****

**Técnico superior en Telecomunicaciones**

**Programación**

Proyecto: Visualizador de Temperatura y Humedad

* **Librerías Importadas**
* cpp
* Copiar código
* #include <Adafruit\_Sensor.h>
* #include <DHT.h>
* #include <Adafruit\_NeoPixel.h>
* #include <WiFi.h>
* **Adafruit\_Sensor.h**: Es una librería general para el manejo de sensores. Se utiliza como base para la integración de varios sensores, incluyendo el DHT11.
* **DHT.h**: Específica para el sensor **DHT11**, utilizada para medir temperatura y humedad.
* **Adafruit\_NeoPixel.h**: Permite el control de LEDs direccionables **WS2812**, como cambiar colores y ajustar brillo.
* **WiFi.h**: Librería para gestionar la conexión **WiFi** del **ESP32**, lo que permite conectarse a una red inalámbrica.

**2. Definición de Pines y Configuración de Sensores**

* #define DHTPIN 15 // Pin donde está conectado el DHT11
* #define DHTTYPE DHT11
* DHT dht(DHTPIN, DHTTYPE);
* **DHTPIN 15**: Define que el sensor DHT11 está conectado al pin **GPIO 15** del ESP32.
* **DHTTYPE DHT11**: Especifica el tipo de sensor (en este caso el DHT11).
* **DHT dht(DHTPIN, DHTTYPE)**: Inicializa el objeto dht que se encargará de manejar las lecturas de temperatura y humedad desde el sensor DHT11.
* #define LED\_PIN 4 // Pin donde está conectado el WS2812
* #define NUMPIXELS 1 // Número de LEDs
* Adafruit\_NeoPixel strip(NUMPIXELS, LED\_PIN, NEO\_GRB + NEO\_KHZ800);
* **LED\_PIN 4**: Define que el **módulo WS2812** está conectado al pin **GPIO 4**.
* **NUMPIXELS 1**: Define que hay un solo LED en la cadena de LEDs.
* **Adafruit\_NeoPixel strip(NUMPIXELS, LED\_PIN, NEO\_GRB + NEO\_KHZ800)**: Crea un objeto strip para controlar el **LED WS2812**. NEO\_GRB define el formato de color, y NEO\_KHZ800 es la frecuencia de señal que usa el WS2812.

**3. Credenciales WiFi**

* const char\* ssid = "TU\_SSID"; // Cambia esto por tu SSID
* const char\* password = "TU\_PASSWORD"; // Cambia esto por tu contraseña
* **ssid**: Es el nombre de la red WiFi a la que el ESP32 se va a conectar.
* **password**: Es la contraseña de la red WiFi. Debes cambiar "TU\_SSID" y "TU\_PASSWORD" por tu red y contraseña WiFi reales.
* **4. Función setup()**
* void setup() {
* Serial.begin(115200);
* dht.begin();
* strip.begin();
* strip.show(); // Inicializar todos los LEDs apagados
* // Conectar a WiFi
* WiFi.begin(ssid, password);
* while (WiFi.status() != WL\_CONNECTED) {
* delay(500);
* Serial.print(".");
* }
* Serial.println("Conectado a WiFi");
* }
* **Serial.begin(115200)**: Inicializa la comunicación serial con una velocidad de **115200 baudios** para la depuración. El ESP32 enviará mensajes por el puerto serie que puedes leer en el monitor serial de PlatformIO.
* **dht.begin()**: Inicializa el sensor **DHT11** para que comience a tomar lecturas.
* **strip.begin()**: Inicializa el control del **LED WS2812**.
* **strip.show()**: Asegura que todos los LEDs estén apagados inicialmente.

**Conexión WiFi:**

* **WiFi.begin(ssid, password)**: Inicia el intento de conexión a la red WiFi con las credenciales que se definieron antes.
* El while (WiFi.status() != WL\_CONNECTED) verifica si la conexión WiFi ha sido exitosa. Mientras no esté conectada, el ESP32 espera medio segundo (500 ms) y muestra puntos (".") en el monitor serial para indicar que sigue intentando conectarse.
* Una vez que se conecta, se imprime "Conectado a WiFi" en el monitor serial.

**5. Función loop()**

* void loop() {
* // Leer temperatura y humedad
* float temp = dht.readTemperature();
* float hum = dht.readHumidity();
* **float temp = dht.readTemperature()**: Lee la temperatura en grados Celsius desde el sensor DHT11 y la guarda en la variable temp.
* **float hum = dht.readHumidity()**: Lee la humedad relativa en porcentaje desde el sensor DHT11 y la guarda en la variable hum.
* if (isnan(temp) || isnan(hum)) {
* Serial.println("Fallo en la lectura del sensor DHT11");
* return;
* }
* Si las lecturas de temperatura o humedad no son números válidos (isnan), significa que hubo un fallo en la lectura del sensor. En ese caso, se imprime "Fallo en la lectura del sensor DHT11" y se sale de la función.

**6. Mostrar Lecturas en el Monitor Serial**

* Serial.print("Temperatura: ");
* Serial.print(temp);
* Serial.print(" C, Humedad: ");
* Serial.print(hum);
* Serial.println(" %");
* Imprime las lecturas de **temperatura** y **humedad** en el monitor serial. Por ejemplo: "Temperatura: 25.5 C, Humedad: 60.0 %".

**7. Cambiar el Color del LED en Función de la Temperatura**

* if (temp > 30) {
* strip.setPixelColor(0, strip.Color(255, 0, 0)); // Rojo si temperatura > 30°C
* } else {
* strip.setPixelColor(0, strip.Color(0, 255, 0)); // Verde si temperatura <= 30°C
* }
* strip.show();
* **if (temp > 30)**: Si la temperatura supera los **30°C**, el **LED WS2812** se enciende en **rojo** (255, 0, 0).
* **else**: Si la temperatura es igual o inferior a **30°C**, el LED se enciende en **verde** (0, 255, 0).
* **strip.show()**: Actualiza el LED para reflejar el nuevo color.

**8. Pausa entre Lecturas**

* delay(2000);
* Hace una pausa de **2 segundos** antes de volver a leer la temperatura y la humedad nuevamente. Esto limita la frecuencia de las lecturas y el cambio de color en el LED.

**9. Lectura del nivel de Batería**

* // Leer el nivel de batería (voltaje)
* int adcValue = analogRead(BATTERY\_PIN); // Leer el valor ADC del pin de batería
* float batteryVoltage = (adcValue / 4095.0) \* 3.3 \* (R1 + R2) / R2; // Calcular el voltaje real de la batería
* // Mostrar el nivel de batería en el monitor serial
* Serial.print("Nivel de batería: ");
* Serial.print(batteryVoltage);
* Serial.println(" V");
* // Cambiar el color del LED en función de la temperatura
* if (temp > 30) {
* strip.setPixelColor(0, strip.Color(255, 0, 0)); // Rojo si temperatura > 30°C
* } else {
* strip.setPixelColor(0, strip.Color(0, 255, 0)); // Verde si temperatura <= 30°C
* }

**Explicación**

1. **Lectura del ADC**

* analogRead(BATTERY\_PIN): Se utiliza para leer el valor de voltaje analógico desde el pin GPIO 36 (ADC0). El valor ADC es un número de 0 a 4095, donde 0 corresponde a 0V y 4095 corresponde al voltaje de referencia (3.3V en el caso del ESP32).

Cálculo del Voltaje de Batería:

* El valor ADC se convierte a un voltaje real utilizando la fórmula:
* float batteryVoltage = (adcValue / 4095.0) \* 3.3 \* (R1 + R2) / R2;
* Donde:
  + - adcValue es el valor leído por el ADC.
    - 3.3 es el voltaje de referencia del ESP32.
    - R1 y R2 son las resistencias del divisor de voltaje que hemos conectado. Ajusta estos valores según las resistencias que uses. Por ejemplo, si usas una resistencia de 100 kΩ para R1 y 200 kΩ para R2, la fórmula queda:
* float batteryVoltage = (adcValue / 4095.0) \* 3.3 \* 3;

1. **Mostrar el nivel de batería**
   * El nivel de voltaje de la batería se imprime en el monitor serial:

* Serial.print("Nivel de batería: ");
* Serial.print(batteryVoltage);
* Serial.println(" V");

1. **Ajustar el delay**
   * Se deja el delay(2000) para que las lecturas de batería, temperatura y humedad se actualicen cada 2 segundos.

**Resumen del Comportamiento del Código**

* El ESP32 se conecta a la red WiFi al iniciar.
* Lee la temperatura y humedad del sensor **DHT11**.
* Si la lectura es válida, muestra la temperatura y la humedad en el monitor serial.
* Dependiendo de la temperatura, cambia el color del **LED WS2812**: rojo si la temperatura es mayor de 30°C, verde si es menor o igual.
* Repite este proceso cada 2 segundos.
* Mide el nivel de batería.