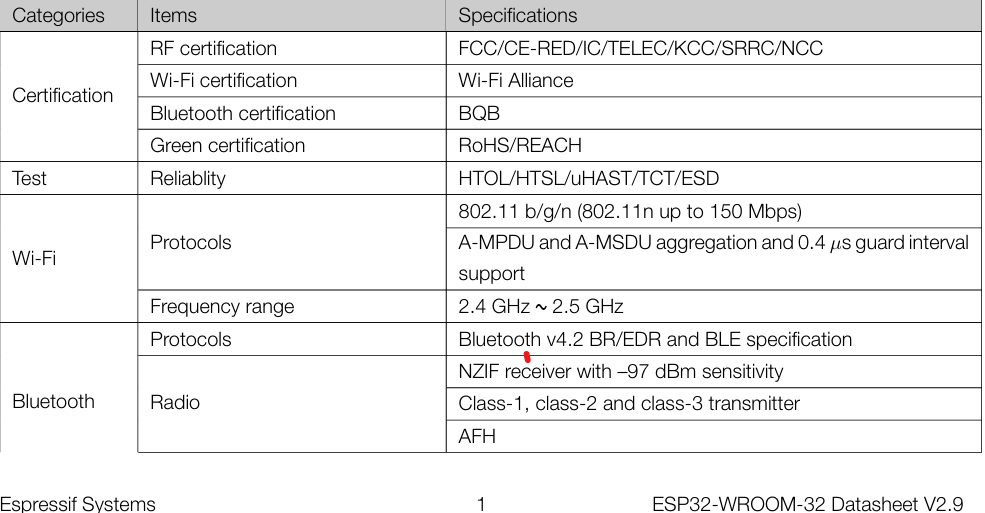


Imagen 3: (Tabla 1: Manual del fabricante)

**Mapeo de los pines ESP32**

El ESP32 utiliza técnicas de mapeo y multiplexado de pines para lograr la multifuncionalidad de sus pines GPIO (General Purpose Input/Output). La multifuncionalidad implica que un mismo pin puede asumir diferentes roles o funciones según la configuración y necesidades específicas del programa.

A través del sistema de configuración de pines, se asignan funciones específicas a los pines GPIO, como entrada analógica, salida digital, interfaz I²C, interfaz SPI, UART, entre otras. Este proceso se realiza mediante el ajuste de registros de multiplexación (MUX) asociados a cada pin.

En resumen, gracias a estas técnicas de mapeo y multiplexado, un mismo pin en el ESP32 puede adaptarse dinámicamente para realizar distintas tareas, proporcionando flexibilidad y eficiencia en la gestión de recursos en entornos con limitaciones de pines.

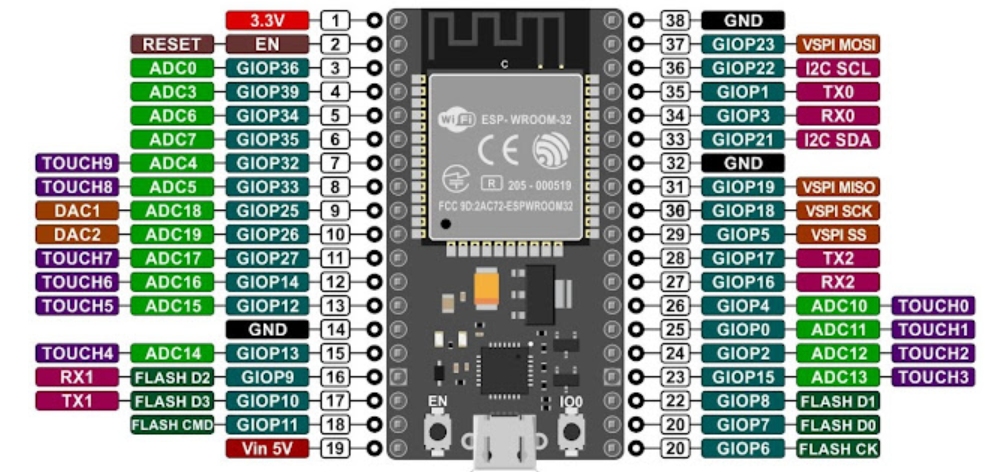


Figura 4: tomada de (puntoFlotante.net)

**Pines de entrada**

El ESP32 Wroom32 tiene un total de 25 GPIOs de los cuales solo unos pocos pines pueden ser configurado dado que no cuentan con la resistencia pull-down entre los que destacamos los del grupo del 34 1l 39.

1. GPIO 34
2. GPIO 35
3. GPIO 36
4. GPIO 37
5. GPIO 38
6. GPIO 39

**Pines de salida**

El ES32 tiene pines que permiten ser programados para la conversión análogo digital, digital análogo y procesos de comunicaciones UART, I2C, SPI, y PWM. Operaciones que se configuran mediante código, destacamos los pines del grupo del 6 al 11:

1. GPIO 6 (SCK/CLK)
2. GPIO 7 (SDO/SD0)
3. GPIO 8 (SDI/SD1)
4. GPIO 9 (SHD/SD2)
5. GPIO 10 (SWP/SD3)
6. GPIO 11 (CSC/CMD)

**Touch Sensor**

El ESP32 dispone de 10 sensores que se pueden utilizar de forma táctil como sensor capacitivo. Estos pines tienen la capacidad de detectar variaciones en la carga eléctrica, como por ejemplo la piel humana. Para que puedan detectar variaciones inducidas al tocar los GPIO con un dedo. Estas entradas se pueden integrar fácilmente en las almohadillas capacitivas y reemplazan los botones mecánicos. Las clavijas táctiles capacitivas también se pueden usar para activar el ESP32 cuando entra en modo Deep-Sleep.

Esos sensores táctiles internos, Destacamos los pines del grupo listado a continuación (**Ver: Tecnotízate**).

1. T0 (GPIO 4)
2. T1 (GPIO 0)
3. T2 (GPIO 2)
4. T3 (GPIO 15)
5. T4 (GPIO 13)
6. T5 (GPIO 12)
7. T6 (GPIO 14)
8. T7 (GPIO 27)
9. T8 (GPIO 33)
10. T9 (GPIO 32)

**Conversores análogos a digital.**

El ESP32 tiene un número importante de pines que pueden ser programados en los procesos de discretización o digitalización de una señal eléctrica, listamos el siguiente grupo de pines:

1. ADC1\_CH0 (GPIO 36)
2. ADC1\_CH1 (GPIO 37)
3. ADC1\_CH2 (GPIO 38)
4. ADC1\_CH3 (GPIO 39)
5. ADC1\_CH4 (GPIO 32)
6. ADC1\_CH5 (GPIO 33)
7. ADC1\_CH6 (GPIO 34)
8. ADC1\_CH7 (GPIO 35)
9. ADC2\_CH0 (GPIO 4)
10. ADC2\_CH1 (GPIO 0)
11. ADC2\_CH2 (GPIO 2)
12. ADC2\_CH3 (GPIO 15)
13. ADC2\_CH4 (GPIO 13)
14. ADC2\_CH5 (GPIO 12)
15. ADC2\_CH6 (GPIO 14)
16. ADC2\_CH7 (GPIO 27)
17. ADC2\_CH8 (GPIO 25)
18. ADC2\_CH9 (GPIO 26)

**Conversor digital a análogo**

El ESP32 dispone de 2 canales DAC de 8 bits, útiles para convertir señales digitales en salidas de señal de voltaje analógicas. Se listan los siguientes pines que pueden ser programados para este fin:

1. DAC1 (GPIO25)
2. DAC2 (GPIO26)

**RTCS**

ESP32 ofrece soporte de RTC GPIO que se encuentran enrutados al subsistema de baja potencia RTC y se pueden usar cuando el ESP32 está en modo Deep-Sleep. Estos RTC GPIO se pueden usar también para reactivar el ESP32 cuando está en uso el coprocesador Ultra Low Power (ULP). Los siguientes GPIO se pueden usar como una fuente de activación externa RTC (**Ver: Tecnotízate**):

1. RTC\_GPIO0 (GPIO36)
2. RTC\_GPIO3 (GPIO39)
3. RTC\_GPIO4 (GPIO34)
4. RTC\_GPIO5 (GPIO35)
5. RTC\_GPIO6 (GPIO25)
6. RTC\_GPIO7 (GPIO26)
7. RTC\_GPIO8 (GPIO33)
8. RTC\_GPIO9 (GPIO32)
9. RTC\_GPIO10 (GPIO4)
10. RTC\_GPIO11 (GPIO0)
11. RTC\_GPIO12 (GPIO2)
12. RTC\_GPIO13 (GPIO15)
13. RTC\_GPIO14 (GPIO13)
14. RTC\_GPIO15 (GPIO12)
15. RTC\_GPIO16 (GPIO14)
16. RTC\_GPIO17 (GPIO27)

**PWM**

El controlador ESP32 LED PWM cuenta con 16 canales independientes que pueden configurarse para generar señales PWM con diferentes propiedades. Todos los pines que pueden actuar como salidas se pueden usar como pines PWM (los GPIOs 34 a 39 no pueden generar PWM ya que como se indicó líneas atrás no tienen la resistencia **Pull-Up/Down.**

Para producir una señal PWM, es necesario definir a través del código los siguientes parametros:

1. frecuencia de la señal;
2. Ciclo de trabajo;
3. canal PWM;
4. GPIO que emite la señal.

**ISP**

La funcionalidad ISP (In-System Programming) permite la programación o carga de firmware en un dispositivo, como un microcontrolador, sin la necesidad de retirarlo físicamente de la placa o sistema en el que está integrado. Esto resulta particularmente valioso cuando retirar el dispositivo para programarlo resulta impráctico. Aquí hay una descripción organizada de la funcionalidad ISP:

**Programación sin retirar el dispositivo**: La ISP posibilita la programación del dispositivo sin desconectarlo de la placa o sistema en el que está integrado, simplificando el desarrollo y las actualizaciones de firmware. En el caso del ESP32, los pines específicos utilizados para la transferencia de datos durante la programación son GPIO0, GPIO2, GPIO12 y GPIO15.

**Pines y protocolos específicos**: La programación in situ implica el uso de pines específicos del dispositivo para la transferencia de datos de programación y, en algunos casos, la utilización de un protocolo de comunicación específico. Por ejemplo, en el ESP32, se utilizan pines como GPIO0, GPIO2, GPIO12 y GPIO15 para la programación mediante ISP.

Hardware externo o interfaces de software: La ISP puede llevarse a cabo utilizando hardware externo, como programadores de ISP, o mediante interfaces de software, como herramientas de programación que se comunican con el dispositivo a través de un puerto de comunicación, como UART o USB.

En resumen, la ISP es una forma de programación y actualización del firmware mientras el dispositivo está integrado en el sistema final, es decir, permite la programación en caliente.

**I2C**

La comunicación I2C (Inter-Integrated Circuit) es un protocolo de comunicación serial que permite la transferencia de datos entre dispositivos en un sistema electrónico. Aquí tienes una descripción clara y concisa:

**I2C (Inter-Integrated Circuit):**

La comunicación I2C es un protocolo serial utilizado para la transferencia de datos entre dispositivos en un sistema electrónico. Está diseñado para facilitar la conexión de múltiples dispositivos a través de dos líneas de comunicación: una línea de datos (SDA) y una línea de reloj (SCL).

**Líneas de comunicación:**

SDA (Serial Data): Línea para la transmisión de datos.

SCL (Serial Clock): Línea que proporciona el pulso de reloj para sincronizar la transferencia de datos.

Funcionamiento:

La comunicación I2C se basa en la idea de un maestro y varios esclavos.

El maestro inicia y controla la comunicación.

Cada dispositivo conectado tiene una dirección única.

Los dispositivos son seleccionados mediante su dirección por el maestro.

La transferencia de datos ocurre en paquetes de 8 bits (bytes).

Ventajas:

1. Permite la conexión de múltiples dispositivos en el mismo bus.
2. Uso eficiente de pines de comunicación.
3. Permite la comunicación a velocidades variables.
4. Aplicaciones:
5. Comúnmente utilizado para conectar sensores, pantallas, EEPROMs y otros dispositivos en sistemas electrónicos.

La comunicación I2C es ampliamente adoptada en sistemas embebidos y es conocida por su simplicidad y eficiencia en la interconexión de componentes electrónicos.

**Módulo WiFi**

El ESP32, incorpora un módulo WiFi que proporciona capacidades de conectividad inalámbrica.

**Aquí algunos aspectos destacados de este módulo:**

**Conectividad WiFi**: El módulo WiFi del ESP32 permite la conexión a redes locales (LAN) y la conexión a Internet a través de conexiones WiFi. Esto es fundamental para la comunicación inalámbrica en una variedad de aplicaciones.

**Modos de operación**: Puede funcionar en modos como estación (conectándose a un punto de acceso WiFi), punto de acceso (creando su propia red WiFi) o en una combinación de ambos, brindando flexibilidad en el establecimiento de conexiones inalámbricas.

Seguridad: Incorpora mecanismos de seguridad WiFi, como WPA/WPA2, asegurando la protección de las comunicaciones inalámbricas contra amenazas potenciales.

API de software: Ofrece una interfaz de programación de aplicaciones (API) de software que permite a los desarrolladores controlar y configurar el módulo WiFi a través del software del dispositivo.

**Funcionalidades avanzadas:** Además de la conectividad básica, el módulo WiFi del ESP32 posibilita operaciones avanzadas como actualizaciones de firmware a través de la red (OTA), facilitando la actualización remota del software.

**Aplicaciones en IoT**: Gracias a su combinación de capacidades WiFi y potencia de procesamiento, el ESP32 se destaca en proyectos de Internet de las cosas (IoT), donde se requiere conectividad inalámbrica para la transferencia de datos.

En resumen, el módulo WiFi integrado en el ESP32 potencia la conectividad inalámbrica en este microcontrolador, siendo una característica clave para una amplia gama de aplicaciones y proyectos. (**Manual del fabricante ESP32)**

**FREERTO Y EL SISTEMA OPERATIVO**

El sistema operativo del ESP32 no es un sistema operativo como los que se usan en las computadoras personales, EL RTO es similar a un sistema operativo como el Windows o Linux, pero tienen grandes diferencias, como por ejemplo la multitarea en un computador se relaciona con la ejecución compartida de varios procesos en una misma CPU o núcleos de una CPU. Por otro lado, en el sistema RTO la multitarea hace referencia a que varias funciones de un mismo programa se puedan ejecutar compartiendo la CPU o los núcleos de la CPU.

**Sistema Operativo de Tiempo Real (RTOS) en el ESP32 - FreeRTOS:**

El ESP32, al utilizar FreeRTOS como su sistema operativo de tiempo real, destaca por ofrecer una gestión eficiente y predecible de tareas en entornos embebidos. A continuación, se detallan aspectos clave, desde la planificación de tiempos de ejecución hasta las herramientas específicas como semáforos y colas.

Planificación de Tiempos de Ejecución:

FreeRTOS implementa la técnica de planificación "Round-Robin", asegurando una asignación justa y predecible de tiempo de ejecución para cada tarea. Esto es esencial para cumplir con los requisitos temporales de las aplicaciones en tiempo real.

**Asignación de CPU a Tareas:**

La asignación de CPU se realiza mediante un programador de tareas, asignando la CPU a la tarea de mayor prioridad disponible. En situaciones de igual prioridad, se utiliza Round-Robin para una distribución equitativa.

**Multitarea**:

La multitarea en FreeRTOS permite la ejecución simultánea de múltiples tareas independientes. Destacan:

**Tareas Independientes**: Operan de forma aislada, realizando operaciones específicas sin interferencias.

**Colas y Semáforos**: Cruciales para la comunicación y sincronización entre tareas.

**Semáforos**: Gestionan la disponibilidad de recursos, utilizando operaciones como "tomar" y "dar".

**Colas**: Facilitan la transferencia segura de datos entre tareas.

Temporizadores: Facilitan operaciones programadas, permitiendo la ejecución de tareas en momentos específicos.

**Semáforos:**

**Los semáforos** en FreeRTOS generalmente tienen dos estados principales:

**Disponible (Take):** Cuando el semáforo tiene un valor mayor que cero, significa que está disponible. Si una tarea intenta tomar el semáforo en este estado, se reduce su valor y la tarea continúa ejecutándose.

**No Disponible (Blocked):** Si el semáforo tiene un valor de cero, y una tarea intenta tomarlo, la tarea se bloquea hasta que el semáforo esté disponible nuevamente.

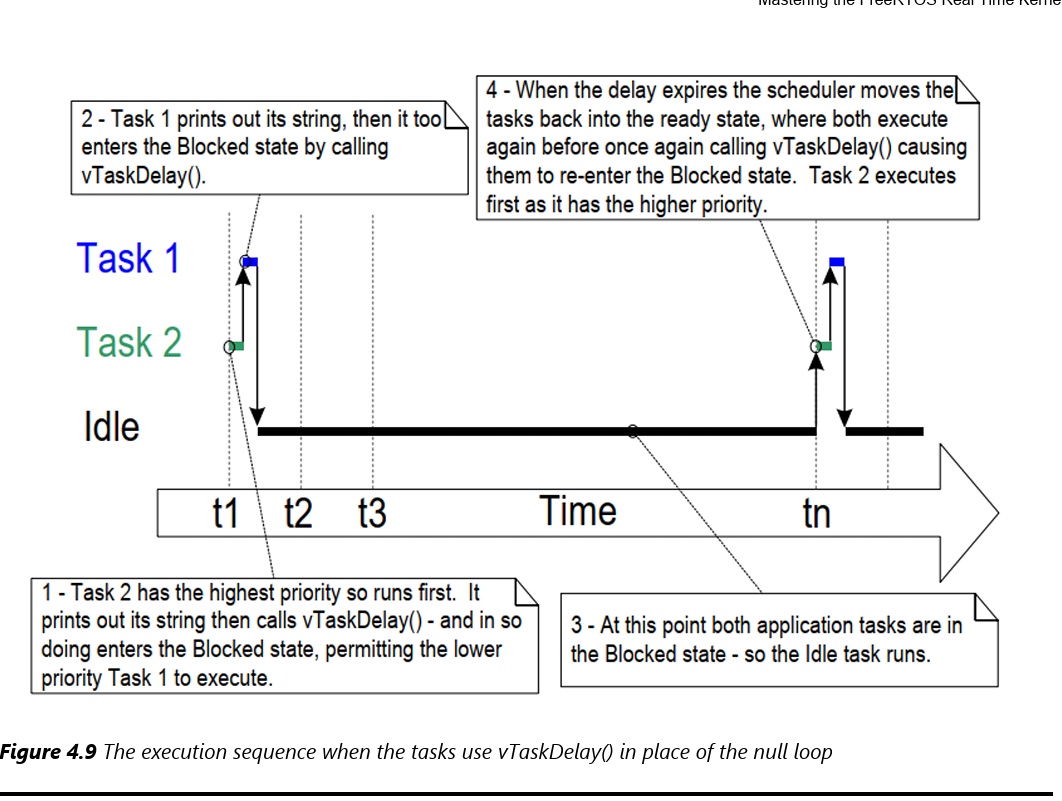
**Interrupciones y prioridades:**

En FreeRTOS, las interrupciones (o servicios de interrupción) pueden tener diferentes niveles de prioridad. La cantidad específica de niveles de prioridad puede variar según la configuración del sistema. En general, los sistemas que implementan FreeRTOS pueden tener hasta 32 niveles de prioridad, permitiendo una asignación flexible de prioridades para gestionar las interrupciones.

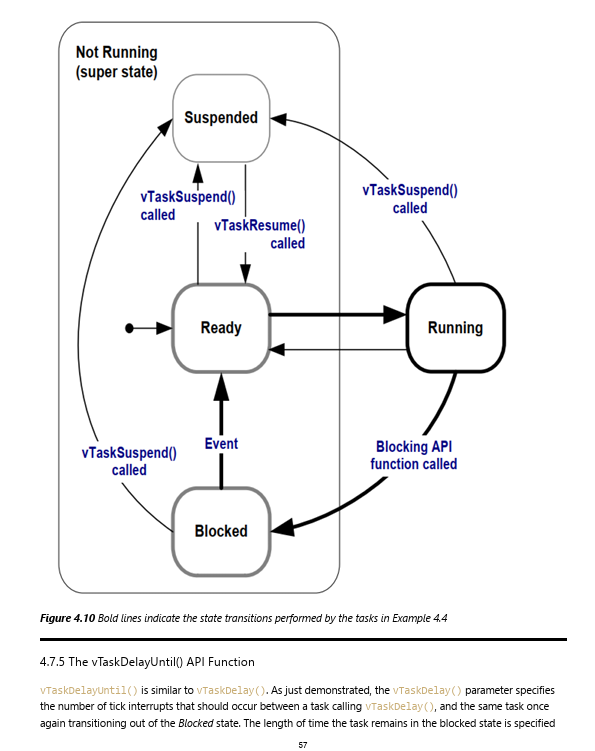
La integración de estas características en FreeRTOS proporciona un entorno robusto y eficiente para aplicaciones embebidas en el ESP32. Estas herramientas son fundamentales para garantizar una ejecución segura, predecible y eficiente en sistemas multitarea en tiempo real (**Sistema Operativo RTO**).

**Programación RTO**

Dos tareas con diferente prioridad comparten los recursos hardware.



**Grafico tomado del documento oficial del RTO (Sistema Operativo RTO)**



**Grafica tomada documento oficial RTO (Sistema Operativo RTO)**

**PROGRAMACION del ESP32**

Para programar el ESP32, existen varias herramientas de desarrollo que facilitan la tarea de escribir, compilar y cargar código en el microcontrolador. Algunas de las herramientas más comunes para programar el ESP32 son:

**Arduino IDE:**

El Arduino IDE es una herramienta de desarrollo ampliamente utilizada para programar microcontroladores, incluido el ESP32. Puedes programar el ESP32 utilizando la plataforma oficial de ESP32 para Arduino, que proporciona una interfaz fácil de usar y una amplia comunidad de usuarios.

**PlatformIO:**

PlatformIO es una plataforma de desarrollo que se integra con diferentes IDEs, como Visual Studio Code y Atom. Ofrece soporte para una variedad de placas, incluido el ESP32, y proporciona una interfaz más avanzada con funciones de desarrollo más potentes.

**Espressif IDF (IoT Development Framework):**

Espressif IDF es el marco de desarrollo oficial de Espressif, el fabricante del ESP32. Es más avanzado que las soluciones de Arduino y proporciona acceso directo a las funciones y características específicas del ESP32. Es adecuado para desarrolladores que buscan un mayor control sobre el código que están desarrollando y el hardware.

**MicroPython:**

MicroPython es una implementación de Python diseñada para microcontroladores, incluido el ESP32. Permite programar el ESP32 en Python, lo que puede ser beneficioso para aquellos familiarizados con este lenguaje.

**NodeMCU/Lua:**

Para aquellos que prefieren el uso de Lua, NodeMCU es una plataforma que permite programar el ESP32 utilizando este lenguaje de scripting.

Cada herramienta tiene sus propias ventajas y desventajas, y la elección depende en gran medida de tus preferencias y requisitos de desarrollo. El Arduino IDE es popular para principiantes debido a su simplicidad, mientras que Espressif IDF proporciona un mayor control y acceso a funciones avanzadas para desarrolladores más experimentados. La elección también puede depender de la comunidad de usuarios, la documentación disponible y las características específicas que necesitas para tu proyecto.

En este trabajo utilizaremos como IDE el Arduino, herramienta que conocemos y nos evita invertir tiempo en entender otras herramientas de desarrollo.

**Adquisición Inicial del ESP32**

Cuando adquieres un módulo ESP32 por primera vez, este generalmente llega en su estado predeterminado de fábrica, listo para ser programado según tus necesidades específicas. A continuación, se describe cómo suele llegar y la necesidad de ponerlo en modo de programación.

**Estado de Fábrica:**

Al recibir un ESP32 nuevo, este generalmente viene con un firmware de fábrica básico instalado. Este firmware puede variar según el fabricante, pero suele ser lo suficientemente simple como para permitir la programación del dispositivo.

Listo para Programar:

El ESP32 no contiene código de usuario específico al principio, lo que significa que está listo para ser programado con el firmware que desees cargar en él. Esto proporciona flexibilidad para adaptar el ESP32 a tus necesidades específicas.

Modo de Programación:

Para cargar nuevo firmware en el ESP32, generalmente es necesario poner el dispositivo en modo de programación. Este modo permite la comunicación con la herramienta de programación y la carga del nuevo código en el ESP32.

Conexión Específica de Pines:

Para activar el modo de programación, debes conectar o desconectar ciertos pines del ESP32 según la variante específica del módulo. Estos pines suelen incluir GPIO0 y GPIO2, entre otros. La combinación exacta de pines puede variar según el fabricante y el modelo.

Interfaz de Programación:

Al poner el ESP32 en modo de programación, este entra en un estado que permite la comunicación con la herramienta de programación a través de interfaces como UART o USB. Este paso es esencial para cargar el firmware de usuario en el ESP32.

La necesidad de poner el ESP32 en modo de programación radica en la capacidad de permitir la carga de nuevo firmware de manera segura y controlada. Al conectar o desconectar ciertos pines, el ESP32 entra en un estado específico que habilita la interfaz de programación y facilita el proceso de carga del código del usuario. Este modo es una característica fundamental que asegura la flexibilidad y la capacidad de personalización del ESP32 para diferentes aplicaciones.

**Modo programación del ESP32 W-ROOM-32**

El procedimiento para poner el microcontrolador ESP32 en estado de programación, consiste en:

1. Abrir el IDE de Arduino.
2. Ir a la pestaña tools de la barra de menú y seleccionar la referencia de la tarjeta ESP32 que en nuestro caso fue **ESP32 Dev Module.**
3. Estando en tools seleccionar el puerto usb en el caso nuestro se seleccionó el puerto COMS 6
4. Compilar el programa a ejecutar.
5. Estando la tarjeta conectada al puerto COMS 6 de la computadora, y energizada, iniciar la carga del programa, El IDEE inicia la compilación al tiempo que genera mensajes en la consola de Arduino, justo cuando sale el mensaje de que se está estableciendo la comunicación, pulsar el botón boot que viene en la tarjeta del ESP32, se mantiene pulsado un instante y se suelta, el programa queda de esta forma cargado con el programa del usuario.

**Internet De las cosas IoT.**

**Problema de aplicación.**

Diseñe un programa que active el conjunto de 8 pines GPIO (12, 13, 14, 25, 26, 27, 33, 32) como salidas, para controlar los artefactos, luces, ventilador, televisión, aire acondicionado, apertura de garaje, arranque bomba de agua, circuito reserva1, circuito reserva2. Para efectos prácticos simule cada carga con un diodo emisor de luz. Cada circuito debe poder ser activado y o apagado individualmente por el cliente que se conecte a la red, debe poderse ver en la pagina un mensaje que indique que circuito esta activado.

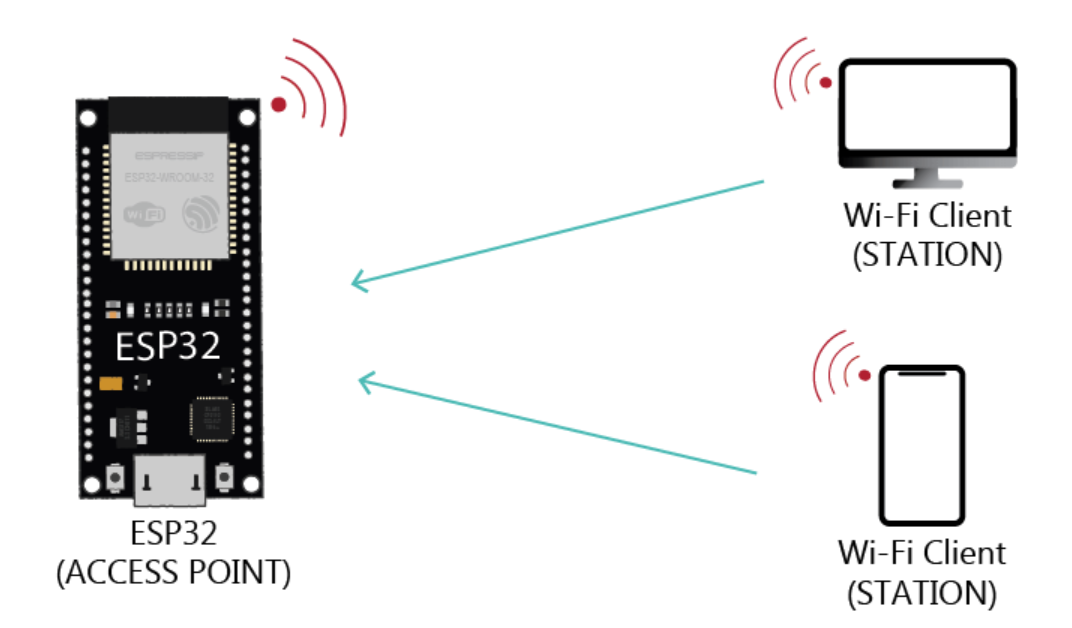
La tarjeta debe programarse como un servidor conectado a una red local wifi , si un usuario hace una requisición de encendido de las luces el programa debe realizar la ejecución de acuerdo con la requisición del cliente.

**Interloops de seguridad.**

El horno solo puede ser arrancado si el ventilador del fondo del horno FA32 esta arrancado y si una de las bombas de agua PP321 o PP333 están arrancadas. Si algunos de estos equipos se apagan el horno de fundición FC01 debe apagarse.

**ESP32 como servidor**

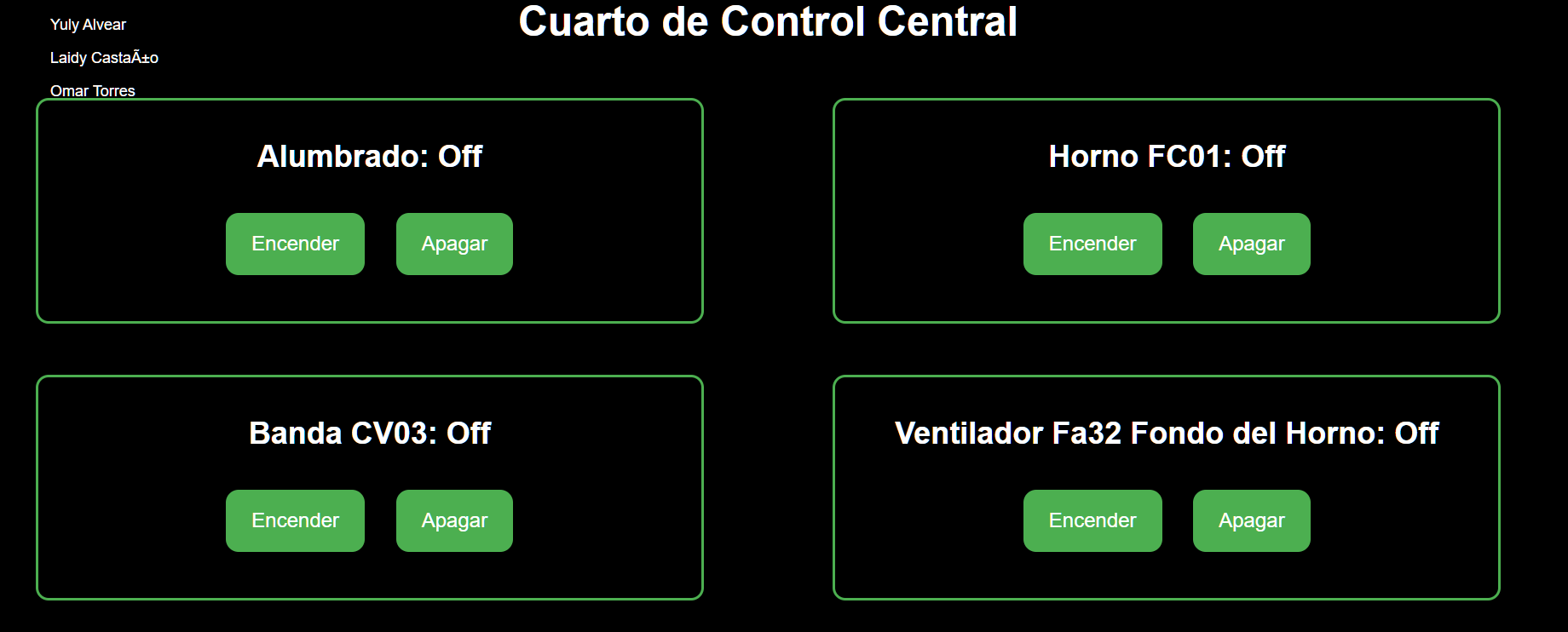
El ESP3232vpermite la conectividad mediante el uso del protocolo de comunicaciones TCP/IP, de tal forma que se puede usar como un servidor web.



**Imagen tomada de: ( Web electrosoftcloud)**

En el ejercicio que hemos planteado como ejemplo, el ESP32 se programó para que se conecte a una red wifi privada. Al energizar el microcontrolador el busca la red wifi cercana y si encuentra una a la que se pueda conectar según las credenciales (nombre de la red, password) se conecta y permite que cualquier cliente le acceda usando la IP que el dispositivo como valida. Se resalta que programar la ESP32 como servidor web es relativamente sencillo gracias a la librería **“<wifi.h>”**

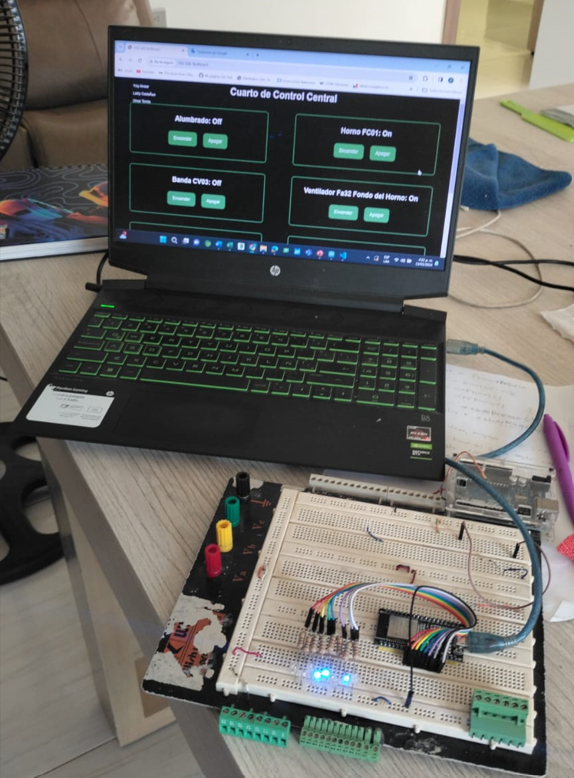
**Vista de la página web**

****

**Programa que resuelve el problema planteado**

El código del programa se entrega como un documento anexo y un archivo físico del programa en lenguaje c para el ESP32.

**Esquema de conexiones**



**CRONOGRAMA**

**CONCLUSIONES**

Durante el desarrollo de este trabajo, se lograron exitosamente los objetivos propuestos. Se exploró extensamente la literatura relacionada con el microcontrolador ESP32, proporcionada por los fabricantes, lo que nos permitió adentrarnos en el amplio ecosistema de hardware y software de este dispositivo.

En el proceso, adquirimos un conocimiento detallado sobre los protocolos de comunicación, analizamos las características eléctricas del chip y comprendimos a fondo el mapeo de los pines. Este mapeo nos permitió asignar funciones diversas a un mismo pin, ya sea como entrada, salida, salida PWM o para funciones de comunicación.

Asimismo, nos sumergimos en los elementos fundamentales de la programación del chip, estableciendo una conexión valiosa entre este conocimiento y nuestra formación como ingenieros de sistemas.

En un logro destacado, implementamos con éxito un sistema que simula la operación de un cuarto de control en una empresa industrial. La aplicación diseñada permite que un operador se conecte para monitorear y controlar un proceso. Este enfoque de simulación encuentra su relevancia en el ámbito de la Internet de las Cosas (IoT), donde la interconexión y la gestión remota son fundamentales.

El logro exitoso de este trabajo fue posible gracias al apoyo y compromiso del equipo, subrayando la importancia de la colaboración en proyectos de esta índole.

**PERSPECTIVAS PARA FUTUROS DESARROLLOS:**

En el horizonte de avances potenciales en este proyecto, se sugieren varias direcciones que podrían enriquecer y optimizar la simulación del cuarto de control industrial basada en el microcontrolador ESP32. Considerando el contexto académico y de investigación, las siguientes áreas de exploración podrían ser de interés:

**Investigación en Sensores y Monitoreo**:

Explorar la integración de sensores adicionales para ampliar la capacidad de monitoreo del entorno industrial simulado. Esto podría incluir investigaciones sobre la selección adecuada de sensores y sus protocolos de comunicación.

**Desarrollo de Protocolos de Comunicación Específicos**:

Investigar y desarrollar protocolos de comunicación específicos para la gestión eficiente de una gama más amplia de dispositivos industriales. Esto podría involucrar la adaptación de estándares existentes o la creación de nuevos enfoques de comunicación.

**Optimización de Interfaces Gráficas**:

Explorar mejoras en las interfaces gráficas de usuario para aumentar la eficiencia y usabilidad. Esto podría incluir investigaciones en visualización de datos, representación gráfica de procesos y diseño centrado en el usuario.

**Integración de Tecnologías Emergentes:**

Investigar la viabilidad de incorporar tecnologías emergentes, como Machine Learning o Inteligencia Artificial, para mejorar la capacidad predictiva y adaptativa del sistema. Evaluar cómo estas tecnologías podrían contribuir a la eficiencia operativa y la toma de decisiones.

**Desarrollo de Medidas de Seguridad Avanzadas:**

Investigar y desarrollar medidas de seguridad avanzadas, tales como autenticación multifactor, cifrado robusto y sistemas de monitoreo continuo. Estos aspectos son cruciales en entornos industriales donde la seguridad es una prioridad fundamental.

**Investigación en Estrategias de Optimización de Energía:**

Explorar estrategias de optimización de energía para gestionar eficientemente el consumo de energía de los dispositivos controlados. Investigar posibles fuentes de energía alternativas y su implementación en el sistema.

**Integración con Plataformas en la nube**:

Investigar la viabilidad y los beneficios de la integración con plataformas de nube. Esto permitiría almacenar datos de manera centralizada, facilitando el análisis de tendencias a largo plazo y el acceso remoto a información crítica.

**Desarrollo de Aplicaciones Móviles Complementarias:**

Investigar y desarrollar aplicaciones móviles complementarias que extiendan la capacidad de monitoreo y control a dispositivos móviles. Evaluar la implementación de funcionalidades adicionales para mejorar la flexibilidad operativa.

Estas perspectivas ofrecen un marco para la investigación y desarrollo continuo de este proyecto, proporcionando oportunidades para contribuciones significativas en el ámbito académico y tecnológico.

**REFERENCIAS BIBLIOGRAFICA.**

**Web electrosoftcloud**

<https://www.electrosoftcloud.com/wlan-en-esp32-primeros-pasos-con-el-wifi/>

**Sistema Operativo RTO**

<https://freertos.org/Documentation/Mastering-the-FreeRTOS-Real-Time-Kernel.v1.0.pdf>

**Tecnotízate**

<https://tecnotizate.es/esp32-mapeo-de-pines-y-sensores-internos/>

**PuntoFlotante.net**

<https://www.puntoflotante.net/MANUAL-BASICO-NODEMCU-ESP32-ARDUINO.pdf>

**Manual del fabricante ESP32**

<https://www.alldatasheet.com/view.jsp?Searchword=Esp-wroom-32%20datasheet&gad_source=1&gclid=CjwKCAiAopuvBhBCEiwAm8jaMUrA9d3ZAeRBJZGn3q_aJnhFT3ny3V6c93gDRD1OtHYEOarCMxLaWRoCNPYQAvD_BwE>

**Web Electronics**

<https://vasanza.blogspot.com/2021/07/especificaciones-del-modulo-esp32.html>