Sistema de medición y monitorización de ambientes (julio de 2021)

Hugo Rafael Villero González
Fundación universitaria del área andina

Resumen - en el presente documento se pretende mostrar el proceso de desarrollo e implementación de un sistema de para monitorear y realizar la medición de temperatura y humedad dentro los espacios dedicados al almacenamiento de documentos, donde en ellos se hace indispensable contener unos niveles ambientales adecuados y que requieren una constante monitorización, ya que dichos documentos representan un valor inmaterial e histórico para toda organización.

Abstract - in this document it is intended to show the process of development and implementation of a system to monitor and measure temperature and humidity within the spaces dedicated to the storage of documents, where it is essential to contain adequate environmental levels and that require constant monitoring, since these documents represent an immaterial and historical value for any organization.

Introducción

A lo largo de los años la información siempre ha tenido una enorme importancia, como para comunicarnos, entretenimiento y actualmente en ámbitos laborales, donde la información, por medio de las tecnologías ha cobrado aún mucho más valor, para el análisis y manejo de estas. Toda esta información se maneja en espacios destinados para almacenamiento y para su posterior consulta, y a su vez se necesita que esta misma perdure la mayor cantidad disponible, de esta manera se debe tener condiciones de luz, temperatura, humedad y en lugares en buenas condiciones para que no ocurra algún altercado con estas características antes mencionadas. Para un buen manejo de temperatura y humedad se requiere implementar lo descrito en el acuerdo 49 de 2000 archivo general de la nación, en su artículo número 4 – condiciones ambientales técnicas, donde las condiciones ambientales como temperatura deben estar dentro de los rangos entre 15 y 20 grados centígrados, como también la humedad relativa a unos rangos recomendados de entre 45 y 60 porciento, todo esto varía de acuerdo al tipo de archivo a almacenar, en este caso hace referencia al soporte de papel.

Manual de usuario

Se Observa el circuito conformado por una pantalla oled y el sensor Dht11 utilizado para medir la temperatura y humedad relativa conectados al dispositivo Esp 32.

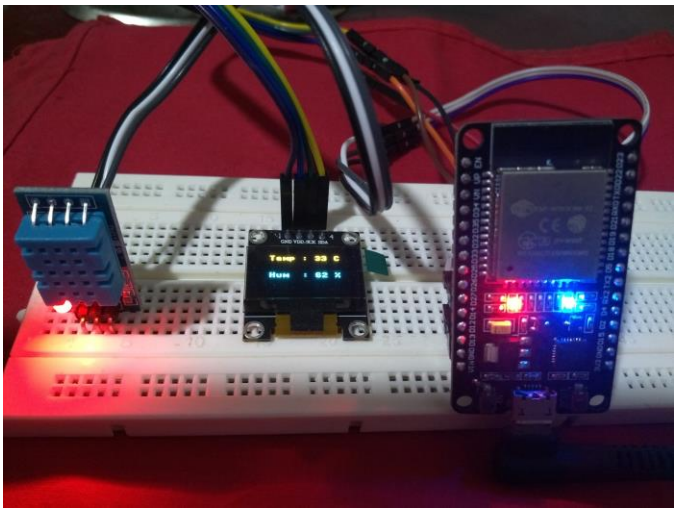

Figura 1
Fuente: propia

Para su utilización se utiliza el aplicativo Thonny donde se especifican las instrucciones de inicio.

```
MicroPython v1.16 on 2021-06-23; ESP32 module with ESP32
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
  Para iniciar digite w para accceder al servidor web, de lo contrario
  digite p para visualizar los datos en la pantalla Oled
  p
```

Figura 2
Fuente: propia

Donde se le da la opción al usuario de utilizar el dispositivo de distintas maneras, por ejemplo, mostrar los datos del sensor por medio de la pantalla oled presionando la letra “P” o acceder al servidor web presionando la tecla “W”

```
Para iniciar digite w para accceder al servidor web, de lo contrario
digite p para visualizar los datos en la pantalla Oled
w
.................Conexión establecida con red2
('192.168.137.188', '255.255.255.0', '192.168.137.1', '192.168.137.1'
)
```

Figura 3
Fuente: propia

Lo siguiente seria dirigirse a la web local que se relaciona con la ip y mostrará los siguientes datos que recogen el sensor.

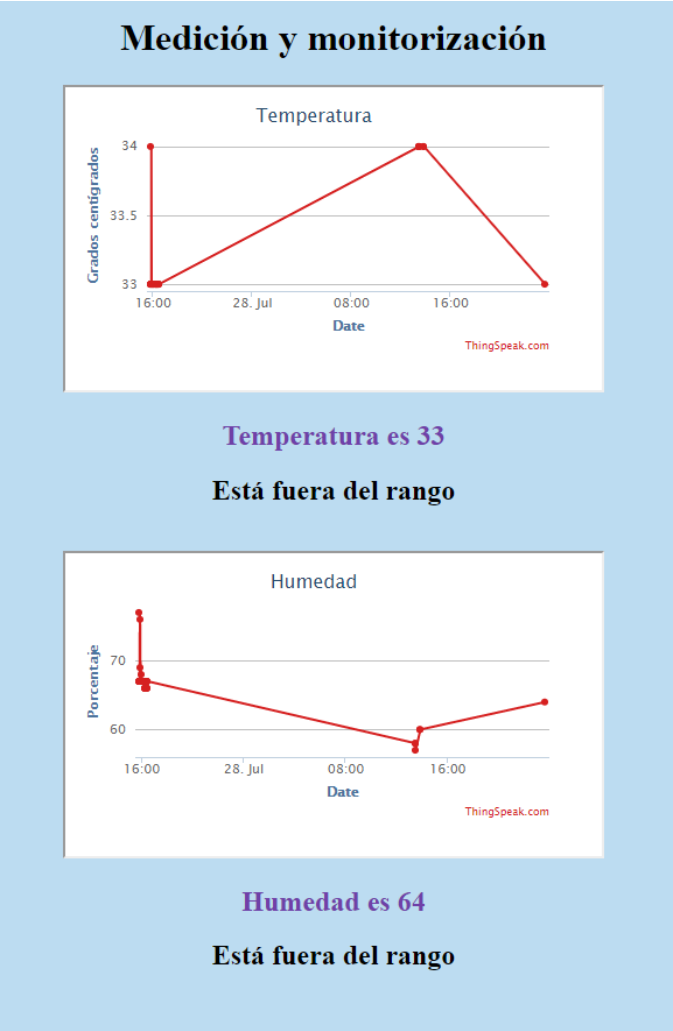

Figura 4
Fuente: propia

Manual del programador

Tenemos el archivo main.py donde importamos los módulos creados de pantalla y webserver. Además, agregamos un ciclo para controlar si el usuario no ingresa el carácter correcto no le permita el acceso

```
import pantalla
import webserver

print("Para iniciar digite w para accceder al servidor web,
dato =input()

while dato != "w" and dato != "p":
    print("Digite la letra correcta")
    dato =input()
```

Figura 5
Fuente: propia

También dentro del main.py tomaremos los datos dependiendo lo que el usuario digite por medio de condicionales controla la entrada.

```
if dato == "w":
    webserver.web()
elif dato == "p":
    pantalla.proyectar()
    print("Visualizando en pantalla Oled")
```

Anteriormente se deben crear los siguientes archivos:
pantalla.py
sensor.py
webserver.py
wifi.py

Nos disponemos a importar las librerías dentro del archivo sensor para su correcto funcionamiento

```
import dht
from time import sleep
from machine import Pin
```

Y creamos la función recibir

```
def recibir():
    """Recibir los datos del sensor y almacenarlos en varriables"""
    #Asignando pin del sensor dht11
    sensor = dht.DHT11(Pin(14))

    #método de medición
    sensor.measure()
    #asignando el valor de la temperatura a la variable "temp"
    temp = sensor.temperature()
    #asignando el valor de la humedad a la variable "hum"
    hum = sensor.humidity()
    return temp, hum
```

Donde en esta lo que hará será recibir todos los datos del sensor para devolverlos en dos variables para ser utilizados en la pantalla y el servidor web.

En el caso de la pantalla, importamos dentro del archivo pantalla las librerías y módulos necesarios.

```
from machine import Pin, SoftI2C
import ssd1306
from time import sleep
import sensor
```

Para seguidamente definir una función que se le llamará proyectar, utilizada para proyectar los datos del sensor en la pantalla oled.
Para que funcione la pantalla utilizamos una librería ssd1306 ya creada, tomada de:
https://www.esploradores.com/oled_ssd1306/

Ahora dentro de la función proyectar agregamos

```
i2c = SoftI2C(scl=Pin(22), sda=Pin(21))

anchoPantalla = 128
altoPantalla = 64
# Asignando los valores de la resolucion de la pantalla oled
oled = ssd1306.SSD1306_I2C(anchoPantalla, altoPantalla, i2c)
```

Donde asignamos los pines de la pantalla

Y dentro de un ciclo while True se van mostrando constantemente los valores en la pantalla oled y también si sobrepasan los rangos propuestos se encenderá el led interno del dispositivo Esp 32

```
while True:
    oled.fill(0)
    sleep(1)
    temp, hum =sensor.recibir()
#encender led si supera humedad
    if (temp < 15 or temp > 20) or (hum < 45 or hum > 60) :
        led = Pin(2, Pin.OUT)
        led.on()
        sleep(0.1)
        led.off()
# convertir valores en texto
    temp = str(temp)
    hum = str(hum)

#imprimir en pantalla valores
    oled.text("Temp : " + temp + " C", 20, 8)
    oled.text("Hum  : " + hum + " %", 20, 32)

    oled.show()
```

Lo siguiente es crear la conexión wifi dentro del archivo ya creado wifi.py, importando las librerías

```
import network
from time import sleep
```

Y definimos la función conectar para acceder a la red

```
def conectar():
        """Conectar a la red wifi"""
        wifi = network.WLAN(network.STA_IF)
        ssid = "red2"
        contraseña = "12345678"
        ssid2 = "BATMAN"
        contraseña2 = "BATSH8213gh"

      #Si no hay conexión, activar wifi y conectar
      #con el nombre de la red y su contraseña
        if not wifi.isconnected():
            wifi.active(True)
            wifi.connect(ssid, contraseña)
            while not wifi.isconnected():
                print(".", end="")
                sleep(1)
        print("Conexión establecida con " + ssid)
        print(wifi.ifconfig())
```

Para el servidor web como siempre importamos las librerías necesarias como también los módulos creados anteriormente

```
import network
from time import sleep
import sensor
import socket
import wifi
import urequests
```

Creamos una función llamada en este caso web

```
def web():
    """Servidor web para mostrar los datos de los sensores de manera local"""
    wifi.conectar()
    #esto para especificar una ip y se deja en blanco para que la tome
    #automaticamente y se le asigna un puerto
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('', 80))
    s.listen(5)
```

Donde usaremos los sockets para asignar ip y puertos
Y dentro de ella un ciclo while True, donde:

```
#Recibir datos del sensor dht11
temp, hum =sensor.recibir()
sleep(2)

#Enviar dato del sensor de temperatura a la api de thingspeak
apiTemp="https://api.thingspeak.com/update?api_key=38YCW8L4OCXQGK6H&field1="
apiTemp = apiTemp + str(temp)
r1 = urequests.get(apiTemp)
print(r1.json())

#Enviar dato del sensor de humedad a la api de thingspeak
apiHum="https://api.thingspeak.com/update?api_key=ORDLJDO3MN6OX9Q6&field1="
apiHum = apiHum + str(hum)
r2 = urequests.get(apiHum)
print(r2.json())
```

Recibamos los datos del sensor y los enviemos al api de thingspeak para poder graficar los resultados obtenidos a cada momento que el dispositivo se encuentre encendido.

También guardar la conexión del dispositivo

```
#guardar la conexión y dirección del dispositivo
conn, addr = s.accept()
print("Nueva conexión desde %s" % str(addr))

request=conn.recv(1024)
print("")
print("Solicitud %s" % str(request))

request = str(request)
```

Además condiciones de si sobrepasa los rangos requeridos como en el caso de los mostrados en la pantalla oled.

```
#condiciones de temperatura si sobrepasan el rango requerido
if (temp < 15 or temp > 20):
    t = "Está fuera del rango"
else:
    t = "Está dentro del rango"

if (hum < 45 or hum > 60):
    h = "Está fuera del rango"
else:
    h = "Está dentro del rango"
```

Posteriormente creamos la web por medio de un sencillo código HTML donde se traerán los datos que se almacenaron en thingspeak y que anteriormente fueron recibidos desde el sensor, y mostrar además si dichos valores están dentro o fuera del rango requerido

```
#Pagina web para mostrar los datos de los sensores y thingspeak
pagina ="""<!DOCTYPE HTML>
        <html>
        <head>
            <meta charset="utf-8">
        </head>
          <body style="background-color:#BCDCF1;">

            <center>
            <h1>Medición y monitorización </h1>
            <iframe src="https://thingspeak.com/channels/1456301/charts/1?bgcolor=%23ffffff&color=%2
            <h2 style="color: #6F40A8;">Temperatura es """ + str(temp) +""" </h2>
            <h2>"""+t+"""</h2>

            <br>
            <iframe src="https://thingspeak.com/channels/1456302/charts/1?bgcolor=%23ffffff&color=%2
            <h2 style="color: #6F40A8;">Humedad es """ + str(hum) + """ </h2>
            <h2>"""+h+"""</h2>
             </center>
              </body>
        </html>"""
```

Finalmente mostramos cerramos la página web para reinicio.

```
#mostramos pagina web y cerramos la pagina web
conn.send('HTTP/1.1 200 OK\n')
conn.send('Content-Type: text/html\n')
conn.send('Connection: close\n\n')
conn.sendall(pagina)
conn.close()
```

Conclusión

Del presente documento se puede concluir, que por medio del uso y aplicación de un dispositivo Esp 32 combinado con un sensor de temperatura y humedad relativa, se pueden monitorear y controlar como esta el ambiente en lugares donde se requiere mantener a unos rangos necesarios para su correcta utilización y durabilidad a lo largo del tiempo, este proyecto enriquecedor se podrá aplicar donde las condiciones no estén dadas y se necesite de la presencia de personal que constantemente esté verificando y/o monitoreando dichos lugares de almacenamiento de información, donde, cuando se lleve a cabo la implementación, logre facilitar su proceso

Referencias

[1] Santos, S. (2021, June 30). MicroPython: OLED Display with ESP32 and ESP8266. Random Nerd Tutorials. https://randomnerdtutorials.com/micropython-oled-display-esp32-esp8266/

[2] No, D. (2019, September 30). Dani No. ESPloradores -Ayuda y consejos para DIY makers-. https://www.esploradores.com/oled_ssd1306/

[3] Overview — MicroPython 1.16 documentation. (n.d.). Micropython.Org. Retrieved July 29, 2021, from https://docs.micropython.org/en/latest/