ДОДАТОК А

Програмний код “index.html” для ПЗ

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Моніторинг ESP32</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}"
/>
</head>
<body>
  <nav>
    <button id="btn-table">Таблиця</button>
    <button id="btn-charts">Графіки</button>
    <button id="btn-clear">Очистити дані</button>
  </nav>

  <section id="table-section">
    <table>
      <thead>
        <tr>
          <th>Час</th>
          <th>Температура (°C)</th>
          <th>Світло</th>
          <th>Газ</th>
        </tr>
      </thead>
      <tbody id="data-table-body"></tbody>
```

```
    </table>
  </section>


  <section id="charts-section" class="hidden">
    <canvas id="chartTemperature"></canvas>
    <canvas id="chartLight"></canvas>
    <canvas id="chartGas"></canvas>
  </section>

  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <script src="{{ url_for('static', filename='js/script.js') }}"></script>
</body>
</html>
```

ДОДАТОК Б

Програмний код “style.css” для ПЗ

```
body {
  font-family: Arial, sans-serif;
  margin: 20px;
  background-color: #f9f9f9;
}

nav {
  margin-bottom: 20px;
}

button {
  margin-right: 10px;
  padding: 10px 20px;
  border: none;
  background-color: #2b7cff;
  color: white;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #1f5edb;
}

.hidden {
  display: none;
}
```

```
table {
  width: 100%;
  border-collapse: collapse;
  background-color: white;
}

th, td {
  padding: 8px;
  text-align: center;
  border: 1px solid #ddd;
}

canvas {
  max-width: 100%;
  max-height: 300px;
  background-color: white;
  border: 1px solid #ddd;
  padding: 10px;
  margin-bottom: 20px;
}
```

ДОДАТОК В

Програмний код “script.js” для ПЗ

```
const btnTable = document.getElementById('btn-table');
const btnCharts = document.getElementById('btn-charts');
const btnClear = document.getElementById('btn-clear');

const tableSection = document.getElementById('table-section');
const chartsSection = document.getElementById('charts-section');

btnTable.onclick = () => {
  tableSection.classList.remove('hidden');
  chartsSection.classList.add('hidden');
};

btnCharts.onclick = () => {
  tableSection.classList.add('hidden');
  chartsSection.classList.remove('hidden');
  updateCharts();
};

btnClear.onclick = () => {
  if (confirm("Ви впевнені, що хочете очистити всі дані?")) {
    fetch('/clear', { method: 'POST' })
      .then(response => response.json())
      .then(data => {
        alert(data.message);
        loadTableData();
        clearCharts();
      });
```

```
  }
};

async function loadTableData() {
  const response = await fetch('/data');
  const data = await response.json();

  const tbody = document.getElementById('data-table-body');
  tbody.innerHTML = '';

  data.forEach(item => {
    const tr = document.createElement('tr');
    tr.innerHTML = `
      <td>${item.timestamp}</td>
      <td>${item.temperature.toFixed(2)}</td>
      <td>${item.light}</td>
      <td>${item.gas}</td>
    `;
    tbody.appendChild(tr);
  });
}

// Графіки
let chartTemp, chartLight, chartGas;

async function updateCharts() {
  const response = await fetch('/data');
  const data = await response.json();

  const labels = data.map(item => item.timestamp);
```

```
  const temps = data.map(item => item.temperature);
  const lights = data.map(item => item.light);
  const gases = data.map(item => item.gas);

  const ctxTemp =
document.getElementById('chartTemperature').getContext('2d');
  const ctxLight = document.getElementById('chartLight').getContext('2d');
  const ctxGas = document.getElementById('chartGas').getContext('2d');

  if (chartTemp) chartTemp.destroy();
  if (chartLight) chartLight.destroy();
  if (chartGas) chartGas.destroy();

  chartTemp = new Chart(ctxTemp, {
    type: 'line',
    data: {
      labels,
      datasets: [{
        label: 'Температура (°C)',
        data: temps,
        borderColor: 'red',
        fill: false,
      }]
    }
  });

  chartLight = new Chart(ctxLight, {
    type: 'line',
    data: {
      labels,
```

```
      datasets: [{
        label: 'Світло',
        data: lights,
        borderColor: 'orange',
        fill: false,
      }]
    }
  });

  chartGas = new Chart(ctxGas, {
    type: 'line',
    data: {
      labels,
      datasets: [{
        label: 'Газ',
        data: gases,
        borderColor: 'green',
        fill: false,
      }]
    }
  });
}

function clearCharts() {
  if (chartTemp) chartTemp.destroy();
  if (chartLight) chartLight.destroy();
  if (chartGas) chartGas.destroy();
}

function autoUpdate() {
```

```
  loadTableData();
  if (!chartsSection.classList.contains('hidden')) {
    updateCharts();
  }
}

// Загрузка даних при старті
loadTableData();

// Автоматичне оновлення кожні 5 секунд
setInterval(autoUpdate, 5000);
```

ДОДАТОК Г

Програмний код “scetch.ino” для ПЗ

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <BH1750.h>

#define ONE_WIRE_BUS 13       // Температурний сенсор DS18B20
#define MQ4_ANALOG_PIN 34     // MQ-4 аналоговий
#define MQ4_DIGITAL_PIN 4     // MQ-4 цифровий

const char* ssid = "local-spot";
const char* password = "qwerty123";
const char* serverName = "http://192.168.1.110:5000/data";

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

BH1750 lightMeter;

void setup() {
  Serial.begin(115200);
  pinMode(MQ4_DIGITAL_PIN, INPUT);
  sensors.begin();

  Wire.begin(21, 22);
  if (lightMeter.begin()) {
```

```
    Serial.println("BH1750 started successfully");
  } else {
    Serial.println("Error starting BH1750");
  }

  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected!");
}

void loop() {
  sensors.requestTemperatures();
  float temperature = sensors.getTempCByIndex(0);
  uint16_t lux = lightMeter.readLightLevel();
  int gasAnalog = analogRead(MQ4_ANALOG_PIN);
  int gasDigital = digitalRead(MQ4_DIGITAL_PIN);

  Serial.printf("Temperature: %.2f °C, Light: %d lux, GasA: %d, GasD:
%d\n", temperature, lux, gasAnalog, gasDigital);

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverName);
    http.addHeader("Content-Type", "application/json");

    String json = "{\"temperature\":" + String(temperature) +
```

```
                ",\"light\":" + String(lux) +
                ",\"gas\":" + String(gasAnalog) + "}";

  int httpResponseCode = http.POST(json);

  Serial.print("POST Response code: ");
  Serial.println(httpResponseCode);
  String response = http.getString();
  Serial.println("Server response: " + response);

  http.end();
 }

 delay(5000);
}
```

ДОДАТОК Ґ

Програмний код “app.py” для ПЗ

```
from flask import Flask, render_template, request, jsonify
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///data.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

# Модель данных
class SensorData(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    temperature = db.Column(db.Float, nullable=False)
    light = db.Column(db.Integer, nullable=False)
    gas = db.Column(db.Integer, nullable=False)
    timestamp = db.Column(db.DateTime, default=datetime.utcnow)

# Создаем таблицы в базе, если их нет
with app.app_context():
    db.create_all()

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/data', methods=['POST'])
def add_data():
```

```
    data = request.get_json()
    if not data:
        return jsonify({"error": "No JSON data received"}), 400

    try:
        temperature = float(data['temperature'])
        light = int(data['light'])
        gas = int(data['gas'])
    except (KeyError, ValueError):
        return jsonify({"error": "Invalid data format"}), 400

    new_entry = SensorData(temperature=temperature, light=light, gas=gas)
    db.session.add(new_entry)
    db.session.commit()
    return jsonify({"message": "Data added"}), 201

@app.route('/data', methods=['GET'])
def get_data():
    entries                                                              =
SensorData.query.order_by(SensorData.timestamp.desc()).limit(100).all()
    result = []
    for e in reversed(entries):
        result.append({
            "temperature": e.temperature,
            "light": e.light,
            "gas": e.gas,
            "timestamp": e.timestamp.strftime("%Y-%m-%d %H:%M:%S")
        })
    return jsonify(result)
```

```
@app.route('/clear', methods=['POST'])
def clear_data():
    SensorData.query.delete()
    db.session.commit()
    return jsonify({"message": "All data cleared"}), 200

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

ДОДАТОК Д

Демонстраційний матеріал

№ документа	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК 411710.027 ПЗ	Пояснювальна записка	А4, 42 с.
		Інші документи	
2		Програмний код «index.html» для ПЗ	А4, 2 с.
3		Програмний код «style.css» для ПЗ	А4, 2 с.
4		Програмний код «script.js» для ПЗ	А4, 5 с.
5		Програмний код «sketch.ino» для ПЗ	А4, 3 с.
6		Програмний код «app.py» для ПЗ	А4, 3 с.
7		Демонстраційний матеріал	А4, 10 с.

					ГЮИК 411710.027 ВД
Змін.	Арк	Номер докум.	Підп.	Дата	
Розробив		Литочкін Н.О.–			Розроблення системи автоматизації стану продукції в промислових холодильниках Відомість кваліфікаційної роботи
Перевірив		Хрустальов К.Л.			
Норм.контр		Демська Н.П.			
Затвердив		Невлюдов І.Ш.			

Літер	Аркуш	Арк
Н	1	1

Кафедра КІТАР ХНУРЕ