TECNICATURA SUPERIOR EN

Telecomunicaciones

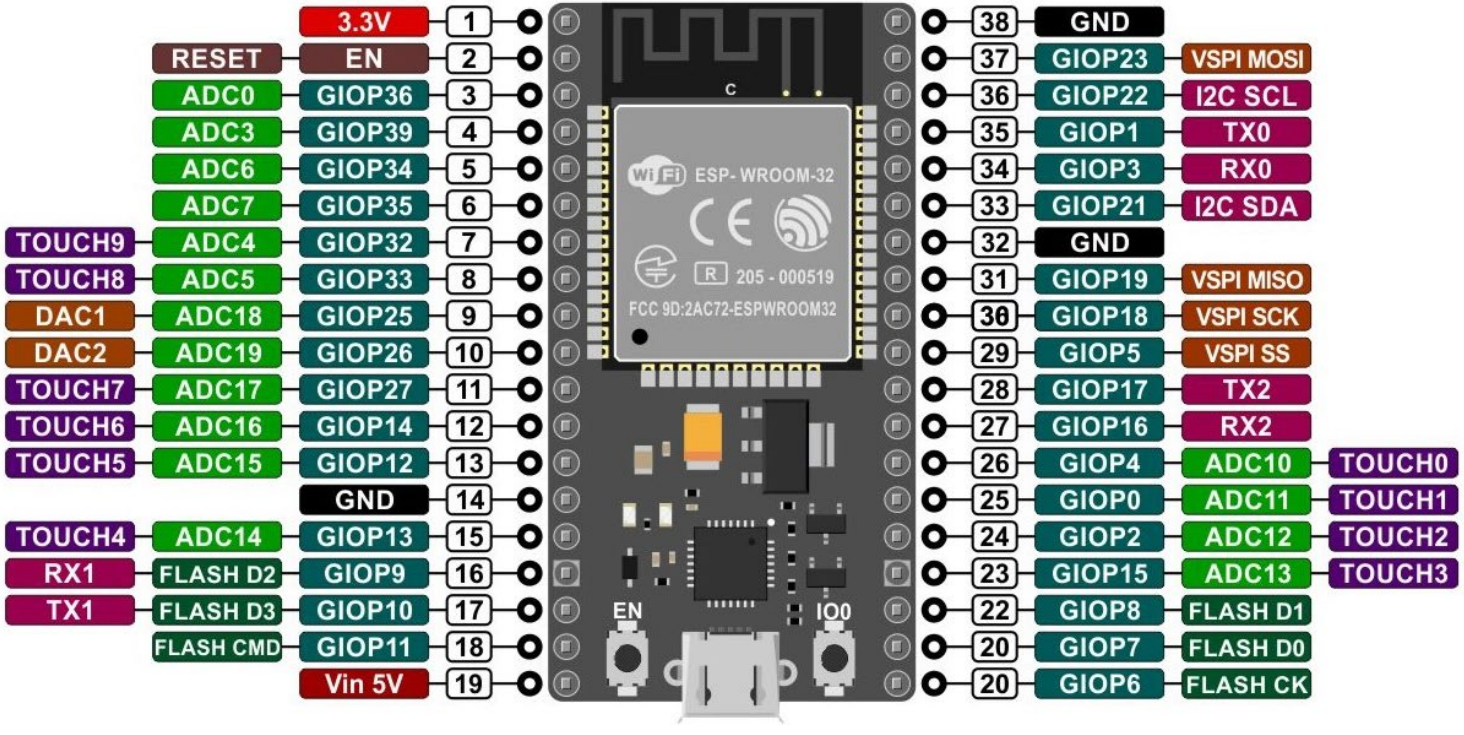
Electrónica Microcontrolada

Optimización 2: Comunicación y Preprocesamiento

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Dirección General de EDUCACIÓN TÉCNICA Y FORMACIÓN PROFESIONAL
Ministerio de EDUCACIÓN
CÓRDOBA entre todos

Optimización 2: Comunicación y Preprocesamiento



Optimización I: Infraestructura

1. Introducción

- **1.1 Procesamiento en Cloud**
  - Infraestructura del servidor (hardware, virtualización, software, etc.).
  - Componente Software del Cloud (microservicios, APIs, middleware).
  - Aplicaciones del procesamiento en Cloud (visualización, generación de informes, toma de decisiones automatizadas).

- **1.2 Presentacion del proyecto de la Unidad**
  - Este proyecto busca implementar prácticas de sensorización en una arquitectura Edge/Fog utilizando ESP32 para la adquisición de datos y herramientas como Grafana para su visualización.

Optimización I: Infraestructura

- **1.3 Objetivos de las prácticas**
  - Implementar un sistema de sensorización utilizando la arquitectura Edge/Fog.Configurar el stack tecnológico para visualizar los datos en Grafana.
  - Explorar la eficiencia de protocolos de comunicación en la infraestructura IoT.
  - Optimizar el uso de servidores y herramientas para la gestión y análisis de datos en tiempo real.

2. Stack Tecnológico

- **2.1 Protocolos de Comunicación en Edge**
  - Wi-Fi: Configuración y aplicaciones para IoT.
  - Bluetooth Low Energy (BLE): Casos de uso y ventajas en IoT.
  - ESP-NOW: Comunicación directa entre dispositivos sin enrutador.
  - ESP-Mesh: Creación de redes Mesh con ESP32.
  - Matter (opcional): Interoperabilidad en IoT, potencial uso.

Optimización I: Infraestructura

- **2.2 Protocolos de Comunicación en Fog**
  - HTTP: Comunicación estándar entre Edge y Fog.
  - MQTT: Protocolo ligero para IoT, eficiencia en la transmisión de datos.
  - WebSocket: Comunicación bidireccional en tiempo real, casos de uso.

- **2.3 Infraestructura de Visualización**
  - Node-Red: Orquestación de dispositivos y flujos de trabajo IoT.
  - Grafana: Herramienta de visualización de datos IoT.
  - InfluxDB: Base de datos para series temporales, integración con Grafana.
  - Flask: Framework para construir una API RESTful y backend del sistema.
  - Nginx: Servidor web y balanceador de carga para mejorar la eficiencia.

Optimización I: Infraestructura

3. Prácticas de Implementación

- **3.1. Implementación 1: Sensorización y Visualización en Grafana**
  - Configuración del servidor con Flask.
  - Almacenamiento de datos en InfluxDB.
  - Visualización de los datos de los sensores en Grafana.

- **3.2. Implementación 2: Orquestación con Node-Red**
  - Implementación de flujos de trabajo con Node-Red.
  - Sensorización y visualización en tiempo real.
  - Comparación con la visualización en Grafana.

Optimización I: Infraestructura

- **3.3. Implementación 3: Integración con MySQL**
  - Configuración de MySQL como base de datos principal.
  - Uso de Nginx y Flask para manejo del servidor.
  - Sensorización con almacenamiento y representación de datos.

- **3.4. Implementación 4: Sensorización Completa con Nginx, Flask y Grafana**
  - Integración total del stack con Nginx, Flask, InfluxDB y Grafana.
  - Visualización y monitoreo en tiempo real.
  - Optimización del sistema de sensorización.

Optimización I: Infraestructura

4. Conclusiones

- Principales aprendizajes sobre la infraestructura Edge/Fog.
- Ventajas y desventajas de los diferentes protocolos y tecnologías.
- Recomendaciones para mejorar la infraestructura en proyectos futuros.

Optimización I: Infraestructura

Componentes y Aplicaciones

Optimización I: Infraestructura

Infraestructura del Servidor
La infraestructura en la nube proporciona la capacidad de escalabilidad y flexibilidad necesarias para manejar grandes volúmenes de datos. Algunos componentes son:

- **Hardware**: Puede variar entre servidores dedicados, sistemas distribuidos o soluciones en la nube como AWS, Azure o Google Cloud.
- **Virtualización**: El uso de máquinas virtuales permite la creación de entornos separados que optimizan el uso de los recursos físicos.
- **Contenedores**: Tecnologías como **Docker** o **Kubernetes** permiten un mejor control de la infraestructura y una implementación más ágil de servicios.
- **Software**: Desde sistemas operativos específicos (Linux) hasta entornos de desarrollo como **Flask** y **Nginx**, que gestionan la lógica de negocio y la interfaz de comunicación entre dispositivos Edge/Fog.

Optimización I: Infraestructura

Componentes Software del Cloud

- **Microservicios**: Arquitectura basada en servicios independientes que se comunican a través de APIs. Facilitan la escalabilidad y el mantenimiento de aplicaciones IoT.

- **APIs**: Las **APIs RESTful** permiten la interacción entre los diferentes componentes de la infraestructura (dispositivos, bases de datos, visualización).

- **Middleware**: Actúa como puente entre los servicios, asegurando la correcta transmisión de datos y la integración de sistemas. Incluye herramientas como **MQTT**, **RabbitMQ**, o **Apache Kafka** para mensajería.

Optimización I: Infraestructura

Aplicaciones del Cloud

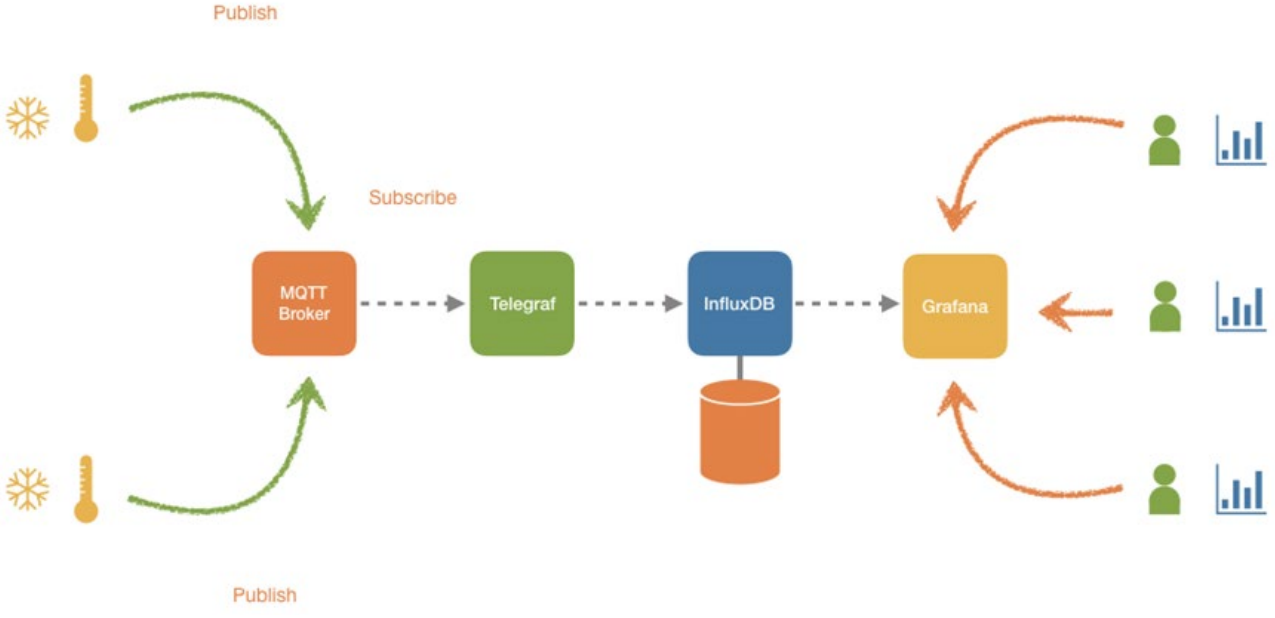
- **Visualización**: Herramientas como **Grafana** permiten crear dashboards interactivos para monitorear los datos recolectados desde la infraestructura Edge/Fog en tiempo real.

- **Generación de informes**: A partir de los datos recolectados y procesados en la nube, se pueden generar informes automáticos que faciliten la toma de decisiones.

- **Toma de decisiones automatizadas**: Mediante el uso de **Machine Learning** o reglas predefinidas, el sistema puede realizar ajustes automáticos en la infraestructura (ej. activar alertas, optimizar recursos).
- El procesamiento en la nube facilita una integración más eficiente con sistemas de inteligencia artificial y análisis predictivo.

Optimización I: Infraestructura

Proyectos de ejemplos



Optimización I: Infraestructura

Descripción general del proyecto IoT

Arquitectura Edge/Fog y Visualización de Datos

- El proyecto se enfoca en implementar una arquitectura **Edge/Fog** utilizando dispositivos **ESP32** para sensorización.
- Se capturan datos de humedad del suelo, temperatura y luz utilizando sensores como el **DHT11** y el **LDR**.
- Los datos serán enviados a la infraestructura Fog para su procesamiento inicial y luego almacenados en bases de datos locales como **InfluxDB**.
- **Grafana** se utiliza para la visualización de los datos en tiempo real.
- El servidor local (Fog) usará **Flask** y **Nginx** para gestionar la recepción y exposición de los datos.
- Este enfoque permite reducir la latencia y mejorar la toma de decisiones en el procesamiento de los datos IoT.

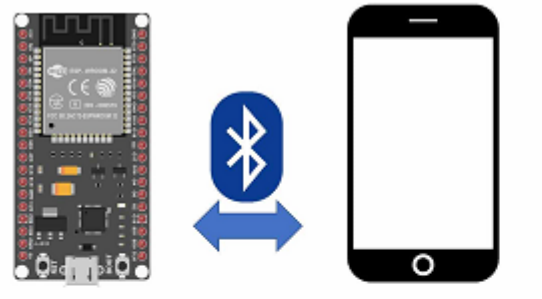
Optimización I: Infraestructura

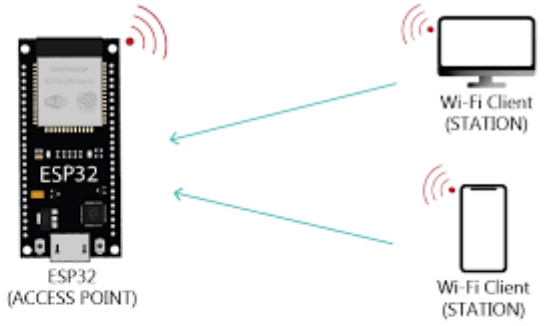
Objetivos

- **Implementar un sistema de sensorización** que recoja datos del entorno utilizando dispositivos **ESP32** en la arquitectura Edge/Fog.
- **Configurar el stack tecnológico** que incluye **Grafana** para la visualización e **InfluxDB** como soporte para el almacenamiento.
- **Evaluar los protocolos de comunicación** en Edge y Fog, como **MQTT**, **HTTP**, y **Protocolos Mesh**, utilizados en diferentes escenarios.
- **Optimizar el uso de servidores locales** mediante la integración de tecnologías como **Flask**, **Nginx**, y **Node-Red** para la gestión y visualización de los datos en tiempo real.

Optimización I: Infraestructura

Optimización 2: Comunicación y Preprocesamiento





Optimización 2: Comunicación y Preprocesamiento

Wi-Fi

- El módulo Wi-Fi es una de las características más poderosas del ESP32. Permite la conexión a redes Wi-Fi en las bandas de 2.4 GHz y soporta los protocolos 802.11 b/g/n. Sus características más destacadas son:

- **Modos de Operación**:
  - **Estación (STA)**: Permite al ESP32 conectarse a un punto de acceso (router) para acceder a internet o a una red local.
  - **Punto de Acceso (AP)**: Permite al ESP32 actuar como un punto de acceso, creando su propia red Wi-Fi a la que otros dispositivos pueden conectarse.
  - **Modo Dual (STA + AP)**: El ESP32 puede funcionar simultáneamente como estación y punto de acceso.

Optimización 2: Comunicación y Preprocesamiento

- **Seguridad**: Soporta los estándares de seguridad WPA/WPA2.
- **Protocolo TCP/IP**: El ESP32 tiene una pila completa de TCP/IP, permitiendo el desarrollo de aplicaciones cliente/servidor HTTP, MQTT, WebSockets, etc.
- **Aplicaciones Típicas**: Comunicación con servidores en la nube, control de dispositivos vía web, actualización OTA (Over-The-Air), etc.

Optimización 2: Comunicación y Preprocesamiento

Bluetooth

- El ESP32 incluye soporte tanto para Bluetooth clásico (BR/EDR) como para Bluetooth Low Energy (BLE), lo que lo hace muy versátil para aplicaciones que requieren comunicación de corto alcance:

- **Bluetooth Clásico (BR/EDR)**:
  - **Velocidad de transmisión**: Mayor que BLE, adecuado para aplicaciones que requieren enviar grandes cantidades de datos.
  - **Perfiles**: Soporta perfiles Bluetooth estándar como SPP (Serial Port Profile) para comunicación serial inalámbrica.
  - **Aplicaciones Típicas**: Transferencia de archivos, audio inalámbrico, control de dispositivos.

Optimización 2: Comunicación y Preprocesamiento

- **Bluetooth Low Energy (BLE)**:
  - **Consumo de Energía**: Mucho más bajo que Bluetooth clásico, ideal para aplicaciones de bajo consumo.
  - **Roles**: Puede actuar como un dispositivo central o periférico.
  - **GATT (Generic Attribute Profile)**: Permite la comunicación estructurada en forma de servicios y características, facilitando la creación de aplicaciones personalizadas.
  - **Aplicaciones Típicas**: Dispositivos wearables, sensores, sistemas de notificación y control remoto.

Optimización 2: Comunicación y Preprocesamiento

ESP-NOW

- ESP-NOW es un protocolo de comunicación inalámbrica propietario de Espressif, que permite la comunicación directa entre dispositivos ESP sin necesidad de una red Wi-Fi. Algunas de sus características clave son:

- **Baja Latencia**: Comunicación casi instantánea, con latencias típicas de menos de 2 ms.
- **Bajo Consumo de Energía**: Ideal para aplicaciones donde la eficiencia energética es crítica.
- **No requiere emparejamiento**: Los dispositivos ESP pueden comunicarse entre sí directamente, sin necesidad de configuraciones previas complejas.
- **Comunicación Unicast y Broadcast**: Permite enviar mensajes a un único dispositivo o a múltiples dispositivos simultáneamente.
- **Aplicaciones Típicas**: Redes de sensores, sistemas de control remoto, aplicaciones de IoT en malla.

Optimización 2: Comunicación y Preprocesamiento

Soporte para Módulos Externos (LoRa, Zigbee, etc.)

- Aunque el ESP32 no tiene soporte nativo para protocolos como LoRa o Zigbee, puede integrarse fácilmente con módulos externos a través de interfaces como SPI o UART:

- **LoRa**:
  - **Largo Alcance**: Permite la comunicación a distancias de varios kilómetros con un consumo energético muy bajo.
  - **Aplicaciones Típicas**: Redes de sensores distribuidos, monitoreo ambiental, aplicaciones agrícolas.

- **Zigbee**:
  - **Malla**: Permite la creación de redes en malla para la comunicación entre múltiples dispositivos.
  - **Aplicaciones Típicas**: Domótica, sistemas de iluminación inteligente, redes de sensores industriales.

Optimización 2: Comunicación y Preprocesamiento

Característica	Wi-Fi	Bluetooth Clásico	Bluetooth Low Energy (BLE)	ESP-NOW
Alcance	Hasta 100 metros (en línea de vista)	Hasta 10 metros	Hasta 50 metros	Hasta 100 metros (en línea de vista)
Velocidad de Transmisión	Hasta 150 Mbps (802.11n)	Hasta 3 Mbps	Hasta 1 Mbps	~1 Mbps
Consumo de Energía	Alto	Medio	Bajo	Muy bajo
Topología	Punto a punto, Infraestructura (Cliente/Servidor)	Punto a punto	Punto a punto, Estrella	Unicast, Broadcast
Seguridad	WPA/WPA2	PIN, Paring (simple)	Autenticación y Encriptación (AES)	Encriptación (AES)
Complejidad de Implementación	Alta (Configuración de red, pila TCP/IP)	Media (Emparejamiento, Perfiles)	Baja (GATT, Perfiles)	Baja (Sin configuración de red)

Optimización 2: Comunicación y Preprocesamiento

Característica	Wi-Fi	Bluetooth Clásico	Bluetooth Low Energy (BLE)	ESP-NOW
Latencia	Media-Alta	Media	Baja	Muy baja (menor a 2 ms)
Aplicaciones Típicas	Transmisión de datos a internet, servidores web, MQTT	Transferencia de archivos, Audio inalámbrico	Wearables, Sensores, Dispositivos de notificación	Redes de sensores, Comunicación rápida entre dispositivos
Modos de Operación	Estación, Punto de Acceso (AP), AP+Estación	Maestro-Esclavo	Central, Periférico	Emisor, Receptor
Configuración Inicial	Compleja (SSID, Contraseña, etc.)	Media (Emparejamiento)	Sencilla (Servicios y Características)	Muy sencilla (Direcciones MAC)
Coste (en términos de recursos del ESP32)	Alto (CPU y Memoria)	Medio	Bajo	Muy bajo

Optimización 2: Comunicación y Preprocesamiento

Optimización 2: Comunicación y Preprocesamiento

Teoría sobre Wi-Fi para el ESP32

1. Conceptos Básicos de Wi-Fi

- Wi-Fi (Wireless Fidelity) es una tecnología de red inalámbrica que permite la transmisión de datos a través de ondas de radio. Los dispositivos Wi-Fi, como el ESP32, pueden conectarse a una red local o a internet a través de un punto de acceso (AP) o crear su propia red para permitir que otros dispositivos se conecten.

Optimización 2: Comunicación y Preprocesamiento

- **Frecuencia**: El ESP32 opera en la banda de 2.4 GHz, comúnmente usada por la mayoría de los dispositivos Wi-Fi. Esto permite velocidades de transmisión relativamente altas a distancias cortas o moderadas.

- **Protocolos**: El ESP32 soporta los estándares IEEE 802.11 b/g/n:
  - **802.11b**: Hasta 11 Mbps
  - **802.11g**: Hasta 54 Mbps
  - **802.11n**: Hasta 150 Mbps

- **Canales**: La banda de 2.4 GHz se divide en varios canales. El ESP32 puede seleccionar automáticamente el canal adecuado o configurarse manualmente.

Optimización 2: Comunicación y Preprocesamiento

2. Modos de Operación del ESP32 en Wi-Fi

- El ESP32 puede funcionar en diferentes modos de Wi-Fi según los requisitos de la aplicación:

- **2.1 Modo Estación (STA - Station Mode)**
- En este modo, el ESP32 se comporta como un cliente que se conecta a un punto de acceso (AP) existente, como un router Wi-Fi. Este es el modo típico cuando queremos que el ESP32 acceda a internet o se comunique con otros dispositivos en la misma red.

Optimización 2: Comunicación y Preprocesamiento

- **Características**:
  - Se conecta a una red Wi-Fi especificando un SSID (nombre de la red) y una contraseña.
  - Una vez conectado, obtiene una dirección IP, ya sea automáticamente (DHCP) o manualmente (IP estática).

- **Aplicaciones Típicas**:
  - Enviar datos a un servidor en la nube.
  - Recibir comandos desde una aplicación móvil.
  - Actualizaciones OTA (Over-The-Air) para firmware.

Optimización 2: Comunicación y Preprocesamiento

2.2 Modo Punto de Acceso (AP - Access Point Mode)

- En este modo, el ESP32 actúa como un punto de acceso, creando su propia red Wi-Fi. Otros dispositivos pueden conectarse a esta red para comunicarse directamente con el ESP32.

- **Características**:
  - Crea su propia red Wi-Fi con un SSID y contraseña definidos por el usuario.
  - Puede configurarse con seguridad WPA/WPA2 para proteger la red.
- **Aplicaciones Típicas**:
  - Crear una red local para conectar múltiples dispositivos en un entorno donde no hay acceso a un router.
  - Configurar el ESP32 como un portal cautivo para configuración inicial.

Optimización 2: Comunicación y Preprocesamiento

2.3 Modo Dual (AP + STA)

- El ESP32 puede funcionar simultáneamente como Estación y Punto de Acceso. En este modo, se conecta a una red Wi-Fi existente mientras también crea su propia red Wi-Fi para otros dispositivos.

- **Características**:
  - Permite al ESP32 acceder a internet y, al mismo tiempo, permitir que otros dispositivos se conecten a él.
- **Aplicaciones Típicas**:
  - Actuar como un puente entre una red local y dispositivos periféricos.
  - Permitir configuración local mientras mantiene conectividad con la nube.

Optimización 2: Comunicación y Preprocesamiento

3. Configuración de Wi-Fi en el ESP32

- Para usar el Wi-Fi en el ESP32, necesitamos configurar varios parámetros básicos en nuestro código, ya sea usando el framework Arduino o PlatformIO.

- **Aplicaciones Comunes con Wi-Fi en el ESP32**
- **Servidor Web**: El ESP32 puede actuar como un servidor web, sirviendo páginas HTML para controlar dispositivos conectados.
- **Cliente HTTP**: El ESP32 puede enviar datos a un servidor remoto o a una API REST.
- **MQTT**: El ESP32 puede comunicarse con un broker MQTT para aplicaciones de IoT, publicando y suscribiéndose a mensajes.
- **OTA Updates**: Permite actualizar el firmware del ESP32 a través de Wi-Fi.

Optimización 2: Comunicación y Preprocesamiento

Consideraciones

- **A. Gestión de Errores y Reconexión Automática**
- El Wi-Fi no siempre es estable, por lo que es fundamental implementar mecanismos para reconexión automática en caso de que se pierda la conexión.

- **B. Optimización del Consumo de Energía**
- El módulo Wi-Fi del ESP32 es uno de los mayores consumidores de energía, por lo que es esencial gestionar el Wi-Fi de forma eficiente, activándolo solo cuando sea necesario y utilizando modos de bajo consumo cuando el dispositivo no esté activo.

Optimización 2: Comunicación y Preprocesamiento

Optimización 2: Comunicación y Preprocesamiento

Teoría sobre Bluetooth para el ESP32

1. Conceptos Básicos de Bluetooth

- Bluetooth es una tecnología de comunicación inalámbrica de corto alcance que permite la transmisión de datos entre dispositivos. El ESP32 soporta dos tipos de Bluetooth:

- **Bluetooth Clásico (BR/EDR)**: Diseñado para aplicaciones que requieren una transmisión de datos más rápida, como audio o transferencia de archivos.

- **Bluetooth Low Energy (BLE)**: Optimizado para aplicaciones de bajo consumo de energía, como sensores, dispositivos wearables y aplicaciones IoT.

Optimización 2: Comunicación y Preprocesamiento

2. Modos de Operación del Bluetooth en el ESP32

- **2.1 Bluetooth Clásico (BR/EDR)**
- El Bluetooth Clásico es adecuado para aplicaciones que necesitan una transmisión de datos constante o de alta velocidad. Es comúnmente utilizado para:

- Comunicación serie entre dispositivos (SPP - Serial Port Profile).
- Transferencia de archivos.
- Transmisión de audio (aunque esta función es limitada en el ESP32).

Optimización 2: Comunicación y Preprocesamiento

- **Características Clave**
- **Velocidad**: Hasta 3 Mbps.
- **Emparejamiento**: Requiere un proceso de emparejamiento (pairing) para conectar dispositivos.
- **Perfiles Soportados**: El ESP32 soporta principalmente el perfil SPP, que permite la comunicación en serie.

- **Aplicaciones Típicas**
- Comunicación serial inalámbrica entre el ESP32 y otros dispositivos, como smartphones o PCs.
- Control de dispositivos desde una aplicación móvil.
- Transferencia de datos sin conexión a internet.

Optimización 2: Comunicación y Preprocesamiento

2.2 Bluetooth Low Energy (BLE)

- BLE está diseñado para aplicaciones que requieren baja energía y transferencia de datos intermitente. Es ideal para dispositivos IoT que envían pequeñas cantidades de datos de forma ocasional.

- **Características Clave**
- **Velocidad**: Hasta 1 Mbps.
- **Consumo de Energía**: Mucho más bajo que Bluetooth Clásico, ideal para dispositivos alimentados por batería.

Optimización 2: Comunicación y Preprocesamiento

- **Estructura de Comunicación**:
  - **Roles**: Un dispositivo BLE puede actuar como "Central" o "Periférico".
  - **GATT (Generic Attribute Profile)**: Define cómo se estructuran y transmiten los datos. Incluye "Servicios" y "Características":
    - **Servicios**: Conjuntos de datos relacionados.
    - **Características**: Unidades individuales de datos (lectura/escritura).

- **Aplicaciones Típicas**
  - Sensores y dispositivos wearables que envían datos a un smartphone.
  - Comunicación entre dispositivos IoT en aplicaciones de hogar inteligente.
  - Notificaciones y control remoto desde dispositivos móviles.

Optimización 2: Comunicación y Preprocesamiento

- **Comparación entre Bluetooth Clásico y BLE**

Característica	Bluetooth Clásico	BLE
Velocidad	Hasta 3 Mbps	Hasta 1 Mbps
Consumo de Energía	Medio	Bajo
Alcance	Hasta 10 metros	Hasta 50 metros
Emparejamiento	Necesario	Opcional
Estructura de Datos	Perfiles estándar (SPP, A2DP)	GATT (Servicios y Características)
Aplicaciones Típicas	Audio, Transferencia de archivos, Comunicación serie	Sensores, Dispositivos portátiles, IoT

Optimización 2: Comunicación y Preprocesamiento

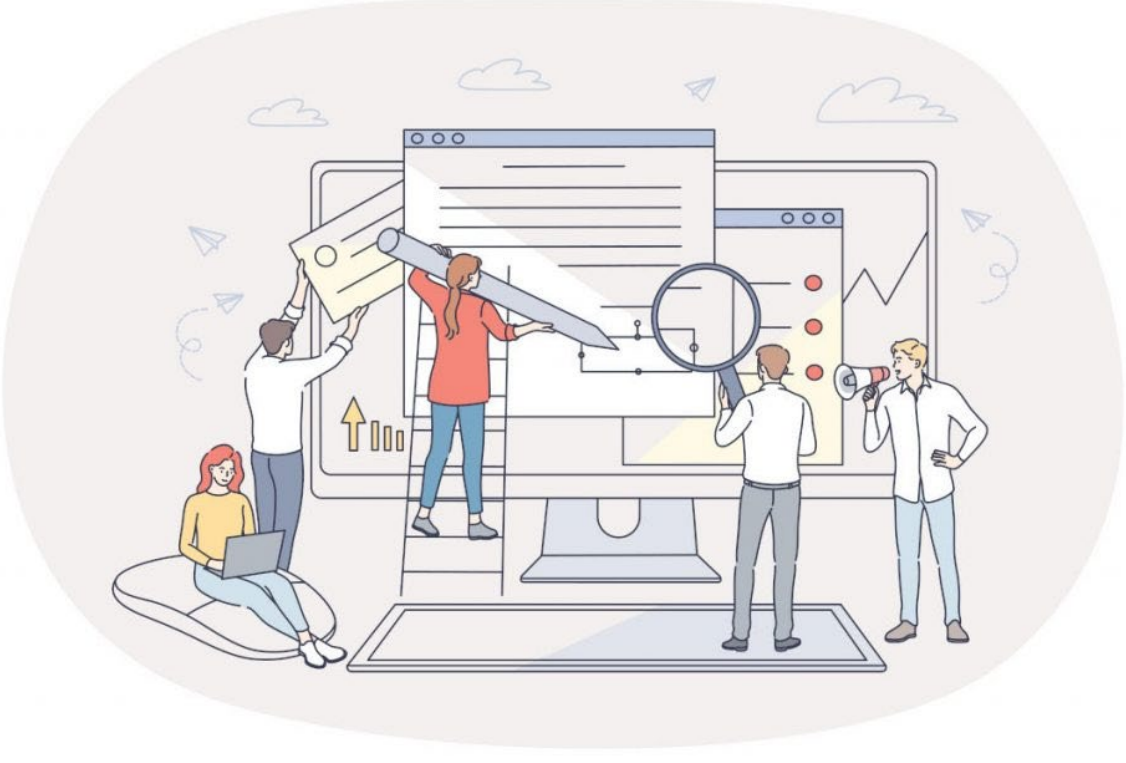
Consideraciones

- **Seguridad en Bluetooth**
- **Bluetooth Clásico**: Utiliza un proceso de emparejamiento con PIN o clave de acceso para establecer una conexión segura.
- **BLE**: Utiliza procesos de autenticación y encriptación, incluidos métodos como Just Works, Passkey, y OOB (Out-of-Band) para proteger la comunicación.

- **Aplicaciones Comunes con Bluetooth en el ESP32**
- **Bluetooth Clásico (SPP)**: Transmisión de datos en serie para control y monitoreo.
- **BLE**: Comunicaciones de baja energía para aplicaciones como seguimiento de actividad, control de dispositivos, y sensores de baja potencia.

- **Optimización y Buenas Prácticas**
- **Gestión de la Energía**: Especialmente con BLE, activar el modo de bajo consumo cuando no se esté transmitiendo activamente.
- **Gestión de la Conexión**: Establecer y liberar conexiones eficientemente para ahorrar energía.

Optimización 2: Comunicación y Preprocesamiento



Optimización 2: Comunicación y Preprocesamiento

TP#20. Trabajo Practico

Objetivos:

- Familiarizarse con las tecnologías de los módulos de comunicación incorporados en el ESP32.
- Conocer las librerías asociadas a cada modulo de comunicación para el ESP32 incluidas en el framework Arduino.

Realizar los siguientes ejercicios según se solicita a continuación, dejar los registros de desarrollo en el repositorio personal de la materia, con la estructura ABP de trabajo habitual.

Optimización 2: Comunicación y Preprocesamiento

Serie de Prácticas Wi-Fi con el ESP32

- **Práctica 1: Conexión Básica a una Red Wi-Fi y Monitorización de Estado**

- **Objetivo**
- Aprender a conectar el ESP32 a una red Wi-Fi en modo Estación (STA), monitorear el estado de la conexión y manejar la reconexión automática en caso de pérdida de la red.

- **Librerías Necesarias**
- **WiFi.h:** Librería estándar de Arduino para manejo de Wi-Fi.

Optimización 2: Comunicación y Preprocesamiento

Pasos a Seguir

1. **Configuración del Entorno de Desarrollo**
   - Configura un nuevo proyecto en PlatformIO para el ESP32 utilizando el framework Arduino.

2. **Implementar Conexión Wi-Fi**
   - Define las credenciales de la red Wi-Fi (SSID y contraseña).
   - Implementa la lógica para conectar el ESP32 a la red Wi-Fi.
   - Agrega un bucle que monitorice la conexión Wi-Fi.

3. **Monitorizar Estado de Conexión**
   - Imprime el estado de la conexión en el Monitor Serie.
   - Implementa la reconexión automática si se pierde la conexión.

Optimización 2: Comunicación y Preprocesamiento

4. **Prueba de Funcionamiento**
    - Prueba el programa con credenciales correctas e incorrectas.
    - Deshabilita el Wi-Fi del router para comprobar la reconexión automática.

- **Herramientas Externas**
- Un router Wi-Fi.
- Monitor Serie de VSCode.

- **Recomendaciones**
- Implementa un tiempo de espera razonable para la reconexión para evitar bloqueos.

Optimización 2: Comunicación y Preprocesamiento

Práctica 2: Configurar el ESP32 como Punto de Acceso (AP)

- **Objetivo**
- Configurar el ESP32 como un punto de acceso (AP) para que otros dispositivos se conecten a él.

- **Librerías Necesarias**
- **WiFi.h:** Librería estándar para manejar la conexión Wi-Fi en modo AP.

Optimización 2: Comunicación y Preprocesamiento

Pasos a Seguir

1. **Configuración del Modo AP**
   - Configura el ESP32 para que funcione en modo Punto de Acceso.
   - Define el SSID y la contraseña de la red que creará el ESP32.

2. **Iniciar el Punto de Acceso**
   - Implementa la lógica para iniciar el punto de acceso y mostrar la dirección IP asignada en el Monitor Serie.

3. **Monitorear Conexiones**
   - Implementa una función que muestre la cantidad de dispositivos conectados al punto de acceso.

Optimización 2: Comunicación y Preprocesamiento

4. **Prueba de Funcionamiento**
    - Conéctate a la red creada por el ESP32 desde un smartphone o PC.
    - Verifica en el Monitor Serie el número de dispositivos conectados.

- **Herramientas Externas**
- Un smartphone o PC con Wi-Fi.

- **Recomendaciones**
- Usa una contraseña segura para evitar conexiones no autorizadas.

Optimización 2: Comunicación y Preprocesamiento

Práctica 3: Servidor Web Simple con Wi-Fi

- **Objetivo**
- Crear un servidor web en el ESP32 para controlar un LED desde un navegador web.

- **Librerías Necesarias**
- **WiFi.h:** Para la conexión Wi-Fi.
- **WebServer.h:** Para implementar el servidor web.

Optimización 2: Comunicación y Preprocesamiento

Pasos a Seguir

1. **Configurar Conexión Wi-Fi**
   - Conecta el ESP32 a una red Wi-Fi.

2. **Implementar el Servidor Web**
   - Configura un servidor web que escuche en el puerto 80.
   - Crea una página web simple con dos botones para encender y apagar un LED.

3. **Controlar el LED**
   - Define rutas para encender y apagar el LED.
   - Implementa la lógica para cambiar el estado del LED según las peticiones recibidas.

Optimización 2: Comunicación y Preprocesamiento

4. **Prueba de Funcionamiento**
   - Accede a la página web alojada por el ESP32 desde un navegador.
   - Interactúa con los botones y observa el comportamiento del LED.

- **Herramientas Externas**
- Un navegador web.
- Un LED conectado al ESP32 (o usar el LED integrado).

- **Recomendaciones**
- Mejora la interfaz de la página web para obtener una experiencia más interactiva.

Optimización 2: Comunicación y Preprocesamiento

Práctica 4: Cliente HTTP para Enviar Datos a un Servidor

- **Objetivo**
- Configurar el ESP32 como cliente HTTP para enviar datos a un servidor remoto.

- **Librerías Necesarias**
- **WiFi.h:** Para la conexión Wi-Fi.
- **HTTPClient.h:** Para realizar solicitudes HTTP.

Optimización 2: Comunicación y Preprocesamiento

Pasos a Seguir

1. **Configurar Conexión Wi-Fi**
   - Conecta el ESP32 a una red Wi-Fi.

2. **Configurar Cliente HTTP**
   - Configura el ESP32 para que realice solicitudes HTTP POST a un servidor.
   - Envía un conjunto de datos (por ejemplo, valores de sensores) al servidor.

3. **Manejar Respuestas del Servidor**
   - Implementa la lógica para recibir y mostrar la respuesta del servidor en el Monitor Serie.

Optimización 2: Comunicación y Preprocesamiento

4. **Prueba de Funcionamiento**
   - Configura un servidor web o usa una API pública para recibir los datos.
   - Observa la respuesta del servidor y verifica que los datos se envíen correctamente.

- **Herramientas Externas**
- Un servidor web o servicio API REST.
- Alternativamente, usa herramientas como httpbin.org para pruebas

- **Recomendaciones**
- Asegúrate de manejar los errores de conexión y respuesta del servidor.
- Considera el uso de HTTPS para mayor seguridad si el servidor lo soporta.

Optimización 2: Comunicación y Preprocesamiento

Serie de Prácticas Bluetooth con el ESP32

Práctica 1: Comunicación Serial con Bluetooth Clásico (SPP)

- **Objetivo**
- Configurar el ESP32 para que funcione como un dispositivo Bluetooth Serial (SPP).
- Permitir la comunicación bidireccional entre el ESP32 y un dispositivo externo (como un smartphone o PC) usando Bluetooth Clásico.

- **Librerías Necesarias**
- **BluetoothSerial.h**: Librería de Arduino para manejar la comunicación Bluetooth Clásica en el ESP32.

Optimización 2: Comunicación y Preprocesamiento

Pasos a Seguir

1. **Configurar Bluetooth Clásico**
   - Configura el ESP32 para que funcione en modo Bluetooth Clásico con el perfil de puerto serie (SPP).
   - Define un nombre de dispositivo Bluetooth para el ESP32.
2. **Implementar Comunicación Serial**
   - Configura el ESP32 para que pueda enviar y recibir datos a través del puerto serie Bluetooth.
   - Envía mensajes recibidos por Bluetooth al Monitor Serie y viceversa.
3. **Emparejamiento**
   - Haz que el ESP32 sea detectable para otros dispositivos Bluetooth para permitir el emparejamiento.

Optimización 2: Comunicación y Preprocesamiento

4. **Prueba de Funcionamiento**
   - Desde un smartphone o PC, empareja el dispositivo con el ESP32.
   - Utiliza una aplicación de terminal Bluetooth (como "Serial Bluetooth Terminal" en Android) para enviar y recibir datos.
   - Observa el intercambio de datos en tiempo real.

- **Herramientas Externas**
- Una aplicación de terminal Bluetooth en un smartphone o PC.
- Monitor Serie de VSCode para observar los datos recibidos.

- **Recomendaciones**
- Asegúrate de manejar correctamente la desconexión y reconexión de dispositivos Bluetooth.

Optimización 2: Comunicación y Preprocesamiento

Práctica 2: Comunicación BLE Básica - Anunciado y Conexión

- **Objetivo**
- Configurar el ESP32 como un dispositivo Bluetooth Low Energy (BLE).
- Implementar el anunciado de un dispositivo BLE y permitir que otros dispositivos se conecten a él.

- **Librerías Necesarias**
- **BLEDevice.h**, **BLEServer.h**, **BLEUtils.h**: Librerías de Arduino para manejar BLE en el ESP32.

Optimización 2: Comunicación y Preprocesamiento

Pasos a Seguir

1. **Configurar BLE**
   - Configura el ESP32 como un periférico BLE.
   - Define un nombre de dispositivo BLE para el ESP32.
2. **Anunciado**
   - Implementa el anunciado para que el ESP32 sea visible para otros dispositivos BLE cercanos.
3. **Prueba de Conexión**
   - Permite que un dispositivo externo, como un smartphone, se conecte al ESP32.

Optimización 2: Comunicación y Preprocesamiento

4. **Prueba de Funcionamiento**
   - Utiliza una aplicación BLE en un smartphone (como "nRF Connect") para escanear dispositivos BLE y encontrar el ESP32.
   - Conéctate al ESP32 y observa el estado de la conexión.

- **Herramientas Externas**
- Aplicación BLE en un smartphone, como "nRF Connect" o "BLE Scanner".

- **Recomendaciones**
- Usa un identificador único (UUID) para el dispositivo BLE para identificarlo claramente entre otros dispositivos.

Optimización 2: Comunicación y Preprocesamiento

Práctica 3: Comunicación BLE - Servicios y Características

- **Objetivo**
- Configurar el ESP32 para que actúe como un servidor GATT BLE.
- Definir servicios y características para permitir la lectura y escritura de datos desde un dispositivo externo.

- **Librerías Necesarias**
- **BLEDevice.h**, **BLEServer.h**, **BLEUtils.h**, **BLECharacteristic.h**: Librerías de Arduino para manejar BLE GATT.

Optimización 2: Comunicación y Preprocesamiento

Pasos a Seguir

1. **Configurar el Servidor GATT**
   - Configura el ESP32 para que actúe como un servidor GATT.

2. **Definir Servicios y Características**
   - Crea un servicio BLE y define al menos una característica que permita la lectura y la escritura de datos.

3. **Implementar Funciones de Lectura y Escritura**
   - Implementa funciones de callback para manejar las operaciones de lectura y escritura en la característica definida.

Optimización 2: Comunicación y Preprocesamiento

4. **Prueba de Funcionamiento**
   - Utiliza una aplicación BLE en un smartphone para escanear el ESP32.
   - Conéctate al dispositivo, encuentra el servicio y la característica, y realiza operaciones de lectura y escritura.

- **Herramientas Externas**
- Aplicación BLE en un smartphone, como "nRF Connect" o "BLE Scanner".

- **Recomendaciones**
- Usa UUIDs personalizados para los servicios y características.
- Implementa la lógica para manejar correctamente las solicitudes de lectura y escritura.

Optimización 2: Comunicación y Preprocesamiento

Práctica 4: Notificaciones BLE

- **Objetivo**
- Configurar el ESP32 para enviar notificaciones BLE a dispositivos conectados cuando cambie el valor de una característica.

- **Librerías Necesarias**
- **BLEDevice.h**, **BLEServer.h**, **BLEUtils.h**, **BLECharacteristic.h**: Librerías de Arduino para manejar BLE y notificaciones.

Optimización 2: Comunicación y Preprocesamiento

- **Pasos a Seguir**

1. **Configurar el Servidor GATT y Características**
   - Configura el ESP32 como un servidor GATT.
   - Crea una característica que permita el envío de notificaciones.
2. **Implementar Notificaciones**
   - Implementa la lógica para actualizar el valor de la característica y enviar una notificación a los dispositivos conectados cuando el valor cambie.
3. **Prueba de Funcionamiento**
   - Utiliza una aplicación BLE en un smartphone para conectarte al ESP32.
   - Activa las notificaciones en la característica y observa cómo se reciben automáticamente las actualizaciones de valor.

Optimización 2: Comunicación y Preprocesamiento

- **Herramientas Externas**
- Aplicación BLE en un smartphone, como "nRF Connect".

- **Recomendaciones**
- Asegúrate de enviar las notificaciones en intervalos razonables para evitar el agotamiento de la batería.

Optimización 2: Comunicación y Preprocesamiento

Práctica 5: Comunicación Bidireccional BLE - Control Remoto

- **Objetivo**
- Configurar el ESP32 como un servidor BLE que permita recibir comandos para controlar un LED y enviar el estado del LED de vuelta al cliente.

- **Librerías Necesarias**
- **BLEDevice.h**, **BLEServer.h**, **BLEUtils.h**, **BLECharacteristic.h**: Librerías de Arduino para manejar BLE.

Optimización 2: Comunicación y Preprocesamiento

Pasos a Seguir

1. **Configurar el Servidor BLE**
   - Configura el ESP32 como un servidor BLE con dos características: una para recibir comandos de control (escritura) y otra para enviar el estado del LED (lectura).
2. **Implementar Control del LED**
   - Implementa la lógica para cambiar el estado del LED según los comandos recibidos.
3. **Implementar Lectura del Estado del LED**
   - Implementa la lógica para leer el estado del LED y enviar este estado a los dispositivos conectados.

Optimización 2: Comunicación y Preprocesamiento

4. **Prueba de Funcionamiento**
   - Utiliza una aplicación BLE en un smartphone para enviar comandos y recibir el estado del LED.
   - Observa cómo el LED cambia de estado según los comandos enviados.

- **Herramientas Externas**
- Aplicación BLE en un smartphone, como "nRF Connect".

- **Recomendaciones**
- Implementa mecanismos de seguridad, como la autenticación, si se usa en un entorno real.

Optimización 2: Comunicación y Preprocesamiento

¡Muchas gracias!