ДОДАТОК А

Апробація результатів роботи

ІНТЕЛЕКТУАЛЬНІ SCADA−СИСТЕМИ

В. Я. Коваленко
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: vladyslav.kovalenko4@nure.ua

Анотація: Розглядається роботизоване виробництво, організоване за принципом Lean Production, яке використовує SCADA−системи для управління технологічним процесом. В критичних умовах, коли кожна секунда має значення, введення додаткових завдань може уповільнити процеси та відволікти операторів від основних обов'язків. За наявності систем SCADA з елементами штучного інтелекту дані про процес можуть бути зібрані автоматично, забезпечуючи більш точний та своєчасний аналіз ситуації.

Ключові слова: SCADA−система, технологічний процес, штучний інтелект, роботизоване виробництво.

INTELLECTUAL SCADA−SYSTEMS

Y. Kovalenko
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av., 14
E-mail: vladyslav.kovalenko4@nure.ua

Annotation: We consider robotic production organized according to the Lean Production principle, which uses SCADA− systems to control the technological process. In critical environments where every second counts, the introduction of additional tasks can slow down processes and distract operators from their core duties. With SCADA − systems with artificial intelligence elements, process data can be collected automatically, providing more accurate and timely analysis of the situation.

Keywords: SCADA− system, technological process, artificial intelligence, robotic production.

АКТУАЛЬНІСТЬ РОБОТИ. Бережливе виробництво (Lean Production) − це ефективна система управління, яка дозволяє мінімізувати витрати та зосередитись на оптимізації виробничих процесів. Тим не менш, заощадливий підхід не завжди є найкращим вибором для сучасних роботизованих виробництв, особливо в тих секторах, де необхідно працювати з точними даними та негайно адаптуватися до змін.

Одним із ризиків впровадження Lean Production є спрощення процесів управління відповідно до шаблону, що в умовах швидких змін іноді призводить до суттєвих витрат.

Одним із шляхів удосконалення управління та підвищення ефективності роботизованих виробництв є використання штучного інтелекту, зокрема в системах типу SCADA.

МАТЕРІАЛ І РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ. Без автоматизованих систем роботизоване виробництво ризикує обмежити контроль та точність управління виробничими процесами. Lean Production вимагає постійного збору інформації та аналізу, який часто передається працівникам, які працюють з автоматизованою системою. За наявності систем SCADA або MES, дані про виробничі процеси можуть бути зібрані автоматично, забезпечуючи більш точний та своєчасний аналіз. В умовах критичних виробництв та в умовах воєнного стану іноді

кожна секунда має значення, тому покладення завдань, які можуть виникнути випадково, на операторів автоматизованих систем є ризиком, який потрібно планувати заздалегідь.

Наприклад, при управлінні екологічно небезпечними технологічними об'єктами, складність яких увесь час зростає, оператор повинен опрацювати дедалі більші потоки вхідної інформації. Результатом такого управління може бути зростання кількості інцидентів, аварій і катастроф. Виходом зі такого становища є передання більшої частини функцій обробки інформації від людини-оператора до засобів автоматизації, тобто посилення інтелектуальних здібностей чинних і проектованих систем управління.

SCADA−системи можна розділити на дві групи за ознакою використання методів штучного інтелекту для розв'язання завдань підтримки ухвалення рішень оператором складного об'єкта контролю та управління.

До першої групи входять SCADA-системи, що реалізують традиційні функції моніторингу та управління процесами:

- ведення бази даних реального часу;
- виконання розрахунків;
- графічне представлення даних і параметрів у вигляді мнемосхем, графіків, діаграм тощо;
- попереджувальна сигналізація;
- архівування інформації;
- генерування звітів.

Другу групу складають SCADA-системи, що використовують методи обробки і подання інформації, засновані на знаннях. Вони стали називатися інтелектуальними SCADAсистемами. У функції таких систем входить інтелектуальна інформаційна підтримка людиниоператора під час керування процесами. До числа цих функцій належать:

- ситуаційний аналіз стану об'єкта контролю та управління;
- оперативний пошук дій оператора-управлінця в разі виникнення аномальних і критичних ситуацій;
- діагностика стану технологічного обладнання;
- діагностика стану технологічного процесу;
- логічний аналіз подій;
- логічний аналіз аномальних ситуацій;
- прогноз поведінки процесу в часі та інші;
- захист від несанкціонованих технологічним регламентом дій оперативного персоналу;
- ведення баз даних і знань реального часу; − ведення гіпертекстових баз експлуатаційних і регламентних знань.

Системи першої та другої групи можуть доповнювати одна одну і застосовуватися спільно. Однак, якщо системи першого типу - це базис сучасних систем управління, то системи, що базуються на знаннях, використовуються поки що не часто.

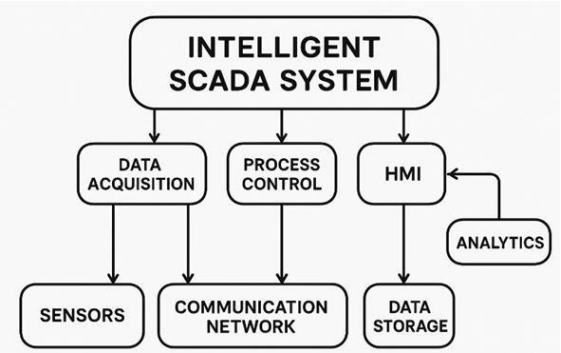

Рисунок1 − Схема інтелектуальної SCADA-системи

Основні складові інтелектуальної SCADA-системи представлені наступними блоками (рис.1).

Збір даних (Data Acquisition): датчики та сенсори (Sensors) отримують дані про фізичні параметри (температура, тиск, рівень рідини тощо); контролери (Controllers) – приймають сигнали від датчиків та передають їх у SCADA-систему.

Процесний контроль (Process Control): PLC (програмовані логічні контролери) управляють виробничими процесами на основі отриманих даних; DCS (розподілена система управління) використовується для більш складних виробничих процесів.

Людино-машинний інтерфейс (HMI - Human Machine Interface) надає операторам доступ до інформації та дозволяє управляти процесами через графічний інтерфейс.

Комунікаційна мережа (Communication Network) передає дані між польовими пристроями, контролерами та SCADA-системою через дротові (Ethernet, OPC-UA) або бездротові (LoRa, 5G) мережі.

Зберігання та обробка даних (Data Storage & Processing): сервери SCADA накопичують історичні та реальні дані; хмарні платформи дозволяють аналізувати та зберігати великі масиви даних у віддалених дата-центрах.

Аналітика та штучний інтелект (Analytics & AI) прогнозує можливі несправності обладнання, оптимізує процеси, виявляє аномалії у роботі системи.

Запропонована модель інтелектуальної SCADA−системи може бути використана для розробки автоматизованої системи

ВИСНОВКИ. Розробка інтелектуальних SCADA-систем є перспективним напрямком, що поєднує класичну автоматизацію з інноваційними технологіями. Впровадження таких систем сприяє підвищенню продуктивності, безпеки та гнучкості підприємств, але вимагає уваги до питань кібербезпеки та інтеграції з іншими цифровими технологіями. Використання штучного інтелекту дозволяє прогнозувати несправності обладнання, автоматизувати прийняття рішень та покращувати ефективність роботи системи. Це сприяє зниженню витрат на технічне обслуговування. Оскільки SCADA-системи пов’язані з інтернетом, вони стають вразливими до кібератак. Тому необхідно впроваджувати сучасні методи шифрування, захисту мереж та доступу до даних.

ЛІТЕРАТУРА

1. Aksyonov, K., & Antonova, A. (2022). Development of an Automated System for Analysis, Modeling, and Decision-Making for Metallurgical Enterprise. В Bratan, & S. Roshchupkin (Ред.), International Conference on Modern Trends in Manufacturing Technologies and Equipment 2021, ICMTMTE 2021 [050028] (AIP Conference Proceedings; Том 2503). American Institute of Physics Inc.. https://doi.org/10.1063/5.0100034

2. Antonova A.S., Aksyonov K.A., Aksyonova O.P. An imitation and heuristic method for scheduling with subcontracted resources. Mathematics 2021, 9(17), 2098; https://doi.org/10.3390/math9172098

3. Antonova A.S., Aksyonov K.A. (2020). Development of simulation model of continuous casting machine with dry change of steel ladles. In T. Simos, & C. Tsitouras (Ed.), International Conference on Numerical Analysis and Applied Mathematics, ICNAAM 2019 [140004] (AIP Conference Proceedings; Vol. 2293). American Institute of Physics Inc.. WOS: 000636709500413; https://doi.org/10.1063/5.0027487

4. Daniel F., José A., and Anibal O., Control aware of limitations of manipulators with claw for aerial robots imitating bird's skeleton, IEEE Robotics and Automation. (2021) 6, no. 4, 6426–6433, https://doi.org/10.1109/LRA.2021.3093282.

ДОДАТОК Б

Код програми

MainForm.cs

```
using SCADA_ALARM;
using System;
using System.Linq;
using System.Windows.Forms;

public partial class MainForm : Form
{
   private AlarmManager alarmManager = new AlarmManager();
   private EquipmentStatus equipmentStatus = new EquipmentStatus();
   private PerformanceMonitor performanceMonitor = new PerformanceMonitor();
   private DataAcquisition dataAcquisition = new DataAcquisition();
   private LLMService llm;

   public MainForm()
   {
      InitializeComponent();
      InitializeAlarmGrid();
      alarmManager.AlarmAdded += OnAlarmAdded;
      dataAcquisition.AlarmDetected += OnExternalAlarm;
   }

   private void MainForm_Load(object sender, EventArgs e)
   {
      UpdateStatus();
      //dataAcquisition.Start();
      llm = new LLMService("sk-or-v1-
6d975c8914288be157af650ba8bbd1906192c802ceb03b6c117baae801fbfaa2");
   }

   private void InitializeAlarmGrid()
   {
      dataGridView1.Columns.Add("Time", "Time");
      dataGridView1.Columns.Add("Message", "Message");
      dataGridView1.Columns.Add("Severity", "Severity");
      dataGridView1.Columns.Add("Ack", "Acknowledged");
   }

   private void LogToUI(string text)
```

```
{
    if (lstLogs.InvokeRequired)
    {
        lstLogs.Invoke(new Action(() =>
            lstLogs.Items.Add($"{DateTime.Now:HH:mm:ss} - {text}")
        ));
    }
    else
    {
        lstLogs.Items.Add($"{DateTime.Now:HH:mm:ss} - {text}");
    }
}

private async void OnAlarmAdded(Alarm alarm)
{

    if (dataGridView1.InvokeRequired)
    {
        dataGridView1.Invoke(new Action(() => AddAlarmToUI(alarm)));
    }
    else
    {
        AddAlarmToUI(alarm);
    }


    equipmentStatus.CurrentState = EquipmentState.Error;
    performanceMonitor.Stop();

    Logger.Log($"Alarm [{alarm.Severity}]: {alarm.Message}");
    LogToUI($"Alarm [{alarm.Severity}] {alarm.Message}");

    UpdateStatus();


    var explanation = await llm.AnalyzeAlarmAsync(alarm.Message);


    if (dataGridView1.InvokeRequired)
```

```
        {
            dataGridView1.Invoke(new Action(() =>
            {
                var form = new ScrollableMessageForm("Інтелектуальний аналіз
тривоги", explanation);
                form.Show();
            }));
        }
        else
        {
            var form = new ScrollableMessageForm("Інтелектуальний аналіз тривоги",
explanation);
            form.Show();
        }
    }

    private void AddAlarmToUI(Alarm alarm)
    {
        dataGridView1.Rows.Add(
            alarm.Timestamp.ToString("HH:mm:ss"),
            alarm.Message,
            alarm.Severity,
            alarm.IsAcknowledged ? "Yes" : "No"
        );

        var index = dataGridView1.Rows.Count - 1;
        ApplyRowColor(dataGridView1.Rows[index], alarm.Severity);
    }

    private void btnViewGroups_Click(object sender, EventArgs e)
    {
        lstGrouped.Items.Clear();
        var groups = alarmManager.GroupBySeverity();

        foreach (var group in groups)
        {
            lstGrouped.Items.Add($"=== {group.Key} ===");
            foreach (var alarm in group.Value)
            {
```

```
            lstGrouped.Items.Add($"[{alarm.Timestamp:HH:mm}]
{alarm.Message}");
        }
    }
}

private void ApplyRowColor(DataGridViewRow row, string severity)
{
    switch (severity)
    {
        case "Critical":
            row.DefaultCellStyle.BackColor = System.Drawing.Color.DarkRed;
            row.DefaultCellStyle.ForeColor = System.Drawing.Color.White;
            break;
        case "High":
            row.DefaultCellStyle.BackColor = System.Drawing.Color.Orange;
            row.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;
            break;
        case "Medium":
            row.DefaultCellStyle.BackColor = System.Drawing.Color.Yellow;
            row.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;
            break;
        case "Low":
            row.DefaultCellStyle.BackColor = System.Drawing.Color.LightGreen;
            row.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;
            break;
        default:
            row.DefaultCellStyle.BackColor = System.Drawing.Color.White;
            row.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;
            break;
    }
}

private void OnExternalAlarm(string message, string severity)
{
    if (equipmentStatus.CurrentState == EquipmentState.Stopped)
    {
        Logger.Log($"⛔ Тривога «{message}» проігнорована — система
зупинена.");
```

```
        return;
    }

    alarmManager.RaiseAlarm(message, severity);
}



private void UpdateStatus()
{
    if (lblStatus.InvokeRequired || lblOEE.InvokeRequired)
    {
        lblStatus.Invoke(new Action(() =>
        {
            lblStatus.Text = $"Статус: {equipmentStatus.CurrentState}";
            lblOEE.Text = $"OEE: {performanceMonitor.GetOEE():F2}%";
        }));
    }
    else
    {
        lblStatus.Text = $"Статус: {equipmentStatus.CurrentState}";
        lblOEE.Text = $"OEE: {performanceMonitor.GetOEE():F2}%";
    }
}

private void btnStart_Click(object sender, EventArgs e)
{
    equipmentStatus.CurrentState = EquipmentState.Running;
    dataAcquisition.Start();
    performanceMonitor.Start();
    Logger.Log("Лінія запущена");
    UpdateStatus();
}

private void btnStop_Click(object sender, EventArgs e)
{
    equipmentStatus.CurrentState = EquipmentState.Stopped;
    performanceMonitor.Stop();
    Logger.Log("Лінія зупинена");
```

```
        UpdateStatus();
    }

    private void btnTriggerAlarm_Click(object sender, EventArgs e)
    {
        var alarmForm = new AlarmInputForm();
        alarmForm.AlarmRaised += (message, severity) =>
        {
            alarmManager.RaiseAlarm(message, severity);
        };
        alarmForm.Show();
    }

    private void btnAcknowledgeSelected_Click(object sender, EventArgs e)
    {
        if (dataGridView1.SelectedRows.Count == 0)
        {
            MessageBox.Show("Будь ласка, виберіть тривогу для підтвердження.",
"Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);
            return;
        }

        foreach (DataGridViewRow selectedRow in dataGridView1.SelectedRows)
        {
            var timeStr = selectedRow.Cells[0].Value?.ToString();
            var message = selectedRow.Cells[1].Value?.ToString();

            if (DateTime.TryParse(timeStr, out var timestamp) && message != null)
            {
                var alarm = alarmManager.ActiveAlarms.FirstOrDefault(a =>
                    a.Timestamp.ToString("HH:mm:ss") ==
timestamp.ToString("HH:mm:ss") &&
                    a.Message == message
                );

                if (alarm != null && !alarm.IsAcknowledged)
                {
                    alarm.IsAcknowledged = true;
                    performanceMonitor.Start();
```

```
                Logger.Log($"Підтверджено тривогу: {alarm.Message}
({alarm.Severity}) о {DateTime.Now:HH:mm:ss}");
            }
        }
    }

    // Оновити візуальне представлення
    dataGridView1.Rows.Clear();
    foreach (var alarm in alarmManager.ActiveAlarms)
    {
        dataGridView1.Rows.Add(
            alarm.Timestamp.ToString("HH:mm:ss"),
            alarm.Message,
            alarm.Severity,
            alarm.IsAcknowledged ? "Yes" : "No"
        );
    }

    // Якщо всі підтверджені — перевести стан у Running
    if (alarmManager.ActiveAlarms.All(a => a.IsAcknowledged))
    {
        equipmentStatus.CurrentState = EquipmentState.Running;
        performanceMonitor.Start();
        Logger.Log("Усі тривоги підтверджено. Стан обладнання: Running.");
    }
}


private void btnExport_Click(object sender, EventArgs e)
{
    using (var sfd = new SaveFileDialog())
    {
        sfd.Filter = "CSV files (*.csv)|*.csv";
        sfd.FileName = "alarms_export.csv";

        if (sfd.ShowDialog() == DialogResult.OK)
        {
            try
            {
```

```
                // Використовуємо UTF8 з BOM для підтримки української
                using (var writer = new System.IO.StreamWriter(sfd.FileName, false,
new System.Text.UTF8Encoding(true)))
                {
                    // Заголовки
                    var headers =
dataGridView1.Columns.Cast<DataGridViewColumn>()
                        .Select(col => col.HeaderText);
                    writer.WriteLine(string.Join(",", headers));

                    // Дані
                    foreach (DataGridViewRow row in dataGridView1.Rows)
                    {
                        if (!row.IsNewRow)
                        {
                            var cells = row.Cells.Cast<DataGridViewCell>()
                                .Select(cell => "\"" + (cell.Value?.ToString().Replace("\"",
"\"\"") ?? "") + "\"");
                            writer.WriteLine(string.Join(",", cells));
                        }
                    }
                }
                MessageBox.Show("Експорт успішний!", "Інформація",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка під час експорту: {ex.Message}",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}

private void btnPredict_Click(object sender, EventArgs e)
{
    var predictions =
alarmManager.PredictFrequentAlarms(TimeSpan.FromMinutes(30), 3);
```

```
        if (predictions.Count == 0)
        {
            MessageBox.Show("Немає частих повторів тривог за останні 30 хв.",
"Прогноз", MessageBoxButtons.OK, MessageBoxIcon.Information);
            return;
        }

        var msg = "⚠ Виявлено підозріло часті тривоги:\n\n" + string.Join("\n",
predictions);
        MessageBox.Show(msg, "Прогноз повторних тривог",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }


    private void btnAckAll_Click(object sender, EventArgs e)
    {
        foreach (var alarm in alarmManager.ActiveAlarms)
        {
            alarm.IsAcknowledged = true;
        }

        dataGridView1.Rows.Clear();
        foreach (var alarm in alarmManager.ActiveAlarms)
        {
            dataGridView1.Rows.Add(
                alarm.Timestamp.ToString("HH:mm:ss"),
                alarm.Message,
                alarm.Severity,
                "Yes"
            );
        }
        performanceMonitor.Start();
        Logger.Log("Всі тривоги підтверджено");
    }
}
```

Alarm.cs

```
using System;
```

```
public class Alarm
{
   public DateTime Timestamp { get; set; }
   public string Message { get; set; }
   public string Severity { get; set; }
   public bool IsAcknowledged { get; set; }
}
```

AlarmInputForm.cs

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

namespace SCADA_ALARM
{
   public partial class AlarmInputForm : Form
   {
      public event Action<string, string> AlarmRaised;

      public AlarmInputForm()
      {
         InitializeComponent();
      }

      private void btnFire_Click(object sender, EventArgs e)
      {
         AlarmRaised?.Invoke("🔥 Пожежа в цеху №3", "Critical");
         this.Close();
      }

      private void btnGasLeak_Click(object sender, EventArgs e)
      {
         AlarmRaised?.Invoke("💨 Витік газу в зоні складу", "High");
         this.Close();
      }

      private void btnOverheat_Click(object sender, EventArgs e)
      {
         AlarmRaised?.Invoke("🌡 Перевищення температури в печі", "Medium");
```

```
            this.Close();
        }
    }
}
```

AlarmManager.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

public class AlarmManager
{
    public List<Alarm> ActiveAlarms { get; private set; } = new List<Alarm>();
    public List<Alarm> AlarmHistory { get; private set; } = new List<Alarm>();
    public event Action<Alarm> AlarmAdded;

    public List<string> PredictFrequentAlarms(TimeSpan interval, int threshold = 3)
    {
        var now = DateTime.Now;
        var recent = AlarmHistory
            .Where(a => (now - a.Timestamp) <= interval)
            .GroupBy(a => a.Message)
            .Select(g => new { Message = g.Key, Count = g.Count() })
            .Where(x => x.Count >= threshold)
            .Select(x => $"{x.Message} — {x.Count} раз(ів) за останні
{interval.TotalMinutes} хв.")
            .ToList();

        return recent;
    }

    public void RaiseAlarm(string message, string severity)
    {
        var alarm = new Alarm
        {
            Timestamp = DateTime.Now,
            Message = message,
            Severity = severity,
            IsAcknowledged = false
```

```
        };

        ActiveAlarms.Add(alarm);
        AlarmHistory.Add(alarm);
        AlarmAdded?.Invoke(alarm);
        Logger.Log($"Alarm [{severity}]: {message}");
    }

    public void AcknowledgeAlarm(Alarm alarm)
    {
        alarm.IsAcknowledged = true;
        Logger.Log($"Acknowledged: {alarm.Message}");
    }

    public Dictionary<string, List<Alarm>> GroupBySeverity()
    {
        return ActiveAlarms
            .GroupBy(a => a.Severity)
            .ToDictionary(g => g.Key, g => g.ToList());
    }
}
```

DataAcquisition.cs

```
using System;
using System.Collections.Generic;
using System.Threading;

public class DataAcquisition
{
    private Timer _timer;
    private Random _random = new Random();
    private bool _isRunning = false;
    private readonly object _lock = new object();

    private List<(string Message, string Severity)> _alarms = new List<(string,
string)>
    {
        ("Перевищення температури", "High"),
        ("Витік газу", "Critical"),
```

```
        ("Низький тиск у системі", "Medium"),
        ("Пожежа в зоні 1", "Critical"),
        ("Перевантаження двигуна", "High"),
        ("Втручання в обладнання", "Low"),
        ("Рівень масла критично низький", "High")
    };

    public event Action<string, string> AlarmDetected;

    public void Start()
    {
        lock (_lock)
        {
            if (_isRunning) return;

            _isRunning = true;
            _timer = new Timer(OnTimerElapsed, null, 5000, 5000); // Запуск через 5
сек і кожні 5 сек
        }
    }

    public void Stop()
    {
        lock (_lock)
        {
            if (!_isRunning) return;

            _isRunning = false;
            _timer?.Dispose();
            _timer = null;
        }
    }

    private void OnTimerElapsed(object state)
    {
        lock (_lock)
        {
            if (!_isRunning) return;
```

```
            var trigger = _random.Next(0, 5) == 0;
            if (trigger)
            {
                var alarm = _alarms[_random.Next(_alarms.Count)];
                AlarmDetected?.Invoke(alarm.Message, alarm.Severity);
            }
        }
    }
}
```

EquipmentStatus.cs

```
public enum EquipmentState
{
    Running,
    Stopped,
    Error
}

public class EquipmentStatus
{
    public EquipmentState CurrentState { get; set; } = EquipmentState.Stopped;
}
```

LLMService.cs

```
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json;

public class LLMService
{
    private readonly HttpClient client;
    private readonly string apiKey;
    private readonly string modelEndpoint;

    public LLMService(string apiKey)
    {
```

```
        this.apiKey = apiKey;
        client = new HttpClient();
        modelEndpoint = "https://openrouter.ai/api/v1/chat/completions"; // OpenRouter
endpoint
    }

    public async Task<string> AnalyzeAlarmAsync(string alarmText)
    {
        var requestBody = new
        {
            model = "google/gemini-2.5-flash",
            messages = new[]
            {
                new { role = "system", content = "Ти — інтелектуальна SCADA-
підсистема для виробничого підприємства типу Lean Production. Твоє завдання
— аналізувати повідомлення тривоги та пояснити її значення й дії для
оператора." },
                new { role = "user", content = $"Тривога: {alarmText}" }
            }
        };

        var json = JsonConvert.SerializeObject(requestBody);
        var request = new HttpRequestMessage(HttpMethod.Post, modelEndpoint)
        {
            Content = new StringContent(json, Encoding.UTF8, "application/json")
        };
        request.Headers.Authorization = new AuthenticationHeaderValue("Bearer",
apiKey);

        var response = await client.SendAsync(request);
        var result = await response.Content.ReadAsStringAsync();

        dynamic obj = JsonConvert.DeserializeObject(result);
        return obj?.choices[0]?.message?.content ?? "Немає пояснення.";
    }
}
```

Logger.cs

```
using System;
```

```
using System.Collections.Generic;

public static class Logger
{
   public static List<string> Logs { get; private set; } = new List<string>();

   public static void Log(string message)
   {
      var entry = $"{DateTime.Now:HH:mm:ss} - {message}";
      Logs.Add(entry);
   }

   public static List<string> GetLogs()
   {
      return new List<string>(Logs);
   }

   public static void Clear()
   {
      Logs.Clear();
   }
}
```

PerformanceMonitor.cs

```
using System;

public class PerformanceMonitor
{
   private DateTime _lastStartTime;
   private DateTime? _lastStopTime;

   private TimeSpan _accumulatedUptime = TimeSpan.Zero;
   private TimeSpan _accumulatedDowntime = TimeSpan.Zero;

   private bool _isRunning = false;

   public TimeSpan Uptime => _accumulatedUptime + (_isRunning ?
DateTime.Now - _lastStartTime : TimeSpan.Zero);
```

```
    public TimeSpan Downtime => _accumulatedDowntime + (!_isRunning &&
_lastStopTime.HasValue ? DateTime.Now - _lastStopTime.Value : TimeSpan.Zero);

    public void Start()
    {
        if (!_isRunning)
        {
            // Якщо до цього була зупинка, додати час простою
            if (_lastStopTime.HasValue)
            {
                _accumulatedDowntime += DateTime.Now - _lastStopTime.Value;
                _lastStopTime = null;
            }

            _lastStartTime = DateTime.Now;
            _isRunning = true;
        }
    }

    public void Stop()
    {
        if (_isRunning)
        {
            // Додати час роботи
            _accumulatedUptime += DateTime.Now - _lastStartTime;
            _lastStopTime = DateTime.Now;
            _isRunning = false;
        }
    }

    public double GetOEE()
    {
        var totalSeconds = Uptime.TotalSeconds + Downtime.TotalSeconds;
        if (totalSeconds == 0) return 0;
        return Uptime.TotalSeconds / totalSeconds * 100;
    }
}
```

ScrollableMessageForm.cs

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

public partial class ScrollableMessageForm : Form
{
    public ScrollableMessageForm(string title, string message)
    {
        InitializeComponent();

        this.Text = title;

        // Примітивна обробка форматування (абзаци, переноси, заголовки)
        string formatted = FormatText(message);
        richTextBox.Text = formatted;
    }

    private string FormatText(string input)
    {
        // Прості заміни для читабельності
        input = input.Replace("\n", Environment.NewLine);
        input = input.Replace("•", "\u2022"); // маркер списку
        input = input.Replace("### ", "").Replace("## ", "").Replace("# ", ""); //
прибрати markdown заголовки

        return input;
    }
}
```

ДОДАТОК В

Демонстраційний матеріал

Формат	Позиція	Найменування	Додаткові відомості
		Текстові документи	
А4	ГЮИК. 442351.012 ПЗ	Пояснювальна записка	73 с.
		Додаткові матеріали	
		Апробація роботи	3 с.
		Код програми	3 с.
		Демонстраційний матеріал	10 с.

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК.442351.012 ВД			
Розроб.		Коваленко В.Я.			Розроблення SCADA-системи для виробничого підприємства типу lean production	Літера	Аркуш	Аркушів
Перевір.		Сезонова І. К.				Н	1	1
Н. контр.		Стародубцев М.П.				Кафедра КІТАР ХНУРЕ		
Затв.		Невлюдов І.Ш.						