📡 Proyecto 1: Lectura de Temperatura y Humedad con ESP32 y AHT10 con BLE

📋 Descripción

Este proyecto utiliza un ESP32 para leer datos del sensor AHT10 (temperatura y humedad) mediante I2C y los envía a una aplicación por Bluetooth BLE utilizando notificaciones. La aplicación cliente puede ser nRF Connect (Android/iOS) u otra compatible con BLE.

🧰 Componentes necesarios

Componente	Cantidad
ESP32 DevKit v1	1
Sensor AHT10	1
Cables Dupont	4
Fuente de alimentación (USB)	1

🔌 Conexiones físicas

AHT10	ESP32
VIN o VCC	3.3V
GND	GND
SDA	GPIO 21
SCL	GPIO 22

🧠 Funcionalidad del código

Inicializa el sensor AHT10 por I2C.

Inicializa el módulo Bluetooth BLE.

Crea un servicio y una característica con UUID personalizados.

Si un cliente BLE se conecta:

Cada 2 segundos lee los datos de temperatura y humedad.

Formatea los datos como:
Temp:26.5;Hum:45.0

Los envía por notificación BLE.

Muestra los datos también por el monitor serial.

🧾 Código fuente

1. Inclusión de librerías

```
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_AHTX0.h>

#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
```

- **Arduino.h**: Librería base del entorno Arduino.
- **Wire.h**: Para la comunicación I2C.
- **Adafruit_AHTX0.h**: Librería para manejar el sensor AHT10.
- **BLEDevice**, **BLEServer, BLEUtils, BLE2902**: Librerías para manejar el Bluetooth Low Energy (BLE) en ESP32.

```
Adafruit_AHTX0 aht;

#define SERVICE_UUID           "12345678-1234-1234-1234-1234567890ab"
#define CHARACTERISTIC_UUID    "abcd1234-5678-90ab-cdef-1234567890ab"

BLECharacteristic *pCharacteristic;
bool deviceConnected = false;
```

3. Clase para manejar eventos BLE

```
class MyServerCallbacks : public BLEServerCallbacks {
  void onConnect(BLEServer* pServer) {
    deviceConnected = true;
  }

  void onDisconnect(BLEServer* pServer) {
    deviceConnected = false;
  }
};
```

- Esta clase gestiona **eventos BLE** como conexión y desconexión.
- Cambia el valor de `deviceConnected` según si hay un cliente conectado o no.

4. Configuración inicial en setup()

```
void setup() {
  Serial.begin(115200);
  while (!Serial);
```

Inicializa la **comunicación por puerto serie** para depuración.

```
 if (!aht.begin()) {
   Serial.println("No se encontró el sensor AHT10, revisa conexión I2C");
   while (1) delay(10);
 }
```

- Intenta iniciar el sensor AHT10.
- Si falla, imprime error y queda en bucle infinito.

```
 BLEDevice::init("ESP32-AHT10");
  BLEServer *pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());
```

- Inicializa el BLE con el nombre "ESP32-AHT10".
- Crea el servidor BLE y le asigna los *callbacks*.

```
 BLEService *pService = pServer->createService(SERVICE_UUID);
  pCharacteristic = pService->createCharacteristic(
                      CHARACTERISTIC_UUID,
                      BLECharacteristic::PROPERTY_NOTIFY
                    );
```

- Crea un servicio BLE con el UUID dado.
- Crea una característica con propiedad de notificación, para enviar datos.

```
 pCharacteristic->addDescriptor(new BLE2902());
  pService->start();
  pServer->getAdvertising()->start();
```

- Añade un descriptor estándar BLE.
- Inicia el servicio y comienza a hacer publicidad BLE.

```
 Serial.println("Bluetooth listo, esperando conexión...");
}
```

- Muestra en el monitor serie que el BLE está listo.

---

5. Bucle principal loop()

```
void loop() {
  sensors_event_t humidity, temp;
  aht.getEvent(&humidity, &temp);  // Leer sensor
```

- Solicita al AHT10 los datos de temperatura y humedad.

```
 if (deviceConnected) {
    char buffer[50];
    snprintf(buffer, sizeof(buffer), "Temp:%.1f;Hum:%.1f", temp.temperature,
humidity.relative_humidity);
```

- Si hay un dispositivo conectado:
    - Formatea los datos como una cadena: **`Temp:25.3;Hum:48.7`**

```
pCharacteristic->setValue((uint8_t*)buffer, strlen(buffer));
    pCharacteristic->notify();  // Notificar al cliente
```

- Asigna el valor a la característica BLE.
- Llama a `notify()` para enviar los datos al cliente.

```
   Serial.print("Enviado por BLE: ");
    Serial.println(buffer);
  }

  delay(2000); // Espera 2 segundos entre mediciones
}
```

- Muestra la cadena enviada por el monitor serie.
- Espera 2 segundos antes de repetir.

📲 ¿Cómo ver los datos en el celular?

1- Instala nRF Connect en tu teléfono.
2- Activa el Bluetooth.
3. Abre la app y escanea.
4. Busca el nombre ESP32-AHT10.
5. Conéctate y abre el servicio con UUID:
`12345678-1234-1234-1234-1234567890ab`.
6. Activa las notificaciones de la característica con UUID
`abcd1234-5678-90ab-cdef-1234567890ab`
7. Verás datos como:
`Temp:26.5;Hum:45.0`