PROYECTO

DE

INGENIERÍA AUTOMÁTICA

**Título: ESP32 leer datos desde un módulo GPS y transmitir mediante Bluetooth**

**Estudiantes:**

**Rene Antonio Jiménez Delgado**

**Alejandro Martin Rodríguez**

**Gabriel Enrique Gutiérrez Fernández**

**Introducción**

**Luego de una búsqueda de diferentes sitios, canales, documentos, blogs y artículos como son los mostrados a continuación**

[**https://naylampmechatronics.com/blog/18\_tutorial-modulo-gps-con-arduino.html**](https://naylampmechatronics.com/blog/18_tutorial-modulo-gps-con-arduino.html)

**El módulo GPS**[**GY-GPS6MV2**](https://naylampmechatronics.com/sensores-posicion-inerciales-gps/106-modulo-gps-neo-6m.html)**está basado en el SoM**[**U-Blox NEO-6M**](https://www.u-blox.com/en/product/neo-6-series?utm_source=en/gps-modules/pvt-modules/previous-generations/neo-6-family.html)**equipado en el PCB, una EEPROM con configuración de fábrica, una pila de botón para mantener los datos de configuración en la memoria EEPROM, un indicador LED y una antena cerámica. También posee los pines o conectores VCC, RX, TX y GND por el que se puede conectar a algún microcontrolador mediante una interfaz serial. Para que nuestro módulo GPS funcione a la perfección se recomienda hacer las pruebas en un ambiente abierto o cercano a una ventana para una correcta recepción de la señal de los satélites. Algunos problemas que se presentaron después de revisar el fórum de comentarios fueron:**

**Problemas de Recepción GPS: un usuario reportó que su módulo GPS no mostraba coordenadas y solo recibía caracteres asterisco. Esto indica que el módulo no estaba fijando una posición, posiblemente debido a la falta de suficiente señal satelital. Se sugirió que el usuario se moviera a un área más abierta para mejorar la recepción.**

**Errores en la Decodificación de Coordenadas: aunque el código se ejecutaba correctamente, las coordenadas obtenidas presentaban un error de 7 km respecto a la ubicación real. Esto puede ser resultado de problemas en la decodificación de los datos del GPS o interferencias en la señal.**

**Problemas de Alimentación: Un usuario en otro foro discutió problemas de alimentación al intentar usar múltiples módulos (GPS, OLED, SIM900) con un solo Arduino. La falta de energía adecuada causó que los módulos dejaran de funcionar correctamente, lo que podría haber afectado la recepción y procesamiento de datos GPS**

**Geolocalización WiFi con Arduino NodeMCU Firebase y Google Maps**

[**https://programarfacil.com/blog/arduino-blog/geolocalizacion-wifi-arduino-nodemcu/**](https://programarfacil.com/blog/arduino-blog/geolocalizacion-wifi-arduino-nodemcu/)

**Este artículo nos explica como sacar provecho a varios servicios de Google para poder utilizar la geolocalización WiFi con Arduino MKR1000 y NodeMCU, claro es un articulo de mucha ayuda pero también presento ciertos defectos como son:**

**Dependencia de la Conexión a Internet: El proyecto se basa completamente en la conectividad WiFi para determinar la ubicación, lo que significa que no funcionará en áreas sin acceso a Internet. Esto limita su aplicabilidad en entornos rurales o en situaciones donde la conectividad es intermitente.**

**Precisión Variable: La precisión de la geolocalización WiFi se menciona como un rango de 10 a 100 metros, dependiendo de varios factores como la cantidad de routers disponibles y la calidad de la señal.**

**Desarrollo de un dispositivo para localización y obtención de parámetros de vehículos mediante una aplicación móvil mediante la puesta en marcha de una red IoT (internet de las cosas por sus siglas en inglés) con el propósito de localizar un dispositivo que proporciona su ubicación exacta mediante GPS incorporado**

[**https://dspace.ups.edu.ec/bitstream/123456789/24370/1/TTS1227.pdf**](https://dspace.ups.edu.ec/bitstream/123456789/24370/1/TTS1227.pdf)

**Este proyecto solicita Requisitos Técnicos Altos como el uso de tecnologías como MQTT y servidores en la nube puede requerir conocimientos técnicos avanzados que no todos los usuarios poseen.**

Objetivo general en si es: ESP32, leer datos desde un módulo GPS y transmitir esta información mediante Bluetooth y pantalla integrada mientras gestiona eficientemente su consumo energético**.**

**Objetivos específicos:**

**Plantear soluciones a las problemáticas anteriores como:**

* **Utilizar un módulo GPS, que puede funcionar independientemente de la conexión a Internet.**
* **Ofrecer una mayor precisión en la localización**
* **Utilizar bibliotecas más directas como TinyGPS++, que puede ser más accesible y fácil de implementar.**
* **Imprimir los datos por pantalla**

**1 Marco Teórico Relevante:**

**El ESP32 es un microcontrolador de bajo costo, alto rendimiento y bajo consumo de energía diseñado por Espressif Systems. Es muy popular en proyectos de Internet de las Cosas (IoT), electrónica y desarrollo embebido gracias a su versatilidad y potentes características.**

**Características principales:**

**CPU y arquitectura: Basado en un microprocesador Xtensa LX6 de 32 bits. Incluye dos núcleos (dual-core) o uno (single-core, en variantes específicas). Velocidad de reloj: hasta 240 MHz. Memoria: Memoria RAM interna de 520 KB. Soporte para almacenamiento externo en Flash SPI y PSRAM. Conectividad inalámbrica: Wi-Fi (802.11 b/g/n): Hasta 150 Mbps. Bluetooth 4.2 y Bluetooth Low Energy (BLE). Entradas y salidas: Hasta 34 pines GPIO configurables. Soporte para ADC (12 bits), DAC, PWM, SPI, I2C, I2S, UART y más. Pines de detección táctil capacitivos. Sensores internos: Sensor de temperatura. Detector de sensibilidad táctil. Consumo de energía: Varias configuraciones de bajo consumo de energía. Modos como Deep Sleep y Light Sleep. Aplicaciones comunes: IoT (Internet de las Cosas):Monitoreo remoto y control de dispositivos. Integración con servicios como AWS IoT, Google Cloud IoT y MQTT. Automatización del hogar: Sistemas de iluminación inteligente. Control de electrodomésticos. Dispositivos portátiles: Monitores de salud. Rastreo GPS (en combinación con módulos externos). Electrónica educativa: Excelente para proyectos académicos y maker. Redes de sensores inalámbricos.**

**Arduino** es una plataforma de hardware y software de código abierto que facilita la creación de proyectos de electrónica y automatización. Es muy popular en la comunidad de "makers" y en la educación debido a su accesibilidad y facilidad de uso.

**El Hardware de Arduino (UNO) posee un microcontrolador como el** ATmega328, pines de Entrada/Salida tanto analógicos como digitales para conectar sensores, actuadores y otros componentes electrónicos, puertos de comunicación y alimentación de 9-12v.

En cuanto al lenguaje de programación utiliza un lenguaje de programación basado en Wiring (similar a C/C++)

**2 Descripción detallada del diseño hardware y software:**

**2.1 Hardware:**

**T-Beam como núcleo principal, con conectividad Wi-Fi y Bluetooth. Utiliza un transceptor LoRa basado en el chip SX1276 (compatible con redes LoRaWAN)**

**Incluye una pantalla OLED para mostrar información en tiempo real**

**Se programa usando Arduino IDE, PlatformIO, o ESP-IDF**

**2.2 Software:**

1. **Inclusión de bibliotecas**



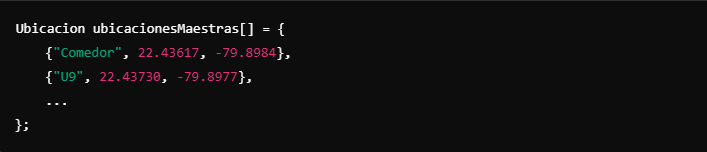
**TinyGPS++: Procesar datos de GPS (latitud, longitud, altitud, etc.). Axp20x: Manejar el gestor de energía AXP192. Esp\_sleep: Gestionar el modo de bajo consumo. BluetoothSerial: Implementar comunicación Bluetooth Serial. SPIFFS: Gestionar el almacenamiento en el sistema de archivos Flash. Wire: Proporcionar comunicación I2C (necesaria para OLED y AXP192). U8g2lib: Controlar la pantalla OLED (modelo SH1106)**

1. **Inicialización de periféricos y variables**

****

u8g2: Instancia para manejar la pantalla OLED con el controlador SH1106 usando I2C. gps: Instancia para procesar datos de GPS. GPS: Puerto serial para comunicación con el módulo GPS (puerto serial hardware. axp: Instancia para interactuar con el gestor de energía. SerialBT: Objeto para manejar la comunicación Bluetooth Serial. Ubicacion: Estructura que almacena información sobre ubicaciones maestras (nombre, latitud y longitud).

1. Definición de ubicaciones maestras



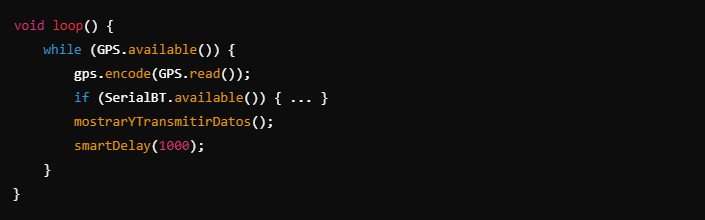
Se define un array con las ubicaciones maestras, incluyendo sus nombres y coordenadas GPS. Esto se utiliza para verificar la cercanía a estas ubicaciones.

1. Configuración inicial (setup)



Serial y Bluetooth: Inicializan comunicación serie y Bluetooth. OLED y SPIFFS: Configuran la pantalla OLED y el sistema de archivos. AXP192: Inicializan el gestor de energía y activan las salidas necesarias para GPS, pantalla y demás periféricos. GPS: Configuran el módulo GPS en un puerto serie hardware con velocidad de 9600 baudios.

1. Bucle principal (loop)



**GPS**: Decodifica datos disponibles en el puerto GPS.

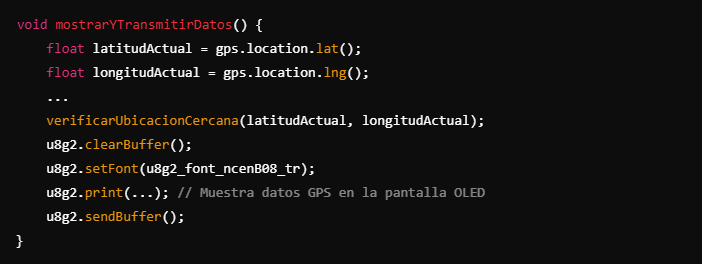
**Bluetooth**: Procesa comandos enviados desde Bluetooth:

* SAVE: Activa el guardado de datos GPS.
* STOP: Detiene el guardado.
* READ: Transmite el archivo almacenado en SPIFFS por Bluetooth.

**mostrarYTransmitirDatos()**: Muestra y envía los datos de GPS.

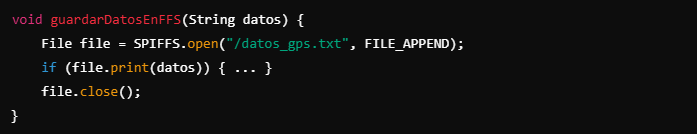
**smartDelay()**: Realiza un retardo mientras sigue leyendo datos GPS.

1. Mostrar y transmitir datos (mostrarYTransmitirDatos)



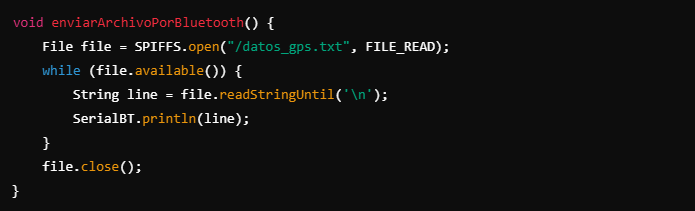
Obtiene latitud, longitud y otros datos GPS. Los muestra en el monitor serie y los envía por Bluetooth. Actualiza la pantalla OLED con la información. Llama a verificarUbicacionCercana para comprobar proximidad a ubicaciones maestras.

7. Guardar datos en SPIFFS



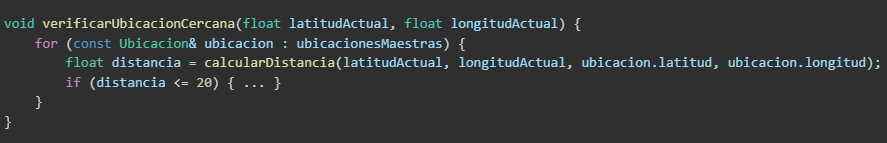
Guarda los datos en un archivo de texto dentro del sistema SPIFFS.

**8. Enviar archivo por Bluetooth**

****

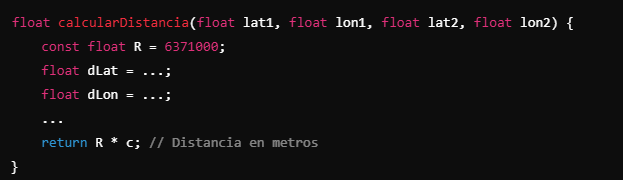
**Lee el archivo almacenado en SPIFFS línea por línea y lo transmite por Bluetooth.**

**9. Verificar proximidad a ubicaciones maestras**

****

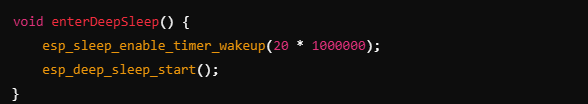
Compara las coordenadas actuales con las ubicaciones maestras usando la función calcularDistancia. Si está dentro de un rango de 20 metros, envía una notificación por Bluetooth y actualiza la pantalla OLED.

10. Cálculo de distancias



Usa la fórmula del **Haversine** para calcular la distancia entre dos puntos geográficos en la Tierra.

11. Modo de sueño profundo



Coloca al dispositivo en modo de bajo consumo durante 20 segundos para ahorrar batería.

**3- Resultados y discusión:**

3.1 Resultados: luego de diversas pruebas para evaluar el desempeño se llegó a que los datos del GPS son procesados y presentados en tiempo real en la pantalla OLED, transmitidos por Bluetooth y almacenados en SPIFFS para análisis posterior. La proximidad a ubicaciones maestras permite enviar alertas específicas al usuario. El modo de bajo consumo garantiza que el dispositivo pueda operar durante largos períodos en escenarios donde la batería es limitada.

**1. Procesamiento de datos GPS (ubicación, altitud, velocidad):**

El código utiliza la biblioteca **TinyGPS++** para decodificar datos recibidos del módulo GPS conectado al ESP32.

Los datos relevantes incluyen:

**Latitud y longitud:** Obtenidos usando gps.location.lat() y gps.location.lng().

**Altitud:** Convertida de pies a metros con gps.altitude.feet() / 3.2808.

**Velocidad:** Calculada en km/h con gps.speed.kmph().

**Número de satélites:** Recuperado con gps.satellites.value().

**Hora:** Extraída mediante gps.time.hour(), gps.time.minute() y gps.time.second().

Estos datos son procesados en tiempo real dentro de la función mostrarYTransmitirDatos(), donde:

Se crean cadenas de texto para su transmisión por Bluetooth.

Se imprimen en el monitor serie.

Se visualizan en la pantalla OLED para proporcionar una interfaz visual al usuario.

**2. Almacenamiento en SPIFFS y transmisión por Bluetooth:**

* El código usa el sistema de archivos SPIFFS para guardar los datos GPS en un archivo llamado datos\_gps.txt.
* Cuando el comando "SAVE" se recibe por Bluetooth, los datos se guardan en el archivo utilizando la función guardarDatosEnFFS(String datos).
* Si el comando "STOP" es recibido, se detiene el guardado.

**Transmisión de datos por Bluetooth:**

* Los datos GPS se transmiten de forma continua por Bluetooth si hay un dispositivo conectado.
* Al recibir el comando "READ", el archivo completo de datos (datos\_gps.txt)se envía línea por línea al cliente Bluetooth. Esto se maneja en la función enviarArchivoPorBluetooth().

**3. Verificación de proximidad a ubicaciones maestras:**

* El código usa la función calcularDistancia() para determinar la distancia entre la ubicación GPS actual y cada ubicación maestra.
* Si la distancia es menor o igual a **20 metros**, se considera que el dispositivo está cerca de esa ubicación.
* Una notificación se envía por Bluetooth con el nombre de la ubicación cercana.

**Lógica para evitar redundancia:**

* La variable mensajeUbicacionActual evita enviar múltiples notificaciones de la misma ubicación.
* Si el dispositivo se aleja, el mensaje se limpia para notificar futuros eventos.

**4. Visualización en pantalla OLED:**

* La pantalla OLED (modelo SH1106) es inicializada usando la biblioteca **U8g2lib**.
* En cada ciclo, se muestran los siguientes datos:
* **Latitud y longitud:** Formateados con 5 decimales.
* **Mensaje de ubicación cercana:** Si aplica.
* La visualización es manejada dentro de la función mostrarYTransmitirDatos():
* El buffer de la pantalla se limpia con u8g2.clearBuffer().
* Se escriben los datos en el buffer con u8g2.print().
* Finalmente, los datos se envían al display con u8g2.sendBuffer().

**5. Modo de bajo consumo para optimizar batería:**

* La función enterDeepSleep() implementa un modo de bajo consumo usando el módulo **esp\_sleep**:
* Configura un temporizador para que el ESP32 despierte después de **20 segundos** con esp\_sleep\_enable\_timer\_wakeup(20 \* 1000000).
* El dispositivo entra en el modo de suspensión profunda con esp\_deep\_sleep\_start().

**Ventajas del modo de bajo consumo:**

* Reduce el consumo de energía apagando los periféricos innecesarios.
* Ideal para dispositivos alimentados por batería que necesitan recolectar datos de forma periódica.

3.2 Discusión:

Evaluación en la utilidad:

El proyecto desarrollado presenta una amplia lista de posibles aplicaciones como son, seguimiento en tiempo real, mostrando datos de posición, velocidad y altitud en la pantalla OLED y transmitiéndolos al usuario por Bluetooth. Notificaciones basadas en proximidad ideal para sistemas de navegación en campus, como universidades, complejos industriales o áreas turísticas, donde los usuarios pueden ser alertados cuando están cerca de puntos de interés. Registro de datos para análisis posterior. Los datos GPS se almacenan en SPIFFS, lo que permite analizarlos posteriormente para evaluar rutas, patrones de movimiento o cualquier otra información útil.

Identificación de Problemas o Limitaciones Encontradas Durante el Proceso:

Algunos de estos problemas pueden ser: p**roblemas de precisión del GPS, interferencias en entornos urbanos o interiores, la señal GPS puede ser débil o inexacta, lo que afectaría la capacidad para detectar correctamente las ubicaciones maestras. Rango de error, el GPS tiene un margen de error que podría ser mayor a los 20 metros establecidos para determinar proximidad a una ubicación maestra. Rendimiento de Bluetooth conexión intermitente, si el dispositivo Bluetooth pierde la conexión con el cliente (por ejemplo, un teléfono), la transmisión de datos se verá interrumpida. Gestión del almacenamiento en SPIFFS, capacidad limitada SPIFFS tiene un espacio reducido en la mayoría de los ESP32. Si los datos GPS se registran continuamente, el archivo datos\_gps.txt podría llenarse rápidamente.**

**Comparacion**

**En comparación con los otros proyectos este facilita ciertas cosas que los otros no, como son bajo consumo de la alimentación, no es necesario conectividad wifi para poder determinar ubicación y sobre todo precisión a la hora de enviar las coordenadas esto puede hacerlo nuestro proyecto y los otros no.**

**Conclusiones:**

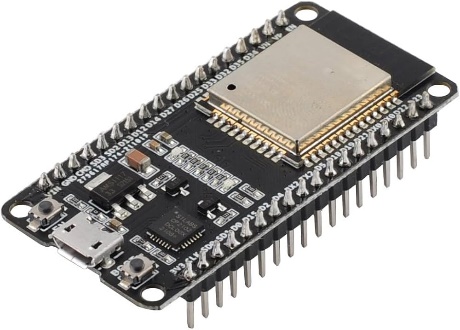
**El desarrollo del sistema basado en el ESP32 ha demostrado ser una solución eficiente para el rastreo y monitoreo en tiempo real, combinando tecnologías como GPS, Bluetooth, SPIFFS y una pantalla OLED. Este proyecto se presenta como una herramienta versátil, capaz de registrar, almacenar y transmitir datos de ubicación con precisión, al tiempo que permite al usuario interactuar de manera intuitiva con el dispositivo. El proyecto demuestra la capacidad de los microcontroladores ESP32 para realizar tareas avanzadas en tiempo real con múltiples tecnologías. Con ajustes, puede ser una herramienta útil tanto en proyectos personales como en aplicaciones industriales.**

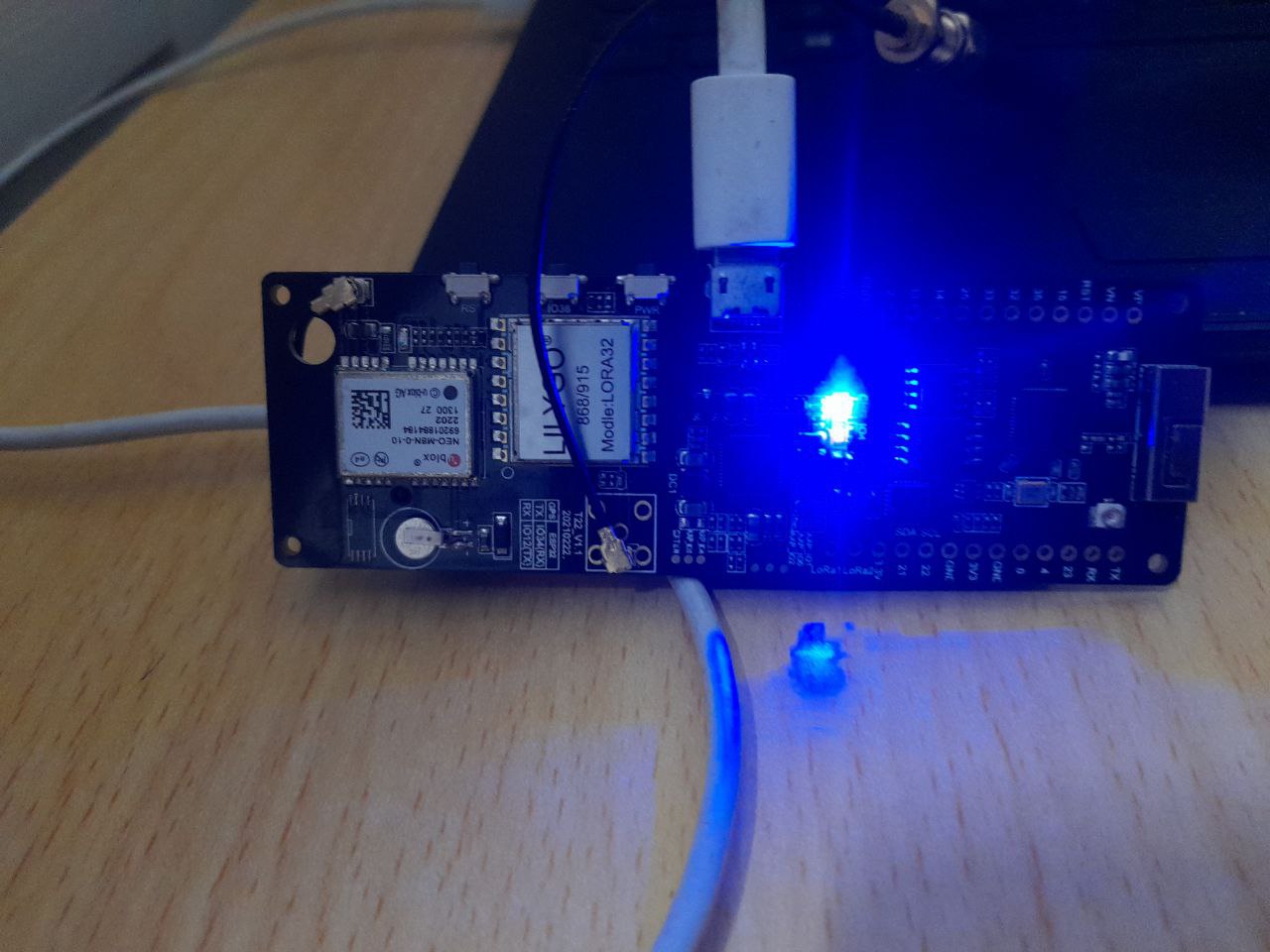
**Recomendaciones:**

**Buscar futuras soluciones para si el dispositivo es expuesto a mucha humedad o diversos problemas externos ambientales**

**Anexos**

**ESP32**





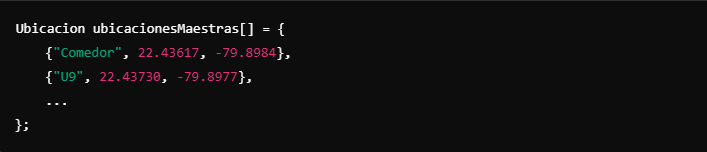




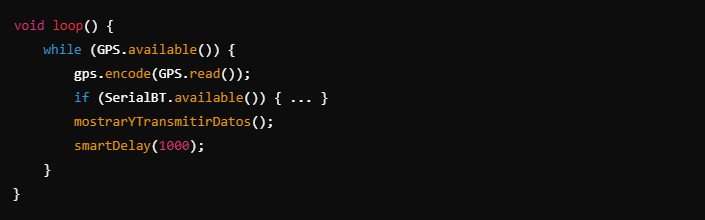
Código

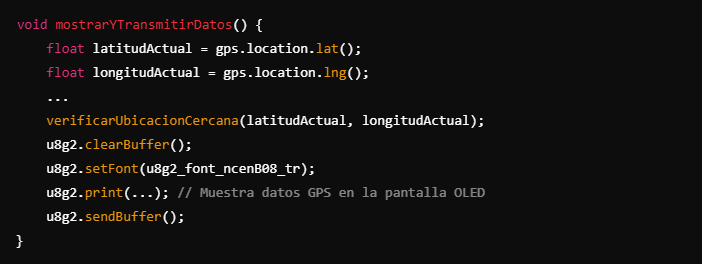


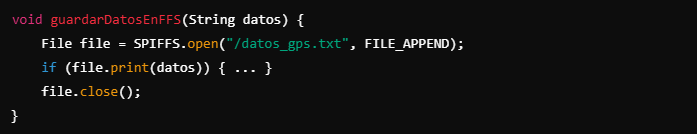
****

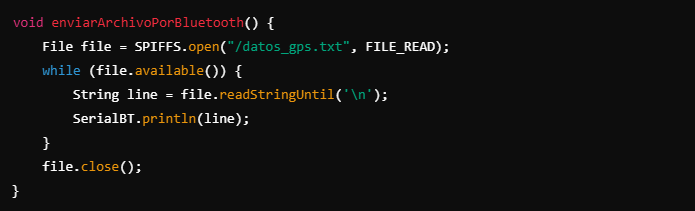


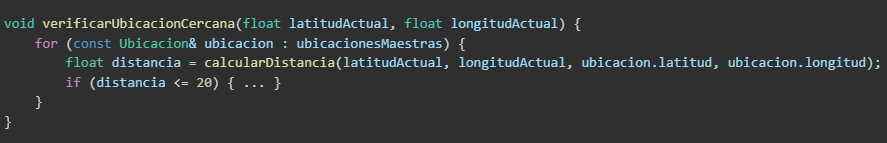








****

****

