import logging

import board

import busio

import adafruit\_ads1x15.ads1115 as ADS

from adafruit\_ads1x15.analog\_in import AnalogIn

import time

from flask import Flask, request, jsonify, render\_template

from flask\_cors import CORS

import RPi.GPIO as GPIO

import requests

import Adafruit\_DHT

import subprocess

from datetime import datetime, timedelta

# Network credentials and host

SSID = "YourWfifi"

PASSWORD = "Password"

HOST = "YourAppHost"

# Initialize Flask app

app = Flask(\_\_name\_\_)

CORS(app)

GPIO.cleanup()

# GPIO configuration

DHT\_PIN = 5 # GPIO5 (BCM)

PIN\_1 = 17 # GPIO17

PIN\_2 = 27 # GPIO27

GPIO.setmode(GPIO.BCM)

GPIO.setup(PIN\_1, GPIO.OUT)

GPIO.setup(PIN\_2, GPIO.OUT)

# DHT sensor configuration

DHT\_SENSOR = Adafruit\_DHT.DHT22

# Initialize the I2C interface

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

ads = ADS.ADS1115(i2c)

# Calibration values (replace with actual measurements for each sensor)

wet\_voltage\_A = 1.48 # Fully submerged in water (Sensor A)

dry\_voltage\_A = 3.3 # Completely dry (Sensor A)

wet\_voltage\_B = 2.3 # Fully submerged in water (Sensor B)

dry\_voltage\_B = 3.5 # Completely dry (Sensor B)

# Last watering time

last\_watering\_time\_A = datetime.now() - timedelta(days=3) # Initially set to 3 days ago

last\_watering\_time\_B = datetime.now() - timedelta(days=3) # Initially set to 3 days ago

def convert\_to\_percent(voltage, wet\_voltage, dry\_voltage):

voltage = max(min(voltage, dry\_voltage), wet\_voltage)

percent = int(round((dry\_voltage - voltage) / (dry\_voltage - wet\_voltage) \* 100))

return max(min(percent, 100), 0)

def read\_moisture(pin, wet\_voltage, dry\_voltage):

chan = AnalogIn(ads, pin)

logging.info(f"Moisture Voltage: {chan.voltage}")

moisture\_percent = convert\_to\_percent(chan.voltage, wet\_voltage, dry\_voltage)

return chan.voltage, moisture\_percent

def read\_dht\_sensor():

humidity, temperature = Adafruit\_DHT.read\_retry(DHT\_SENSOR, DHT\_PIN)

if humidity is None or temperature is None:

return 0, 0 # Return 0 if failed to read

return humidity, temperature

def send\_data\_to\_server(sensor\_number, data):

url = f"{HOST}/sensor{sensor\_number}?token={TOKEN}&sensor{sensor\_number}={data}"

try:

response = requests.get(url)

if response.status\_code == 200:

print(f"Data sent to server: {data}")

else:

print(f"Failed to send data to server: {response.status\_code}")

except requests.exceptions.RequestException as e:

print(f"Request to server failed: {e}")

# Flask routes

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/sensors', methods=['GET'])

def get\_sensor\_data():

if request.args.get('token') == TOKEN:

humidity, temperature = read\_dht\_sensor()

voltage\_A, moisture\_percent\_A = read\_moisture(ADS.P0, wet\_voltage\_A, dry\_voltage\_A)

voltage\_B, moisture\_percent\_B = read\_moisture(ADS.P1, wet\_voltage\_B, dry\_voltage\_B)

return jsonify({

'temperature': temperature,

'humidity': humidity,

'moisture\_voltage\_A': voltage\_A,

'moisture\_percent\_A': moisture\_percent\_A,

'moisture\_voltage\_B': voltage\_B,

'moisture\_percent\_B': moisture\_percent\_B

})

else:

return "Unauthorized", 403

@app.route('/4/on', methods=['GET'])

def turn\_on\_1():

if request.args.get('token') == TOKEN:

GPIO.output(PIN\_1, GPIO.HIGH)

return f"GPIO {PIN\_1} turned on", 200

else:

return "Unauthorized", 403

@app.route('/4/off', methods=['GET'])

def turn\_off\_1():

if request.args.get('token') == TOKEN:

GPIO.output(PIN\_1, GPIO.LOW)

return f"GPIO {PIN\_1} turned off", 200

else:

return "Unauthorized", 403

@app.route('/5/on', methods=['GET'])

def turn\_on\_2():

if request.args.get('token') == TOKEN:

GPIO.output(PIN\_2, GPIO.HIGH)

return f"GPIO {PIN\_2} turned on", 200

else:

return "Unauthorized", 403

@app.route('/5/off', methods=['GET'])

def turn\_off\_2():

if request.args.get('token') == TOKEN:

GPIO.output(PIN\_2, GPIO.LOW)

return f"GPIO {PIN\_2} turned off", 200

else:

return "Unauthorized", 403

@app.route('/reboot', methods=['GET'])

def reboot():

if request.args.get('token') == TOKEN:

try:

subprocess.run(['sudo', 'reboot'], check=True)

return "Rebooting...", 200

except subprocess.CalledProcessError as e:

return str(e), 500

else:

return "Unauthorized", 403

@app.route('/shutdown', methods=['GET'])

def shutdown():

if request.args.get('token') == TOKEN:

try:

subprocess.run(['sudo', 'shutdown', '-h', 'now'], check=True)

return "Shutting down...", 200

except subprocess.CalledProcessError as e:

return str(e), 500

else:

return "Unauthorized", 403

# Main function to read sensor data and send to server

def main():

global last\_watering\_time\_A, last\_watering\_time\_B

while True:

try:

humidity, temperature = read\_dht\_sensor()

if humidity is not None and temperature is not None:

send\_data\_to\_server(3, temperature)

send\_data\_to\_server(4, humidity)

else:

logging.warning("Failed to read DHT sensor.")

voltage\_A, moisture\_percent\_A = read\_moisture(ADS.P0, wet\_voltage\_A, dry\_voltage\_A)

voltage\_B, moisture\_percent\_B = read\_moisture(ADS.P1, wet\_voltage\_B, dry\_voltage\_B)

send\_data\_to\_server(1, moisture\_percent\_A)

send\_data\_to\_server(2, moisture\_percent\_B)

# Check watering conditions

if moisture\_percent\_A < 50 and (datetime.now() - last\_watering\_time\_A).days >= 3:

GPIO.output(PIN\_1, GPIO.HIGH)

logging.info("Watering Sensor A")

time.sleep(1 \* 60) # Water for 1 minute

GPIO.output(PIN\_1, GPIO.LOW)

last\_watering\_time\_A = datetime.now()

if moisture\_percent\_B < 50 and (datetime.now() - last\_watering\_time\_B).days >= 3:

GPIO.output(PIN\_2, GPIO.HIGH)

logging.info("Watering Sensor B")

time.sleep(1 \* 60) # Water for 1 minute

GPIO.output(PIN\_2, GPIO.LOW)

last\_watering\_time\_B = datetime.now()

time.sleep(60)

except Exception as e:

logging.error(f"Exception in main loop: {e}")

if \_\_name\_\_ == '\_\_main\_\_':

from threading import Thread

server\_thread = Thread(target=lambda: app.run(host='0.0.0.0', port=8000))

server\_thread.start()

main()