ДОДАТОК А

Посібник користувача

ЗМІСТ

ВСТУП

А. 1 Призначення компонентів Azure у системі моніторингу

Моніторинг ресурсів є ключовим елементом ефективного функціонування сучасних систем автоматизації, забезпечуючи контроль за використанням обчислювальних, мережевих та хмарних ресурсів. У контексті хмарної платформи Microsoft Azure впровадження моніторингу ресурсів дозволяє оптимізувати продуктивність системи, забезпечувати її стабільну роботу та вчасно виявляти потенційні проблеми.

Основною метою впровадження моніторингу ресурсів є забезпечення прозорості у використанні обчислювальних потужностей, мережевого трафіку та доступу до хмарних сервісів. Це досягається за рахунок збору метрик продуктивності, таких як завантаженість процесора, обсяг використаної оперативної пам’яті, швидкість передачі даних та час відгуку сервісів. У рамках роботи з Azure ці дані збираються за допомогою сервісу Azure Monitor, який забезпечує централізований моніторинг усіх компонентів системи.

Azure Monitor надає інструменти для візуалізації та аналізу отриманих даних, дозволяючи будувати інтерактивні дашборди та встановлювати порогові значення для критичних метрик. Наприклад, для IoT Hub можна відслідковувати кількість прийнятих повідомлень, статус підключень пристроїв та обсяг використаного каналу зв’язку. Для Stream Analytics – це показники часу обробки подій, затримки виконання запитів та ефективності використання обчислювальних ресурсів. Крім того, за допомогою Azure Monitor можна налаштувати оповіщення про перевищення порогових значень, що дозволяє вчасно реагувати на перевантаження системи або інші критичні події.

Важливим аспектом моніторингу є його інтеграція з іншими компонентами системи. Наприклад, налаштування взаємодії між Azure Monitor і Logic Apps дозволяє створювати автоматизовані робочі процеси для реагування на певні події.

У разі виявлення аномальної активності, система може автоматично надсилати сповіщення адміністраторам або запускати додаткові ресурси для стабілізації роботи. Такий підхід забезпечує високий рівень адаптивності системи до змін у робочому навантаженні та підвищує її надійність.

Моніторинг ресурсів також сприяє оптимізації витрат на використання хмарних сервісів. Аналіз використання ресурсів дозволяє виявити зайві або недостатньо ефективні елементи системи, що дає можливість скоротити витрати, зберігаючи необхідний рівень продуктивності. Наприклад, на основі даних моніторингу можна масштабувати обчислювальні потужності відповідно до реального навантаження, уникнувши переплат за невикористані ресурси.

Впровадження моніторингу ресурсів у системах моніторингу технологічних процесів є невід'ємною складовою побудови надійних та ефективних рішень. Це дозволяє забезпечити стабільну роботу системи, своєчасно реагувати на зміни у навантаженні та підтримувати оптимальне використання ресурсів, що особливо важливо у масштабованих хмарних середовищах, таких як Azure.

А. 2 Технічні вимоги для інтеграції з платформою Azure

Інтеграція системи моніторингу з хмарною платформою Microsoft Azure вимагає дотримання ряду технічних вимог, які забезпечують ефективну, безпечну та стабільну роботу системи. Ці вимоги охоплюють апаратну та програмну сумісність, мережеві параметри, а також вимоги до облікового запису та конфігурації Azure.

Перш за все, для інтеграції необхідний стабільний доступ до Інтернету із мінімальною пропускною здатністю, достатньою для передачі телеметрії від IoT-пристроїв до хмарної платформи. Рекомендується використовувати високошвидкісне та низьколатентне підключення, щоб уникнути затримок у передачі даних, особливо у режимах реального часу. Крім того, пристрої, що підключаються до IoT Hub, повинні підтримувати стандартні протоколи передачі даних, такі як MQTT, AMQP або HTTPS.

Для роботи з Azure потрібен обліковий запис, який має доступ до необхідних сервісів, таких як IoT Hub, Stream Analytics, Blob Storage та Power BI. Рекомендується використовувати підписку рівня Standard або вище для забезпечення достатньої кількості обчислювальних ресурсів та підтримки масштабованості. Обліковий запис має бути налаштований із використанням багатофакторної аутентифікації та ролей з обмеженими правами доступу для різних категорій користувачів, щоб забезпечити безпеку даних і контролювати доступ до ключових компонентів системи.

Обчислювальні ресурси мають відповідати вимогам сервісів Azure. Наприклад, для Stream Analytics потрібно забезпечити обчислювальну потужність для обробки великих обсягів даних у реальному часі. Це включає налаштування розміру обчислювальних вузлів залежно від передбачуваного навантаження. Для Blob Storage потрібно передбачити достатній обсяг пам’яті для зберігання історичних даних та резервних копій, а також вибрати оптимальний тип сховища (наприклад, «Hot» для частого доступу або «Cool» для архівних даних).

Для інтеграції необхідно налаштувати IoT Hub, який слугує комунікаційним вузлом між пристроями та хмарою. Кожен пристрій має бути зареєстрований у системі з використанням унікальних ідентифікаторів, а автентифікація має виконуватися за допомогою SAS-токенів або сертифікатів. Важливо також налаштувати політики доступу, щоб обмежити можливості несанкціонованого підключення до IoT Hub.

Для захисту переданих даних необхідно використовувати протоколи шифрування, такі як TLS, які забезпечують безпечний обмін інформацією між пристроями та хмарними сервісами. Крім того, слід використовувати Azure Security Center для моніторингу безпеки та виявлення потенційних загроз.

Окрім цього, інтеграція з Power BI для візуалізації даних вимагає створення дашбордів, які можуть динамічно оновлюватися на основі оброблених даних. Для цього слід забезпечити коректне налаштування потоків даних із Stream Analytics до Power BI та перевірити їх сумісність із форматами, необхідними для побудови звітів.

Отже, інтеграція з платформою Azure вимагає дотримання низки технічних вимог, які включають забезпечення відповідних обчислювальних ресурсів, налаштування мережевої інфраструктури, використання сучасних протоколів безпеки та правильну конфігурацію хмарних сервісів. Виконання цих умов є важливим для стабільної та ефективної роботи системи моніторингу в середовищі Azure.

А.3 Опис основних функцій реалізації системи в Azure-середовищі

Реалізація системи моніторингу технологічних процесів у середовищі Microsoft Azure забезпечує інтеграцію хмарних технологій для збору, обробки та аналізу даних, а також ефективне управління технологічними процесами. Основні функції системи, реалізовані в Azure-середовищі, охоплюють кілька ключових компонентів, кожен із яких виконує специфічні завдання, спрямовані на забезпечення продуктивності, безпеки та масштабованості.

Однією з важливих функцій є збір даних із підключених IoT-пристроїв, які є джерелами сенсорної інформації. Для цього використовується сервіс IoT Hub, який забезпечує надійний і безпечний канал передачі даних від пристроїв до хмарної платформи. IoT Hub підтримує різні протоколи передачі даних, такі як MQTT та HTTPS, дозволяючи інтегрувати широкий спектр пристроїв. Кожен пристрій реєструється в системі із використанням унікального ідентифікатора, що спрощує управління їх підключенням та автентифікацією.

Ще однією ключовою функцією є обробка та попередня класифікація даних у режимі реального часу. Цю роль виконує сервіс Stream Analytics, який приймає дані з IoT Hub, виконує фільтрацію, згладжування та аналіз, використовуючи задані SQL-запити. Наприклад, Stream Analytics дозволяє виявляти аномалії у даних, спираючись на алгоритми Z-score, Spike and Dip або визначення змінних точок, що допомагає виявляти відхилення від нормальних параметрів роботи технологічного обладнання.

Функція збереження даних реалізується за допомогою Azure Blob Storage, який дозволяє зберігати великі обсяги даних у структурованій та неструктурованій формі. Використання Blob Storage забезпечує економічне управління даними, пропонуючи різні рівні доступу («Hot», «Cool», «Archive») залежно від частоти звернень до даних.

Візуалізація отриманих даних для аналітичних цілей здійснюється за допомогою Power BI, який інтегрується з потоками даних, зібраних у Stream Analytics. Цей інструмент дозволяє створювати інтерактивні дашборди та звіти, що відображають поточний стан технологічних процесів, статистику та виявлені аномалії. Завдяки візуалізації користувачі отримують можливість швидко оцінити продуктивність системи та приймати обґрунтовані рішення.

Автоматизація дій у відповідь на події реалізується за допомогою Azure Logic Apps, які дозволяють запускати автоматизовані робочі процеси у разі виявлення критичних ситуацій. Наприклад, при перевищенні критичних значень температури Logic Apps може відправити сповіщення операторам або зупинити роботу відповідного обладнання. Це дозволяє підвищити оперативність реагування та знизити ризик виникнення аварійних ситуацій.

Забезпечення безпеки переданих даних і користувачів є невід'ємною частиною функціонування системи. Для цього використовується Azure Security Center, який дозволяє відстежувати потенційні загрози, а також протоколи шифрування TLS для захисту передачі даних у мережі. Користувачі автентифікуються за допомогою SAS-токенів або сертифікатів, що забезпечує контроль доступу до системи.

Основні функції, реалізовані в Azure-середовищі, забезпечують комплексне управління технологічними процесами, надаючи інструменти для збору, обробки, аналізу, візуалізації та реагування на дані. Інтеграція з Azure дозволяє створити гнучке, масштабоване та надійне рішення, здатне задовольнити потреби сучасних підприємств.

А.4 Рекомендації щодо ефективного використання сервісів Azure

Ефективне використання сервісів Microsoft Azure забезпечує максимальну продуктивність та надійність автоматизованих систем моніторингу технологічних процесів. Для цього необхідно врахувати кілька ключових аспектів, які стосуються як технічної конфігурації, так і стратегій управління хмарними ресурсами.

Перш за все, важливо налаштувати моніторинг ресурсів за допомогою Azure Monitor. Цей сервіс дозволяє отримувати метрики продуктивності, журнали активності та сповіщення про критичні події для всіх компонентів системи. Використання цього інструменту сприяє виявленню вузьких місць у роботі сервісів, що дозволяє своєчасно оптимізувати їхню продуктивність. Наприклад, моніторинг навантаження на IoT Hub або Stream Analytics допоможе уникнути перевантажень та запобігти втратам даних.

Для оптимізації обробки даних у реальному часі слід налаштувати відповідні вікна обробки (windowing) у Azure Stream Analytics. Це дозволяє адаптувати систему до динамічного характеру вхідних даних, забезпечуючи ефективну фільтрацію та агрегацію. Наприклад, використання ковзних або сесійних вікон дозволяє більш точно визначати аномалії чи тренди в даних.

Безпека даних є ще одним критично важливим аспектом. Рекомендується використовувати шифрування TLS для передачі даних між пристроями IoT та Azure IoT Hub. Для автентифікації пристроїв слід застосовувати SAS-токени або сертифікати, що гарантують захищений доступ до хмарних ресурсів. Крім того, регулярний аудит безпеки за допомогою Azure Security Center допоможе виявити потенційні вразливості та вжити відповідних заходів для їх усунення.

Масштабованість системи є важливим фактором для великих і динамічних середовищ. Azure дозволяє автоматично масштабувати обчислювальні ресурси залежно від навантаження. Наприклад, для обробки зростаючого обсягу даних можна збільшити кількість обчислювальних вузлів Stream Analytics або змінити рівень підписки IoT Hub. Це допоможе уникнути простоїв і забезпечити стабільну роботу системи навіть під час пікових навантажень.

Щоб уникнути зайвих витрат, рекомендується використовувати економічно ефективні режими роботи хмарних сервісів. Наприклад, для довгострокового зберігання даних у Azure Blob Storage слід налаштувати доступ у режимі «Cool» або «Archive» залежно від частоти використання даних. Регулярний аналіз використання ресурсів за допомогою Azure Cost Management допоможе оптимізувати витрати без шкоди для продуктивності системи.

Додатково варто інтегрувати автоматизовані процеси реагування за допомогою Azure Logic Apps. Цей сервіс дозволяє налаштувати робочі процеси для автоматичного реагування на критичні події, наприклад, надсилання сповіщень або запуск додаткових ресурсів у разі виявлення аномалій. Це забезпечує гнучкість системи та зменшує залежність від ручного втручання.

Загалом, ефективне використання сервісів Azure передбачає комплексний підхід, що поєднує правильну конфігурацію ресурсів, забезпечення безпеки, оптимізацію витрат і автоматизацію процесів. Дотримання цих рекомендацій дозволяє створити надійну, масштабовану та економічно вигідну систему моніторингу, яка відповідає сучасним вимогам промислової автоматизації.

ДОДАТОК Б

Апробація результатів кваліфікаційної роботи

SCI-CONF.COM.UA

GLOBAL SCIENCE: PROSPECTS AND INNOVATIONS

PROCEEDINGS OF XI INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE JUNE 20-22, 2024

LIVERPOOL 2024

GLOBAL SCIENCE: PROSPECTS AND INNOVATIONS

Proceedings of XI International Scientific and Practical Conference

Liverpool, United Kingdom

20-22 June 2024

Liverpool, United Kingdom

2024

UDC 001.1

The 11th International scientific and practical conference “Global science: prospects and innovations” (June 20-22, 2024) Cognum Publishing House, Liverpool, United Kingdom. 2024. 547 p.

ISBN 978-92-9472-196-9

The recommended citation for this publication is:

Ivanov I. Analysis of the phaunistic composition of Ukraine // Global science: prospects and innovations. Proceedings of the 11th International scientific and practical conference. Cognum Publishing House. Liverpool, United Kingdom. 2024. Pp. 21-27. URL: https://sci-conf.com.ua/xi-mizhnarodna-naukovo-praktichna-konferentsiya-global-science-prospects-and-innovations-20-22-06-2024-liverpul-velikobritaniya-arhiv/.

Editor
Komarytskyy M.L.
Ph.D. in Economics, Associate Professor

Collection of scientific articles published is the scientific and practical publication, which contains scientific articles of students, graduate students, Candidates and Doctors of Sciences, research workers and practitioners from Europe, Ukraine and from neighbouring countries and beyond. The articles contain the study, reflecting the processes and changes in the structure of modern science. The collection of scientific articles is for students, postgraduate students, doctoral candidates, teachers, researchers, practitioners and people interested in the trends of modern science development.

e-mail: liverpool@sci-conf.com.ua

homepage: *https://sci-conf.com.ua*

УДК 621.865.8

АВТОМАТИЗОВАНІ СИСТЕМИ МОНІТОРИНГУ ТА АЛГОРИТМИ ПОШУКУ АНОМАЛІЙ У РАМКАХ INDUSTRY 4.0

Новоселов Сергій Павлович,
к.т.н., доцент
Канунніков Микита Юрійович
студент
Харківський національний університет радіоелектроніки
м. Харків, Україна
mykyta.kanunnikov@nure.ua

Анотація: У статті розглядається роль автоматизованих систем моніторингу як одного з ключових елементів концепції Industry 4.0. Здійснюється аналіз принципів побудови таких систем, їхньої інтеграції з хмарними технологіями та алгоритмами пошуку аномалій. Особлива увага приділяється перевагам використання автоматизованих рішень у забезпеченні ефективного моніторингу, прогнозування стану технологічних процесів, оптимізації виробничих операцій та підвищення рівня безпеки на підприємствах.

Ключові слова: Industry 4.0, автоматизація, моніторинг технологічних процесів, прогнозування, пошук аномалій, хмарні технології, IoT, безпека на виробництві.

Industry 4.0, або Четверта промислова революція, представляє собою трансформацію сучасного виробництва завдяки впровадженню кіберфізичних систем, Інтернету речей (IoT), великих даних (Big Data) та штучного інтелекту (AI). Основні характеристики Industry 4.0 включають автоматизовані процеси, керування якими здійснюється в режимі реального часу із врахуванням змінних зовнішніх умов. Завдяки інтеграції IoT пристроїв і хмарних платформ,

підприємства отримують можливість швидкого аналізу даних, оптимізації операцій та підвищення продуктивності.

Одним із ключових елементів Industry 4.0 є автоматизовані системи моніторингу, які забезпечують збір, обробку та аналіз телеметричних даних у режимі реального часу. Ці системи дозволяють виявляти відхилення у параметрах роботи обладнання, прогнозувати потенційні збої та забезпечувати стабільність технологічних процесів. У їхній основі лежать сучасні алгоритми машинного навчання, які здатні працювати з великими обсягами даних, розпізнавати шаблони та ідентифікувати аномалії.

Алгоритми пошуку аномалій є важливою складовою автоматизованих систем моніторингу. Вони дозволяють виявляти небажані відхилення в роботі обладнання чи процесів, які можуть сигналізувати про потенційні проблеми. Серед найбільш поширених алгоритмів виділяють наступні:

- Spike&Dip Detection — виявлення різких коливань параметрів, які можуть свідчити про аномальні стани або несправності обладнання.
- Change Point Detection — аналіз змін у трендах даних, що дозволяє виявити довгострокові зрушення у поведінці системи.
- Manual Range Checks — визначення порогових значень для ключових параметрів, які є критичними для стабільності процесів.
- Z-Score Analysis — оцінка статистичних відхилень від середніх значень, що дозволяє ідентифікувати малопомітні, але важливі аномалії.

У сучасних автоматизованих системах моніторингу ці алгоритми інтегруються з хмарними платформами, такими як Azure, для забезпечення масштабованої обробки даних у реальному часі. Наприклад, Azure Stream Analytics дозволяє ефективно комбінувати дані з різних джерел, виконувати їхню попередню обробку та передавати результати для подальшої візуалізації або реагування.

Інтеграція хмарних технологій надає автоматизованим системам моніторингу гнучкість і надійність. Завдяки таким рішенням, як Azure IoT Hub, забезпечується стабільна комунікація між пристроями, зберігання даних у

Azure Blob Storage та можливість аналізу інформації у Power BI. Це дозволяє підприємствам здійснювати детальний аналіз виробничих даних, отримувати візуалізовані звіти та приймати обґрунтовані управлінські рішення.

Переваги автоматизації включають також оптимізацію виробничих процесів і логістики. Завдяки виявленню аномалій та прогнозуванню збоїв, підприємства можуть своєчасно планувати технічне обслуговування, уникати незапланованих простоїв та підвищувати ефективність операцій. Наприклад, роботизовані системи з комп'ютерним зором і алгоритмами аналізу даних здатні автоматично транспортувати продукцію, оптимізуючи маршрути та забезпечуючи високу точність у логістичних операціях.

Узагальнюючи, автоматизовані системи моніторингу та алгоритми пошуку аномалій є невід'ємними складовими Industry 4.0. Їхнє впровадження дозволяє досягти високого рівня автоматизації, забезпечити точність у моніторингу та прогнозуванні, підвищити продуктивність і безпеку виробництва.

СПИСОК ЛІТЕРАТУРИ

1. Explainers, M. "What are Industry 4.0, the Fourth Industrial Revolution, and 4IR." (2022).

2. Gregolinska, E., et al. "Capturing the True Value of Industry 4.0, McKinsey & Company." (2022).

3. ItEnterprise : вебсайт URL : https://www.it.ua/knowledge-base/technology-innovation/industry-4 (дата звернення 17.06.2024).

4. Загальні відомості про IoT: вебсайт URL : https://learn.ztu.edu.ua/pluginfile.php/272132/mod_resource/content/1/ШІ_Л-11_КЗір_2022.pdf#:~:text=ЗІР%20–%20джерело%20інформації%20про%20відстані%20і%20розміри%20об%27єктів.&text=Комп%27ютерний%20зір%20або%20Комп,стеження%20та%20класифікацію%20об%27єктів. (дата звернення 18.06.2024).

ДОДАТОК В

Текст програми

BackgroundQueue.cs

```
using System;
using System.Threading;
using System.Threading.Tasks;
namespace SmartMeterSimulator
{
   public class BackgroundQueue
   {
      private Task previousTask = Task.FromResult(true);
      private object key = new object();
      public Task QueueTask(Action action)
      {
         lock (key)
         {
            previousTask = previousTask.ContinueWith(t => action()
               , CancellationToken.None
               , TaskContinuationOptions.None
               , TaskScheduler.Default);
            return previousTask;}}}}
```

DeviceManager.cs

```
using System;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Azure.Devices.Provisioning.Client;
using Microsoft.Azure.Devices.Provisioning.Client.Transport;
using Microsoft.Azure.Devices.Shared;

namespace SmartMeterSimulator
```

```
{
    class DeviceManager
    {
        /// <summary>
        /// Register a single device with the device provisioning service.
        /// </summary>
        /// <param name=«enrollmentKey»>Group Enrollment Key</param>
        /// <param name=«idScope»>DPS Service ID Scope</param>
        /// <param name=«deviceId»>Device Id of the device being registered</param>
        /// <returns></returns>
        public async static Task<SmartMeterDevice> RegisterDeviceAsync(string enrollmentKey, string idScope, string deviceId)
        {
            var globalEndpoint = «global.azure-devices-provisioning.net»;
            SmartMeterDevice device = null;

            //TODO: 1. Derive a device key from a combination of the group enrollment key and the device id
            var primaryKey = ComputeDerivedSymmetricKey(enrollmentKey, deviceId);

            //TODO: 2. Create symmetric key with the generated primary key
            using (var security = new SecurityProviderSymmetricKey(deviceId, primaryKey, null))
            using (var transportHandler = new ProvisioningTransportHandlerMqtt())
            {
                //TODO: 3. Create a Provisioning Device Client
                var client = ProvisioningDeviceClient.Create(globalEndpoint, idScope, security, transportHandler);

                //TODO: 4. Register the device using the symmetric key and MQTT
                DeviceRegistrationResult result = await client.RegisterAsync();

                //TODO: 5. Populate the device provisioning details
                device = new SmartMeterDevice()
                {
                    AuthenticationKey = primaryKey,
                    DeviceId = deviceId,
```

```
                IoTHubHostName = result.AssignedHub
            };
        }

        //return the device
        return device;
    }

    /// <summary>
    /// Compute a symmetric key for the provisioned device from the enrollment group symmetric key used in attestation.
    /// </summary>
    /// <param name=«enrollmentKey»>Enrollment group symmetric key.</param>
    /// <param name=«deviceId»>The device Id of the key to create.</param>
    /// <returns>The key for the specified device Id registration in the enrollment group.</returns>
    /// <seealso>
    /// https://docs.microsoft.com/en-us/azure/iot-edge/how-to-auto-provision-symmetric-keys?view=iotedge-2018-06#derive-a-device-key
    /// </seealso>
    private static string ComputeDerivedSymmetricKey(string enrollmentKey, string deviceId)
    {
        if (string.IsNullOrWhiteSpace(enrollmentKey))
        {
            return enrollmentKey;
        }

        var key = ««;
        using (var hmac = new HMACSHA256(Convert.FromBase64String(enrollmentKey)))
        {
            key = Convert.ToBase64String(hmac.ComputeHash(Encoding.UTF8.GetBytes(deviceId)));
        }

        return key;
    }
}}
```

ListViewColumnSorter.cs

```
using System.Collections;
using System.Windows.Forms;

namespace SmartMeterSimulator
{
    public class ListViewColumnSorter : IComparer
    {
        /// <summary>
        /// Specifies the column to be sorted
        /// </summary>
        private int ColumnToSort;
        /// <summary>
        /// Specifies the order in which to sort (i.e. 'Ascending').
        /// </summary>
        private SortOrder OrderOfSort;
        /// <summary>
        /// Case insensitive comparer object
        /// </summary>
        private CaseInsensitiveComparer ObjectCompare;

        /// <summary>
        /// Class constructor.  Initializes various elements
        /// </summary>
        public ListViewColumnSorter()
        {
            // Initialize the column to '0'
            ColumnToSort = 0;

            // Initialize the sort order to 'none'
            OrderOfSort = SortOrder.None;

            // Initialize the CaseInsensitiveComparer object
```

```
            ObjectCompare = new CaseInsensitiveComparer();
        }

        /// <summary>
        /// This method is inherited from the IComparer interface.  It compares the two objects passed using a case insensitive comparison.
        /// </summary>
        /// <param name=«x»>First object to be compared</param>
        /// <param name=«y»>Second object to be compared</param>
        /// <returns>The result of the comparison. «0» if equal, negative if 'x' is less than 'y' and positive if 'x' is greater than 'y'</returns>
        public int Compare(object x, object y)
        {
            int compareResult;
            ListViewItem listviewX, listviewY;

            // Cast the objects to be compared to ListViewItem objects
            listviewX = (ListViewItem)x;
            listviewY = (ListViewItem)y;

            // Compare the two items
            compareResult = ObjectCompare.Compare(listviewX.SubItems[ColumnToSort].Text, listviewY.SubItems[ColumnToSort].Text);

            // Calculate correct return value based on object comparison
            if (OrderOfSort == SortOrder.Ascending)
            {
                // Ascending sort is selected, return normal result of compare operation
                return compareResult;
            }
            else if (OrderOfSort == SortOrder.Descending)
            {
                // Descending sort is selected, return negative result of compare operation
                return (-compareResult);
            }
```

```
    else
    {
        // Return '0' to indicate they are equal
        return 0;
    }
}

/// <summary>
/// Gets or sets the number of the column to which to apply the sorting operation (Defaults to '0').
/// </summary>
public int SortColumn
{
    set
    {
        ColumnToSort = value;
    }
    get
    {
        return ColumnToSort;
    }
}

/// <summary>
/// Gets or sets the order of sorting to apply (for example, 'Ascending' or 'Descending').
/// </summary>
public SortOrder Order
{
    set
    {
        OrderOfSort = value;
    }
    get
    {
        return OrderOfSort;
```

```
        }
    }

}
}
```

Program.cs

```
using System;
using System.Windows.Forms;

namespace SmartMeterSimulator
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

Sensor.cs

```
using System;
using System.Text;
using Microsoft.Azure.Devices.Client;
```

```
using Newtonsoft.Json;

namespace SmartMeterSimulator
{
    /// <summary>
    /// A sensor represents a Smart Meter in the simulator with multiple metrics.
    /// </summary>
    class Sensor
    {
        private DeviceClient DeviceClient;
        private string IotHubUri { get; set; }
        public string DeviceId { get; set; }
        public string DeviceKey { get; set; }
        public DeviceState State { get; set; }
        public string StatusWindow { get; set; }
        public string ReceivedMessage { get; set; }
        public double? ReceivedTemperatureSetting { get; set; }

        /// <summary>
        /// Температура - існуюча логіка
        /// </summary>
        public double CurrentTemperature
        {
            get
            {
                double avgTemperature = 60;
                Random rand = new Random();

                double currentTemperature = avgTemperature + rand.Next(-10, 10);

                // Аномалія з імовірністю ~1%
                if (rand.NextDouble() < 0.01)
                {
                    currentTemperature = rand.Next(80, 100);
```

```
        }

        if (ReceivedTemperatureSetting.HasValue)
        {
            currentTemperature = ReceivedTemperatureSetting.Value;
        }

        // Визначення стану (Cold, Normal, Hot)
        if (currentTemperature <= 68)
            TemperatureIndicator = SensorState.Cold;
        else if (currentTemperature < 72)
            TemperatureIndicator = SensorState.Normal;
        else
            TemperatureIndicator = SensorState.Hot;

        return currentTemperature;
    }
}

public SensorState TemperatureIndicator { get; set; }

/// <summary>
/// Напруга - існуюча логіка
/// </summary>
public double CurrentVoltage
{
    get
    {
        double avgVoltage = 0.002;
        Random rand = new Random();

        double currentVoltage = avgVoltage + rand.NextDouble() * 0.05;

        // Аномалія з імовірністю ~1%
```

```
        if (rand.NextDouble() < 0.01)
        {
            // Аномально високий діапазон
            currentVoltage = rand.NextDouble() * 0.2 + 0.8;
        }

        return currentVoltage;
    }
}

// ----------------- ДОДАТКОВІ 8 ПОКАЗНИКІВ ------------------------

/// <summary>
/// Вологість (Humidity), діапазон 0..100%
/// Аномалія ~ 2% (значення > 120)
/// </summary>
public double CurrentHumidity
{
    get
    {
        Random rand = new Random();
        double baseHumidity = rand.NextDouble() * 100.0; // [0..100]
        if (rand.NextDouble() < 0.02) // 2% chance anomaly
        {
            // Аномальна вологість
            baseHumidity = 120 + rand.NextDouble() * 30; // 120..150
        }
        return baseHumidity;
    }
}

/// <summary>
/// Тиск (Pressure), нехай приблизно діапазон 700..800 (умовні одиниці)
/// Аномалія ~ 1.5%
```

```
/// </summary>
public double CurrentPressure
{
    get
    {
        Random rand = new Random();
        double basePressure = 700 + rand.NextDouble() * 100; // 700..800
        if (rand.NextDouble() < 0.015) // 1.5%
        {
            basePressure = 900 + rand.NextDouble() * 100; // аномал. 900..1000
        }
        return basePressure;
    }
}

/// <summary>
/// Рівень CO2 (ppm). Звичайно ~ 400..800
/// Аномалія: >2000 ppm з імовірністю 0.5%
/// </summary>
public double CurrentCo2Level
{
    get
    {
        Random rand = new Random();
        double co2 = 400 + rand.NextDouble() * 400; // 400..800
        if (rand.NextDouble() < 0.005) // 0.5% аномалія
        {
            co2 = 2000 + rand.NextDouble() * 1000; // 2000..3000
        }
        return co2;
    }
}

/// <summary>
```

```
/// Освітленість (LightLevel), припустимо 0..10000 лк
/// Аномалія: різке >30000
/// </summary>
public double CurrentLightLevel
{
    get
    {
        Random rand = new Random();
        double light = rand.NextDouble() * 10000.0;
        if (rand.NextDouble() < 0.01) // 1%
        {
            light = 30000 + rand.NextDouble() * 20000; // 30k..50k
        }
        return light;
    }
}

/// <summary>
/// Вібрація (Vibration), від 0..5 (умовні одиниці)
/// Аномалія 2% (значення > 10)
/// </summary>
public double CurrentVibration
{
    get
    {
        Random rand = new Random();
        double vib = rand.NextDouble() * 5.0; // 0..5
        if (rand.NextDouble() < 0.02)
        {
            vib = 10 + rand.NextDouble() * 5; // 10..15
        }
        return vib;
    }
}
```

```
/// <summary>
/// Шум (NoiseLevel), 0..120 дБ
/// Аномалія: >160 дБ  (неможливо, але для прикладу)
/// Імовірність 0.5%
/// </summary>
public double CurrentNoiseLevel
{
    get
    {
        Random rand = new Random();
        double noise = rand.NextDouble() * 120.0; // 0..120 дБ
        if (rand.NextDouble() < 0.005)
        {
            noise = 160 + rand.NextDouble() * 40; // 160..200
        }
        return noise;
    }
}

/// <summary>
/// Заряд батареї (Battery), від 0..100 %
/// Аномалія: стрибок <0 чи >110 з імовірністю 1%
/// </summary>
public double CurrentBattery
{
    get
    {
        Random rand = new Random();
        double battery = rand.NextDouble() * 100; // 0..100
        if (rand.NextDouble() < 0.01)
        {
            // Аномальна
            battery = -10 + rand.NextDouble() * 150; // -10..140
```

```
        }
        return battery;
    }
}

/// <summary>
/// Пил (DustParticles), 0..500 мкг/м3
/// Аномалія: >1000
/// </summary>
public double CurrentDustParticles
{
    get
    {
        Random rand = new Random();
        double dust = rand.NextDouble() * 500; // 0..500
        if (rand.NextDouble() < 0.015) // 1.5%
        {
            dust = 1000 + rand.NextDouble() * 500; // 1000..1500
        }
        return dust;
    }
}

// ----------------------------------------

public Sensor(string deviceId)
{
    DeviceId = deviceId;
}

public void SetRegistrationInformation(string iotHubUri, string deviceKey)
{
    IotHubUri = iotHubUri;
    DeviceKey = deviceKey;
```

```
    State = DeviceState.Registered;
}
public void InstallDevice(string statusWindow)
{
    StatusWindow = statusWindow;
    State = DeviceState.Installed;
}

public void ConnectDevice()
{
    DeviceClient = DeviceClient.Create(
        IotHubUri,
        new DeviceAuthenticationWithRegistrySymmetricKey(DeviceId, DeviceKey)
    );

    State = DeviceState.Connected;
}

public void DisconnectDevice()
{
    DeviceClient = null;
    State = DeviceState.Registered;
}

/// <summary>
/// Надсилає повідомлення з 10-ма показниками в IoT Hub
/// </summary>
public async void SendMessageAsync()
{
    var telemetryDataPoint = new
    {
        id = DeviceId,
        time = DateTime.UtcNow.ToString(«o»),
```

```
        // Оригінальні 2
        temp = CurrentTemperature,
        voltage = CurrentVoltage,

        // Додаткові 8
        humidity = CurrentHumidity,
        pressure = CurrentPressure,
        co2 = CurrentCo2Level,
        light = CurrentLightLevel,
        vibration = CurrentVibration,
        noise = CurrentNoiseLevel,
        battery = CurrentBattery,
        dust = CurrentDustParticles
    };

    var messageString = JsonConvert.SerializeObject(telemetryDataPoint);
    var message = new Message(Encoding.ASCII.GetBytes(messageString));

    if (DeviceClient != null)
        await DeviceClient.SendEventAsync(message);
}

public async void ReceiveMessageAsync()
{
    if (DeviceClient == null)
        return;

    try
    {
        Message receivedMessage = await DeviceClient?.ReceiveAsync();
        if (receivedMessage == null)
        {
            ReceivedMessage = null;
            return;
```

```
            }

            ReceivedMessage = Encoding.ASCII.GetString(receivedMessage.GetBytes());
            if (double.TryParse(ReceivedMessage, out var requestedTemperature))
            {
                ReceivedTemperatureSetting = requestedTemperature;
            }
            else
            {
                ReceivedTemperatureSetting = null;
            }

            await DeviceClient?.CompleteAsync(receivedMessage);
        }
        catch (Exception)
        {
            // ...
        }
    }
}

public enum DeviceState
{
    New,
    Installed,
    Registered,
    Connected,
    Transmit
}

public enum SensorState
{
    Cold,
    Normal,
```

```
        Hot
    }
}
```

SmartMeterDevice.cs

```
namespace SmartMeterSimulator
{
    public class SmartMeterDevice
    {
        public string DeviceId { get; set; }

        public string IoTHubHostName { get; set; }

        public string AuthenticationKey { get; set; }
    }
}
```

MainForm.cs

```
using System;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SmartMeterSimulator
{
    public partial class MainForm : Form
    {
```

```
//milliseconds to delay device transmit
private int delay = 1000;

//Queued Background Worker
private BackgroundQueue queue;
private bool stopQueue;

//Delegate to set device data back from DoWork()
delegate void UpdateDisplayCallback(Sensor sensor);

//Class to custom sort ListView
private ListViewColumnSorter lvwColumnSorter;

public MainForm()
{
   InitializeComponent();

   //Instantiate Queue Worker
   queue = new BackgroundQueue();
   stopQueue = false;

   // Create an instance of a ListView column sorter and assign it
   // to the ListView control.
   lvwColumnSorter = new ListViewColumnSorter();
   lvSensorData.ListViewItemSorter = lvwColumnSorter;
   lvwColumnSorter.SortColumn = 2; //TimeStamp column
   lvwColumnSorter.Order = SortOrder.Descending;
   lvSensorData.Sort();

   //Ensure IoT Edge Gateway device root certificate is trusted
   //DeviceManager.InstallCACert();
}

private void btnRegister_Click(object sender, EventArgs e)
```

```
{
    //There are 10 devices (Sensors) in this sample, Device0 - Device9
    //Provision installed devices with a Device Key
    //Use buttons as a container for each device
    try
    {
        //Loop through the building windows (Buttons)
        foreach (Button button in pDevices.Controls)
        {
            if (button.Tag == null)
                continue;

            var sensor = (Sensor)button.Tag;
            if (sensor.State == DeviceState.Installed)
            {
                string deviceId = button.Name;

                Task<SmartMeterDevice> taskResult = Task.Run(() =>
DeviceManager.RegisterDeviceAsync(txtGroupEnrollmentKey.Text, txtIdScope.Text, deviceId));
                var device = taskResult.Result;

                //Create a new device (Sensor) object, embed its unique device key
                sensor.SetRegistrationInformation(device.IoTHubHostName, device.AuthenticationKey);

                //Change each button color from Yellow to Cyan
                button.BackColor = Color.Cyan;
            }
            else
            {
                //disable uninstalled devices
                button.Enabled = false;
            }
        }
        btnRegister.Enabled = false;
```

```
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void InstallDevice(object sender, EventArgs e)
{
    //Clicking on a button window represents installing the device locally
    //by marrying the device to its matching Transmit Status Window

    //Get the button control which is the container for this device
    Button button = (Button)sender;

    //Set matching Transmit Status Window name, this window shows
    //when a device is transmitting temperature data to IoT Hubs
    string statusId = «Status» + button.Name.Last();
    var sensor = new Sensor(button.Name);
    button.Tag = sensor;

    //Install the device
    sensor.InstallDevice(statusId);

    //Change the color to Yellow to indicate the device has been installed
    button.BackColor = Color.Yellow;
    button.Click -= InstallDevice;
}

private void btnConnect_Click(object sender, EventArgs e)
{
    //Connect all of the activated devices and initiate an event loop
    // for each connected device to transmit data on regular interval
    stopQueue = false;
```

```
    foreach (Button button in pDevices.Controls)
    {
        if (button.Tag == null)
            continue;

        //Get the sensor object from the button container
        Sensor sensor = (Sensor)button.Tag;

        //Select devices which have been installed and activated
        if (sensor.State == DeviceState.Registered || button.BackColor == Color.Cyan)
        {
            //Connect the device, puts into Ready State
            sensor.ConnectDevice();

            //add hover event for tool tip for device info
            button.MouseHover += Device_Hover;

            //Add Sensor to loop
            if (!stopQueue)
                queue.QueueTask(() => DoWork(sensor));
        }
    }
}

private void btnDisconnect_Click(object sender, EventArgs e)
{
    //Stop Loop
    stopQueue = true;

    foreach (Button button in pDevices.Controls)
    {
        if (button.Tag == null)
            continue;
```

```
        button.MouseHover -= Device_Hover;

        //Retrieve the sensor
        Sensor sensor = (Sensor)button.Tag;

        if (sensor.State == DeviceState.Connected || sensor.State == DeviceState.Transmit)
        {
            sensor.DisconnectDevice();
            button.BackColor = Color.Cyan;

            if (!string.IsNullOrEmpty(sensor.StatusWindow))
                pStatus.Controls[sensor.StatusWindow].Visible = false;
        }
    }
}

private void DoWork(Sensor sensor)
{
    switch (sensor.State)
    {
        case DeviceState.Connected:
            Thread.Sleep(delay / 2);
            sensor.State = DeviceState.Transmit;
            sensor.SendMessageAsync();
            break;
        case DeviceState.Transmit:
            Thread.Sleep(delay / 2);
            sensor.State = DeviceState.Connected;
            break;
    }

    if (sensor.State == DeviceState.Connected || sensor.State == DeviceState.Transmit)
    {
```

```
        sensor.ReceiveMessageAsync();
    }

    WorkComplete(sensor);
}

private void WorkComplete(Sensor sensor)
{
    try
    {
        Button button = (Button)pDevices.Controls[sensor.DeviceId];

        if (button.InvokeRequired)
        {
            UpdateDisplayCallback d = new UpdateDisplayCallback(WorkComplete);
            Invoke(d, new object[] { sensor });
        }
        else
        {
            switch (sensor.State)
            {
                case DeviceState.Transmit:
                    pStatus.Controls[sensor.StatusWindow].Visible = true;

                    //Set the Sensor's Temperature color on the building
                    switch (sensor.TemperatureIndicator)
                    {
                        case SensorState.Cold:
                            button.BackColor = Color.Blue;
                            break;
                        case SensorState.Normal:
                            button.BackColor = Color.Green;
                            break;
                        case SensorState.Hot:
```

```
        button.BackColor = Color.Red;
        break;
}

//=======================================
// NEW: Gathering all 10 metrics from sensor
//=======================================

string deviceId = sensor.DeviceId;
string currTemp = String.Format(«{0:0.0}», sensor.CurrentTemperature);
string currVoltage = String.Format(«{0:0.000}», sensor.CurrentVoltage);
string currHumidity = String.Format(«{0:0.0}», sensor.CurrentHumidity);
string currPressure = String.Format(«{0:0.0}», sensor.CurrentPressure);
string currCo2 = String.Format(«{0:0.0}», sensor.CurrentCo2Level);
string currLight = String.Format(«{0:0.0}», sensor.CurrentLightLevel);
string currVibration = String.Format(«{0:0.0}», sensor.CurrentVibration);
string currNoise = String.Format(«{0:0.0}», sensor.CurrentNoiseLevel);
string currBattery = String.Format(«{0:0.0}», sensor.CurrentBattery);
string currDust = String.Format(«{0:0.0}», sensor.CurrentDustParticles);

string timeStamp = DateTime.UtcNow
  .ToLocalTime()
  .ToString(«s»);

// Add sensor data to the list
// (Make sure your ListView has 12 columns in the correct order.)
var item1 = new ListViewItem(new[]
{
    deviceId,
    currTemp,
    currVoltage,
    currHumidity,
    currPressure,
    currCo2,
```

```
                    currLight,
                    currVibration,
                    currNoise,
                    currBattery,
                    currDust,
                    timeStamp
                });

                lvSensorData.Items.Add(item1);
                break;

            case DeviceState.Connected:
                pStatus.Controls[sensor.StatusWindow].Visible = false;
                break;
        }

        button.Tag = sensor;

        if (!string.IsNullOrWhiteSpace(sensor.ReceivedMessage))
        {
            var item1 = new ListViewItem(new[] { sensor.DeviceId, $»Message: {sensor.ReceivedMessage}», DateTime.UtcNow.ToString(«s») });
            item1.BackColor = Color.Yellow;
            lvSensorData.Items.Add(item1);
            sensor.ReceivedMessage = null;
        }

        if (!stopQueue)
            queue.QueueTask(() => DoWork(sensor));
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
```

```
        }
    }

    private void Device_Hover(object sender, EventArgs e)
    {
        Button button = (Button)sender;

        if (button.Tag != null)
        {
            Sensor sensor = (Sensor)button.Tag;

            string tempIndicator = sensor.TemperatureIndicator.ToString();

            StringBuilder sb = new StringBuilder();
            sb.AppendLine(«Device Id: « + sensor.DeviceId);
            sb.AppendLine(«Device Key: « + sensor.DeviceKey);
            sb.AppendLine();
            sb.AppendLine(«Temperature: « + string.Format(«{0:0.0}», sensor.CurrentTemperature));
            sb.AppendLine(«Indicator: « + tempIndicator);

            ttDeviceStatus.Show(sb.ToString(), button);
        }
    }
  }
}
```

AlertToEmail StreamAnalyticJob Query

```
-----------------------------------------------------------------------------
-- 1. InitialData
--    Читаємо одне джерело (один потік), 10 полів. Додавайте TIMESTAMP BY за потреби
-----------------------------------------------------------------------------
WITH
```

```
InitialData AS
(
    SELECT
        CAST(temp      AS FLOAT) AS raw_temp,
        CAST(voltage   AS FLOAT) AS raw_voltage,
        CAST(humidity  AS FLOAT) AS raw_humidity,
        CAST(pressure  AS FLOAT) AS raw_pressure,
        CAST(co2       AS FLOAT) AS raw_co2,
        CAST(light     AS FLOAT) AS raw_light,
        CAST(vibration AS FLOAT) AS raw_vibration,
        CAST(noise     AS FLOAT) AS raw_noise,
        CAST(battery   AS FLOAT) AS raw_battery,
        CAST(dust      AS FLOAT) AS raw_dust,

        id,
        EventEnqueuedUtcTime AS event_time
    FROM [DataFromSensors]
    -- TIMESTAMP BY event_time
),

-------------------------------------------------------------------------------
-- 2. Agg30s
--    30-секундне згладження для всіх 10 полів:
--    - AVG(...)
--    - SUM(...^2) (для Z-score)
--    - COUNT(*)
-------------------------------------------------------------------------------
Agg30s AS
(
    SELECT
        System.Timestamp() AS window_end_30s,
        id,

        AVG(raw_temp)          AS avg_temp_30s,
```

```
        AVG(raw_voltage)        AS avg_voltage_30s,
        AVG(raw_humidity)       AS avg_humidity_30s,
        AVG(raw_pressure)       AS avg_pressure_30s,
        AVG(raw_co2)            AS avg_co2_30s,
        AVG(raw_light)          AS avg_light_30s,
        AVG(raw_vibration)      AS avg_vibration_30s,
        AVG(raw_noise)          AS avg_noise_30s,
        AVG(raw_battery)        AS avg_battery_30s,
        AVG(raw_dust)           AS avg_dust_30s,

        -- Для Z-score рахуємо суму квадратів
        SUM(raw_temp*raw_temp)           AS sum_sq_temp,
        SUM(raw_voltage*raw_voltage)     AS sum_sq_volt,
        SUM(raw_humidity*raw_humidity)   AS sum_sq_humidity,
        SUM(raw_pressure*raw_pressure)   AS sum_sq_pressure,
        SUM(raw_co2*raw_co2)             AS sum_sq_co2,
        SUM(raw_light*raw_light)         AS sum_sq_light,
        SUM(raw_vibration*raw_vibration) AS sum_sq_vibration,
        SUM(raw_noise*raw_noise)         AS sum_sq_noise,
        SUM(raw_battery*raw_battery)     AS sum_sq_battery,
        SUM(raw_dust*raw_dust)           AS sum_sq_dust,

        COUNT(*) AS cnt_30s

    FROM InitialData
    GROUP BY TUMBLINGWINDOW(second, 30), id
),

--------------------------------------------------------------------------------
-- 3. AnomalyDetectionStep
--    Викликаємо 4 методи для кожного з 10 полів:
--    A) Spike&Dip
--    B) ChangePoint
--    C) Manual Range Check
```

```
--    D) Z-score
--------------------------------------------------------------------------------
AnomalyDetectionStep AS
(
   SELECT
      window_end_30s,
      id,

      -- Середні (для подальшого зручного виводу)
      avg_temp_30s,
      avg_voltage_30s,
      avg_humidity_30s,
      avg_pressure_30s,
      avg_co2_30s,
      avg_light_30s,
      avg_vibration_30s,
      avg_noise_30s,
      avg_battery_30s,
      avg_dust_30s,

      ------------------------------------------------------------------------
      -- A) Spike&Dip для кожного поля (threshold=80, sensitivity=60, 'spikesanddips')
      ------------------------------------------------------------------------
      AnomalyDetection_SpikeAndDip(avg_temp_30s,    80, 60, 'spikesanddips')
       OVER(PARTITION BY id LIMIT DURATION(second, 60)) AS spike_temp_obj,

      AnomalyDetection_SpikeAndDip(avg_voltage_30s, 80, 60, 'spikesanddips')
       OVER(PARTITION BY id LIMIT DURATION(second, 60)) AS spike_volt_obj,

      AnomalyDetection_SpikeAndDip(avg_humidity_30s,80, 60, 'spikesanddips')
       OVER(PARTITION BY id LIMIT DURATION(second, 60)) AS spike_humidity_obj,

      AnomalyDetection_SpikeAndDip(avg_pressure_30s,80, 60, 'spikesanddips')
       OVER(PARTITION BY id LIMIT DURATION(second, 60)) AS spike_pressure_obj,
```

```
AnomalyDetection_SpikeAndDip(avg_co2_30s,     80, 60, 'spikesanddips')
  OVER(PARTITION BY id LIMIT DURATION(second, 60)) AS spike_co2_obj,

AnomalyDetection_SpikeAndDip(avg_light_30s,   80, 60, 'spikesanddips')
  OVER(PARTITION BY id LIMIT DURATION(second, 60)) AS spike_light_obj,

AnomalyDetection_SpikeAndDip(avg_vibration_30s,80,60,'spikesanddips')
  OVER(PARTITION BY id LIMIT DURATION(second, 60)) AS spike_vibration_obj,

AnomalyDetection_SpikeAndDip(avg_noise_30s,   80, 60, 'spikesanddips')
  OVER(PARTITION BY id LIMIT DURATION(second, 60)) AS spike_noise_obj,

AnomalyDetection_SpikeAndDip(avg_battery_30s, 80, 60, 'spikesanddips')
  OVER(PARTITION BY id LIMIT DURATION(second, 60)) AS spike_battery_obj,

AnomalyDetection_SpikeAndDip(avg_dust_30s,    80, 60, 'spikesanddips')
  OVER(PARTITION BY id LIMIT DURATION(second, 60)) AS spike_dust_obj,

-------------------------------------------------------------------------
-- B) ChangePoint для кожного поля
-------------------------------------------------------------------------
AnomalyDetection_ChangePoint(avg_temp_30s,    80, 60)
  OVER(PARTITION BY id LIMIT DURATION(second, 120)) AS change_temp_obj,

AnomalyDetection_ChangePoint(avg_voltage_30s, 80, 60)
  OVER(PARTITION BY id LIMIT DURATION(second, 120)) AS change_volt_obj,

AnomalyDetection_ChangePoint(avg_humidity_30s,80,60)
  OVER(PARTITION BY id LIMIT DURATION(second, 120)) AS change_humidity_obj,

AnomalyDetection_ChangePoint(avg_pressure_30s,80,60)
  OVER(PARTITION BY id LIMIT DURATION(second, 120)) AS change_pressure_obj,
```

```
AnomalyDetection_ChangePoint(avg_co2_30s,     80,60)
  OVER(PARTITION BY id LIMIT DURATION(second, 120)) AS change_co2_obj,

AnomalyDetection_ChangePoint(avg_light_30s,   80,60)
  OVER(PARTITION BY id LIMIT DURATION(second, 120)) AS change_light_obj,

AnomalyDetection_ChangePoint(avg_vibration_30s,80,60)
  OVER(PARTITION BY id LIMIT DURATION(second, 120)) AS change_vibration_obj,

AnomalyDetection_ChangePoint(avg_noise_30s,   80,60)
  OVER(PARTITION BY id LIMIT DURATION(second, 120)) AS change_noise_obj,

AnomalyDetection_ChangePoint(avg_battery_30s,80,60)
  OVER(PARTITION BY id LIMIT DURATION(second, 120)) AS change_battery_obj,

AnomalyDetection_ChangePoint(avg_dust_30s,   80,60)
  OVER(PARTITION BY id LIMIT DURATION(second, 120)) AS change_dust_obj,

-------------------------------------------------------------------------
-- C) Manual Range Check (приклади діапазонів,
--    ПІДСТАНОВКА залежно від вашої логіки)
-------------------------------------------------------------------------
CASE WHEN avg_temp_30s < 0 OR avg_temp_30s>100 THEN 1 ELSE 0 END AS range_flag_temp,
CASE WHEN avg_voltage_30s<0.0 OR avg_voltage_30s>1.0 THEN 1 ELSE 0 END AS range_flag_volt,
CASE WHEN avg_humidity_30s<0 OR avg_humidity_30s>110 THEN 1 ELSE 0 END AS range_flag_humidity,
CASE WHEN avg_pressure_30s<600 OR avg_pressure_30s>1000 THEN 1 ELSE 0 END AS range_flag_pressure,
CASE WHEN avg_co2_30s<300 OR avg_co2_30s>5000 THEN 1 ELSE 0 END AS range_flag_co2,
CASE WHEN avg_light_30s<0 OR avg_light_30s>20000 THEN 1 ELSE 0 END AS range_flag_light,
CASE WHEN avg_vibration_30s>10 THEN 1 ELSE 0 END AS range_flag_vibration,
CASE WHEN avg_noise_30s>140 THEN 1 ELSE 0 END AS range_flag_noise,
CASE WHEN avg_battery_30s<0 OR avg_battery_30s>110 THEN 1 ELSE 0 END AS range_flag_battery,
CASE WHEN avg_dust_30s>1000 THEN 1 ELSE 0 END AS range_flag_dust,

-------------------------------------------------------------------------
```

```
-- D) Z-score (спрощений)
--    1) Спочатку обчислюємо stdev (вже в Agg30s)
-------------------------------------------------------------------------------
CASE WHEN cnt_30s>1 THEN
    SQRT(
      (sum_sq_temp - (avg_temp_30s*avg_temp_30s*cnt_30s))
       / cnt_30s
    )
    ELSE 0
END AS stdev_temp,

CASE WHEN cnt_30s>1 THEN
    SQRT(
      (sum_sq_volt - (avg_voltage_30s*avg_voltage_30s*cnt_30s))
       / cnt_30s
    )
    ELSE 0
END AS stdev_volt,

CASE WHEN cnt_30s>1 THEN
    SQRT(
      (sum_sq_humidity - (avg_humidity_30s*avg_humidity_30s*cnt_30s))
       / cnt_30s
    )
    ELSE 0
END AS stdev_humidity,

CASE WHEN cnt_30s>1 THEN
    SQRT(
      (sum_sq_pressure - (avg_pressure_30s*avg_pressure_30s*cnt_30s))
       / cnt_30s
    )
    ELSE 0
END AS stdev_pressure,
```

```
CASE WHEN cnt_30s>1 THEN
    SQRT(
      (sum_sq_co2 - (avg_co2_30s*avg_co2_30s*cnt_30s))
       / cnt_30s
    )
    ELSE 0
END AS stdev_co2,

CASE WHEN cnt_30s>1 THEN
    SQRT(
      (sum_sq_light - (avg_light_30s*avg_light_30s*cnt_30s))
       / cnt_30s
    )
    ELSE 0
END AS stdev_light,

CASE WHEN cnt_30s>1 THEN
    SQRT(
      (sum_sq_vibration - (avg_vibration_30s*avg_vibration_30s*cnt_30s))
       / cnt_30s
    )
    ELSE 0
END AS stdev_vibration,

CASE WHEN cnt_30s>1 THEN
    SQRT(
      (sum_sq_noise - (avg_noise_30s*avg_noise_30s*cnt_30s))
       / cnt_30s
    )
    ELSE 0
END AS stdev_noise,

CASE WHEN cnt_30s>1 THEN
```

```
            SQRT(
              (sum_sq_battery - (avg_battery_30s*avg_battery_30s*cnt_30s))
               / cnt_30s
            )
            ELSE 0
        END AS stdev_battery,

        CASE WHEN cnt_30s>1 THEN
            SQRT(
              (sum_sq_dust - (avg_dust_30s*avg_dust_30s*cnt_30s))
               / cnt_30s
            )
            ELSE 0
        END AS stdev_dust

    FROM Agg30s
),

-------------------------------------------------------------------------------
-- 4. AnomalyDetectionFinal
--    «Розпаковуємо» (IsAnomaly) + Z-score перевірка (>3)
-------------------------------------------------------------------------------
AnomalyDetectionFinal AS
(
    SELECT
        window_end_30s,
        id,

        avg_temp_30s,
        avg_voltage_30s,
        avg_humidity_30s,
        avg_pressure_30s,
        avg_co2_30s,
        avg_light_30s,
```

```
avg_vibration_30s,
avg_noise_30s,
avg_battery_30s,
avg_dust_30s,

-------------------------------------------------------------------------
-- Spike & Dip flags
-------------------------------------------------------------------------
CAST(GetRecordPropertyValue(spike_temp_obj,    'IsAnomaly') AS BIGINT) AS spike_flag_temp,
CAST(GetRecordPropertyValue(spike_volt_obj,    'IsAnomaly') AS BIGINT) AS spike_flag_volt,
CAST(GetRecordPropertyValue(spike_humidity_obj,'IsAnomaly') AS BIGINT) AS spike_flag_humidity,
CAST(GetRecordPropertyValue(spike_pressure_obj,'IsAnomaly') AS BIGINT) AS spike_flag_pressure,
CAST(GetRecordPropertyValue(spike_co2_obj,     'IsAnomaly') AS BIGINT) AS spike_flag_co2,
CAST(GetRecordPropertyValue(spike_light_obj,   'IsAnomaly') AS BIGINT) AS spike_flag_light,
CAST(GetRecordPropertyValue(spike_vibration_obj,'IsAnomaly')AS BIGINT) AS spike_flag_vibration,
CAST(GetRecordPropertyValue(spike_noise_obj,   'IsAnomaly') AS BIGINT) AS spike_flag_noise,
CAST(GetRecordPropertyValue(spike_battery_obj, 'IsAnomaly') AS BIGINT) AS spike_flag_battery,
CAST(GetRecordPropertyValue(spike_dust_obj,    'IsAnomaly') AS BIGINT) AS spike_flag_dust,

-------------------------------------------------------------------------
-- ChangePoint flags
-------------------------------------------------------------------------
CAST(GetRecordPropertyValue(change_temp_obj,    'IsAnomaly') AS BIGINT) AS change_flag_temp,
CAST(GetRecordPropertyValue(change_volt_obj,    'IsAnomaly') AS BIGINT) AS change_flag_volt,
CAST(GetRecordPropertyValue(change_humidity_obj,'IsAnomaly') AS BIGINT) AS change_flag_humidity,
CAST(GetRecordPropertyValue(change_pressure_obj,'IsAnomaly') AS BIGINT) AS change_flag_pressure,
CAST(GetRecordPropertyValue(change_co2_obj,     'IsAnomaly') AS BIGINT) AS change_flag_co2,
CAST(GetRecordPropertyValue(change_light_obj,   'IsAnomaly') AS BIGINT) AS change_flag_light,
CAST(GetRecordPropertyValue(change_vibration_obj,'IsAnomaly')AS BIGINT) AS change_flag_vibration,
CAST(GetRecordPropertyValue(change_noise_obj,   'IsAnomaly') AS BIGINT) AS change_flag_noise,
CAST(GetRecordPropertyValue(change_battery_obj, 'IsAnomaly') AS BIGINT) AS change_flag_battery,
CAST(GetRecordPropertyValue(change_dust_obj,    'IsAnomaly') AS BIGINT) AS change_flag_dust,

-------------------------------------------------------------------------
```

```
    -- Manual Range flags (вже бінарні)
    -------------------------------------------------------------------------
    range_flag_temp,
    range_flag_volt,
    range_flag_humidity,
    range_flag_pressure,
    range_flag_co2,
    range_flag_light,
    range_flag_vibration,
    range_flag_noise,
    range_flag_battery,
    range_flag_dust,


    -------------------------------------------------------------------------
    -- Z-score flags (>3)
    -------------------------------------------------------------------------
    CASE WHEN stdev_temp<>0      AND ABS((avg_temp_30s - avg_temp_30s)/stdev_temp)>3              THEN 1 ELSE 0 END AS zscore_flag_temp,
    CASE WHEN stdev_volt<>0      AND ABS((avg_voltage_30s - avg_voltage_30s)/stdev_volt)>3        THEN 1 ELSE 0 END AS zscore_flag_volt,
    CASE WHEN stdev_humidity<>0  AND ABS((avg_humidity_30s - avg_humidity_30s)/stdev_humidity)>3  THEN 1 ELSE 0 END AS zscore_flag_humidity,
    CASE WHEN stdev_pressure<>0  AND ABS((avg_pressure_30s - avg_pressure_30s)/stdev_pressure)>3  THEN 1 ELSE 0 END AS zscore_flag_pressure,
    CASE WHEN stdev_co2<>0       AND ABS((avg_co2_30s - avg_co2_30s)/stdev_co2)>3                THEN 1 ELSE 0 END AS zscore_flag_co2,
    CASE WHEN stdev_light<>0     AND ABS((avg_light_30s - avg_light_30s)/stdev_light)>3          THEN 1 ELSE 0 END AS zscore_flag_light,
    CASE WHEN stdev_vibration<>0 AND ABS((avg_vibration_30s - avg_vibration_30s)/stdev_vibration)>3 THEN 1 ELSE 0 END AS zscore_flag_vibration,
    CASE WHEN stdev_noise<>0     AND ABS((avg_noise_30s - avg_noise_30s)/stdev_noise)>3          THEN 1 ELSE 0 END AS zscore_flag_noise,
    CASE WHEN stdev_battery<>0   AND ABS((avg_battery_30s - avg_battery_30s)/stdev_battery)>3    THEN 1 ELSE 0 END AS zscore_flag_battery,
    CASE WHEN stdev_dust<>0      AND ABS((avg_dust_30s - avg_dust_30s)/stdev_dust)>3              THEN 1 ELSE 0 END AS zscore_flag_dust


  FROM AnomalyDetectionStep
```

```
),

-------------------------------------------------------------------------------
-- 5. AggregatedResults (2-хвилинне вікно):
--    Рахуємо, скільки разів за 2 хв кожен метод «спрацював»
-------------------------------------------------------------------------------
AggregatedResults AS
(
   SELECT
      id,
      System.Timestamp() AS TimeBucket,

      -- Spike counts
      COUNT(CASE WHEN spike_flag_temp=1       THEN 1 END) AS spike_temp_count,
      COUNT(CASE WHEN spike_flag_volt=1       THEN 1 END) AS spike_volt_count,
      COUNT(CASE WHEN spike_flag_humidity=1   THEN 1 END) AS spike_humidity_count,
      COUNT(CASE WHEN spike_flag_pressure=1   THEN 1 END) AS spike_pressure_count,
      COUNT(CASE WHEN spike_flag_co2=1        THEN 1 END) AS spike_co2_count,
      COUNT(CASE WHEN spike_flag_light=1      THEN 1 END) AS spike_light_count,
      COUNT(CASE WHEN spike_flag_vibration=1  THEN 1 END) AS spike_vibration_count,
      COUNT(CASE WHEN spike_flag_noise=1      THEN 1 END) AS spike_noise_count,
      COUNT(CASE WHEN spike_flag_battery=1    THEN 1 END) AS spike_battery_count,
      COUNT(CASE WHEN spike_flag_dust=1       THEN 1 END) AS spike_dust_count,

      -- Change counts
      COUNT(CASE WHEN change_flag_temp=1      THEN 1 END) AS change_temp_count,
      COUNT(CASE WHEN change_flag_volt=1      THEN 1 END) AS change_volt_count,
      COUNT(CASE WHEN change_flag_humidity=1  THEN 1 END) AS change_humidity_count,
      COUNT(CASE WHEN change_flag_pressure=1  THEN 1 END) AS change_pressure_count,
      COUNT(CASE WHEN change_flag_co2=1       THEN 1 END) AS change_co2_count,
      COUNT(CASE WHEN change_flag_light=1     THEN 1 END) AS change_light_count,
      COUNT(CASE WHEN change_flag_vibration=1 THEN 1 END) AS change_vibration_count,
      COUNT(CASE WHEN change_flag_noise=1     THEN 1 END) AS change_noise_count,
      COUNT(CASE WHEN change_flag_battery=1   THEN 1 END) AS change_battery_count,
```

```
COUNT(CASE WHEN change_flag_dust=1      THEN 1 END) AS change_dust_count,

-- Range counts
COUNT(CASE WHEN range_flag_temp=1       THEN 1 END) AS range_temp_count,
COUNT(CASE WHEN range_flag_volt=1       THEN 1 END) AS range_volt_count,
COUNT(CASE WHEN range_flag_humidity=1   THEN 1 END) AS range_humidity_count,
COUNT(CASE WHEN range_flag_pressure=1   THEN 1 END) AS range_pressure_count,
COUNT(CASE WHEN range_flag_co2=1        THEN 1 END) AS range_co2_count,
COUNT(CASE WHEN range_flag_light=1      THEN 1 END) AS range_light_count,
COUNT(CASE WHEN range_flag_vibration=1  THEN 1 END) AS range_vibration_count,
COUNT(CASE WHEN range_flag_noise=1      THEN 1 END) AS range_noise_count,
COUNT(CASE WHEN range_flag_battery=1    THEN 1 END) AS range_battery_count,
COUNT(CASE WHEN range_flag_dust=1       THEN 1 END) AS range_dust_count,

-- Z-score counts
COUNT(CASE WHEN zscore_flag_temp=1      THEN 1 END) AS zscore_temp_count,
COUNT(CASE WHEN zscore_flag_volt=1      THEN 1 END) AS zscore_volt_count,
COUNT(CASE WHEN zscore_flag_humidity=1  THEN 1 END) AS zscore_humidity_count,
COUNT(CASE WHEN zscore_flag_pressure=1  THEN 1 END) AS zscore_pressure_count,
COUNT(CASE WHEN zscore_flag_co2=1       THEN 1 END) AS zscore_co2_count,
COUNT(CASE WHEN zscore_flag_light=1     THEN 1 END) AS zscore_light_count,
COUNT(CASE WHEN zscore_flag_vibration=1 THEN 1 END) AS zscore_vibration_count,
COUNT(CASE WHEN zscore_flag_noise=1     THEN 1 END) AS zscore_noise_count,
COUNT(CASE WHEN zscore_flag_battery=1   THEN 1 END) AS zscore_battery_count,
COUNT(CASE WHEN zscore_flag_dust=1      THEN 1 END) AS zscore_dust_count,

-- Для перегляду середніх
AVG(avg_temp_30s)       AS avg_temp_in_2m,
AVG(avg_voltage_30s)    AS avg_voltage_in_2m,
AVG(avg_humidity_30s)   AS avg_humidity_in_2m,
AVG(avg_pressure_30s)   AS avg_pressure_in_2m,
AVG(avg_co2_30s)        AS avg_co2_in_2m,
AVG(avg_light_30s)      AS avg_light_in_2m,
AVG(avg_vibration_30s)  AS avg_vibration_in_2m,
```

```
        AVG(avg_noise_30s)      AS avg_noise_in_2m,
        AVG(avg_battery_30s)    AS avg_battery_in_2m,
        AVG(avg_dust_30s)       AS avg_dust_in_2m

    FROM AnomalyDetectionFinal
    GROUP BY TUMBLINGWINDOW(minute, 2), id
),

-------------------------------------------------------------------------------
-- 6. FinalSelect
--    Відбираємо 2-хвилинні блоки, де хоча б 1 аномалія
--    (по будь-якому з 10 полів і 4 методів => 40 флагів)
-------------------------------------------------------------------------------
FinalSelect AS
(
    SELECT
        id,
        TimeBucket,

        -- Spike counts
        spike_temp_count, spike_volt_count, spike_humidity_count, spike_pressure_count,
        spike_co2_count,  spike_light_count, spike_vibration_count, spike_noise_count,
        spike_battery_count, spike_dust_count,

        -- Change counts
        change_temp_count, change_volt_count, change_humidity_count, change_pressure_count,
        change_co2_count,  change_light_count, change_vibration_count, change_noise_count,
        change_battery_count, change_dust_count,

        -- Range counts
        range_temp_count, range_volt_count, range_humidity_count, range_pressure_count,
        range_co2_count,  range_light_count, range_vibration_count, range_noise_count,
        range_battery_count, range_dust_count,
```

```
    -- Z-score counts
    zscore_temp_count, zscore_volt_count, zscore_humidity_count, zscore_pressure_count,
    zscore_co2_count,  zscore_light_count, zscore_vibration_count, zscore_noise_count,
    zscore_battery_count, zscore_dust_count,

    -- Середні за 2 хв
    avg_temp_in_2m,    avg_voltage_in_2m,   avg_humidity_in_2m,
    avg_pressure_in_2m,avg_co2_in_2m,       avg_light_in_2m,
    avg_vibration_in_2m,avg_noise_in_2m,    avg_battery_in_2m,
    avg_dust_in_2m

FROM AggregatedResults
WHERE
    (
     -- spike
     spike_temp_count>0       OR spike_volt_count>0
     OR spike_humidity_count>0 OR spike_pressure_count>0
     OR spike_co2_count>0     OR spike_light_count>0
     OR spike_vibration_count>0 OR spike_noise_count>0
     OR spike_battery_count>0   OR spike_dust_count>0

     -- change
     OR change_temp_count>0   OR change_volt_count>0
     OR change_humidity_count>0 OR change_pressure_count>0
     OR change_co2_count>0    OR change_light_count>0
     OR change_vibration_count>0 OR change_noise_count>0
     OR change_battery_count>0   OR change_dust_count>0

     -- range
     OR range_temp_count>0    OR range_volt_count>0
     OR range_humidity_count>0 OR range_pressure_count>0
     OR range_co2_count>0     OR range_light_count>0
     OR range_vibration_count>0 OR range_noise_count>0
     OR range_battery_count>0   OR range_dust_count>0
```

```
        -- zscore
        OR zscore_temp_count>0   OR zscore_volt_count>0
        OR zscore_humidity_count>0 OR zscore_pressure_count>0
        OR zscore_co2_count>0    OR zscore_light_count>0
        OR zscore_vibration_count>0 OR zscore_noise_count>0
        OR zscore_battery_count>0   OR zscore_dust_count>0
       )
)

--------------------------------------------------------------------------------
-- Фінальний вивід
--------------------------------------------------------------------------------
SELECT *
FROM FinalSelect
ORDER BY TimeBucket DESC;
```

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮІК. 505200.011 ПЗ	Пояснювальна записка	69 с., ф. А4
		Графічні документи	
		Презентація	у форматі
			*.ppt

					ГЮІК. 505200.011 ВД		
Змін.	Арк.	№ документа	Підп.	Дата			
Розроб.		Кануніков М.Ю.				Літ.	Аркуш	Аркушів
Перевір.		Сичова О.В.					1	1
Н. контр.		Бронніков А.І.				ХНУРЕ		
Затвер.		Невлюдов І.Ш				Кафедра КІТАР		