ДОДАТОК А

Код програми

climate_monitoring_simulation:

```
function climate_monitoring_simulation
    hFig = figure('Name', 'Система Моніторингу Кліматичних
Параметрів', ...
                  'NumberTitle', 'off', ...
                  'Position', [100, 100, 600, 500]);

    % Графіки для відображення даних
    hAxesTemp = subplot(2, 1, 1, 'Parent', hFig);
    hAxesHumidity = subplot(2, 1, 2, 'Parent', hFig);

    numPoints = 100;
    temperatureData = NaN(numPoints, 1);
    humidityData = NaN(numPoints, 1);
    timeData = 1:numPoints;

    targetTemp = 22;
    targetHumidity = 50;
    currentTemp = 22;
    currentHumidity = 50;

    broker = MQTTBroker();
    if isempty(broker)
        error('Не вдалося створити брокера MQTT.');
    end
    publisher = MQTTClient(broker, 'climate/parameters',
```

```
@updateLog);
    if isempty(publisher)
        error('Не вдалося створити клієнта MQTT.');
    end
    subscriber = MQTTSubscriber(@updateLog);
    if isempty(subscriber)
        error('Не вдалося створити підписувача MQTT.');
    end
    broker.subscribe('climate/parameters', subscriber);

    % Створення вікна журналу
    hLogFig = figure('Name', 'Журнал MQTT', ...
                     'NumberTitle', 'off', ...
                     'Position', [720, 100, 600, 500]);

    uicontrol('Style', 'text', 'Position', [20, 415, 150, 20], 'String',
'Поточна Температура (°C):', 'Parent', hLogFig);
    hCurrentTemp = uicontrol('Style', 'text', 'Position', [180, 415,
100, 20], 'String', 'Н/Д', 'Parent', hLogFig);

    uicontrol('Style', 'text', 'Position', [20, 385, 150, 20], 'String',
'Поточна Вологість (%):', 'Parent', hLogFig);
    hCurrentHumidity = uicontrol('Style', 'text', 'Position', [180,
385, 100, 20], 'String', 'Н/Д', 'Parent', hLogFig);

    uicontrol('Style', 'pushbutton', 'Position', [300, 405, 200, 30],
'String', 'Встановити Параметри', 'Parent', hLogFig, ...
              'Callback', @openParameterWindow);

    % Вікно журналу
    hLogBox = uicontrol('Style', 'listbox', 'Position', [20, 20, 560,
350], 'Parent', hLogFig);

    % Таймер для оновлення даних
```

```
    t = timer('ExecutionMode', 'fixedRate', 'Period', 1, 'TimerFcn',
@updateData);
    if isempty(t)
        error('Не вдалося створити таймер.');
    end
    start(t);

    % Функція для відкриття вікна налаштування параметрів
    function openParameterWindow(~, ~)
        hParamFig = figure('Name', 'Встановити Параметри', ...
                           'NumberTitle', 'off', ...
                           'Position', [150, 150, 300, 200]);

        uicontrol('Style', 'text', 'Position', [20, 140, 150, 20], 'String',
'Цільова Температура (°C):');
        hTargetTemp = uicontrol('Style', 'edit', 'Position', [180, 140,
100, 20], 'String', num2str(targetTemp));

        uicontrol('Style', 'text', 'Position', [20, 100, 150, 20], 'String',
'Цільова Вологість (%):');
        hTargetHumidity = uicontrol('Style', 'edit', 'Position', [180,
100, 100, 20], 'String', num2str(targetHumidity));

        uicontrol('Style', 'pushbutton', 'Position', [100, 30, 100, 30],
'String', 'Встановити', ...
                  'Callback', @setParameters);

        function setParameters(~, ~)
            temp = str2double(get(hTargetTemp, 'String'));
            humidity = str2double(get(hTargetHumidity, 'String'));

            % Перевірка діапазону температури
            if isnan(temp) || temp < -20 || temp > 30
                errordlg('Температура повинна бути в діапазоні від -
20 до 30 °C', 'Помилка');
```

```
            return;
        end

        if isnan(humidity) || humidity < 1 || humidity > 100
            errordlg('Вологість повинна бути в діапазоні від 1 до
100 %', 'Помилка');
            return;
        end

        targetTemp = temp;
        targetHumidity = humidity;
        disp(['Цільова температура встановлена на ',
num2str(targetTemp), ' °C']);
        disp(['Цільова вологість встановлена на ',
num2str(targetHumidity), ' %']);
        close(hParamFig); % Закрити вікно налаштування
параметрів

        message = sprintf('Temperature: %.2f, Humidity: %.2f',
targetTemp, targetHumidity);
        publisher.publish(message);
    end
end

% Функція для оновлення даних
function updateData(~, ~)
    % Плавна зміна поточних значень
    changeRate = 0.1; % Швидкість зміни
    tolerance = 0.05; % Допустиме відхилення від цільового
значення

    if abs(currentTemp - targetTemp) > tolerance
        if currentTemp < targetTemp
            currentTemp = currentTemp + changeRate;
```

```
        elseif currentTemp > targetTemp
            currentTemp = currentTemp - changeRate;
        end
    else
        currentTemp = targetTemp; % Встановлення точного
значення, якщо відхилення менше за допустиме
    end

    if abs(currentHumidity - targetHumidity) > tolerance
        if currentHumidity < targetHumidity
            currentHumidity = currentHumidity + changeRate;
        elseif currentHumidity > targetHumidity
            currentHumidity = currentHumidity - changeRate;
        end
    else
        currentHumidity = targetHumidity;
    end

    if ishandle(hCurrentTemp)
        set(hCurrentTemp, 'String', sprintf('%.2f', currentTemp));
    end
    if ishandle(hCurrentHumidity)
        set(hCurrentHumidity, 'String', sprintf('%.2f',
currentHumidity));
    end

    temperatureData = [temperatureData(2:end); currentTemp];
    humidityData = [humidityData(2:end); currentHumidity];

    if ishandle(hAxesTemp)
        plot(hAxesTemp, timeData, temperatureData, '-o');
        title(hAxesTemp, 'Температура');
        xlabel(hAxesTemp, 'Час');
```

```
            ylabel(hAxesTemp, 'Температура (°C)');
            grid(hAxesTemp, 'on');
        end
        if ishandle(hAxesHumidity)
            plot(hAxesHumidity, timeData, humidityData, '-o');
            title(hAxesHumidity, 'Вологість');
            xlabel(hAxesHumidity, 'Час');
            ylabel(hAxesHumidity, 'Вологість (%)');
            grid(hAxesHumidity, 'on');
        end
    end

    function updateLog(message)
        if ishandle(hLogBox)
            logEntries = get(hLogBox, 'String');
            logEntries{end+1} = message;
            set(hLogBox, 'String', logEntries);
            set(hLogBox, 'Value', length(logEntries));
            saveLogToFile(message); % Збереження повідомлення
в файл
        end
    end

    function saveLogToFile(message)
        logFile = 'mqtt_log.txt';
        fid = fopen(logFile, 'a');
        if fid == -1
            error('Не вдалося відкрити файл для запису');
        end
        fprintf(fid, '%s\n', message);
        fclose(fid);
    end
end
```

MQTTBroker:

```
classdef MQTTBroker < handle
    properties
        Subscribers = containers.Map;
        BrokerName = 'Брокер'
    end

    methods
        function subscribe(obj, topic, subscriber)
            if isKey(obj.Subscribers, topic)
                obj.Subscribers(topic) = [obj.Subscribers(topic),
{subscriber}];
            else
                obj.Subscribers(topic) = {subscriber};
            end
        end

        function forwardMessage(obj, topic, message)
            if isKey(obj.Subscribers, topic)
                subscribers = obj.Subscribers(topic);
                for i = 1:length(subscribers)
                    subscribers{i}.receiveMessage(topic, message);
                end
            end
        end
    end
end
```

MQTTClient:

```
classdef MQTTClient < handle
    properties
        Broker
        Topic
        LogCallback
    end

    methods
        function obj = MQTTClient(broker, topic, logCallback)
            obj.Broker = broker;
            obj.Topic = topic;
            obj.LogCallback = logCallback;
        end

        function publish(obj, message)
            logMsg = sprintf('Публікація на %s за темою %s: %s',
obj.Broker.BrokerName, obj.Topic, message);
            fprintf('%s\n', logMsg);
            if ~isempty(obj.LogCallback)
                obj.LogCallback(logMsg);
            end
            % Імітація пересилання повідомлення брокером до
підписників
            obj.Broker.forwardMessage(obj.Topic, message);
        end
    end
end
```

MQTTSubscriber:

```
classdef MQTTSubscriber < handle
    properties
        LogCallback
    end

    methods
        function obj = MQTTSubscriber(logCallback)
            obj.LogCallback = logCallback;
        end

        function receiveMessage(obj, topic, message)
            logMsg = sprintf('Отримано за темою %s: %s', topic,
message);
            fprintf('%s\n', logMsg);
            if ~isempty(obj.LogCallback)
                obj.LogCallback(logMsg);
            end
        end
    end
end
```

ДОДАТОК Б

Демонстраційний матеріал

№	Формат	Позначення	Найменування	Кіл.	Прим.
			Документація текстова		
1	*А4*	*ГЮИК. 531016.001ПЗ*	*Пояснювальна записка*	*84 с.*	
			Інші документи		
2	*А4*		*Демонстраційний матеріал*	*10 с.*	*у форматі *.pdf*

Змі	Арк	№ докум.	Підпис	Дата	ГЮИК.531016.001ВД		
Розроб		БутенкоБ.А.			*КТ 1. Розробка системи моніторингу кліматичних параметрів приміщення. ПТ 2. Програмна реалізація*	Літера	Аркуш 1 / Аркушів
Перевірив		МаксимоваС.С.					
Н. Контр.		СтародубцевМ.Г.				ХНУРЕ кафедра КІТАМ	
Затвер.		НевлюдовІ.Ш.					