Técnico superior en Telecomunicaciones

Programación

Proyecto: Visualizador de Temperatura y Humedad

Alumnos:

• Macarena Aylén Carballo.

• Nicolás Nahuel Barrionuevo.

• Raúl Antonio Jara.

Profesor:

• Lisandro Lanfranco

Informe de la Entrega N°3

1. Introducción

En esta tercera entrega, se trabajó en el desarrollo de un sistema de monitoreo ambiental utilizando

un microcontrolador ESP32, un sensor DHT11 para la medición de temperatura y humedad, y un

LED WS2812 como actuador visual. El sistema fue implementado utilizando PlatformIO como

entorno de desarrollo en el lenguaje C++, con funcionalidades para adquirir datos del sensor y

emitir una alerta visual en función de la temperatura ambiente.

Este informe detalla las herramientas utilizadas, el funcionamiento del código, y las conclusiones

obtenidas durante esta etapa del proyecto.

2. Descripción del Proyecto

El objetivo principal de esta entrega fue desarrollar un sistema IoT que monitoree la temperatura y

humedad de un ambiente, mostrando visualmente el estado de la temperatura mediante un LED.

Para ello, se utilizó el sensor DHT11 y un LED WS2812 controlado por el ESP32. El LED cambia de

color según el nivel de temperatura detectado:

- Rojo: Temperatura superior a 30°C.

- Amarillo: Temperatura entre 20°C y 30°C.

- Verde: Temperatura inferior o igual a 20°C.

Además, los datos de temperatura y humedad se envían al puerto serial para su posterior análisis.

3. Explicación Técnica del Código

3.1. main.cpp

Este archivo contiene el código principal del programa, donde se inicializan el sensor DHT11 y el

LED WS2812, y se implementa el ciclo loop() que realiza la lectura de datos y controla el

comportamiento del LED.

A continuación, se muestran las partes más importantes del código:

```
// Definición de pines y tipo de sensor DHT11
#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
WS2812 led(13, 1);
void setup() {
 Serial.begin(115200);
 dht.begin();
 led.begin();
}
void loop() {
 float temperatura = leerTemperatura(dht);
 float humedad = leerHumedad(dht);
 String color;
 if (temperatura > 30.0) {
```

```
 color = "rojo";
 led.cambiarColor(255, 0, 0);
 } else if (temperatura > 20.0) {
 color = "amarillo";
 led.cambiarColor(255, 255, 0);
 } else {
 color = "verde";
 led.cambiarColor(0, 255, 0);
 }
 Serial.print(temperatura);
 Serial.print(",");
 Serial.print(humedad);
 Serial.print(",");
 Serial.println(color);
 delay(2000);
}
```

3.2. sensor_dht11.cpp

Este archivo define las funciones para interactuar con el sensor DHT11. Se implementan dos

funciones:

```
float leerTemperatura(DHT &dht) {
 float t = dht.readTemperature();
 if (isnan(t)) {
 Serial.println("Error al leer la temperatura");
 return 0.0;
 }
```

```
 return t;
}
float leerHumedad(DHT &dht) {
 float h = dht.readHumidity();
 if (isnan(h)) {
 Serial.println("Error al leer la humedad");
 return 0.0;
 }
 return h;
}
```

3.3. ws2812.cpp

Este archivo define una clase para controlar el LED WS2812. La clase tiene los siguientes métodos:

```
WS2812::WS2812(int pin, int numPixeles) : strip(numPixeles, pin, NEO_GRB + NEO_KHZ800)
{}
void WS2812::begin() {
 strip.begin();
 strip.show();
}
void WS2812::cambiarColor(uint8_t r, uint8_t g, uint8_t b) {
 strip.setPixelColor(0, strip.Color(r, g, b));
 strip.show();
}
```

4. Herramientas Utilizadas

- PlatformIO: Entorno de desarrollo para trabajar con el ESP32.

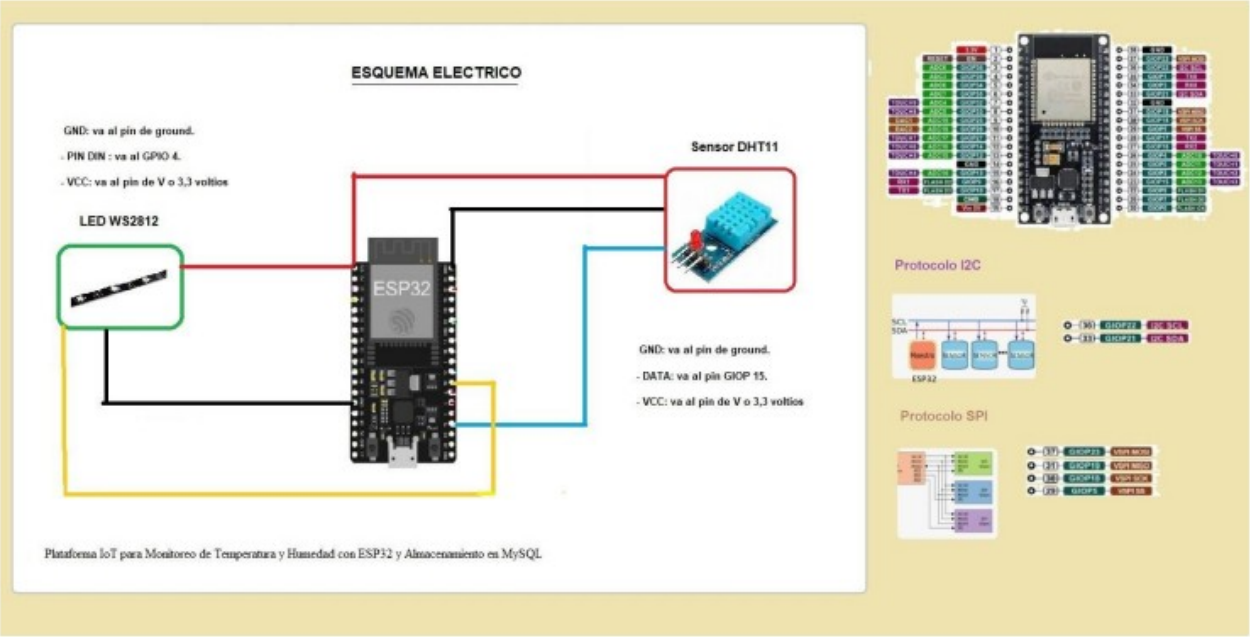
- ESP32: Microcontrolador utilizado para controlar el sensor DHT11 y el LED WS2812.

- Sensor DHT11: Sensor para medir temperatura y humedad.

- LED WS2812: LED RGB utilizado como actuador visual.

- Monitor Serial: Herramienta para visualizar los datos transmitidos por el puerto serial.

Diagrama de Conexión



5. Conclusión

En esta entrega, logramos integrar exitosamente el sensor DHT11 y el LED WS2812 en un sistema

capaz de monitorear la temperatura y humedad de un ambiente en tiempo real, utilizando el ESP32.

La implementación en C++ y la utilización de PlatformIO nos permitieron organizar el código de

manera modular, facilitando el mantenimiento y escalabilidad futura del sistema.

Además, la interfaz serial implementada permitirá, en futuras etapas, agregar almacenamiento de

datos y un análisis más profundo del comportamiento ambiental, lo que abre la posibilidad de

integrar conectividad a internet y análisis remoto.