****

**Técnico superior en Telecomunicaciones**

**Programación**

Proyecto: Visualizador de Temperatura y Humedad

Alumnos:

• Karina Jazmin Barbero.

• Macarena Aylén Carballo.

• Nicolás Nahuel Barrionuevo.

• Raúl Antonio Jara.

Profesor:

• Lisandro Lanfranco

**Índice**

Descripción del Proyecto................................................................................................................... 2

Objetivo……………............................................................................................................................... 2

Estructura de Archivos del Proyecto................................................................................................. 4

Implementación Técnica................................................................................................................... 6

Documentación de Pruebas y Optimización..................................................................................... 6

Conclusión........................................................................................................................................ 7

**Informe de Proyecto: Visualizador de Temperatura y Humedad con ESP32 y Sensor DHT11**

**Descripción del proyecto**

Este proyecto implementa un sistema de monitoreo de temperatura y humedad mediante el microcontrolador ESP32, usando un sensor DHT11 y un LED WS2812 como indicador visual. Los datos capturados por el sensor se envían a una base de datos MySQL mediante una conexión WiFi, facilitando el análisis y almacenamiento.

La estructura del proyecto sigue una organización modular en el entorno PlatformIO y está escrita en C++, cumpliendo con los requisitos funcionales y de conectividad para un sistema embebido IoT.

**Objetivos**

1. **Recopilar datos del sensor y preparar los datos para su envío** : Capturar valores de temperatura y humedad del sensor DHT11.
2. **Programación del comportamiento del actuador** : Actualizar el color del LED WS2812 según los valores de temperatura.
3. **Probar la comunicación entre el dispositivo IoT y la PC** : Asegurar una conexión correcta entre el ESP32 y la base de datos MySQL en la PC.
4. **Optimización y corrección** : Realizar pruebas y ajustes para garantizar un funcionamiento confiable y eficiente.
5. **Documentación del código y de las pruebas** : Generar documentación completa y clara del código fuente y de las pruebas realizadas.

**Estructura de Archivos del Proyecto**

El proyecto se organiza en tres archivos principales:

1. **sensor.h** : Define la clase Sensor\_DHT11, responsable de encapsular la configuración del sensor DHT11, la comunicación con la base de datos MySQL y el control del LED.
2. **sensor.cpp** : Implementa los métodos de la clase Sensor\_DHT11para inicializar el sensor y LED, leer los datos del sensor, enviar los datos a MySQL y actualizar el LED en función de la temperatura.
3. **main.cpp** : Configura los parámetros de conexión WiFi, los datos de la base de datos, y el loop principal del ESP32 que gestiona la lectura de datos y actualización del LED.

**Implementación Técnica**

**Archivo sensor.h**

Este archivo contiene la definición de la clase Sensor\_DHT11y sus métodos principales.

#ifndef SENSOR\_H

#define SENSOR\_H

#include <Arduino.h>

#include <DHT.h>

#include <Adafruit\_NeoPixel.h>

#include <MySQL\_Connection.h>

#include <MySQL\_Cursor.h>

#include <WiFi.h>

class Sensor\_DHT11 {

public:

Sensor\_DHT11(int dhtPin, int dhtType, int ledPin, int numPixels);

void conectarMySQL(const char\* user, const char\* password, const char\* host, int port);

void begin();

void leerDatos();

void enviarDatosMySQL(float temperatura, float humedad);

void actualizarLED();

void conectarWiFi(const char\* ssid, const char\* password);

private:

int dhtPin;

int dhtType;

int ledPin;

int numPixels;

DHT dht;

Adafruit\_NeoPixel strip;

WiFiClient client;

MySQL\_Connection conn{&client};

MySQL\_Cursor\* cursor;

};

#endif // SENSOR\_H

**Archivo sensor.cpp**

Implementa los métodos de la clase Sensor\_DHT11.

1. **Constructor de la clase** : Configura el sensor DHT11 y el LED WS2812.
2. **MétodoconectarMySQL** : Establece la conexión con la base de datos MySQL.
3. **Métodobegin** : Inicializa el sensor y el LED.
4. **MétodoleerDatos** : Captura los datos de temperatura y humedad del sensor y los envía a la base de datos.
5. **MétodoenviarDatosMySQL** : Inserta los valores de temperatura y humedad en la base de datos MySQL.
6. **MétodoactualizarLED** : Cambia el color del LED en función de la temperatura.
7. **MétodoconectarWiFi** : Conecta el ESP32 a la red WiFi especificada.

#include "sensor.h"

// Constructor de la clase Sensor

Sensor\_DHT11::Sensor\_DHT11(int dhtPin, int dhtType, int ledPin, int numPixels)

: dhtPin(dhtPin), dhtType(dhtType), ledPin(ledPin), numPixels(numPixels),

dht(dhtPin, dhtType), strip(numPixels, ledPin, NEO\_GRB + NEO\_KHZ800) {}

// Método conectarMySQL para establecer la conexión con la base de datos

void Sensor\_DHT11::conectarMySQL(const char\* user, const char\* password, const char\* host, int port) {

IPAddress server\_ip;

if (WiFi.hostByName(host, server\_ip)) {

if (conn.connect(server\_ip, port, (char\*)user, (char\*)password)) {

cursor = new MySQL\_Cursor(&conn);

}

}

}

// Método para inicializar el sensor DHT y los LEDs

void Sensor\_DHT11::begin() {

dht.begin();

strip.begin();

strip.show();

}

// Método para leer datos de temperatura y humedad

void Sensor\_DHT11::leerDatos() {

float temperatura = dht.readTemperature();

float humedad = dht.readHumidity();

if (!isnan(temperatura) && !isnan(humedad)) {

enviarDatosMySQL(temperatura, humedad);

}

}

// Método enviar DatosMySQL

void Sensor\_DHT11::enviarDatosMySQL(float temperatura, float humedad) {

if (conn.connected()) {

char query[128];

snprintf(query, sizeof(query),

"INSERT INTO mediciones (temperatura, humedad) VALUES (%.2f, %.2f);",

temperatura, humedad);

cursor->execute(query);

}

}

// Método para actualizar el LED según la temperatura

void Sensor\_DHT11::actualizarLED() {

float temperatura = dht.readTemperature();

if (temperatura > 30) {

strip.setPixelColor(0, strip.Color(255, 0, 0));

} else {

strip.setPixelColor(0, strip.Color(0, 255, 0));

}

strip.show();

}

// Método para conectar a WiFi

void Sensor\_DHT11::conectarWiFi(const char\* ssid, const char\* password) {

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

}

}

**Archivo main.cpp**

Contiene la configuración inicial y el bucle principal del sistema.

#include <Arduino.h>

#include "sensor.h"

// Configuración de pines y credenciales

const int DHT\_PIN = 15;

const int DHT\_TYPE = DHT11;

const int LED\_PIN = 4;

const int NUM\_PIXELS = 1;

const char\* ssid = "Raul";

const char\* password = "ni334233";

const char\* db\_user = "root";

const char\* db\_password = "root";

const char\* db\_host = "localhost";

const int db\_port = 3306;

Sensor\_DHT11 sensor(DHT\_PIN, DHT\_TYPE, LED\_PIN, NUM\_PIXELS);

void setup() {

Serial.begin(115200);

sensor.conectarWiFi(ssid, password);

sensor.conectarMySQL(db\_user, db\_password, db\_host, db\_port);

sensor.begin();

}

void loop() {

sensor.leerDatos();

sensor.actualizarLED();

delay(2000);

}

**Documentación de Pruebas y Optimización**

**Pruebas realizadas**

1. **Prueba de conexión WiFi y MySQL** : Verificación de la conexión inicial con la red y la base de datos MySQL.
2. **Prueba de lectura del sensor** : Evaluación de la captura de datos de temperatura y humedad para asegurar valores precisos.
3. **Prueba de actualización del LED** : Verificación del cambio de color según la medida de temperatura.
4. **Prueba de envío de datos a MySQL** : Confirmación de la correcta inserción de datos en la tabla mediciones.

**Correcciones y Optimización**

* **Manejo de errores** : Añadido para reconexiones en caso de pérdida de WiFi o MySQL.
* **Ajuste de lectura de datos** : Validación para evitar valores NaNde la lectura del sensor.

**Conclusión**

Este proyecto demuestra una aplicación práctica de IoT usando ESP32, DHT11 y MySQL para monitoreo ambiental en tiempo real, proporcionando un sistema confiable y extensible para aplicaciones en domótica o industrial.