Documentație Proiect

pentru

Stație Meteo Portabilă

Version 1.0 approved

Prepared by

Bran Ioana-Andreea: ioana-andreea.bran@student.tuiasi.ro,

Zabara Sonia: sonia.zabara@student.tuiasi.ro

Universitatea Tehnică Gheorghe Asachi Iași, Facultatea de Automatică și Calculatoare, Calculatoare și Tehnologia Informației, grupa 1310A

2025

1. Motivația alegerii temei

Într-un context în care informațiile despre condițiile meteo sunt esențiale în viața de zi cu zi, dezvoltarea unei stații meteo portabile reprezintă o oportunitate practică de a înțelege, construi și personaliza un sistem funcțional de monitorizare ambientală. Proiectul de față are ca scop realizarea unei aplicații meteo autonome, bazată pe microcontrolerul Raspberry Pi Pico 2W, care integrează senzori de temperatură și umiditate, un ecran LED, LED-uri de stare și un difuzor pentru semnalizare sonoră.

Motivația principală din spatele acestui proiect este una atât educațională cât și practică, vizând aprofundarea cunoștințelor în domenii precum electronică digitală, programarea microcontrolerelor (în MicroPython), comunicația între dispozitive (I2C, GPIO), dar și posibilitatea de a obține date meteo locale, în timp real, fără a depinde de o conexiune permanentă la internet sau servicii externe. Această aplicație este utilă pentru automatizări casnice sau pentru utilizatorii preocupați de confidențialitatea datelor personale. Dispozitivul poate afișa intuitiv starea vremii prin culori și sunete (ex: roșu + alertă sonoră = temperaturi ridicate), ceea ce îl face prietenos și ușor de înțeles pentru toate categoriile de utilizatori, inclusiv copii sau persoane cu deficiențe de vedere.

2. Descriere

Proiectul constă în realizarea unei stații meteo portabile bazate pe microcontrolerul Raspberry Pi Pico 2W, capabilă să măsoare în timp real temperatura și umiditatea din mediul înconjurător, folosind un senzor digital (DHT11). Datele sunt afișate local pe un ecran LCD (2004A), însoțite de semnalizări vizuale cu LED RGB (care își schimbă culoarea în funcție de temperatură) și alerte sonore generate de un buzzer activ pentru evidențierea unor condiții meteo speciale (temperaturi extreme sau umiditate ridicată).

Pe lângă funcționarea autonomă locală, stația meteo include și o componentă de interfață web, implementată prin capabilitățile Wi-Fi ale modulului Raspberry Pi Pico W. Microcontrolerul funcționează ca un server HTTP, generând o pagină web proprie accesibilă din browserul oricărui dispozitiv conectat la aceeași rețea locală. Această pagină afișează în timp real valorile temperaturii și umidității și include două butoane care permit actualizarea manuală a acestor valori (/temp și /humidity).

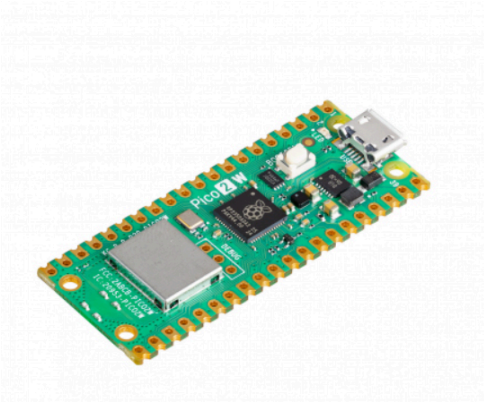
Pagina web este simplă, dar eficientă, având un design intuitiv realizat cu HTML și stilizare CSS de bază, și se actualizează automat la fiecare 5 secunde pentru a menține

datele afișate în timp real. Interacțiunea dintre utilizator și microcontroler se face prin cereri HTTP directe, fără a fi nevoie de servicii externe sau conexiune la internet.

Dispozitivul este portabil, ușor de utilizat, alimentabil prin USB și combină funcționalitatea hardware (afișaj, LED RGB, buzzer) cu accesibilitatea unei interfețe web, fiind ideal pentru aplicații educaționale, demonstrative sau pentru monitorizarea ambientală locală, în spații precum camere de locuit, sere, birouri sau laboratoare.

Componente:

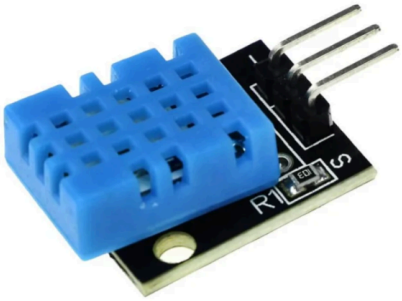
- Raspberry Pi Pico 2W



- LCD 2004A

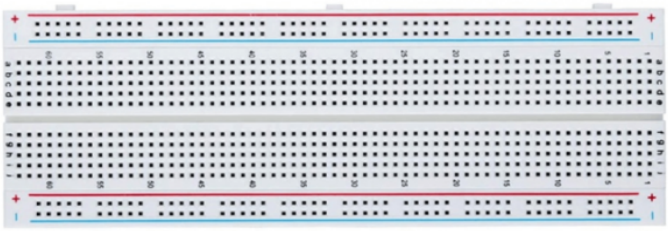
- LED RGB

- Senzor de temperatură și umiditate DHT11



- Cabluri de contact tata-tata, tata-mama

- Breadboard



- Buzzer activ

3. Importanța în domeniul Embedded System-SI

Integrarea diverselor componente: Proiectul oferă o oportunitate excelentă pentru învățarea și aplicarea practică a interfețelor hardware-software prin conectarea mai multor componente: senzor de temperatură și umiditate (DHT11), afișaj LCD (prin I2C), LED RGB controlat prin PWM, buzzer activ și interfață Wi-Fi. Fiecare componentă este controlată și sincronizată software într-un sistem embedded complet funcțional.

Programare în MicroPython: Proiectul utilizează MicroPython, un limbaj interpretat ușor de folosit în mediul embedded. Studenții învață despre structura programelor embedded, gestionarea porturilor GPIO, PWM, comunicația I2C și gestionarea conexiunilor de rețea prin module precum socket și network. De asemenea, este utilizată o bibliotecă personalizată (h2RGB.py) pentru conversia temperaturii în culori RGB, simulând un gradient de culoare în funcție de temperatură.

Control și interacțiune prin rețea: Stația meteo funcționează și ca un server web local, permițând utilizatorului să acceseze în timp real datele de mediu dintr-un browser, fără a fi necesar un server extern. Se folosesc cereri HTTP simple, iar pagina web permite atât afișarea automată a datelor, cât și declanșarea manuală a actualizărilor prin butoane.

Feedback vizual și auditiv adaptiv: LED-ul RGB oferă o reprezentare vizuală intuitivă a temperaturii, utilizând un gradient de culoare generat dinamic (de la albastru la roșu, în funcție de temperatură, incluzând temperaturi negative). Buzzerul oferă feedback sonor pentru condiții meteo speciale (temperaturi extreme), consolidând legătura dintre semnale digitale și percepția umană.

Consum redus de energie și portabilitate: Dispozitivul este optimizat pentru rulare continuă și alimentare USB de joasă putere. Codul este structurat astfel încât senzorul este interogat doar la

nevoie, iar afișajul este actualizat periodic, ceea ce contribuie la eficiența energetică și durata de viață a sistemului.

4. Analiza Design

Cerințe Funcționale:

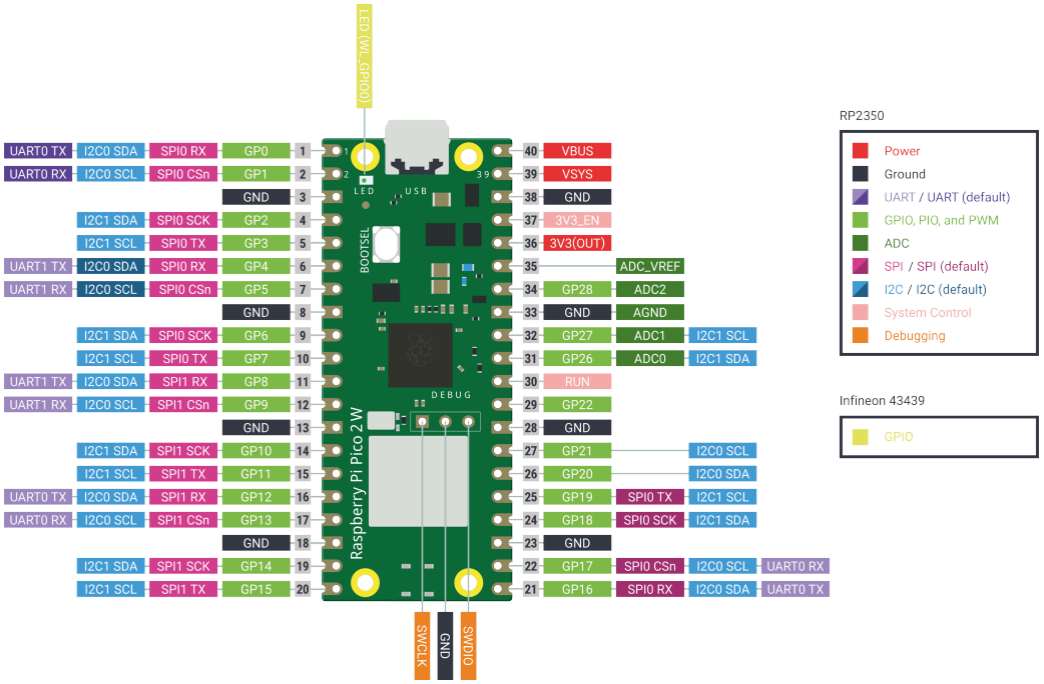
- Măsurarea temperaturii și umidității cu senzorul DHT11.
- Afișarea în timp real a valorilor pe un **LCD 2004A** cu interfață I2C.

- Afișarea valorilor și pe o **pagină web locală** găzduită pe microcontroler, accesibilă prin Wi-Fi.
- Notificare **vizuală** prin LED RGB cu culori dinamice (gradient între albastru – rece și roșu – cald) în funcție de temperatură, folosind o bibliotecă personalizată (h2RGB.py)
- Notificare **auditivă** prin buzzer activ în caz de temperaturi extreme (ex. sub 0°C sau peste 30°C).
- Interacțiune cu utilizatorul prin **butoane web** pentru a actualiza manual temperatura sau umiditatea.
- Funcționare continuă, cu actualizări automate periodice (la fiecare 5 secunde).
- Cod eficient, modular și optimizat pentru dispozitive cu resurse limitate.

Configurare Hardware

- Conectează Raspberry Pi Pico 2W la breadboard.
- Conectează senzorul DHT11 la Raspberry Pi Pico conform diagramei de conexiuni.
- Conectează LED-ul la Raspberry Pi Pico conform diagramei de conexiuni.
- Conectează buzzer -ul activ la Raspberry Pi Pico conform diagramei de conexiuni.
- Conectează LCD -ul la Raspberry Pi Pico conform diagramei de conexiuni.

Pico 2w Pinout



Configurare Software

Descarcă și instalează firmware-ul **MicroPython** corespunzător pentru Raspberry Pi Pico W.

În Thonny sau alt IDE compatibil:

- Creează și încarcă fișierele:

  main.py– codul principal al proiectului

  secrets.py– fișierul cu rețeaua Wi-Fi (SSID și parolă)

  h2RGB.py – biblioteca care mapează temperatura în culori RGB

- Verifică adresa IP afișată în consolă după conectare și acceseaz-o în browser pentru a vedea interfața web.
- Testează senzorul, LED-ul RGB, buzzer-ul și afișajul LCD pentru a confirma funcționarea corectă.
- Asigură-te că rețeaua Wi-Fi locală este stabilă și Pico 2W are semnal bun.

5. Competențe dobândite după realizarea proiectului

1. **Introducere în Microcontroller-e și Programarea Embedded**

    - **Raspberry Pi Pico**: Cunoștințe despre folosirea unui microcontroller, inclusiv instalarea și configurarea MicroPython pe Raspberry Pi Pico.
    - **Programare Embedded**: Cunoștințe despre conceptele de bază ale programării embedded, inclusiv cum să scrii cod care interacționează direct cu hardware-ul.

2. **Protocolul de Comunicare I2C**

    - **I2C Protocol**: Cunoștințe despre cum funcționează protocolul de comunicare I2C, care este utilizat pentru a conecta senzorii și alte dispozitive la microcontroller.
    - **Configurare și Scanare I2C**: Cunoștințe despre cum să configurezi și să scanezi magistrala I2C pentru a detecta dispozitivele conectate.

3. **Lucrul cu senzori de mediu**

    - **Senzor DHT11:** Cunoștințe despre cum să utilizezi senzorul DHT pentru a măsura temperatura și umiditatea. De asemenea, vei învăța despre principiile de funcționare ale acestui tip de senzor.
    - **Interfațarea cu Senzorii:** Cunoștințe despre cum să citești date de la senzori și cum să procesezi aceste date în cod.

4. **Afișarea datelor pe un LCD**

    - **LCD 2004A:** Cunoștințe despre cum să controlezi un LCD folosind microcontrollerul, inclusiv cum să inițializezi ecranul, să afișezi texte și să actualizezi afișajul.

5. **Electronică de Bază și Conectarea Componentelor**

    - **Breadboard și Conectarea Componentelor**: Cunoștințe despre cum să folosești un breadboard pentru a face conexiunile între microcontroller, senzor și display.
    - **Diagrame de Conexiuni**: Cunoștințe despre cum să interpretezi și să urmezi diagramele de conexiuni pentru a asambla circuitele corect.

6. **Abilități de Proiectare și Implementare**

- **Designul Sistemului:** Cunoștințe despre cum să concepi un sistem electronic complet, de la identificarea cerințelor până la implementarea și testarea finală.
- **Iterație și Îmbunătățire:** Cunoștințe despre cum să îmbunătățești un proiect prin iterație, adăugând noi funcționalități sau optimizând codul existent.

7. **LED RGB cu mapare dinamică:**

    - Ai învățat să controlezi un LED RGB folosind semnale PWM și să mapezi valori de temperatură într-un **spectru de culori** folosind o bibliotecă personalizată (h2RGB.py). Această mapare acoperă temperaturi de la **–100°C la +100°C**, oferind o reprezentare vizuală intuitivă și fluidă (ex: tonuri de albastru pentru temperaturi negative, verde pentru valori normale și roșu pentru temperaturi ridicate).

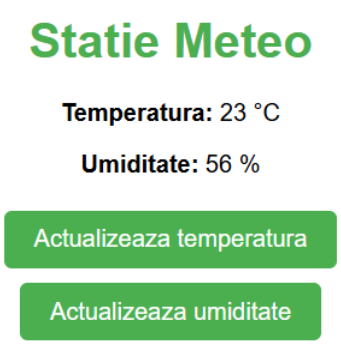
8. **Interfață Web Integrată:**
    - Ai învățat cum să configurezi Raspberry Pi Pico W pentru a acționa ca un **server HTTP**. Acesta generează și răspunde cu o **pagină HTML dinamică**, ce afișează în browser temperatura și umiditatea în timp real.

9. **Comunicare Client–Server:**
   Prin gestionarea cererilor HTTP, ai implementat o **formă simplă de interactivitate web**, permițând utilizatorului să actualizeze datele printr-un click, fără a fi nevoie de reîmprospătarea manuală a paginii.

10. **Sincronizare cu Afișajul Fizic:**
    Ai integrat **afișajul fizic (LCD) cu pagina web**, astfel încât datele să fie actualizate simultan, atât în interfața locală cât și în browser, consolidând conceptele de **interfațare multi-platformă embedded**.

6. Interfața web

- Afișează temperatura și umiditatea în timp real.
- Include două butoane: „Actualizează Temperatură” și „Actualizează Umiditate”.
- Actualizare automată la 5 secunde.

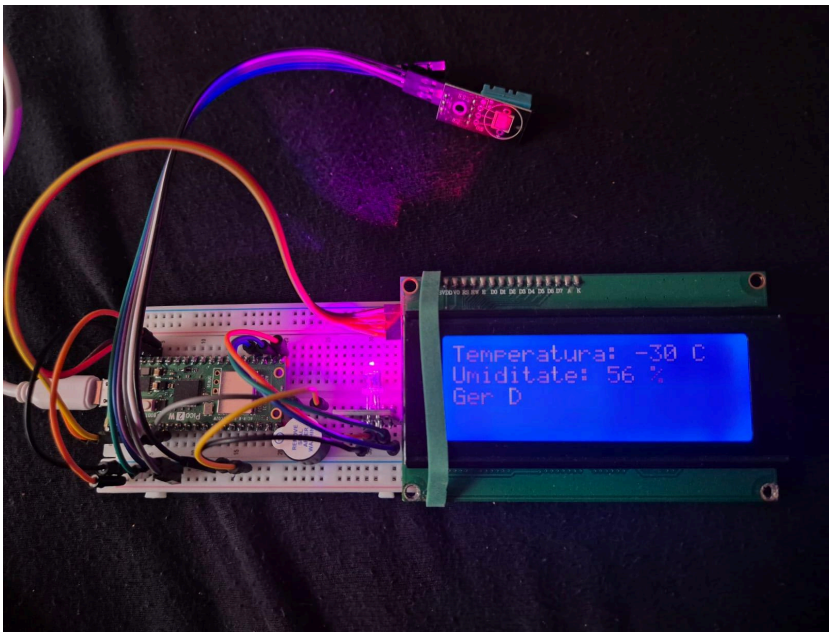


7. Testare și rezultate

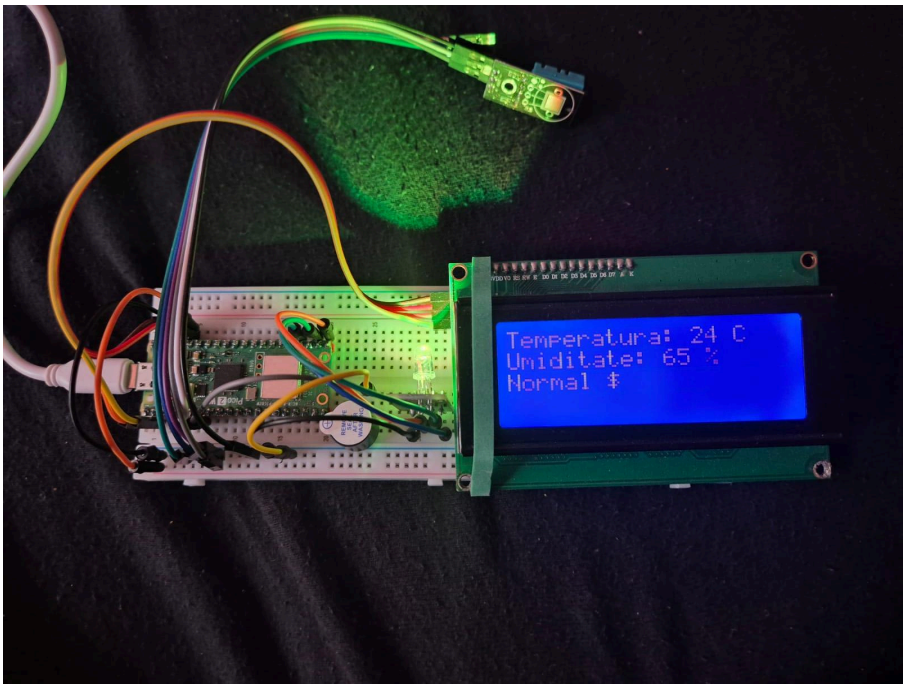
Testarea s-a realizat în mai multe etape:

- **Temperatură simulată (manual):**

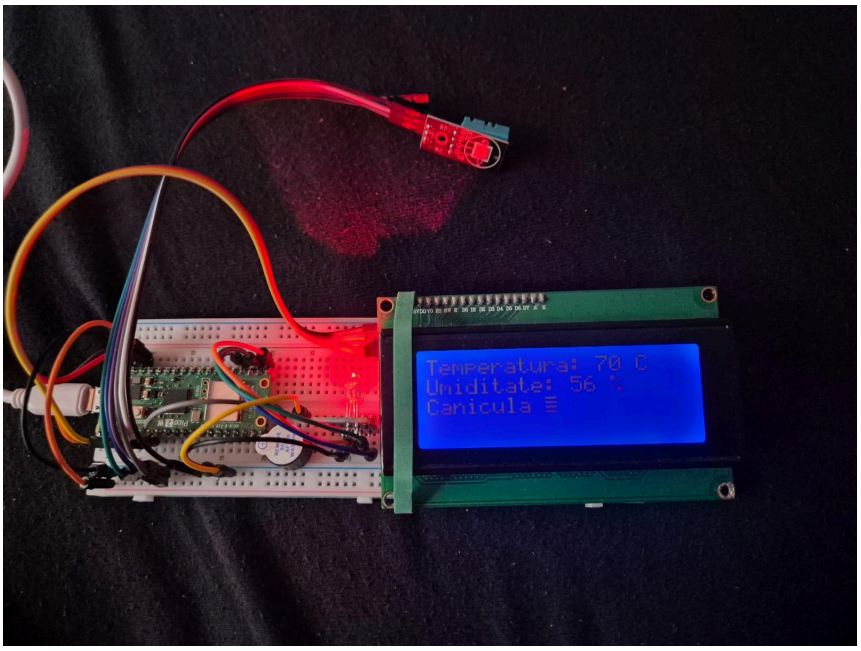
- Sub 0°C – LED-ul RGB a afișat o nuanță de mov-albastrui, iar buzzer -ul a emis un sunet de alerta



- Între 0°C și 30°C – LED-ul a trecut treptat prin verde.



- Peste 30°C – LED-ul a devenit roșu, iar buzzerul a emis un sunet de alertă.



- **Umiditate simulată:**
    - Umiditatea ridicată (>80%) a fost marcată de un sunet de avertizare.
- **Funcționare server web:**
    - Conectarea la pagina web a fost stabilă.
    - Datele s-au actualizat constant fără blocaje.
    - Butoanele /temp și /humidity au declanșat corect actualizarea valorilor.

8. Concluzii

Realizarea acestei stații meteo portabile a permis aplicarea practică a conceptelor teoretice studiate în domeniul sistemelor embedded. Proiectul a combinat programarea microcontrolerelor, interfațarea cu senzori, generarea de semnale PWM, afișarea datelor pe LCD și comunicarea prin rețea locală.

Punctele forte ale proiectului:

- Portabilitate și consum redus de energie.
- Interfață intuitivă, atât fizică (LED/buzzer/LCD), cât și digitală (web).
- Design modular, ușor de extins.

Provocări întâmpinate:

- Stabilizarea conexiunii Wi-Fi.
- Reglarea curbelor RGB în funcție de temperatură pentru a face gradientul intuitiv.
- Sincronizarea datelor între LCD și pagina web.

9. Bibliografie

Raspberry Pi Pico 2W Docs:
https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html
MicroPython: https://micropython.org/
DHT11 Datasheet: https://components101.com/sensors/dht11-temperature-sensor
Paul McWorther: Raspberry Pi Pico W LESSON 1: Write Your First Program for Absolu…

10. Cod sursa

GitHub: https://github.com/ioana-and30/MicroStatie-Meteo

main.py

```
import network
import socket
import time
from machine import I2C, Pin, PWM
from DIYables_MicroPython_LCD_I2C import LCD_I2C
import dht
import h2RGB
import secrets
from machine import Timer

def interruption_handler(timer):
    global sensor
    sensor.measure()
    temp = sensor.temperature()
    humidity = sensor.humidity()
    actualizeaza_afisaj(temp, humidity)
```

```
soft_timer = Timer(mode=Timer.PERIODIC, period=1000,
callback=interruption_handler)

# LCD setup
I2C_ADDR = 0x27
LCD_COLS = 20
LCD_ROWS = 4
i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)
lcd = LCD_I2C(i2c, I2C_ADDR, LCD_ROWS, LCD_COLS)
lcd.backlight_on()
lcd.clear()

# Sensor
sensor = dht.DHT11(Pin(4))

# RGB LED
redLED = PWM(Pin(16))
greenLED = PWM(Pin(17))
blueLED = PWM(Pin(18))
for led in [redLED, greenLED, blueLED]:
    led.freq(1000)
    led.duty_u16(0)

# Buzzer
buzzer = Pin(5, Pin.OUT)

# WiFi
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(secrets.SSID, secrets.PASSWORD)
while not wlan.isconnected():
    time.sleep(1)

# HTTP server
addr = socket.getaddrinfo('0.0.0.0', 12335)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)

print("Server pornit la:", wlan.ifconfig()[0])

def set_rgb(r, g, b):
    redLED.duty_u16(int(r * 65535))
    greenLED.duty_u16(int(g * 65535))
    blueLED.duty_u16(int(b * 65535))

def actualizeaza_afisaj(temp, humidity):
    lcd.clear()
    lcd.set_cursor(0, 0)
    lcd.print("Temperatura: {} C".format(temp))
    lcd.set_cursor(0, 1)
    lcd.print("Umiditate: {} %".format(humidity))

    if temp > 30:
```

```
            lcd.set_cursor(0, 2)
            lcd.print("Canicula 🔆")
            buzzer.on()
            time.sleep(1)
            buzzer.off()
        elif temp < 0:
            lcd.set_cursor(0, 2)
            lcd.print("Ger ❄")
            buzzer.on()
            time.sleep(1)
            buzzer.off()
        else:
            lcd.set_cursor(0, 2)
            lcd.print("Normal 🌤️")

    try:
        sensor.measure()
        temp = sensor.temperature()
        humidity = sensor.humidity()

        r, g, b = h2RGB.getRGB(temp * 3)
        set_rgb(r, g, b)

        actualizeaza_afisaj(temp, humidity)

    except Exception as e:
        lcd.clear()
        lcd.set_cursor(0, 0)
        lcd.print("Eroare senzor")

    while True:
        client, addr = s.accept()
        request = client.recv(1024).decode()
        print("Cerere de la:", addr)

        temp = "-"
        humidity = "-"
        sensor.measure()
        temp = sensor.temperature()
        humidity = sensor.humidity()
        actualizeaza_afisaj(temp, humidity)
        if '/temp' in request or '/humidity' in request or '/' in request:
            try:
                sensor.measure()
                temp = sensor.temperature()
                humidity = sensor.humidity()

                r, g, b = h2RGB.getRGB(temp * 3)
                set_rgb(r, g, b)

                actualizeaza_afisaj(temp, humidity)

            except Exception as e:
                temp = "Err"
```

```
            humidity = "Err"
            lcd.clear()
            lcd.set_cursor(0, 0)
            lcd.print("Eroare senzor")

        response = """\
HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n
<!DOCTYPE html>
<html>
<head>
    <title>Statie meteo</title>
    <style>
        body {{ font-family: Arial; text-align: center; margin-top: 50px; }}
        h1 {{ color: #4CAF50; }}
        button {{
            background-color: #4CAF50; color: white;
            padding: 10px 20px; font-size: 16px;
            border: none; border-radius: 5px;
            cursor: pointer; margin: 5px;
        }}
        button:hover {{ background-color: #45a049; }}
    </style>
</head>
<body>
    <h1>Statie Meteo</h1>
    <p><strong>Temperatura:</strong> {} &deg;C</p>
    <p><strong>Umiditate:</strong> {} %</p>
    <form action="/temp"><button>Actualizeaza temperatura</button></form>
    <form action="/humidity"><button>Actualizeaza umiditate</button></form>
</body>
</html>
""".format(temp, humidity)

        client.send(response)
        client.close()
```

secrets.py

```
SSID="Galaxy S24+ 6A40"
PASSWORD="parola"
```

h2RGB.py

```
def getRGB(temp):
    # Mapează temperatura din intervalul [-100, 100] în intervalul [0, 360]
grade
    temp_min = -100
    temp_max = 100
    temp = max(temp_min, min(temp, temp_max))  # limităm între -100 și 100
    deg = int((temp - temp_min) * 360 / (temp_max - temp_min))  # scala pe
360°
```

```
m = 1 / 60
if deg >= 0 and deg < 60:
    R = 1
    G = 0
    B = m * deg
elif deg < 120:
    R = 1 - m * (deg - 60)
    G = 0
    B = 1
elif deg < 180:
    R = 0
    G = m * (deg - 120)
    B = 1
elif deg < 240:
    R = 0
    G = 1
    B = 1 - m * (deg - 180)
elif deg < 300:
    R = m * (deg - 240)
    G = 1
    B = 0
else:
    R = 1
    G = 1 - m * (deg - 300)
    B = 0

return R, G, B
```