Programa ESP8266 para sensor DS18B20 y Sonda TDS

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// **Configuración de la red WiFi**
const char *ssid = "Familia Baretta* 2.4GHz";      // Reemplaza con el
nombre de tu red WiFi
const char *password = "jesusesvida";              // Reemplaza con la
contraseña de tu red WiFi

// **Configuración de Adafruit IO**
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883                        // Puerto MQTT para
Adafruit IO
#define AIO_USERNAME    "Charly76"                  // Reemplaza con tu
nombre de usuario de Adafruit IO
#define AIO_KEY         "aio_peET13z39trG2TzxrBPVyNzC0qm9"  //
Reemplaza con tu llave de Adafruit IO

WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT,
AIO_USERNAME, AIO_KEY);

// **Configuración de los feeds de Adafruit IO**
Adafruit_MQTT_Publish temperatureFeed = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/temperatura");
Adafruit_MQTT_Publish tdsFeed = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/conductividad");

// **Configuración de ThingSpeak**
const char* THINGSPEAK_API_KEY = "XHHZDQFO5NNW6ZN4";  // Reemplaza con
tu Write API Key de ThingSpeak
const char* THINGSPEAK_SERVER = "api.thingspeak.com";
const int THINGSPEAK_PORT = 80;

// **Pines del sensor DS18B20**
#define ONE_WIRE_BUS D2  // Pin donde se conecta el DS18B20
OneWire oneWire(ONE_WIRE_BUS);
```

```
DallasTemperature sensors(&oneWire);

// **Pin del sensor TDS**
const int tdsPin = A0;

// **Variables globales para los datos**
float temperatureC = 0.0;
float tdsInPpm = 0.0;

// **Función para conectarse a la red WiFi**
void connectWiFi() {
  Serial.print("Conectando a ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\nConectado a WiFi");
  Serial.print("Dirección IP: ");
  Serial.println(WiFi.localIP());
}

// **Función para conectar al servidor de Adafruit IO**
void MQTT_connect() {
  int8_t ret;

  if (mqtt.connected()) {
    return;
  }

  Serial.print("Conectando al servidor de Adafruit IO...");

  while ((ret = mqtt.connect()) != 0) {
    Serial.println(mqtt.connectErrorString(ret));
    Serial.println("Reintentando conexión en 5 segundos...");
    delay(5000);
  }

  Serial.println("Conectado a Adafruit IO!");
}
```

```
// **Función para enviar datos a ThingSpeak**
void sendToThingSpeak(float temperature, float tds) {
  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;
    HTTPClient http;

    String url = "http://" + String(THINGSPEAK_SERVER) +
"/update?api_key=" + THINGSPEAK_API_KEY +
                 "&field1=" + String(temperature) +
                 "&field2=" + String(tds);

    http.begin(client, url);
    int httpCode = http.GET();

    if (httpCode > 0) {
      Serial.print("Datos enviados a ThingSpeak. Código de respuesta:
");
      Serial.println(httpCode);
    } else {
      Serial.print("Error al enviar datos a ThingSpeak: ");
      Serial.println(http.errorToString(httpCode).c_str());
    }

    http.end();
  } else {
    Serial.println("Error: No conectado a WiFi");
  }
}

// **Función para leer la temperatura**
void readTemperature() {
  sensors.requestTemperatures();
  temperatureC = sensors.getTempCByIndex(0);

  if (temperatureC == DEVICE_DISCONNECTED_C) {
    Serial.println("Error: No se pudo leer la temperatura del sensor
DS18B20");
    temperatureC = 0.0; // Valor por defecto en caso de error
  } else {
    Serial.print("Temperatura leída: ");
    Serial.print(temperatureC);
    Serial.println(" °C");
```

```
  }
}

// **Función para leer el valor TDS**
void readTDS() {
  int tdsValue = analogRead(tdsPin);
  float voltage = (tdsValue / 1024.0) * 5.0; // Ajusta si usas un
divisor de voltaje

  // Fórmula base para TDS sin corrección
  float tdsRaw = (133.42 * voltage * voltage * voltage) - (255.86 *
voltage * voltage) + (857.39 * voltage);

  // Corrección de temperatura
  float alpha = 0.02; // Coeficiente de temperatura
  tdsInPpm = tdsRaw / (1 + alpha * (temperatureC - 25.0))-1330;

  Serial.print("TDS leído (ajustado por temperatura): ");
  Serial.print(tdsInPpm);
  Serial.println(" ppm");
}

void setup() {
  Serial.begin(115200);
  sensors.begin();
  connectWiFi();
  MQTT_connect();
}

void loop() {
  // **Asegurarse de que la conexión a Adafruit IO esté activa**
  MQTT_connect();

  // **Leer sensores**
  readTemperature();
  readTDS();

  // **Publicar la temperatura en Adafruit IO**
  if (!temperatureFeed.publish(temperatureC)) {
    Serial.println("Error publicando la temperatura en Adafruit IO");
  } else {
    Serial.print("Temperatura publicada en Adafruit IO: ");
    Serial.print(temperatureC);
```

```
    Serial.println(" °C");
  }

  // **Publicar el valor TDS en Adafruit IO**
  if (!tdsFeed.publish(tdsInPpm)) {
    Serial.println("Error publicando el valor TDS en Adafruit IO");
  } else {
    Serial.print("Valor TDS publicado en Adafruit IO: ");
    Serial.print(tdsInPpm);
    Serial.println(" ppm");
  }

  // **Enviar datos a ThingSpeak**
  sendToThingSpeak(temperatureC, tdsInPpm);

  // **Esperar 30 segundos antes de la próxima lectura**
  delay(30000);
}
```