TECNICATURA SUPERIOR EN TELECOMUNICACIONES

VISUALIZADOR DE TEMPERATURA Y HUMEDAD

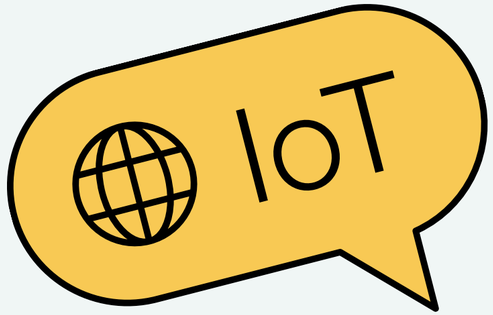
Modulo Programacion

- Lisandro Lanfranco.

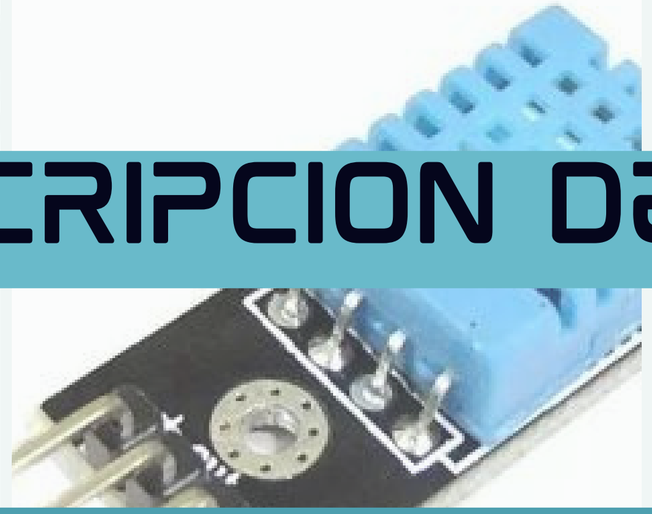
Estudiantes:

- Nicolas Barrionuevo.
- Raul Jara.
- Macarena Carballo.

OBJETIVOS



- Recopilar datos del sensor y preparar los datos para su envío.
- Programación del comportamiento del actuador.
- Probar la comunicación entre el dispositivo IoT y la PC .
- Optimización y corrección.
- Documentación del código y de las pruebas.



DESCRIPCION DEL PROYECTO

- Microcontrolador: ESP32.
- Sensor: DHT11.
- Indicador visual: LED WS2812
- Base de datos: MySQL mediante conexión WiFi.
- Entorno: PlatformIO.
- Lenguajes: C++.

ESTRUCTURA DE ARCHIVOS

- **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.
- **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.
- **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.

IMPLEMENTACIÒN TÈCNICA

```
#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.H

Este archivo contiene la definición de la clase Sensor_DHT11 y sus métodos principales.

ARCHIVO SENSOR.CPP

Implementa los métodos de la clase Sensor_DHT11.

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

ARCHIVO SENSOR.CPP

```
#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);
  }
```

Continua

ARCHIVO SENSOR.CPP

```
// 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);
  }
```

```
#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);
}
```

ARCHIVO MAIN.CPP

Contiene la configuración inicial y el bucle principal del sistema.

DIAGRAMA DE CONEXIÒN

ESQUEMA ELECTRICO

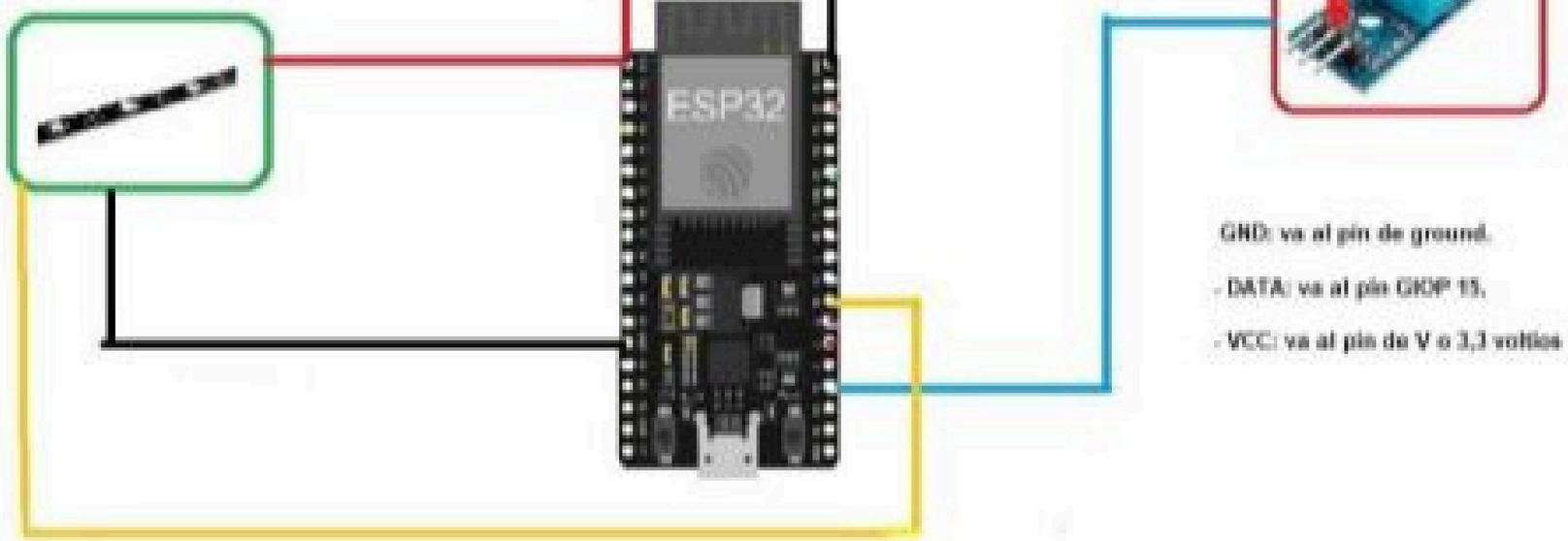
GND: va al pin de ground.

- PIN DIN : va al GPIO 4.

- VCC: va al pin de V o 3,3 voltios

LED WS2812

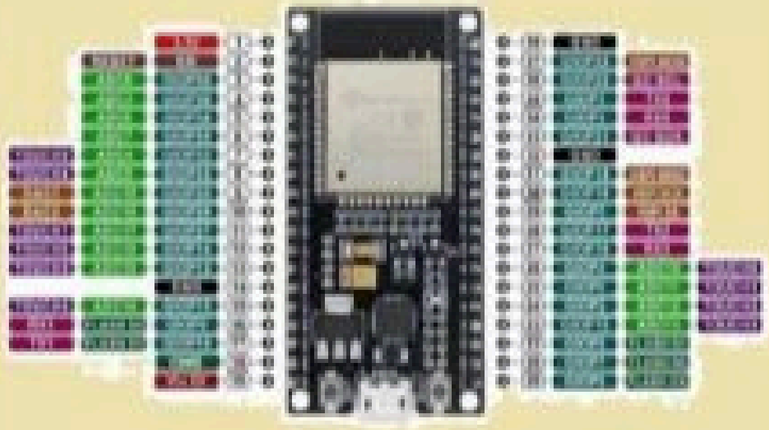
Sensor DHT11



GND: va al pin de ground.

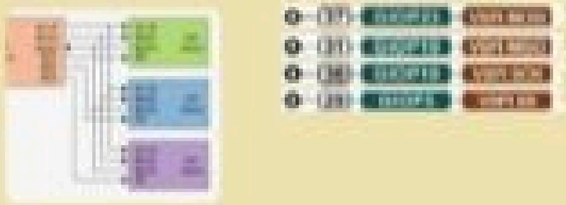
- DATA: va al pin GIOP 15.

- VCC: va al pin de V o 3,3 voltios



Protocolo I2C

Protocolo SPI



CODIGO PYTHON

```python
import mysql.connector  # Importa el módulo para interactuar con una base de datos MySQL
from mysql.connector import Error  # Importa la clase de errores para manejar excepciones en MySQL
import serial  # Importa la librería para manejar la comunicación serial
import time  # Importa el módulo para manejar retrasos de tiempo

# Conectar al puerto serial (ajusta 'COM3' al puerto que estés usando)
ser = serial.Serial('COM3', 115200, timeout=1)  # Configura el puerto serial con la velocidad en baudios (1
time.sleep(2)  # Espera 2 segundos para establecer la conexión serial correctamente

# Función para conectar a la base de datos MySQL
def conectar_base_datos():
    try:
        # Intenta establecer la conexión con los parámetros proporcionados
        conexion = mysql.connector.connect(
            host='localhost',  # Dirección del servidor MySQL (localhost en este caso)
            database='sensores',  # Nombre de la base de datos
            user='root',  # Usuario de la base de datos
            password='123456'  # Contraseña del usuario de la base de datos
        )
        if conexion.is_connected():
            print("Conexión exitosa a la base de datos")
            return conexion  # Devuelve la conexión si se establece correctamente
    except Error as e:
        # Muestra un mensaje de error si la conexión falla
        print(f"Error de conexión: {e}")
        return None  # Devuelve None si no se puede conectar

# Función para insertar los datos de temperatura, humedad y el color del LED en la base de datos
def insertar_datos(temperatura, humedad, color):
    conexion = conectar_base_datos()  # Conectar a la base de datos
    if not conexion:
        return  # Si no se puede conectar, termina la función
```

CODIGO PYTHON

```python
    try:
        with conexion.cursor() as cursor:
            # Inserta los datos de temperatura y humedad en la tabla 'Mediciones'
            query = "INSERT INTO Mediciones (temperatura, humedad) VALUES (%s, %s)"
            cursor.execute(query, (temperatura, humedad))

            # Inserta el color del LED en la tabla 'Estado_led'
            query_led = "INSERT INTO Estado_led (color) VALUES (%s)"
            cursor.execute(query_led, (color,))

            conexion.commit()  # Guarda los cambios en la base de datos
            print(f"Datos insertados: Temperatura = {temperatura}, Humedad = {humedad}, Color LED = {color}
    except Error as e:
        # Si ocurre un error al insertar los datos, se muestra un mensaje
        print(f"Error al insertar los datos: {e}")
    finally:
        conexion.close()  # Cierra la conexión a la base de datos

# Bucle principal que ejecuta continuamente el programa
while True:
    # Si hay datos disponibles en el puerto serial
    if ser.in_waiting > 0:
        # Lee la línea del puerto serial, la decodifica y elimina los espacios en blanco
        line = ser.readline().decode('utf-8').strip()

        # Divide los datos recibidos separados por comas
        data = line.split(',')

        # Si se reciben exactamente 3 elementos (temperatura, humedad, color LED)
        if len(data) == 3:
            temperatura, humedad, color = data  # Asigna cada valor a sus respectivas variables
```

CODIGO PYTHON

```
64
65                  # Inserta los datos en la base de datos
66                  insertar_datos(float(temperatura), float(humedad), color)
67
68          # Espera 2 segundos antes de leer nuevamente del puerto serial
69          time.sleep(2)
70
```

Fin del còdigo

DOCUMENTACION DE PRUEBAS Y OPTIMIZACION

PRUEBAS REALIZADAS

- **Conexión WiFi y MySQL:** Verificación de la conexión inicial con la red y la base de datos MySQL.
- **Lectura del sensor**: Evaluación de la captura de datos de temperatura y humedad para asegurar valores precisos.
- **Actualización del LED**: Verificación del cambio de color según la medida de temperatura.
- **Envío de datos a MySQL:** Confirmación de la correcta inserción de datos en la tabla mediciones.

CORRECCIONES

- **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.

CONCLUSIÒN

En resumen, cumplimos con el objetivo de crear una aplicacion practica de IoT, usando ESP32, DHT11 y MySql para monitoreo ambiental en tiempo real. Ademas, proporcionamos un sistema confiable y extensible para aplicaciones en domòtica o industrial. Este proyecto sirve de base para seguir midiendo nuevos parametros conectando nuevos sensores y poder tener mejor escabilidad en el futuro.

MUCHAS GRACIAS