Partie VIII

Ajout de la sonde BMP280 aux scripts

Nous allons modifier nos scripts existants pour tester la sonde bmp280 aussi bien dans le **terminal** que sur le **serveur web (flask)**.

Modification du script station.py

```
#Importation
import adafruit_dht
import adafruit_bmp280
import board
import busio
import time
import math
from datetime import datetime

#Déclaration du capteur
dhtDevice = adafruit_dht.DHT22(board.D4)

i2c = busio.I2C(board.SCL, board.SDA)

bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c, address=0x76)

#Couleurs
RED = "\033[91m"
GREEN = "\033[92m"
YELLOW = "\033[93m"
```

```
BLUE = "\033[94m"
MAGENTA = "\033[95m"
CYAN = "\033[96m"
RESET = "\033[0m"

#Calcul du point de rosée
def calculer_point_de_rosee(temperature, humidity):
    # Formule pour calculer le point de rosée
    alpha = 17.27
    beta = 237.7
    gamma = (alpha * temperature) / (beta + temperature) + math.log(humidity / 100.0)
    point_de_rosee = (beta * gamma) / (alpha - gamma)
    return point_de_rosee

#Calcul de l'humidex
def calculer_humidex(temperature, point_de_rosee):
    # Formule pour calculer l'humidex
    humidex = temperature + (5/9) * (6.11 * math.exp(5417.7530 * ((1/273.16) - (1/273.15 +
point_de_rosee))) - 10)
    return humidex

#Boucle et affichage
while True:
    humidity = dhtDevice.humidity
    temperature = dhtDevice.temperature
    pression = bmp280.pressure
    if humidity is not None and temperature is not None:
        now = datetime.now()
        date_heure = now.strftime("%d-%m-%Y %H:%M:%S")
        point_de_rosee = calculer_point_de_rosee(temperature, humidity)
        humidex = calculer_humidex(temperature, point_de_rosee)
        print(f"{BLUE}Date et heure:{RESET} {date_heure}")
        print(f"{GREEN}Température:{RESET} {round(temperature, 1)}°C")
        print(f"{YELLOW}Humidité:{RESET} {round(humidity, 1)}%")
        print(f"{RED}Point de rosée:{RESET} {round(point_de_rosee, 1)}°C")
        print(f"{MAGENTA}Humidex:{RESET} {round(humidex, 1)}")
        print(f"{CYAN}Pression atmosphérique:{RESET} {round(pression, 2)} hPa")
        print("----")
    else:
        print("Échec de la lecture du capteur")
```

```
    #Pause de 20 secondes
    time.sleep(20)
```

Modification du script capteur.py

```
#importations
import adafruit_dht
import adafruit_bmp280
import board
import busio #Bibliothèque permettant d'ouvrir une communication sur un bus matériel
import math
from datetime import datetime

#Initialisation du bus I2C
i2c = busio.I2C(board.SCL, board.SDA)

#Initialisation de la sonde
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c, address=0x76)

def lire_donnees_capteur():
    try:
        dhtDevice = adafruit_dht.DHT22(board.D4)
        humidity = dhtDevice.humidity
        temperature = dhtDevice.temperature
        dhtDevice.exit()
        if humidity is not None and temperature is not None:
            return round(humidity, 1), round(temperature, 1)
        else:
            return None, None

    except RuntimeError as error:
        print("Erreur de lecture :", error)
        return None, None

    except Exception as error:
        dhtDevice.exit() #Libérer le GPIO même en cas de crash
        raise error #Le programme crashe, car c'est une erreur critique
```

```
def calculer_point_de_rosee(temperature, humidity):
    #Formule pour calculer le point de rosée
    alpha = 17.27
    beta = 237.7
    gamma = (alpha * temperature) / (beta + temperature) + math.log(humidity / 100)
    point_de_rosee = (beta * gamma) / (alpha - gamma)
    return round(point_de_rosee, 1)

def calculer_humidex(temperature, point_de_rosee):
    #Formule pour calculer l'indice humidex
    humidex = temperature + (5/9) * (6.11 * math.exp(5417.7530 * ((1/273.16) - (1/273.15 +
point_de_rosee))) - 10)
    return round(humidex, 1)

def lire_pression():
    try:
        pression = bmp280.pressure
        return round(pression, 1)
    except Exception as error:
        print("Erreur de lecture BMP280 :", error)
        return None

def recuperer_date_heure():
    return datetime.now().strftime("%d-%m-%Y %H:%M:%S")
```

Modification du script app.py

```
#importations
from flask import Flask, render_template_string
from capteur import (
    lire_donnees_capteur,
    calculer_point_de_rosee,
    calculer_humidex,
    recuperer_date_heure,
    lire_pression
)

# Définition de l'application Flask
app = Flask(__name__)
```

```
@app.route('/')
def index():
    humidity, temperature = lire_donnees_capteur()
    if humidity is not None and temperature is not None:
        point_de_rosee = calculer_point_de_rosee(temperature, humidity)
        humidex = calculer_humidex(temperature, point_de_rosee)
        date_heure = recuperer_date_heure()
        pression = lire_pression()

        html = f"""
        <h1>Données météo locales</h1>
        <ul>
            <li>Date et heure : {date_heure}</li>
            <li>Température : {temperature} °C</li>
            <li>Humidité : {humidity} %</li>
            <li>Point de rosée : {point_de_rosee} °C</li>
            <li>Humidex : {humidex}</li>
            <li>Pression atmosphérique : {pression} hPa</li>
        </ul>
        """
    else:
        html = "<p>Erreur de lecture du capteur.</p>"

    return render_template_string(html)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Révision #3
Créé 2025-08-11 09:10:24 UTC
Mis à jour 2025-08-11 09:20:44 UTC