Logotipo

El contenido generado por IA puede ser incorrecto.UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA



Proyecto de Software:

“ScanEth: Mapeo de las señales Wifi de la UNI”

Integrantes:

Jaco Malpartida, Jazmin Fiorella

Mallqui Payano, Andree Alexander

Institución:

Universidad Nacional de Ingeniería

Curso:

Programación Orientada a Objetos

Fecha:

25 de junio del 2025

**Resumen**

Este proyecto plantea una solución práctica, económica y de bajo consumo para evaluar la cobertura de redes Wi-Fi dentro del campus de la Universidad Nacional de Ingeniería (UNI). Utilizando un ESP32 junto con un módulo GPS, se capturan en tiempo real los niveles de señal Wi-Fi junto con sus coordenadas geográficas. Estos datos se procesan automáticamente mediante scripts en Python que generan tanto archivos CSV como un mapa interactivo clasificado por calidad de señal (buena, intermedia o mala). Finalmente, toda la información se visualiza en una plataforma web accesible desde cualquier navegador, con funcionalidades como búsqueda dinámica y modo oscuro.

La herramienta no solo permite detectar zonas con mala cobertura en la universidad, sino que también puede ser adaptada fácilmente para su uso en otras instituciones o incluso en zonas rurales donde no existen diagnósticos técnicos formales de conectividad. Gracias a su diseño portátil y escalable, el sistema puede convertirse en una base para iniciativas de mejora en entornos con acceso limitado a internet.

**Abstract**

This project proposes a practical, affordable, and low-power solution to assess Wi-Fi coverage within the campus of the National University of Engineering (UNI). By using an ESP32 microcontroller along with a GPS module, Wi-Fi signal levels and their geographic coordinates are captured in real time. The data is automatically processed using Python scripts, which generate both CSV files and an interactive map classified by signal quality (good, intermediate, or poor). All the information is then displayed on a web platform accessible from any browser, featuring dynamic search and dark mode functionality.

The tool not only helps detect areas with poor coverage within the university, but it can also be easily adapted for use in other institutions or even rural areas where formal connectivity assessments are lacking. Thanks to its portable and scalable design, the system can serve as a foundation for initiatives aimed at improving internet access in underserved environments.

**Introducción**

En la era de la conectividad digital, el acceso confiable a redes inalámbricas se ha convertido en un recurso fundamental dentro de instituciones educativas. En universidades como la Universidad Nacional de Ingeniería (UNI), la red WiFi es una herramienta clave para el desarrollo académico, la investigación y la gestión administrativa. Sin embargo, es común encontrar zonas del campus con señal débil o inestable, lo que genera dificultades para los usuarios al momento de conectarse a internet.

Aunque existen diversas tecnologías para monitorear redes inalámbricas, muchas de ellas requieren equipos costosos o soluciones comerciales poco accesibles para instituciones educativas. Ante esta realidad, surge la necesidad de implementar herramientas portátiles, económicas y eficientes que permitan realizar diagnósticos de conectividad en tiempo real.

Este proyecto propone el desarrollo de un mapeador de señales WiFi utilizando un microcontrolador ESP32 y un módulo GPS, con el objetivo de recolectar datos sobre la intensidad de señal y su ubicación geográfica dentro del campus universitario. Estos datos se utilizarán para generar un mapa de cobertura WiFi, que permitirá identificar zonas con buena o deficiente conectividad, optimizando así la toma de decisiones para la mejora de la infraestructura de red.

El sistema se orienta no solo a solucionar una problemática técnica concreta, sino también a fomentar el desarrollo de soluciones tecnológicas locales, accesibles y replicables en otros contextos educativos. De este modo, el proyecto contribuye a la mejora continua del entorno digital universitario y al fortalecimiento de competencias en áreas como electrónica, programación y redes inalámbricas.

**Antecedentes:**

El desarrollo de herramientas para el análisis de cobertura inalámbrica ha sido abordado por diversas investigaciones y entidades, tanto a nivel nacional como internacional, lo cual demuestra la importancia creciente de contar con información precisa sobre la intensidad y calidad de señal en distintos entornos. Este proyecto se inspira en esas experiencias previas para adaptar una solución local, accesible y educativa aplicable al campus de la Universidad Nacional de Ingeniería (UNI). En el contexto nacional, el Organismo Supervisor de Inversión Privada en Telecomunicaciones (OSIPTEL) ofrece el portal interactivo “Cobertura Móvil”, que permite visualizar mapas georreferenciados de intensidad de señal de redes móviles a nivel nacional, diferenciadas por tecnología (2G, 3G, 4G, 5G) y por operador. Esta herramienta es de acceso público y permite a los usuarios conocer la calidad de señal en distintas regiones, departamentos y distritos. Aunque está enfocada en redes móviles y no en WiFi, sirve como referencia en cuanto a la forma de representar gráficamente la cobertura y orientar la toma de decisiones técnicas (OSIPTEL, 2024).

Asimismo, en la Pontificia Universidad Católica del Perú (PUCP) se desarrolló un sistema de monitoreo de espectro WiFi, enfocado en el análisis de canales inalámbricos y su utilización en el campus. El trabajo de Vásquez y Nilton (2016) consistió en una red de sensores que registraban la saturación de frecuencias en bandas de 2.4 GHz y 5 GHz, lo cual permitió analizar interferencias y proponer mejoras de canalización. Aunque no incorporaron georreferenciación, su propuesta sentó una base técnica para estudios de calidad de red en espacios académicos. A nivel internacional, Easha, Abbas y Daley (2020) desarrollaron el estudio “Campus Wi‑Fi Coverage Mapping and Analysis”, donde realizaron un análisis detallado de la cobertura de red WiFi en un entorno universitario. Utilizaron mediciones de RSSI (intensidad de señal recibida), encuestas de campo y mapas de calor para identificar puntos ciegos o de baja calidad de conexión. Este estudio es altamente relevante para el presente proyecto, ya que combina escaneo de redes con representación gráfica para optimizar la distribución de puntos de acceso.

Finalmente, se considera el trabajo de Flores Hernández y De los Reyes Quiroz (2017) en el Instituto Tecnológico Superior de Irapuato (ITESI), donde se diseñó e implementó un dispositivo basado en Arduino para medir la intensidad de señal WiFi en el Edificio B del campus. El prototipo incluía un módulo ESP8266, una pantalla OLED y tiras de LEDs RGB para representar gráficamente los niveles de señal. Los datos obtenidos se utilizaron para generar mapas de calor y fundamentar la necesidad de reubicar o instalar nuevos puntos de acceso. Esta experiencia resulta valiosa por su enfoque práctico y su similitud técnica con el presente proyecto.

Todos estos estudios y experiencias demuestran la necesidad y viabilidad de contar con sistemas que permitan analizar la cobertura de redes inalámbricas. El presente proyecto recoge sus aportes y los adapta a una propuesta concreta dentro del campus universitario, integrando detección WiFi, georreferenciación y visualización sobre mapas.

**Problemática**

En la actualidad, el acceso a redes WiFi se ha vuelto esencial para las actividades académicas y administrativas dentro de instituciones universitarias como la Universidad Nacional de Ingeniería (UNI). Sin embargo, estudiantes, docentes y el personal muchas veces enfrentan dificultades relacionadas con zonas de baja cobertura, desconexiones frecuentes o señales débiles, especialmente en áreas alejadas de los puntos de acceso principales. A pesar de que existen puntos de conexión distribuidos en diferentes pabellones y espacios abiertos de la universidad, no se cuenta con un mapeo detallado que permita visualizar la cobertura y calidad de señal en tiempo real o con datos históricos. Esta carencia limita la posibilidad de mejorar la infraestructura de red de forma estratégica, ya que no se identifican con precisión las áreas críticas o de baja intensidad.

Frente a esta situación, surge la necesidad de desarrollar una herramienta de bajo costo, portátil y confiable, que permita registrar, geolocalizar y visualizar la intensidad de señales WiFi dentro del campus universitario. Este sistema servirá como base para futuras mejoras, redistribución de puntos de acceso y diagnósticos de conectividad en diferentes facultades y zonas comunes hasta diferentes contextos geográficos del país.

**Objetivo General**

Desarrollar un sistema portátil, basado en ESP32 y módulo GPS, que permita registrar y georreferenciar la intensidad de señales WiFi dentro del campus de la Universidad Nacional de Ingeniería, con el objetivo de generar un mapa de cobertura útil para el análisis y mejora de la conectividad inalámbrica.

**Objetivos Específicos**

* Capturar datos de señal Wi-Fi y ubicación geográfica mediante un ESP32 con módulo GPS.
* Clasificar automáticamente la calidad de las señales detectadas según su nivel de intensidad.
* Procesar los datos capturados para generar un archivo CSV y un mapa interactivo.
* Visualizar los resultados en una interfaz web accesible, que incluya una tabla de redes y herramientas de búsqueda.
* Evaluar el desempeño del sistema en diferentes ubicaciones del campus universitario.

**Metodología**

Para el desarrollo del proyecto se siguió un enfoque práctico y progresivo, combinando hardware de bajo costo con herramientas de software libre. El proceso se dividió en varias etapas, cada una enfocada en lograr un componente funcional del sistema:

1. **Diseño del sistema y elección de componentes**

Se seleccionó un ESP32 como unidad central por su capacidad de conexión Wi-Fi y procesamiento, junto con un módulo GPS para obtener coordenadas geográficas, esta combinación permitió capturar tanto la intensidad de la señal como la ubicación en tiempo real.

1. **Programación del ESP32**

Se desarrolló un código en Arduino que realiza escaneos periódicos de redes Wi-Fi, captura los valores de RSSI (Indicador de Intensidad de Señal Recibida) y las coordenadas GPS, y estructura esta información en formato JSON. Los datos son enviados por el puerto serial hacia un computador.

1. **Implementación del servidor Flask**

En el equipo receptor, se configuró un servidor ligero con Flask en Python, que escucha solicitudes entrantes y guarda los datos recibidos para su posterior análisis.

1. **Procesamiento de datos**

Mediante un script en Python, los datos JSON son transformados en una tabla (archivo CSV), clasificados según la calidad de señal y visualizados en un mapa interactivo utilizando la librería folium.

1. **Diseño de la interfaz web**

Se creó una página web que integra el mapa generado y una tabla interactiva de redes detectadas. La interfaz permite buscar redes, ver su clasificación por colores y navegar entre los datos recolectados, todo desde un navegador.

1. **Pruebas en campo**

Se recorrieron diversas zonas del campus de la UNI capturando información real para probar el sistema, validar la precisión de los datos y generar el mapa final.

Esta metodología permitió construir un sistema funcional y replicable, capaz de diagnosticar la calidad de la cobertura Wi-Fi en cualquier espacio geográfico.

**Fundamento Teórico**

Para el correcto diseño del sistema fue necesario comprender diversos conceptos técnicos relacionados con la conectividad inalámbrica, los sensores de posicionamiento y el procesamiento de datos:

* **Señal Wi-Fi y RSSI**

El RSSI (Indicador de Intensidad de Señal Recibida) es un valor que indica la potencia con la que un dispositivo recibe una señal Wi-Fi. Su valor se expresa en decibelios (dBm), y mientras más cercano a cero, mejor será la señal. En este proyecto, se clasificaron los niveles de calidad según los siguientes rangos:

* **Buena**: ≥ –69 dBm
* **Intermedia**: ≥ –79 dBm
* **Mala**: < –79 dBm

Esta clasificación permite detectar zonas con buena cobertura y aquellas donde la señal es insuficiente.

* **Microcontrolador ESP32**

El ESP32 es un microcontrolador de bajo costo que incluye conectividad Wi-Fi y Bluetooth, así como varios puertos GPIO, UART, I2C, entre otros. Su capacidad de procesamiento lo convierte en una excelente opción para proyectos IoT como este.

* **Módulo GPS**

El GPS permite obtener la ubicación geográfica (latitud, longitud y altitud) con un nivel aceptable de precisión. Se usó un módulo GPS que se comunica con el ESP32 mediante el protocolo UART (serie), entregando datos en tiempo real que se sincronizan con cada lectura Wi-Fi.

* **Visualización con Python y Folium**

Para la generación del mapa se utilizó la librería folium, que permite construir mapas interactivos a partir de datos geográficos. Se aplicaron clústeres y marcadores coloreados según la calidad de señal, facilitando la interpretación visual de los resultados. Además, los datos fueron almacenados en archivos CSV para análisis adicional.

**Diseño del Sistema de Software**

Para representar la arquitectura lógica, funcional y estructural del sistema *ScanEth*, se han empleado los principales diagramas UML. Estos diagramas permiten comprender cómo interactúan los diferentes componentes del sistema, qué roles cumplen, cómo se comunican entre ellos y cómo están organizados tanto a nivel de software como de hardware. A continuación, se describen los diagramas elaborados para el presente proyecto.

1. **Diagrama de Casos de Uso**

Este diagrama representa las funcionalidades principales del sistema desde el punto de vista del usuario y de los actores externos, como el dispositivo ESP32 y el navegador web. Muestra las interacciones clave como escanear redes, obtener ubicación, enviar datos al servidor, generar mapas y visualizar resultados en la interfaz, es útil para entender qué puede hacer cada actor dentro del sistema.

Diagrama

El contenido generado por IA puede ser incorrecto.

**Ilustración 1: Diagrama de Casos de Uso**

1. **Diagrama de Clases**

El diagrama de clases modela la estructura lógica del software involucrado, incluyendo las clases del backend en Python y componentes del frontend. Se especifican atributos y métodos clave para el manejo de datos de redes Wi-Fi, lectura y escritura de archivos CSV, generación de mapas con Folium, y la interfaz visual del sistema. También se incluye una representación abstracta del firmware del ESP32.

Diagrama

El contenido generado por IA puede ser incorrecto.

**Ilustración 2: Diagrama de Clases**

1. **Diagrama de Secuencia**

Este diagrama detalla el flujo de mensajes entre los diferentes componentes del sistema durante la ejecución de una tarea completa. Inicia con el escaneo de redes y coordenadas GPS por el ESP32, seguido del envío al servidor Flask, procesamiento y generación del mapa HTML, y finaliza con la visualización por parte del usuario en el navegador. Refleja el orden cronológico de los eventos.

Tabla

El contenido generado por IA puede ser incorrecto.

**Ilustración 3: Diagrama de Secuencia**

1. **Diagrama de Componentes**

Representa la arquitectura modular del sistema, mostrando cómo se dividen los bloques lógicos y físicos: el dispositivo ESP32 con su firmware, el servidor con los scripts Python (Flask, Folium) y el frontend con HTML, CSS y JavaScript. También se indican las relaciones entre estos componentes y los artefactos que intercambian datos, como el archivo CSV o el HTML Diagrama

El contenido generado por IA puede ser incorrecto.generado.

**Ilustración 4: Diagrama de Componentes**

1. **Diagrama de Despliegue**

Diagrama

El contenido generado por IA puede ser incorrecto.Este diagrama muestra cómo se distribuyen los elementos del sistema en diferentes nodos físicos: el dispositivo IoT (ESP32), el servidor web y el cliente navegador. Cada nodo contiene los artefactos o programas que ejecuta, y se representan las conexiones de red entre ellos. Es útil para visualizar la infraestructura física del proyecto.

**Ilustración 5: Diagrama de Despliegue**

1. **Diagrama de actividades**

Muestra instancias concretas de clases utilizadas durante una ejecución real del sistema. Incluye objetos como redes Wi-Fi escaneadas, datos GPS, el archivo CSV, el servidor y el navegador web, este diagrama es útil para entender cómo se comportan los objetos en tiempo de ejecución y cómo se relacionan entre sí.

Diagrama

El contenido generado por IA puede ser incorrecto.

**Ilustración 6: Diagrama de Actividades**

1. **Diagrama de Estados**

Describe el flujo de trabajo general del sistema como una secuencia de acciones. Inicia con la orden del usuario, sigue con el escaneo, obtención de datos y envío, y concluye con la generación del mapa y la visualización en el navegador, este diagrama permite ver de forma clara las decisiones lógicas y los posibles caminos de ejecución.

Diagrama

El contenido generado por IA puede ser incorrecto.

**Ilustración 7: Diagrama de Estados**

1. **Diagrama de objetos**

Refleja los diferentes estados internos por los que pasa el dispositivo ESP32 durante su operación. Comienza en estado de espera, pasa por escaneo, lectura GPS, envío de datos, y vuelve a esperar o pasa a un estado de error, es útil para diseñar sistemas embebidos o con lógica de control basada en estados.

Diagrama

El contenido generado por IA puede ser incorrecto.

**Ilustración 8: Diagrama de Objetos**

1. **Diagrama Temporal**

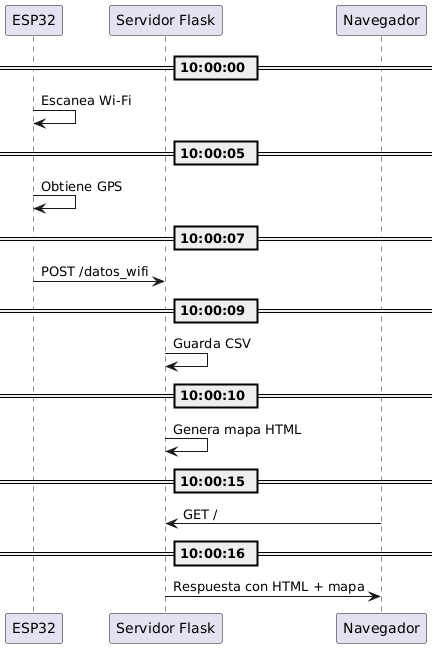
Este diagrama (una extensión del diagrama de secuencia) incluye marcas de tiempo para reflejar la evolución temporal de los eventos del sistema. Ayuda a visualizar cuánto tiempo toma cada etapa y en qué orden exacto suceden las interacciones entre el ESP32, el servidor y el navegador web.

Ilustración 9: Diagrama Temporal

**Desarrollo del Sistema del Software**

El sistema completo se compone de tres grandes partes: el dispositivo de captura (ESP32 + GPS), el servidor de recepción y procesamiento, y la interfaz web de visualización. A continuación, se detallan sus componentes principales y el funcionamiento de cada uno.

**1. Captura de datos con ESP32 y GPS (Arduino)**

El código desarrollado para el ESP32 tiene como objetivo escanear redes Wi-Fi cercanas, registrar su nivel de intensidad de señal (RSSI), identificar su SSID y MAC, y asociar esa información a una coordenada geográfica obtenida mediante un módulo GPS conectado por UART.

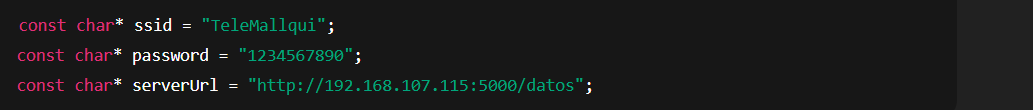
A continuación la explicación detallada del código del Arduino:

1. Forma

   El contenido generado por IA puede ser incorrecto.**Inclusión de bibliotecas**

* WiFi.h: para conectarse y escanear redes Wi-Fi.
* HTTPClient.h: para enviar datos por HTTP (POST) al servidor.
* TinyGPS++: para interpretar los datos del módulo GPS (lat, lng, satélites, etc.).
* HardwareSerial.h: permite usar un segundo puerto serie (para conectar el GPS al ESP32).

1. **Constantes y configuración inicial**



* Credenciales de la red Wi-Fi a la que se conectará el ESP32.
* URL del servidor local Flask que recibirá los datos.

1. **Objetos y variables**

Se crea un objeto GPS usando el puerto UART1 (HardwareSerial(1)), típico en ESP32.



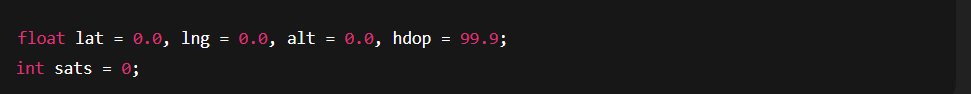
Controla cada cuánto tiempo se hace un escaneo y envío (cada 10s).



Controla el intervalo de impresión del estado del GPS.



Variables que almacenan la última posición y precisión GPS obtenidas.



1. **setup()**



Inicia la comunicación por USB (para depuración) y UART1 para GPS (TX en pin 17, RX en pin 16).



Escanea redes Wi-Fi disponibles e imprime sus nombres y niveles de señal, para prueba inicial.



Intenta conectarse a la red Wi-Fi configurada.



Espera hasta estar conectado. Luego imprime la IP asignada por el router.

1. **loop(): lectura y envío de datos**

* Leer datos del GPS



Cada carácter recibido desde el GPS se decodifica para actualizar gps.

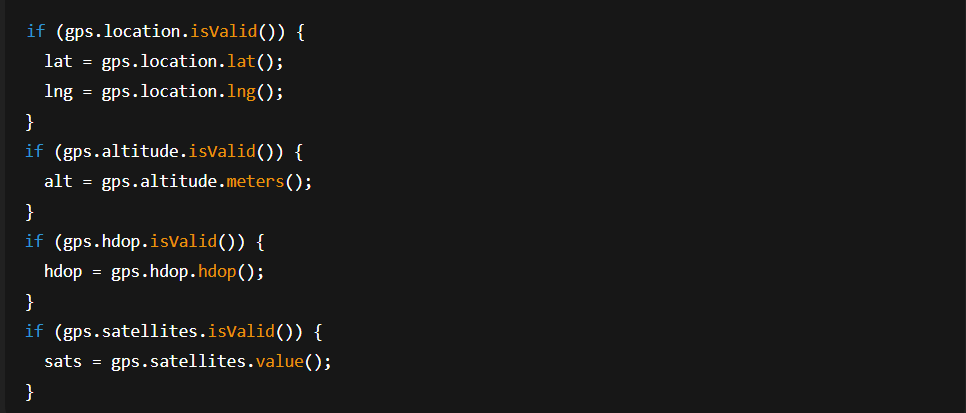
* Mostrar estado del GPS (cada 5 segundos)

Texto

El contenido generado por IA puede ser incorrecto.

Muestra por pantalla cuántos satélites ve, su precisión (HDOP) y si tiene una posición válida.

* Guardar datos GPS actuales si son válidos



Si los datos GPS están disponibles, se guardan en las variables globales.

* Esperar buena señal GPS antes de continuar

Texto

El contenido generado por IA puede ser incorrecto.

Si aún no hay buena señal (menos de 3 satélites), espera y no continúa con el escaneo.

* Verificar si pasaron 10 segundos desde el último escaneo

Texto

El contenido generado por IA puede ser incorrecto.

Control del tiempo: solo escanea cada 10 segundos para evitar saturación.

* Escaneo de redes Wi-Fi



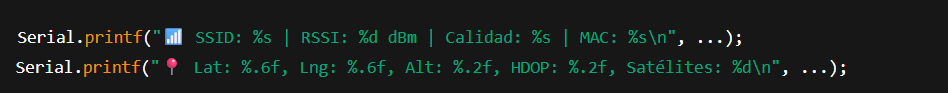
Obtiene una lista de redes visibles.

* Procesamiento de cada red detectada

Texto

El contenido generado por IA puede ser incorrecto.

Guarda el nombre de red, intensidad, dirección MAC y evalúa la calidad.



Muestra por serial la red detectada y su ubicación.



Llama a la función enviarDatos() para enviar la red detectada como JSON al servidor.

1. **Función auxiliar clasificarCalidadRSSI()**

Texto

El contenido generado por IA puede ser incorrecto.

* Asigna una etiqueta de calidad a cada red según su RSSI.

1. **Función enviarDatos() – envío HTTP POST**

Texto

El contenido generado por IA puede ser incorrecto.

* Construye un string JSON con todos los datos capturados.
* Envía ese JSON al servidor Flask mediante una solicitud POST.
* Muestra si la solicitud fue exitosa o falló.

Este JSON se transmite por el puerto serial a una computadora conectada, donde será recibido por un servidor local, la captura es continua y puede interrumpirse manualmente. El diseño permite modificar la frecuencia de escaneo o aplicar filtros por SSID.

**2. Recepción de datos vía servidor Flask (Python)**

Este código implementa un servidor web utilizando **Flask**, que recibe los datos enviados desde el **ESP32** (como JSON) y responde con un mensaje de confirmación.

1. **Importación de librerías necesarias**



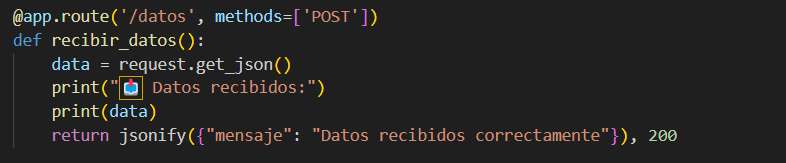
* **Flask**: un microframework para Python que facilita la creación de aplicaciones web.
* **request**: permite acceder a los datos enviados en las solicitudes HTTP (en este caso, los datos JSON).
* **jsonify**: facilita la conversión de respuestas a formato JSON.

1. **Configuración de la aplicación Flask**



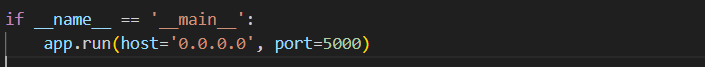
* Inicializa la aplicación Flask, que será la encargada de manejar las solicitudes HTTP.

1. **Definición de la ruta para recibir datos**



* **@app.route('/datos', methods=['POST']):** define una ruta en el servidor para recibir solicitudes HTTP POST en la URL /datos.
* **request.get\_json():** extrae los datos JSON de la solicitud POST.
* Los datos recibidos se imprimen en consola para monitorear el flujo.
* **jsonify({"mensaje": "Datos recibidos correctamente"}):** devuelve una respuesta en formato JSON confirmando que los datos fueron recibidos.
* **200:** es el código de estado HTTP que indica éxito.

1. **Iniciar el servidor Flask**



* Esto arranca el servidor Flask en la dirección IP 0.0.0.0 y en el puerto 5000, lo que significa que escuchará todas las conexiones entrantes en ese puerto.

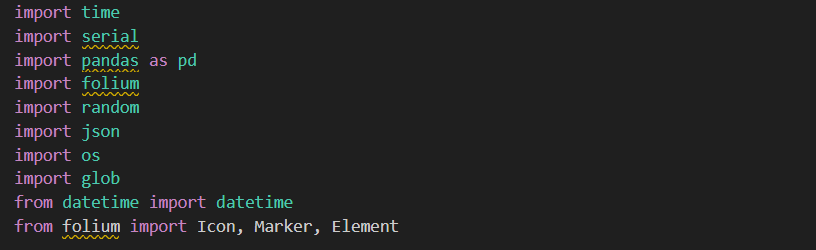
**3. Procesamiento, clasificación y generación del mapa (Python)**

El núcleo del procesamiento se encuentra en el script codigo.py, que contiene tres clases principales:

* **ESP32Reader**
* **DataSaver**
* **WiFiMapper**

El código realizado contiene:

1. **La importación de librerías**

****

* **serial:** permite la comunicación con el puerto serial (usado para la entrada de datos desde el ESP32).
* **pandas:** para manipular y almacenar datos en formato tabla (CSV).
* **folium:** para generar mapas interactivos.
* **random**: para generar datos simulados (usado en modo de prueba).
* **json, os, glob:** para manejar archivos JSON y trabajar con archivos del sistema.

1. **Configuración de parámetros**

**Texto

El contenido generado por IA puede ser incorrecto.**

* **PUERTO\_SERIAL**: especifica el puerto serial donde está conectado el ESP32 (cambia según el dispositivo).
* **SIMULACION**: permite habilitar o deshabilitar el modo simulado de datos.
* **TEMPLATE\_HTML**: define el archivo de plantilla HTML para la interfaz web.
* **HTML\_FINAL**: archivo donde se guardará el HTML final con el mapa y tabla generados.

1. **Conservación de archivos previos**

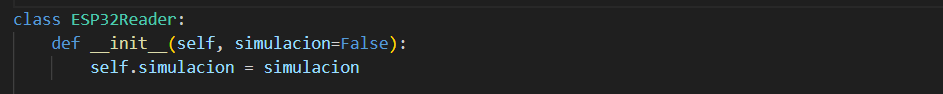
**Texto

El contenido generado por IA puede ser incorrecto.**

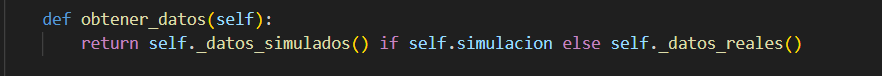
Esta función se asegura de conservar solo los últimos archivos generados, eliminando los más antiguos según el patrón de búsqueda.

1. **Clase ESP32Reader:**

La clase ESP32Reader se encarga de leer los datos de las redes Wi-Fi escaneadas desde el ESP32 y los procesa de acuerdo con la configuración. Aquí, tenemos dos métodos principales: lectura real (desde el puerto serial) y simulada (para pruebas).

****

**Método \_\_init\_\_()**: Este método es el constructor de la clase. Recibe un parámetro simulacion que indica si se deben generar datos simulados (para pruebas) o si se deben leer los datos reales del puerto serial del ESP32.

****

**Método obtener\_datos():** Este método decide qué tipo de datos se deben obtener. Si simulacion es True, obtiene datos simulados a través del método \_datos\_simulados(). Si es False, llama al método \_datos\_reales() para leer desde el ESP32.

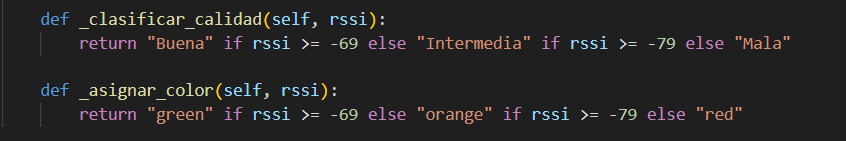
****

* **Apertura del puerto serial**: serial.Serial(PUERTO\_SERIAL, VELOCIDAD\_BAUDIOS, timeout=5) abre el puerto serial donde está conectado el ESP32. Los parámetros indican el puerto, la velocidad en baudios (115200) y un tiempo de espera de 5 segundos para leer.
* **Lectura de líneas del ESP32**: Dentro de un **bucle infinito**, lee líneas del puerto serial que contienen los datos JSON enviados por el ESP32.
* **Conversión de JSON**: Convierte cada línea en un objeto JSON y extrae el valor de **RSSI**.
* **Clasificación de la señal**: Llama a los métodos \_clasificar\_calidad() y \_asignar\_color() para determinar la calidad y el color asociado con el valor RSSI.
* **Adición de marca de tiempo**: Añade un **timestamp** con la fecha y hora de la lectura.
* **Manejo de excepciones**: Si la lectura del puerto serial falla o los datos no son válidos, el código sigue ejecutándose sin interrupciones.

**Método \_datos\_simulados():**

**Texto

El contenido generado por IA puede ser incorrecto.**

****

* Este método simula la recolección de 30 datos de red Wi-Fi aleatorios para pruebas.
* Generación aleatoria:
* RSSI: Aleatorio entre -90 y -30 dBm.
* Latitud y longitud: Aleatorios dentro de un rango específico.
* Altitud, HDOP (precisión del GPS) y número de satélites: Aleatorios dentro de rangos razonables.
* Clasificación y color: Similar al método anterior, se clasifica el RSSI y se asigna un color.
* Adición de marca de tiempo: Se genera un timestamp para cada entrada.

1. **Clase DataSaver:**

La clase DataSaver es responsable de almacenar los datos obtenidos en un archivo CSV para su análisis posterior.

Texto

El contenido generado por IA puede ser incorrecto.

**Método guardar\_csv():**

* **Comprobación de datos**: Si no hay datos para guardar, imprime un mensaje de advertencia y termina la función.
* **Uso de pandas**: Los datos se convierten en un **DataFrame** de pandas y luego se guardan en un archivo CSV utilizando to\_csv().
* **Mensaje de éxito**: Imprime la ubicación del archivo guardado.

1. **Clase WifiMapper**:

La clase WiFiMapper se encarga de generar un mapa interactivo utilizando los datos GPS y la clasificación de las señales Wi-Fi. Usa la librería folium para crear mapas basados en Leaflet.

Texto

El contenido generado por IA puede ser incorrecto.

**Comprobación de datos**: Si no hay datos, muestra un mensaje y termina la función.

**Creación de mapa**: Usa folium.Map() para crear un mapa centrado en el promedio de latitudes y longitudes de las redes detectadas.

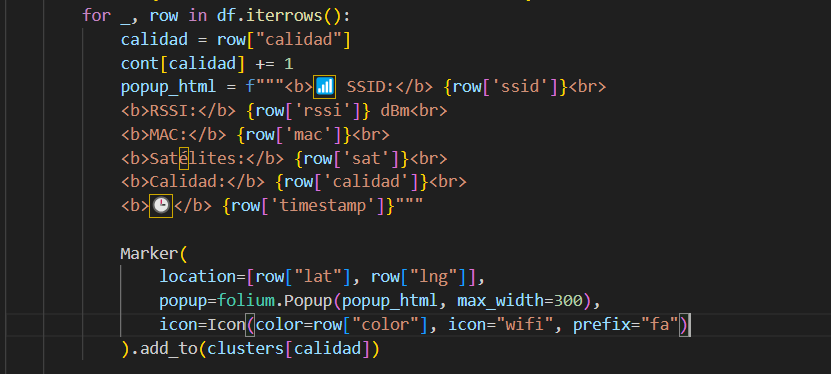
* zoom\_start=17: Nivel de zoom inicial.
* min\_zoom, max\_zoom: Límites del zoom.
* **Ajustes adicionales**: Control de escala y zoom.

**Agregar marcadores de redes Wi-Fi:**

**Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.**

* Clústeres por calidad de señal: Crea tres grupos de marcadores, uno para cada tipo de calidad de señal (buena, intermedia, mala).

****

**Añadir marcadores:** Itera sobre cada red detectada y agrega un marcador en el mapa:

* Popup: Muestra la información de la red (SSID, RSSI, etc.) al hacer clic en el marcador.
* Icono: Color y símbolo del marcador dependen de la calidad de señal.

**Añadir los clústeres al mapa**

**Texto

El contenido generado por IA puede ser incorrecto.**

**clusters** es un diccionario que contiene tres grupos de redes: **"Buena", "Intermedia" y "Mala"**. Cada grupo es un **MarkerCluster,** que agrupa las redes según su calidad de señal.

**cluster.add\_to(mapa):** Para cada grupo de redes, este método añade el clúster al mapa para que se visualice.

**Añadir control de capas (LayerControl)**

****

**LayerControl:** Es un control interactivo que permite al usuario activar o desactivar las capas del mapa (en este caso, las capas que representan las redes Wi-Fi según su calidad).

**collapsed=False:** Deja el control de capas abierto por defecto, para que el usuario vea las opciones inmediatamente.

**Crear leyenda de calidad de señal**

Texto

El contenido generado por IA puede ser incorrecto.

Se crea una leyenda fija que aparece en la esquina inferior izquierda del mapa. Esta leyenda muestra el número de redes detectadas en cada categoría de calidad de señal (buena, intermedia, mala).

* cont es un diccionario que cuenta cuántas redes hay en cada categoría, basado en la clasificación de RSSI.

Estilos CSS: La leyenda está estilizada para que sea fácilmente visible y estéticamente agradable.

**Añadir la leyenda al mapa**

****

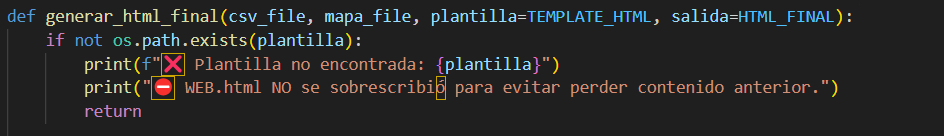
* Esta línea añade la leyenda al mapa. Utiliza el HTML para incrustar el bloque de leyenda que se creó en el paso anterior, usando **add\_child(Element(leyenda))** para agregar el HTML al contenedor del mapa.

**Guardar el mapa**

****

* Guarda el mapa como archivo HTML en la ubicación especificada.

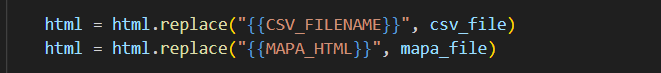
**Función generar\_html\_final():**

****

* Comienza la función que generará el archivo HTML.
* Comprobación de la plantilla: Se verifica si el archivo de plantilla HTML existe en la ubicación esperada (TEMPLATE\_HTML). Si no es así, se imprime un mensaje de error y se detiene la ejecución de la función sin sobrescribir el archivo final (WEB.html).

****

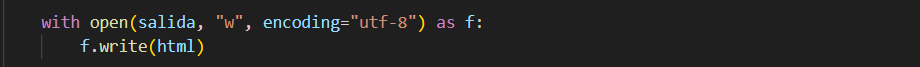
* Si la plantilla existe, se abre el archivo de plantilla HTML en modo de lectura (r) y se guarda su contenido en la variable html.
* Se utiliza **UTF-8** para asegurar que el archivo se maneje correctamente con caracteres especiales.

****

En esta parte, **se reemplazan los marcadores de posición** ({{CSV\_FILENAME}} y {{MAPA\_HTML}}) en la plantilla con las rutas de los archivos generados:

* **{{CSV\_FILENAME}}** se reemplaza con el archivo CSV que contiene los datos de las redes Wi-Fi.
* **{{MAPA\_HTML}}** se reemplaza con el archivo HTML que contiene el mapa interactivo.

Este proceso permite que la plantilla se convierta en un archivo final con los datos específicos.

****

Se abre (o crea si no existe) el archivo de salida (HTML\_FINAL), en modo de escritura (w), y se escribe el contenido del HTML modificado, con los datos reemplazados, en el archivo.

Se utiliza **UTF-8** para asegurar la compatibilidad con caracteres especiales.

****

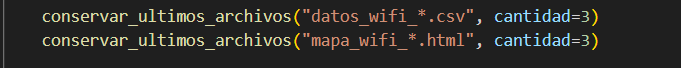
Finalmente, se imprime un mensaje indicando que el archivo HTML ha sido generado correctamente, junto con la ruta del archivo de salida.

**Bloque principal (\_\_main\_\_)**

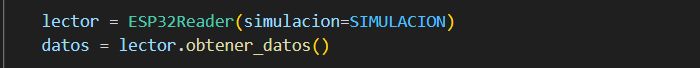


Esta condición se asegura de que el bloque de código solo se ejecute si el archivo es ejecutado directamente (y no importado como módulo).

Imprime un mensaje indicando que el escaneo de redes Wi-Fi ha comenzado.



Llama a la función **conservar\_ultimos\_archivos()**, que se encargará de conservar solo los **últimos 3 archivos CSV** y **3 archivos HTML generados**. Los archivos más antiguos se eliminarán automáticamente.

****Se crea un objeto de la clase **ESP32Reader**, que se utilizará para obtener los datos. Si **SIMULACION** es True, se generarán datos simulados; si es False, se leerán desde el puerto serial.

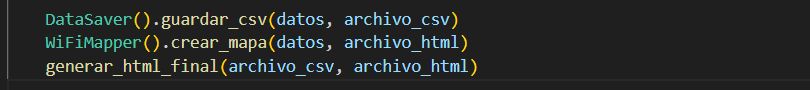
Los datos obtenidos se almacenan en la variable datos.

**Texto

El contenido generado por IA puede ser incorrecto.**

Se genera un **timestamp** (marca de tiempo) que se utiliza para nombrar los archivos CSV y HTML de manera única.

Los archivos se nombran con el formato datos\_wifi\_YYYYMMDD\_HHMMSS.csv y mapa\_wifi\_YYYYMMDD\_HHMMSS.html, usando el timestamp generado.

****

Se llama a la clase **DataSaver** para guardar los datos obtenidos en el archivo CSV.

Luego, se llama a la clase **WiFiMapper** para generar un mapa interactivo con los datos y guardarlo en el archivo HTML.

Finalmente, se llama a la función **generar\_html\_final()** para generar el archivo HTML final, que incluirá el mapa y los datos CSV incrustados.

**Visualización de Resultados**

Una vez que se han recopilado los datos sobre las redes Wi-Fi (como SSID, RSSI, ubicación GPS y otros parámetros), el sistema genera dos elementos principales para la visualización:

1. **Mapa interactivo**: Creado con **folium** y centrado en la ubicación de las redes Wi-Fi detectadas.
2. **Tabla dinámica**: Generada a partir del archivo CSV con los datos de las redes y visualizada en la interfaz web.

**Mapa Interactivo (con folium)**

El mapa interactivo muestra la ubicación geográfica de las redes Wi-Fi detectadas, utilizando la latitud y longitud obtenidas del GPS. Los puntos en el mapa se agrupan en **clústeres** según la calidad de la señal (buena, intermedia, mala), lo que permite ver rápidamente la cobertura de red en diferentes áreas.

**Características del mapa:**

* **Marcadores agrupados**: Usando **MarkerCluster** para agrupar redes de calidad similar.
* **Colores según calidad**: Los puntos se colorean según la calidad de la señal:
  + **Buena**: Verde
  + **Intermedia**: Naranja
  + **Mala**: Roja
* **Información adicional**: Al hacer clic en un marcador, aparece un **popup** con detalles sobre la red (SSID, RSSI, MAC, calidad de señal, ubicación GPS, etc.).
* **Leyenda de calidad**: En la esquina inferior izquierda, una leyenda muestra la cantidad de redes detectadas en cada categoría de calidad, para dar un resumen visual.

**Tabla de Datos (con búsqueda dinámica)**

La tabla de datos muestra la información detallada de cada red Wi-Fi detectada. La tabla es interactiva, lo que permite al usuario buscar redes específicas, filtrar por nombre o características, y ver la información asociada a cada red.

**Características de la tabla**:

* **Búsqueda en tiempo real**: Los usuarios pueden buscar redes por nombre (SSID) o dirección MAC.
* **Clasificación por calidad**: Cada red es clasificada y visualizada con su color correspondiente (verde, naranja, rojo) según la calidad de la señal.
* **Columnas de información**: Incluye SSID, RSSI, MAC, latitud, longitud, calidad de señal, y otros parámetros como altitud, HDOP y número de satélites GPS.

**Escalabilidad**

El diseño modular del sistema permite su escalabilidad tanto en términos de ubicación geográfica como de funcionalidad. Aunque este proyecto se ha centrado en la cobertura Wi-Fi del campus de la Universidad Nacional de Ingeniería, su arquitectura permite adaptarlo a otros entornos, tales como:

1. **Ampliación a otras universidades o instituciones**:
   * Dado que el sistema no depende de infraestructura de red específica y se utiliza hardware de bajo costo (ESP32 + GPS), puede replicarse fácilmente en otras universidades para realizar un diagnóstico de cobertura en tiempo real.
   * Las plantillas y los archivos generados (HTML, CSV) son portátiles, lo que facilita la replicación sin necesidad de configuraciones adicionales.
2. **Adaptación a zonas rurales**:
   * El sistema también tiene un alto potencial para adaptarse a zonas rurales donde la conectividad Wi-Fi es limitada o inexistente. Con solo un ESP32 y un módulo GPS, el dispositivo es lo suficientemente portátil como para realizar mediciones en diferentes ubicaciones geográficas.
   * Se puede modificar el sistema para integrar diferentes tipos de redes (por ejemplo, redes móviles 4G/5G) o adaptarlo a nuevas tecnologías de conectividad, ampliando su capacidad para realizar análisis de cobertura.
3. **Mejoras futuras**:
   * El sistema también puede expandirse para incorporar otras métricas de red, como velocidad de conexión o latencia, brindando así una evaluación más completa de la calidad de red.
   * Con la integración de sistemas de monitoreo remoto o el uso de drones (en zonas de difícil acceso), el sistema podría escalar a proyectos más grandes o zonas geográficas más amplias, sin perder su simplicidad y accesibilidad.

**Sostenibilidad**

La sostenibilidad de este proyecto se basa en varios factores que permiten su funcionamiento eficiente a largo plazo, tanto desde el punto de vista económico como ecológico:

1. **Bajo costo y bajo consumo energético**:
   * El uso del ESP32, un microcontrolador económico y de bajo consumo, junto con el módulo GPS, hace que el sistema sea asequible tanto en términos de inversión inicial como de mantenimiento.
   * El consumo energético es muy bajo, lo que hace que el sistema sea sostenible desde el punto de vista ecológico, ya que no requiere fuentes de energía costosas o contaminantes. Incluso, el sistema puede alimentarse con baterías o paneles solares si se requiere autonomía en zonas remotas.
2. **Uso de software libre**:
   * El sistema se basa en tecnologías de código abierto (como Flask, Folium, Python, HTML) que eliminan costos de licencias y permiten a cualquier usuario o institución modificar y adaptar el sistema sin restricciones.
   * Esta apertura fomenta la colaboración continua entre universidades, instituciones y comunidades, lo que aumenta la sostenibilidad a largo plazo.
3. **Mantenimiento sencillo y actualización**:
   * El sistema es fácil de mantener debido a su bajo nivel de complejidad y la utilización de herramientas ampliamente documentadas y soportadas.
   * Además, al estar basado en componentes modulares, se pueden realizar actualizaciones rápidas sin interrumpir su funcionamiento general. Si es necesario, se pueden incorporar nuevas funcionalidades sin generar costos adicionales importantes.
4. **Beneficios sociales**:
   * La inclusión digital que facilita la mejora de la conectividad Wi-Fi en zonas rurales o áreas con cobertura deficiente contribuye directamente al desarrollo social.
   * Este tipo de proyectos también puede formar parte de iniciativas de educación tecnológica, ya que estudiantes y profesionales pueden aprender y contribuir al desarrollo del sistema.

**4. Visualización interactiva en la web**

La visualización se realiza a través de una página HTML (WEB.html) generada dinámicamente con base en una plantilla (WEB\_template.html). La interfaz incluye:

* Un iframe con el mapa interactivo.
* Una tabla dinámica cargada desde el archivo CSV.
* Clasificación visual por colores (verde, naranja, rojo).
* Búsqueda en tiempo real por nombre de red, dirección MAC, etc.
* Modo oscuro activable por el usuario.

El mapa permite hacer zoom, desplazarse por las zonas del campus y ver los puntos de acceso mapeados, mientras que la tabla complementa con datos detallados.

**Conclusiones**  
El sistema desarrollado proporciona una solución accesible, económica y efectiva para mapear la cobertura Wi-Fi en el campus de la Universidad Nacional de Ingeniería. A través de la integración de hardware de bajo costo (ESP32 + GPS) y software libre, es posible obtener un análisis detallado de la calidad de señal Wi-Fi, permitiendo identificar áreas con cobertura insuficiente y tomar decisiones informadas para mejorar la infraestructura.

Gracias a su diseño modular y portátil, este sistema puede escalarse y replicarse fácilmente en otras universidades, instituciones o incluso en zonas rurales. El hecho de que no dependa de infraestructura de red especializada, y que use componentes de bajo costo, lo convierte en una opción viable para muchos entornos con diferentes niveles de conectividad.

La interfaz web generada (que incluye el mapa interactivo y la tabla de datos) ofrece una visualización clara y detallada de las redes Wi-Fi, con información accesible sobre su calidad de señal, ubicación y otros parámetros relevantes. La adición de la leyenda de calidad y el control de capas hace que el análisis de los resultados sea intuitivo y fácil de interpretar.

Además de ser útil para la gestión de redes dentro de la universidad, este sistema tiene el potencial de contribuir al desarrollo de zonas rurales al mejorar el acceso a la conectividad en lugares donde no existen herramientas similares para medir la cobertura. Su bajo costo y flexibilidad también lo convierten en una herramienta de aprendizaje para estudiantes y profesionales del área de telecomunicaciones y redes.

**Recomendaciones**

1. **Ampliación del sistema a otras métricas**:

Sería útil ampliar el sistema para incluir métricas adicionales de red, como la velocidad de conexión y la latencia, que proporcionarían una visión más completa de la calidad de las redes Wi-Fi. Esta ampliación podría integrar tecnologías de medición más avanzadas y equipos como medidores de velocidad.

1. **Automatización y monitoreo continuo**:

El sistema podría beneficiarse de una automación más avanzada, donde los datos se envíen en tiempo real a una base de datos centralizada para realizar un análisis continuo. Esto permitiría tener un monitoreo permanente de la calidad de la red y podría ser útil para la planificación a largo plazo de la infraestructura de red.

1. **Integración con otras tecnologías de redes**:

Aunque este sistema está centrado en redes Wi-Fi, su diseño modular permite su expansión a otros tipos de redes, como 4G, 5G, o incluso redes privadas de telecomunicaciones. Se podría integrar con análisis de cobertura de red celular, lo que sería especialmente valioso en entornos rurales.

1. **Expandir el uso a comunidades fuera de la universidad**:

Como parte de su escalabilidad, el sistema podría extenderse a comunidades rurales o zonas suburbanas donde la cobertura Wi-Fi es limitada. Además de su uso académico, el sistema podría formar parte de proyectos de inclusión digital que fomenten la educación tecnológica y el acceso a la conectividad en áreas más desfavorecidas.

1. **Desarrollo de una versión móvil**:

Considerando el uso creciente de dispositivos móviles en la recopilación de datos, sería recomendable desarrollar una versión del sistema para smartphones o tablets, aprovechando sus capacidades de conectividad y GPS. Esto permitiría realizar diagnósticos más ágiles y acceder al sistema desde cualquier lugar.

**BIBLIOGRAFÍA:**

*Señal Osiptel*. (s. f.). <https://serviciosweb.osiptel.gob.pe/CoberturaMovil/>

Nilton, L. V. J. (2016b, agosto 2). *Diseño de una red de sensores de espectro para la selección de canales de operación óptimos en la red wifi del campus PUCP*. <https://tesis.pucp.edu.pe/items/bf9249c8-5af7-4a06-86a9-c9a3f5e73672>

Easha, F. B. K., Abbas, R., & Daley, M. (2020, 3 abril). *Campus Wi-Fi Coverage Mapping and Analysis*. arXiv.org. <https://arxiv.org/abs/2004.01561>

Hernandez, M. D. F. (2017). *Mapeo de la señal WiFi mediante un dispositivo implementado en Arduino*. <http://repositorio.ugto.mx/handle/20.500.12059/3294>