ДОДАТОК А

Текст програми

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using DiplomaFomin.Database;
using DiplomaFomin.Services;
using Timer = System.Windows.Forms.Timer;

namespace DiplomaFomin
{
   public partial class SimulationForm : Form
   {
      private Timer simulationTimer;
      private Random rnd = new Random();
      private readonly TelegramService _telegramService;

      private Dictionary<int, bool> deviceFailed = new Dictionary<int, bool>();

      public SimulationForm()
      {
         InitializeComponent();
         this.FormClosing += SimulationForm_FormClosing;

         _telegramService = new TelegramService(
            "8061010453:AAGbcpz7YRY6rHHrUEEdvLxvQU-dVduZ81g",
            -4799699635);
      }

      private void StartSimulation()
      {
         simulationTimer = new Timer();
         simulationTimer.Interval = 5000;
         simulationTimer.Tick += SimulationTimer_Tick;
         simulationTimer.Start();
      }

      private async void SimulationTimer_Tick(object sender, EventArgs e)
      {
         var (deviceIds, thresholds) = LoadDevicesAndThresholds();
```

```
        if (deviceIds.Count == 0) return;

        foreach (int deviceId in deviceIds)
        {
            if (deviceFailed.TryGetValue(deviceId, out var failed) && failed)
            {
                await LogDeviceReadingsAsync(deviceId, 0, 0,
                    anomaly: true, permanentFailure: true);
                continue;
            }

            double temp = rnd.NextDouble() * 10 + 20;
            double wet = rnd.NextDouble() * 20 + 40;

            if (thresholds.TryGetValue(deviceId, out var th))
            {

                double targetTemp = (th.MinTemp + th.MaxTemp) / 2.0;
                double targetHum = (th.MinHum + th.MaxHum) / 2.0;

                double errorT = targetTemp - temp;
                double errorH = targetHum - wet;

                double kT = 0.5;
                double kH = 0.5;


                temp += kT * errorT;
                wet += kH * errorH;
            }

            if (thresholds.TryGetValue(deviceId, out var thCheck) &&
                (temp < thCheck.MinTemp || temp > thCheck.MaxTemp ||
                 wet < thCheck.MinHum || wet > thCheck.MaxHum))
            {
                await LogDeviceReadingsAsync(deviceId, temp, wet,
                    anomaly: true, permanentFailure: false);

                string deviceName = GetDeviceName(deviceId);
                string reason = "Threshold breach";
                string msg = $"Пристрій «{deviceName}» (ID {deviceId}) за
межами порогів: T={temp:F1}, H={wet:F1}";

                if (!NotificationExists(deviceId, reason))
```

```
            {
                InsertNotification(deviceId, deviceName, reason, msg);
                await _telegramService.SendMessageAsync(msg);
            }

            continue;
        }

        if (rnd.NextDouble() < 0.1)
        {
            if (rnd.NextDouble() < 0.5)
            {
                deviceFailed[deviceId] = true;
                MarkDeviceAsFailedInDB(deviceId);
                await LogDeviceReadingsAsync(deviceId, 0, 0,
                    anomaly: true, permanentFailure: true);
            }
            else
            {
                await LogDeviceReadingsAsync(deviceId, 0, 0,
                    anomaly: true, permanentFailure: false);
            }
        }
        else
        {
            await LogDeviceReadingsAsync(deviceId, temp, wet,
                anomaly: false, permanentFailure: false);
        }
    }
}

private (List<int> deviceIds, Dictionary<int, Threshold> thresholds)
    LoadDevicesAndThresholds()
{
    var ids = new List<int>();
    var ths = new Dictionary<int, Threshold>();
    deviceFailed.Clear();

    const string sql = @"
SELECT d.id, d.is_failed,
       l.min_temp, l.max_temp, l.min_hum, l.max_hum
  FROM devices d
  JOIN locations l ON d.location_id = l.id
 WHERE d.status = 'active'";
```

```
            using var conn = new DbConnection().GetConnection();
            conn.Open();
            using var cmd = new MySqlCommand(sql, conn);
            using var rdr = cmd.ExecuteReader();
            while (rdr.Read())
            {
                int id = rdr.GetInt32(0);
                bool isFailed = rdr.GetBoolean(1);
                double? minT = rdr.IsDBNull(2) ? (double?)null : rdr.GetDouble(2);
                double? maxT = rdr.IsDBNull(3) ? (double?)null : rdr.GetDouble(3);
                double? minH = rdr.IsDBNull(4) ? (double?)null : rdr.GetDouble(4);
                double? maxH = rdr.IsDBNull(5) ? (double?)null : rdr.GetDouble(5);

                ids.Add(id);
                deviceFailed[id] = isFailed;

                ths[id] = new Threshold
                {
                    MinTemp = minT ?? double.MinValue,
                    MaxTemp = maxT ?? double.MaxValue,
                    MinHum = minH ?? double.MinValue,
                    MaxHum = maxH ?? double.MaxValue
                };
            }

            return (ids, ths);
        }

        private class Threshold
        {
            public double MinTemp, MaxTemp, MinHum, MaxHum;
        }

        private void MarkDeviceAsFailedInDB(int deviceId)
        {
            try
            {
                DbConnection db = new DbConnection();
                using (MySqlConnection conn = db.GetConnection())
                {
                    conn.Open();
                    string updateQuery = "UPDATE devices SET is_failed = 1 WHERE
id = @deviceId";
                    using (MySqlCommand cmd = new MySqlCommand(updateQuery,
conn))
```

```
                {
                    cmd.Parameters.AddWithValue("@deviceId", deviceId);
                    cmd.ExecuteNonQuery();
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Помилка оновлення статусу пристрою: " +
ex.Message);
        }
    }

    private async Task LogDeviceReadingsAsync(int deviceId, double
temperature, double wetness, bool anomaly, bool permanentFailure = false)
    {
        try
        {
            using var conn = new DbConnection().GetConnection();
            await conn.OpenAsync();
            const string insertQuery = @"
        INSERT INTO device_readings
            (device_id, reading_date, reading_temp, reading_wet,
reading_temp_value, reading_wet_value)
        VALUES
            (@deviceId, NOW(), @readingTemp, @readingWet, @tempValue,
@wetValue)";
            using var cmd = new MySqlCommand(insertQuery, conn);
            cmd.Parameters.AddWithValue("@deviceId", deviceId);

            if (anomaly && !permanentFailure)
            {
                cmd.Parameters.AddWithValue("@readingTemp", "threshold");
                cmd.Parameters.AddWithValue("@readingWet", "threshold");
                cmd.Parameters.AddWithValue("@tempValue", temperature);
                cmd.Parameters.AddWithValue("@wetValue", wetness);
            }
            else if (anomaly && permanentFailure)
            {
                cmd.Parameters.AddWithValue("@readingTemp", "permanent");
                cmd.Parameters.AddWithValue("@readingWet", "permanent");
                cmd.Parameters.AddWithValue("@tempValue", temperature);
                cmd.Parameters.AddWithValue("@wetValue", wetness);
            }
            else
```

```
            {
                cmd.Parameters.AddWithValue("@readingTemp", "temperature");
                cmd.Parameters.AddWithValue("@readingWet", "humidity");
                cmd.Parameters.AddWithValue("@tempValue", temperature);
                cmd.Parameters.AddWithValue("@wetValue", wetness);
            }

            await cmd.ExecuteNonQueryAsync();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Помилка запису показання: " + ex.Message);
            return;
        }

        if (anomaly)
        {
            string deviceName = GetDeviceName(deviceId);
            string reason;
            string messageText;

            if (!permanentFailure)
            {
                reason = "Threshold breach";
                messageText = $"Пристрій \"{deviceName}\" (ID {deviceId}) за
межами порогів: T={temperature:F1}, H={wetness:F1}";
            }
            else
            {
                reason = "Permanent failure";
                messageText = $"Пристрій \"{deviceName}\" (ID {deviceId})
вийшов з ладу остаточно: T={temperature:F1}, H={wetness:F1}";
            }

            if (!NotificationExists(deviceId, reason))
            {

                InsertNotification(deviceId, deviceName, reason, messageText);

                try
                {
                    await _telegramService.SendMessageAsync(messageText);
                }
                catch (Exception)
                {
```

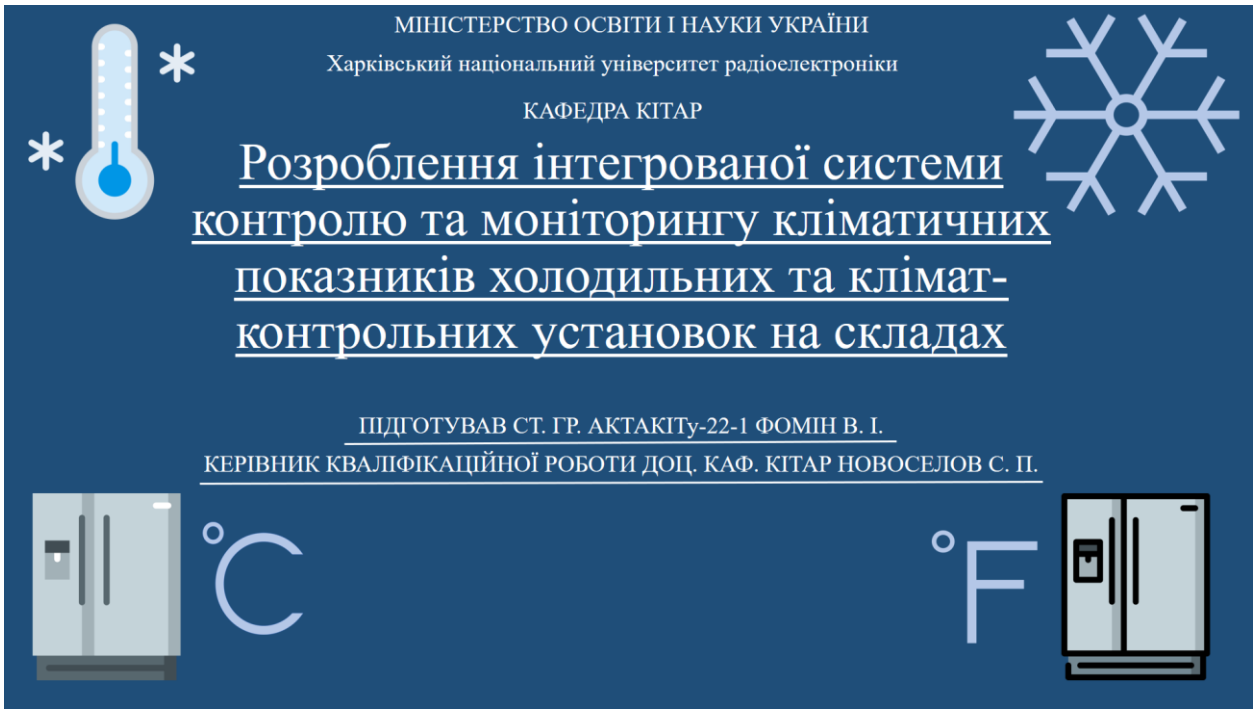
```
                System.Diagnostics.Debug.WriteLine("Не вдалося відправити
Telegram-повідомлення");
            }
        }
    }
}
private string GetDeviceName(int deviceId)
{
    try
    {
        string deviceName = "";
        DbConnection db = new DbConnection();
        using (MySqlConnection conn = db.GetConnection())
        {
            conn.Open();
            string query = "SELECT name FROM devices WHERE id =
@deviceId";
            using (MySqlCommand cmd = new MySqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@deviceId", deviceId);
                object result = cmd.ExecuteScalar();
                if (result != null && result != DBNull.Value)
                    deviceName = result.ToString();
            }
        }
        return deviceName;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Помилка отримання назви пристрою: " +
ex.Message);
        return "";
    }
}

private void InsertNotification(int deviceId, string deviceName, string
anomalyReason, string message)
{
    try
    {
        if (NotificationExists(deviceId, anomalyReason))
        {
            return;
        }
```

```
            DbConnection db = new DbConnection();
            using (MySqlConnection conn = db.GetConnection())
            {
                conn.Open();
                string query = @"
                    INSERT INTO device_notifications (device_id, device_name,
anomaly_reason, message)
                    VALUES (@deviceId, @deviceName, @anomalyReason,
@message)";
                using (MySqlCommand cmd = new MySqlCommand(query, conn))
                {
                    cmd.Parameters.AddWithValue("@deviceId", deviceId);
                    cmd.Parameters.AddWithValue("@deviceName", deviceName);
                    cmd.Parameters.AddWithValue("@anomalyReason",
anomalyReason);
                    cmd.Parameters.AddWithValue("@message", message);
                    cmd.ExecuteNonQuery();
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Помилка запису сповіщення: " + ex.Message);
        }
    }

    private bool NotificationExists(int deviceId, string anomalyReason)
    {
        bool exists = false;
        try
        {
            DbConnection db = new DbConnection();
            using (MySqlConnection conn = db.GetConnection())
            {
                conn.Open();
                string query = "SELECT COUNT(*) FROM device_notifications
WHERE device_id = @deviceId AND anomaly_reason = @anomalyReason";
                using (MySqlCommand cmd = new MySqlCommand(query, conn))
                {
                    cmd.Parameters.AddWithValue("@deviceId", deviceId);
                    cmd.Parameters.AddWithValue("@anomalyReason",
anomalyReason);
                    long count = (long)cmd.ExecuteScalar();
                    exists = count > 0;
                }
```

```
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Помилка перевірки існування сповіщення: " +
ex.Message);
        }
        return exists;
    }

    private void SimulationForm_FormClosing(object sender,
FormClosingEventArgs e)
    {
        if (simulationTimer != null)
        {
            simulationTimer.Stop();
            simulationTimer.Dispose();
        }
    }

    private void buttonStarSim_Click(object sender, EventArgs e)
    {
        if (simulationTimer != null && simulationTimer.Enabled)
        {
            MessageBox.Show("Симуляція вже запущена.");
            return;
        }
        StartSimulation();
        MessageBox.Show("Симуляцію запущено.");
    }

    private void buttonStopSim_Click(object sender, EventArgs e)
    {
        if (simulationTimer != null && simulationTimer.Enabled)
        {
            simulationTimer.Stop();
            MessageBox.Show("Симуляцію зупинено.");
        }
        else
        {
            MessageBox.Show("Симуляція не запущена.");
        }
    }
  }
}
```

ДОДАТОК Б

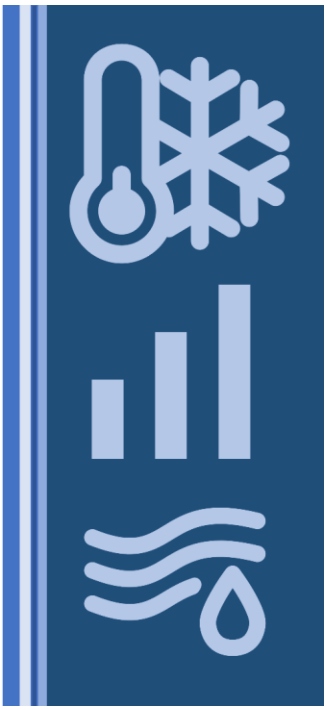
Демонстраційно-графічний матеріал



АКТУАЛЬНІСТЬ РОБОТИ

Стабільність кліматичних параметрів, особливо температури та вологості, має ключову роль у забезпеченні збереження якості продукції під час зберігання або транспортування. Ці параметри безпосередньо впливають на фізичний стан, термін придатності та споживчі властивості товарів. Найбільш чутливими до коливань мікроклімату є продукти харчування, фармацевтична продукція, хімікати, квіти та інші матеріали, які потребують суворого дотримання певних умов зберігання. Відхилення від нормативних умов може призвести до псування, втрати товарного вигляду або навіть повної непридатності продукції, що в свою чергу спричиняє фінансові збитки, втрату клієнтів та погіршення репутації компанії.

Передусім це актуально для складських і холодильних приміщень, де контроль клімату є необхідною умовою ефективного функціонування логістичних ланцюгів. У таких середовищах навіть незначні відхилення від встановлених норм можуть мати суттєві наслідки, тому підприємства дедалі частіше впроваджують сучасні автоматизовані системи контролю та моніторингу мікроклімату. Такі системи здатні в реальному часі відслідковувати ключові параметри, зберігати історію змін, а також надсилати повідомлення про критичні відхилення, що дозволяє оперативно вживати заходів для їх усунення.



МЕТА ТА ЗАДАЧІ РОБОТИ

Метою кваліфікаційної роботи є розробка інтегрованої системи для моніторингу температури та вологості з реалізацією сповіщення про критичні відхилення та відправкою цих сповіщень в Telegram.

Задачами, що потрібно вирішити у кваліфікаційної роботі є:

– провести аналіз сучасних систем моніторингу кліматичних показників які використовуються на складах;

– дослідити можливості застосування Telegram API для оперативного інформування користувачів про відхилення параметрів;

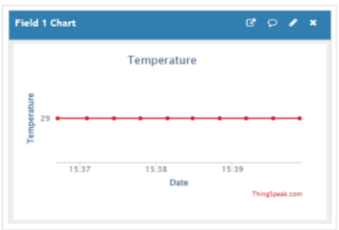
– вибрати та обгрунтувати інструменти розробки програмного забезпечення на основі C# і Windows Forms для реалізації користувацького інтерфейсу та симуляції IoT-пристроїв;

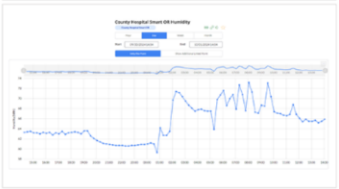
– розробити програмний модуль, що забезпечує обробку, логування даних, а також формування та надсилання сповіщень в Telegram у разі критичних відхилень;

– розглянути питання охорони праці.

СУЧАСНІ СИСТЕМИ МОНІТОРИНГУ КЛІМАТУ

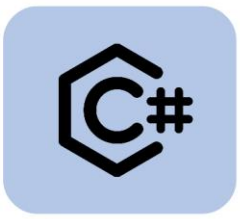
Однією з головних складових сучасних кліматичних систем виступає ПЗ, що інтегрується з базою даних. За аналогією з програмами для обліку складських операцій, у кліматичних системах також фіксуються показники температури, вологості, тиску та стану вентиляційного обладнання. Зазвичай, такі програми часто оснащені графічним інтерфейсом, котрий відображає поточні параметри і дає можливість дистанційно коригувати їх. Зараз із популярних на ринку є: SensorPush, TempTRIP, Monnit.





ВИКОРИСТАНІ ТЕХНОЛОГІЇ ПІД ЧАС РОЗРОБКИ

У проєкті використано мову програмування C# та платформу Windows Forms у середовищі .NET для створення десктопного інтерфейсу. Для збереження та обробки даних є MySQL з підключенням через бібліотеку MySql.Data. Графічна візуалізація кліматичних показників реалізована за допомогою бібліотеки LiveCharts.WinForms. Для експорту зібраних даних у форматі Excel використано бібліотеку ClosedXML. Надсилання повідомлень користувачам реалізовано за допомогою Telegram API через офіційну бібліотеку Telegram.Bot. Робота IoT-пристроїв симулюється із використанням таймерів та генерації випадкових значень температури й вологості у заданих межах.



Мова програмування

Платформа

База даних

Симуляція IoT

Графіки

Telegram API

Експорт до Excel

Робота з БД

МОДУЛЬ СИМУЛЯЦІЇ ПРИСТРОЇВ

Модуль симуляції відповідає за імітацію роботи IoT-пристроїв, які вимірюють температуру та вологість у складських приміщеннях.

Це дозволяє тестувати систему без фізичних сенсорів.

Симуляція відбувається за допомогою таймера, що кожні 5 секунд генерує нові дані для кожного активного пристрою.

Згенеровані значення порівнюються з порогами, встановленими для локацій. У разі перевищення або зниження — створюється аномалія.

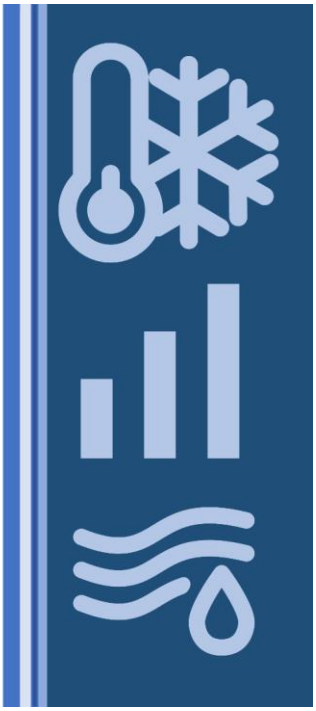
Система обробляє два типи відхилень:

– перевищення допустимих меж (аналітична аномалія);

– випадковий повний вихід пристрою з ладу (перманентна аномалія).

Аномалії записуються в базу даних та викликають надсилання Telegram-сповіщення.

```
if (thresholds.TryGetValue(deviceId, out var thCheck) &&
    (temp < thCheck.MinTemp || temp > thCheck.MaxTemp ||
     wet < thCheck.MinHum || wet > thCheck.MaxHum))
{
    await LogDeviceReadingsAsync(deviceId, temp, wet,
        anomaly: true, permanentFailure: false);

    string deviceName = GetDeviceName(deviceId);
    string reason = "Threshold breach";
    string msg = $"⚠ Пристрій «{deviceName}» (ID {deviceId}) за межами порогів: T={temp:F1}, H={wet:F1}";

    if (!NotificationExists(deviceId, reason))
    {
        InsertNotification(deviceId, deviceName, reason, msg);
        await _telegramService.SendMessageAsync(msg);
    }

    continue;
}
```



TELEGRAM ІНТЕГРАЦІЯ

У проєкті реалізовано повноцінну інтеграцію з Telegram-ботом для інформування користувачів у режимі реального часу.

Система автоматично надсилає сповіщення про:

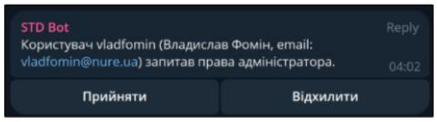
– критичні відхилення температури або вологості;

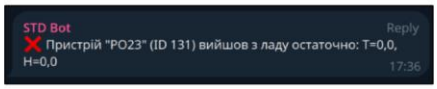
– вихід пристрою з ладу;

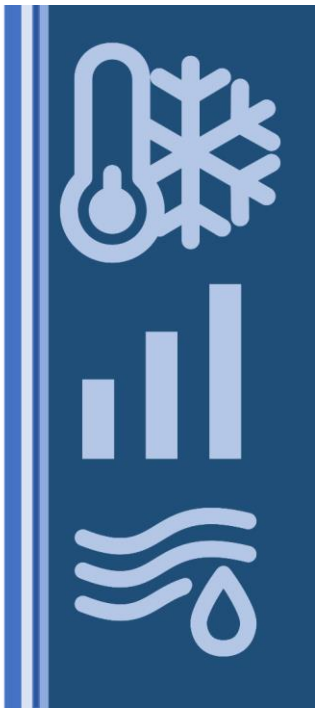
– запити користувачів на отримання прав адміністратора.

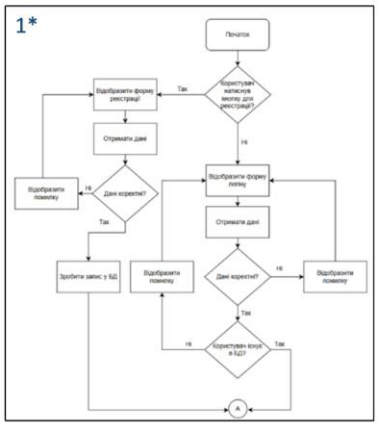
Повідомлення формуються та надсилаються за допомогою бібліотеки Telegram.Bot.

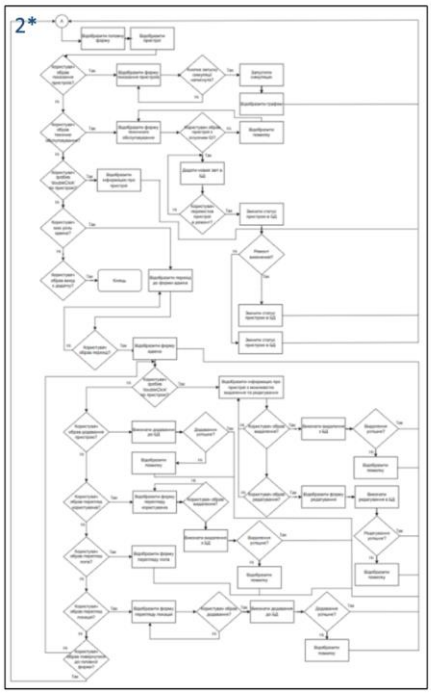
Для обробки кнопок (inline-клавіатури) застосовано механізм CallbackQuery, що дозволяє адміністраторам взаємодіяти з ботом напряму.





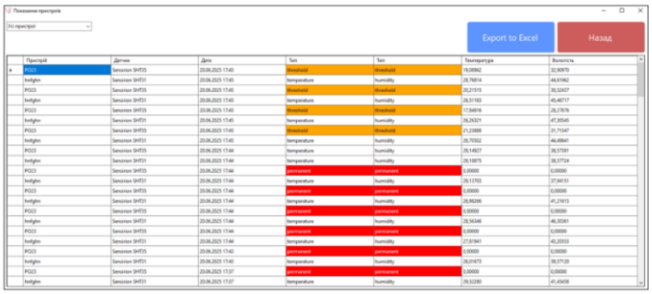






Блок-схема алгоритму роботи системи

Візуальне представлення роботи програми, а саме показ результатів графічним та табличним способами



ВИСНОВКИ

У результаті виконаної кваліфікаційної роботи було створено систему моніторингу кліматичних параметрів для складських приміщень, яка забезпечує зручну взаємодію з користувачем, надійне зберігання даних, своєчасне реагування на критичні відхилення та можливість подальшої адаптації під реальні IoT-пристрої.

Задачами, що були вирішені у кваліфікаційній роботі є:

– проведено аналіз сучасних систем моніторингу кліматичних показників які використовуються на складах;

– досліджено можливості застосування Telegram API для оперативного інформування користувачів про відхилення параметрів;

– вибрано та обгрунтовано інструменти розробки програмного забезпечення на основі C# і Windows Forms для реалізації користувацького інтерфейсу та симуляції IoT-пристроїв;

– розроблено додаток, що забезпечує обробку, логування даних, а також формування та надсилання сповіщень в Telegram у разі критичних відхилень;

– розглянуто питання охорони праці.

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮІК. 505300.011 ПЗ	Пояснювальна записка	63 с.	
			Додаткові матеріали		
А4	2		Текст програми	9 с.	
А4	3		Демонстраційний матеріал у	10 с.	
			вигляді презентації		

Змін.	Арк	Номер докум.	Підпис	Дата	ГЮІК. 505300.011 ПЗ			
Розроб.		Фомін В. І.			Розроблення інтегрованої системи контролю та моніторингу кліматичних показників холодильних та клімат-контрольних установок на складах	Літера	Арку	Аркуш
Перевір.		Новоселов С. П.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Демська Н. П.						
Затв.		Невлюдов І. Ш.						