БАКАЛАВРСЬКА РОБОТА

КРБ.СІ – 13.00.000 ПЗ

Група СІ-21-1

Максим Настащук

2025

Міністерство освіти і науки України

Івано-Франківський національний університет нафти і газу

Інститут інформаційних технологій

Кафедра інформаційно-телекомунікаційних технологій та систем

Настащук Максим Юрійович

(прізвище, ім'я, по-батькові)

УДК 004.4
(індекс)

БАКАЛАВРСЬКА РОБОТА

Апаратно-програмне забезпечення лабораторного практикуму «Проєктування WEB-орієнтованих систем управління»
(назва роботи)

Системна інженерія - Інтернет речей
(назва освітньої програми)

151 Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____ М. Ю. Настащук
(підпис, ініціали та прізвище здобувача)

Науковий керівник _____ М. Я. Николайчук к.т.н., доцент кафедри ІТТС
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

д.т.н., професор _____ Л. М. Заміховський
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

(повне найменування закладу вищої освіти)

Інститут інформаційних технологій

Кафедра інформаційно-телекомунікаційних технологій та систем

Освітній рівень бакалавр

Спеціальність 151 «Автоматизація та ком'ютерно-інтегровані технології»

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТТС

д.т.н., проф. Заміховський Л.М

«___» _____2025 року

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Настащуку Максиму Юрійовичу

(прізвище, ім'я, по-батькові)

1. Тема роботи *Апаратно-програмне забезпечення лабораторного практикуму «Проєктування WEB-орієнтованих систем управління»*
   керівник роботи *Николайчук М. Я. к.т.н., доцент кафедри ІТТС*,
   затверджені наказом закладу вищої освіти від «05» травня 2025 року №281/7
2. Строк подання студентом роботи *15.06.2025*
3. Вихідні дані до роботи: *Програмні компоненти PLC S-7-1500, TIA PORTAL V20, Simantic HMI Unified Comfort Panetl, Simantic WinCC Unified, дані зібрані під час переддипломної практики у NetGroup Service*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1.*Функціональність і структура проєктів на базі SCADA WinCC.*
   2.*Технологія розробки сценаріїв мнемосхем і динамізації обє'ктів.*
   3.*Організація і робота з графічними об'єктами*
   4.*Розробка методичного прикладу на базі лабораторного практикуму*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
   1. Параметрування комунікації PLC S7-1500. Схема функціональна (1 аркуш). 2. Процес налаштування нових елементів екрану. Схема структурна (1 аркуш). 3. Апаратна конфігурація тестового проєкту. Схема структурна (1 аркуш). 4. Приклад роботи тестового проєкту. Схема структурна

6. Дата видачі завдання *01.03.2025*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Термін виконання етапів роботи	Примітка
1	*Функціональність і структура проєктів на базі SCADA WinCC*	*1.04.2025*	*виконано*
2	*Технологія розробки сценаріїв мнемосхем і динамізації обє'ктів*	*1.05.2025*	*виконано*
3	*Організація і робота з графічними об'єктами*	*1.06.2025*	*виконано*
4	*Розробка методичного прикладу на базі лабораторного практикуму*	*5.06.2025*	*виконано*
5	*Оформлення пояснювальної записки та графічної частини*	*15.06.2025*	*виконано*

Студент _____ М. Ю. Настащук
(підпис) (прізвище та ініціали)

Керівник роботи _____ М. Я. Николайчук
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Настащук М. Ю. Апаратно-програмне забезпечення лабораторного практикуму «Проектування WEB-орієнтованих систем управління» ІФНТУНГ, 2025. 79 с.

Бакалаврська робота на здобуття освітнього ступеня бакалавр за освітньо-професійною програмою «Системна інженерія – Інтернет речей», спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка». Івано-Франківський національний університет нафти і газу. Івано-Франківськ, 2025.

У роботі розглянуто склад і функціональні можливості апаратно-програмного забезпечення, що використовується в лабораторному практикумі з дисципліни «Проєктування WEB-орієнтованих систем управління». Проведено аналіз архітектури WEB-орієнтованих систем управління, принципів його роботи, а також взаємодії між програмними модулями та апаратними компонентами. Описано типове програмне середовище для створення WEB-інтерфейсів управління.

Розроблено методичний приклад виконання проєкту на базі описаного програмного середовища.

Ключові слова: WEB-ОРІЄНТОВАНІ СИСТЕМИ, АВТОМАТИЗАЦІЯ ТЕХНОЛОГІЧНИХ ОБ’ЄКТІВ, PLC, SCADA, HMI, IOT.

ANNOTATION

Nastashchuk M. Y. Hardware and Software of the Laboratory Practicum "Design of WEB-Oriented Control Systems".
IFNTUNG, 2025. 79 pages.

Bachelor's thesis submitted for the Bachelor's Degree in the educational and professional program "System Engineering – Internet of Things", specialty 174 "Automation, Computer-Integrated Technologies and Robotics". Ivano-Frankivsk National Technical University of Oil and Gas. Ivano-Frankivsk, 2025.

The thesis examines the composition and functional capabilities of the hardware and software used in the laboratory practicum focused on the design of WEB-oriented control systems. The work provides an analysis of the engineering system architecture, its operating principles, and the interaction between software modules and hardware components. A typical software environment for developing WEB-based control interfaces is described.

A methodological example of project implementation based on the described software environment has been developed.

Keywords: WEB-ORIENTED SYSTEMS, AUTOMATION OF TECHNOLOGICAL OBJECTS, PLC, SCADA, HMI, IOT, PLC.

РЕФЕРАТ

Розрахункова-пояснювальна записка: 78 с., 85 рисунків., 4 табл. , 14 джерел.

Об’єкт дослідження – Інформаційні процеси у WEB-орієнтованих системах управління на базі PLC і SCADA WinCC Unified.

Предмет дослідження – технологія розробки Апаратно-програмне забезпечення лабораторного практикуму.

Мета роботи – розроблення апаратно-програмного і методичного забезпечення лабораторного практикуму «Проектування WEB-орієнтованих систем управління».

Роботу виконано на основі PLC Simatic S7 і SCADA WinCC Unified.

У даній роботі проаналізовано апаратно-програмні засоби для виконання забезпечення лабораторного практикуму «Проєктування WEB-орієнтованих систем управління». Застосовано програмні засоби, що забезпечують візуалізацію, логіку керування та віддалену взаємодію між компонентами системи управління. Результати розробки можуть бути використані для подальшої модернізації апаратно-програмних засобів, розробки нових практичних завдань, а також для підвищення ефективності навчання здобувачів у галузі сучасних інформаційних технологій.

ЗМІСТ

					КРБ.СІ-13.00.000 ПЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Настащук М. Ю.			*Апаратно-програмне забезпечення лабораторного практикуму «Проектування WEB-орієнтованих систем управління»* *Пояснювальна записка*	*Літ.*	*Арк.* 7 / *Аркушів* 78
Перевірив		Николайчук М.Я.					
						ІФНТУНГ, СІ-21-1	
Н. Контр.		Возний А. В.					
Затвердив		Заміховський Л. М.					

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

HMI – Human Machine Interface (Інтерфейс людина-машина)

PLC – Programmable Logic Controller (Програмований логічний конролер)

UDT – User Defined Type (Тип даних користувача у PLC)

TIA – Totally Integrated Automation (Інтегрована платформа автоматизації Siemens)

RT – Runtime (Середовище виконання HMI-додатку)

JS – JavaScript (Мова програмування для створення сценаріїв у HMI)

SVG – Scalable Vector Graphics (Векторна графіка здатна до масштабування без втрати якості)

RTIL – Runtime Trace Information Logger (Засіб аналізу виконання проєкту)

V18/V19/V20 – Version 18/19/20 (Версії WinCC Unified)

HTTPS – HyperText Transfer Protocol Secure (Схема доступу до інтернет-ресурсів)

FBD - Functional Block Diagram (графічна мова програмування, призначена для програмування PLC)

OB - Organization Block (Структурна одиниця мови програмування FBD)

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій зростає попит на ефективні, гнучкі та масштабовані системи управління, які забезпечують віддалений доступ, моніторинг та керування технологічними процесами через WEB-інтерфейси. WEB-орієнтовані системи управління стають невіддільною частиною індустріальної автоматизації, енергетики, логістики, «розумного дому» та інших сфер діяльності.

Одним із ключових аспектів у підготовці фахівців з автоматизації та комп'ютерно-інтегрованих технологій є набуття практичних навичок проєктування та впровадження подібних систем. З цією метою у навчальному процесі доцільно використовувати інтерактивні лабораторні практикуми з можливістю симуляції компонентів системи, що забезпечує можливість роботи із сучасним апаратно-програмним забезпеченням на базі PLC і SCADA.

Розроблено методичне забезпечення лабораторного практикуму з дисципліни «Проєктування WEB-орієнтованих систем управління» на основі структури алгоритму послідовності проектних процедур і завдань. Наведено аналіз структурних компонентів, функціональні можливості застосованих технологій, принципи взаємодії між апаратною частиною та програмними інструментами, а також практичні аспекти їх використання у навчальному процесі.

1 ФУНКЦІОНАЛЬНІСТЬ І СТРУКТУРА ПРОЕКТІВ SCADA WINCC Unified

1.1 Функціональність SCADA WinCC Unified

SIMATIC WinCC Unified — це повністю нова система візуалізації, розроблена компанією Siemens для використання в середовищі автоматизації. Система SIMATIC WinCC Unified складається з програмного забезпечення для візуалізації SIMATIC WinCC Unified, нових SIMATIC HMI Unified Comfort Panels, а також Unified Basic Panels [1].

Unified Comfort Panels розширюють продуктову лінійку SIMATIC Advanced Panel-based HMIs і є пристроями-наступниками SIMATIC HMI Comfort Panels. Окрім нового апаратного забезпечення, присутні численні інноваційні функції, які суттєво впливають на спосіб їх використання (табл. 1.1).

Таблиця 1.1 - Використані компоненти

Компонент	**Кількість**	**Артикул**	**Примітка**
WinCC Unified Engineering V18	1	6AV215.-…01-8A.5	Engineering у TIA Portal
WinCC Unified Engineering V19	1	6AV215.-…02-3.A5	Engineering у TIA Portal
WinCC Unified Engineering V20	1	6AV2151-…..-4AA5	Engineering у TIA Portal
SIMATIC HMI Unified PC Runtime V18	1	6AV2154-..B01-8.A0	Runtime System
SIMATIC HMI Unified PC Runtime V19	1	6AV2154-..B02-3.A0	Runtime System
SIMATIC HMI Unified PC Runtime V20	1	6AV2154-…..-4AA0	Runtime System
SIMATIC HMI Unified Comfort Panel MTP1200	1	6AV2128-3MB06-0AX1	Альтернативно, можна використовувати будь-яку іншу SIMATIC HMI Unified Comfort Panel

Пристрої Unified Panel та PC-RT пропонують новітні технології, такі як HTML5, JavaScript та масштабовану векторну графіку (SVG). Це робить їх потужнішими та здатними забезпечити ширший спектр функцій і можливостей порівняно зі старими Comfort Panels і системами WinCC RT Advanced, що дозволяє отримати більше від системи [2].

Проєкти в WinCC Unified будуються у середовищі **TIA Portal** (Totally Integrated Automation), де вони інтегруються з іншими компонентами автоматизації Siemens, такими як контролери SIMATIC S7, пристрої HMI, приводи та інші модулі.

WinCC Unified включає у себе широкий набір функцій, які покривають основні потреби SCADA/HMI-систем, наприклад створення інтерактивних HMI-екранів, підтримку трендів, гістограм і діаграм, анімацію і масштабування та гнучку зміну графічних об’єктів [4].

Система обробки тривог дозволяє гнучко налаштувати систему під необхідний проєкт. В рамках цієї системи можна визначати порогові значення змінних, формувати повідомлення з різним рівнем пріоритету, а також виводити тривоги на окремі екрани при потребі.

Історизація, або Архівування – основа функція збору та збереження робочої інформації по тегам. Саме ця функція дозволяє перегляд трендів та експорт даних у різноманітних форматах.

В свою чергу головною особливістю нових версій WinCC Unified є саме розширення можливостей роботи з WEB-інтерфейсами, що дозволяє отримувати доступ до HMI через браузери по протоколу з’єднання HTTPS.

Доступ з браузеру забезпечує можливість підключення інтерфейсу керування системами навіть з мобільних пристроїв без встановлення будь-яких додаткових клієнтів [1].

1.2 Структура проєктів на базі WinCC Unified

Проєкти в WinCC Unified будуються у середовищі **TIA Portal** (Totally Integrated Automation), де вони інтегруються з іншими компонентами автоматизації Siemens, такими як контролери SIMATIC S7, пристрої HMI, приводи та інші модулі [2].

Проєкт WinCC Unified складається з наступних ключових структурних елементів:

Екрани є основними візуальними інтерфейсами оператора. Вони містять компоненти візуалізації, зокрема кнопки, індикатори, діаграми, графічні елементи тощо. Для кожного екрана можуть бути визначені події, скрипти та анімації. Підтримується ієрархія екранів із можливістю вкладеного використання через **Faceplates** [3].

Кожен елемент на екрані має набір властивостей: розмір, колір, текст, прив’язка до тегів, видимість, активність тощо. Вони можуть бути статичними або динамізованими — тобто змінюватися залежно від значень змінних (тегів). Динамізація реалізується через зв’язування з PLC-тегами, функціями, скриптами або системними подіями [4].

Теги — це змінні, які зв'язують графічні елементи з даними з PLC або внутрішніми змінними. Вони можуть бути:

-зовнішніми (external) — отримують значення з контролера;

-внутрішніми (internal) — використовуються для локальних обчислень і логіки;

-глобальними (global) — доступні у всьому проєкті;

-локальними — доступні лише в межах певного екрана або компонента.

WinCC Unified підтримує скрипти на мові JavaScript для реалізації додаткової логіки управління, обробки подій та взаємодії з елементами екранів.

Скрипти можуть бути:

-подійні (onClick, onChange, onLoaded);

-часові (за таймером);

-викликані вручну.

Подійні скрипти виконуються автоматично у відповідь на певну подію у інтерфейсі користувача. Часові скрипти виконуються через певні сталі або змінні інтервали часу, які встановлюються користувачем. Натомість, викликані вручну скрипти не прив’язані до будь-яких подій або таймерів, а запускаються з допомогою додаткових скриптів, або за умови певної запрограмованої користувачем логіки [8].

Скрипти дозволяють реалізувати нестандартні сценарії взаємодії з інтерфейсом користувача.

Момент, коли ми починаємо створювати абсолютно нові конфігурації або навіть просто нові екрани, є ідеальною можливістю для максимально ефективного використання нових функцій WinCC Unified [1].

У цьому контексті використовується термін view (подання), який означає повне відображення HMI, а отже - комбінацію кількох екранів у структурі вікон екрану.

Під час створення нових подань першим кроком є попереднє збирання всього контенту, який має бути відображений. Залежно від можливості повторного використання, ми приймаємо рішення про поділ на індивідуальний контент або контент, який буде використовуватися на кількох екранах.

Контент можна налаштовувати незалежно один від одного, або у відповідності з розташуванням контенту на кількох екранах. Доступ до цих екранів можна здійснювати з одного інтерфейсу користувача, який може розташовуватись на будь-якому приладі, завдяки функціям WEB-сторінок [5].

Процедура найбільш ефективного підходу пояснюється в наступній схемі відображеній на рисунку 1.1.

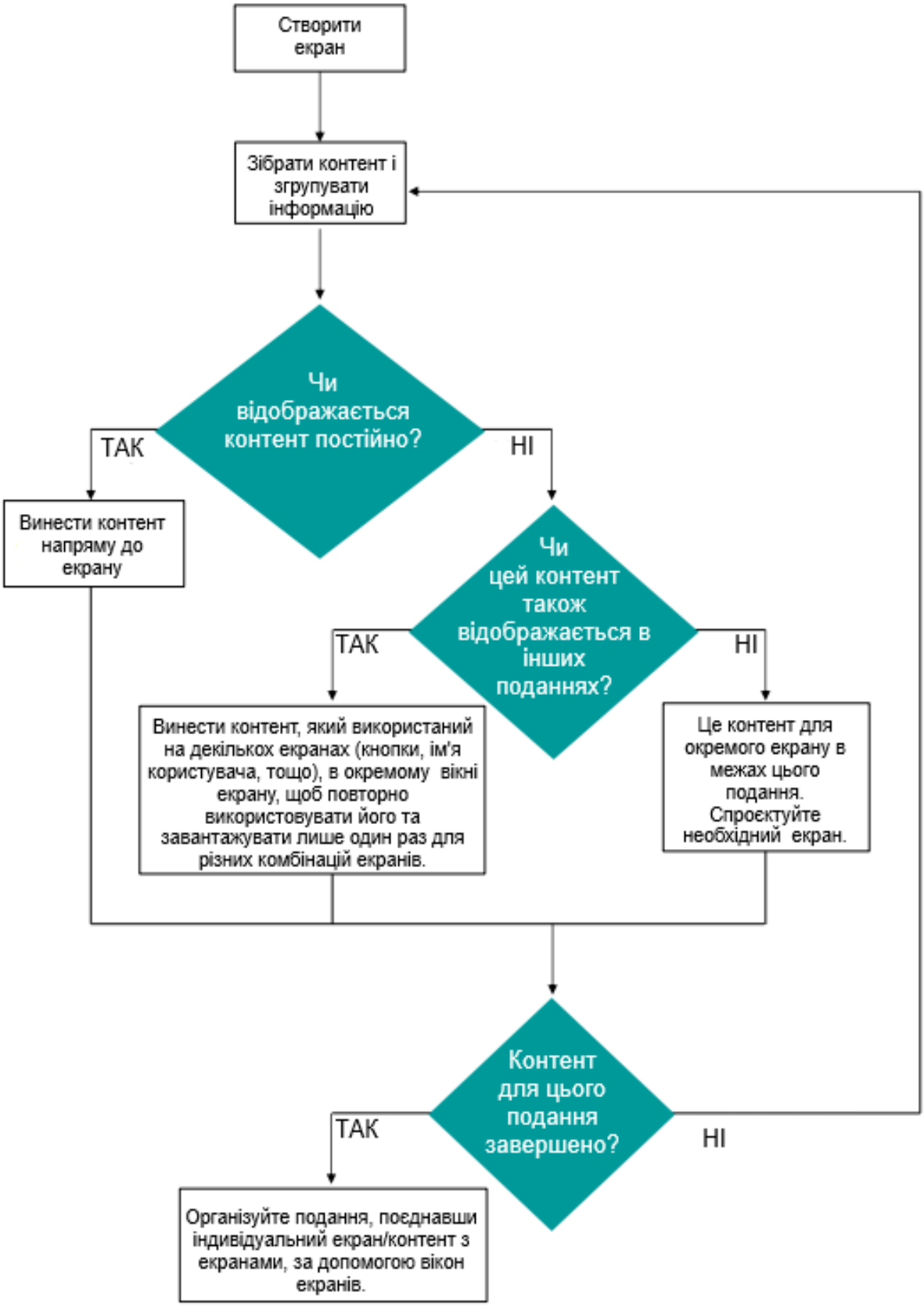


Рисунок 1.1 – Процес створення нового подання

Результатом цієї конфігурації може стати макет екрана, як показано на наступному зображенні. Усі індивідуальні вікна екрана виділені помаранчевою рамкою.

Для зручності структурування проєкту рекомендується використовувати HMI Template Suite. Цей шаблон допомагає створити та організувати область відображення, розділивши її на окремі сегменти з гнучкою навігацією по екранах (рис. 1.2). Він зосереджений на продуктивності та аспектах дизайну, забезпечуючи інтуїтивну концепцію керування.

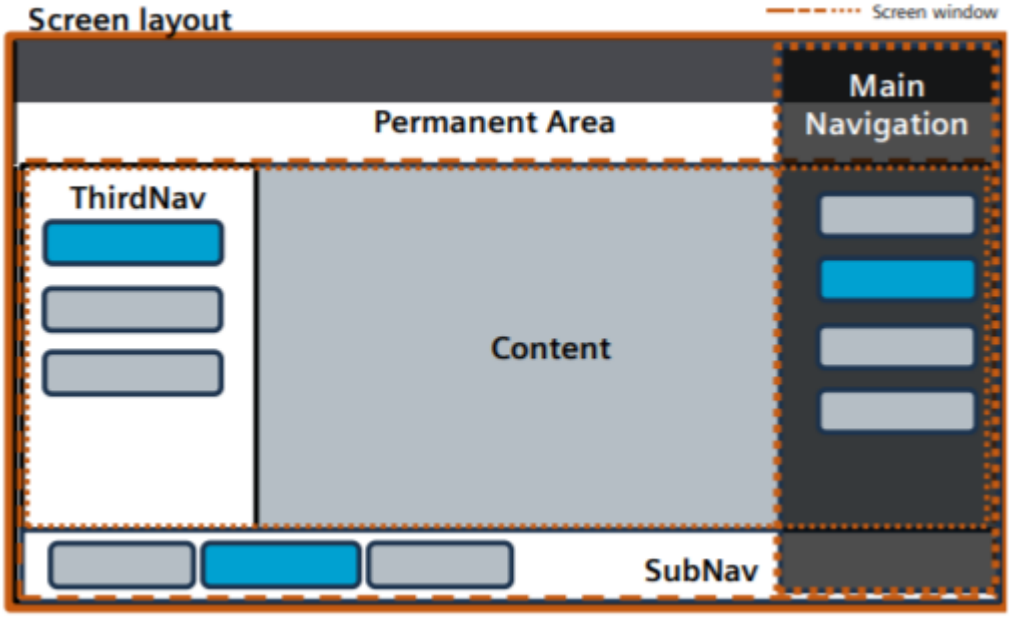


Рисунок 1.2 - Приклад подання з використанням кількох вікон екрана

Після того як ми визначили, які елементи повинні бути розміщені на яких екранах, можна продовжувати роботу з окремим екранами. Більшість елементів екрана потребують динамічних властивостей або використовуватимуть події для запуску системних функцій чи скриптів.

Щоб реалізувати ці динамічні властивості, наступна схема (рис. 1.3) допоможе нам визначити, яка реалізація буде найкращою для вашого застосування. Залежно від бажаної функціональності, оберемо відповідний елемент з панелі інструментів та перетягнемо його на екран, якщо він ще не розміщений.

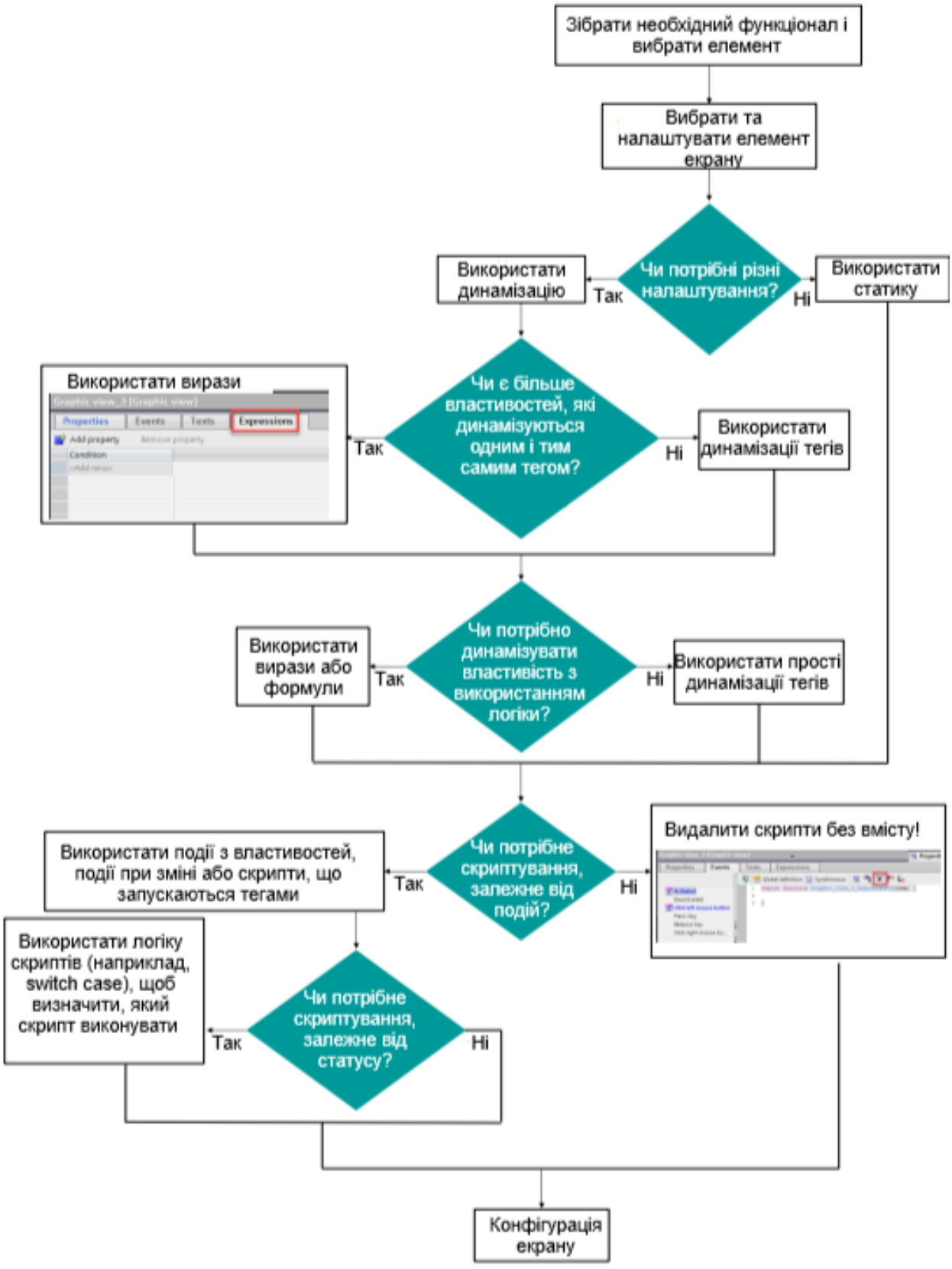


Рисунок 1.3 - Діаграма потоку налаштування нового елемента екрану

2 ТЕХНОЛОГІЯ РОЗРОБКИ СЦЕНАРЇВ МНЕМОСХЕМ І ДИНАМІЗАЦІЇ ОБЄКТІВ

2.1 Технологія розробки сценаріїв мнемосхем

Сценарії в **WinCC Unified** тепер пропонують набагато більше можливостей для налаштування елементів екрана і роблять їх значно більш гнучкими. Як описано у наступному прикладі випадку використання, одна кнопка може запускати різні дії залежно від поточного стану [12].

Конкретна дія або сценарій, що стоїть за подією, часто залежить від поточного стану. У цьому демонстраційному випадку використання повинна бути кнопка, яка показує або приховує певний вміст екрана (тут: прямокутник), залежно від поточної видимості цього вмісту. Одним із рішень може бути використання двох різних кнопок: одна реалізує дію показу, інша — дію приховування, і видимість цих двох кнопок також змінюється відповідно до поточного стану. Друге рішення — використання лише однієї кнопки з логікою сценарію (switch … case), яка визначає, яку дію слід виконати [4].

Використаємо одну кнопку для цього завдання та реалізуємо логіку сценарію для визначення, чи потрібно встановити видимість вмісту. Використовуємо один і той самий тег (тут: **“ContentVisible”**) для:

- відображення поточного стану;
- динамізації видимості вмісту;
- динамізації тексту кнопки.

Оскільки цей випадок використання дуже простий, інверсію тегу стану **“ContentVisible”** у логіці сценарію можна було б також реалізувати простою системною функцією через подію натискання кнопки. Але щоб продемонструвати, як може виглядати реалізація для складнішого випадку використання, на скріншоті показана реалізація з **switch case** [3].

Щоб змінити текст кнопки для різних станів, його динамізовано за допомогою списку текстів та знову того ж тегу стану (рис. 2.1-2.2).

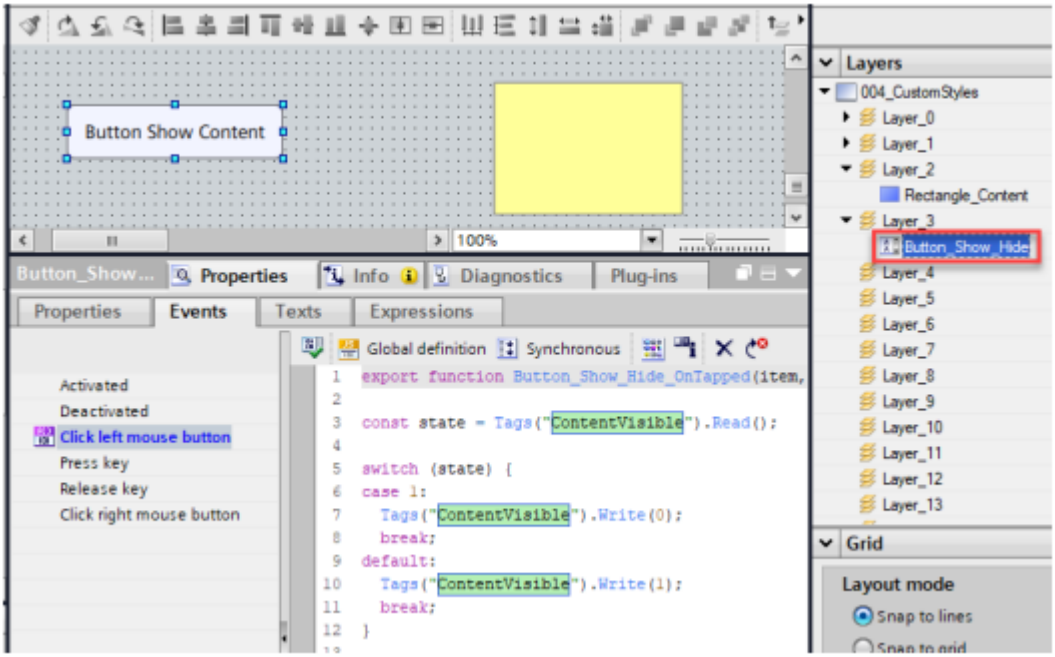


Рисунок 2.1 - Логіка сценарію для кнопки Показати/Приховати

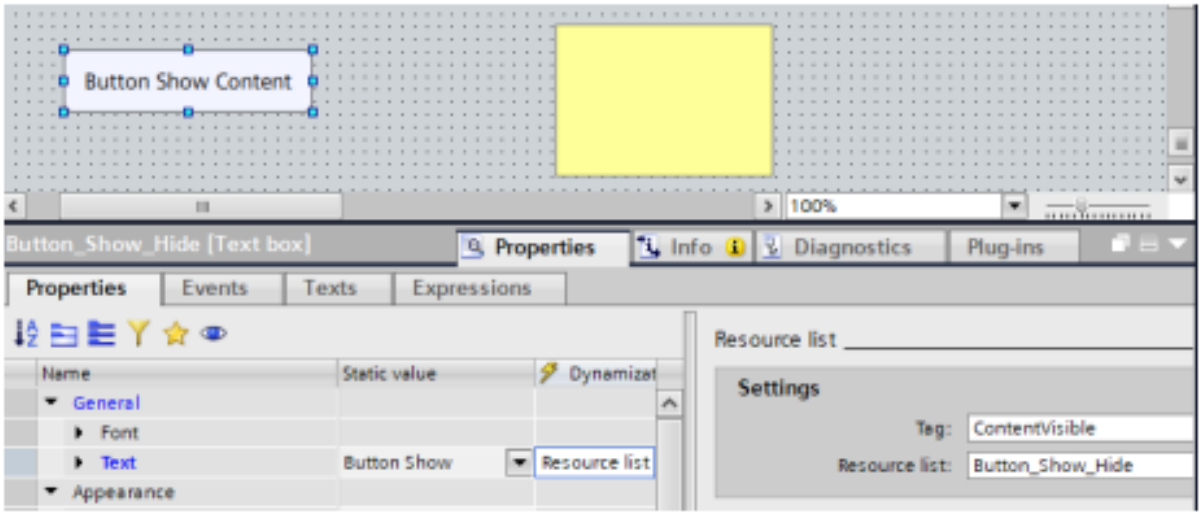


Рисунок 2.2 - Динамізація тексту кнопки у випадку використання Показати/Приховати

Ефективність виконання проєктів сильно різниться від підходу до використання функцій програмного забезпечення.

У наведеному нижче рішенні, яке **не рекомендується**, використовуються дві кнопки, що накладаються одна на одну: одна встановлює видимість прямокутника в **true**, а інша — у **false**. Щоб зробити кнопки доступними у відповідний момент, їх видимість також динамізується відповідно до стану видимості прямокутника. Для таких випадків використання, де дія залежить від стану, попередньо показане рішення зі сценарієм (scripting) є кращим за використання кількох накладених елементів екрана, оскільки це дозволяє мінімізувати кількість об’єктів на екрані (рис. 2.3-2.4).

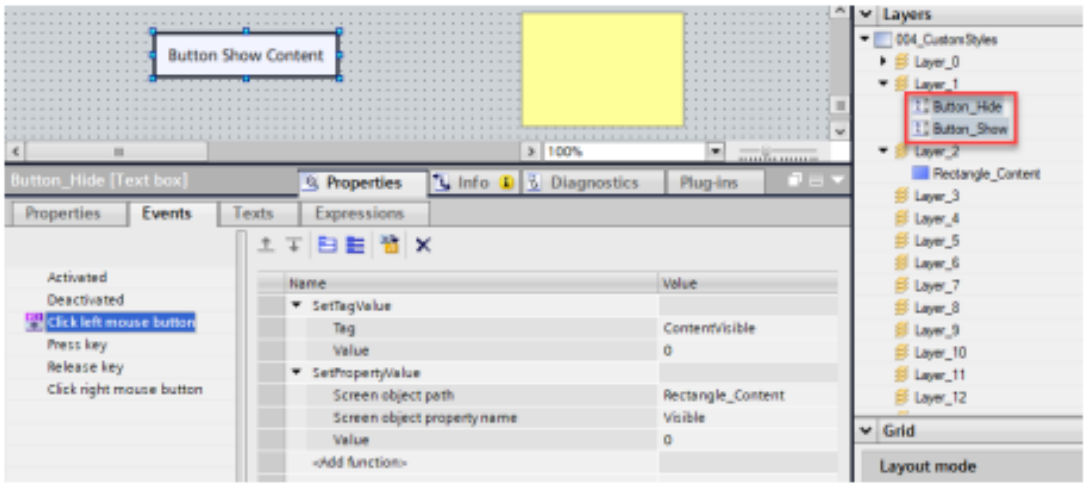


Рисунок 2.3 - Дві накладені кнопки для завдання Показати/Приховати

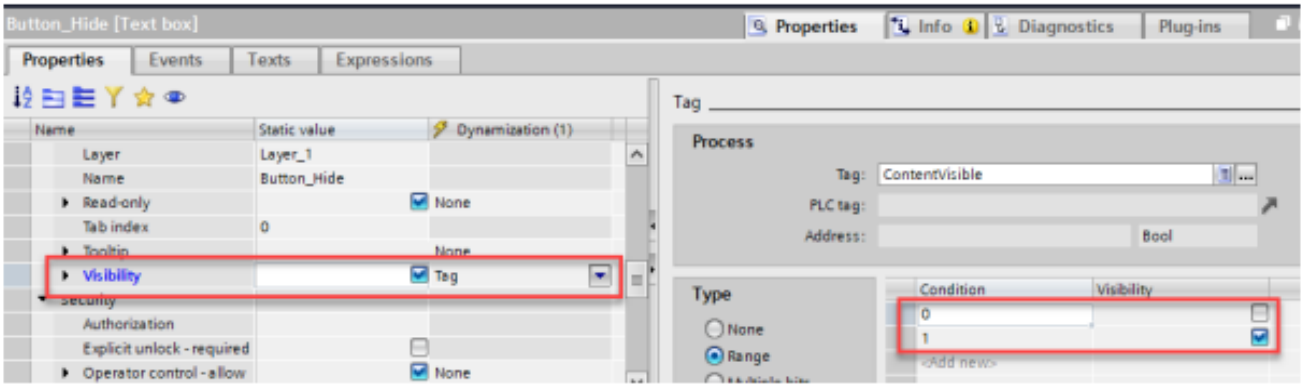


Рисунок 2.4 - Динамізація видимості двох кнопок

У **Unified**, як і в інших системах візуалізації, зміна екранів відіграє центральну роль у представленні різних екранів установки та у взаємодії з користувачем. Швидка зміна екрана створює у користувача відчуття ефективності проектованих елементів, а повільна або затяжна зміна викликає дискомфорт. За допомогою кількох простих прийомів ми можемо ще краще підготувати зміну екранів.

Щоб максимально ефективно реалізувати кроки інженерії, важливо розуміти, як саме реалізовуються ці кроки найкращим чином. Перш ніж заглиблюватися в деталі щодо «правильної» реалізації, також важливо ознайомитися з процедурою завантаження екрана (рис. 2.5) під час виконання (runtime).

Рисунок 2.5 - Процедура завантаження екрана

Процедура завантаження окремого екрана в основному розділена на синхронні етапи, тобто одна дія відбувається послідовно після іншої. Оскільки компонування часто здійснюється за допомогою вкладених екранів і Faceplates, важливо знати, що для кожного вікна екрана та кожного екземпляра Faceplate.

На самому початку процесу зміни екрана, ще до завантаження нового екрана, попередній екран очищається.

Після завершення цього кроку будується базовий екран, і всі властивості, навіть якщо вони динамізовані (наприклад, пов’язані з тегами), завантажуються зі статичним значенням для початкового завантаження. Усі вкладені Faceplates і екрани у вікнах екрана також завантажуються у власному незалежному циклі – спочатку також у статичному циклі.

Далі виконується подія **“loaded”**, яка також може бути налаштована під час інженерного процессу [3].

Реєстрація тегів відбувається після початкового завантаження екрана. Після завершення реєстрації тегів враховуються всі динамізації, і властивості елементів екрана, які мають прив’язку до тегів у зоні динамізації, можуть знову змінитися — тепер зі **статичного значення на значення динамізованого тегу**.

Якщо в цьому контексті змінюються теги, пов’язані з будь-якою динамізацією або властивістю, то відповідні властивості елементів екрана також можуть змінитися знову. Завдяки цьому порядку виклику **подія зміни властивості (on-change event)** може бути виконана **двічі** під час процедури завантаження екрана.

У разі використання шаблонів або повторного використання компонентів, їх завантаження та ініціалізація проходять аналогічно – спочатку зі статичним значенням, а потім із урахуванням динамізації.

Особливу увагу слід також звернути і на те, що деякі події може здійснюватися багаторазово - якщо значення тегів змінювалось декілька разів під час циклу завантаження [4].

Іншим важливим аспектом є контексти виконання. **Контекст** — це незалежне середовище **Runtime**, у якому виконується певний скрипт або завдання. Різні контексти працюють паралельно у **Runtime**; отже, можна обробляти кілька скриптів і завдань одночасно (рис. 2.6).

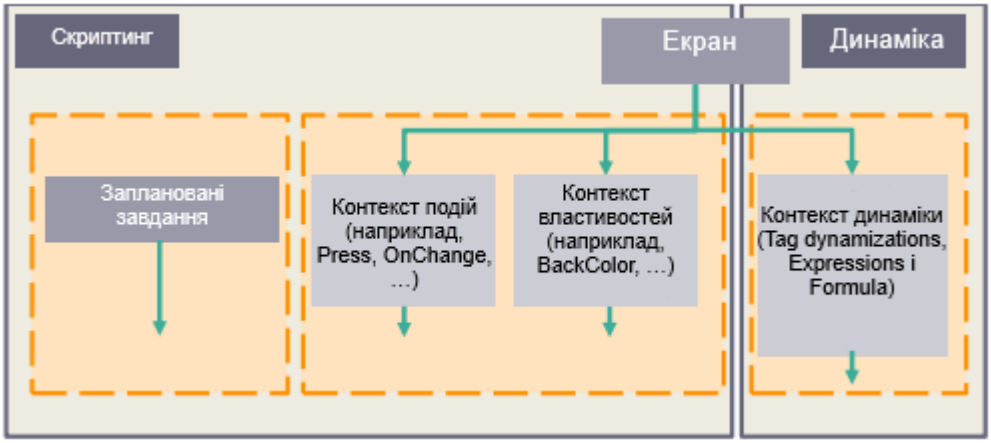


Рисунок 2.6 - Контексти виконання скриптів і динаміки

Як видно з рисунка вище, основний поділ здійснено між скриптами та динамікою. Це вже вказує, чому **Tag dynamization** та **Expressions** мають перевагу над скриптовою динамікою — вони працюють паралельно до всіх виконуваних скриптів.

Для скриптів розділення відбувається між контекстом **Scheduled tasks** і контекстом **Screen**. Кожен екран має власний скриптовий контекст для скриптів і системних функцій. Простір імен (область глобального визначення) є унікальним для кожного скриптового контексту. Скрипти у **Scheduled tasks** або у контексті **Screen** можуть бути викликані циклічно, за подією тега або тривогою.

Обидва варіанти важливі для процедури завантаження: прийняття рішення, чи скрипт реалізується у контексті **Scheduled task** чи **Screen**, а також який тип тригера використовується [13].

На основі цієї інформації вже можна визначити, які конфігурації є релевантними для завантаження екрану (табл. 2.1-2.2).

Таблиця 2.1 - Огляд скриптів, релевантних для завантаження екрана

Скрипт	**Актуальність**	**Додаткова інформація**
Подія **Screen loaded**	✓	**Рисунок 2-5** Процедура
Скрипт у глобальній	✓	–
Scheduled tasks	–	**Рисунок 2-6** Контексти
Скрипти у подіях типу	✓	–
Властивість для скриптів	–	–

Таблиця містить інформацію про різні типи скриптів та їх актуальність, а також додаткові пояснення або зображення, що допомагають зрозуміти їхнє використання. У колонці **"Актуальність"** зазначено, чи є кожен скрипт важливим для конкретної задачі (позначка ✓ або –).

У колонці **"Додаткова інформація"** надано додаткові відомості або посилання на ілюстрації, що пояснюють процеси чи контексти, пов'язані з кожним скриптом.

Відповідно до цієї таблиці, деякі скрипти мають додаткові інструкції або візуалізації, що сприяють кращому розумінню їхнього застосування, в той час як інші скрипти потребують лише базових описів або взагалі не містять додаткових даних.

Це дозволяє оптимізувати використання скриптів в залежності від ситуації, що виникає.

Таблиця 2.2 - Огляд об'єктів, релевантних для завантаження екрана

Об'єкт / Реалізація	**Актуальність**	**Додаткова інформація**
Інтерфейс **Faceplate**	✓	Використання **Faceplates** [5].
Expressions	✓	–
Dynamizations	✓	–
Кількість використаних HMI-тегів	✓	Див. системну інформацію обмеження в Unified manual [9].
Кількість елементів екрана (елементи, екземпляри Faceplate, вікна екрана)	✓	Див. системні обмеження в Unified manual [9].
Тип елемента екрана	✓	Вибір об'єкта екрана [8].

Як ми можемо побачити, існує багато чинників, які впливають на процедуру зміни екрана на **runtime**-пристрої. Щоб надати нам детальну інформацію щодо окремих аспектів, ми ще більше зосередимося на наступних напрямках. Ми також отримаємо опис і **best practice** (найкращі практики) щодо реалізації функцій найбільш ефективним способом [6].

Але спершу ця робота детальніше описує **best practices** для проєктування екранів (**screen engineering**) (рис. 2.7).

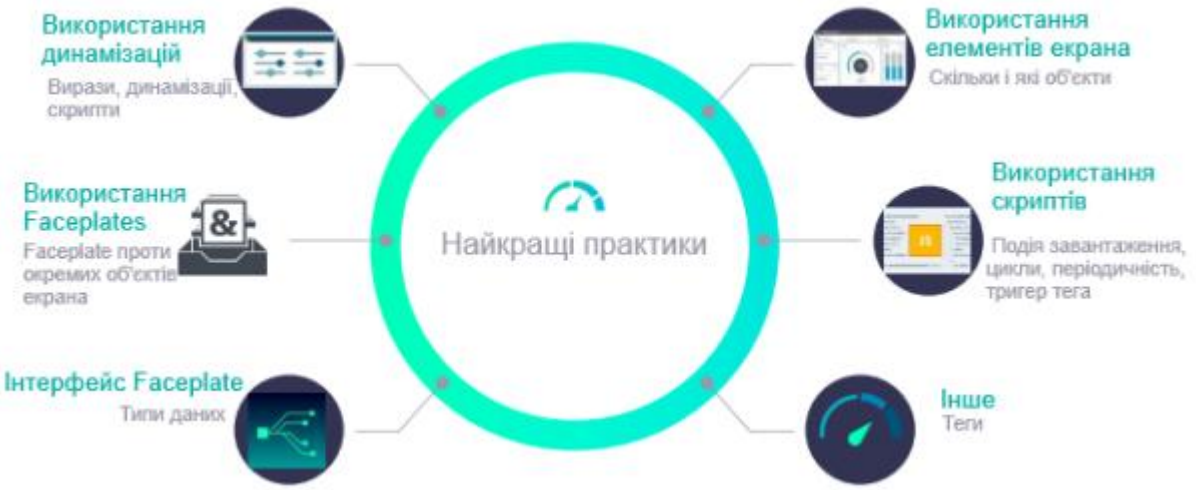


Рисунок 2.7 - Огляд тем найкращих практик (Best Practices Topics Overview)

2.2 Задачі та інструментарій динамізації обєктів

У цьому розділі наведено багато рекомендацій щодо реалізації **HMI**. Тому спочатку ми дізнаємось, на які аспекти ми можемо впливати та які можливості маємо. Потім ми надамо детальне пояснення з ілюстраціями на основі прикладу використання (**use case**). Розділи поділено за такими темами: використання елементів екрана, динамізацій, **Faceplates**, скриптів та інші [7].

Примітка: Щоб краще зрозуміти ці **best practices**, у розділі 2.3 описано загальну процедуру завантаження екрана.

На наступному рисунку (рис. 2.8) зображено доступні елементи з **Unified Toolbox** для **PC-RT** (вміст для **Unified panel** може відрізнятись). Загалом є п’ять основних секцій, що їх розділяють: **Basic objects**, **Elements**, **Controls**, **My controls** та **Dynamic widgets**.

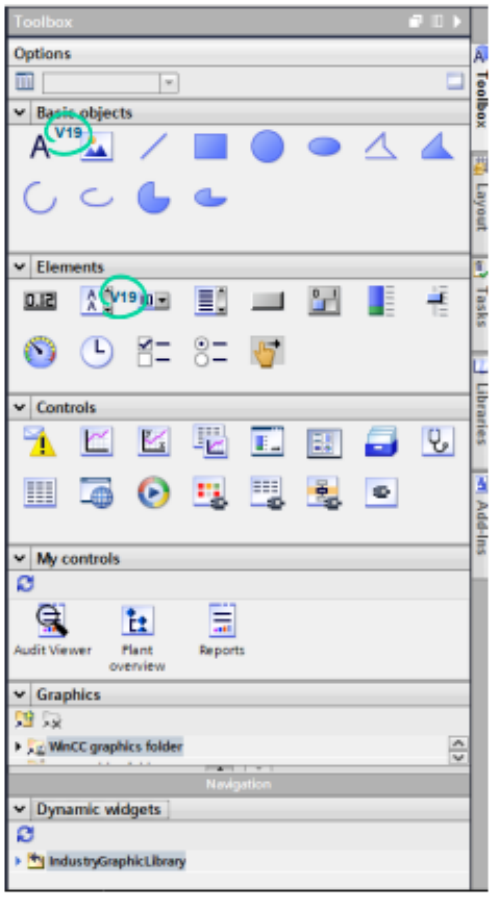


Рисунок 2.8 - Unified Toolbox V19 для PC-RT

Перш ніж продовжити, важливо зазначити, що кожен об’єкт має «footprint» (вплив на продуктивність). Під footprint мається на увазі навантаження, пов’язане з візуалізацією об’єкта. Малий footprint означає незначну кількість властивостей і низьку складність елемента екрана, наприклад, прямокутник. Об’єкти з вищим footprint зазвичай містять велику кількість властивостей, як це притаманно елементам керування (controls).

Окрім фіксованих властивостей, індивідуальні налаштування за допомогою динамізацій також можуть підвищити навантаження на рендеринг елемента екрана, а отже — вплинути на процедуру завантаження всього екрана.

Клієнтська частина побудована із застосуванням фреймворку Vue.js, що забезпечує створення сучасного, адаптивного, інтуїтивно зрозумілого інтерфейсу користувача. Завдяки SPA-архітектурі інтерфейс має високу швидкість завантаження та реагування, що покращує загальний користувацький досвід [5].

По-перше, при розміщенні нового об’єкта на екрані обирайте той, що має найменший слід (footprint) та лише необхідний функціонал. Загалом можна сказати, що складність зростає від Basic об’єктів до Controls, як у порядку в панелі інструментів (toolbox).

Наступна ілюстрація (рис. 2.9) показує всі об’єкти екрана у порядку зростання необхідних ресурсів для рендерингу. Звернемо увагу, що зображений у цій ілюстрації обсяг рендерингу не має часової еквівалентності і лише відображає співвідношення. Фактичні витрати на рендеринг також залежать від конфігурації (динамічні налаштування та підключені дані).

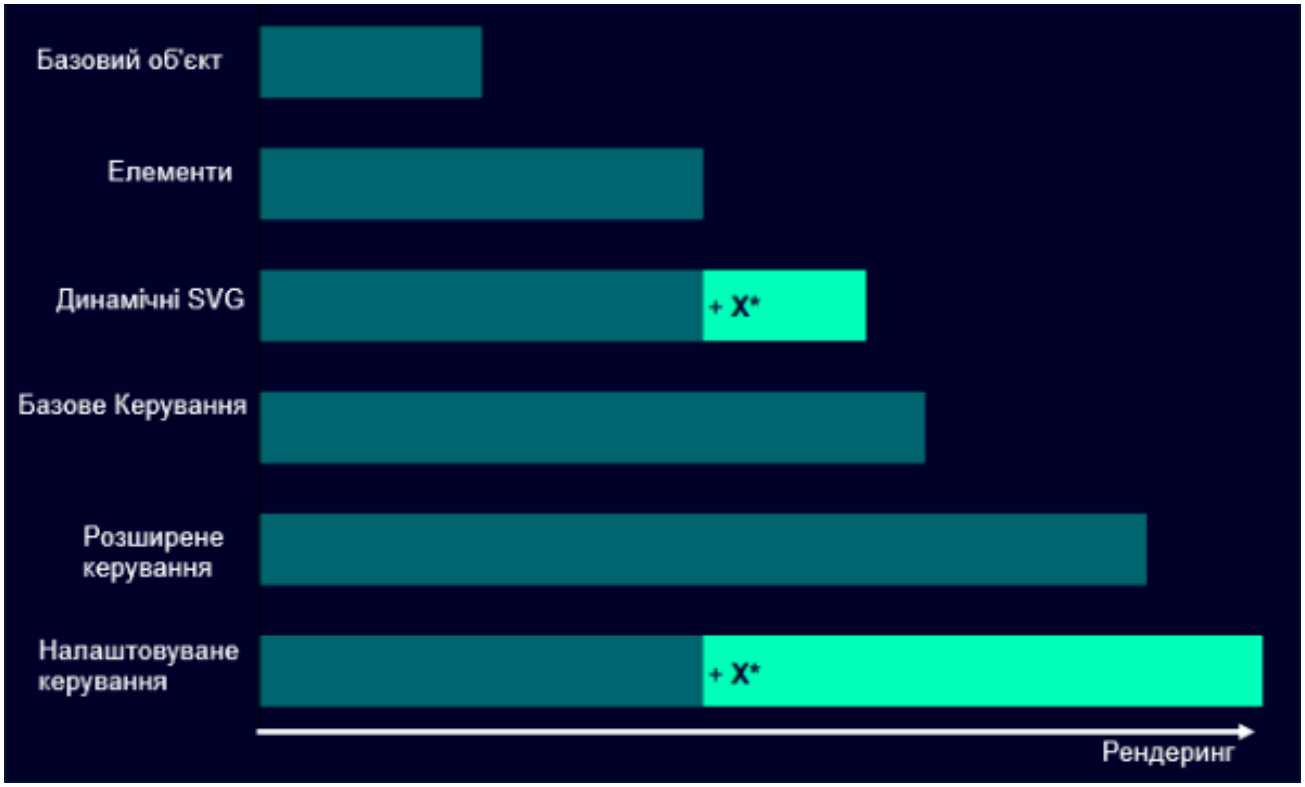


Рисунок 2.9 - Відносне порівняння витрат на рендеринг елементів екрана

Крім того, рендеринг та його час виконання напряму залежать від апаратного обладнання, системи та машини, на якій процес здійснюється. Наявність програмних елментів на зовнішніх носіях даних також впливає на швидкість рендерингу.

У самих групах також є відмінності, і найважливіше зрозуміти, що кожен об’єкт екрана має індивідуальні витрати на рендеринг (footprint). Перед тим як рухатися далі, переконуємося, що ми знайомі з елементами панелі інструментів (toolbox), а також з адаптаціями у версії V19. Адаптація тексту та текстового поля показана на рисунку нижче (рис. 2.10).

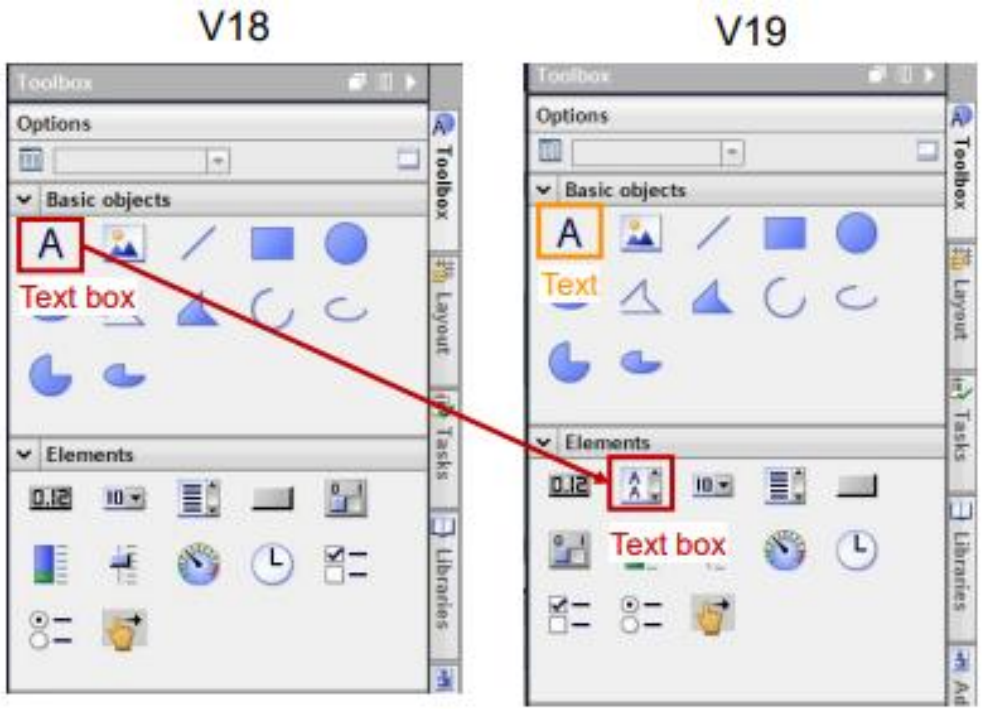


Рисунок 2.10 - Toolbox V18 vs Toolbox V19

Іноді існує більше ніж один спосіб відобразити певну функціональність за допомогою різних об’єктів. Навіть якщо два об’єкти на екрані виглядають однаково при налаштуванні, вони можуть по-різному впливати на час завантаження всього екрану. Нижче наведено кілька прикладів.

Навіть якщо ці приклади не охоплюють повний реальний випадок використання, враховуйте ці моменти при налаштуванні нового екрану.

Цей випадок використання стосується налаштування кольорового квадратного блока, як показано на рисунку нижче. Можливими рішеннями можуть бути прямокутник або текстове поле без призначення тексту, оскільки вони можуть виглядати однаково на екрані [6].

2.3 Параметрування стилів об’єктів мнемосхем

Замість використання текстового поля краще обрати елемент прямокутника через менші витрати на рендеринг (рис. 2.11).

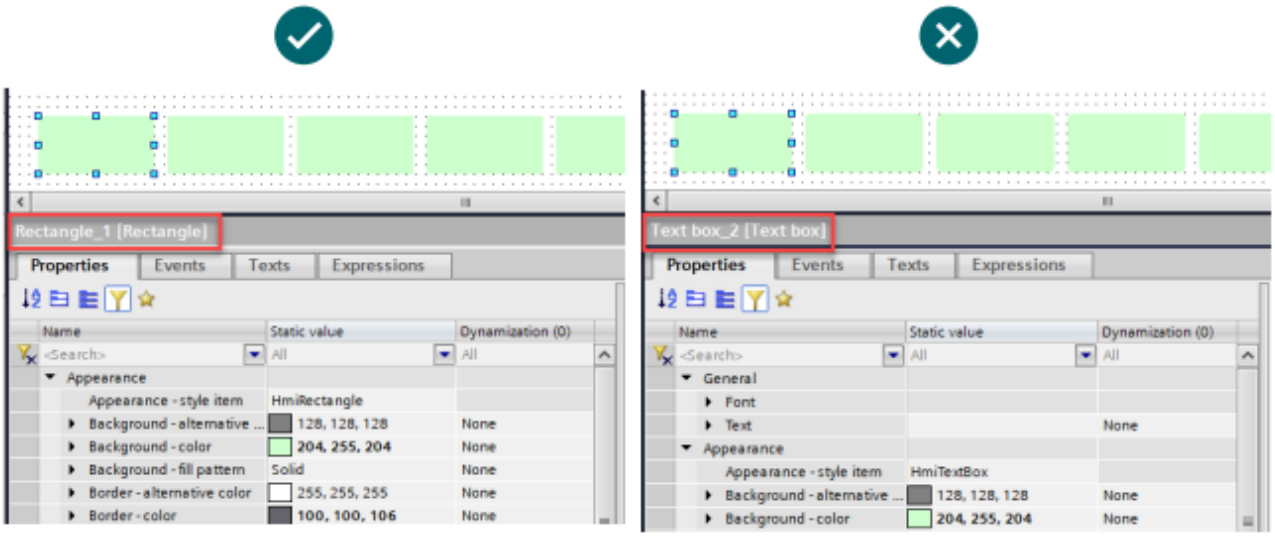


Рисунок 2.11 - Прямокутник vs Текстове поле

Цей випадок використання стосується налаштування кнопки, яка використовує лише подію «Click left mouse button» (натискання лівої кнопки миші) без реалізації подій натискання (press) або відпускання (release). Можливими рішеннями можуть бути текстове поле, текст (V19) або кнопка.

За витратами на рендеринг текст має найменший слід (footprint), далі йде текстове поле, а потім кнопка. Тому використовуємо текстове поле, якщо потрібен фон кольору, інакше – текст, якщо він відповідає нашим вимогам. Також ці елементи підтримують події кліку мишею. Кнопку слід використовувати лише тоді, коли необхідні події натискання і відпускання (рис. 2.12-2.14).

Кнопки надають більше функціональності в контексті взаємодії з користувачем, оскільки вони також можуть вбудовувати складніші механізми обробки подій, такі як анімації або зміни станів, що зазвичай не підтримується тестовими полями. Проте більші витрати на рендеринг не роблять кнопки доцільними поза використанням в описаних вище умовах [8].

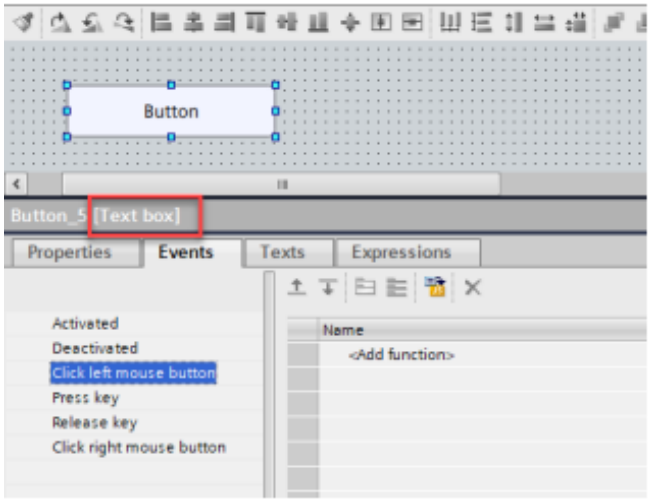


Рисунок 2.12 - Текстове поле як кнопка

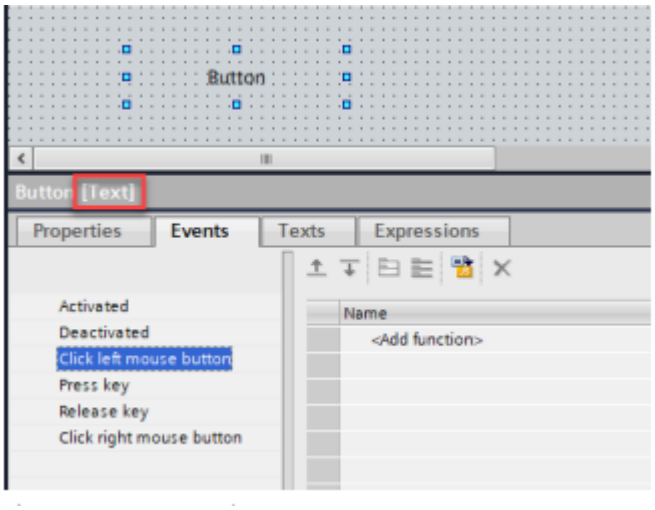


Рисунок 2.13 -Текст як кнопка

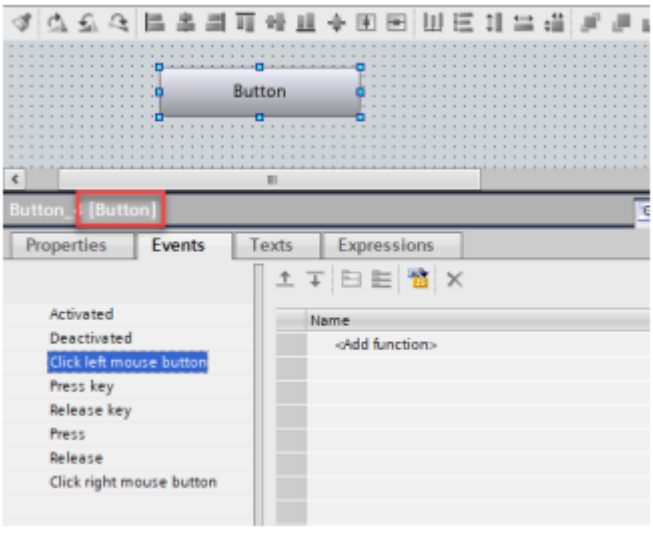


Рисунок 2.14 - Кнопка як кнопка

Якщо можливо, особливо для простих ярликів, слід використовувати Text замість Text box. У наступній таблиці (табл. 3.1) показані різні набори функцій Text у порівнянні з Text box. Через менший функціональний обсяг Text має менший слід (footprint) і витрати на рендеринг на пристрої під час виконання.

Таблиця 3.1 - Порівняння функцій Text та Text box

Властивість	**Text**	**Text box**
Кольори	Передній	Передній план, фон, рамка,
Рамка	✗	Колір, товщина
Формат – Інтервали	✗	✓
Формат – Обрізка	✗	✓
Формат – Перенос	✗	✓

Таблиця порівнює властивості елементів Text (текст) і Text box (текстове поле). Для кожної властивості вказано, чи підтримується вона тим чи іншим елементом.

1. Кольори: Текст підтримує лише передній план, тоді як текстове поле дозволяє налаштовувати передній план, фон, рамку та альтернативні кольори.

2. Рамка: Текст не має можливості налаштування рамки, а текстове поле дозволяє змінювати колір та товщину рамки.

3. Формат – Інтервали, Обрізка тексту, Перенос тексту: Ці властивості підтримуються тільки текстовим полем, дозволяючи додаткову налаштування відображення тексту (відступи, обрізання, автоматичний перенос).

На наступних рисунках показаний процес налаштування цих функцій (рис. 2.15-2.16).

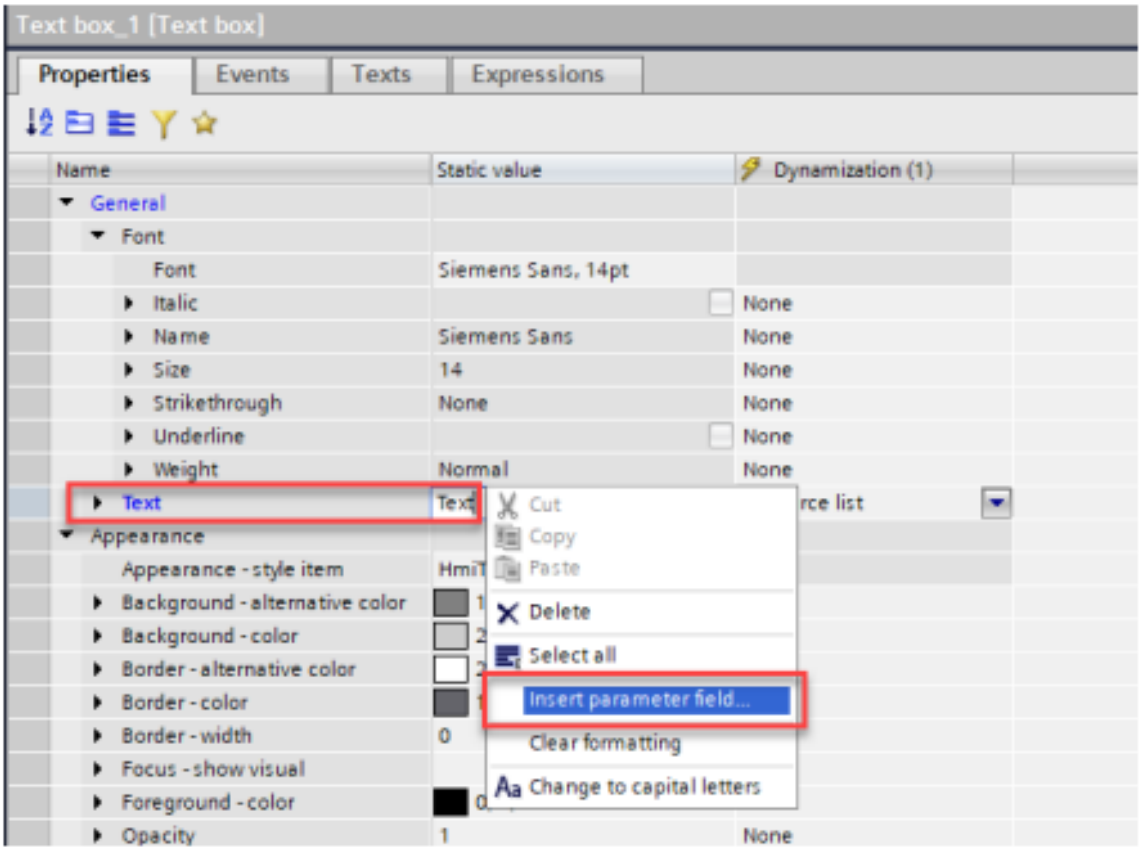


Рисунок 2.15 - Поле параметрів властивостей Text

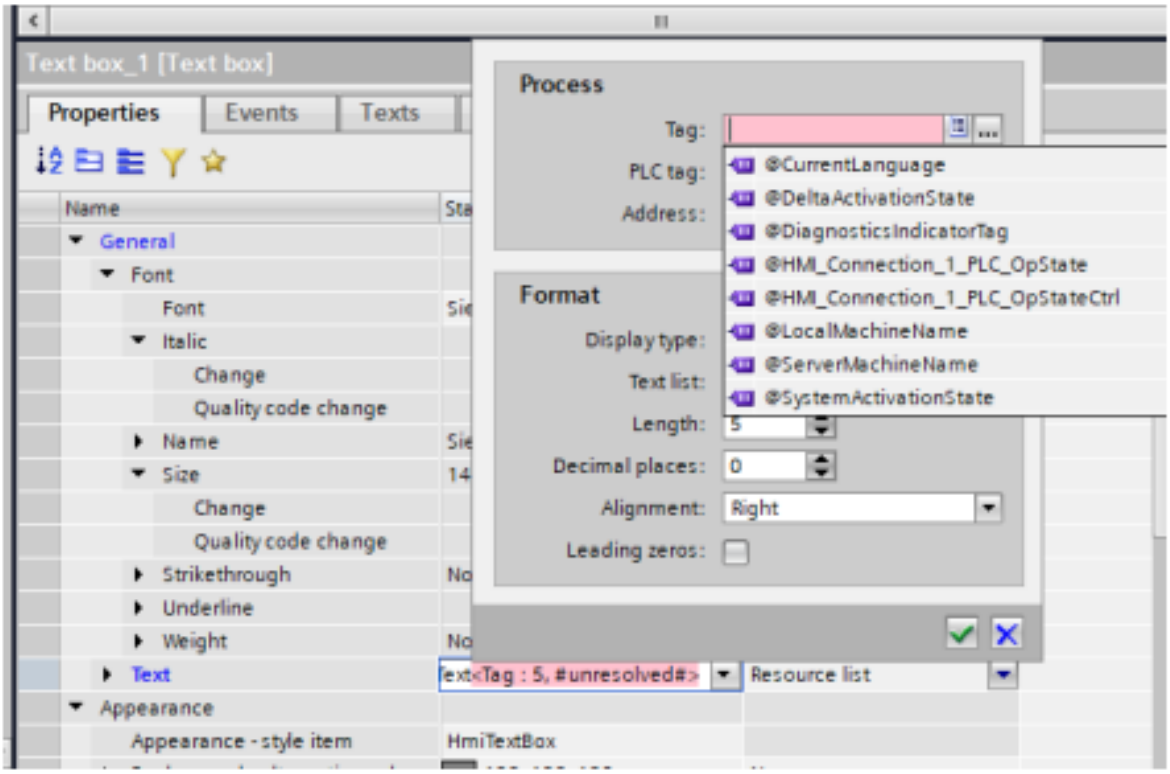


Рисунок 2.16 - Поле параметрів властивостей Text

Випадок використання: Кольоровий фон екрана
Опис

У деяких випадках потрібно змінити колір фону екрана за замовчуванням.

Рішення

Замість того, щоб додавати на екран додатковий прямокутник і використовувати його колір фону як фон екрана, змініть колір безпосередньо у властивостях елемента екрану. Це також актуально для Faceplates (рис. 2.17).

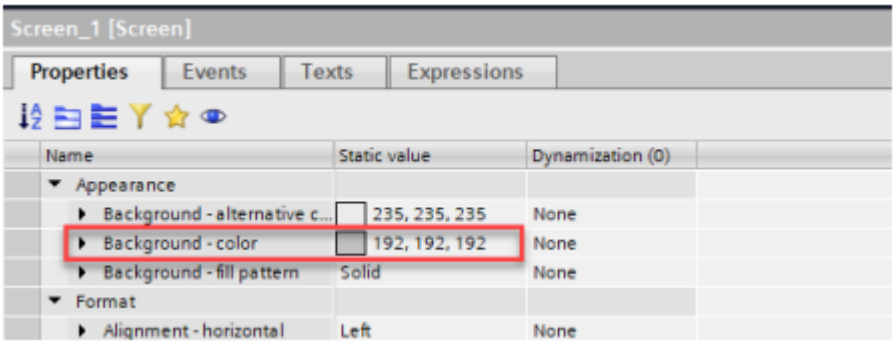


Рисунок 2.17 - Налаштування кольору фону елемента екрану

Варіант використання: Custom Styles

Опис

Часто корпоративне оформлення включає також корпоративну палітру кольорів і дизайн. За допомогою властивостей об'єктів елементи екрану можна адаптувати до цього оформлення та дизайну. Проте загальний вигляд об'єктів, таких як кнопки, також може бути частиною корпоративного дизайну, як і візуалізація більш складних об'єктів, наприклад слайдерів.

Рішення

Завжди бажано використовувати об’єкти екрана Unified відповідно до їхнього призначення. Відтворення користувацьких дизайнів елемента за допомогою декількох екранних об'єктів слід уникати, оскільки це призводить до збільшення кількості об'єктів і зазвичай — до додаткових невикористовуваних функцій допоміжних елементів. Також використання Faceplates як методу стандартизації об'єктів корпоративного дизайну створює зайве навантаження [7].

Використовуйте **SIMATIC WinCC Unified Corporate Designer**, щоб створювати такі дизайнерські об'єкти, а також кольори як частину власного стилю, і використовуйте цей стиль у вашому проєкті [9].

У **Corporate Designer** можна створити новий стиль на основі наявних стилів. Далі ви можете змінити стиль існуючих об’єктів або створити повністю нові об’єкти. Ці стилі також можуть бути пов’язані з палітрами кольорів і шрифтами.

Після завершення розробки стилю його необхідно експортувати, а згенерований файл скопіювати до директорії проєкту (Userfiles > Styles). Після цих кроків стиль можна буде обрати в налаштуваннях runtime (рис. 2.18-2.21).

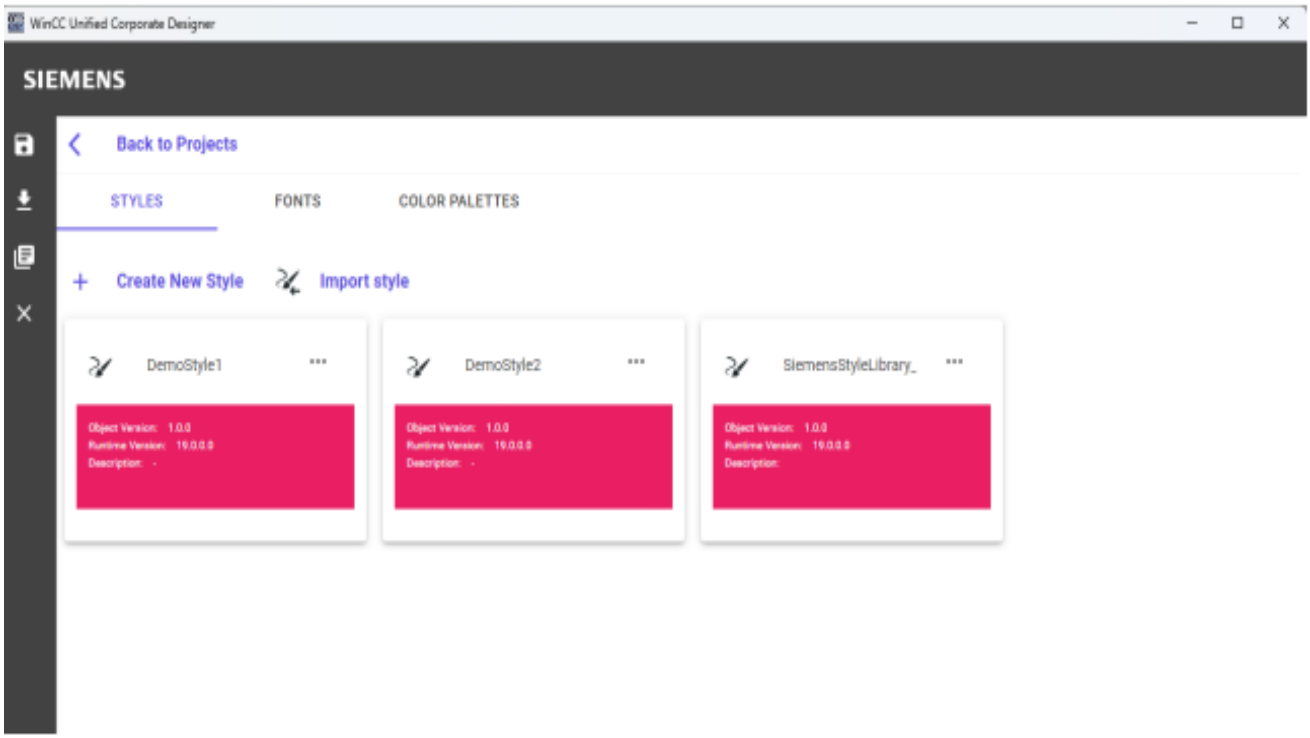


Рисунок 2.18 - Вигляд проєкту в Unified Corporate Designer

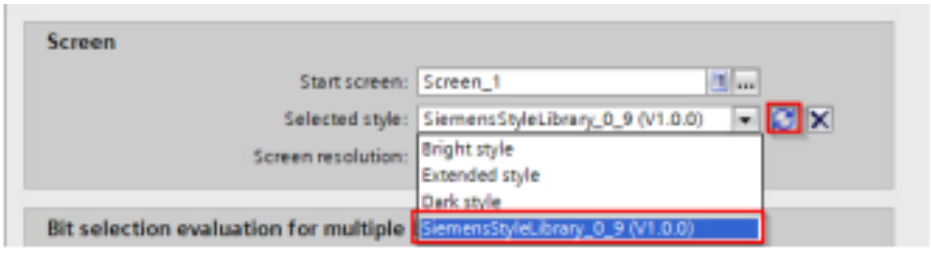


Рисунок 2.19 - Вибір стилю для Unified Runtime

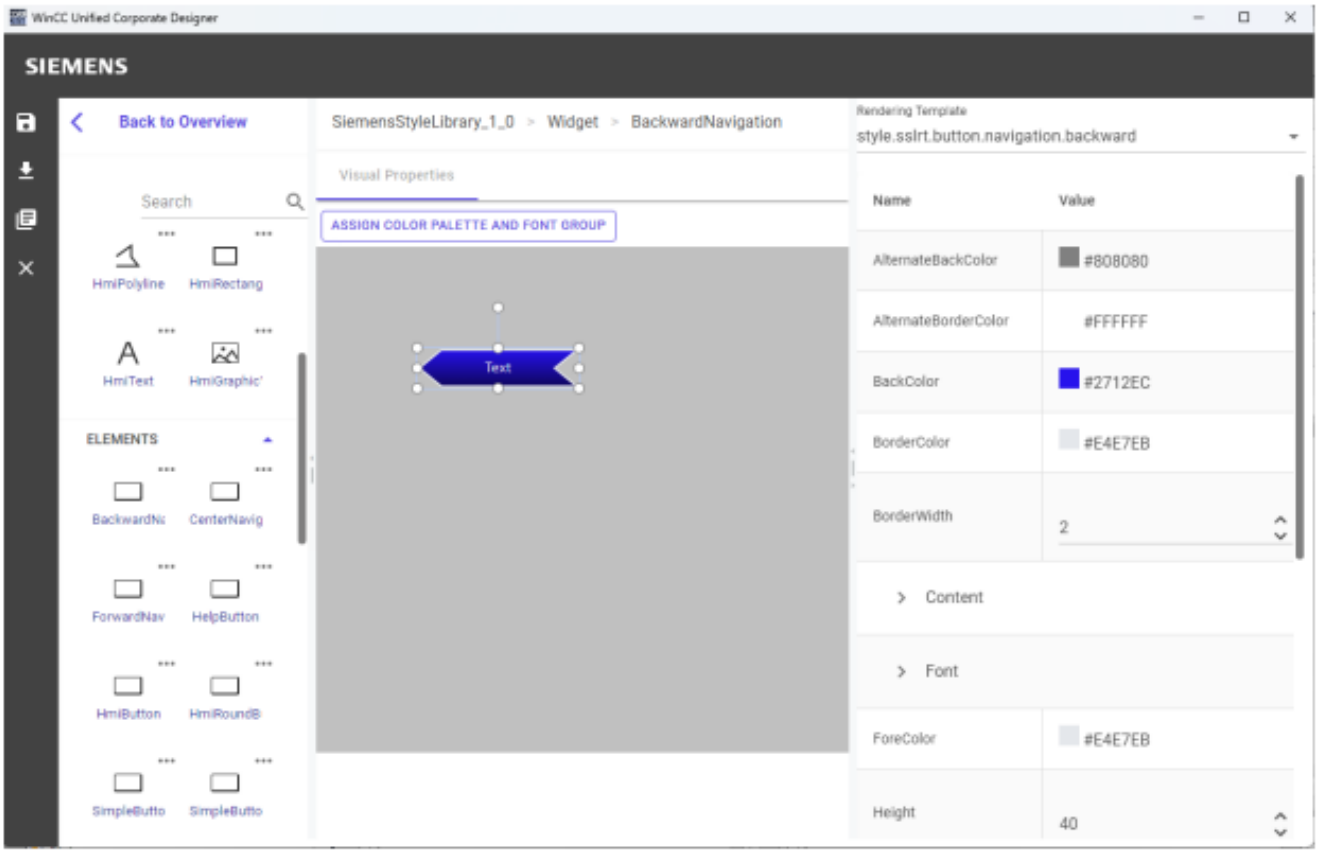


Рисунок 2.20 - Unified Corporate Designer: Edit View

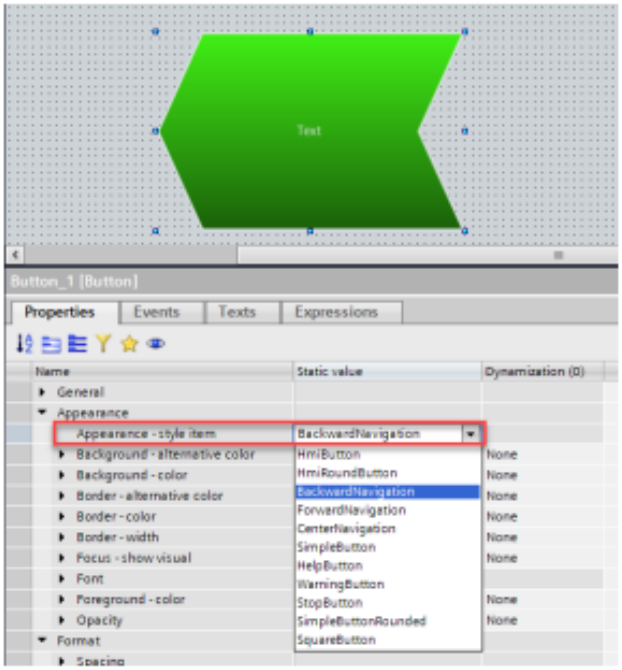


Рисунок 2.21 - Зміна елемента стилю системних та користувацьких екранних об’єктів у середовищі розробки (engineering)

Корпоративний редактор задіює додаткові ресурси візуалізації, що збільшу час завантаження продуктів. Як ми вже бачили, існують елементи, що потребують більше ресурсів для візуалізації, і ті, що потребують менше. У будь-якому випадку кожен елемент на екрані впливає на час завантаження, і при цьому визначено системні обмеження, яких необхідно дотримуватись [7].

Щоб досягти найкоротшого часу завантаження екрана, особливо безпосередньо перед тим, як проєкт почне використовуватись у продуктивному середовищі, усі об’єкти, які були створені лише для етапу розробки — зневадження або інших допоміжних цілей — слід видалити:

- видаляйте невикористовувані об’єкти, що знаходяться на фоні або не є видимими;

- видаляйте невикористовувані об’єкти, що знаходяться за межами екрана. Також у розділі компонування (layout) доступна картка завдань, яка вказує на об’єкти за межами ділянки перегляду (рис. 2.22).

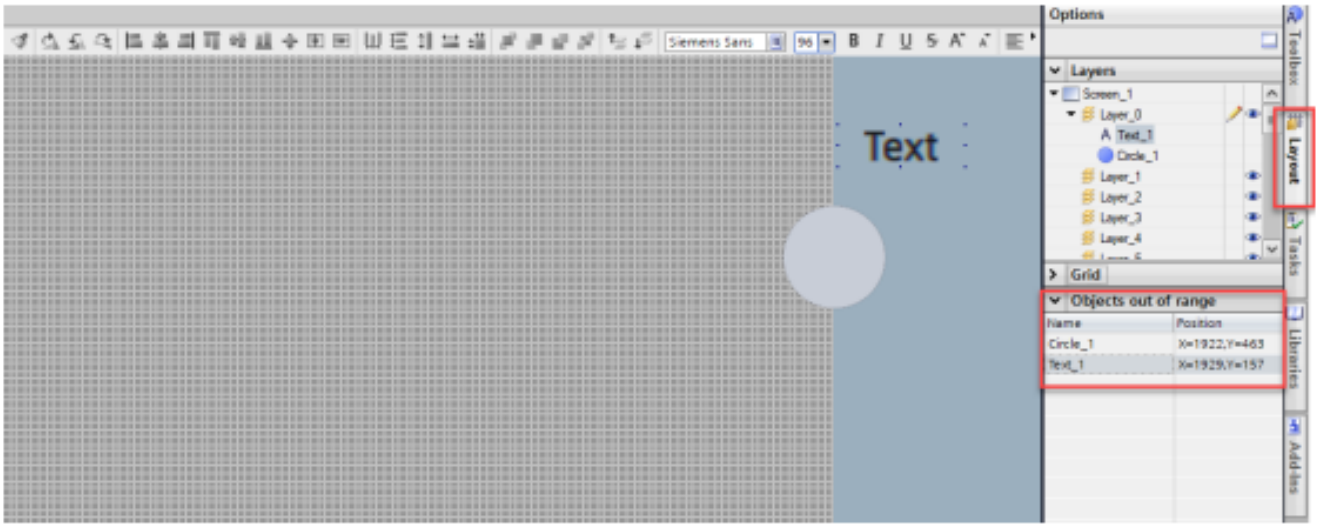


Рисунок 2.22 - Картка завдань: об’єкти за межами ділянки перегляду

2.4 Структурна організація мнемосхем

Екранні вікна часто використовуються для створення макету екрана та для відокремлення індивідуального вмісту від загального, який є видимим на кількох представленнях і не потребує завантаження при кожній зміні екрана (наприклад, Header). Вони також застосовуються для спливаючих екранів (pop up screens).

Щоб зрозуміти, коли слід використовувати screen window як pop up і коли його слід конфігурувати вже під час виконання (runtime), дотримуйтеся наведеної нижче блок-схеми (рис. 2.23).

Рекомендація наведена лише з позиції належної інженерної практики. Матимемо на увазі, що екранні вікна загалом мають іншу поведінку, ніж pop up-екрани, створені системною функцією **OpenScreenInPopUp**, з огляду на масштабування (zoom), прокрутку (scroll), позицію (position), шар екрана (screen layer) та час життя (live time). Отже, конкретний сценарій використання може вимагати явного вибору одного з варіантів реалізації.

Екранні вікна зазвичай використовуються для організації та структурування інтерфейсу, особливо коли потрібно відокремити окремі елементи, які не потребують частого оновлення або завантаження. Це дозволяє оптимізувати роботу системи та зменшити навантаження на ресурси. Вони можуть бути корисними в таких ситуаціях, як відображення статичних елементів, таких як заголовки (Header), навігаційні панелі або інші елементи інтерфейсу, що залишаються незмінними під час роботи з різними вікнами.

У той час як екранні вікна мають стабільну та передбачувану поведінку, pop up-екрани, як правило, мають іншу мету — забезпечення швидкої взаємодії з користувачем, не блокуючи основне вікно.

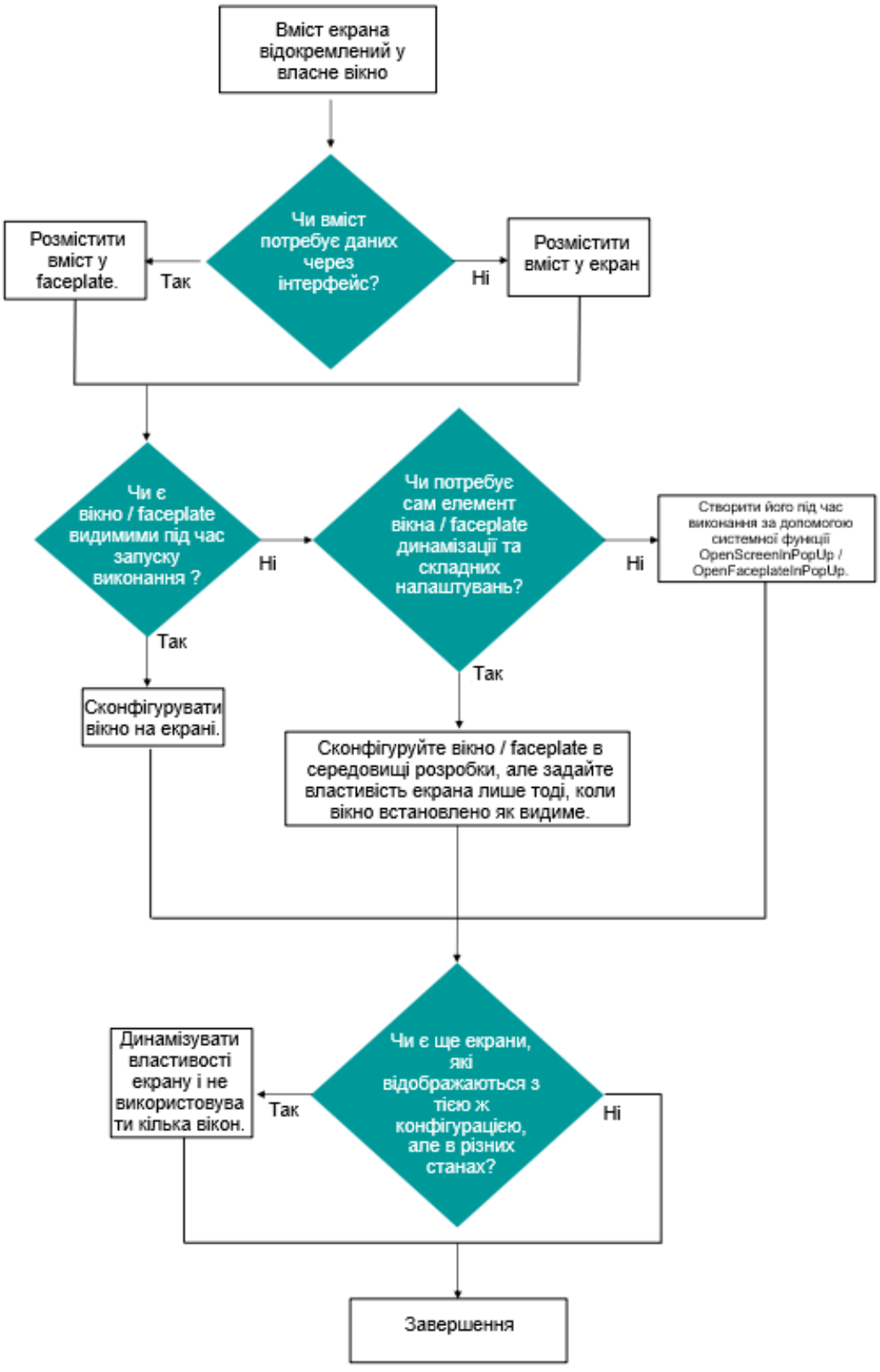


Рисунок 2.23 - Конфігурація screen windows

Якщо видимість елемента екрана динамізується, слід встановити статичне значення у false, особливо коли можна припустити, що використана динамізація, наприклад, динамізація через тег, спричиняє невидимість об’єкта під час завантаження екрана (рис. 2.24).

Рисунок 2.24 Статичне значення для властивості Visibility, якщо елемент припускається невидимим під час завантаження екрана

Опис

Видимість кількох елементів екрана потрібно змінювати під час виконання (runtime) за однією й тією ж умовою.

Рішення 1

Якщо видимість кількох об’єктів змінюється залежно від однієї умови, визначте один тег, щоб динамізувати видимість усіх об’єктів одночасно (рис. 2.25-2.26).

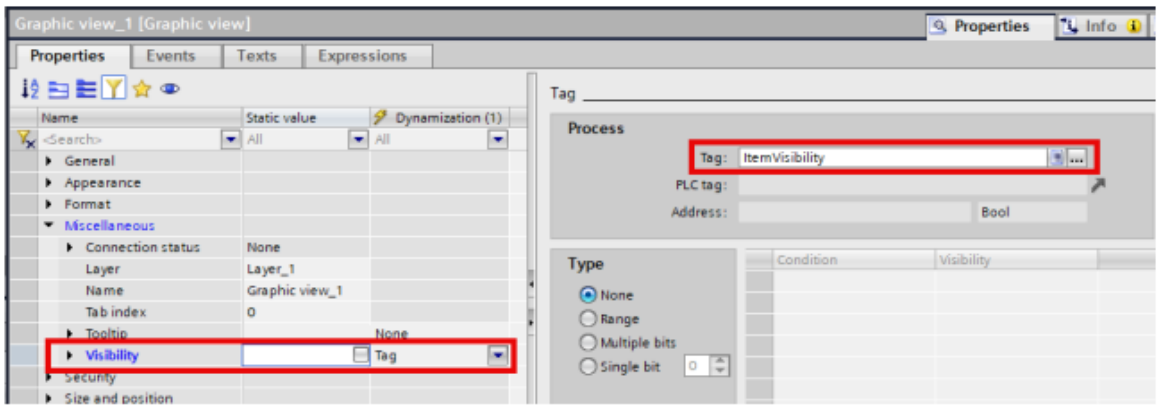


Рисунок 2.25 - Видимість об’єкта екрана під час виконання

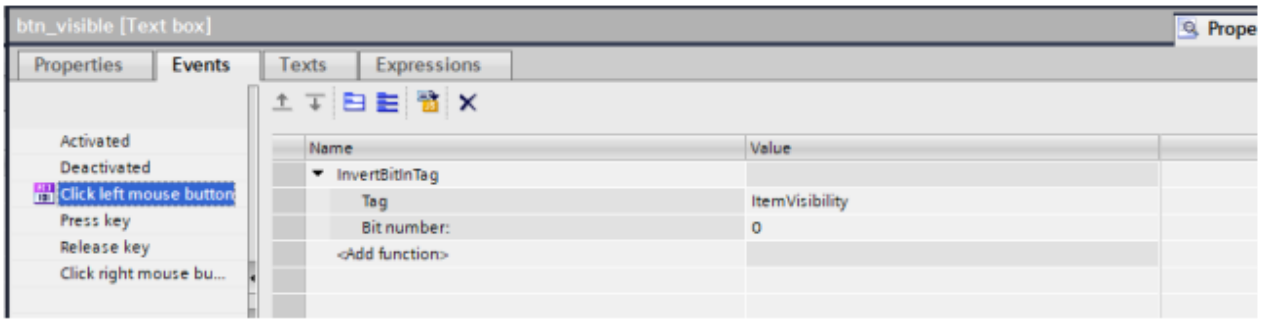


Рисунок 2.26 - Перемикання видимості об’єкта:

Рішення 2

Якщо змінюється лише видимість залежно від однієї умови, перемістіть усі об’єкти в один шар і динамізуйте видимість цього шару під час виконання (runtime). Видимість шару також можна змінювати в runtime за допомогою скриптування (рис. 2.27-2.28).

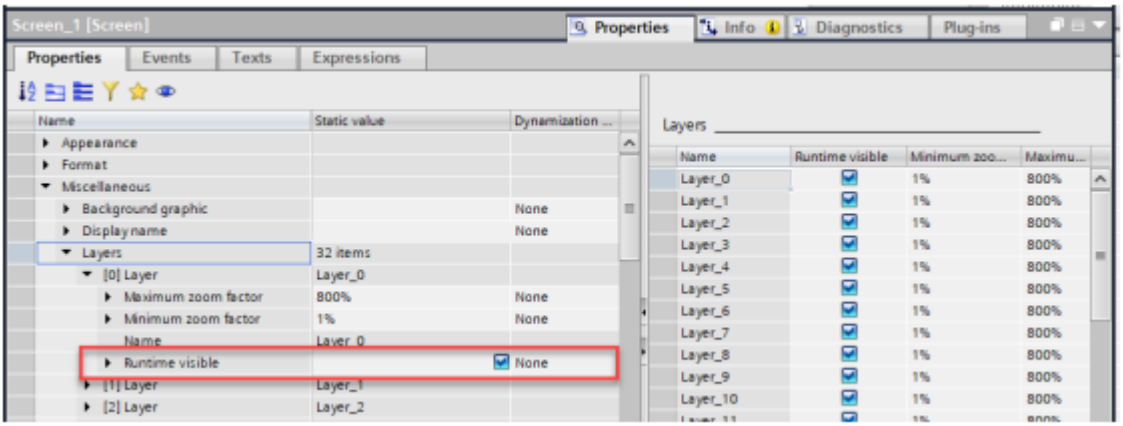


Рисунок 2.27 - Видимість шару екрана під час виконання

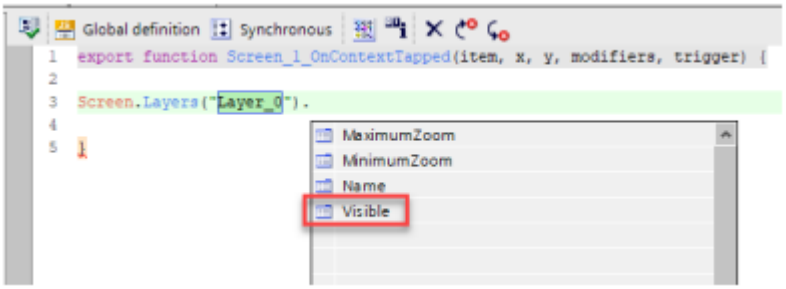


Рисунок 2.28 - Доступ до видимості шару в runtime

Рішення 3

Якщо існує кілька властивостей, що мають однакову поведінку або умову (наприклад, колір фону, авторизація тощо), згрупуйте об’єкти і налаштуйте зміну властивостей для всієї групи одночасно (рис. 2.29).

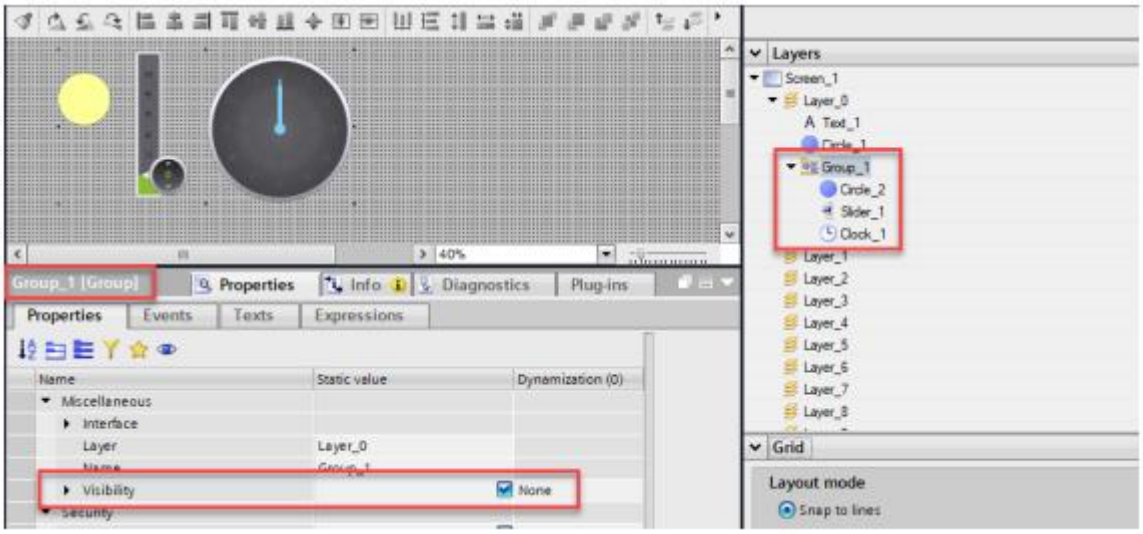


Рисунок 2.29 - Динамізація видимості групи елементів екрана

Уніфіковане керування (Unified Controls)

Уніфіковане керування значно полегшує реалізацію типових випадків використання, таких як сигналізація, звітування та контроль трендів.

Проте в цій сфері також є аспекти, які варто враховувати під час використання.

Випадок використання: Різні налаштування в runtime для уніфікованого керування

Існує багато налаштувань, які можна зробити в середовищі розробки (engineering) для уніфікованих керувань. Іноді під час виконання (runtime) потрібен контроль з різними налаштуваннями. Через дуже обмежені можливості динамізації контролів у попередніх проєктах WinCC Basic, Comfort та Advanced було поширеною практикою розміщувати на екранах кілька статично параметризованих контролів і робити їх видимими по черзі під час виконання [8].

WinCC Unified надає дуже гнучкі, конфігурувані та динамізовані контролі, тому потребу в різних налаштуваннях під час виконання можна задовольнити шляхом реалізації та керування одного контролю.

Рішення

Змінюйте властивості під час виконання, не налаштовуйте кілька контролів і не змінюйте екран для реалізації різних налаштувань. Для цього використовуйте системну функцію **SetPropertyValue** або об'єктну модель **Screen.Items("Itemname")**. Імена властивостей легко копіювати в середовищі розробки з вкладки властивостей (properties tab) (рис. 2.30).

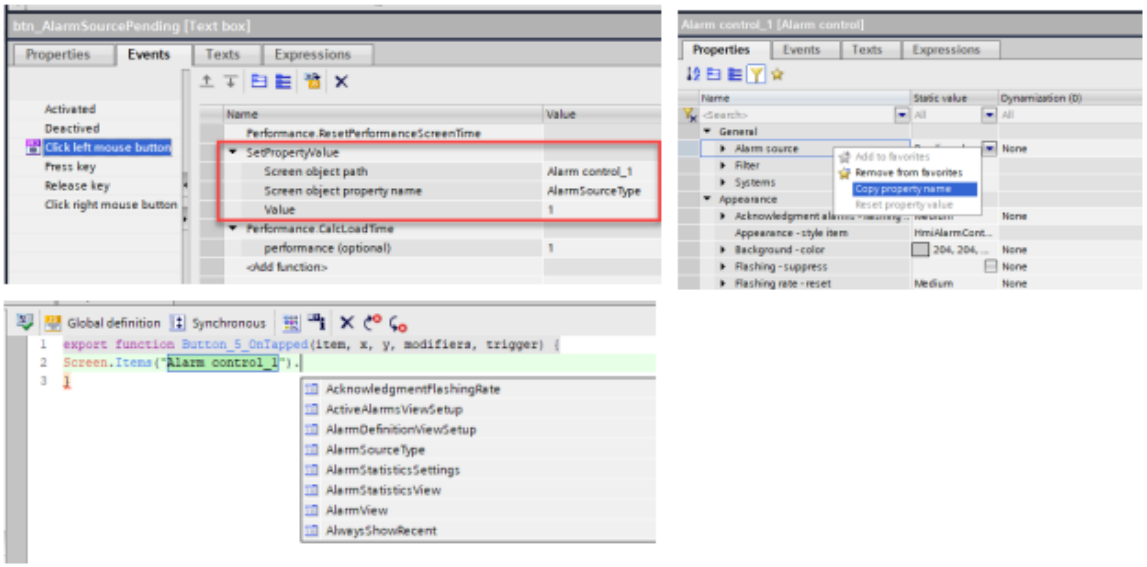


Рисунок 2.30 Доступ до властивостей Unified control

Всі доступні властивості можна знайти в об'єктній моделі WinCC Unified безпосередньо в довідці TIA або в посібнику **SIMATIC HMI WinCC Unified V20 - PLC Programming with SIMATIC S7** [9].

2.5 Організація і параметрування сигналізацій в проектах

Опис

Для загального доступу до сигналізацій (alarms) надається контрол Alarm View. Цей контрол має багато функціональності як у середовищі розробки (engineering) для налаштування, так і під час виконання (runtime). Крім того, існує багато системних функцій, які дозволяють отримати доступ до тієї ж функціональності через скриптування, без використання Unified control.

Рішення

Оскільки ми загалом рекомендуємо проєктувати макет екрану з використанням screen windows , існують екрани, що змінюють вміст, та екрани, які залишаються незмінними. Alarm control належить до елементів екрана з великим ресурсним навантаженням, тому рекомендується розміщувати його у screen windows, які не перезавантажуються часто, наприклад, у Header.

Оскільки ці screen windows, як Header, не дуже великі, можна налаштувати Alarm View у вигляді Alarm Line за наступними кроками:

1. Використайте спрощений стиль відображення (simplified appearance style) (рис. 2.31).

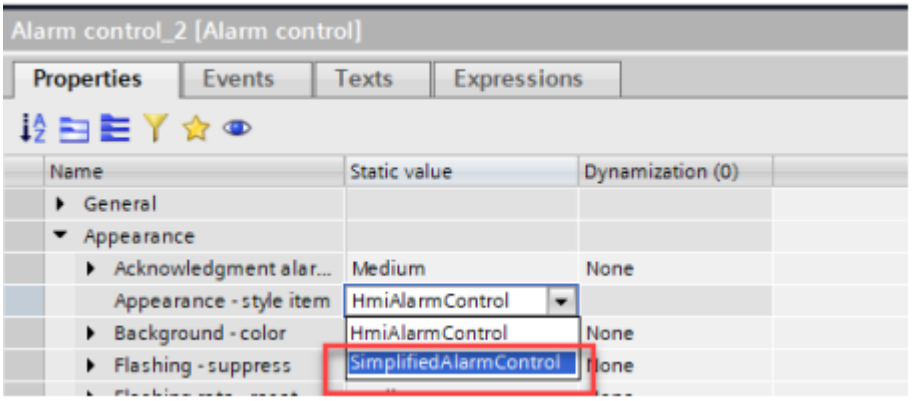


Рисунок 2.31 - Спрощений стиль Alarm Control

2. Встановіть налаштування вікна на «завжди зверху» (рис. 2.32).

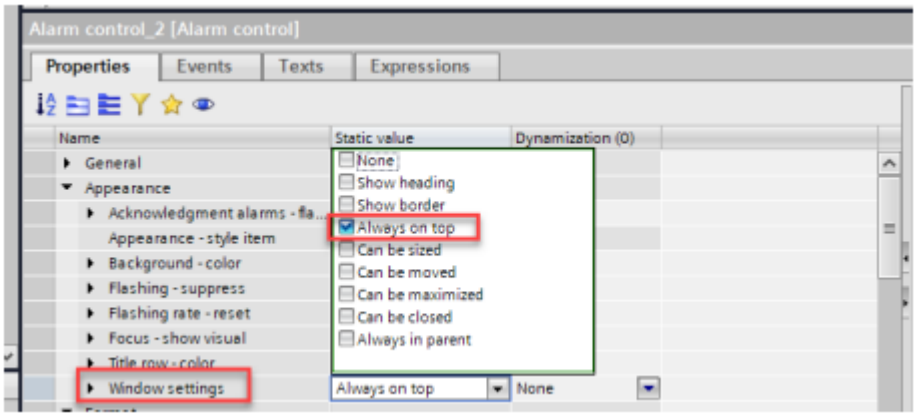


Рисунок 2.32 - Налаштування вікна Alarm Control

3. Встановіть напрям сортування за зростанням (ascending) (рис. 2.33).

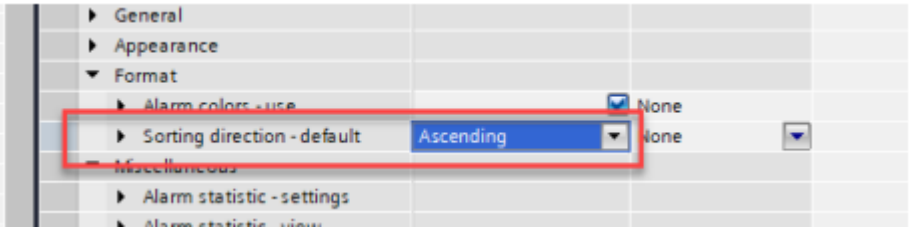


Рисунок 2.33 - Напрям сортування Alarm Control

4. Перейдіть до Miscellaneous > Alarm view > Header-settings і змініть заголовок колонки (column header) та заголовок рядка (row header) на «None» (рис. 2.34).

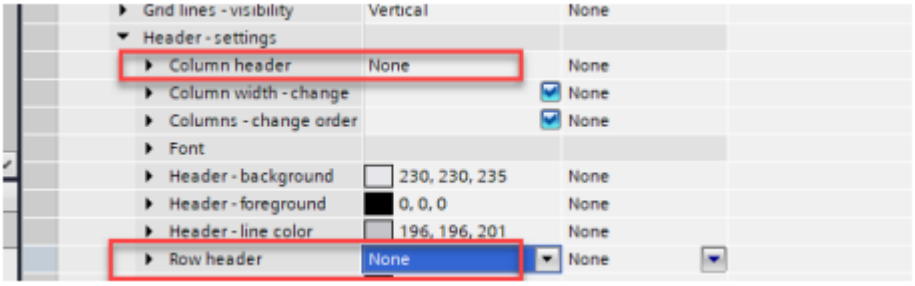


Рисунок 2.34 - Заголовки колонок та рядків Alarm Control

5. Перейдіть до Miscellaneous > Alarm view та сховайте (collapse) горизонтальні та вертикальні смуги прокручування (рис. 2.35).

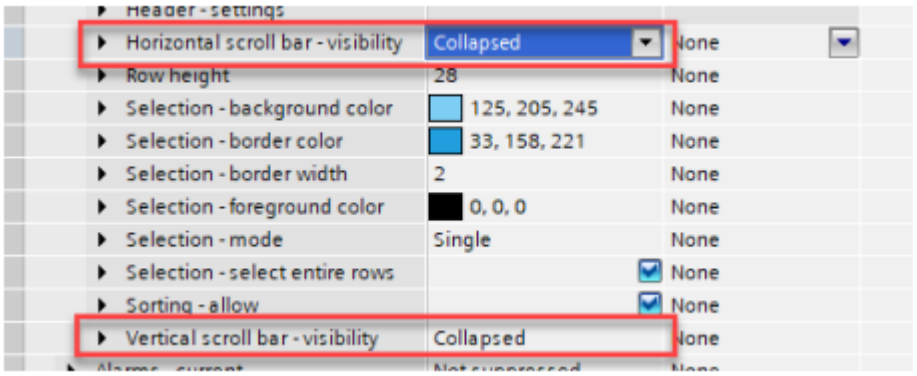


Рисунок 2.35 - Горизонтальні та вертикальні смуги прокручування Alarm Control

6. Підберіть розмір для одного або кількох рядків відображення (рис. 2.36).

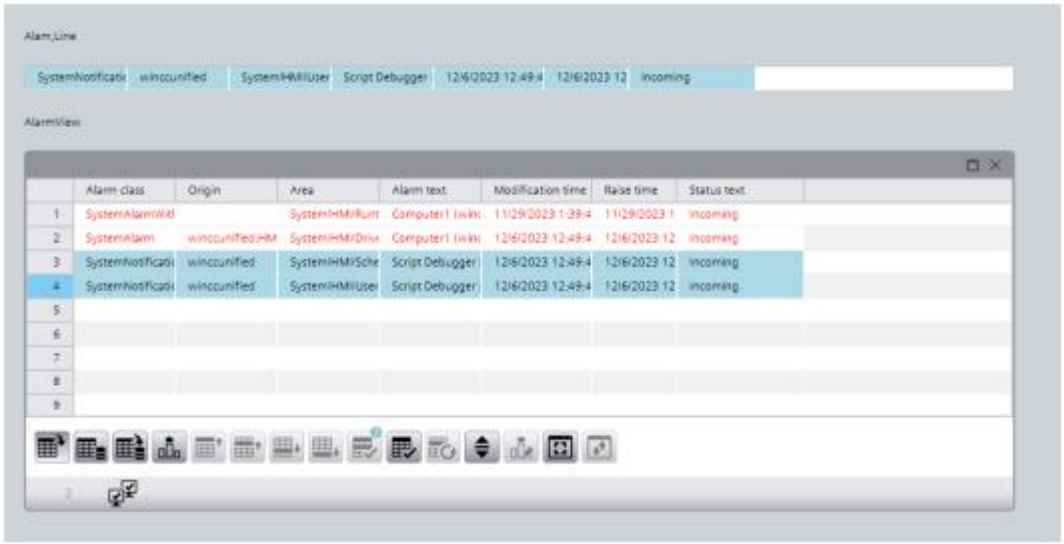


Рисунок 2.36 - Alarm line проти Alarm View

Якщо потрібен Alarm View з повною функціональністю, показуйте його за потребою у вигляді спливаючого вікна (pop up).

Випадок використання: Кастомне рішення Alarm Line

Опис

Для загального доступу до сигналізацій (alarms) надається Alarm View. Як було згадано раніше (випадок використання: Alarm Control), якщо ця інформація повинна відображатися постійно в видимому **screen window**, наприклад, у Header, спрощений стиль відображення (simplified appearance style) є хорошим рішенням. Але це не рекомендується, якщо екран, на якому відображаються тривоги, часто перезавантажується [13].

Реалізація кастомного рішення Alarm Line включає такі кроки:

1. Створіть новий Global Module у вашому проекті TIA Portal та скопіюйте код з файлу **Alarmline.js** (рядки 1–171) у область глобальних визначень. Тут визначено клас Alarm Manager із кількома методами та властивостями (ініціалізує екземпляр Alarm Manager із заданими опціями, запускає підписку на тривоги, задає порядок сортування і зупиняє підписку).

2. Додайте нову функцію **UpdateActiveAlarms** (рис. 2.37) та скопіюйте відповідний код з **Alarmline.js** (рядки 174–201). Ця функція оновлює теги тривог на основі наданого масиву тривог, створює набір тегів і оновлює їх значення [9].

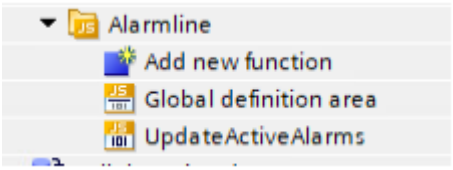


Рисунок 2.37 - Глобальні модулі кастомного рішення Alarm Lin

3. Створіть нове заплановане завдання (Scheduled Task), наприклад, **AlarmUpdate**, і виберіть тег **@SystemActivationState** як тригер. Це завдання буде оновлювати тривоги, запускаючи підписку на них відповідно (рис. 2.38).

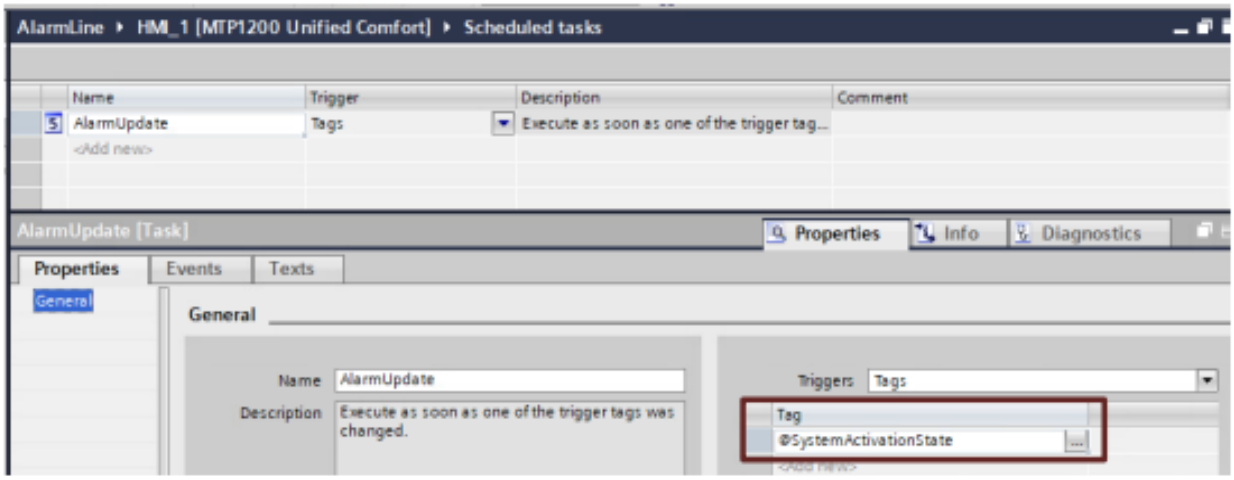


Рисунок 2.38 Заплановане завдання кастомного рішення Alarm Line

4. Використайте код з файлу **ScheduledTasks.js** у події **Event > Update** вашого нового завдання. У глобальних визначеннях скопіюйте код із рядків 1 до 9, а для самої події — код з рядка 11 до кінця.

5. Для візуалізації тривог можна налаштувати базові об’єкти та елементи з Toolbox (див. Demo-проєкт).

6. Створіть необхідні HMI теги для відображення значень Alarm Line та динамізації властивостей. Кількість потрібних тегів залежатиме від максимальної кількості тривог, які ви хочете показувати одночасно (рис. 2.39).

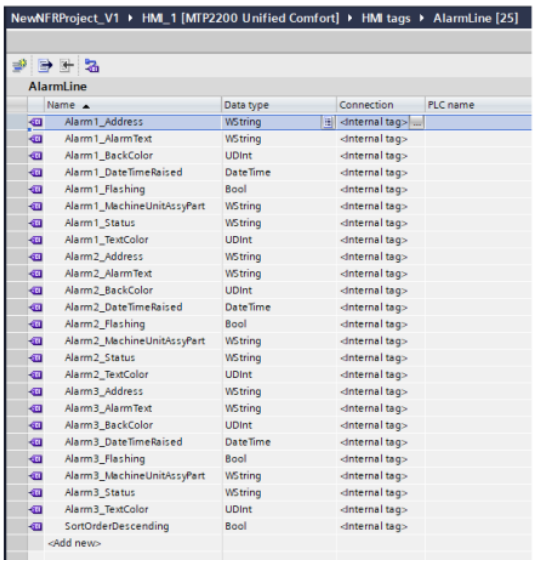


Рисунок 2.39 - Кастомні теги Alarm Line

Максимальна кількість тривог: максимальна кількість тривог, які відображаються одночасно, може бути змінена у глобальних визначеннях модуля Alarmline, у рядку 164. Врахуємо, що при збільшенні кількості тривог для відображення потрібна більша кількість HMI тегів та базових елементів для їх показу (рис. 2.40).

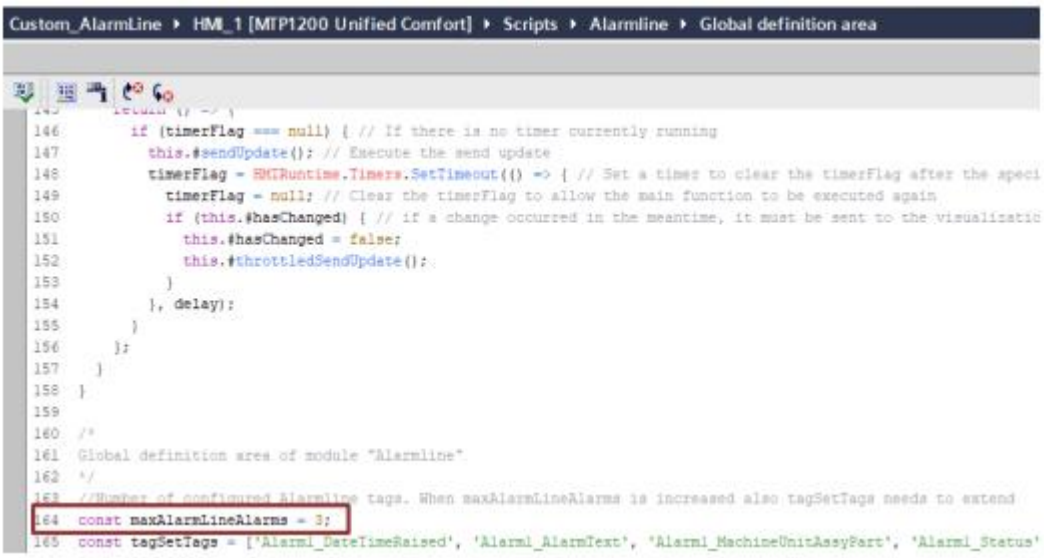


Рисунок 2.40 - Кастомна Alarm Line: MaxAlarmLineAlarms

Мова тривог (Alarm language):

У коді події запланованого завдання (Scheduled task event code) мову тривог можна встановити, змінивши значення (десятковий ідентифікатор мови) у рядку 7. У цьому прикладі «1033» відповідає англійській мові (тут можна знайти інші ідентифікатори мов). Також можна використати значення «127», що є мовою за замовчуванням, налаштованою у вашому HMI.

Фільтр тривог (Alarm filter):

У глобальних визначеннях запланованого завдання налаштовується фільтр для тривог. Його також можна змінити відповідно до ваших потреб.

Інтервал оновлення (Update interval, опціонально):

У коді події запланованого завдання можна додати параметр **delayInMilliseconds** до виклику Alarm Manager(). Рекомендований час - 250 мс.

Звернемо увагу, що збільшення цього значення не означає, що оновлення не відбуватиметься вчасно. Цей параметр працює так: уявіть, що встановлено затримку 10 000 мс (10 секунд). Якщо виникає нова тривога, вона відразу надсилається на візуалізацію. Якщо через 2 секунди з'являється інша тривога, вона не буде показана відразу, а з'явиться через 8 секунд (10 секунд затримки мінус 2 секунди). Це дозволяє вчасно бачити тривоги та запобігати перевантаженню системи.

Цей параметр є необов'язковим — ви можете вибрати затримку на свій розсуд або не використовувати її взагалі (рис. 2.41).

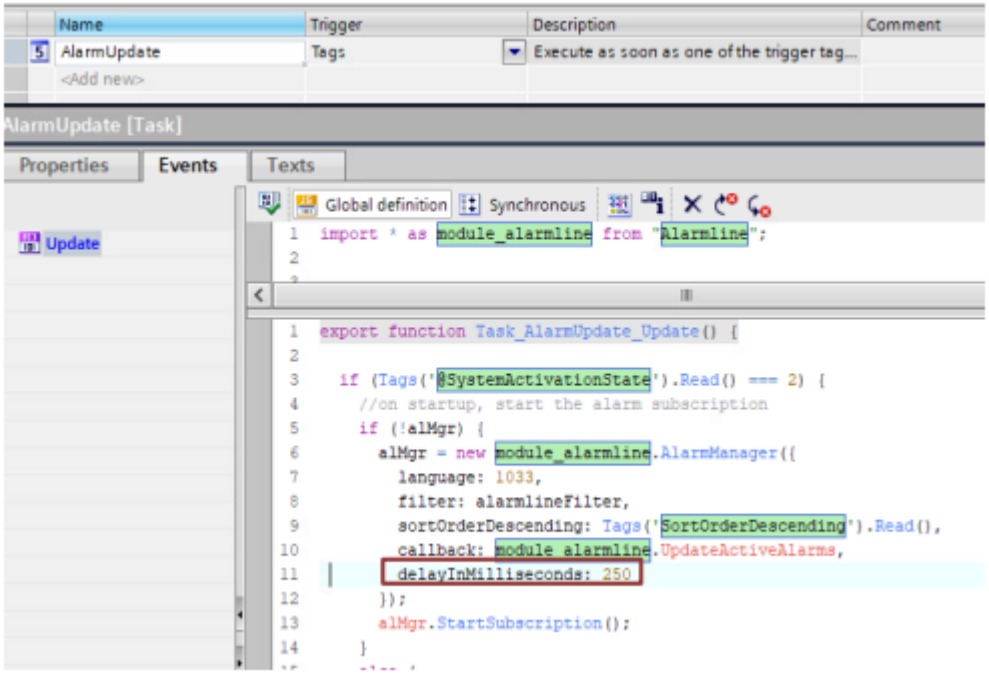


Рисунок 2.41 - Кастомна Alarm Line: затримка в мілісекундах

3 ОРГАНІЗАЦІЯ І РОБОТА З ГАРФІЧНИМИ ОБЄКТАМИ ТА SVG

3.1 Типи і формати графічних об’єктів

Окрім усієї текстової інформації, HMI зазвичай також містить зображення. Це можуть бути маленькі іконки, більші зображення або анімована графіка / динамічні SVG. Залежно від випадку використання існують рекомендації щодо застосування форматів зображень:

1. JPG (Joint Photographic Experts Group) рекомендується для UCP, оскільки цей формат зменшує розмір великих файлів. При цьому відбувається втрата якості через стиснення, але файли декодуються швидше. Якщо потрібне зображення високої роздільної здатності, JPG також може використовуватися у високій якості. JPG не означає автоматично низьку якість.

2. PNG (Portable Network Graphic) — растровий графічний формат зі стисненням без втрат. Рекомендується, коли потрібна прозорість або точна піксельна точність.

3. SVG (Scalable Vector Graphics) використовується для відображення двовимірної графіки, діаграм та ілюстрацій на вебсайтах. Як векторний формат, SVG-зображення можна масштабувати до різних розмірів без втрати роздільної здатності. Тому цей формат рекомендований, коли потрібне якісне масштабування або використання динамічних SVG. В інших випадках краще конвертувати файл у розмір, який буде використовуватись, у PNG або, якщо прозорість не потрібна, у JPG, при цьому слід обирати найбільший розмір для багаторазового використання.

ПРИМІТКА: Формати можна поділити на дві групи:

1. Растрові графічні формати, що базуються на заздалегідь відрендерених бітмапах і тому менш вимогливі до ресурсів системи, рекомендовані для runtime (JPG, PNG);

2. Описові графічні формати, що використовують код для визначення вигляду графіки. Вони рендеряться лише під час runtime, що робить їх легко адаптивними по розміру.

Формат графіки

Рекомендований порядок вибору формату графіки:

1. JPEG.

2. PNG.

3. SVG.

Якщо є додаткові вимоги до графіки, SVG може бути найбільш підходящим форматом, але вибирайте його обережно!

Випадок використання: Візуалізація патернів і складених об’єктів

Опис

Деякі об’єкти або патерни можуть бути створені з кількох окремих об’єктів. Крім того, ці комбіновані об’єкти можуть багаторазово використовуватись на екрані.

Рішення

Не створюйте патерн за допомогою окремих елементів, а об’єднуйте їх в одне зображення. Це зменшує кількість об’єктів на екрані. Навіть якщо об’єкти базові, наприклад, прямокутники, об’єднання в один об’єкт знижує навантаження на рендеринг.

Якщо потрібна динамізація, створюйте кастомний динамічний SVG (рис. 3.1). Це також зменшує кількість контейнерів об’єктів на екрані та допомагає відповідати системним обмеженням.

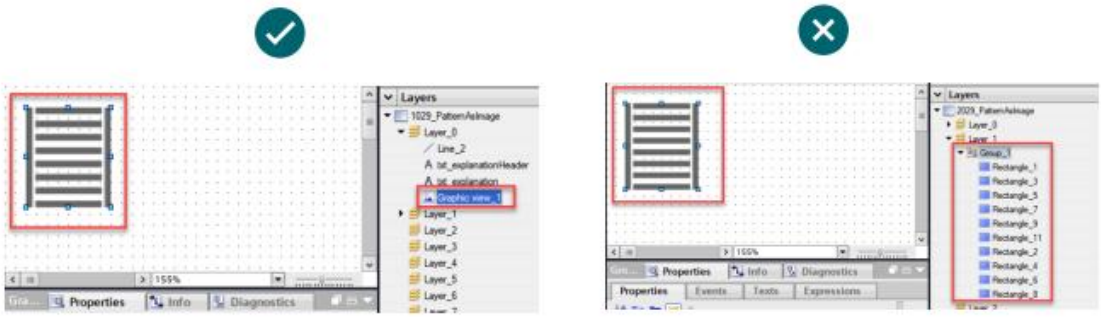


Рисунок 3.1 - Приклад патерну у форматі SVG

3.2 Робота з комбінованими графічними об’єктами

Опис

Деякі ілюстрації з динамічною візуалізацією можна створити або шляхом композиції кількох динамізованих об’єктів екрану, або безпосередньо за допомогою динамічного SVG.

Рішення

Як вже описано у попередньому випадку, щоб мінімізувати кількість елементів на екрані, складені об’єкти слід об’єднувати в один контейнер об’єктів. У IndustryGraphicLibrary є доступні динамічні SVG для широкого спектру стандартних компонентів. Перш ніж створювати власний компонент із кількох об’єктів екрану, перевірте, чи вже існує рішення в графічній бібліотеці. Через інтерфейс деякі властивості можна налаштовувати і змінювати динамічно під час runtime [11].

Більше інформації про використання динамічних віджетів можна знайти в мануалі **Configuring screens (RT Unified)** (рис. 3.2-3.4).

Рисунок 3.2 - Industry Graphic Library

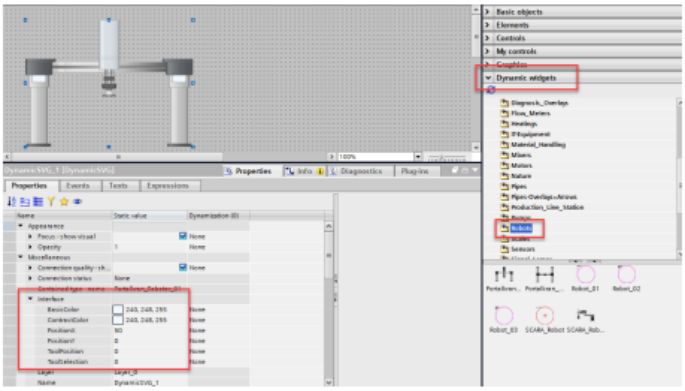


Рисунок 3.3 - Динамічний SVG робота з бібліотеки графіки Siemens

Конвеєри є особливою формою динамічних віджетів.

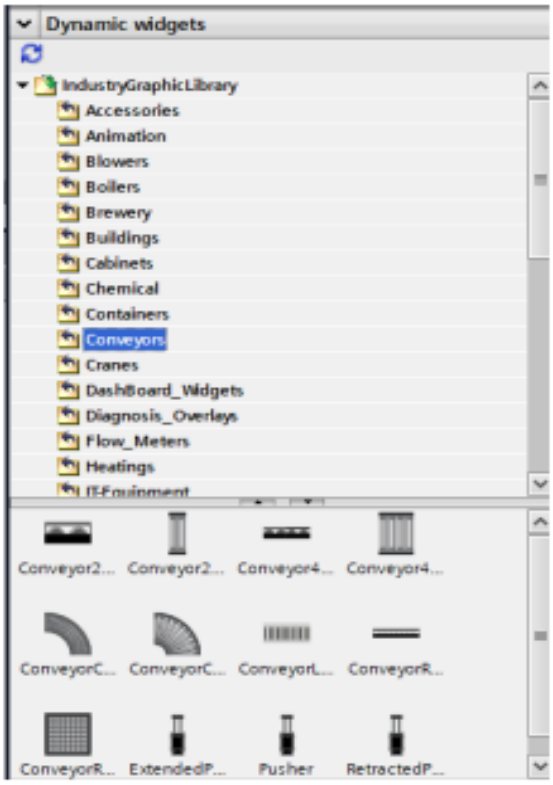


Рисунок 3.4 - Приклади динамічних віджетів — конвеєри

Проста динамізація тегом

Динамізація тегом показана на наступній ілюстрації. Вона може мати різні типи конфігурацій. Можна вибрати тип «None», щоб безпосередньо передавати значення тегу до властивості. Тип «Range» дозволяє задати умови для декількох діапазонів значень тегу і зв’язати їх з певним значенням властивості. Тип «Multiple bits» дає можливість змінювати властивість залежно від різних окремих позицій бітів тегу динамізації. І нарешті, тип «Single bit» обмежується одним бітом тегу динамізації для визначення умов властивості [9].

Кожного разу, коли змінюється значення тегу, значення властивості адаптується.

У цьому контексті важливо згадати про подію «Change». За допомогою цієї події можна виконати додатковий скрипт, якщо властивість змінюється під час runtime, наприклад, якщо значення тегу динамізації змінилося. Детальніше про це можна побачити на рисунках 3.5-3.6.

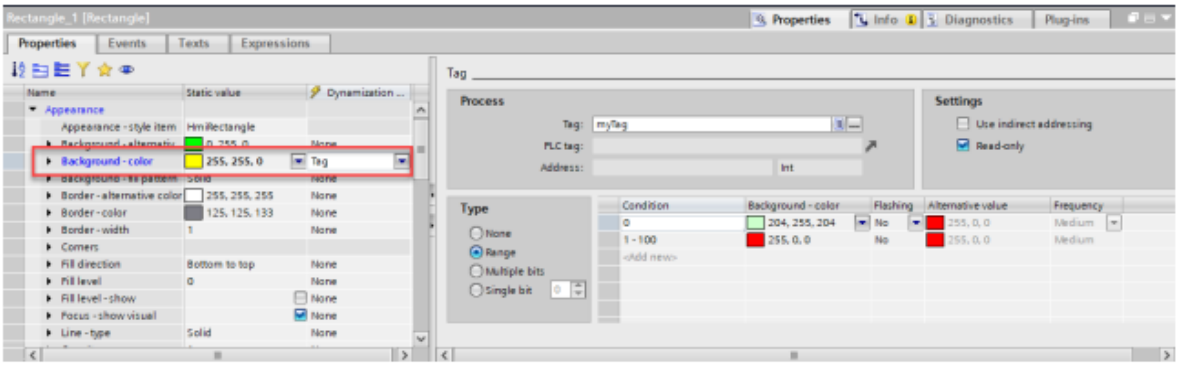


Рисунок 3.5 - Динамізація тегом

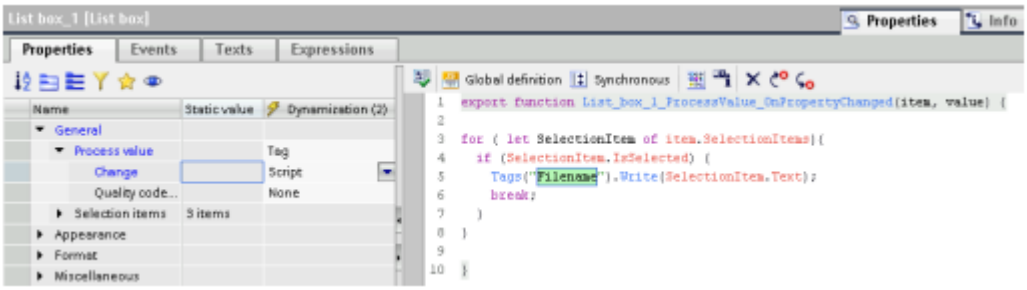


Рисунок 3.6 - Динамізація тегом – подія зміни

Скриптова динамізація

Другий тип динамізації — це скриптова динамізація. Повернене значення цього скрипта визначає значення властивості після виконання скрипта. Для запуску скрипта необхідно вказати тригер (рис. 3.7).

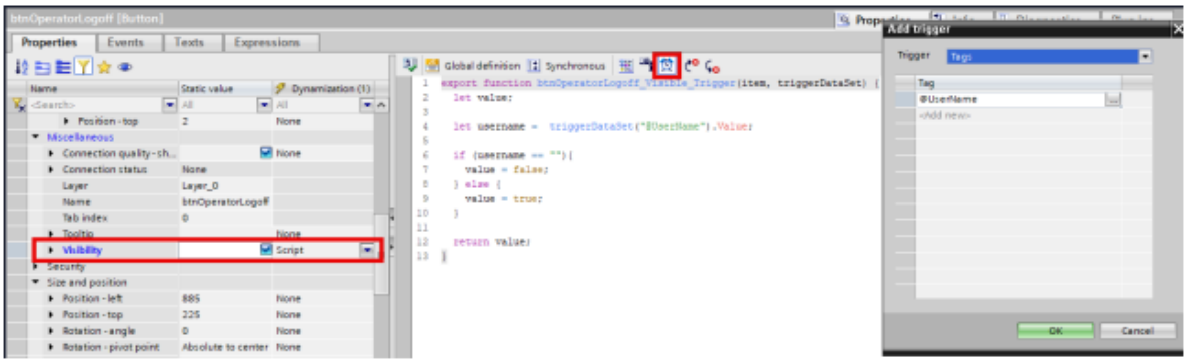


Рисунок 3.7 - Скриптова динамізація

Вирази (Expressions) є ще одним способом динамізації властивостей елемента екрана. Для їхньої конфігурації доступна додаткова вкладка (рис. 3.8-3.9). За допомогою виразу можна задати умову, що базується на кількох тегах, та змінювати кілька властивостей відповідно до цих умов.

- вираз оцінюється, коли змінюється значення одного з тегів;
- властивості змінюються одразу після зміни результату виразу;
- якщо жодна з умов не повертає TRUE, використовується значення за замовчуванням;
- якщо задано кілька умов, застосовується перша в списку, що повертає TRUE;
- вирази, які не можуть бути оцінені, пропускаються, наприклад, через синтаксичну помилку або недоступний тег.

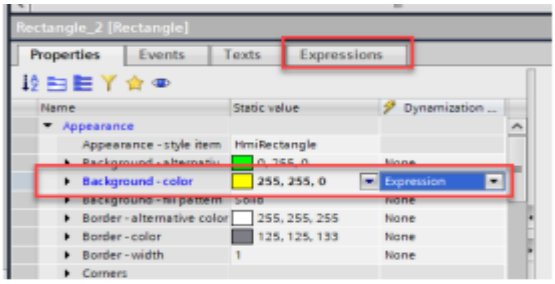


Рисунок 3.8 - Вираз як засіб динамізації

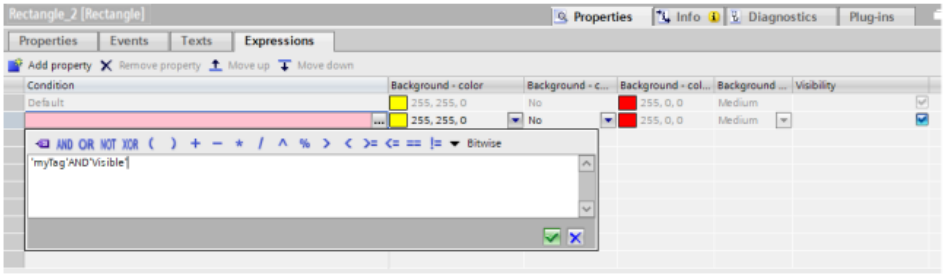


Рисунок 3.9 - Вираз як засіб динамізації

Слід звернути увагу на порядок, у якому визначені вирази (Expressions). Вони виконуються згори донизу, і як тільки якась умова стає істинною (true), інші вже не виконуються (рис. 3.10).

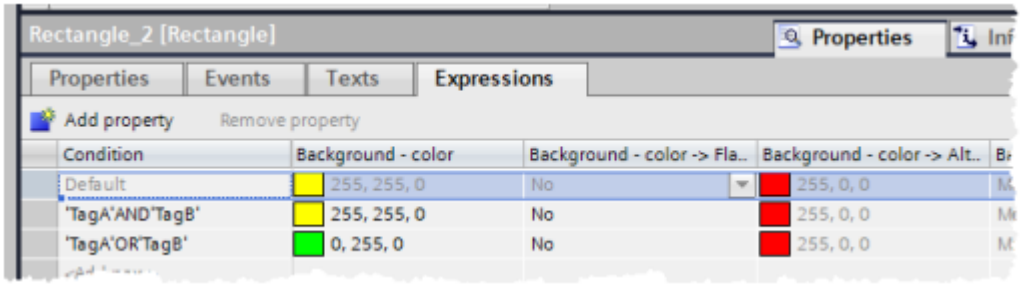


Рисунок 3.10 - Порядок виконання виразів

Варіант використання: Динамізація властивості залежно від окремих бітів

Визначення

Деякі варіанти використання (Use Cases) потребують динамізації властивості залежно від окремих бітів у слові. Раніше це потрібно було реалізовувати за допомогою скриптів, як показано в наступному прикладі. Залежно від бітів повертається різне значення кольору (рис. 3.11).

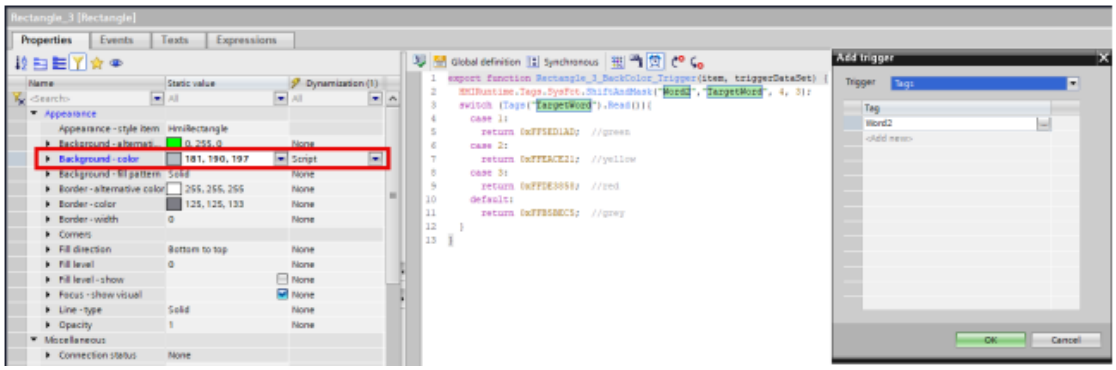


Рисунок 3.11 - Скрипт для динамізації властивості

Рішення

Застосовуйте побітові операції у виразах (Expressions), щоб замінити скриптову динамізацію властивості (рис. 3.12-3.13). Оператор SR8() використовується для зсуву бітів праворуч, а оператор AND8() — для маскування [13].

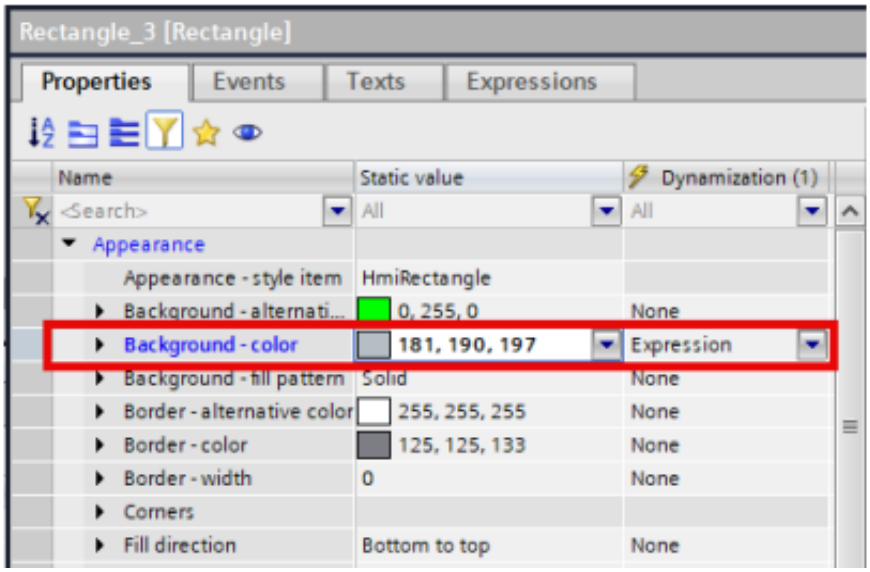


Рисунок 3.12 - Властивість виразу (Expression Property)

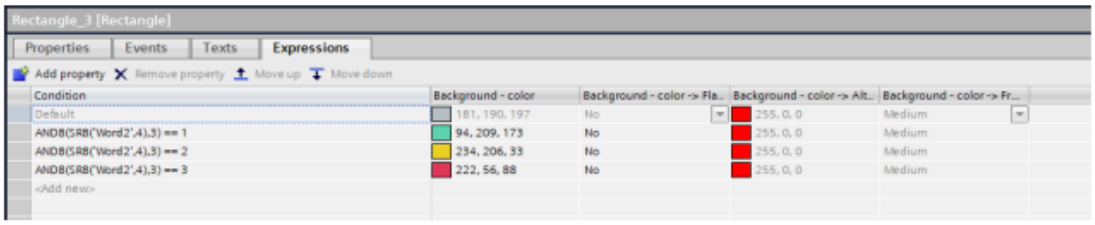


Рисунок 3.13 - Вираз із використанням операторів Shift і Mask

Формула (Formula)

Формулу (Formula) можна застосовувати до будь-якої числової властивості (наприклад, колір фону, видимість, розмір — висота), яка динамізується за допомогою тега. Властивість можна легко динамізувати без використання жодного скрипта. За допомогою формули можна задати умову, що базується на кількох тегах, і вона буде виконана, якщо зміниться значення хоча б одного з використаних тегів. Формула має ті самі параметри перетворення, що й вирази (Expressions).

Крім того, система надає деякі функції перетворення у формулах для конвертації значень або тегів у різні групи одиниць. Використання функцій перетворення є продуктивнішим, ніж застосування скриптів [9].

3.3 Організація тегів у проектах

Варіант використання: Перетворення значень тегів для візуалізації

Визначення

Деякі варіанти використання (Use Cases) потребують перетворення тега зі значенням (наприклад, множення на 10) або конвертації тега в іншу одиницю виміру. Раніше це потрібно було реалізовувати за допомогою скриптів, як показано в наступних прикладах.

Варіант використання 1:

Тег множиться на 10, і залежно від отриманого значення відповідно встановлюється колір фону прямокутника (рис. 3.14).

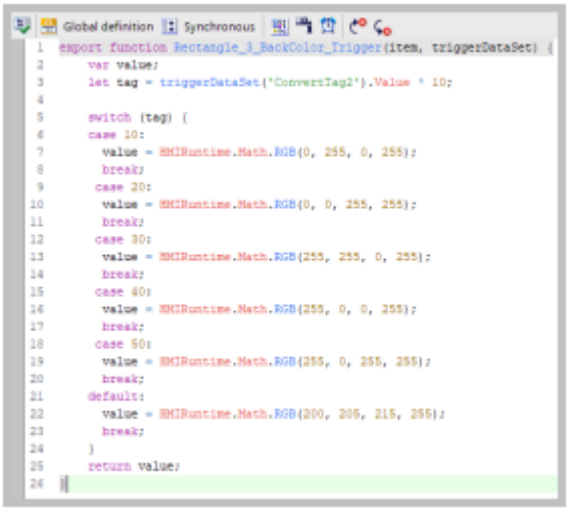


Рисунок 3.14 - Скрипт для зміни кольору фону залежно від перетвореного тега

Варіант використання 2: ValueConverter

Тег з одиницею виміру *miles* буде перетворено в одиницю km (рис. 3.15).

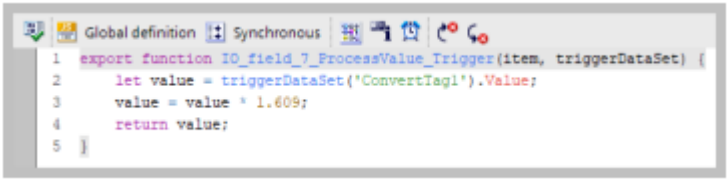


Рисунок 3.15 - Скрипт для конвертації з miles у km

Рішення

Використовуйте **формулу (Formula)** та надані функції перетворення (conversion functions) для динамізації властивостей, щоб замінити скриптову динамізацію (рис. 3.16-3.17).

Варіант використання 1: Вирази у Формулі (Formula Expressions)

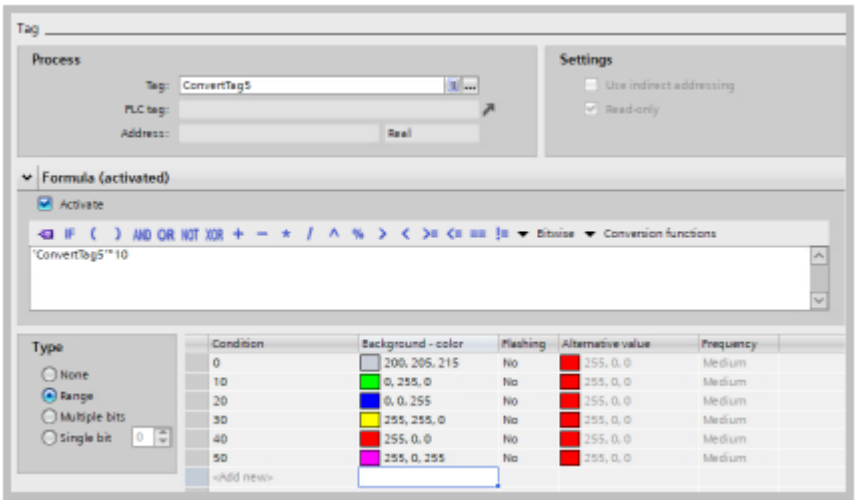


Рисунок 3.16 - Приклад використання формули з діапазоном (range)

Варіант використання 2: ValueConverter

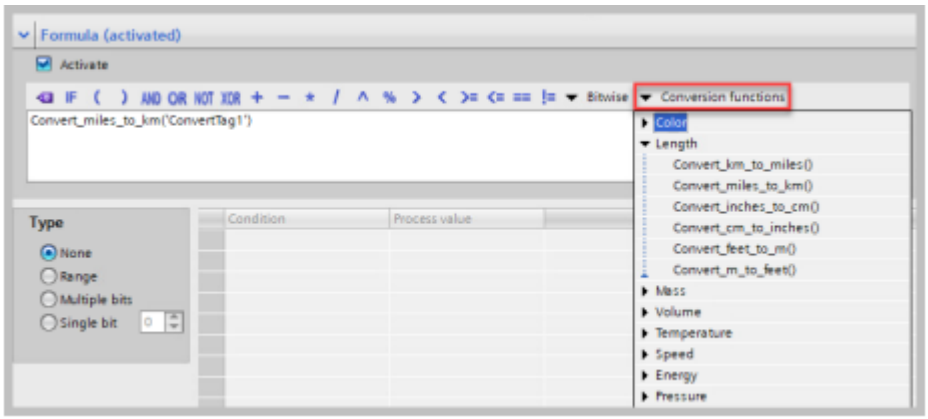


Рисунок 3.17 - Формульне використання з функціями конвертації

Додаткові опції

Залежно від властивості, також можливо використання списку ресурсів або миготіння як засобу динамізації (рис. 3.18-3.19).

Опція **flashing** (миготіння) доступна для властивостей кольору. Список ресурсів — для властивостей тексту.

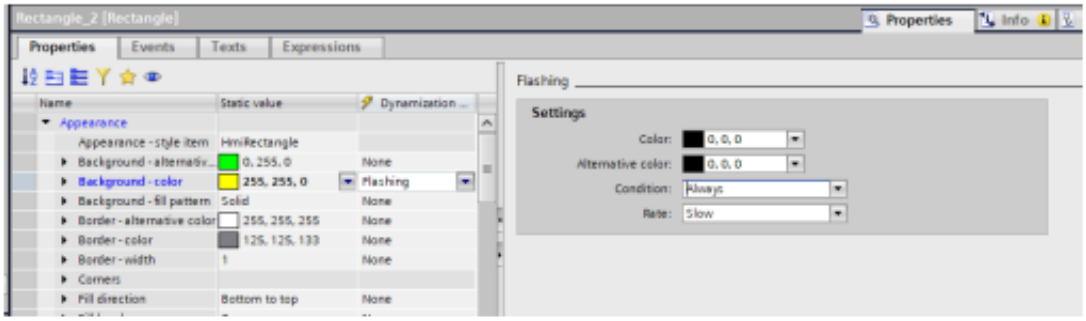


Рисунок 3.18 - Динамізація миготінням для властивостей кольору

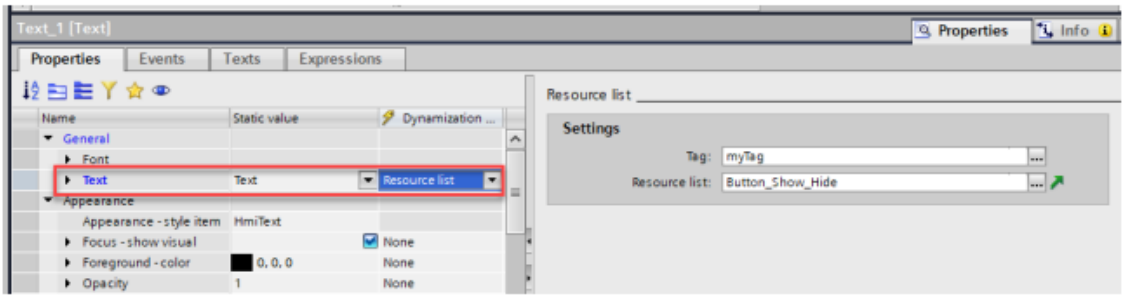


Рисунок 3.19 - Динамізація списком ресурсів для властивостей тексту

Випадок використання: Комбінація статичного тексту з динамічними параметрами для текстових властивостей

Використання **Formatted Text** (форматованого тексту) — це ще одна опція для включення динамічних параметрів (тегів або текстових списків) у властивість тексту, наприклад, для кнопки.

Formatted Text дозволяє поєднувати статичний текст із динамічними значеннями, отриманими з тегів або текстових списків, у властивості **Text** статичного об’єкта на екрані.

Formatted Text також може використовуватися з **faceplates**.

Використання текстових списків у полі параметра для **faceplates** неможливе.

Це спрощує динамізацію властивості **Text**, оскільки сценарії (скрипти) можна не використовувати для цього випадку. Також це дозволяє поєднати в одному об’єкті динамічний вивід значення і статичний текстовий вивід, наприклад, значення тегу з відповідною одиницею виміру (рис. 3.20).

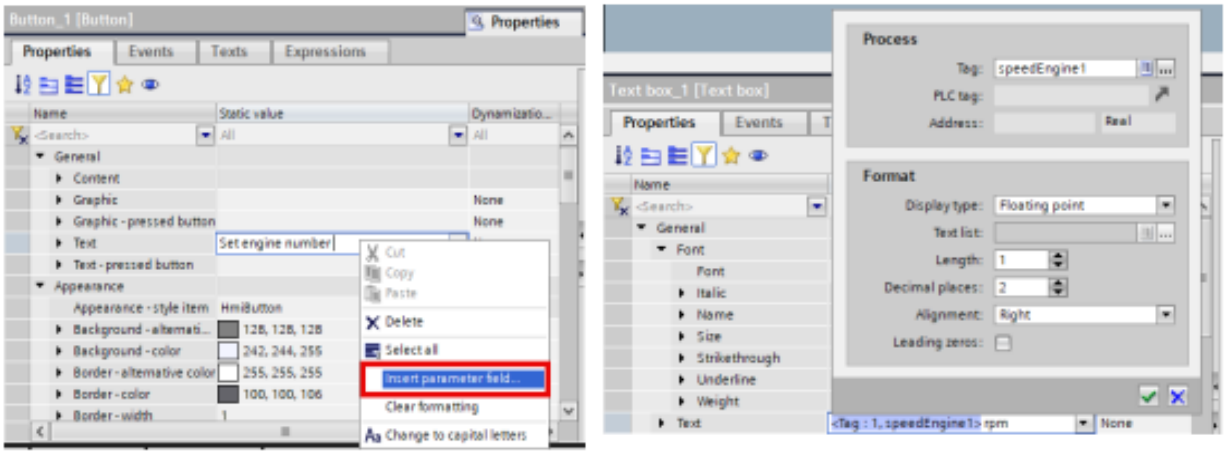


Рисунок 3.20 - Динамічний текст кнопки з використанням Formatted Text

Dynamizations

Використовуйте, якщо можливо, динамізацію через теги (**tag dynamization**).

Коли потрібна складніша логіка — використовуйте **Formula** або **Expressions**.

Намагайтеся уникати динамізації через скрипти!

Скрипти рекомендуються **лише** у випадках, коли їх використання дозволяє уникнути створення кількох об’єктів на екрані.

Інформація про різні типи тригерів (циклічний, теговий, тривожний, подійний тощо) детально задокументована у SIMATIC WinCC Unified – Tips and Tricks for Scripting (Java Script) [6].

4 РОЗРОБКА МЕТОДИЧНОГО ПРИКЛАДУ НА БАЗІ ЛАБОРАТОРНОГО ПРАКТИКУМУ

4.1 Створення і параметрування PLC S7-1500

Тепер створимо приклад проєкту на базі отриманої інформації. Робота починається з створення проєкту в середовищі Step7 V20 [11] (рис. 4.1).

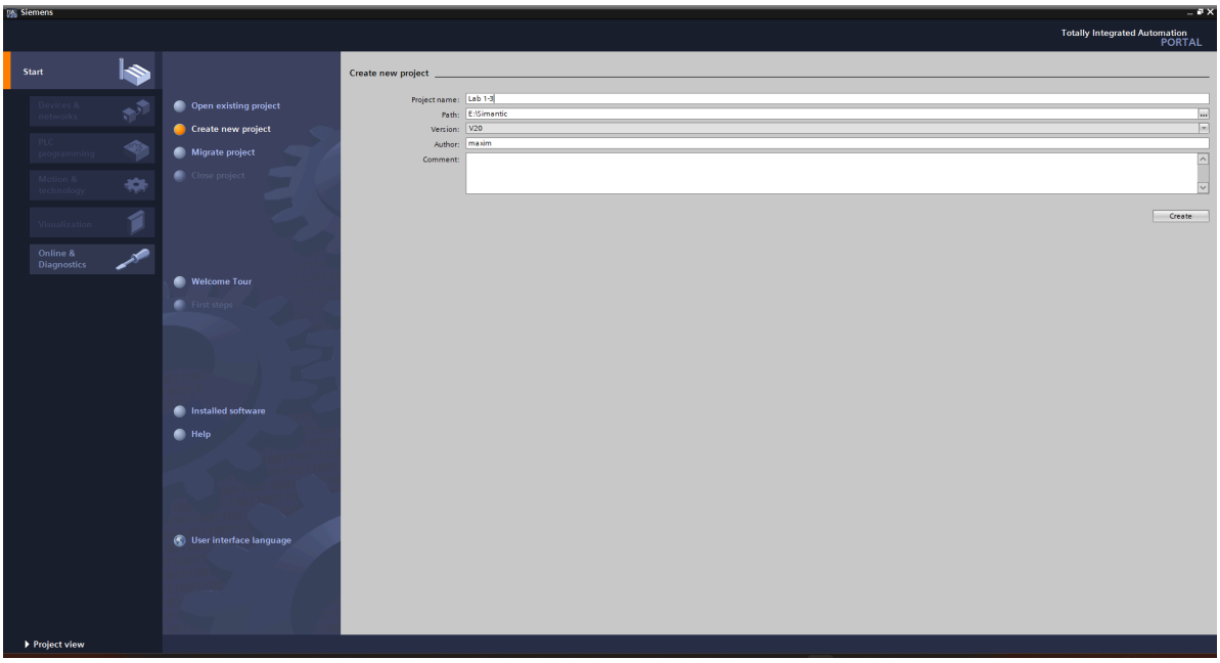


Рисунок 4.1 - Процедура створення нового проекту в програмному модулі STEP 7 Professional V20

Для початку роботи необхідно обрати відповідний PLC, в даному випадку це буде CPU 1511-1 PN (рис. 4.2). PLC мають різні версії, та різний набір додаткових компонентів, які у них можна вкласти. Необхідно враховувати ці фактори при виборі контролера.

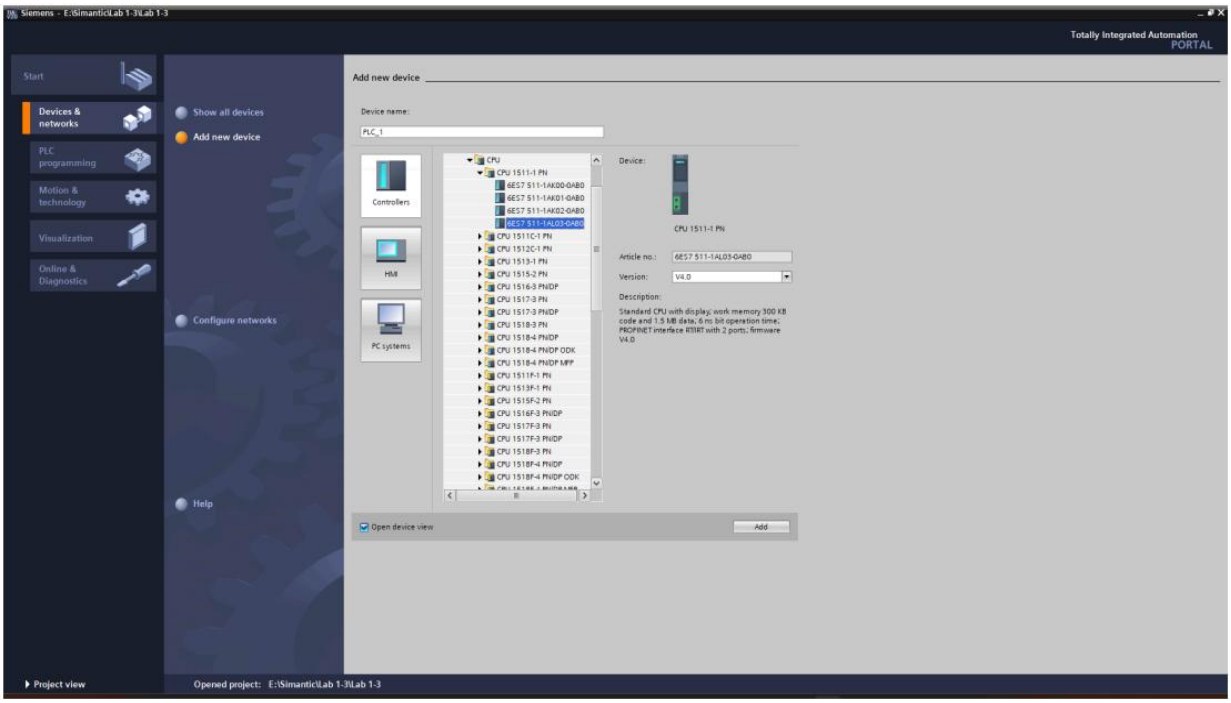


Рисунок 4.2 – Процедура вибору PLC Simatic S7-1500 (CPU 1511-1 PN Version

Після вибору відповідного PLC Simatic S7-1500 – необхідно виконати команду «Add» і виконати параметрування розділу «PLC security settings» і перейти в проектний інтерфейс (Project view) (рис. 4.3)- динамізації видимості вмісту [13].

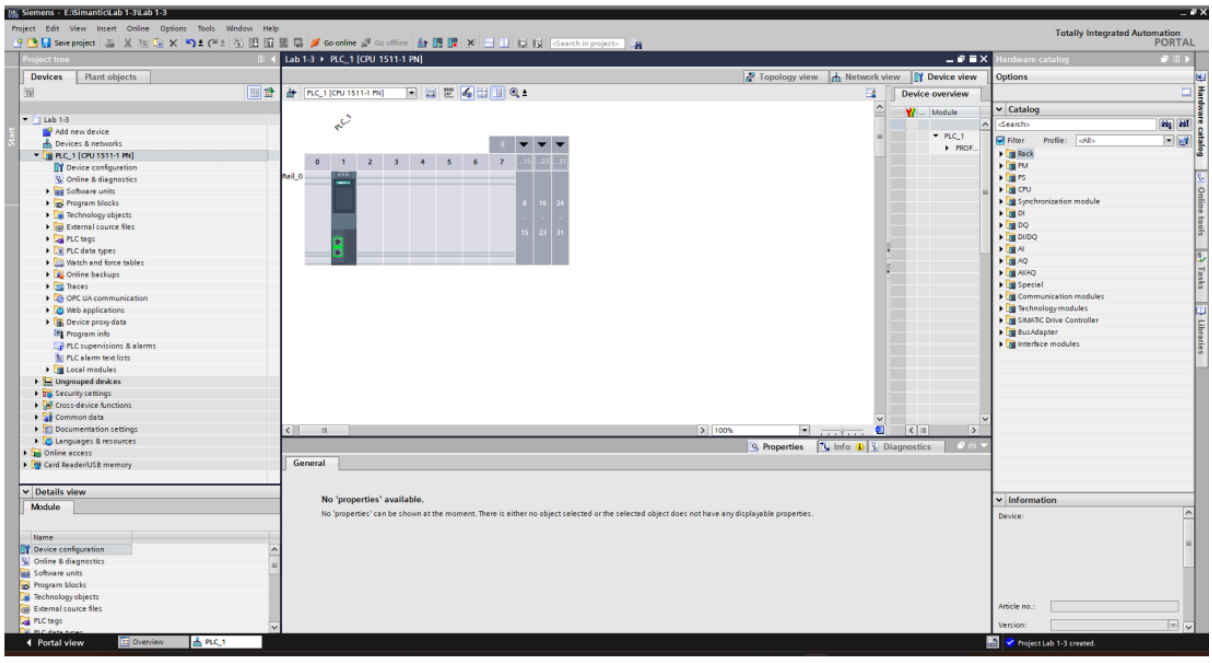


Рисунок 4.3 – Проєктний інтерфейс STEP 7 Professional V20

Далі необхідно провести процедури виконання конфігурації апаратних засобів PLC Simatic S7-1500. Конфігурація апаратних засобів полягає у встановленні необхідних апаратних модулів з (Hardware catalog), відповідно до реальної конфігурації системи (програмна копія чи програмне відображення). В цілях демонстрації оберемо довільні модулі (рис. 4.4).

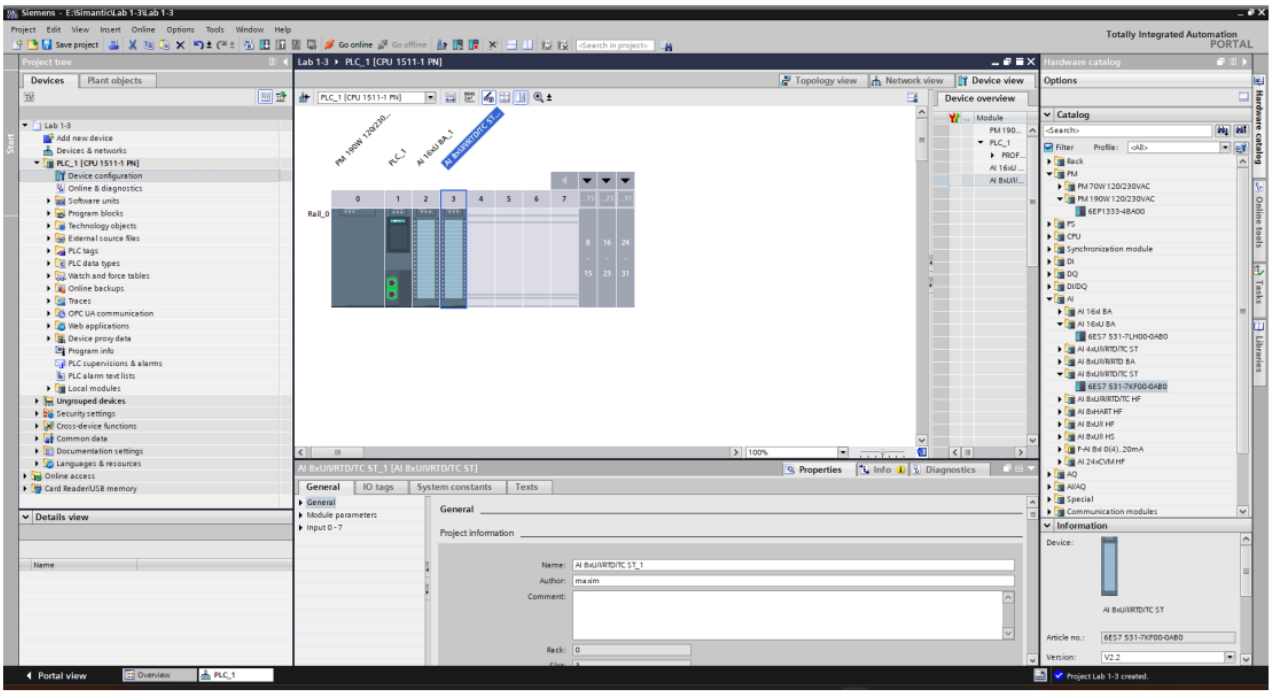


Рисунок 4.4 - Конфігурація апаратних засобів PLC Simatic S7-1500

Для початку роботи, достатньо встановити наступні параметри PLC Simatic S7-1500 через команду «Properties»: - системну пам’ять (System memory bits); - таймерний байт (Clock memory bits) (рис. 4.5).

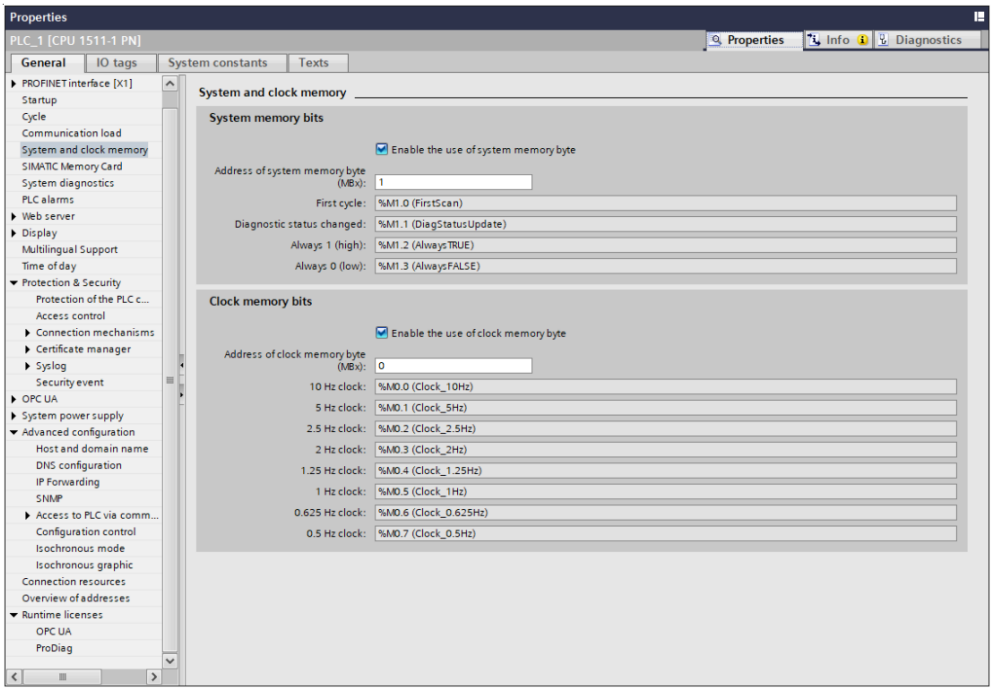


Рисунок 4.5 - Процес параметрування PLC Simatic S7-1500 шляхом активації (System memory bits і Clock memory bits)

Далі потрібно відтворити процедури налагодження комунікаційного середовища проекту для комунікації між програматором-персональним комп'ютером (PG/PC) і симулятором PLCSIM V20 [13]. Без цього кроку ми не зможемо в подальшому провести симуляцію (рис. 4.6).

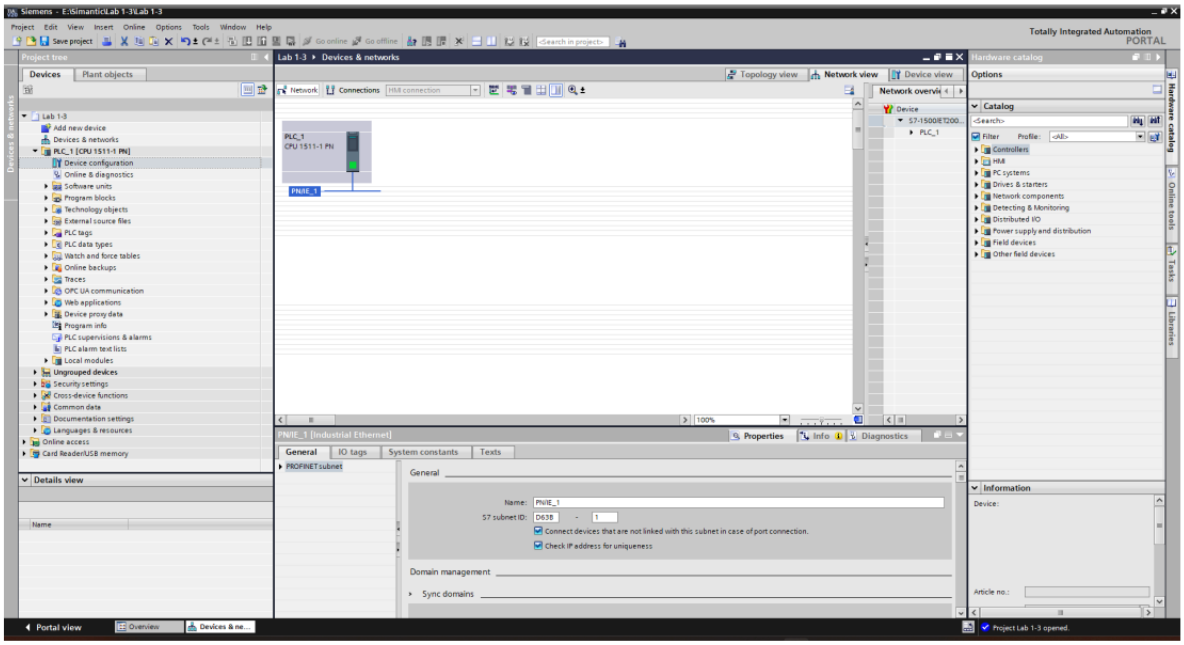


Рисунок 4.6 - Результат додавання нової підмережі PROFINET/Industrial Ethernet (PN/IE_1) в редакторі мереж «Network view»

Для продовження процесу налаштування, потрібно провести з’єднання з PLCSIM V20 через функцію «Connection to interface/subnet».

Після завантаження симуляції буде відкрита можливість виконання функції «Go online» (рис. 4.7).

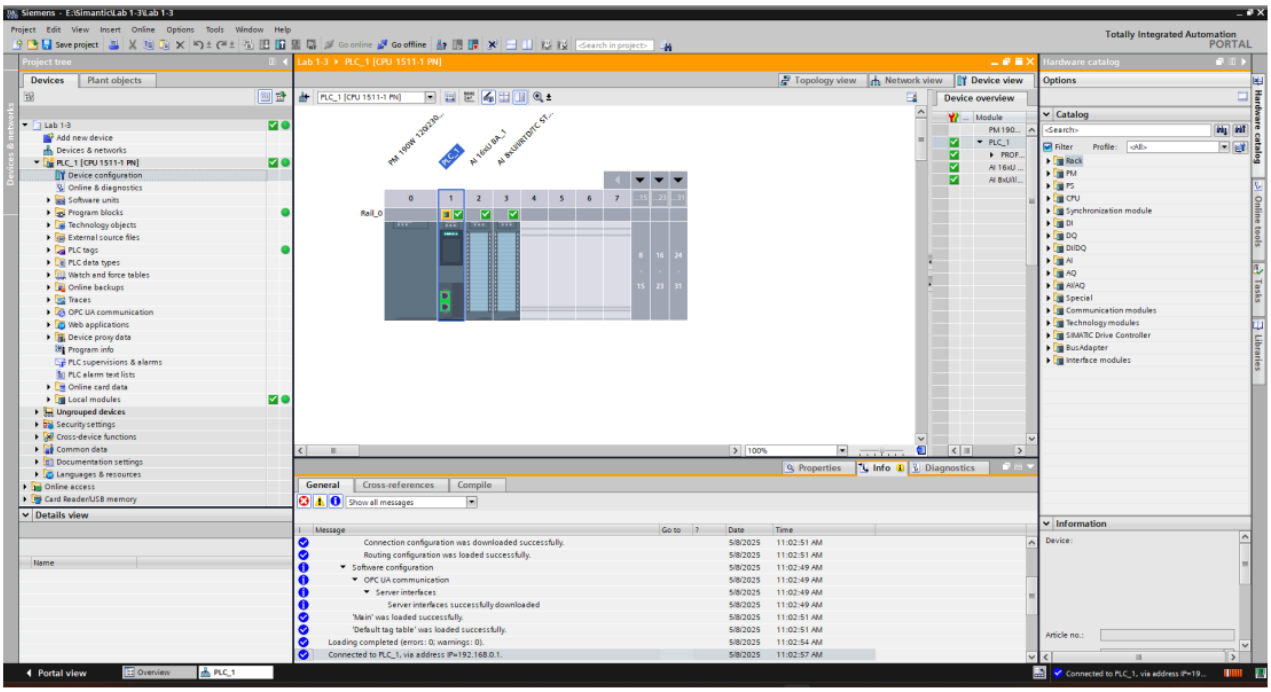


Рисунок 4.7 - Результат виконання команди «Go online» в меню команд після завантаження проекту в симулятор PLCSIM V20

Останній крок налаштування - створенням тестової програми на мові FBD (Functional Block Diagram). Для створенням тестової програми на мові FBD (Functional Block Diagram) необхідно вибрати мову програмування FBD перед відкриттям організаційного блоку (Organization Block) OB1 в дереві проекту [12].

В рамках цього організаційного блоку складемо базову програмну інструкцію обчислення («CALCTULATE»), яку ми будемо використовувати для генерації числових значень. На основі цих значень і буде виконуватись симуляція (рис. 4.8).

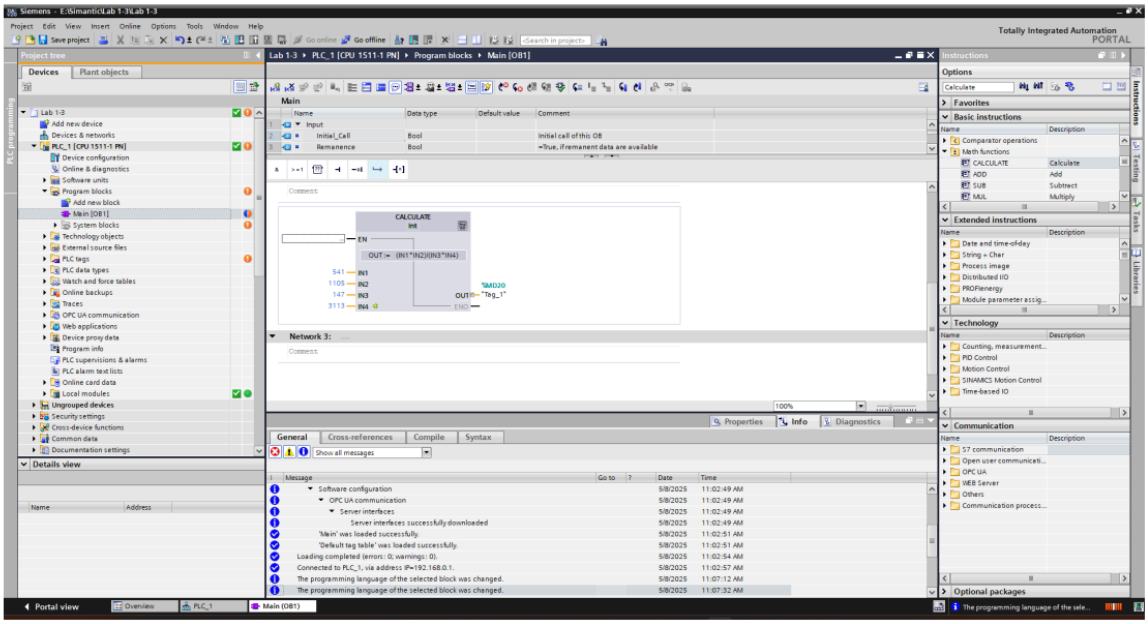


Рисунок 4.8 - Програмна інструкція «CALCULATE» з папки (Math function)

Отже, PLC 1511-1 PN був повністю налаштований.

4.1 Інтеграція HMI на базі WinCC Unified в проєкт

Після створення та параметрування основного програмованого контролера PLC, необхідно додати до нього людинно-машинний інтерфейс. Для реалізації цього потрібно додати компоненти люди-машинного інтерфейсу через команду «Add» (рис. 4.9).

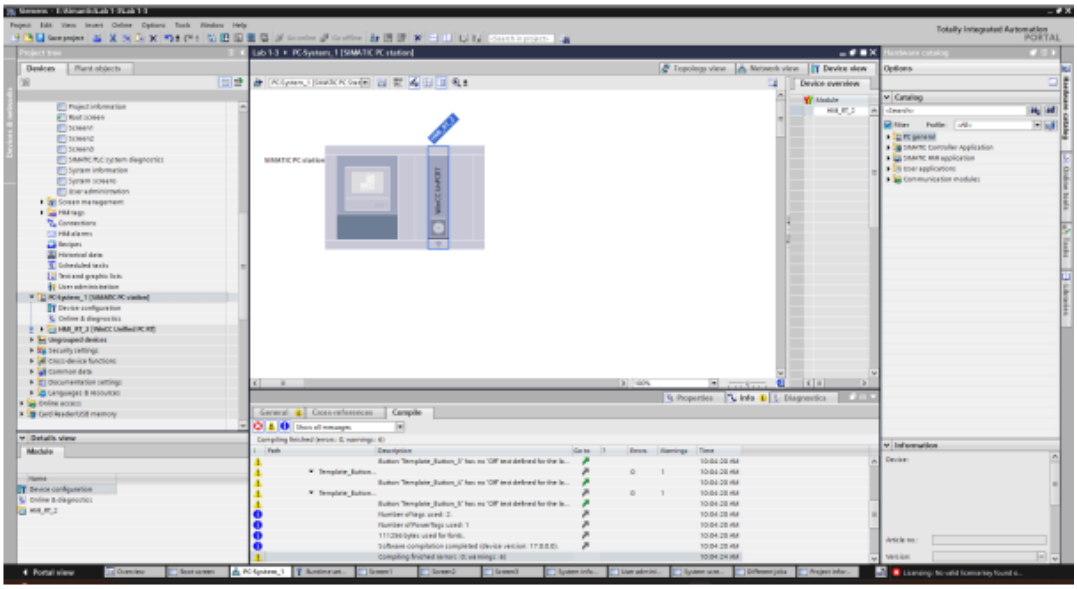


Рисунок 4.9 - Результат інтеграції в проект компонентів людино-машинного інтерфейсу (PC systems, PC station, SIMATIC PC-station, WinCC RT Unified)

Інтегровану станцію після включення у проєкт можна параметрувати, задавши станції ім’я, та приєднавши необхідний комунікаційний модуль. Додавання мережі потребує налагодження комунікаційного середовища у відповідному розділі дерева проєкту «Connections» [14].

Після налагодження комунікацій, створимо мнемосхему – віртуальний екран, який і буде відображати наш людино-машинний інтерфейс (рис. 4.10).

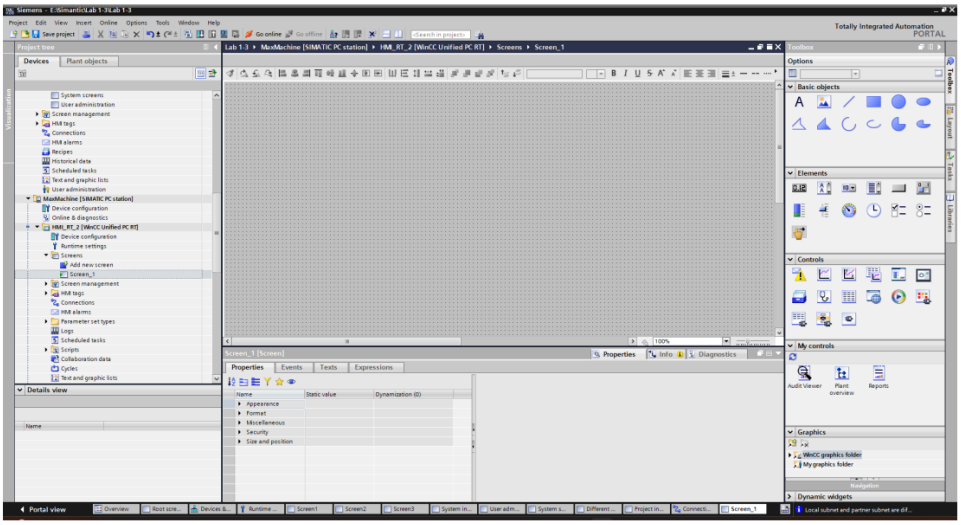


Рисунок 4.10 - Процедура і результат створення мнемосхеми (Screen_1) через команду в дереві проекту «Add new screen»

Реалізуємо на мнемосхемі два об’єкти – «поле вводу/виводу», та «шкала». Ці об’єкти будуть використовувати параметри організаційного блоку OB1, який ми вже налаштували (рис. 4.11).

Параметрування об’єктів може здійснюватися різними способами динамізації.

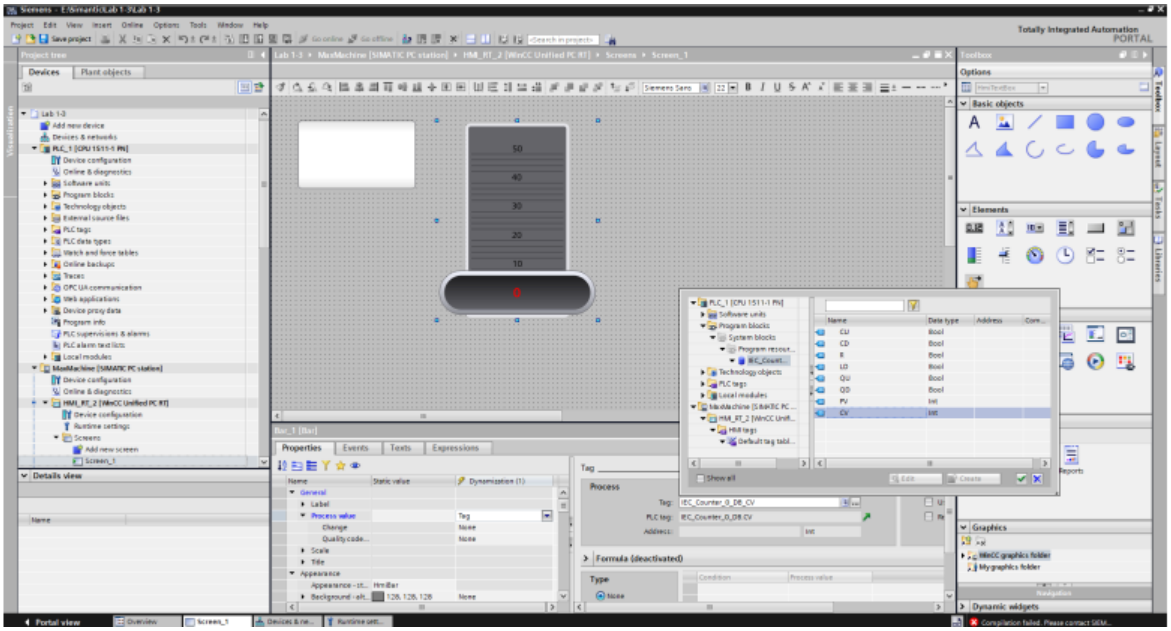


Рисунок 4.11 – Створення і параметрування об’єктів «поле вводу/виводу - I/O Field» і «шкала - Bar»

Об’єкти потребують додаткової параметризації. В її рамках ми можемо налаштувати цикли опитування тегів в розділі «HMI tags», я також налаштувати відповідні параметри динамізації – колір і інтервали числових значень (рис. 4.12).

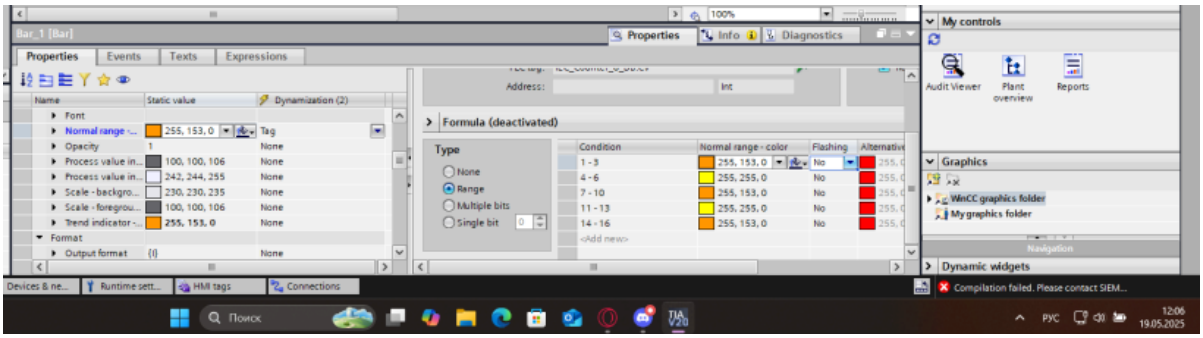


Рисунок 4.12 – Додаткове параметрування об’єктів «поле вводу/виводу - I/O Field» і «шкала - Bar»

Після виконання усіх базових налаштувань – завантажимо проєкт у «Simatic Runtime Manager». Цей крок необхідний для зв'язку з симулятором PLCSIM Advanced V7.0, який і відтворить для нас необхідну WEB-сторінку (рис. 4.13).

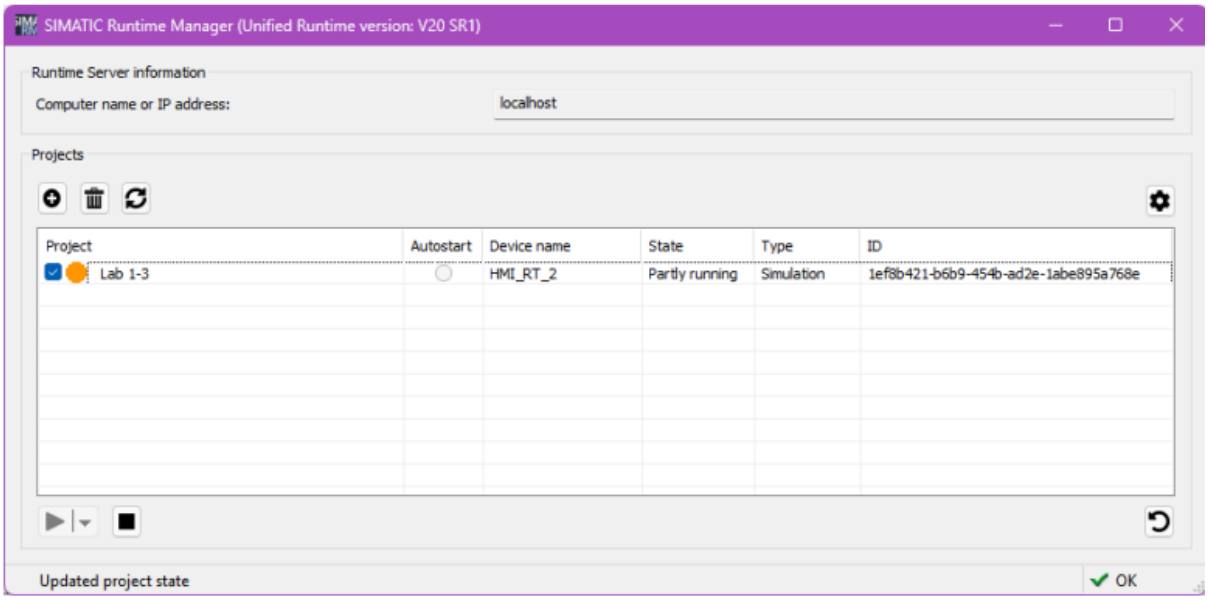


Рисунок 4.13 – Результат виконання процедури завантаження проекту в «Simatic Runtime Manager» через команду «Start simulation» в режимі «Simulation»

Активація симулятора PLCSIM Advanced V7.0 дозволяє отримати доступ до відповідних WEB-сторінок, та провести симуляцію нашої системи (рис. 4.14).

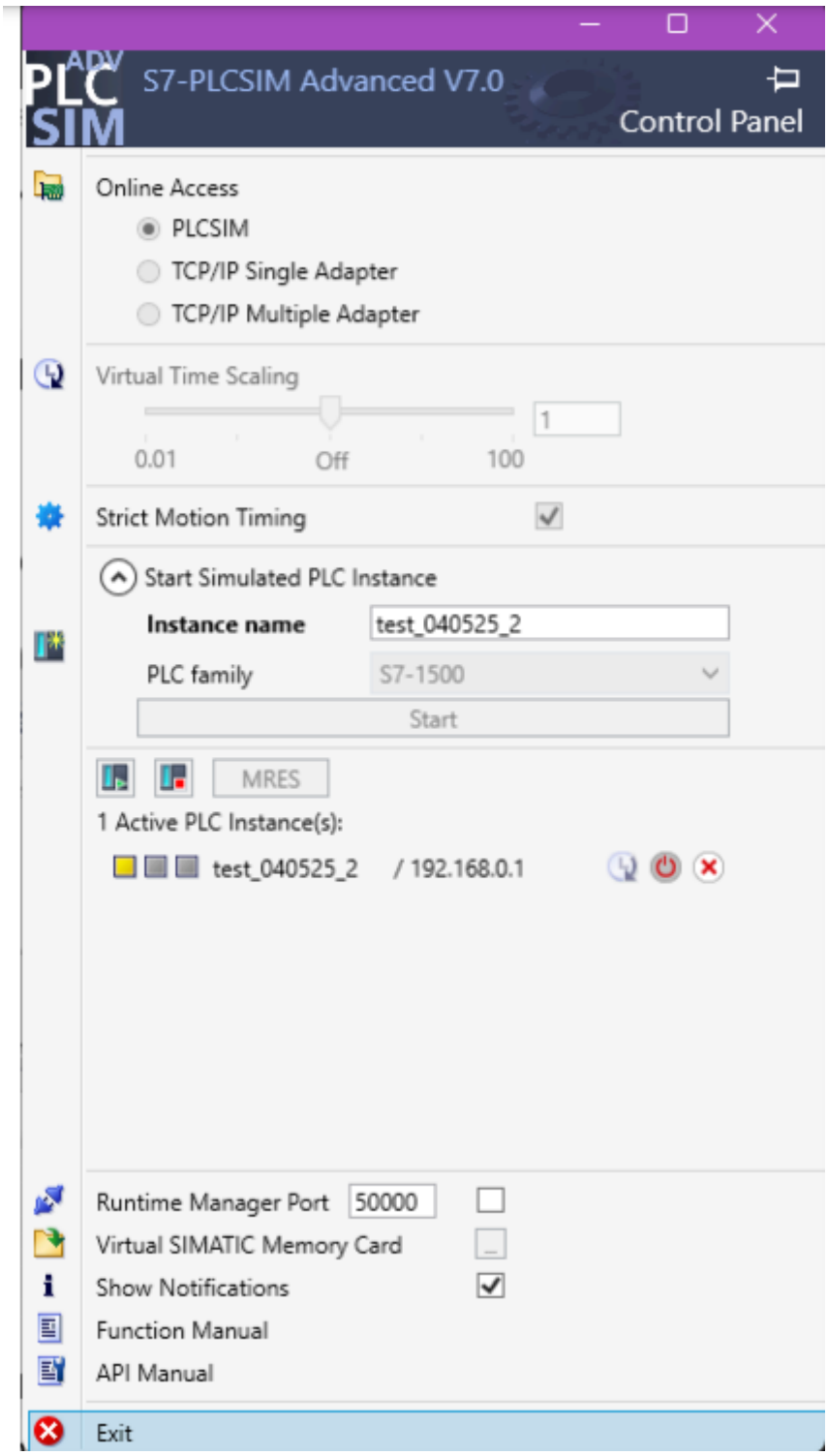


Рисунок 4.14 - Процедура активації сисмулятора PLCSIM Advanced V7.0 в режимі «PLCSIM»

3 Симуляція проєкту засобами PLCSIM Advanced

Далі буде наведено результат тестування клієнт-серверної топології автоматизованої системи управління на базі WEB-сервера WinCC Unified V20 PC RT і клієнта на базі WEB-браузера «Google Chrome» (рис. 4.15-4.16).

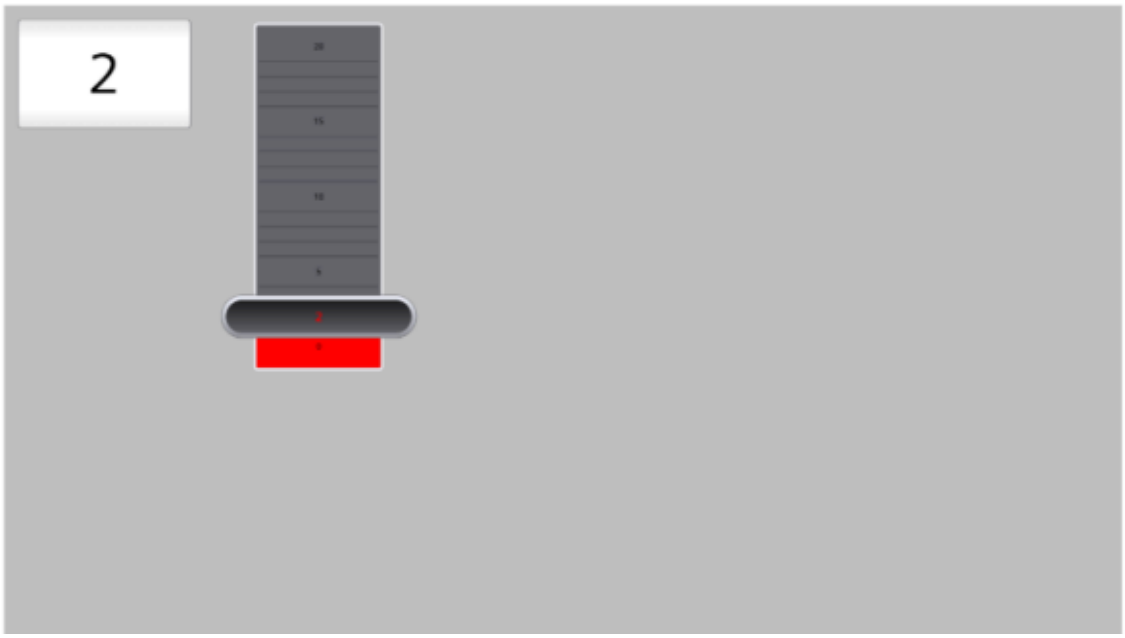


Рисунок 4.15 – Процедура симуляції графічних обє’ктів «поле вводу/виводу - I/O Field»

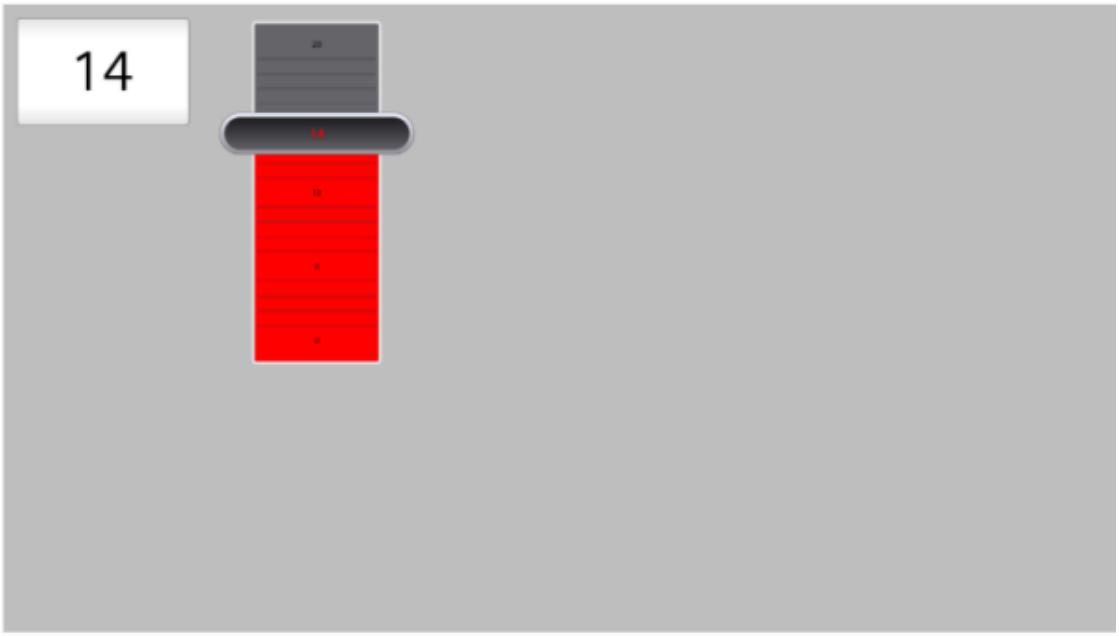


Рисунок 4.16 – Процедура симуляції графічних обє’ктів «шкала - Bar»

Для тестування стандартної (діагностичної) сторінки WEB-сервера PLC Simatic S7-1500 необхідно ввести поточну IP-адресу «PLC_1» проекту у WEB-браузері «Google Chrome» (рис. 4.17-4.21).

Рисунок 4.17 - Тестування стандартної (діагностичної) сторінки WEB-сервера PLC Simatic S7-1500

Рисунок 4.18 Тестування стандартної (діагностичної) сторінки WEB-сервера PLC Simatic S7-1500, розділ «Start page»

Рисунок 4.19 - Тестування стандартної (діагностичної) сторінки WEB-сервера PLC Simatic S7-1500, розділ «Diagnostics» → «Runtime information»

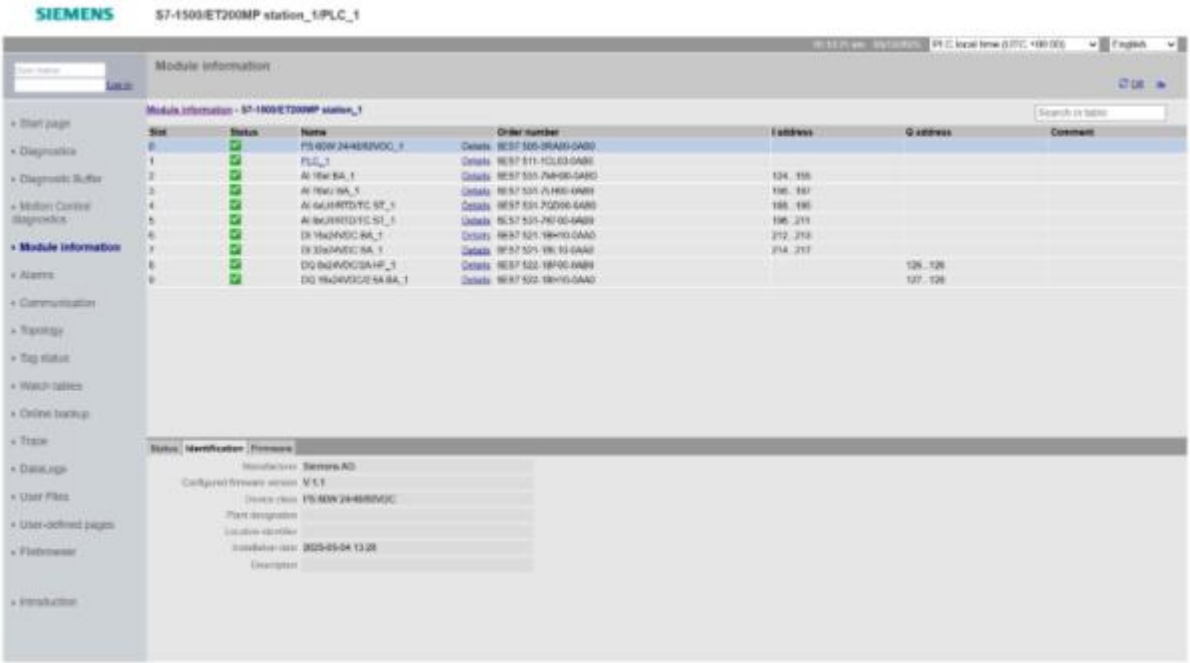


Рисунок 4.20 - Тестування стандартної (діагностичної) сторінки WEB-сервера PLC Simatic S7-1500, розділ «Module Information»

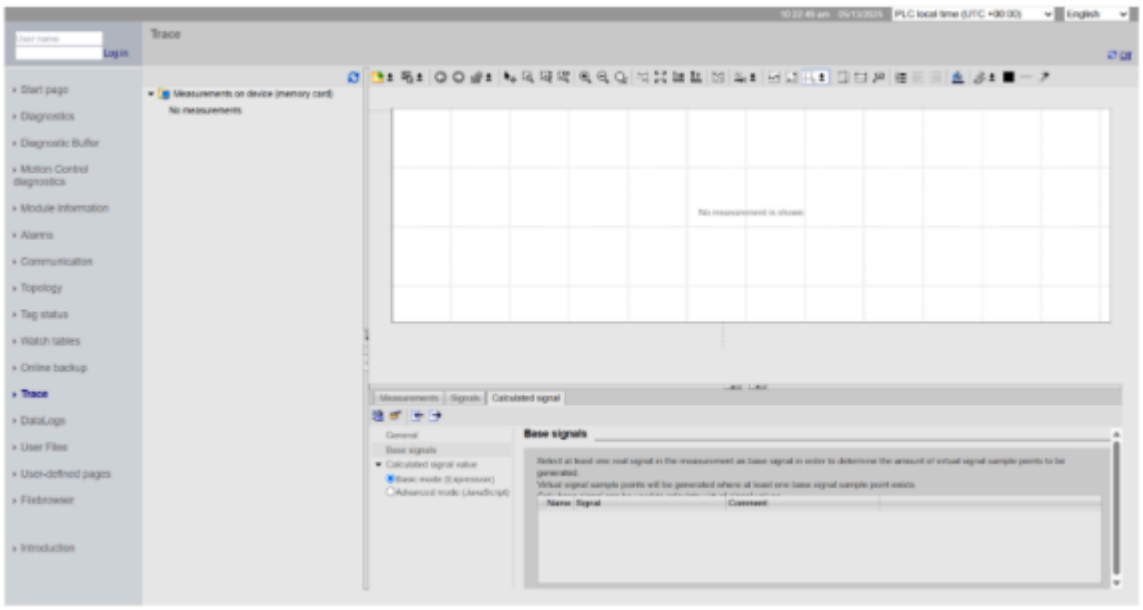


Рисунок 4.21 - Тестування стандартної (діагностичної) сторінки WEB-сервера PLC Simatic S7-1500, розділ «Trace»

Отже, у результаті виконання проєкту ми змогли провести симуляцію двох елементів у рамках WEB-середовища, та отримати повну діагностичну інформацію про наш проєкт.

Будь-який проєкт у рамках WinCC Unified та TIA PORTAL V20 повторює кроки, які були показані у цьому прикладі. Відмінності є лише у параметризації окремих елементів системи, яка залежить від індивідуальних цілей кожного окремого користувача [14].

ВИСНОВКИ

У ході виконаної роботи було досліджено та розроблено підходи до проєктування WEB-орієнтованих систем управління з використанням сучасного апаратно-програмного забезпечення на базі PLC та SCADA. Акцент було зроблено на практичному засвоєнні здобувачами ключових етапів розробки таких систем, що є надзвичайно важливими у контексті підготовки фахівців з автоматизації та комп'ютерно-інтегрованих технологій.

WEB-технології, як частина індустріальних систем керування, відкривають нові можливості для реалізації віддаленого моніторингу та керування, забезпечують гнучкість, масштабованість та інтеграцію з іншими галузями – від енергетики до «розумного дому». Розроблений лабораторний практикум дозволяє відтворити ключові компоненти систем симульованого середовища, що сприяє кращому розумінню структури, функціональності та принципів взаємодії між апаратними та програмними елементами.

Таким чином, впровадження подібних навчальних рішень у освітній процес дозволяє підвищити якість підготовки майбутніх інженерів, забезпечити набуття актуальних практичних навичок та сформувати компетентності, необхідні для роботи з сучасними WEB-орієнтованими системами управління в умовах цифрової трансформації промисловості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Siemens Industry Online Support [Електронний ресурс]. – Режим доступу: https://support.industry.siemens.com

2. Siemens AG. Engineering guideline for WinCC Unified (V3.0) [Електронний ресурс]. – Режим доступу: https://support.industry.siemens.com/cs/ww/en/view/109827603

3. Siemens AG. Beginners Guide: Quick entry and conversion with WinCC Unified [Електронний ресурс]. – Режим доступу: https://support.industry.siemens.com/cs/ww/en/view/109810917

4. Siemens AG. HMI design with the HMI Template Suite [Електронний ресурс]. – Режим доступу: https://support.industry.siemens.com/cs/ww/en/view/91174767

5. Siemens AG. SIMATIC WinCC Unified Tutorial Center (Videos) [Електронний ресурс]. – Режим доступу: https://support.industry.siemens.com/cs/ww/en/view/109782433

6. Siemens AG. SIMATIC WinCC Unified – Tips and Tricks for Scripting (JavaScript) [Електронний ресурс]. – Режим доступу: https://support.industry.siemens.com/cs/ww/en/view/109758536

7. TIA Portal Applications. TIA-Add-In - ShowScripts [Електронний ресурс]. – Режим доступу: https://github.com/tia-portal-applications/TIA-Add-In-ShowScripts

8. Siemens AG. Developing WinCC Unified JavaScript code and checking style guide with Visual Studio Code [Електронний ресурс]. – Режим доступу: https://support.industry.siemens.com/cs/ww/en/view/109801600

9. SIMATIC HMI WinCC Unified V20 system limits [Електронний ресурс]. – Режим доступу: https://docs.tia.siemens.cloud/r/en-us/v20/wincc-unified-rt-unified/performance-features-rt-unified/general-technicaldata-rt-unified

10. Заміховський Л. М., Заміховська О.Л., Зікратий С.В., Николайчук М.Я. Організаця підготовки бакалаврів. Бакалаврська кваліфікаційна робота. Вимоги до змісту та оформлення: методичні вказівки /Л. М. Заміховський, О.Л. Заміховська, С.В. Зікратий, М.Я. Николайчук М.Я – Івано-Франківськ: ІФНТУНГ, 2019. – 73 с. (МВ 02070855-14027-2019).

11. М.Я.Николайчук, Н.М.Орловський, О.Л.Заміховська. Розробка та імітаційне моделювання режимів роботи системи керування технологічним об'єктом на базі апаратно-програмних засобів Simatic S7. Збірник наукових праць Проблемно-наукової міжгалузевої конференції «Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, моделювання та управління (ISCM -2022)» м. Надвірна (14-15 липня 2022 року). – С. 181-185.

12. Николайчук М.Я Методика організації лабораторного практикуму «Технології і засоби проєктування систем управління» [Текст] / М.Я. Николайчук // Збірник тез доповідей Всеукр. наук.- практ. конф. молодих учених і студентів «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ОСВІТІ, ТЕХНІЦІ ТА ПРОМИСЛОВОСТІ» / - 13 жовтня 2022, м. Івано-Франківськ. – С. 239-240.

13. Zamikhovskyi, L., Nykolaychuk, M., & Levytskyi, I. (2024). Organizing the automated system of dispatch control over pump units at water pumping stations. Eastern-European Journal of Enterprise Technologies, 5(2 (131), 61–75. https://doi.org/10.15587/1729-4061.2024.313531.

14. Николайчук М.Я., Левицький І.Т. Методичні вказівки до виконання курсового проекту. Проектування WEB-орієнтованих систем управління. Івано-Франківськ: ІФНТУНГ, 2025. 51 с. (МВ 02070855-20645-2025).

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи – «Апаратно-програмне забезпечення лабораторного практикуму «Проектування WEB-орієнтованих систем управління»»

Обсяг пояснювальної записки в аркушах – 78

Перелік креслень графічної частини:

КРБ.СІ - 13.00.00.001 Е2 Параметрування комунікації PLC S7-1500. Схема функціональна (аркушів – 1).

КРБ.СІ - 13.00.00.002 Е1 Процес налаштування нових елементів екрану. Схема структурна (аркушів – 1).

КРБ.СІ - 13.00.00.003 Е2 Апаратна конфігурація тестового проєкту. Схема функціональна (аркушів – 1).

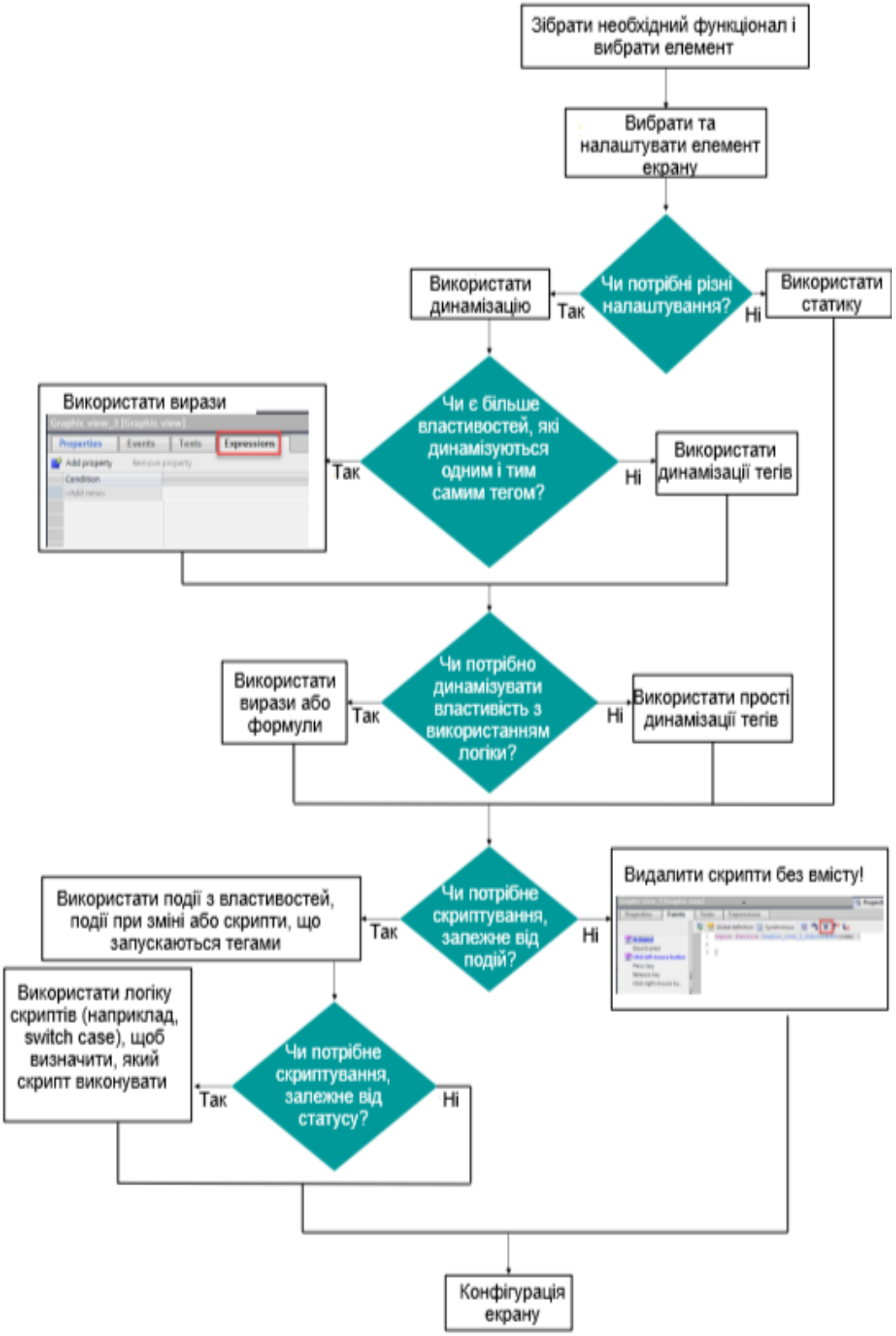
КРБ.СІ - 13.00.00.004 Е2 Приклад роботи тестового проєкту. Схема функціональна (аркушів – 1).

Дата закінчення бакалаврської роботи «15» червня 2025 р.

Студент _____ Настащук М. Ю.

КРБ.СІ-13.00.001 Е2

					КРБ.СІ-13.00.001 Е2		
Змн.	Лист	№ докум.	Підпис	Дата			
Розроб.		Настащук М. Ю.			*Параметрування комунікації PLC S7-1500. Схема функціональна*	Літ.	Маса / Масштаб
Перевір.		Николайчук М. Я.					
Т. Контр.						Арк. 1	Аркушів 1
Реценз.							
Н. Контр.		Возний А. В.				ІФНТУНГ СІ-21-1	
Затверд.		Заміховський Л. М.					



КРБ.СІ-13.00.002 Е1

Змн.	Лист	№ докум.	Підпис	Дата		Літ.	Маса	Масштаб
Розроб.		Настащук М. Ю.			*Процес налаштування нових елементів екрану. Схема структурна*			
Перевір.		Николайчук М. Я.				Арк. 1	Аркушів 1	
Т. Контр.								
Реценз.								
Н. Контр.		Возний А. В.				ІФНТУНГ СІ-21-1		
Затверд.		Заміховський Л. М.						

					КРБ.СІ-13.00.003 Е2		
Змн.	Лист	№ докум.	Підпис	Дата			
Розроб.		Настащук М. Ю.			*Апаратна конфігурація тестового проєкту. Схема функціональна*	Літ.	Маса / Масштаб
Перевір.		Николайчук М. Я.					
Т. Контр.							
Реценз.						Арк. 1	Аркушів 1
Н. Контр.		Возний А. В.			ІФНТУНГ СІ-21-1		
Затверд.		Заміховський Л. М.					

КРБ.СІ-13.00.004 Е2

					КРБ.СІ-13.00.004 Е2			
Змн.	Лист	№ докум.	Підпис	Дата		Літ.	Маса	Масштаб
Розроб.		Настащук М. Ю.			*Приклад роботи тестового проєкту. Схема функціональна*			
Перевір.		Николайчук М. Я.				Арк. 1	Аркушів 1	
Т. Контр.								
Реценз.								
Н. Контр.		Возний А. В.				ІФНТУНГ СІ-21-1		
Затверд.		Заміховський Л. М.						