Proyecto de Temperatura y Humedad

Erik Harrinson Mancera Bernal, Camila Sanchez Amaya
Fundación Universitaria del Área Andina
Bogotá Colombia
emancera2@estudiantes.areandina.edu.co
csanchez130@estudiantes.areandina.edu.co

Resumen - Esta actividad muestra cómo se puede realizar la conexión entre un ESP32 y un sensor de temperatura y humedad mediante una API, para obtener información del ambiente en que nos encontramos en tiempo real, que al conectarse a una red wifi se encenderá el ledRGB y se podrá visualizar, tanto si, la conexión es exitosa o no, así como los datos del sensor, en la pantalla OLED, y en una página web, haciendo uso de Micropython.

Abstrac - This activity shows how the connection between an ESP32 and a temperature and humidity sensor can be made through an API, to obtain information about the environment in which we are in real time, which when connected to a wifi network will turn on the ledRGB and can be displayed, whether the connection is successful or not, as well as the sensor data, on the OLED screen, and on a web page, using Micropython.

INTRODUCCION

Los sensores de temperatura y humedad, como su nombre lo mencionan nos permite conocer el ambiente en que nos encontramos con mayor exactitud, mayor mente necesarios para espacios cerrados, con esta herramienta se busca de manera sencilla y dinámica, se pueda estar al tanto de la temperatura y humedad de cualquier lugar en diferentes dispositivos

MANUAL DE USUARIO

El usuario debera ejecuatarlo solamente en la placa ESP32, al correrlo lo primero que observara en la pantalla, sera si la conexión a la red Wi-Fi fue exitosa o no y el ledRGB encendera verde o rojo según corresponda, como se puede observar en el la Fig.1 donde la conexión fue exitosa

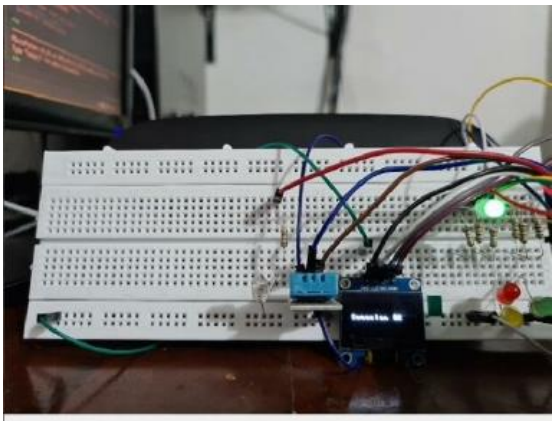

Fig.1 Conexión a red exitosa

Ahora en la pantalla se mostrará los resultados de la toma que realizo el sensor

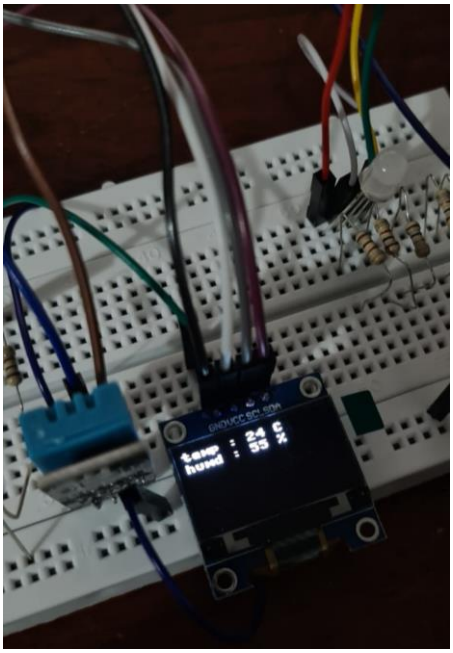

Fig. 2 Datos de la lectura del sensor

Por medio del servidor Web podremos ver gráficamente los datos que ha tomado el sensor

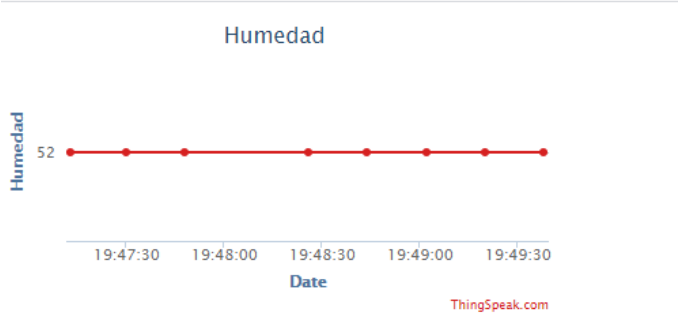

Fig.3 Grafica de Humedad

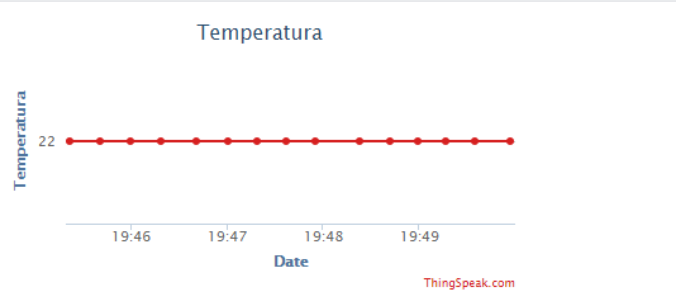

Fig.4 Grafica de Temperatura

MANUAL DE PROGRAMADOR

Para esta herramienta hacemos las respectivas variables para los dispositivos que usaremos de acuerdo con las necesidades y luego cómo deseamos visualizar los datos adquiridos. Veamos

1. *ESP32*

Como se observa en la Fig.5, se indica al microprocesador los datos de una red Wi-Fi, para que una vez activo se conecte a esta.

```
ssid = '******'
password = '*******'
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)
```

Fig.5 Conexión inicial de Wi-Fi

2. *Pantalla OLED*

Se especifica el tipo de pantalla la clase que se usa y como está conectada.

```
# ESP32 Pin assignment
i2c = SoftI2C(scl=Pin(22), sda=Pin(21))
oled = ssd1306.SSD1306_I2C(128, 64, i2c)
```

Fig.6 Configuración de pantalla OLED

3. *Sensor de Temperatura y Humedad*

Al igual que con la pantalla oled se le indica al ESP32 el tipo de sensor que es

```
sensort = dht.DHT11(Pin(13))
sleep(2)
sensort.measure()
```

Fig.7 Configuración sensor de temperatura.

4. *Led RGB*

Luego de indicar el dispositivo que se conectará, se crea un ciclo que verifique la conexión, se crean las condiciones que notificarán las conexiones exitosas. El led mostrará un color especifico y si la conexión es fallida igualmente el color y mensaje por mostrar

```
    if wlan.isconnected() == True:#Conexión Exitosa
        rgb.verde() #En caso de ser exitosa enciende el led RGB verde
        utime.sleep(1)
        rgb.apagado()
        print(wlan.ifconfig()) #Muestra la IP y otros datos del Wi-Fi
        oled.text('Conexion OK', 10, 32) #Muestra mensaje en pantalla oled
        oled.show()
    else: #Conexión no Exitosa
        rgb.rojo() #En caso de no ser exitosa enciende el led RGB Rojo
        utime.sleep(1)
        rgb.apagado()
        print(wlan.ifconfig())
        oled.text('Conexion No/OK', 10, 32)#Muestra mensaje en pantalla oled
        oled.show()
_thread.start_new_thread(ConexionRED,())
```

Fig.8 Condiciones si la conexión es exitosa o no

5. *Servicio de Web*

Permite mostrar de manera gráfica y cronológica los datos procedentes del sensor, manteniéndose actualizado para una mejor exactitud

```
def web_page():
    html = """
<html>
<head>
<title>Medidor de humedad y temperatura</title>
</head>
<body>
<center>
<iframe width="450" height="260" style="border: 1px solid #cccccc;"
src="https://thingspeak.com/channels/1447645/charts/1?
bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=15&type=spline"></iframe>
<br>
<br>
<iframe width="450" height="260" style="border: 1px solid #cccccc;"
src="https://thingspeak.com/channels/1454840/charts/1?
bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=10&type=spline"></iframe>
</center>
</body>
</html>  """
    return html
tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp_socket.bind(('', 80))
tcp_socket.listen(3)
```

Fig.9 Servicio Web que muestra graficamente los datos

6. *API ThingSpeak*

Se hace uso de la API ThingSpeak que nos permite tomar los datos del sensor y plasmarlos en una gráfica.

```
while True:
    oled.fill(0)
    sleep(2)
    sensort.measure()
    t = sensort.temperature()
    h = sensort.humidity()
    oled.text('temp : '+str(t) + " C", 0,0,1)
    oled.text('humd : '+str(h) + " %", 0,10,1)
    oled.show()
    utime.sleep(2)
    url1="https://api.thingspeak.com/update?api_key=UF6WR2PQ8W3EFKS5&field1="
    url1 += str(t)
    r1 = urequests.get(url1)
    print(r1.json())
    url2="https://api.thingspeak.com/update?api_key=6ZLA92ULEGMRBEWU&field1="
    url2 += str(h)
    r2 = urequests.get(url2)
    print(r2.json())
```

Fig.10 Estructura de API

CONCLUSIONES

Como mejora se proyecta que el ESP32 se pueda conectar por corriente para un uso más eficaz e independiente