Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеню вищої освіти – магістр
за освітньо-професійною програмою
«Кіберфізичні системи в промисловості, бізнесі та транспорті»

зі спеціальності
174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка

тема роботи:

«*Розробка веб-інтерфейсу системи візуалізації комплексу моделювання вантажно-розвантажувальних операцій на базі SCARA-роботу*»

Виконав ст. гр. АКІТР-23-1м _____ Леонов Д. В.

Керівник _____ Харламенко В. Ю.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2024

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Магістр*

Спеціальність: *174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка технології та робототехніка*

ЗАТВЕРДЖУЮ

Зав. кафедри: *к.т.н. Рубан С.А.*

_____

«___»_____2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра

студентові групи АКІТР-23-1м *Леонову Данилу Володимировичу*

1. Тема кваліфікаційної роботи: *«Розробка веб-інтерфейсу системи візуалізації комплексу моделювання вантажно-розвантажувальних операцій на базі SCARA-роботу»*

затверджено наказом по університету № *590с* від *04.07.2024 р.*

2. Термін здачі кваліфікаційної роботи: *01.12.2024 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 94с., додатки, презентація у Microsoft PowerPoint (11 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-3 к.т.н Харламенко В. Ю.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*05.07.24*
2	*Розділ 1*	*25.07.24*
3	*Розділ 2*	*18.10.24*
4	*Розділ 3*	*19.11.24*
5	*Висновки*	*19.11.24*
6	*Оформлення кваліфікаційної роботи*	*25.11.24*
7	*Підготовка презентації та графічного матеріалу*	*28.11.24*
8	*Підготовка доповіді до захисту*	*07.12.24*

6. Дата видачі завдання: *29.06.2024р.*

Керівник _____ /Харламенко В. Ю./

7. Запевнення: *Я, Леонов Данило Володимирович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Здобувач _____ / Леонов Д. В./

АНОТАЦІЯ

Кваліфікаційна робота на здобуття ступеню вищої освіти магістр за освітньо-професійною програмою «Кіберфізичні системи в промисловості, бізнесі та транспорті» зі спеціальності 174 – Автоматизація, комп'ютерно – інтегровані технології та робототехніка– Криворізький національний університет, Кривий Ріг, 2024.р.

В даній магістерській роботі було розглянуто розробку та реалізацію архітектури веб-інтерфейсу з використанням технологій Internet of Things.

У першому розділі представлено комплексний аналіз сучасного стану автоматизації вантажно-розвантажувальних операцій, досліджено актуальні тенденції в логістиці та виробництві, розглянуто економічні та технологічні передумови автоматизації, проаналізовано вплив систем візуалізації на ефективність промислових процесів.

У другому розділі розкрито архітектуру системи автоматичного керування стенду, обґрунтовано вибір технологій для створення інформаційно-комунікаційної системи, висвітлено загальні принципи архітектури Industrial Internet of Things, представлено структуру системи та логіку її роботи.

Третій розділ присвячений програмній реалізації веб-інтерфейсу для візуалізації вантажно-розвантажувальних операцій, включаючи налаштування проєкту в TIA Portal, розробку коду для платформи Node-RED, створення веб-клієнта для візуалізації та підготовку інструкції користувача.

ВЕБ-ІНТЕРФЕЙС, СИСТЕМА ВІЗУАЛІЗАЦІЇ, OPC UA, MQTT, NODE-RED, IOT, TIA PORTAL, SCARA-РОБОТ, ПРОМИСЛОВИЙ ІНТЕРНЕТ РЕЧЕЙ

ANNOTATION

Graduation master`s work for obtaining an educational degree «Master» for the educational and professional program « Cyber-physical systems in industry, business and transport » in specialty 174 – «Automation, computer-integrated technologies, and robotics». – Kryvyi Rih National University, Kryvyi Rih, 2024.

This master's thesis considered the development and implementation of a web interface architecture using Internet of Things technologies.

The first section presents a comprehensive analysis of the current state of automation of loading and unloading operations, explores current trends in logistics and production, considers the economic and technological prerequisites for automation, and analyzes the impact of visualization systems on the efficiency of industrial processes.

The second section reveals the architecture of the automatic control system of the stand, justifies the choice of technologies for creating an information and communication system, highlights the general principles of the Industrial Internet of Things architecture, presents the system structure and the logic of its operation.

The third section is devoted to the software implementation of a web interface for visualizing loading and unloading operations, including project setup in TIA Portal, code development for the Node-RED platform, creation of a web client for visualization, and preparation of user manual.

WEB INTERFACE, VISUALIZATION SYSTEM, OPC UA, MQTT, NODE-RED, IOT, TIA PORTAL, SCARA ROBOT, INDUSTRIAL INTERNET OF THINGS

ЗМІСТ

ВСТУП

У промисловості дедалі більше уваги приділяється автоматизації та оптимізації виробничих і логістичних процесів. SCARA-роботи відіграють ключову роль у вдосконаленні вантажно-розвантажувальних операцій, забезпечуючи високу точність, продуктивність і безпеку. Їх застосування відкриває нові можливості для створення сучасних інтелектуальних систем керування.

Необхідність ефективного моніторингу та керування складними технологічними комплексами обумовлює постійний розвиток людино-машинних інтерфейсів. Веб-технології надають унікальні можливості для створення інтуїтивно зрозумілих та функціональних систем візуалізації, які забезпечують оператору повний контроль над технологічними процесами в режимі реального часу.

Метою магістерської роботи є розроблення веб-інтерфейсу для системи моделювання вантажно-розвантажувальних операцій на базі SCARA-роботу, який забезпечує ефективний моніторинг та керування технологічного процесу.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– Провести комплексний аналіз сучасного стану автоматизації вантажно-розвантажувальних операцій

– Дослідити існуючі системи та підходи до створення веб-інтерфейсів для промислових систем

– Розробити архітектуру інформаційно-комунікаційної системи

– Створити веб-інтерфейс для моніторингу та керування SCARA-роботом

– Забезпечити ефективну взаємодію між апаратними компонентами та веб-клієнтом

Об’єктом дослідження є процес моделювання та автоматизації вантажно-розвантажувальних операцій з використанням SCARA-роботу.

Предметом дослідження є розроблення веб-інтерфейсу для системи моніторингу та керування технологічним комплексом на базі SCARA-роботу.

Практичне значення отриманих результатів полягає у створенні уніфікованого рішення для моніторингу та керування робото-технічними комплексами, яке може бути адаптоване та впроваджене на різних виробничих та навчальних платформах. Розроблений веб-інтерфейс демонструє ефективний підхід до побудови інтерактивних систем керування, що підвищують продуктивність та зручність роботи операторів.

РОЗДІЛ 1

АНАЛІЗ СУЧАСНОГО СТАНУ АВТОМАТИЗАЦІЇ НАВАНТАЖУВАЛЬНО-РОЗВАНТАЖУВАЛЬНИХ ОПЕРАЦІЙ

1.1 Актуальність та тенденції розвитку систем візуалізації в промисловій автоматизації

У сучасному світі, де технологічний прогрес стрімко змінює ландшафт промисловості та логістики, системи візуалізації відіграють ключову роль у підвищенні ефективності та продуктивності виробничих процесів. Актуальність розвитку таких систем обумовлена зростаючою потребою в оптимізації операцій, зменшенні витрат та підвищенні конкурентоспроможності підприємств на глобальному ринку. Цей розділ присвячений аналізу сучасних тенденцій у логістиці та виробництві, а також економічних і технологічних передумов автоматизації, які стимулюють розвиток систем візуалізації в промисловій автоматизації.

1.1.1 Сучасні тенденції в логістиці та виробництві

Логістика та виробництво переживають період глибоких трансформацій, викликаних впровадженням інноваційних технологій та зміною глобальних економічних умов. Однією з ключових тенденцій є перехід до концепції «Індустрія 4.0», яка передбачає інтеграцію цифрових технологій у всі аспекти виробничого процесу. Ця концепція базується на використанні таких технологій, як інтернет речей (IoT), великі дані (Big Data), штучний інтелект (AI) та машинне навчання (ML), які разом створюють фундамент для розумних фабрик майбутнього [7].

У контексті логістики спостерігається тенденція до автоматизації складських операцій та оптимізації ланцюгів поставок. Впровадження автоматизованих систем зберігання та пошуку (AS/RS), автономних мобільних роботів (AMR) та систем управління складом (WMS) дозволяє значно підвищити ефективність операцій та

зменшити час виконання замовлень. Крім того, зростає попит на гнучкі та адаптивні логістичні рішення, здатні швидко реагувати на зміни ринкових умов та споживчого попиту [8].

Виробничий сектор також зазнає суттєвих змін. Спостерігається тенденція до персоналізації продукції та скорочення виробничих циклів, що вимагає від підприємств більшої гнучкості та швидкості реагування. Це стимулює впровадження адитивних технологій (3D-друк), роботизованих виробничих ліній та систем предиктивного технічного обслуговування. Важливим аспектом сучасного виробництва стає також екологічна стійкість, що призводить до розробки та впровадження енергоефективних технологій та систем управління відходами.

У цьому контексті системи візуалізації набувають особливого значення. Вони дозволяють операторам та менеджерам отримувати повну та актуальну картину виробничих процесів у режимі реального часу, що є критичним для прийняття швидких та обґрунтованих рішень. Сучасні системи візуалізації інтегрують дані з різних джерел, включаючи датчики IoT, системи планування ресурсів підприємства (ERP) та системи управління виробництвом (MES), представляючи їх у зрозумілому та інтуїтивно зрозумілому форматі [9].

Важливою тенденцією є також розвиток технологій доповненої (AR) та віртуальної (VR) реальності у промисловій візуалізації. Ці технології дозволяють створювати імерсивні інтерфейси, які надають операторам та інженерам можливість взаємодіяти з віртуальними моделями обладнання та процесів, що значно полегшує навчання, планування та вирішення проблем.

1.1.2 Економічні та технологічні передумови автоматизації

Економічні фактори відіграють ключову роль у стимулюванні розвитку систем візуалізації та автоматизації в промисловості. Глобалізація ринків та посилення конкуренції змушують підприємства шукати нові способи підвищення ефективності та зниження витрат. Автоматизація та впровадження передових систем візуалізації дозволяють досягти цих цілей шляхом оптимізації

використання ресурсів, зменшення часу простою обладнання та підвищення загальної продуктивності.

Зростання вартості робочої сили у багатьох розвинених країнах також є важливим економічним фактором, що стимулює автоматизацію. Впровадження роботизованих систем та автоматизованих ліній виробництва дозволяє підприємствам зменшити залежність від ручної праці та підвищити конкурентоспроможність на глобальному ринку. При цьому системи візуалізації відіграють критичну роль у забезпеченні ефективного контролю та управління автоматизованими процесами [10].

Ще одним важливим економічним аспектом є зростаючий попит на персоналізовану продукцію та послуги. Це вимагає від виробників більшої гнучкості та здатності швидко адаптувати виробничі процеси. Системи візуалізації, інтегровані з системами управління виробництвом, дозволяють оперативно реконфігурувати виробничі лінії та відстежувати виконання індивідуальних замовлень.

З технологічної точки зору, розвиток систем візуалізації та автоматизації стимулюється стрімким прогресом у сфері інформаційних технологій. Зростання обчислювальної потужності комп'ютерів, розвиток хмарних технологій та вдосконалення алгоритмів обробки даних створюють фундамент для реалізації складних систем візуалізації та управління.

Важливою технологічною передумовою є розвиток сенсорних технологій та інтернету речей (IoT). Сучасні датчики здатні збирати величезні обсяги даних про стан обладнання та параметри виробничих процесів. Ці дані стають основою для створення детальних цифрових двійників фізичних об'єктів та процесів, які можна візуалізувати та аналізувати в режимі реального часу.

Прогрес у сфері штучного інтелекту та машинного навчання також відіграє значну роль у розвитку систем візуалізації. Алгоритми AI дозволяють автоматично виявляти аномалії, прогнозувати можливі збої обладнання та оптимізувати виробничі процеси. Інтеграція цих технологій у системи візуалізації дозволяє

створювати інтелектуальні інтерфейси, які не тільки відображають поточний стан системи, але й надають прогнози та рекомендації щодо оптимізації процесів.

Розвиток технологій віртуальної та доповненої реальності відкриває нові можливості для створення імерсивних інтерфейсів у промисловій автоматизації. Ці технології дозволяють операторам та інженерам взаємодіяти з віртуальними моделями обладнання та процесів, що значно полегшує навчання, планування та вирішення проблем. Наприклад, використання AR-окулярів дозволяє техніку отримувати інструкції та діагностичну інформацію безпосередньо під час роботи з обладнанням, що підвищує ефективність технічного обслуговування [11].

Важливо відзначити, що розвиток систем візуалізації тісно пов’язаний з прогресом у сфері мережевих технологій. Впровадження 5G мереж та розвиток промислового Ethernet створюють інфраструктуру для передачі великих обсягів даних з мінімальною затримкою, що є критичним для систем реального часу. Це дозволяє реалізувати розподілені системи візуалізації, які можуть обслуговувати великі промислові комплекси та забезпечувати доступ до даних з будь-якої точки світу.

1.1.3 Вплив систем візуалізації на ефективність промислових процесів

Впровадження сучасних систем візуалізації має значний вплив на ефективність промислових процесів. Перш за все, вони забезпечують повну прозорість операцій, дозволяючи менеджерам та операторам отримувати актуальну інформацію про стан виробництва в режимі реального часу. Це сприяє швидкому виявленню проблем та прийняттю обґрунтованих рішень.

Системи візуалізації також відіграють ключову роль у підвищенні продуктивності праці. Інтуїтивно зрозумілі інтерфейси дозволяють операторам ефективно керувати складним обладнанням та процесами, зменшуючи ймовірність помилок та підвищуючи загальну ефективність виробництва. Крім того, візуалізація історичних даних та тенденцій дозволяє виявляти можливості для оптимізації процесів та підвищення ефективності використання ресурсів.

Важливим аспектом є також вплив систем візуалізації на безпеку виробництва. Своєчасне відображення критичних параметрів та попереджень дозволяє запобігати аваріям та мінімізувати ризики для персоналу та обладнання. Інтеграція систем візуалізації з системами безпеки та контролю доступу підвищує загальний рівень захищеності промислових об'єктів.

1.1.4 Перспективи розвитку систем візуалізації в промисловій автоматизації

Аналіз сучасних тенденцій та технологічних передумов дозволяє окреслити перспективи подальшого розвитку систем візуалізації в промисловій автоматизації. Очікується, що майбутні системи будуть ще більш інтелектуальними та адаптивними, здатними автоматично налаштовуватися під потреби конкретного користувача та специфіку виробничого процесу.

Важливим напрямком розвитку є подальша інтеграція технологій штучного інтелекту та машинного навчання. Це дозволить створювати системи, здатні не тільки відображати поточний стан, але й прогнозувати майбутні події, пропонувати оптимальні рішення та автоматично адаптуватися до змінних умов виробництва.

Очікується також подальший розвиток мобільних та хмарних рішень для промислової візуалізації. Це дозволить забезпечити доступ до критичної інформації з будь-якої точки світу, що особливо важливо в умовах глобалізації виробництва та розподілених команд.

Інтеграція технологій віртуальної та доповненої реальності в системи промислової візуалізації також має значний потенціал. Ці технології можуть революціонізувати процеси навчання персоналу, планування виробництва та технічного обслуговування обладнання.

1.2 Аналіз існуючих систем візуалізації розвантажувальних операцій

У цьому розділі ми проведемо детальний аналіз існуючих систем візуалізації, зосередившись на SCADA-системах та спеціалізованих рішеннях для логістики, а також порівняємо локальні та хмарні системи візуалізації.

SCADA-системи (Supervisory Control and Data Acquisition) вже давно стали стандартом у промисловій автоматизації, і їх застосування у сфері логістики та вантажно-розвантажувальних операцій не є винятком. Ці системи забезпечують комплексний підхід до моніторингу та управління процесами, поєднуючи збір даних, їх обробку та візуалізацію в єдиному інтерфейсі.

Однією з найбільш відомих SCADA-систем, що використовується у логістиці, є Wonderware InTouch. Ця система пропонує широкий спектр можливостей для візуалізації розвантажувальних операцій, включаючи створення інтерактивних панелей управління та звітів у реальному часі. InTouch відрізняється гнучкістю налаштування та можливістю інтеграції з різноманітним обладнанням, що робить її популярним вибором для великих логістичних центрів та складських комплексів.

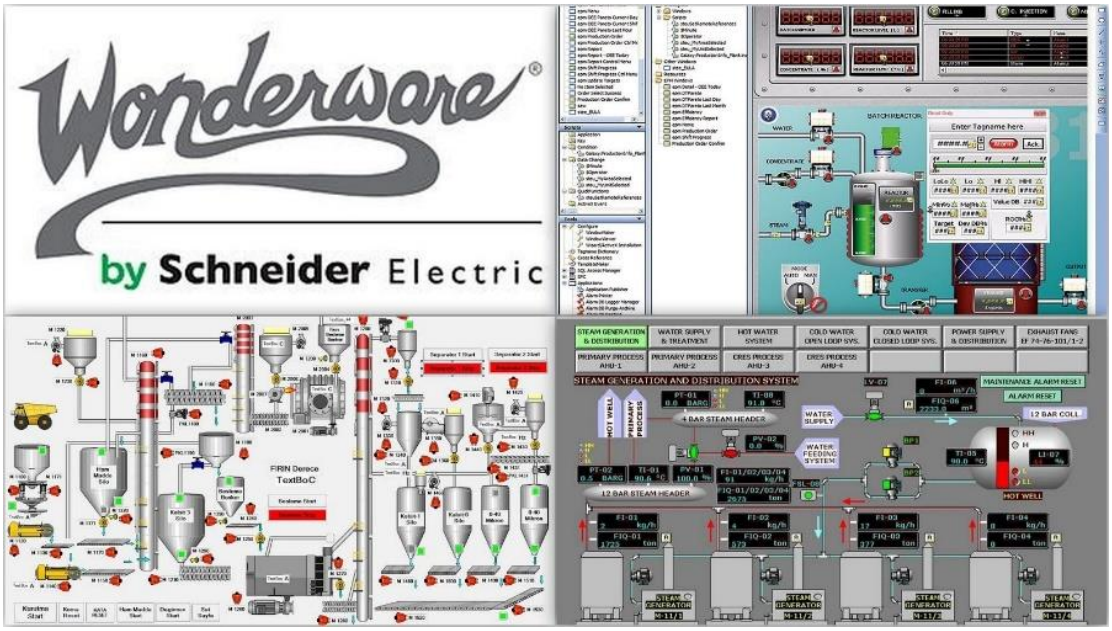


Рисунок 1.1 – Приклад панелі візуалізації Wonderware InTouch

Інша потужна SCADA-система, яка часто застосовується у візуалізації логістичних процесів, це Siemens WinCC. Ця система відрізняється високою масштабованістю та надійністю, що робить її ідеальним вибором для підприємств різного масштабу. WinCC пропонує розширені можливості для 3D-візуалізації, що особливо корисно при моделюванні складних вантажно-розвантажувальних операцій з використанням роботизованих систем, таких як SCARA-роботи.

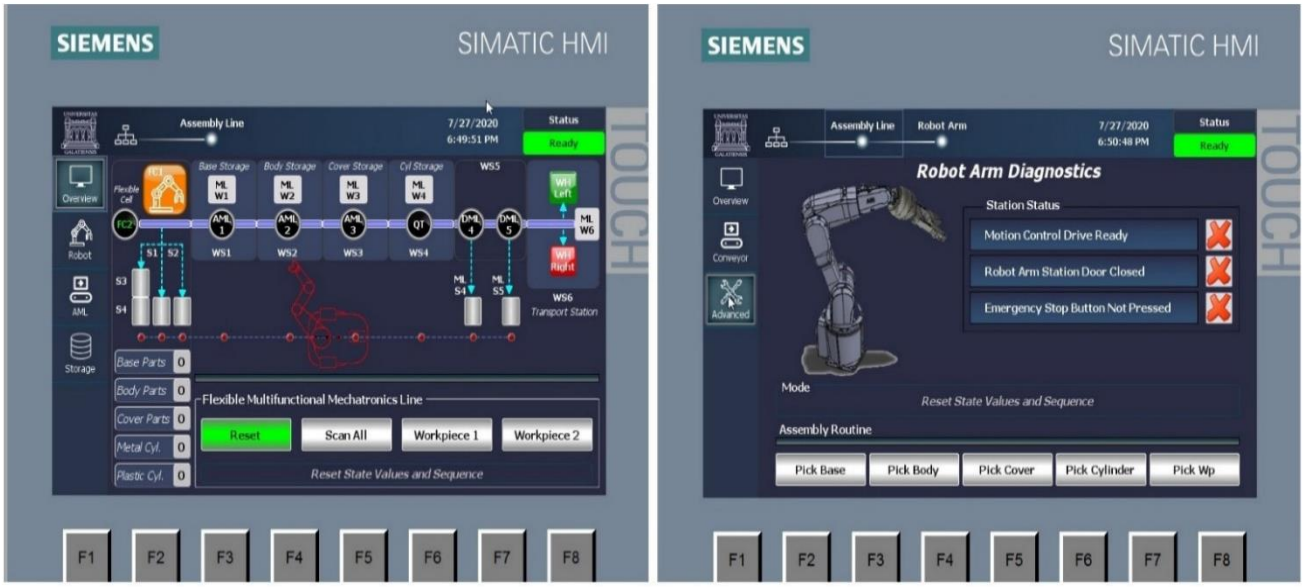


Рисунок 1.2 – Приклад HMI панелі Siemens WinCC

Ще одним важливим аспектом аналізу існуючих систем візуалізації є порівняння локальних та хмарних рішень. Традиційно, системи візуалізації для промислових та логістичних процесів розгорталися локально, на серверах підприємства. Це забезпечувало високий рівень контролю над даними та мінімізувало затримки при обробці інформації. Однак, з розвитком хмарних технологій, все більше компаній розглядають можливість переходу на хмарні системи візуалізації.

Локальні системи візуалізації, такі як згадані раніше Wonderware InTouch та Siemens WinCC, певні особливості. По-перше, вони забезпечують максимальну швидкість обробки даних та відображення інформації, що критично важливо для реагування на надзвичайні ситуації. По-друге, локальні системи дають повний контроль над безпекою даних, що особливо важливо для підприємств з високими вимогами до конфіденційності. Нарешті, локальні системи можуть працювати автономно, не залежачи від доступності інтернет-з’єднання.

З іншого боку, хмарні системи візуалізації, такі як Amazon AWS IoT SiteWise або Microsoft Azure IoT Central, пропонують ряд унікальних переваг. Головною перевагою хмарних рішень є їх масштабованість та гнучкість. Підприємства можуть легко збільшувати або зменшувати обсяг використовуваних ресурсів

залежно від поточних потреб, що особливо корисно для бізнесів з сезонними коливаннями активності.

Хмарні системи також забезпечують легкий доступ до даних з будь-якої точки світу, що стає все більш важливим в умовах глобалізації бізнесу та поширення практик віддаленої роботи. Крім того, хмарні платформи часто пропонують вбудовані інструменти аналітики та машинного навчання, які можуть бути використані для оптимізації логістичних процесів.

Однак, при виборі між локальними та хмарними рішеннями, компанії повинні враховувати специфіку своїх операцій. Для підприємств з критично важливими процесами, де навіть короткочасна втрата зв'язку може призвести до значних збитків, локальні системи все ще залишаються пріоритетним вибором. З іншого боку, для компаній, які прагнуть до максимальної гнучкості та мінімізації витрат на IT-інфраструктуру, хмарні рішення можуть бути оптимальним вибором.

Важливо відзначити, що багато сучасних систем візуалізації пропонують гібридні рішення, які поєднують переваги локальних та хмарних підходів. Наприклад, система може зберігати критично важливі дані та виконувати основні операції локально, одночасно використовуючи хмарні ресурси для довгострокового зберігання даних та складної аналітики [12].

Окремо варто розглянути тенденцію до використання мобільних додатків для візуалізації логістичних процесів. Багато сучасних систем, як локальних, так і хмарних, пропонують мобільні інтерфейси, які дозволяють операторам та менеджерам отримувати доступ до ключової інформації та управляти процесами з мобільних пристроїв. Це особливо корисно для великих складських комплексів, де оперативність реагування на зміни ситуації може значно підвищити ефективність роботи.

Підсумовуючи аналіз існуючих систем візуалізації розвантажувальних операцій, можна зробити висновок, що ринок пропонує широкий спектр рішень, здатних задовольнити потреби підприємств різного масштабу та специфіки. Від потужних SCADA-систем до спеціалізованих логістичних платформ, від локальних рішень до хмарних сервісів – кожен підхід має свої переваги та обмеження.

Вибір оптимальної системи візуалізації залежить від багатьох факторів, включаючи масштаб операцій, вимоги до безпеки та конфіденційності даних, необхідність інтеграції з існуючими системами, а також бюджетні обмеження. При цьому важливо враховувати не лише поточні потреби підприємства, але й перспективи його розвитку, оскільки гнучкість та масштабованість системи візуалізації може стати ключовим фактором у забезпеченні довгострокової ефективності логістичних процесів.

У контексті розробки веб-інтерфейсу для системи візуалізації комплексу моделювання вантажно-розвантажувальних операцій на базі SCARA-роботу, аналіз існуючих рішень дозволяє визначити ключові вимоги та найкращі практики. Зокрема, важливо забезпечити баланс між детальністю відображення інформації та простотою інтерфейсу, реалізувати можливості для якісної візуалізації роботи SCARA-робота, передбачити інтеграцію з іншими системами підприємства та забезпечити доступ до інформації з різних пристроїв, включаючи мобільні.

Таким чином, розробка нової системи візуалізації повинна враховувати сильні сторони існуючих рішень, одночасно адаптуючи їх до специфічних вимог роботи з SCARA-роботами та сучасних тенденцій у веб-розробці. Це дозволить створити інноваційний продукт, який буде відповідати як поточним, так і майбутнім потребам підприємств у сфері автоматизації вантажно-розвантажувальних операцій.

1.3 Аналіз сучасних технологій та підходів до створення веб-інтерфейсів для промислових систем

Візуалізація даних та процесів у промисловості пройшла довгий шлях від простих двовимірних схем до складних тривимірних моделей та інтерактивних інтерфейсів. Сьогодні ми спостерігаємо активне впровадження технологій доповненої (AR) та віртуальної реальності (VR) у промислові системи візуалізації. Кожен з цих підходів має свої переваги та обмеження, які необхідно враховувати при розробці ефективних інтерфейсів для промислових систем.

Двовимірна візуалізація залишається основним підходом для багатьох промислових інтерфейсів завдяки своїй простоті та ефективності. 2D-інтерфейси дозволяють швидко та інтуїтивно представляти інформацію про стан обладнання, потоки матеріалів та виробничі процеси. Вони особливо ефективні для відображення схем, діаграм та панелей управління. Проте, у випадку складних просторових операцій, таких як керування роботизованими маніпуляторами, двовимірне представлення може бути недостатнім для повного розуміння ситуації оператором.

Тривимірна візуалізація надає можливість створювати більш реалістичні та інформативні представлення промислових процесів. 3D-моделі дозволяють операторам краще зрозуміти просторові відношення між об'єктами, що особливо важливо при керуванні роботизованими комплексами. Використання 3D-графіки у веб-інтерфейсах стало можливим завдяки розвитку технологій WebGL та Three.js, які дозволяють створювати високопродуктивні тривимірні сцени безпосередньо у веб-браузері. Це відкриває нові можливості для розробки інтуїтивно зрозумілих інтерфейсів керування складними промисловими системами.

Технології доповненої реальності (AR) починають активно застосовуватися у промисловості, дозволяючи накладати цифрову інформацію на реальні об'єкти. Це особливо корисно для технічного обслуговування, навчання персоналу та візуалізації даних безпосередньо на виробничому майданчику. AR може значно підвищити ефективність роботи операторів, надаючи їм контекстну інформацію та інструкції в реальному часі. Однак, впровадження AR-технологій у промислові системи вимагає вирішення ряду технічних та ергономічних проблем, таких як точність позиціонування, стійкість до промислових умов та зручність використання протягом тривалого часу.

Віртуальна реальність (VR) знаходить своє застосування в основному у сфері навчання та моделювання складних промислових процесів. VR дозволяє створювати повністю імерсивні середовища для тренування персоналу без ризику для реального обладнання. У контексті вантажно-розвантажувальних операцій, VR

може бути використана для симуляції різних сценаріїв та навчання операторів роботі з новим обладнанням.

Веб-інтерфейси мають ряд суттєвих переваг над традиційними десктопними додатками у контексті промислових систем візуалізації. По-перше, веб-технології забезпечують кросплатформенність, дозволяючи доступ до системи з різних пристроїв без необхідності встановлення спеціального програмного забезпечення. Це особливо важливо в умовах сучасного виробництва, де потрібен швидкий та гнучкий доступ до інформації з різних локацій [13].

По-друге, веб-інтерфейси спрощують процес оновлення та підтримки системи. Зміни та покращення можуть бути впроваджені централізовано на сервері, забезпечуючи миттєвий доступ усіх користувачів до актуальної версії інтерфейсу. Це значно знижує витрати на технічне обслуговування та мінімізує ризики, пов’язані з використанням застарілих версій програмного забезпечення.

По-третє, сучасні веб-технології, такі як Progressive Web Apps (PWA), дозволяють створювати інтерфейси, які можуть працювати офлайн та забезпечувати швидкий доступ до критично важливої інформації навіть при нестабільному з'єднанні. Це особливо актуально для промислових об'єктів, де можуть виникати проблеми з мережевим покриттям.

Однак, впровадження веб-інтерфейсів у промислові системи ставить питання безпеки та надійності на перший план. Промислові системи часто працюють з критично важливими процесами, де навіть короткочасний збій може призвести до значних економічних втрат або загроз безпеці. Тому при розробці веб-інтерфейсів для промислових систем необхідно приділяти особливу увагу аспектам кібербезпеки [14].

Для забезпечення безпеки веб-рішень у промисловому середовищі застосовується ряд стратегій. Перш за все, це використання захищених протоколів зв’язку, таких як HTTPS, для шифрування усіх даних, що передаються між клієнтом та сервером. Впровадження строгої аутентифікації та авторизації користувачів з використанням багатофакторної аутентифікації дозволяє мінімізувати ризики несанкціонованого доступу.

Важливим аспектом є також сегментація мережі, яка дозволяє ізолювати критичні системи керування від загальнодоступних мереж. Використання віртуальних приватних мереж (VPN) для доступу до промислових систем з віддалених локацій забезпечує додатковий рівень захисту.

Надійність веб-рішень у промисловому середовищі забезпечується за рахунок використання відмовостійких архітектур. Це включає балансування навантаження між кількома серверами, використання систем кешування для зменшення навантаження на базу даних та забезпечення швидкої роботи інтерфейсу, а також впровадження механізмів автоматичного відновлення після збоїв.

Важливим аспектом надійності є також забезпечення сумісності з різними браузерами та версіями операційних систем. Це досягається шляхом використання стандартизованих веб-технологій та проведення ретельного тестування на різних платформах.

При розробці веб-інтерфейсів для візуалізації комплексу моделювання вантажно-розвантажувальних операцій на базі SCARA-робота необхідно враховувати специфіку даної предметної області. Інтерфейс повинен забезпечувати точне та інтуїтивно зрозуміле представлення положення робота, стану захватів та розташування вантажів. Важливо передбачити можливість швидкого реагування на нештатні ситуації та забезпечити зручні інструменти для планування та оптимізації вантажно-розвантажувальних операцій.

Сучасні веб-технології дозволяють створювати високопродуктивні інтерфейси з використанням таких фреймворків як React, Angular або Vue.js. Ці інструменти забезпечують можливість розробки компонентної архітектури, що спрощує подальшу підтримку та розширення функціональності системи. Для візуалізації SCARA-робота можна використовувати бібліотеки Three.js або Babylon.js, які надають широкі можливості для створення інтерактивних тривимірних сцен у веб-браузері.

Важливим аспектом розробки веб-інтерфейсу для промислової системи є оптимізація продуктивності. Це включає мінімізацію часу завантаження сторінки,

оптимізацію рендерингу складних візуальних елементів та ефективну роботу з великими обсягами даних. Використання технологій серверного рендерингу (SSR) може значно покращити швидкість початкового завантаження інтерфейсу, що особливо важливо в умовах обмеженої пропускної здатності мережі.

Інтеграція веб-інтерфейсу з існуючими промисловими системами керування є ще одним важливим аспектом розробки. Це може вимагати створення спеціалізованих API для обміну даними між веб-додатком та системами реального часу, які безпосередньо керують обладнанням. При цьому необхідно забезпечити надійну синхронізацію даних та мінімізувати затримки в передачі критично важливої інформації.

У контексті візуалізації роботи SCARA-робота, веб-інтерфейс повинен надавати можливість не лише пасивного спостереження за процесом, але й активного керування та планування операцій. Це може включати функціональність для програмування послідовності дій робота, визначення оптимальних траєкторій руху та аналізу ефективності виконання завдань.

Важливим аспектом є також забезпечення масштабованості рішення. Веб-інтерфейс повинен бути здатним адаптуватися до зростання обсягів даних та збільшення кількості користувачів без значного падіння продуктивності. Це може бути досягнуто за рахунок використання мікросервісної архітектури та технологій контейнеризації, таких як Docker, які дозволяють гнучко масштабувати окремі компоненти системи відповідно до зростаючих потреб [15].

У підсумку, створення ефективного веб-інтерфейсу для візуалізації комплексу моделювання вантажно-розвантажувальних операцій на базі SCARA-робота вимагає комплексного підходу, який враховує як технологічні аспекти розробки, так і специфіку промислового середовища. Використання сучасних веб-технологій у поєднанні з передовими методами візуалізації дозволяє створювати інтуїтивно зрозумілі, безпечні та високопродуктивні інтерфейси, які здатні значно підвищити ефективність управління складними промисловими процесами.

1.4 Людино-машинні інтерфейси для вантажно-розвантажувальних операцій

Людино-машинний інтерфейс (HMI) є ключовим елементом у забезпеченні ефективної взаємодії операторів з автоматизованими системами вантажно-розвантажувальних операцій. У контексті роботизованих комплексів на базі SCARA-роботів, проектування та реалізація HMI вимагає особливого підходу, який враховує специфіку роботи з такими системами.

При проектуванні HMI для SCARA-роботів необхідно враховувати специфіку їх кінематики та робочого простору. SCARA-роботи мають унікальну конструкцію, яка дозволяє їм виконувати швидкі та точні операції у горизонтальній площині. Це вимагає від інтерфейсу можливості відображення та контролю положення робота в реальному часі.

Ключовим аспектом проектування ефективних HMI є забезпечення чіткого та зрозумілого представлення інформації. Інтерфейс повинен надавати всю необхідну інформацію про стан системи, положення вантажів та статус виконання завдань, не перевантажуючи оператора надмірною кількістю даних [16].

Ергономіка відіграє ключову роль в розробці HMI для промислових систем. Це включає в себе оптимальне розташування елементів керування, використання контрастних кольорів для покращення читабельності, а також можливість налаштування інтерфейсу під індивідуальні потреби користувача.

Аналіз існуючих рішень для візуалізації роботи SCARA-роботів показує, що більшість сучасних систем використовують комбінацію 2D та 3D візуалізації. 2D представлення зазвичай використовується для відображення загального стану системи та панелей керування. 3D візуалізація, у свою чергу, дозволяє оператору більш детально оцінити положення робота та вантажів у просторі.

Важливим аспектом проектування HMI для роботизованих комплексів є забезпечення можливості віддаленого моніторингу та керування. Веб-інтерфейси в цьому контексті надають значні переваги, дозволяючи здійснювати моніторинг та керування системою з будь-якого пристрою, що має доступ до мережі.

У процесі розробки HMI для вантажно-розвантажувальних операцій важливо проводити регулярне тестування та оцінку ефективності інтерфейсу. Це може включати як лабораторні випробування, так і збір зворотного зв’язку від операторів в реальних умовах експлуатації.

Важливим аспектом проектування HMI є забезпечення можливості ефективного навчання нових операторів. Інтерфейс повинен мати вбудовані навчальні модулі, які дозволяють покроково ознайомитися з функціональністю системи. 3D симуляції можуть бути особливо корисними для навчання, дозволяючи операторам практикуватися в керуванні SCARA-роботом без ризику пошкодження реального обладнання.

У підсумку, проектування та реалізація HMI для вантажно-розвантажувальних операцій на базі SCARA-роботів є комплексним завданням. Ефективний інтерфейс повинен забезпечувати високу продуктивність роботи, мінімізувати ризик помилок та бути інтуїтивно зрозумілим. Використання 3D візуалізації, принципів ергономіки та UX/UI дизайну дозволяє створювати інтерфейси, які не тільки підвищують ефективність роботи, але й покращують умови праці операторів.

1.5 Аналіз структури системи візуалізації комплексу моделювання вантажно-розвантажувальних операцій на базі SCARA-роботу

1.5.1. Компоненти SCARA-робота

SCARA-робот, який є центральним елементом автоматизованої системи, являє собою маніпулятор з трьома ступенями вільності, оптимізований для виконання вантажно-розвантажувальних операцій. Конструкція робота складається з наступних ключових компонентів:

– Вертикальна стійка (башта): Служить основою конструкції та забезпечує обертання навколо вертикальної осі.

– Горизонтальне плече (стріла): Кріпиться до вертикальної стійки та забезпечує рух у горизонтальній площині.

– Підйомний механізм: Розташований на кінці горизонтального плеча, забезпечує вертикальне переміщення робочого органу.

– Робочий орган: Представлений магнітним захватом, здатним утримувати та переміщувати вантажі.

Привід SCARA-робота реалізований на базі трьох двофазних біполярних крокових двигунів 28BJ48-12-300-01. Кожен двигун оснащений відповідним драйвером керування TB6560 V2, що забезпечує точне позиціонування та плавність руху. Використання крокових двигунів дозволяє досягти високої точності позиціонування без необхідності використання додаткових датчиків положення (див рис. 1.3).

Для забезпечення точного контролю руху, кожна вісь робота оснащена енкодером. Ці енкодери надають інформацію про поточне положення кожної ланки робота, що дозволяє системі керування здійснювати точне позиціонування та відслідковувати траєкторію руху.

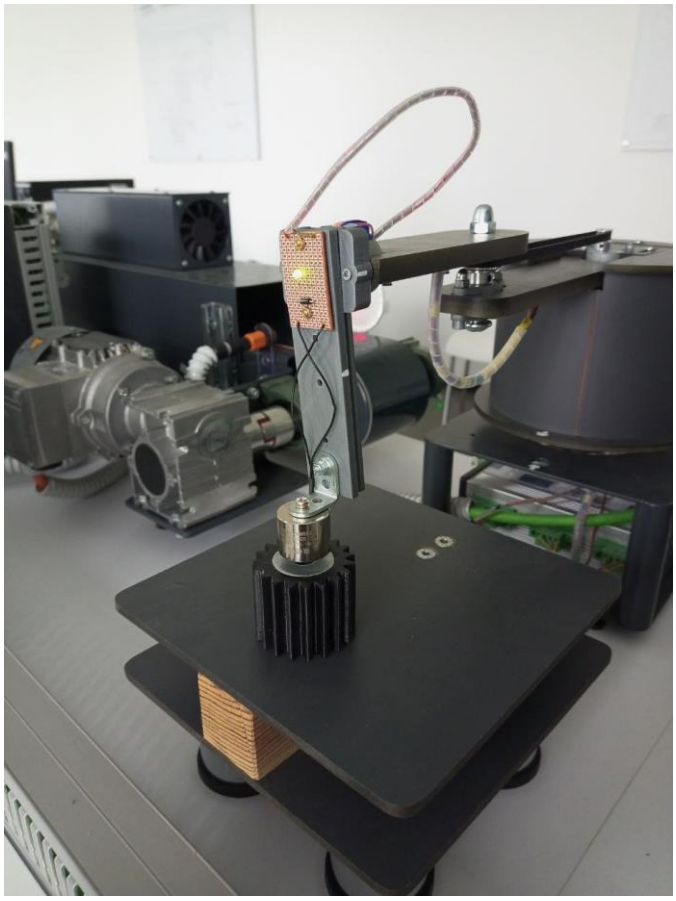


Рисунок 1.3 – SCARA-робот системи візуалізації комплексу моделювання вантажно-розвантажувальних операцій

1.5.2. Технічне забезпечення

Система керування SCARA-робота базується на програмованому логічному контролері (ПЛК) Siemens S7-1200, який забезпечує надійне та ефективне керування всіма компонентами стенду. Зокрема, використовується модель CPU 1215C (6ES7215-1AG40-0XB0), яка оптимально підходить для задач керування роботом та обробки даних у реальному часі.

Основні функції ПЛК S7-1200 в системі керування включають:

– Генерацію керуючих сигналів для крокових двигунів маніпулятора.

– Обробку сигналів зворотного зв’язку від енкодерів.

– Керування допоміжними пристроями (нагрів, вентилятори).

– Взаємодію з панеллю оператора та іншими периферійними пристроями.

– Реалізацію алгоритмів керування та обробки даних.

Для розширення можливостей системи керування використовується модуль аналогового вводу/виводу SM1234 (6ES7234-4HE32-0XB0). Цей модуль забезпечує інтерфейс для підключення додаткових аналогових датчиків та виконавчих механізмів, розширюючи функціональність стенду.

Живлення системи забезпечується двома блоками живлення:

– PM1207 (6EP1332-1SH71) на 24В постійного струму - для живлення ПЛК та основних компонентів.

– LOGO!Power (6EP3310-6SB00-0AY0) на 5В постійного струму - для живлення низьковольтних компонентів та датчиків.

Додатково в системі використовується регулятор живлення PSU100D (6EP1334-1LD00), який забезпечує додаткове джерело 24В постійного струму для периферійних пристроїв.

Для забезпечення комунікації між компонентами системи використовується комунікаційний модуль CSM1277 (6GK7277-1AA10-0AA0). Цей модуль створює локальну мережу Ethernet/PROFINET, що дозволяє здійснювати швидкий обмін даними між ПЛК, панеллю оператора та іншими пристроями.

1.5.3. Компонування та конструкція випробувального стенду

Випробувальний стенд SCARA-робота спроектовано з урахуванням вимог компактності, функціональності та зручності використання в лабораторних умовах.

Всі компоненти стенду розміщені на міцній металевій рамі з габаритними розмірами 275x350x190 мм.

Конструкція стенду включає наступні основні елементи:

– Рама: Забезпечує стабільність конструкції та зменшує вплив зовнішніх вібрацій.

– Електрична шафа: Розміром 25x40 см, розташована в нижній частині рами. В ній змонтовано ПЛК, блоки живлення, та інше електрообладнання.

– SCARA-робот: Встановлений на верхній частині рами, з робочою зоною, оптимізованою для демонстрації та вивчення принципів роботи.

– Зона для тестових об’єктів: Призначена для розміщення вантажів та моделювання різних сценаріїв роботи.

Система безпеки стенду включає аварійну кнопку «СТОП» на передній панелі та програмні засоби захисту від перевантажень та зіткнень.

Кабельні з’єднання організовані з урахуванням мінімізації електромагнітних завад. Для заземлення використовується провід марки LAPP KABEL 2,5мм² з ізоляцією зелено-жовтого кольору.

Стенд має інтерфейс системи візуалізації на базі SIMATIC WinCC, який забезпечує:

– Відображення поточних та цільових позицій механізмів робота.

– Керування режимами роботи (ручний, автоматичний, стоп).

– Моніторинг стану магнітного захвату та інших параметрів системи.

– Візуалізацію технологічного процесу в режимі реального часу.

Хоча існуюча HMI-система (див. рис. 1.4) забезпечує основні функції керування та моніторингу, вона має певні обмеження, зокрема відсутність віддаленого доступу та обмежені можливості аналітики. Ці обмеження створюють передумови для розробки веб-інтерфейсу системи візуалізації, що дозволить

розширити функціональність, забезпечити гнучкість та покращити ефективність роботи з системою.

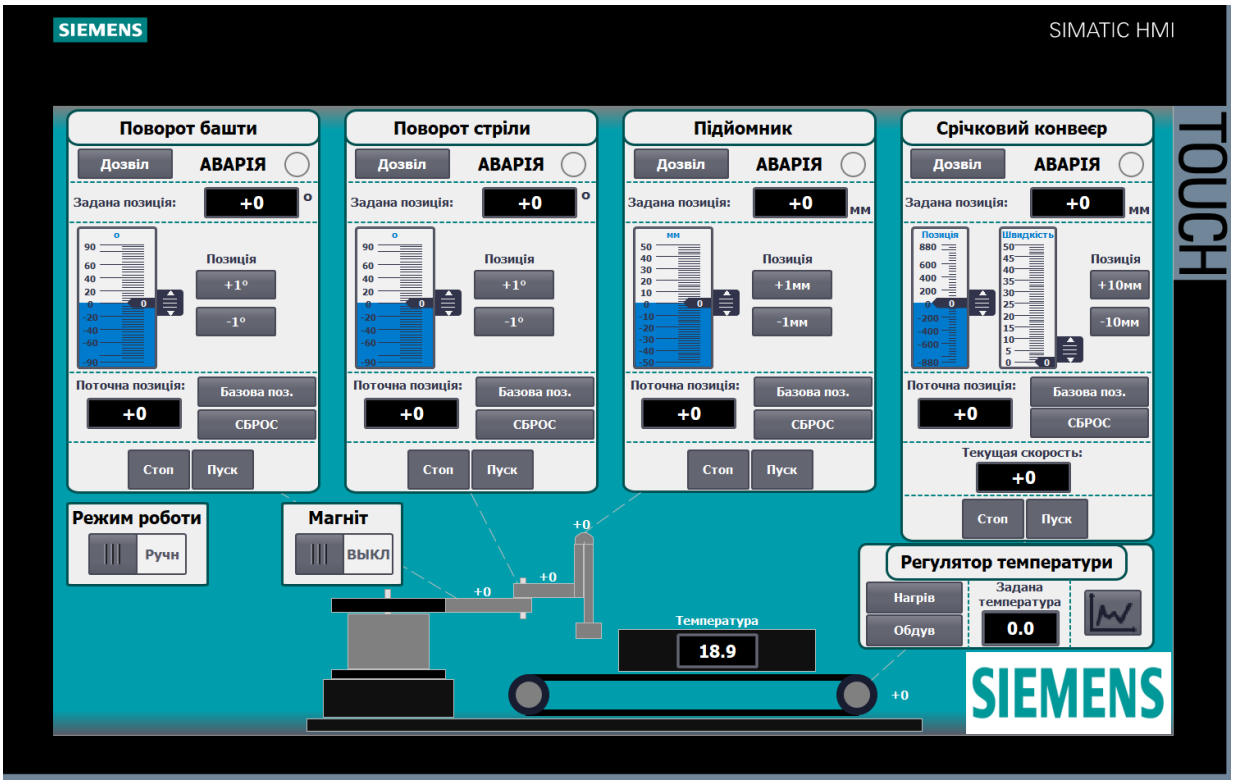


Рисунок 1.4 – HMI панель заданої системи на базі WinCC

1.6 Формулювання вимог та постановка задачі розробки веб-інтерфейсу для візуалізації комплексу моделювання

На основі проведеного аналізу сучасних тенденцій в промисловій автоматизації, існуючих систем візуалізації розвантажувальних операцій, технологій створення веб-інтерфейсів та особливостей проектування людино-машинних інтерфейсів для роботизованих комплексів, можна сформулювати ключові вимоги та завдання щодо розробки веб-інтерфейсу системи візуалізації комплексу моделювання вантажно-розвантажувальних операцій на базі SCARA-роботу.

Як було встановлено в попередніх підрозділах, сучасні тенденції в логістиці та виробництві вимагають все більшої автоматизації процесів, що підкреслює актуальність розробки ефективних систем візуалізації та керування. Аналіз

існуючих рішень показав, що більшість з них не повною мірою відповідають вимогам сучасного виробництва, особливо в контексті роботи зі специфічними роботизованими системами, такими як SCARA-роботи.

Врахувавши переваги веб-технологій, виявлені в підрозділі 1.3, стає очевидним, що розробка саме веб-орієнтованого інтерфейсу є оптимальним рішенням для візуалізації та керування комплексом моделювання вантажно-розвантажувальних операцій. Веб-інтерфейс забезпечить необхідну гнучкість, доступність та можливість віддаленого керування, що є критично важливим в сучасних умовах виробництва.

Базуючись на принципах проектування ефективних HMI та ергономіки промислових інтерфейсів, розглянутих у підрозділі 1.4, можна сформулювати наступні ключові вимоги до розроблюваної системи:

– Комплексна 2D-візуалізація SCARA-робота, що дозволить операторам точно оцінювати ситуацію.

– Забезпечення відображення даних у реальному часі, включаючи позицію складових робота, стан виконання завдань та ключові показники процесу.

– Інтуїтивно зрозумілий інтерфейс керування, який дозволить операторам ефективно взаємодіяти з системою, мінімізуючи ризик помилок.

– Впровадження системи сповіщень та попереджень для своєчасного інформування про потенційні проблеми чи нестандартні ситуації.

– Адаптивний дизайн: Забезпечити, щоб веб-застосунок коректно відображався та функціонував на різних пристроях та екранах, від настільних комп’ютерів до мобільних девайсів.

– Можливість управляти та контролювати кожен елемент системи

Висновки до розділу:

У цьому розділі було проведено комплексний аналіз сучасних технологій та підходів до створення ефективних людино-машинних інтерфейсів (HMI) для роботизованих комплексів, зокрема SCARA-роботів, які використовуються для

виконання вантажно-розвантажувальних операцій. Дослідження охоплювало кілька важливих аспектів, зокрема специфіку роботи SCARA-роботів, сучасні методи візуалізації промислових процесів, а також можливості використання веб-технологій у розробці інтерфейсів для промислових систем.

Аналіз продемонстрував, що для ефективної взаємодії операторів із системами керування роботами необхідно розробити якісний веб-інтерфейс. Цей інтерфейс повинен забезпечувати реалістичне та інформативне відображення роботи SCARA-робота, передачу даних у режимі реального часу, зручні засоби для віддаленого керування, а також високий рівень ергономічності, надійності та безпеки.

Розроблюваний веб-інтерфейс має вирішити такі ключові завдання:

– Забезпечити повноцінний моніторинг стану елементів системи
– Надати інструменти для оперативного керування процесами.
– Створити систему сповіщень про потенційні проблеми.
– Реалізувати адаптивний дизайн для роботи на різних пристроях.

Впровадження такого веб-інтерфейсу суттєво підвищить ефективність управління роботизованими комплексами, створюючи для операторів зручний, надійний та інформативний інструмент взаємодії із системою.

РОЗДІЛ 2

РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ АВТОМАТИЧНОГО КЕРУВАННЯ СТЕНДУ

2.1 Загальні принципи архітектури IIoT

Під час проектування загальної архітектури веб-інтерфейсу системи візуалізації комплексу моделювання вантажно-розвантажувальних операцій на базі SCARA-роботу основну увагу буде приділено концепції Промислового Інтернету Речей (IIoT - Industrial Internet of Things).

IIoT є одним з ключових компонентів парадигми Індустрії 4.0 та революційним підходом до організації виробничих процесів, який кардинально відрізняється від традиційних промислових систем, характерних для Індустрії 3.0. На відміну від застарілих ізольованих та негнучких систем управління, IIoT передбачає інтеграцію фізичних пристроїв, машин та обладнання в єдину мережу для безперервного збору, обміну та аналізу даних у режимі реального часу [17].

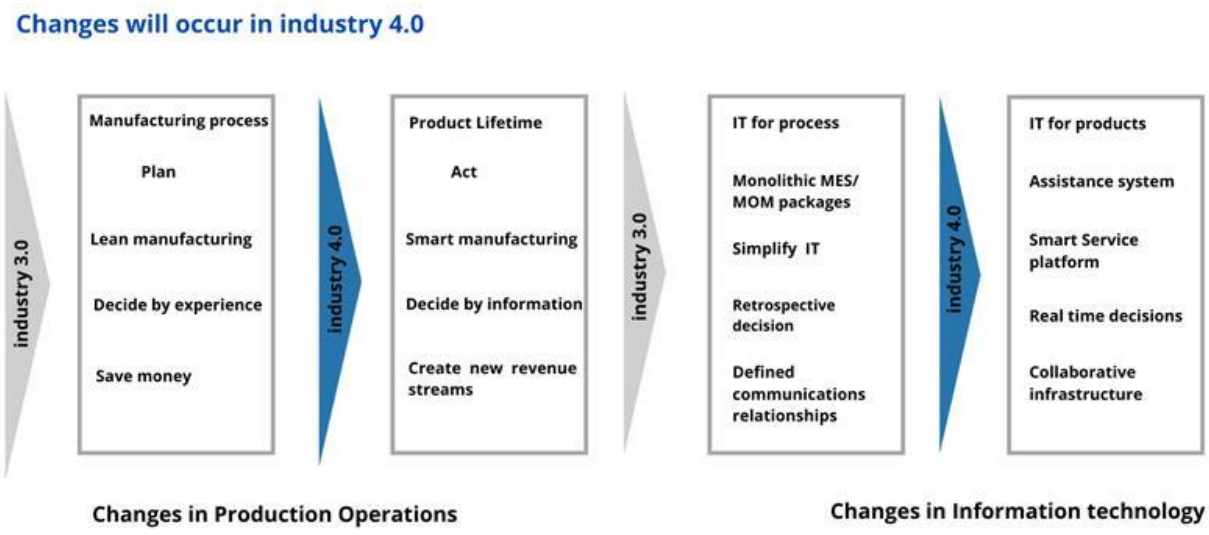


Рисунок 2.1 – Парадигми Індустрії 4.0 проти Індустрії 3.0

Концепція IIoT зародилася на початку 2010-х років як логічне продовження ідей Інтернету речей (IoT) для побутової сфери, але з урахуванням специфічних

вимог промислового середовища. Основними проблемами, які IIoT покликаний вирішити, є:

- Ізольованість та неефективна інтеграція різнорідних систем управління та обладнання на виробництві.
- Відсутність безперервного моніторингу стану обладнання та процесів у режимі реального часу.
- Складність збору, консолідації та аналізу великих обсягів виробничих даних.
- Недостатня гнучкість та адаптивність виробничих систем для швидкого реагування на зміни.
- Обмежені можливості для оптимізації процесів та прогнозної аналітики.

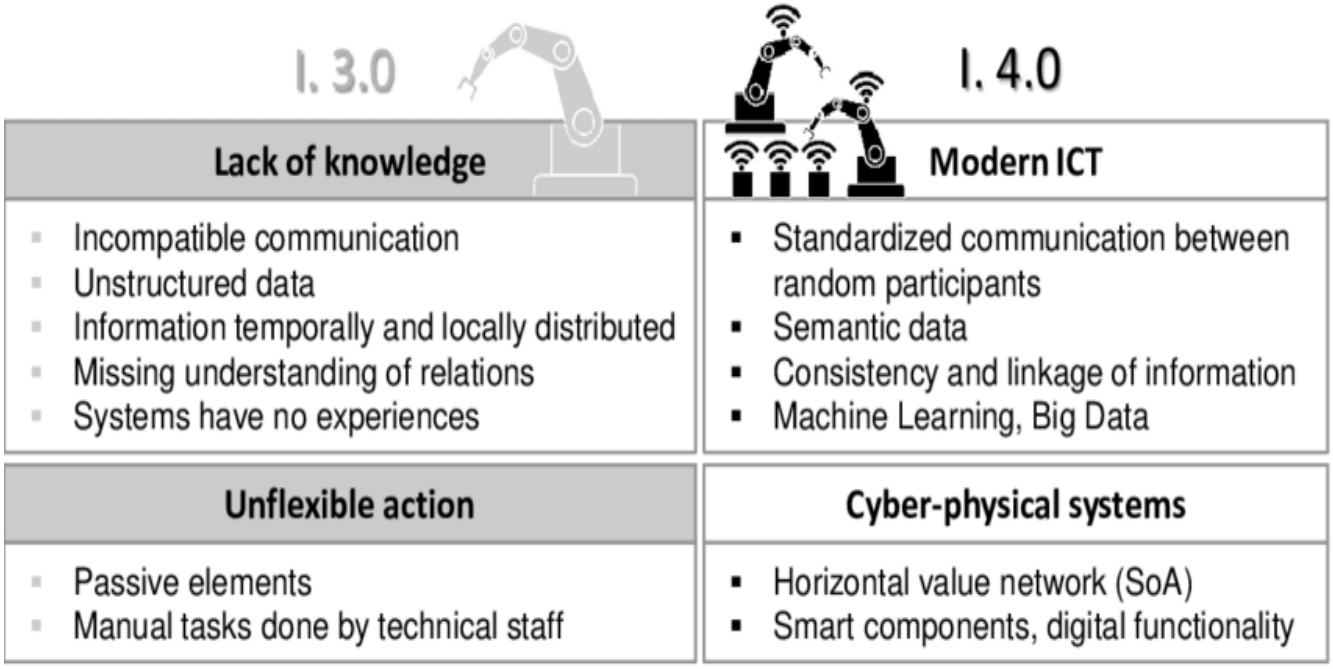


Рисунок 2.2 – Переваги Індустрії 4.0

Для вирішення цих проблем IIoT пропонує архітектуру, яка складається з трьох основних рівнів:

– Рівень пристроїв (датчики, виконавчі механізми, контролери).

– Рівень передачі даних (мережі, шлюзи, брокери повідомлень).

– Рівень аналітики та управління (сервери, хмарні платформи, системи візуалізації).

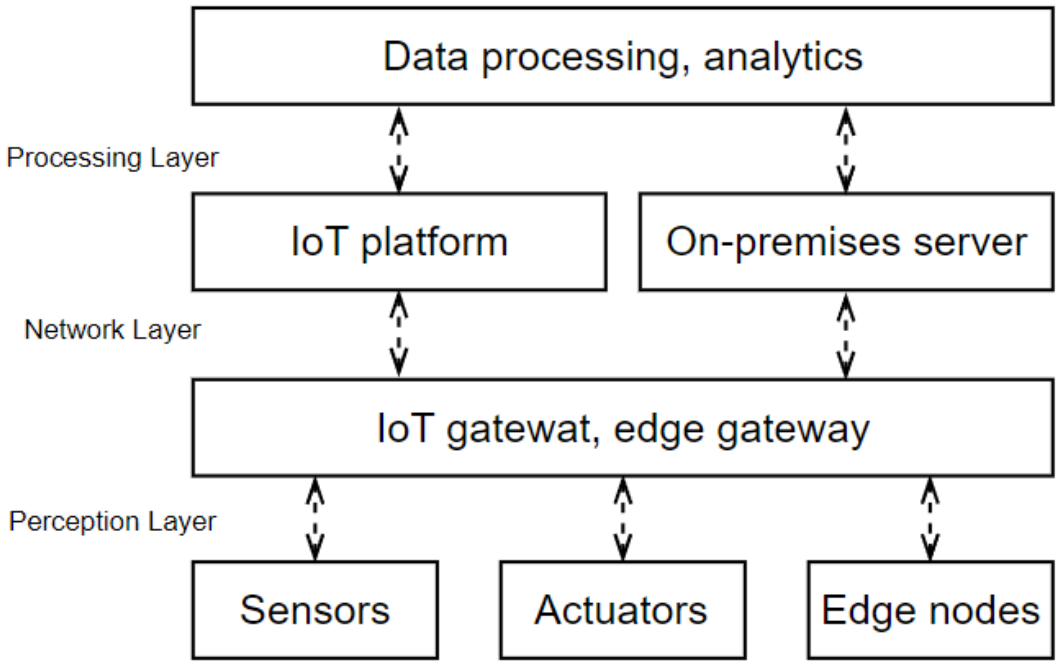


Рисунок 2.3 – Архітектура IIoT

На рівні пристроїв розміщуються різноманітні датчики, виконавчі механізми, програмовані логічні контролери та інші компоненти, що безпосередньо взаємодіють з фізичним процесом. Ці пристрої збирають дані про стан обладнання, параметри процесів та передають їх на вищі рівні.

Рівень передачі даних забезпечує зв'язок між пристроями та рівнем аналітики та управління. Він включає промислові мережі, шлюзи, брокери повідомлень та інші компоненти, що дозволяють організувати безпечний та ефективний обмін даними між розподіленими компонентами системи.

Рівень аналітики та управління є «серцем» IIoT-системи. На цьому рівні розміщуються потужні обчислювальні ресурси, такі як сервери, хмарні платформи, системи великих даних та машинного навчання. Вони забезпечують збір, консолідацію, аналіз та візуалізацію даних з нижчих рівнів, а також надають можливості для прогнозування, оптимізації та прийняття рішень на основі аналітики.

Таким чином, IIoT дозволяє створити інтегровану екосистему, в якій дані безперервно збираються з різноманітних джерел, передаються та аналізуються для отримання цінних бізнес-інсайтів та підвищення ефективності виробничих

процесів. Саме на принципах IIoT буде побудовано загальну архітектуру веб-інтерфейсу системи візуалізації, забезпечуючи гнучку інтеграцію компонентів, ефективний збір даних, потужну аналітику та зручну візуалізацію у веб-середовищі.

2.2 Вибір технологій для створення інформаційно-комунікаційної системи

2.2.1 Рівень пристроїв

У контексті архітектури Промислового Інтернету Речей (IIoT), описаної в попередньому розділі, рівень пристроїв є фундаментальною ланкою, що забезпечує первинний збір даних та їх подальшу передачу для обробки та візуалізації. У розроблюваній системі, особливого значення набуває вибір ефективних протоколів для взаємодії компонентів на рівні пристроїв.

Первинний збір даних про положення маніпулятора, стан двигунів, параметри руху та спрацювання захватного пристрою відбувається на рівні програмованого логічного контролера (ПЛК) Siemens S7-1200 з моделлю CPU 1215C (6ES7215-1AG40-0XB0). Ця модель ПЛК забезпечує надійне та ефективне керування всіма компонентами стенду, виконуючи такі основні функції:

- Генерація керуючих сигналів для крокових двигунів маніпулятора
- Обробка сигналів зворотного зв'язку від енкодерів
- Керування допоміжними пристроями (нагрів, вентилятори)
- Взаємодія з панеллю оператора та іншими периферійними пристроями
- Реалізація алгоритмів керування та обробки даних

Для розширення можливостей системи керування використовується модуль аналогового вводу/виводу SM1234 (6ES7234-4HE32-0XB0), який забезпечує інтерфейс для підключення додаткових аналогових датчиків та виконавчих механізмів.

Зобразимо на рисунку схему підключення компонентів стенду для керування двигуном потужністю 0,12 кВт (див. рис. 2.4):

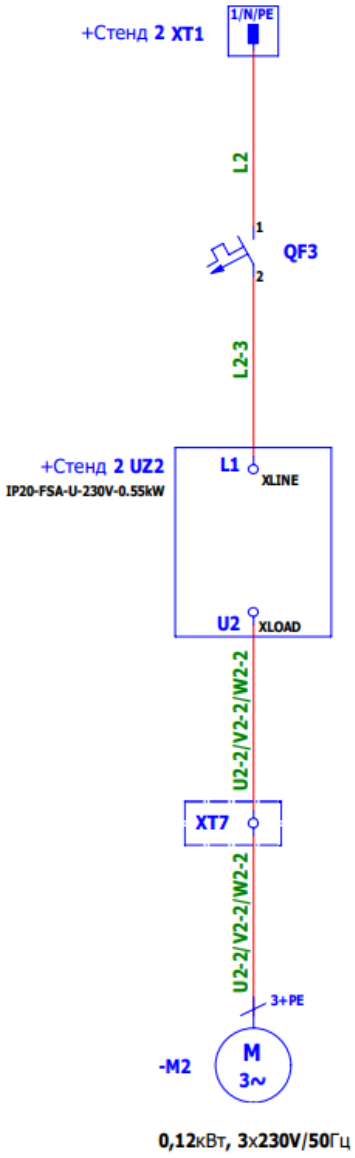


Рисунок 2.4 – Схема підключення двигуна

Основні елементи, відображені на схемі:

– XT1: Вхідний клемний блок, що забезпечує підключення трифазної напруги 230 В.

– QF3: Автоматичний вимикач для захисту обладнання від перевантаження та короткого замикання.

– U22: Силовий модуль SINAMICS G120 (6SL3210-1PB13-0UL0), який виконує функцію керування двигуном через інтегровані інтерфейси.

– M2: Трифазний двигун (0,12 кВт, 3×230 В, 50 Гц) з прямим підключенням до виходу силового модуля.

– XT7: Клемний блок для підключення вихідних проводів до двигуна.

Силовий модуль SINAMICS G120 (елемент UZ2) також містить блок управління с (6SL3244-0BB12-1FA0) з підтримкою PROFINET, що дозволяє інтеграцію до загальної мережі керування. Управління частотою двигуна забезпечує високу точність виконання команд, отриманих від ПЛК.

Живлення системи здійснюється двома блоками живлення: PM1207 (6EP1332-1SH71) на 24В постійного струму для живлення ПЛК та основних компонентів, та LOGO!Power (6EP3310-6SB00-0AY0) на 5В постійного струму для живлення низьковольтних компонентів та датчиків. Також використовується регулятор живлення PSU100D (6EP1334-1LD00), який забезпечує додаткове джерело 24В постійного струму для периферійних пристроїв.

Для забезпечення комунікації між компонентами системи використовується комунікаційний модуль CSM1277 (6GK7277-1AA10-0AA0), який створює локальну мережу Ethernet/PROFINET.

На рівні безпосереднього керування виконавчими механізмами та зчитування сигналів від датчиків використовуються протокол PROFINET.

Наступним завданням є передача даних від ПЛК до IoT-платформи для їх подальшої обробки, інтеграції та візуалізації. Для цього обрано протокол OPC UA. Основними перевагами OPC UA є:

– Незалежність від апаратної платформи та операційної системи

– Висока масштабованість, що дозволяє розширювати систему без зміни базової архітектури

– Інтегровані механізми безпеки, включно з шифруванням та автентифікацією

– Можливість роботи з великими обсягами даних

– Підтримка семантичного моделювання, що спрощує інтеграцію та обробку даних

Таким чином, використання PROFINET та OPC UA у комплексі забезпечує надійну, високошвидкісну та безпечну передачу даних між різними рівнями системи – від польового рівня датчиків/приводів до верхнього рівня IoT-платформи. Ця архітектура з використанням різних промислових протоколів

дозволяє ефективно інтегрувати компоненти системи керування SCARA-робота, забезпечуючи потрібну функціональність, продуктивність та масштабованість.

Зібрані ПЛК дані транслюються через OPC UA до IoT-платформи, яка виступає проміжним шаром між апаратним забезпеченням та інтерфейсом користувача. Цей підхід забезпечує точну, синхронізовану та безпечну передачу даних із мінімальними затримками, що є критично важливим для системи управління та візуалізації SCARA-робота.

2.2.2 Рівень передачі даних

Рівень передачі даних виступає ланкою, що забезпечує інтеграцію між різнорідними компонентами інформаційної системи. Проєктування цього рівня вимагає створення архітектури, здатної обробляти потоки даних із високою надійністю, низькими затримками та високою масштабованістю.

Для реалізації системи моделювання вантажно-розвантажувальних операцій я обираю платформу Node-RED, яка забезпечує всі необхідні можливості для інтеграції та управління даними в рамках IIoT-рішень. Node-RED є легковажним середовищем на базі Node.js із потужним графічним інтерфейсом, який дозволяє реалізовувати доволі складну логіку [18].

Однією з головних переваг Node-RED є її модульна архітектура та розширюваність. Завдяки доступу до великої екосистеми модулів, платформа легко адаптується до специфічних вимог проєкту. Вбудована підтримка JavaScript дозволяє створювати власні вузли для реалізації найскладніших сценаріїв інтеграції. Node-RED також підтримує широкий спектр протоколів і підходів до обміну даними, що робить її ідеальним вибором для роботи в промислових середовищах.

На рівні передачі даних промислові системи висувають суворі вимоги до протоколів. У розроблюваній системі обрано протокол MQTT (Message Queuing Telemetry Transport), який ідеально підходить для передачі даних у розподілених системах. MQTT базується на принципі видавець-підписник і характеризується низькими витратами мережевих ресурсів, високою стабільністю навіть за умов

обмеженої пропускної спроможності каналів зв'язку, а також підтримкою різних рівнів якості обслуговування (QoS).

Для забезпечення сумісності MQTT із веб-середовищем використовується технологія MQTT over WebSocket, яка дозволяє транслювати повідомлення через стандартні WebSocket-з'єднання. Це рішення успішно долає обмеження базового протоколу та забезпечує прозору інтеграцію з сучасними клієнтськими інтерфейсами.

2.2.3 Сервіс візуалізації

У процесі розробки веб-інтерфейсу системи моніторингу та управління SCARA-роботом особливої ваги набуває вибір архітектурних рішень та технологічного стеку, здатного забезпечити високу продуктивність, масштабованість та зручність взаємодії користувача з промисловою системою. Архітектура сервісу візуалізації виступає критичним елементом, який перетворює потоки технічних даних у зрозумілий та інформативний користувацький інтерфейс.

Важливим аспектом архітектурного рішення є також забезпечення кібербезпеки та надійного захисту інформаційних потоків. Впровадження сучасних протоколів автентифікації та шифрування даних становить фундаментальну умову для промислових систем управління. Криптографічний захист комунікаційних каналів між веб-інтерфейсом та SCARA-роботом реалізується через використання протоколів TLS/SSL та багаторівневої системи автентифікації користувачів.

Фундаментом архітектурного підходу обрано Single Page Application (SPA), яка дозволяє створити динамічний, інтерактивний інтерфейс з миттєвою реакцією на зміни даних. SPA-архітектура особливо ефективна в контексті промислових систем, де необхідність безперервного оновлення інформації про стан обладнання є критичною. React як базова бібліотека для реалізації інтерфейсу надає потужний інструментарій для побудови компонентної структури, що максимально наближає веб-інтерфейс до принципів модульності та перевикористання коду [19].

Архітектурний підхід також передбачає впровадження стратегії прогресивного покращення продуктивності та оптимізації рендерингу

компонентів. Використання механізмів мемоїзації, оптимізації повторних рендерів та лінивого завантаження компонентів дозволяє досягти високої ефективності роботи веб-інтерфейсу навіть при складній архітектурі та великій кількості динамічних елементів.

Архітектурний патерн Model-View-Controller (MVC) забезпечує чітке розмежування логіки додатку, що особливо важливо при роботі з складними промисловими системами. Модель відповідає за зберігання та обробку даних, отриманих від SCARA-робота через MQTT-брокер, View відповідає за презентаційний шар, а Controller опосередковує взаємодію між ними, реалізуючи бізнес-логіку додатку [20].

Додатковою перевагою обраної архітектури є можливість імплементації розширених механізмів логування та моніторингу системних подій. Впровадження детального трекінгу операцій, автоматичної генерації звітів про системні взаємодії та журналювання критичних подій дозволяє забезпечити високий рівень діагностики та контролю за роботою промислової системи.

Особливістю обраного технологічного рішення є глибока інтеграція з бібліотекою MQTT, яка забезпечує миттєву комунікацію між контролером та веб-інтерфейсом. Цей підхід дозволяє реалізувати принцип реактивності – будь-які зміни стану обладнання негайно відображаються в інтерфейсі без додаткових запитів чи перезавантажень сторінки.

Бібліотека Material-UI (MUI) обрана як базовий інструмент для створення уніфікованого та естетичного інтерфейсу. Особливо важливо, що MUI надає спеціалізовані компоненти для відображення технічних даних, наприклад індикатори стану, які критично необхідні при роботі з промисловими системами.

Архітектура додатку передбачає чітке структурування коду з використанням сучасних практик модульної розробки. Кожен компонент інтерфейсу розглядається як автономна одиниця, що інкапсулює власну логіку та стан. Такий підхід спрощує подальшу підтримку та масштабування системи, дозволяючи легко додавати нові функціональні можливості без перебудови всієї архітектури.

Наведена архітектура демонструє комплексний підхід до проектування високоефективних веб-інтерфейсів для промислових систем управління, де кожен архітектурний рішення підпорядковане меті максимальної функціональності, безпеки та зручності користувача.

2.3 Структура системи та логіка роботи

Маючи сформуваний список програмних та технічних рішень, можна сформувати структуру комплексу засобів, що необхідні для реалізації веб-інтерфейсу, а також схему цієї структури, що буде відображати типи інформаційних потоків у даній системи (див. рис. 2.4).

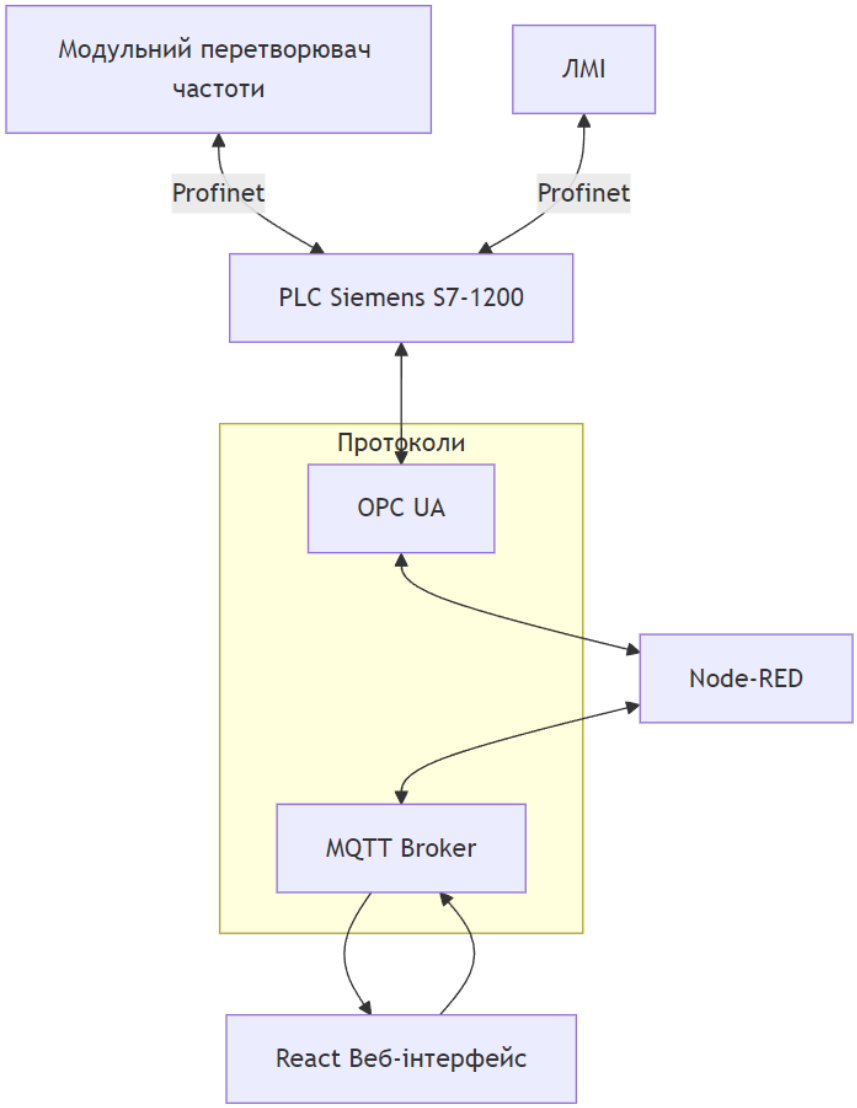


Рисунок 2.5 – Відображення інформаційних потоків в системі

На найнижчому рівні ієрархіє знаходяться датчики та виконавчі пристрої, а саме: лазерні датчики, сервоприводи, двигун конвеєра та інше. Вони підключені до ПЛК Siemens S7-1200, який приймає дані з датчиків та відправляє команди керування на виконавчі механізми.

Дані з OPC UA сервера на контролері S7-1200 передаватимуться через мережевий шлюз на сервер Node-RED. Цей програмний комплекс буде здійснювати початкову обробку (наприклад, запис даних або базове зчитування) і агрегацію даних, а також забезпечуватиме візуалізацію основних параметрів системи в зручному для користувача інтерфейсі. Для комунікації між сервером Node-RED та React можна використовувати протокол MQTT, через вказаний в попередньому підпункті брокері.

Для роботи з MQTT у Node-RED використовуються спеціалізовані вузли «mqtt in» та «mqtt out». Вони дозволяють підписуватися на певні топіки MQTT та публікувати повідомлення на інші топіки відповідно.

Процес обміну даними через MQTT виглядає так:

– Node-RED публікує оброблені дані від ПЛК на певні топіки MQTT.

– Веб-інтерфейс на основі React підписується на ці топіки за допомогою бібліотеки MQTT.

– Брокер MQTT розсилає повідомлення усім підписникам, включаючи React-додаток.

– React-додаток отримує повідомлення та відображає дані у веб-інтерфейсі.

– Коли користувач взаємодіє з інтерфейсом, React публікує команди на інші топіки MQTT.

– Node-RED отримує команди користувача та передає їх до ПЛК через OPC UA.

Така архітектура забезпечує гнучкість, масштабованість та надійність передачі даних між компонентами системи, дозволяючи легко інтегрувати додаткові елементи через підключення до відповідних топіків MQTT.

Висновки до розділу:

У другому розділі кваліфікаційної роботи представлено дослідження архітектури системи автоматичного керування стендом на базі SCARA-робота з використанням концепції Промислового Інтернету Речей (IIoT). Розроблена архітектура демонструє сучасний підхід до проектування інтегрованих промислових систем, який принципово відрізняється від традиційних підходів Індустрії 3.0.

Ключовим елементом досілідження став багаторівневий архітектурний підхід, що включає рівень пристроїв, передачі даних та аналітики й управління. На рівні пристроїв головним елементом є програмований логічний контролер Siemens S7-1200, який забезпечує надійне керування компонентами стенду та первинний збір даних. В рамках технічного рішення описано використання протоколів PROFINET для локальної комунікації та OPC UA для передачі даних на верхні рівні системи, що гарантує високу швидкодію, безпеку та масштабованість.

Рівень передачі даних реалізовано на базі Node-RED та протоколу MQTT, які створюють гнучке проміжне програмне середовище для інтеграції та маршрутизації потоків інформації. Використання MQTT over WebSocket дозволило забезпечити прозору комунікацію між апаратними компонентами та веб-інтерфейсом.

Архітектура сервісу візуалізації базується на сучасних веб-технологіях, зокрема фреймворк React, що забезпечує динамічний, інтерактивний користувацький інтерфейс з миттєвою реакцією на зміни системного стану. Впровадження архітектурного патерну Model-View-Controller та використання бібліотеки Material-UI дозволить створити модульну, легко розширювану систему з високим рівнем естетики та функціональності.

Розроблена система не лише вирішує поточні завдання моделювання вантажно-розвантажувальних операцій, але й створює гнучку платформу для подальшої модернізації та розширення функціоналу.

РОЗДІЛ 3

ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ІНТЕРФЕЙСУ ДЛЯ ВІЗУАЛІЗАЦІЇ ВАНТАЖНО-РОЗВАНТАЖУВАЛЬНИХ ОПЕРАЦІЙ

3.1 Налаштування проєкту в TIA Portal

Так як в даній магістерській роботі я працюю з реальним проєктом стенду, то розпочну роботу з його огляду та проведення налаштувань під свої вимоги.

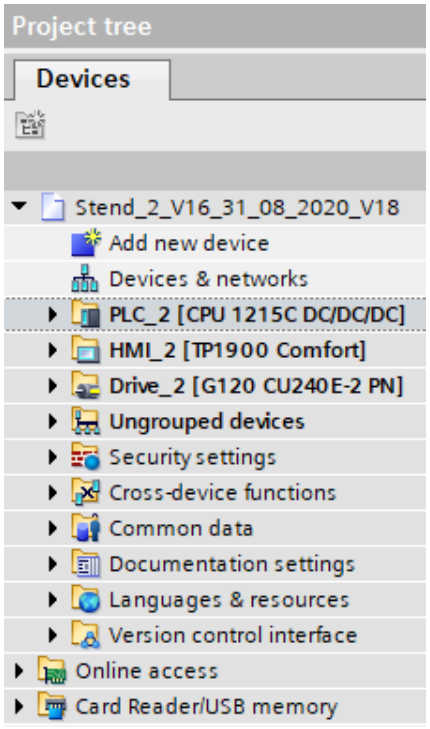


Рисунок 3.1 – Огляд структури проєкту

Як видно з рисунку, до складу проєкту входять наступні основні компоненти, що були розглянуті в розділі 1 та 2:

– PLC_2 [CPU 1215C DC/DC/DC]: Логічний контролер Siemens серії S7-1200, модель CPU 1215C. Це компактний контролер із живленням постійним струмом (DC), з використанням як для входів, так і для виходів.

– HMI_2 [TP1900 Comfort]: Панель оператора Siemens TP1900 Comfort, призначена для взаємодії користувача з системою.

– Drive_2 [G120 CU240E-2 PN]: Перетворювач частоти Siemens SINAMICS G120 з модулем управління CU240E-2 PN.

Для початку роботи з контролером, детальніше оглянемо версію його процесора та змінимо її за потреби.

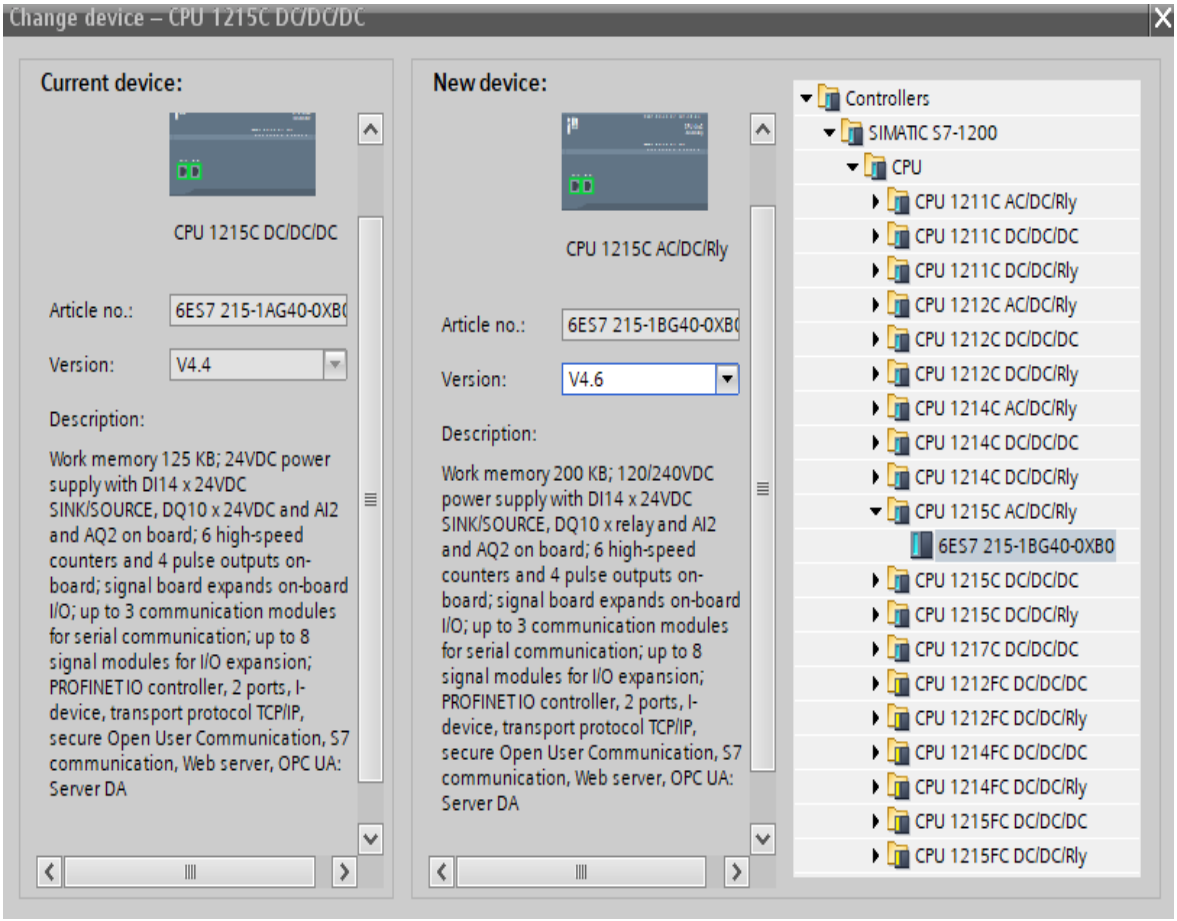


Рисунок 3.2 – Заміна процесу контролера

Отже, в результаті перевірки встановленого процесору (6ES7 215-1AG40-0XB0), видно, що версія 4.2 не підтримує протокол OPC UA, що є однією з головних складових створеної архітектури. Замінюємо його на версію 4.6, яка має всі необхідні можливості.

3.1.1 Налаштування серверу OPC UA

Наступним етапом є налаштування серверу OPC UA. Для цього переходимо в налаштування контролера та встановлюємо відповідні значення (див. рис. 3.3).

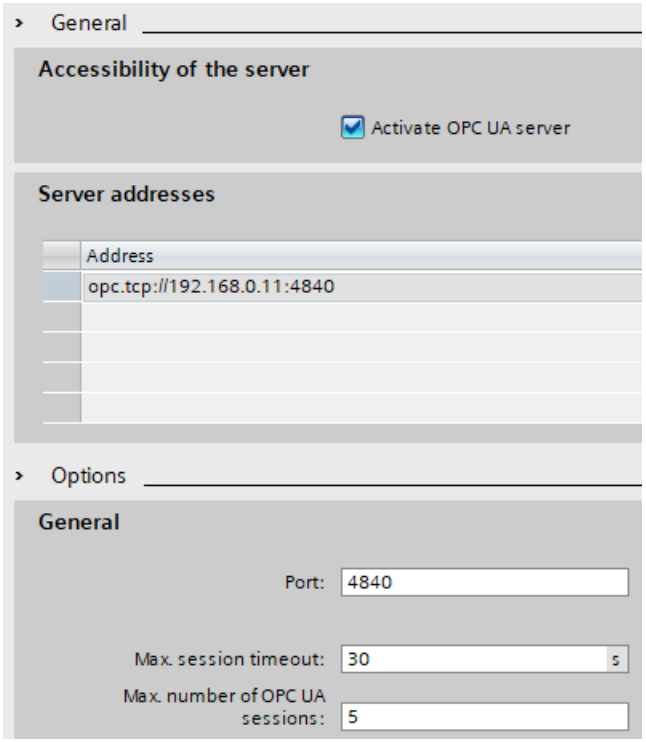


Рисунок 3.3 – Активація серверу

Також задаємо тип ліцензії:

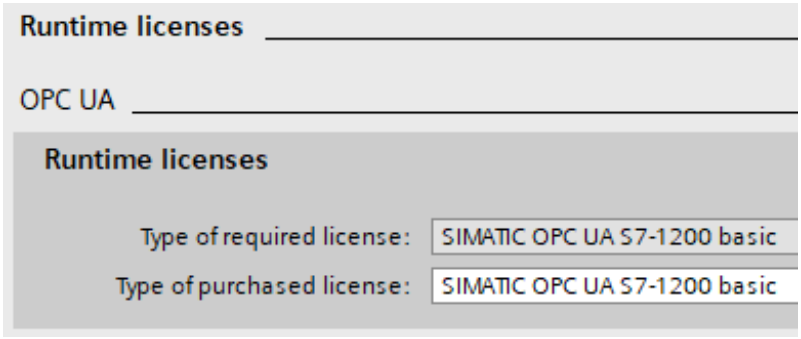


Рисунок 3.4 – Встановлення типу ліцензії

3.1.2 Розробка інтерфейсу серверу

Задля забезпечення можливості зчитування тегів OPC UA клієнтом треба назначити змінні як вузли OPC UA.

Для цього створимо інтерфейс та налаштуємо доступ на запис або читання:

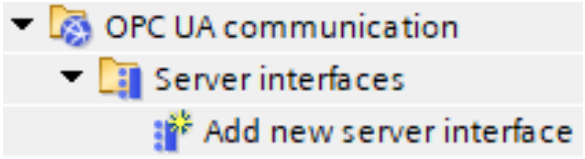


Рисунок 3.5 – Створення нового інтерфейсу серверу

Заповнюємо таблицю інтерфейсу потрібними змінними (таблиця 1), використовуючи, перетягуючи їх з OPC UA елементів, що включають: технологічні об’єкти, програмні блоки та PLC теги (див. рис. 3.6).

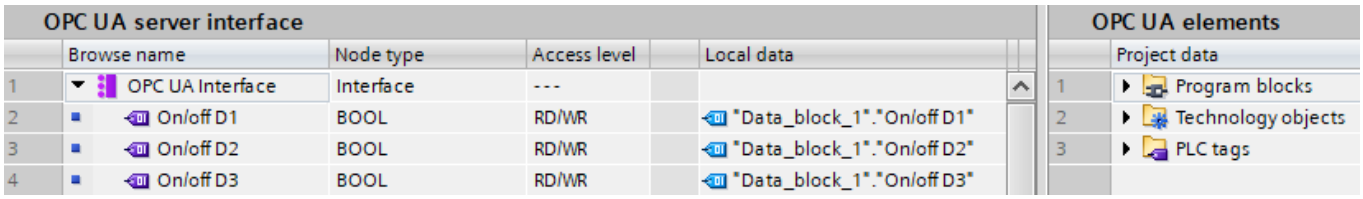


Рисунок 3.6 – Налаштування вузлів

Створимо таблицю зі всіма вузлами та їх параметрами:

Таблиця 3.1 – Створені вузли інтерфейсу серверу OPC UA

Локальна назва	Тип вузла	OPC ID
Axis_1_Error	BOOL	5
Axis_2_Error	BOOL	6
Axis_3_Error	BOOL	7
Data_block_1_OnOffD1	BOOL	8
Data_block_1_OnOffD2	BOOL	9
Data_block_1_OnOffD3	BOOL	10
Data_block_1_angle D1	INT	11
Data_block_1_angle D2	INT	12
Data_block_1_angle D3	INT	13
Data_block_1_home D1	BOOL	14
Data_block_1_home D2	BOOL	15
Data_block_1_home D3	BOOL	16
Data_block_1_move D1	BOOL	17
Data_block_1_move D2	BOOL	18
Data_block_1_move D3	BOOL	19
Data_block_1_reset D1	BOOL	20
Data_block_1_reset D2	BOOL	21
Data_block_1_reset D3	BOOL	22

Продовження табл. 3.1

1	2	3
Data_block_1_stop D1	BOOL	23
Data_block_1_stop D2	BOOL	24
Data_block_1_stop D3	BOOL	25
RightSensor	BOOL	26
LeftSensor	BOOL	27
Start_AUT	BOOL	28
Stop_AUT	BOOL	29
Switch_AUTMAN	BOOL	30
Set_temperature	REAL	31
Actual_temperature	REAL	32
Data_block_1_conv ONOFF	BOOL	33
Data_block_1_conv home	BOOL	34
Data_block_1_conv reset	BOOL	35
Data_block_1_conv stop	BOOL	36
Data_block_1_conv move	BOOL	37
Data_block_1_conv position	INT	38
Data_block_1_conv speed	INT	39
conv_error	BOOL	4
Heating	BOOL	43
Cooling	BOOL	44
magnit	BOOL	45
Axis_1_Actual	REAL	48
Axis_2_Actual	REAL	49
Axis_3_Actual	REAL	50
conv_ActualPosition	REAL	46
conv_ActualVelocity	REAL	47

Після внесення остаточних змін в проєкт – скомпілюємо та завантажимо його на пристрій.

Алгоритм наступний:

– Компіляція (hardware and software)

– Завантаження на пристрій (download to device)

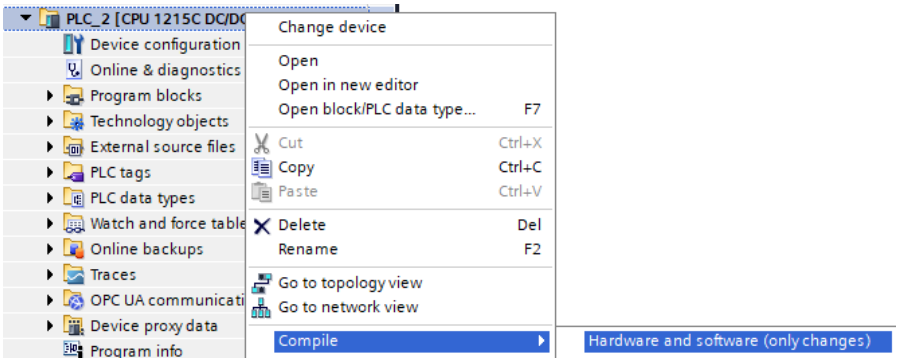


Рисунок 3.7 – Компіляція проєкту

Ця операція (див. рис. 3.7) виконує компіляцію змін у конфігурації апаратного забезпечення та програмного забезпечення контролера. Вибір опції «only changes» означає, що компіляція відбувається лише для тих елементів проєкту, які зазнали змін, що прискорює процес порівняно з повною компіляцією.

Щоб перевірити коректність роботи серверу можна використовувати будь-який OPC UA клієнт, наприклад UaExpert, з яким в мене є досвід роботи.

Отже, у поточному стані проєкту я реалізував всі зміни необхідні для подальшої розробки.

3.2 Розробка коду для платформи Node-RED

Наступним етапом розробки є реалізація інтеграційної платформи між веб-інтерфейсом та стендом, за допомогою інструменту Node-RED.

Розробку програмної частини для заданої реалізації можна розбити на декілька етапів:

– Налаштування з’єднання з OPC UA сервером

– Розробка механізму збору та моніторингу даних

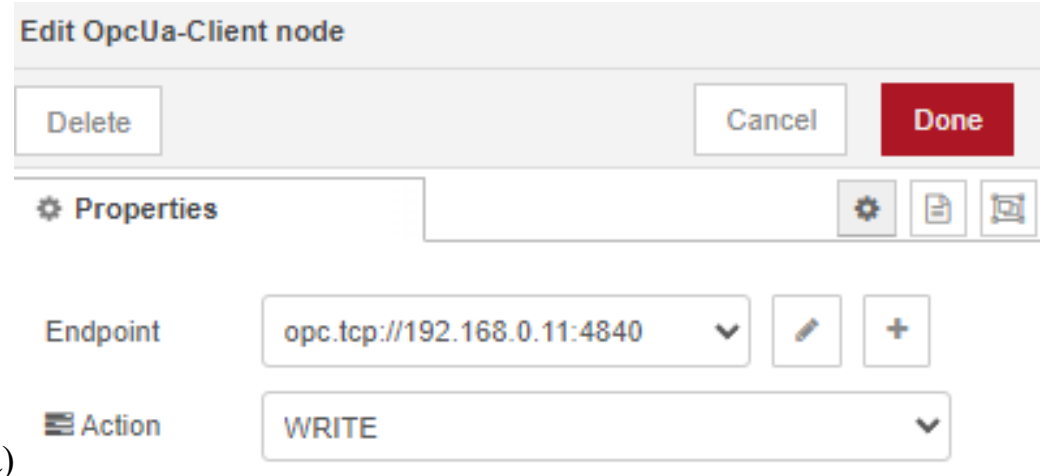
– Реалізація MQTT комунікації

– Розробка механізму контролю та запису

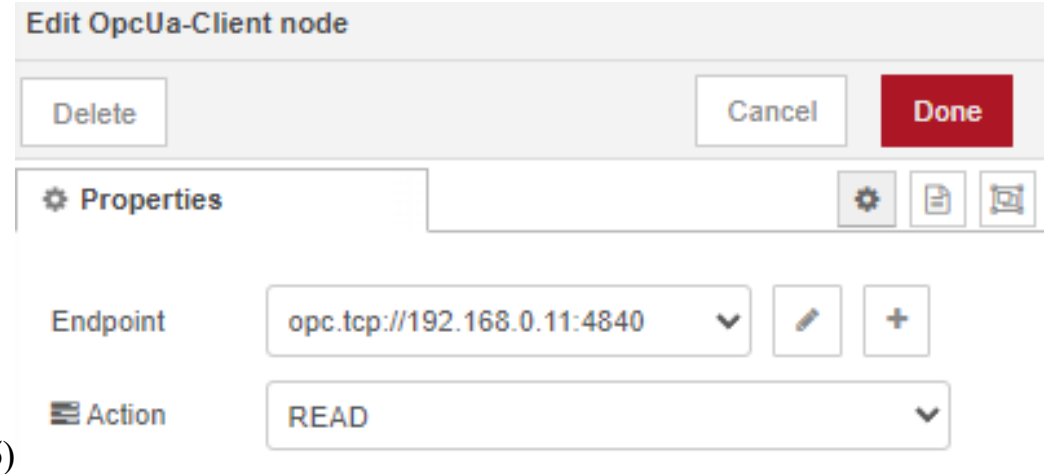
Створимо новий потік в середині середовища розробки платформи. Для налаштування з'єднання з сервером використаємо вузол OPC UA Client з бібліотеки «node-red-contrib-opcua».

Так як поточна реалізація повинна бути посередником між двома платформами, то потрібно налаштувати два вузли для операцій читання та запису.

Розмістимо два елементи клієнту в робочому середовищі та налаштуймо їх, вказавши відповідні операції та endpoint, з параметрами підключення за адресом «opc.tcp://192.168.0.11:4840» (див. рис. 3.8 а, б).



а)



б)

Рисунок 3.8 – а) Параметри налаштування клієнту для запису,

б) Параметри налаштування клієнту для читання

Перед реалізацією механізмів збору, збереження, обробки та запису даних розробимо алгоритм роботи публікації даних до брокеру MQTT (див. рис. 3.9).

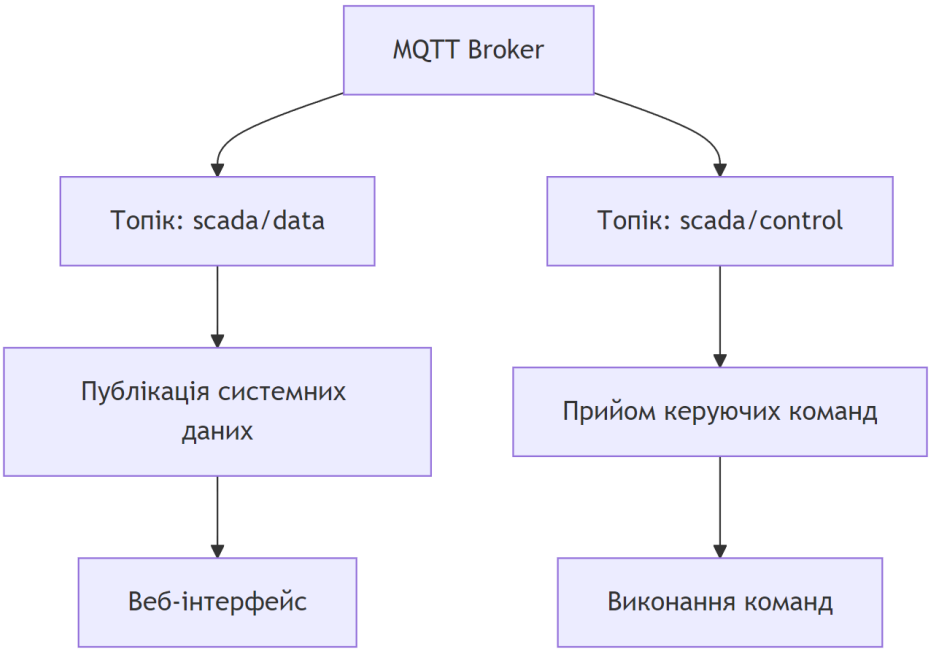


Рисунок 3.9 – Завантаження на пристрій

Для реалізації комунікації треба додати два вузли MQTT: один для публікації даних на топік «scada/data», а інший - для отримання керуючих команд з топіка «scada/control».

Додамо до середовища розробки вузли «mqtt in» «mqtt out» з відповідними топіками та наступними налаштуваннями серверу:

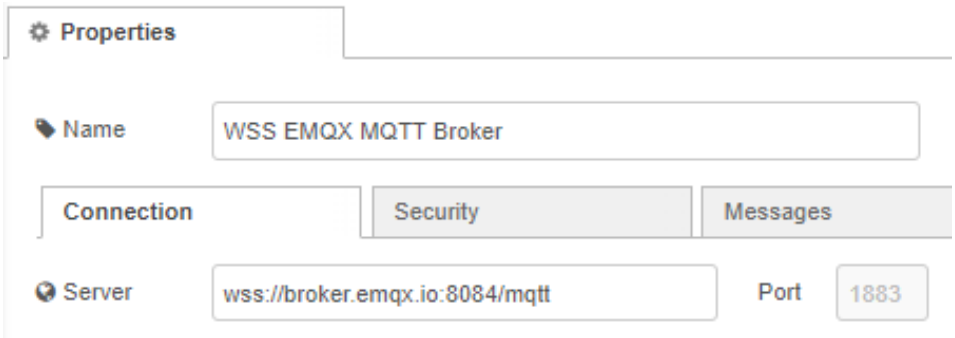


Рисунок 3.10 – Адреса серверу MQTT

Як видно з рисунку 3.10, для організації обміну даними між вузлами системи було використано публічний MQTT брокер, а саме – broker.emqx.io. Цей брокер надавав безкоштовний доступ до MQTT сервісу з підтримкою WebSocket протоколу, що дозволяло підключатись до нього без необхідності налаштовувати власний фізичний сервер.

Використання публічного MQTT брокера спростило розгортання системи та виключило необхідність налаштовувати власну інфраструктуру.

Після розгляду розгортання комунікаційних вузлів, переходимо до детального опису алгоритму роботи всієї програми. Цей алгоритм наочно відображено на блок-схемі, наведеній нижче (див. рис. 3.11).

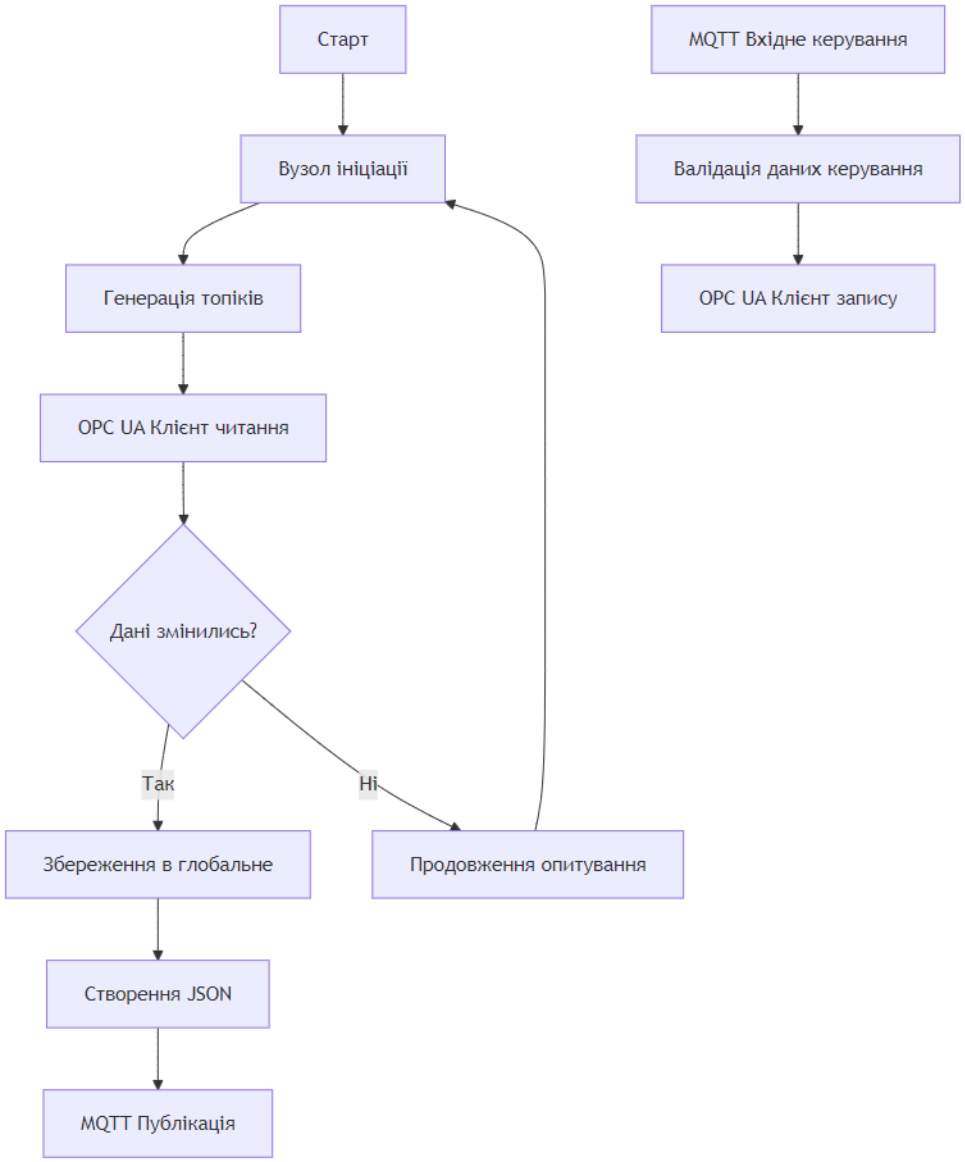


Рисунок 3.11 – Алгоритм роботи програми

Процес починається з ініціації системи - запуску вузла, який періодично генерує списки топіків для зчитування даних з OPC UA сервера. Ці топіки відповідають ідентифікаторам тегів у системі, дані з яких необхідно моніторити та

керувати ними. Генерація списку топіків здійснюється у вузлі «Generate Topics» на блок-схемі. Реалізацію функції представлено на рисунку 3.12.

Далі OPC UA клієнт, налаштований на читання даних, циклічно опитує сервер, зчитуючи поточні значення для кожного топіка. Ці значення порівнюються з попередньо збереженими, і в разі виявлення змін, нові дані записуються до глобального контексту Node-RED (див. рис. 3.13). Варто відзначити, що глобальний контекст використовується для збереження поточного стану системи, що дозволяє легко отримувати необхідні дані в будь-який момент.

```
const nodeIds = [
    "ns=4;i=5", "ns=4;i=6", "ns=4;i=7", "ns=4;i=8", "ns=4;i=9",
    "ns=4;i=10", "ns=4;i=11", "ns=4;i=12", "ns=4;i=13", "ns=4;i=14",
    "ns=4;i=15", "ns=4;i=16", "ns=4;i=17", "ns=4;i=18", "ns=4;i=19",
    "ns=4;i=20", "ns=4;i=21", "ns=4;i=22", "ns=4;i=23", "ns=4;i=24",
    "ns=4;i=25", "ns=4;i=26", "ns=4;i=27", "ns=4;i=28", "ns=4;i=29",
    "ns=4;i=30", "ns=4;i=31", "ns=4;i=32", "ns=4;i=33", "ns=4;i=34",
    "ns=4;i=35", "ns=4;i=36", "ns=4;i=37", "ns=4;i=38", "ns=4;i=39",
    "ns=4;i=42", "ns=4;i=43", "ns=4;i=44", "ns=4;i=45", "ns=4;i=46",
    "ns=4;i=47", "ns=4;i=48", "ns=4;i=49", "ns=4;i=50"
];

let messages = nodeIds.map(nodeId => {
    return { topic: nodeId };
});

return messages;
```

Рисунок 3.12 – Лістинг коду функції Generate Topics

```
let previousValue = global.get(msg.topic);
let currentValue = msg.payload;

global.set(msg.topic, currentValue);

global.set(`${msg.topic}_prev`, previousValue);

return null;
```

Рисунок 3.13 – Лістинг коду функції Save to Global

Коли система вперше запускається, функція «Create JSON» ініціалізує значення для всіх тегів з використанням відповідного глобального контексту. На першому кроці кожному тегу присвоюється початкове значення, якщо воно ще не існує в системі, після чого зберігається його попереднє значення.

Під час подальшої роботи функція перевіряє актуальні значення тегів і порівнює їх з попередніми. Якщо відбулися зміни, вузол формує структуру, що

містить змінені теги та їх нові значення, використовуючи відповідні зручні для читання імена. Ця структура передається в повідомлення формату JSON, яке може бути опубліковане в потрібний MQTT-топік. Якщо ж змін не виявлено, вузол нічого не публікує. Реалізація функції в Додатку.

Таке рішення дозволяє мінімізувати навантаження на систему та зменшити кількість непотрібних повідомлень, гарантуючи при цьому, що кожна зміна буде належно зафіксована та доступна для подальшої обробки.

Окрім публікації оновлених даних, система також обробляє вхідні команди керування, які надходять на топік «scada/control». Ці команди зчитуються MQTT клієнтом, валідуються на предмет відповідності очікуваним ідентифікаторам тегів, і, у разі успішної перевірки, передаються на OPC UA клієнт для запису нових значень на сервері. Таким чином забезпечується надійна та контрольована зміна стану системи. Реалізація функції в Додатку.

Наведена вище блок-схема ілюструє описаний алгоритм роботи програми. Вона включає всі ключові етапи: ініціацію, генерацію топіків, зчитування та порівняння даних, публікацію оновлень, обробку вхідних команд і, врешті-решт, запис нових значень на OPC UA сервер.

Зобразимо остаточний вигляд програми:

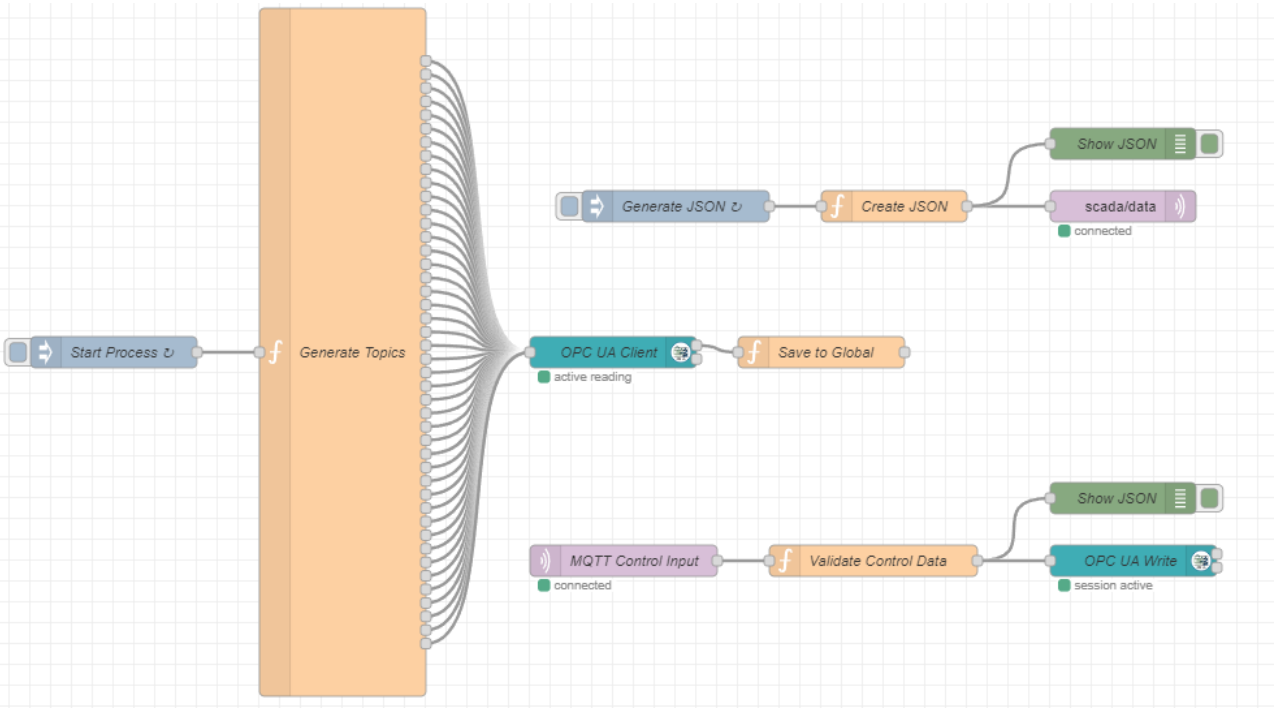


Рисунок 3.14 – Фінальний вигляд системи

У цьому стані система вже готова до тестування, і можна перевірити, в якому форматі дані публікуються на брокер (див. рис. 3.15).

Рисунок 3.15 – Результат формування JSON змінених значень

3.3 Розробка веб-клієнта для візуалізації

Архітектура проєкту побудована з використанням сучасних підходів до створення інтерактивних веб-додатків, зокрема концепції Single Page Application (SPA) з React.js. Вибрані технологічні рішення забезпечують високу продуктивність, масштабованість та зручність розробки користувацького інтерфейсу для системи моделювання.

Технологічний стек включає React.js як базову бібліотеку для побудови компонентної архітектури, Material-UI для уніфікованого дизайну та готових

компонентів інтерфейсу, MQTT.js для забезпечення комунікації з брокером повідомлень та механізми локального зберігання стану додатку.

Структура компонентів веб-клієнта розроблена з урахуванням принципів модульності та розподіленої відповідальності. Основні компоненти включають:

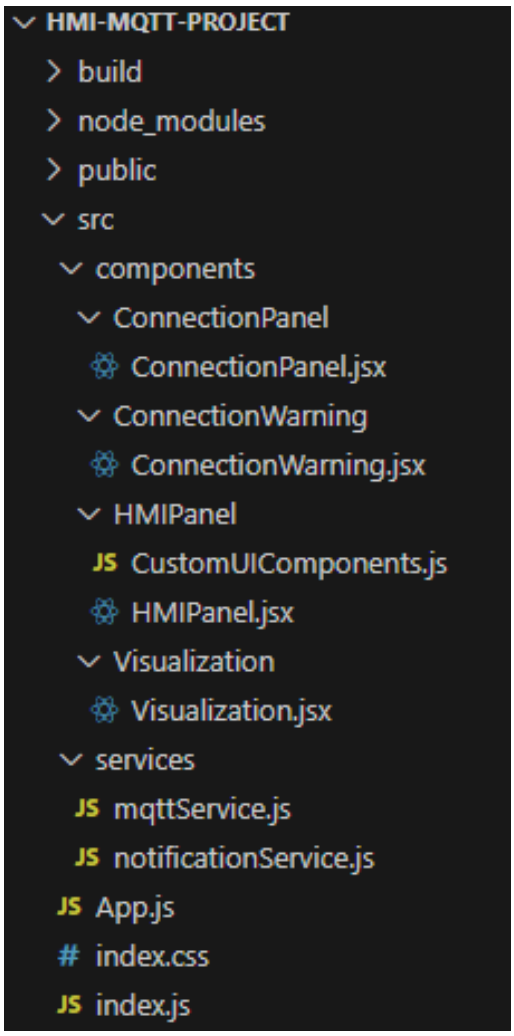


Рисунок 3.16 – Огляд структури веб-інтерфейсу

Детальний розгляд архітектурних рішень дозволяє зрозуміти логіку побудови інтерфейсу. Код файлів наведений в додатках А, Б та В.

Компонент підключення (ConnectionPanel) відповідає за встановлення з’єднання з MQTT-брокером, використовуючи відповідний розроблений сервіс. Реалізація підключення базується на використанні бібліотеки mqtt.js, яка забезпечує надійну комунікацію з брокером повідомлень. Компонент реалізує складну логіку підключення, включаючи механізми повторного з’єднання, обробки помилок та керування станом підключення.

Демонстрація деталей реалізації підключення:

```
export const connectToMQTT = () => {
  return new Promise((resolve, reject) => {
    if (client && client.connected) {
      resolve(client);
      return;
    }

    client = mqtt.connect(MQTT_BROKER_URL, {
      reconnectPeriod: RECONNECT_DELAY,
      connectTimeout: 10000,
    });
```

Рисунок 3.17 – Лістинг функції підключення до MQTT-брокера

HMI-панель (HMIPanel) є ключовим компонентом інтерфейсу, який забезпечує інтерактивне керування системою. Розроблена з використанням підходів реактивного програмування, панель надає користувачеві широкий набір елементів управління основними параметрами роботи комплексу моделювання.

Архітектура компонента побудована з використанням складних патернів React, зокрема мемоїзації компонентів за допомогою React.memo для оптимізації продуктивності. Кожен підкомпонент (ControlUnit, TemperatureControl) має власну логіку обробки взаємодії користувача та передачі керуючих команд.

Компонент Visualization реалізує графічне відображення стану системи. Використання SVG-графіки дозволяє створити динамічну та інтерактивну візуалізацію з можливістю миттєвого оновлення кольорів та станів елементів залежно від отриманих даних. Підхід до побудови графічного інтерфейсу базується на принципах адаптивності та інформативності.

Система сповіщень (NotificationStack) забезпечує зворотній зв’язок з користувачем через механізм спливаючих повідомлень. Реалізована з використанням власних налаштованих компонентів Material-UI, система надає миттєву інформацію про стан з’єднання, результати виконання команд та системні події.

Важливим аспектом архітектури є механізм локального зберігання стану додатку. Використання localStorage дозволяє зберігати поточний стан між

перезавантаженнями сторінки, що підвищує досвід користувача та забезпечує неперервність роботи з системою.

Додаток реалізований з дотриманням принципів реактивного програмування, що забезпечує ефективне оновлення інтерфейсу при надходженні нових даних від MQTT-брокера. Графічне оформлення виконане з використанням Material Design, який надає сучасний, інтуїтивно зрозумілий та лаконічний інтерфейс.

3.4 Інструкція користувача

3.4.1 Загальний огляд сайту

Розроблений веб-інтерфейс є системою візуалізації та управління процесом вантажно-розвантажувальних операцій із застосуванням SCARA-робота. Сайт являє собою повноцінний інструмент для моніторингу та контролю роботи технологічного комплексу, створеного на базі навчального стенда.

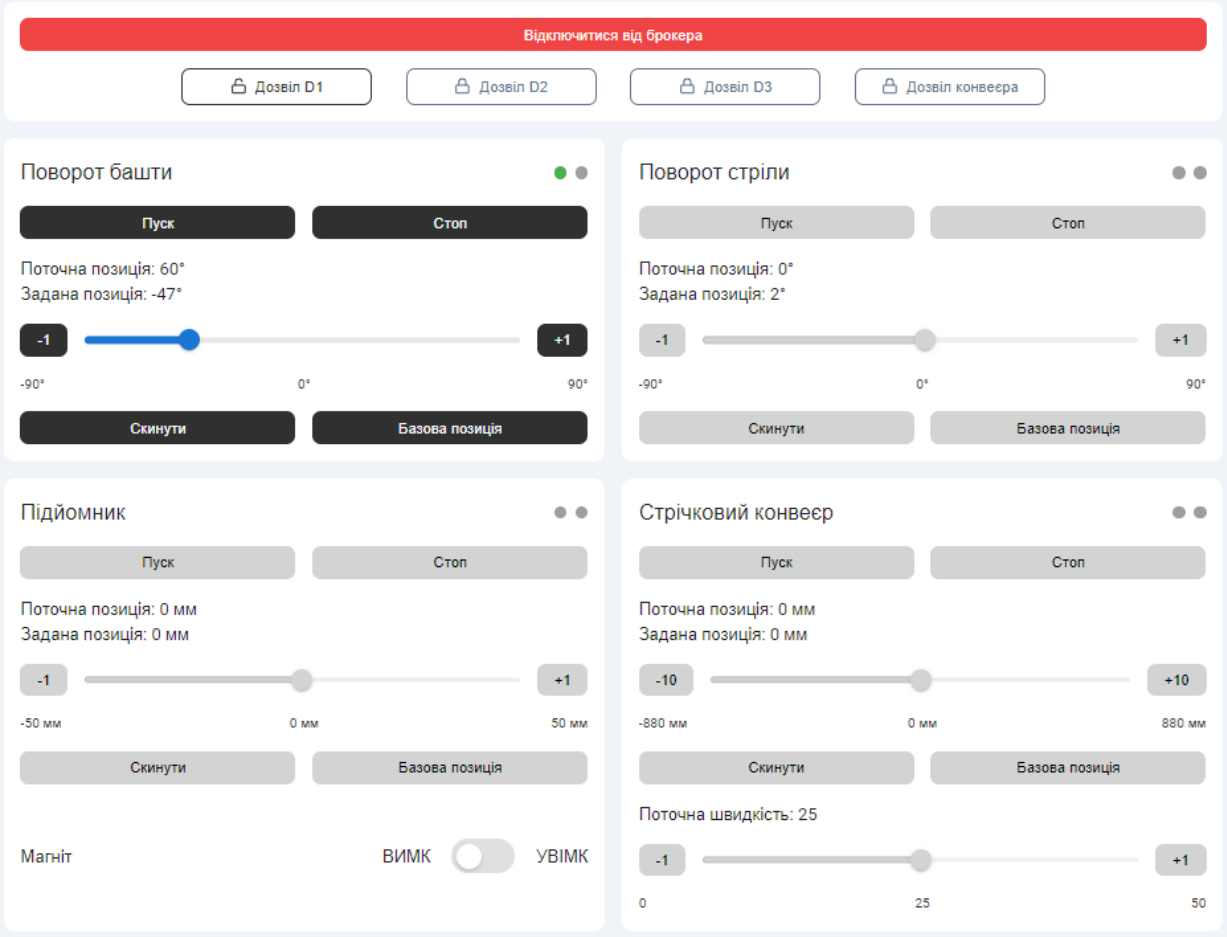


Рисунок 3.18 – Панель керування веб-інтерфейсу

Основним функціональним призначенням сайту є надання користувачеві зручного і наочного інструментарію для управління всіма аспектами діяльності SCARA-робота. Оператор, отримує можливість як спостерігати за поточним станом і процесом виконання операцій, так і безпосередньо втручатись у хід технологічного процесу, коригуючи параметри роботи робота та інших машин.

Структурно сайт складається з таких ключових елементів:

– Панель керування. Центральна частина інтерфейсу, що надає користувачеві повний контроль над роботою стенда (див. рис. 3.18).

– Блок візуалізації. Частина сторінки, де в динамічному режимі відображається процес переміщення робота, конвеєра та стан розміщених датчиків (див. рис. 3.19).

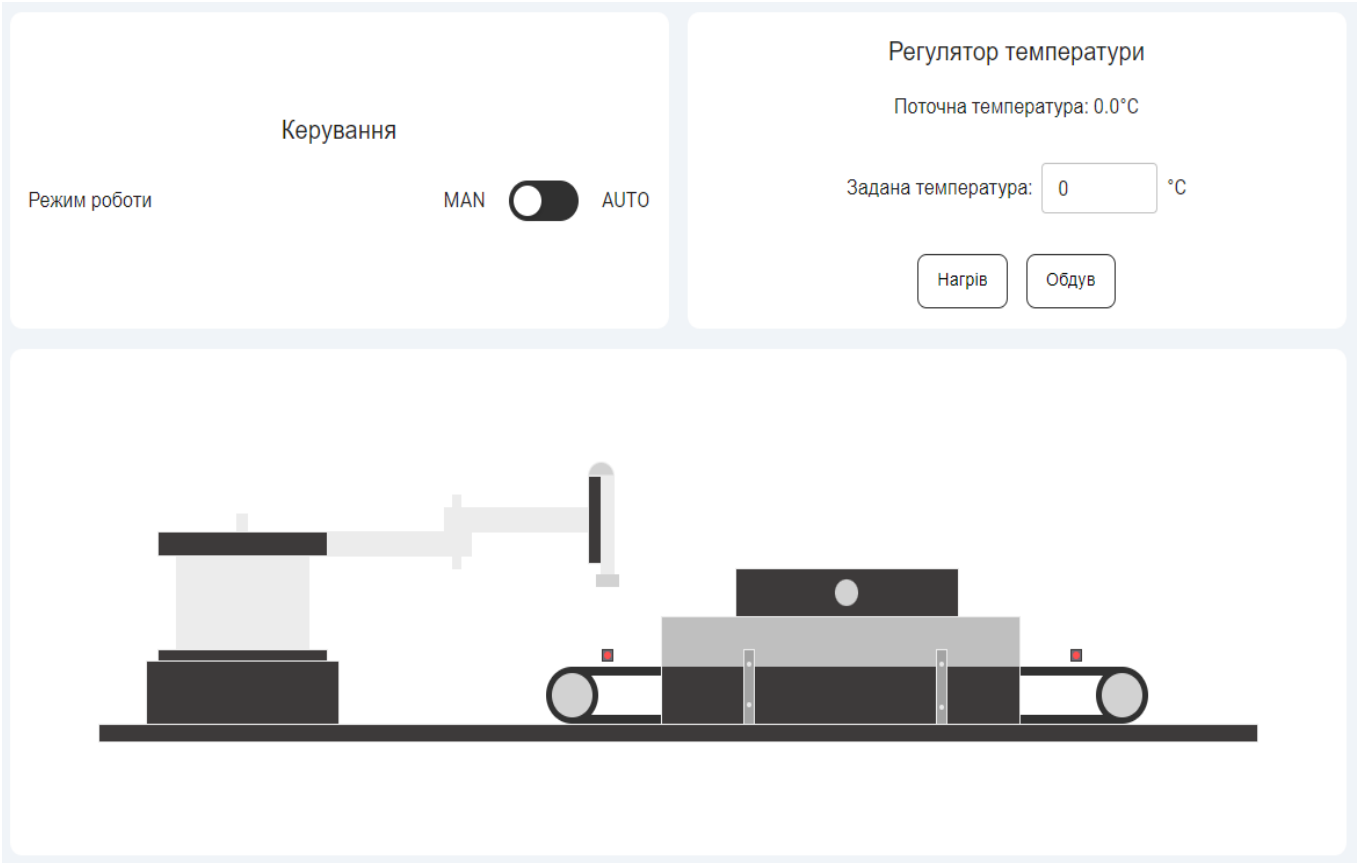


Рисунок 3.19 – Блок візуалізації стенда

Таким чином, сайт являє собою комплексне рішення, що поєднує в собі функціонал моніторингу, управління та візуалізації ключових аспектів роботи SCARA-робота.

3.4.2 Панель керування

Розроблена панель керування візуалізує основні параметри та дозволяє контролювати роботу SCARA-робота. Вона складається з декількох блоків, кожен з яких відповідає за певний аспект управління.

Панель підключення виступає первинним елементом інтерфейсу та відповідає за встановлення з'єднання з MQTT-брокером. Головна функція цього компоненту – забезпечення комунікації між веб-інтерфейсом та системою управління. Панель містить кнопку підключення/відключення та набір кнопок дозволу для окремих вузлів системи (D1, D2, D3 та конвеєр). Кожна кнопка дозволу має два стани – активований та деактивований, що дозволяє здійснювати точний контроль над окремими системами.

Блок «Поворот башти» дозволяє керувати поворотом башти робота. Тут можна задавати цільову позицію, відстежувати поточну позицію, а також запускати, зупиняти та скидати рух башти. Користувач може плавно регулювати положення башти за допомогою слайдера або робити крокові коригування на -1 або +1 градус.

Наступний блок «Поворот стріли» аналогічно керує поворотом стріли робота. Функціонал ідентичний до блоку «Поворот башти», дозволяючи точно налаштовувати кут повороту стріли.

Блок «Підйомник» відповідає за вертикальне переміщення частини руки робота. Тут можна задавати цільову висоту, бачити поточну висоту, а також увімкнути/вимкнути магнітний захват. Користувач має можливість плавно регулювати висоту за допомогою слайдера або робити крокові налаштування з кроком 1 мм.

Блок «Стрічковий конвеєр» дає змогу керувати швидкістю та напрямком руху конвеєра. Аналогічно до попередніх блоків, тут є слайдер для плавного регулювання швидкості в діапазоні від 0 до 50 мм/с, а також кнопки для крокового коригування швидкості.

Окремим блоком представлено «Керування», де користувач може переключати режим роботи між «MAN» (ручним) та «AUTO» (автоматичним). У

разі вибору автоматичного режиму, тут доступні кнопки для запуску та зупинки автоматичного циклу.

Завершує панель керування блок «Регулятор температури», де відображається поточна температура, є можливість задати бажану температуру, а також увімкнути або вимкнути режими нагріву та охолодження.

Загалом, розроблена панель керування надає повний набір інструментів для ефективного управління SCARA-роботом. Зручний інтерфейс з різноманітними органами керування дозволяє користувачу гнучко налаштовувати роботу робота відповідно до поточних потреб виробничого процесу.

3.4.3 Візуалізація роботи SCARA-роботу

Для наочного відображення роботи SCARA-роботу в розробленому веб-інтерфейсі було створено спеціалізовану панель візуалізації. Ця панель покликана надавати користувачеві повну картину поточного стану та динаміки руху робота під час моделювання вантажно-розвантажувальних операцій.

Центральною частиною панелі є детальне графічне 2D зображення SCARA-робота. Даний блок відображає основні конструктивні елементи робота, такі як основа, плече, передпліччя та захоплюючий механізм. Окрім статичного зображення, компоненти ілюстрації пов’язані з внутрішніми параметрами моделі, що дозволяє динамічно змінювати їх положення та орієнтацію відповідно до симульованих рухів робота.

Навколо центральної ілюстрації розміщено допоміжні елементи, які надають додаткову інформацію про стан робота та його підсистем. Зокрема, у нижній частині панелі знаходяться індикатори датчиків, що візуально відображають їх поточний стан (увімкнено/вимкнено).

У нижній верхній панелі, над конвеєром, міститься температурний індикатор, який змінює колір залежно від поточного режиму роботи робота (нагрівання чи охолодження). Ця інформація допомагає користувачеві стежити за тепловим режимом системи під час моделювання навантажень.

Окрім візуальних елементів, панель також реагує на зміну внутрішніх параметрів моделі в реальному часі. Наприклад, при активації певних команд керування, відповідні компоненти SCARA-руки змінюють своє візуальне представлення, демонструючи динаміку роботи робота. Ця інтерактивність дозволяє користувачеві спостерігати за симульованими процесами у режимі «живого» відображення.

3.4.4 Приклад роботи

Для демонстрації роботи веб-інтерфейсу можна провести тестову операцію на стенді комплексу моделювання вантажно-розвантажувальних операцій на базі SCARA-роботу.

Перед початком роботи необхідно встановити з’єднання з MQTT-брокером. На головній сторінці веб-додатку розміщено панель підключення, яка дозволяє користувачеві ініціювати з’єднання (див. рис. 3.20).

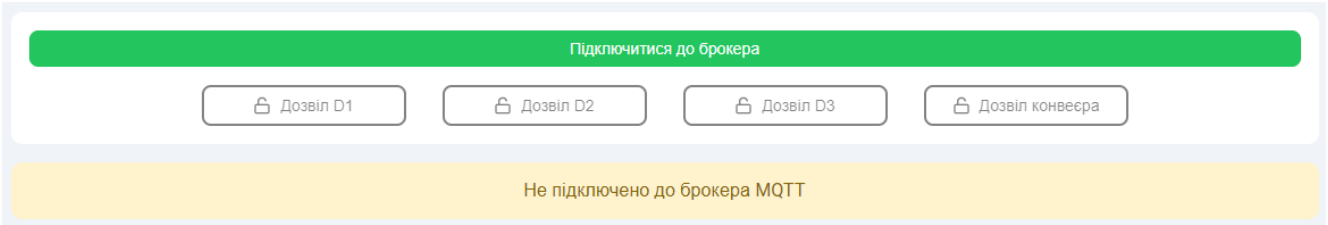


Рисунок 3.20 – Попередження користувача про статус підключення

Після того, як з’єднання було встановлено, попередження зникає та система відображає повідомлення про успішне підключення до брокеру(див. рис. 3.21).

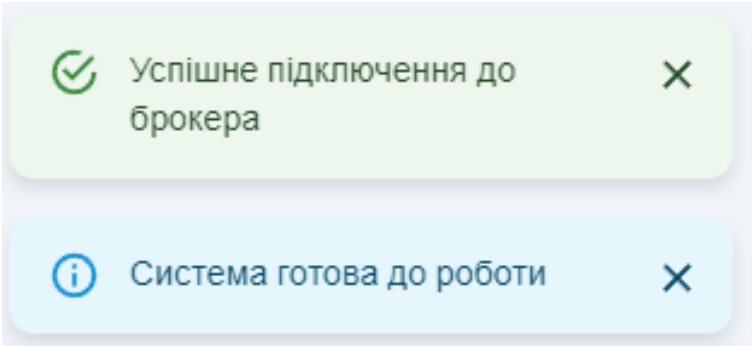


Рисунок 3.21 – Повідомлення про статус підключення

Після підключення користувач отримує доступ до всіх елементів керування стендом. Для прикладу, дамо дозвіл для повороту башти, змінимо позицію слайдера на 60 та натиснемо кнопку «Пуск» (див. рис 3.22).

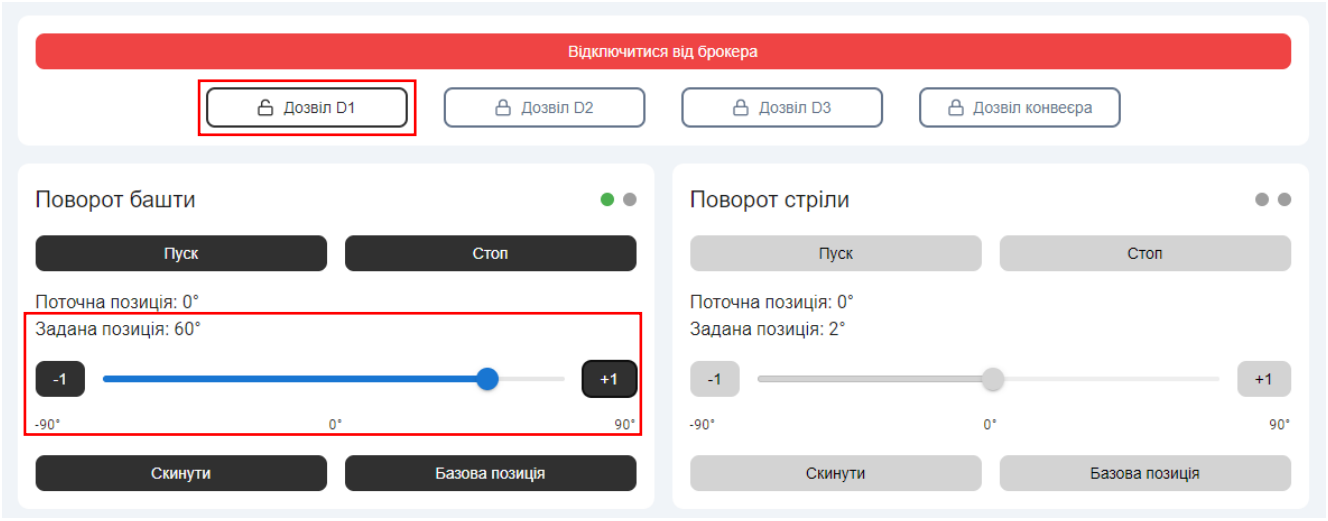
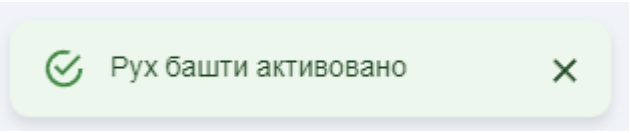


Рисунок 3.22 – Встановлення заданої позиції

Отримуємо повідомлення про успішність пуску та чекаємо закінчення операції (див. рис. 3.23)

а)



б)

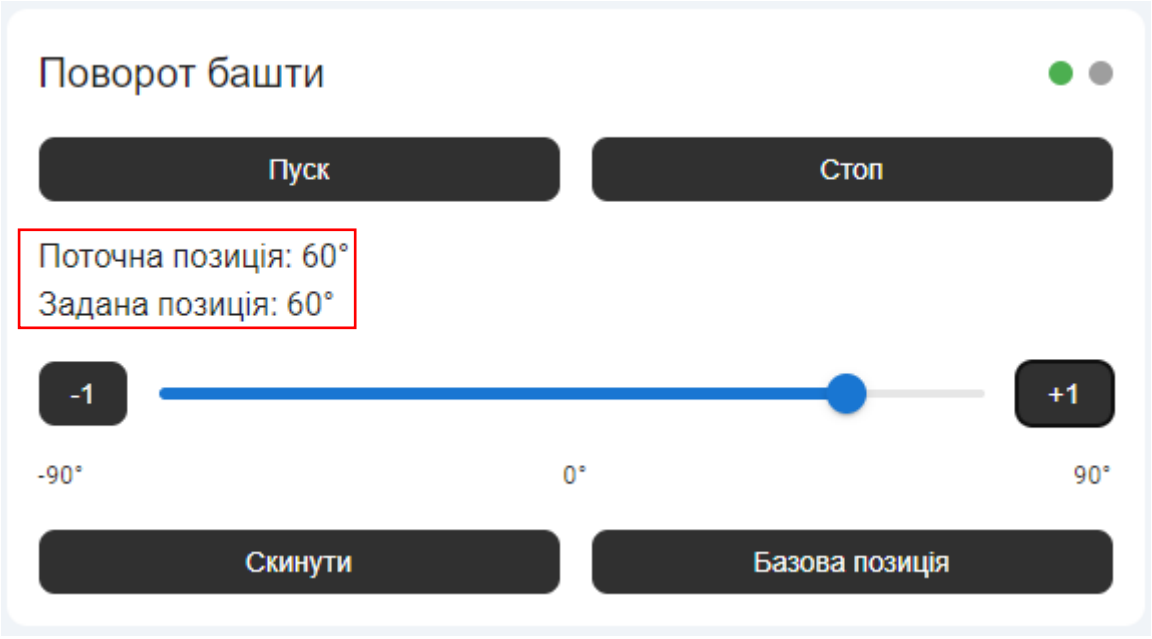


Рисунок 3.23 – а) Повідомлення про успішність операції б) Результат операції

Тепер перевіримо роботу нагрівальної камери. Встановимо температуру 20.2 градуси та активуймо режим нагріву. Чекаємо результат операції (див. рис. 3.24).

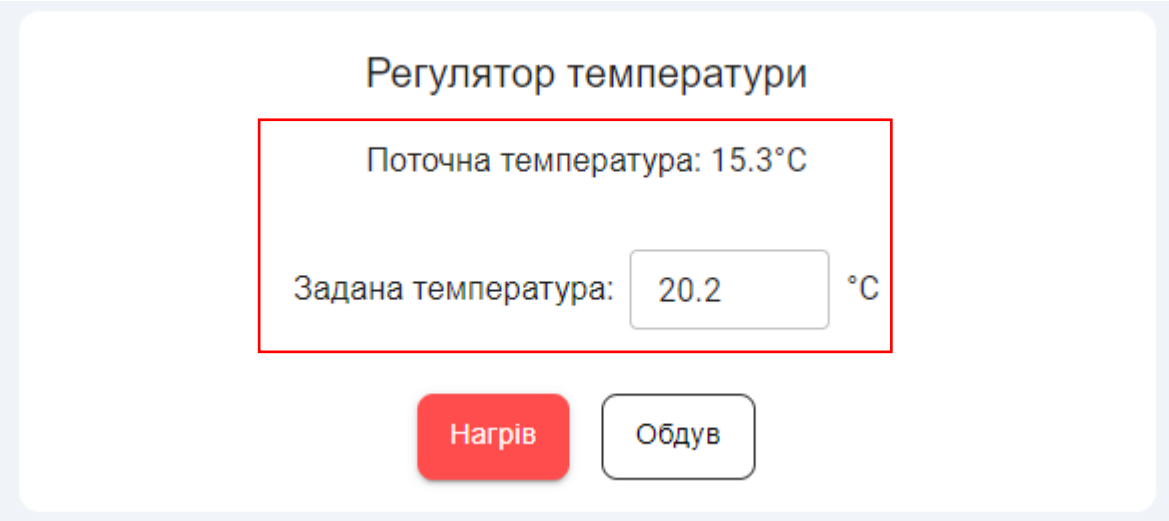


Рисунок 3.24 – Результат операції нагріву

Також можна помітити, що на візуалізації активувався індикатор нагріву (див. рис. 3.25).

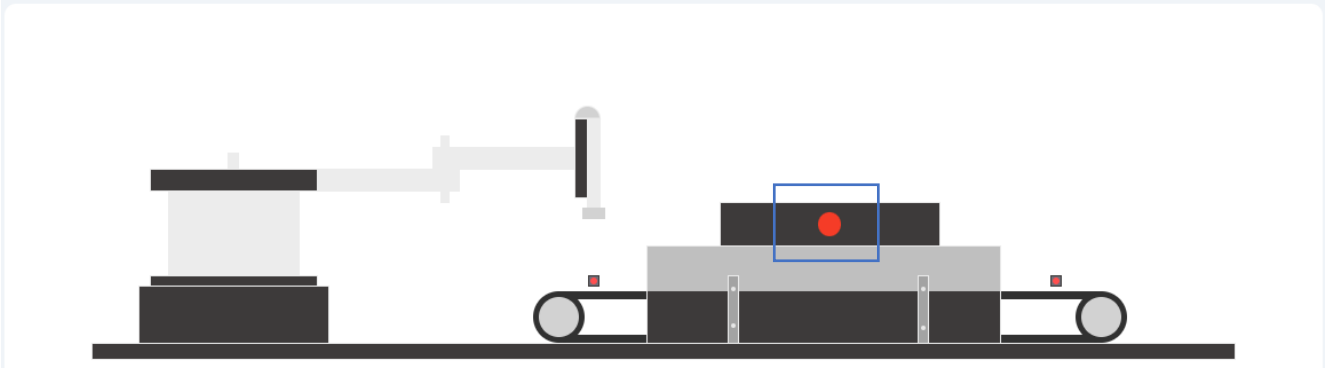


Рисунок 3.25 – Динамічна зміна стану індикаторів

Висновки до розділу:

У розділі представлено повний цикл програмної реалізації веб-інтерфейсу для автоматизованої системи управління вантажно-розвантажувальними операціями, що охоплює три ключові технологічні рівні: апаратне налаштування, інтеграційну платформу та клієнтський інтерфейс.

На рівні апаратного налаштування в TIA Portal виконано принципово важливу модернізацію контролера – оновлення версії процесора, що уможливило

підтримку протоколу OPC UA. Після цього було налаштовано вузли для передачі даних сервером.

Інтеграційна платформа Node-RED реалізована як потужний комунікаційний прошарок, що забезпечує надійний обмін даними між OPC UA сервером та MQTT-брокером. Розроблений алгоритм дозволяє мінімізувати навантаження на систему через публікацію лише змінених значень, що підвищує ефективність передачі даних.

Клієнтський веб-інтерфейс створено з використанням сучасних технологій React.js. Архітектура додатку передбачає модульність, реактивність та зручність управління, що досягається через продуману компонентну структуру та використання бібліотек Material-UI.

Розроблене рішення являє собою комплексну систему моніторингу та управління SCARA-роботом, яка забезпечує повний спектр функціональних можливостей: від встановлення з’єднання до візуалізації та керування технологічним процесом у реальному часі.

ВИСНОВКИ

Дана кваліфікаційна робота присвячена актуальному напрямку сучасної промислової автоматизації – розробці високоефективного веб-інтерфейсу для системи керування та моніторингу технологічними операціями. Проведене дослідження комплексно розкриває проблематику створення інноваційних людино-машинних інтерфейсів у контексті розвитку Промислового Інтернету Речей (IIoT) та концепції Індустрії 4.0.

У процесі роботи було здійснено науково-технічний аналіз сучасних підходів до автоматизації виробничих процесів, досліджено еволюцію систем візуалізації та особливості впровадження роботизованих комплексів у логістичні та виробничі середовища. Особливу увагу приділено розкриттю потенціалу веб-технологій як ефективного інструменту для побудови інтерактивних та функціональних промислових інтерфейсів.

Архітектура розробленої системи базується на сучасних принципах IIoT та демонструє принципово новий підхід до побудови комунікаційних систем керування. Запропоноване рішення реалізує трирівневу модель, яка включає апаратний рівень пристроїв, рівень передачі даних та рівень аналітики, що забезпечує максимальну гнучкість, масштабованість та ефективність взаємодії компонентів.

Ключовою перевагою розробленого веб-інтерфейсу є його здатність забезпечувати повноцінний моніторинг технологічного процесу в режимі реального часу, надавати інтуїтивно зрозумілі інструменти керування та миттєво реагувати на зміни стану системи. Використання сучасних технологічних рішень, таких як OPC UA, Node-RED, MQTT, React.js – дозволило створити високопродуктивний, безпечний та легко інтегрований програмний продукт.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Маринич І. А., Тронь В. В. Методичні рекомендації до виконання кваліфікаційної роботи магістра для студентів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології». Кривий Ріг : Видавничий центр КНУ, 2022. 50с.

2. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

3. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

4. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові.

5. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація)

6. ДСТУ 3651.0-97 Метрологія. Одиниці фізичних величин. Основні одиниці фізичних величин Міжнародної системи одиниць. Основні положення, назви та позначення Київ, Держстандарт України, 1998. 27 с. (Інформація та документація).

7. Скіцько В. І. Логістика в індустрії 4.0. Економіка та держава. 2016. № 4. С. 28–33.

8. Найпопулярніші рішення для автоматизації складу. *Lading*: веб-сайт. URL: https://lading.ua/news/najpopulyarnishi-rishennya-dlya-avtomatizaciyi-skladu/ (дата звернення: 05.07.24).

9. IoT technology and integration with ERP and MES systems: benefits and challenges. *Zerynth*: веб-сайт. URL: https://zerynth.com/blog/iot-technology-and-integration-with-erp-and-mes-systems-benefits-and-challenges/ (дата звернення: 06.07.24).

10. Автоматизований завод. Промислові роботи на виробництві. *ROBOTICS*: веб-сайт. URL: https://robotics.ua/avtomatyzyrovannyi-zavod.-promyshlennye-roboty-na-

proyzvodstve/?srsltid=AfmBOoqlrI1CkuYPft8X5rlyuaILhv6i6yFrg4Sd4Uvcip8In9rpPLNJ (дата звернення: 08.07.24).

11. AR/VR Take Smart Manufacturing Beyond Automation. *EETimes*: веб-сайт. URL: https://www.eetimes.com/ar-vr-take-smart-manufacturing-beyond-automation/ (дата звернення: 08.07.24).

12. On-premises or cloud-based SCADA? *WATERINDUSTRYJOURNAL*: веб-сайт. URL: https://www.waterindustryjournal.co.uk/on-premises-or-cloud-based-scada (дата звернення: 08.07.24).

13. What is a Web HMI? *MapleSystems*: веб-сайт. URL: https://maplesystems.com/what-is-web-hmi/?srsltid=AfmBOorASC0TewuyFJFojH4Yyx1rT7Jb49PhsOa1sNL7wvIhQQhCf9OQ: 09.07.24).

14. Web-Based Industrial HMIs: What's The Difference? *CONTROL*: веб-сайт. URL: https://control.com/technical-articles/web-based-industrial-hmis-whats-the-difference/ (дата звернення: 10.07.24).

15. Visualisation for visionaries with a cloud-based HMI platform.*CONTROLS, DRIVES & AUTOMATION*: веб-сайт. URL: https://www.controlsdrivesautomation.com/Rockwell-cloud-based-HMI-platform (дата звернення: 03.09.24).

16. Top HMI Design Best Practices: An Ultimate Guide. *aufaitUX*: веб-сайт. URL: https://www.aufaitux.com/blog/hmi-design-best-practices/ (дата звернення: 15.07.24).

17. How Industrial IoT Architecture Enhances Decision-Making. *ELIFTECH*: веб-сайт. URL: https://www.eliftech.com/insights/next-level-industrial-iot-architecture-upgrades-business-decision-making/ (дата звернення: 20.07.24).

18. Node-RED, the visual programming tool for Internet of Things. *PickData*: веб-сайт. URL: https://www.pickdata.net/news/node-red-visual-programming-tool-iot (дата звернення: 25.07.24).

19. The pros and cons of single page applications (SPAs). *Vention*: веб-сайт. URL: https://ventionteams.com/blog/pros-cons-of-single-page-applications (дата звернення: 30.07.24).

20. What is MVC? Advantages and Disadvantages of MVC. *interServer*: веб-сайт. URL: https://www.interserver.net/tips/kb/mvc-advantages-disadvantages-mvc/ (дата звернення: 05.08.24).

21. Efficiency Comparison: OPC-UA, Modbus, MQTT, Sparkplug, HTTP. *Iotforall*: веб-сайт. URL: https://www.iotforall.com/efficiency-comparison-opc-ua-modbus-mqtt-sparkplug-http (дата звернення: 06.08.24).

22. OPC UA Access to S7-1200 PLC via modeled OPC UA Server Interface. URL:https://cache.industry.siemens.com/dl/files/701/109781701/att_1038809/v3/109781701_S7_1200_OPC_UA_Server_DOCU_V10_en.pdf (дата звернення: 13.08.24).

23. How to Create a Node-RED Flow with Simulated OPC-UA Data. *UHM Blog*: веб-сайт. URL: https://learn.umh.app/course/creating-a-node-red-flow-with-simulated-opc-ua-data/ (дата звернення: 11.08.24).

24. АРХІТЕКТУРА ВЕБ-ОРІЄНТОВАНОЇ SCADA-СИСТЕМИ. URL: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/245568d0-75e7-4c7e-9995-4c5a1ec08612/content (дата звернення: 13.06.24).

25. Free Public MQTT Broker. *EMQX*: веб-сайт. URL: https://www.emqx.com/en/mqtt/public-mqtt5-broker (дата звернення: 20.09.24).

26. How to Use MQTT in The React Project. *EMQX*: веб-сайт. URL: https://www.emqx.com/en/blog/how-to-use-mqtt-in-react (дата звернення: 05.08.24).

27. Material UI - Overview. *Material UI*: веб-сайт. URL: https://mui.com/material-ui/getting-started/ (дата звернення: 25.08.24).

28. ДОВІДНИК З NODE-RED. URL: https://pupenasan.github.io/NodeREDGuidUKR/ (дата звернення: 13.08.24).

29. Installing the OPCUA Nodes in Node-RED. URL: https://docs.clarify.io/developers/industrial-protocols/opcua-node-red (дата звернення: 12.08.24).

30. Netlify Composable Web Platform. URL: https://www.netlify.com/ (дата звернення: 25.08.24).

ДОДАТОК А

Лістинг App.js

```
import React, { useState, useEffect } from 'react';
import { ThemeProvider, createTheme } from '@mui/material/styles';
import CssBaseline from '@mui/material/CssBaseline';
import { Box, Container } from '@mui/material';
import HMIPanel from './components/HMIPanel/HMIPanel';
import ConnectionPanel from './components/ConnectionPanel/ConnectionPanel';
import ConnectionWarning from './components/ConnectionWarning/ConnectionWarning';
import Visualization from './components/Visualization/Visualization';
import { connectToMQTT, subscribeTopic, publishToTopic, disconnectMQTT } from './services/mqttService';
import { useNotifications, NotificationStack, getNotificationMessage } from './services/notificationService';

const theme = createTheme({
  palette: {
    background: {
      default: '#f0f4f8',
    },
  },
  components: {
    MuiPaper: {
      styleOverrides: {
        root: {
          borderRadius: '10px',
          boxShadow: 'none',
```

```
      },
    },
  },
 },
});

const getInitialState = () => {
 const savedData = localStorage.getItem('scadaData');
 if (savedData) {
  try {
   return JSON.parse(savedData);
  } catch (error) {
   console.error('Помилка при парсингу даних з localStorage:', error);
  }
 }

 return {
  "Data_block_1_OnOffD1": true,
  "Data_block_1_OnOffD2": true,
  "Data_block_1_OnOffD3": true,
  "Data_block_1_move D1": false,
  "Data_block_1_move D2": false,
  "Data_block_1_move D3": false,
  "Data_block_1_stop D1": true,
  "Data_block_1_stop D2": true,
  "Data_block_1_stop D3": true,
  "Data_block_1_angle D1": 0,
  "Data_block_1_angle D2": 0,
  "Data_block_1_angle D3": 0,
  "Axis_1_ActualPosition": 0,
```

"Axis_2_ActualPosition": 0,
"Axis_3_ActualPosition": 0,
"Data_block_1_home D1": false,
"Data_block_1_home D2": false,
"Data_block_1_home D3": false,
"Data_block_1_reset D1": false,
"Data_block_1_reset D2": false,
"Data_block_1_reset D3": false,
"Axis_1_Error": false,
"Axis_2_Error": false,
"Axis_3_Error": false,
"Start_AUT": false,
"Stop_AUT": false,
"Heating": false,
"Cooling": false,
"magnit": false,
"RightSensor": false,
"LeftSensor": false,
"Switch_AUTMAN": false,
"Data_block_1_conv move": false,
"Data_block_1_conv stop": false,
"Actual_temperature": 0,
"Set_temperature": 0,
"Data_block_1_conv ONOFF": true,
"Data_block_1_conv reset": false,
"Data_block_1_conv position": 0,
"Data_block_1_conv speed": 0,
"conv_ActualPosition": 0,
"Data_block_1_conv home": false,
"conv_Error": false,

```
    "conv_ActualVelocity": 0
  };
};

const App = () => {
  const [data, setData] = useState(getInitialState());
  const [isConnected, setIsConnected] = useState(false);
  const { notifications, addNotification, removeNotification } = useNotifications();
  useEffect(() => {
    localStorage.setItem('scadaData', JSON.stringify(data));
  }, [data]);
  const handleDataMessage = (message) => {
    try {
      const newData = JSON.parse(message);
      setData(prevData => ({ ...prevData, ...newData }));
    } catch (error) {
      addNotification('Помилка обробки даних', 'error');
    }
  };
  const handleControlChange = (key, value) => {
    if (!isConnected) {
      addNotification("Спочатку встановіть з'єднання", 'warning');
      return;
    }
    const controlMessage = { [key]: value };
    setData(prevData => ({ ...prevData, ...controlMessage }));
    publishToTopic('scada/control', JSON.stringify(controlMessage));
    const notification = getNotificationMessage(key, value);
    if (notification) {
      addNotification(notification.message, notification.severity);
```

```
  }
};

const handleConnect = async () => {
  try {
    await connectToMQTT();
    setIsConnected(true);
    subscribeTopic('scada/data', handleDataMessage);
    subscribeTopic('scada/control', handleDataMessage);
    addNotification('Успішне підключення до брокера', 'success');
    addNotification('Система готова до роботи', 'info', 2000);
  } catch (error) {
    setIsConnected(false);
    addNotification('Помилка підключення до брокера', 'error');
    addNotification('Перевірте налаштування підключення', 'warning', 4000);
  }
};
const handleDisconnect = () => {
  disconnectMQTT();
  setIsConnected(false);
  addNotification("Від'єднано від брокера", 'info');
  addNotification('Роботу системи призупинено', 'warning', 4000);
};
const disabledStyles = {
  opacity: isConnected ? 1 : 0.6,
  pointerEvents: isConnected ? 'auto' : 'none',
  transition: 'opacity 0.3s ease',
};
return (
  <ThemeProvider theme={theme}>
```

```
      <CssBaseline />
      <Box sx={{ minHeight: '100vh', bgcolor: 'background.default', py: 4 }}>
        <Container maxWidth="lg">
          <ConnectionPanel
            isConnected={isConnected}
            onConnect={handleConnect}
            onDisconnect={handleDisconnect}
            data={data}
            onControlChange={handleControlChange}
          />
          <NotificationStack
            notifications={notifications}
            onClose={removeNotification}
          />
          {!isConnected && <ConnectionWarning />}
          <Box sx={disabledStyles}>
            <HMIPanel data={data} onControlChange={handleControlChange} />
          </Box>
          <Box sx={disabledStyles}>
            <Visualization data={data} />
          </Box>
        </Container>
      </Box>
    </ThemeProvider>
  );
};
export default App;
```

ДОДАТОК Б

Лістинг HMIPanel.jsx

```
import React, { useCallback, useState, useMemo } from "react";
import { Box, Typography, Grid, TextField } from "@mui/material";
import { BaseButton, CustomSwitch, TemperatureButton, CustomSlider, StyledPaper,
ControlBlock, SwitchControlContainer } from "./CustomUIComponents";

const ControlUnit = React.memo(({ title, data = {}, onControlChange }) => {
  const prefix =
    title === "Стрічковий конвеєр"
      ? "conv"
      : `D${
          ["Поворот башти", "Поворот стріли", "Підйомник"].indexOf(title) + 1
        }`;
  const isConveyor = prefix === "conv";
  const hasError = () => {
    const errorKey = isConveyor ? "conv_Error" : `Axis_${prefix.slice(1)}_Error`;
    return data[errorKey] === true;
  };
  const getOnOffStatus = () => {
    if (hasError()) return false;
    const statusKey = isConveyor
      ? "Data_block_1_conv ONOFF"
      : `Data_block_1_OnOffD${prefix.slice(1)}`;
    return data[statusKey] === true;
  };
  const isEnabled = getOnOffStatus();
  const errorDetected = hasError();
```

```
  const isDisabled = useMemo(() => !isEnabled || errorDetected, [isEnabled,
errorDetected]);

  const getRange = () => {
    if (title === "Поворот башти" || title === "Поворот стріли")
      return { min: -90, max: 90 };
    if (title === "Підйомник") return { min: -50, max: 50 };
    if (title === "Стрічковий конвеєр") return { min: -880, max: 880 };
    return { min: 0, max: 1000 };
  };
  const { min, max } = getRange();
  const unit =
    title === "Поворот башти" || title === "Поворот стріли" ? "°" : " мм";
  const currentPosition =
    data[`${isConveyor ? prefix : "Axis_" + prefix.slice(1)}_ActualPosition`] ??
    0;
  const setPosition =
    data[
      `Data_block_1_${isConveyor ? `${prefix} position` : `angle ${prefix}`}`
    ] ?? 0;
  const conveyorSpeed = data[`Data_block_1_conv speed`] ?? 25;
  const handlePositionChange = useCallback(
    (_, newPosition) => {
      onControlChange(
        `Data_block_1_${isConveyor ? `${prefix} position` : `angle ${prefix}`}`,
        newPosition
      );
    },
    [onControlChange, isConveyor, prefix]
  );
```

```
const handleSpeedChange = useCallback(
  (_, newSpeed) => {
    onControlChange("Data_block_1_conv speed", newSpeed);
  },
  [onControlChange]
);
const handleSpeedAdjustment = useCallback(
  (adjustment) => {
    const newSpeed = conveyorSpeed + adjustment;
    if (newSpeed >= 0 && newSpeed <= 50) {
      handleSpeedChange(null, newSpeed);
    }
  },
  [conveyorSpeed, handleSpeedChange]
);
const handleAdjustPosition = useCallback(
  (adjustment) => {
    const newPosition = setPosition + adjustment;
    if (newPosition >= min && newPosition <= max) {
      handlePositionChange(null, newPosition);
    }
  },
  [setPosition, min, max, handlePositionChange]
);
const handleStart = useCallback(() => {
  onControlChange(`Data_block_1_move ${prefix}`, true);
  setTimeout(() => {
    onControlChange(`Data_block_1_move ${prefix}`, false);
  }, 100);
}, [onControlChange, prefix]);
```

```
const handleStop = useCallback(() => {
  onControlChange(`Data_block_1_stop ${prefix}`, true);
  setTimeout(() => {
    onControlChange(`Data_block_1_stop ${prefix}`, false);
  }, 100);
}, [onControlChange, prefix]);
const handleReset = useCallback(() => {
  onControlChange(`Data_block_1_reset ${prefix}`, true);
  setTimeout(() => {
    onControlChange(`Data_block_1_reset ${prefix}`, false);
  }, 100);
}, [onControlChange, prefix]);
const handleHome = useCallback(() => {
  onControlChange(`Data_block_1_home ${prefix}`, true);
  setTimeout(() => {
    onControlChange(`Data_block_1_home ${prefix}`, false);
  }, 100);
}, [onControlChange, prefix]);
const showCenterLabel = true;
const labelPosition = isConveyor ? "50%" : "50%";
return (
  <StyledPaper sx={{ opacity: errorDetected ? 0.6 : 1 }}>
    <Box
      display="flex"
      justifyContent="space-between"
      alignItems="center"
      mb={2}
    >
      <Typography variant="h6">{title}</Typography>
```

```
  <Box display="flex" gap={1}>
    <Box
      width={12}
      height={12}
      borderRadius="50%"
      bgcolor={isEnabled ? "#4CAF50" : "#9e9e9e"}
    />
    <Box
      width={12}
      height={12}
      borderRadius="50%"
      bgcolor={hasError() ? "#f44336" : "#9e9e9e"}
    />
  </Box>
</Box>
<Grid container spacing={2} mb={2}>
  <Grid item xs={6}>
    <BaseButton onClick={handleStart} disabled={isDisabled}>
      Пуск
    </BaseButton>
  </Grid>
  <Grid item xs={6}>
    <BaseButton onClick={handleStop} disabled={isDisabled}>
      Стоп
    </BaseButton>
  </Grid>
</Grid>

<Typography>
  Поточна позиція: {currentPosition}
```

```
 {unit}
</Typography>
<Typography>
 Задана позиція: {setPosition}
 {unit}
</Typography>

<Box mt={2} display="flex" alignItems="center">
 <BaseButton
  onClick={() => handleAdjustPosition(isConveyor ? -10 : -1)}
  disabled={isDisabled || setPosition <= min}
  style={{ width: "auto" }}
 >
  {isConveyor ? "-10" : "-1"}
 </BaseButton>
 <CustomSlider
  min={min}
  max={max}
  step={1}
  value={setPosition}
  onChange={handlePositionChange}
  disabled={isDisabled}
  sx={{ mx: 2, flex: 1 }}
 />
 <BaseButton
  onClick={() => handleAdjustPosition(isConveyor ? 10 : 1)}
  disabled={isDisabled || setPosition >= max}
  style={{ width: "auto" }}
 >
  {isConveyor ? "+10" : "+1"}
```

```
  </BaseButton>
</Box>

<Box
  display="flex"
  justifyContent="space-between"
  mt={2}
  position="relative"
>
  <Typography variant="caption">
    {min}
    {unit}
  </Typography>
  {showCenterLabel && (
    <Typography
      variant="caption"
      sx={{
        position: "absolute",
        left: labelPosition,
        transform: "translateX(-50%)",
      }}
    >
      0{unit}
    </Typography>
  )}
  <Typography variant="caption">
    {max}
    {unit}
  </Typography>
</Box>
```

```
<Box mt={2}>
  <Grid container spacing={2}>
    <Grid item xs={6}>
      <BaseButton onClick={handleReset} disabled={isDisabled}>
        Скинути
      </BaseButton>
    </Grid>
    <Grid item xs={6}>
      <BaseButton
        onClick={handleHome} disabled={isDisabled} >

        Базова позиція
      </BaseButton>
    </Grid>
  </Grid>
</Box>

{isConveyor && (
  <>
    <Typography mt={2}>Поточна швидкість: {conveyorSpeed}</Typography>
    <Box mt={2} display="flex" alignItems="center">
      <BaseButton
        onClick={() => handleSpeedAdjustment(-1)}
        disabled={isDisabled || conveyorSpeed <= 0}
        style={{ width: "auto" }}
      >
        -1
      </BaseButton>
      <CustomSlider
```

```
    min={0}
    max={50}
    step={1}
    value={conveyorSpeed}
    onChange={handleSpeedChange}
    disabled={isDisabled}
    sx={{ mx: 2, flex: 1 }}
   />
   <BaseButton
    onClick={() => handleSpeedAdjustment(1)}
    disabled={isDisabled || conveyorSpeed >= 50}
    style={{ width: "auto" }}
   >
    +1
   </BaseButton>
  </Box>
  <Box
   display="flex"
   justifyContent="space-between"
   mt={2}
   position="relative"
  >
   <Typography variant="caption">0</Typography>
   <Typography
    variant="caption"
    sx={{
     position: "absolute",
     left: "50%",
     transform: "translateX(-50%)",
    }}
```

```
          >
            25
          </Typography>
          <Typography variant="caption">50</Typography>
        </Box>
      </>
    )}
    {title === "Підйомник" && (
      <Box mt={6.5} display="flex" alignItems="center">
        <Typography>Магніт</Typography>
        <CustomSwitch
          checked={data.magnit ?? false}
          onChange={() => onControlChange("magnit", !data.magnit)}
          leftLabel="ВИМК"
          rightLabel="УВІМК"
        />
      </Box>
    )}
  </StyledPaper>
 );
});
const TemperatureControl = React.memo(({ data = {}, onControlChange }) => {
 const [tempInput, setTempInput] = useState(
  data.Set_temperature?.toString() || ""
 );

 const handleTempChange = (event) => {
  setTempInput(event.target.value);
 };
```

```
const handleTempSubmit = (event) => {
  if (event.key === "Enter" || event.type === "blur") {
    const numValue = parseFloat(tempInput);
    if (!isNaN(numValue)) {
      onControlChange("Set_temperature", numValue);
    }
  }
};
return (
  <ControlBlock>
    <Typography variant="h6" align="center">
      Регулятор температури
    </Typography>
    <Typography align="center">
      Поточна температура: {(data.Actual_temperature ?? 0).toFixed(1)}°C
    </Typography>
    <Box mt={2} display="flex" justifyContent="center" alignItems="center">
      <Typography>Задана температура: </Typography>
      <TextField
        size="small"
        value={tempInput}
        onChange={handleTempChange}
        onKeyPress={handleTempSubmit}
        onBlur={handleTempSubmit}
        sx={{ width: "100px", ml: 1 }}
      />
      <Typography ml={1}>°C</Typography>
    </Box>
    <Box display="flex" justifyContent="center" gap={2} mt={2}>
      <TemperatureButton
```

```
    variant={data.Heating ? "contained" : "outlined"}
    onClick={() => onControlChange("Heating", !data.Heating)}
    sx={{
      backgroundColor: data.Heating ? "#ff4d4d" : "white",
      color: data.Heating ? "white" : "black",
      "&:hover": {
        backgroundColor: data.Heating
          ? "#ff3333"
          : "rgba(255, 255, 255, 0.8)",
      },
    }}
  >
    Нагрів
  </TemperatureButton>
  <TemperatureButton
    variant={data.Cooling ? "contained" : "outlined"}
    onClick={() => onControlChange("Cooling", !data.Cooling)}
    sx={{
      backgroundColor: data.Cooling ? "#4d94ff" : "white",
      color: data.Cooling ? "white" : "black",
      "&:hover": {
        backgroundColor: data.Cooling
          ? "#3385ff"
          : "rgba(255, 255, 255, 0.8)",
      },
    }}
  >
    Обдув
  </TemperatureButton>
</Box>
```

```
    </ControlBlock>
  );
});
const HMIPanel = React.memo(({ data = {}, onControlChange }) => {
  const handleControlChange = useCallback(
    (key, value) => {
      onControlChange(key, value);
    },
    [onControlChange]
  );
  return (
    <Box sx={{ background: "f0f0f0" }}>
      <Grid container spacing={2} mb={2}>
        <Grid item xs={12} md={6}>
          <ControlUnit
            title="Поворот башти"
            data={data}
            onControlChange={handleControlChange}
          />
        </Grid>
        <Grid item xs={12} md={6}>
          <ControlUnit
            title="Поворот стріли"
            data={data}
            onControlChange={handleControlChange}
          />
        </Grid>
        <Grid item xs={12} md={6}>
          <ControlUnit
            title="Підйомник"
```

```
    data={data}
    onControlChange={handleControlChange}
   />
  </Grid>
  <Grid item xs={12} md={6}>
   <ControlUnit
    title="Стрічковий конвеєр"
    data={data}
    onControlChange={handleControlChange}
   />
  </Grid>
  <Grid item xs={12} md={6}>
   <ControlBlock>
    <Typography variant="h6" align="center">
     Керування
    </Typography>
    <Box sx={{ display: "flex", flexDirection: "column", gap: 3 }}>
     <SwitchControlContainer>
      <Typography>Режим роботи</Typography>
      <Box className="switch-container">
       <CustomSwitch
        checked={data.Switch_AUTMAN}
        onChange={() =>
         handleControlChange("Switch_AUTMAN", !data.Switch_AUTMAN)
        }
        leftLabel="MAN"
        rightLabel="AUTO"
        changeColor={false}
       />
      </Box>
```

```
            </SwitchControlContainer>

            {data.Switch_AUTMAN && (
              <Box sx={{ display: "flex", gap: 2, justifyContent: "center" }}>
                <BaseButton
                  onClick={() => handleControlChange("Start_AUT", true)}
                >
                  Старт Авт.
                </BaseButton>
                <BaseButton
                  onClick={() => handleControlChange("Stop_AUT", true)}
                >
                  Стоп Авт.
                </BaseButton>
              </Box>
            )}
          </Box>
        </ControlBlock>
      </Grid>
      <Grid item xs={12} md={6}>
        <TemperatureControl
          data={data}
          onControlChange={handleControlChange}
        />
      </Grid>
    </Grid>
  </Box>
 );
});
export default HMIPanel;
```

ДОДАТОК В

Лістинг mqttService.js

```
import mqtt from 'mqtt';

const MQTT_BROKER_URL = 'wss://broker.emqx.io:8084/mqtt';
const RECONNECT_DELAY = 5000;

let client = null;
let lastSentMessages = new Map();

const logMessage = (type, message) => {
  const emoji = {
    'connect': '🟢',
    'disconnect': '🔴',
    'error': '❌',
    'reconnect': '🔄',
    'publish': '📤',
    'receive': '📥'
  };
  console.log(`${emoji[type] || ''} ${message}`);
};

export const connectToMQTT = () => {
  return new Promise((resolve, reject) => {
    if (client && client.connected) {
      resolve(client);
      return;
```

```
    }

    client = mqtt.connect(MQTT_BROKER_URL, {
      reconnectPeriod: RECONNECT_DELAY,
      connectTimeout: 10000,
    });

    client.on('connect', () => {
      logMessage('connect', 'Підключено до MQTT брокера');
      resolve(client);
    });

    client.on('error', (error) => {
      logMessage('error', `Помилка підключення до MQTT: ${error}`);
      reject(error);
    });

    client.on('close', () => {
      logMessage("disconnect', 'З'єднання з MQTT закрито");
      lastSentMessages.clear();
    });

    client.on('reconnect', () => {
      logMessage('reconnect', 'Спроба повторного підключення до MQTT брокера');
    });
  });
};

export const subscribeTopic = (topic, callback) => {
  if (!client || !client.connected) {
```

```
  logMessage('error', 'MQTT клієнт не підключений. Неможливо підписатися.');
  return;
}

client.subscribe(topic, (err) => {
  if (!err) {
    logMessage('connect', `Підписано на тему ${topic}`);
  } else {
    logMessage('error', `Помилка підписки на тему ${topic}: ${err}`);
  }
});

client.on('message', (receivedTopic, message) => {
  if (receivedTopic === topic) {
    const messageStr = message.toString();

    try {
      const receivedData = JSON.parse(messageStr);
      const now = Date.now();

      let isEcho = false;
      Object.entries(receivedData).forEach(([key, value]) => {
        const lastSent = lastSentMessages.get(key);
        if (lastSent &&
            lastSent.value === value &&
            (now - lastSent.timestamp) < 1000) {
          isEcho = true;
          logMessage('receive', `Виявлено відлуння для поля ${key}: ${value}`);
        }
      });
```

```
      if (isEcho) {
        logMessage('receive', 'Пропуск повідомлення-відлуння');
        return;
      }

      logMessage('receive', `Повідомлення від ${receivedTopic}: ${messageStr}`);
      callback(messageStr);

    } catch (error) {
      logMessage('error', `Помилка обробки повідомлення: ${error}`);
    }
  }
 });
};

export const publishToTopic = (topic, message) => {
 if (!client || !client.connected) {
  logMessage('error', 'MQTT клієнт не підключений. Неможливо опублікувати.');
  return;
 }

 try {
  const messageData = JSON.parse(message);

  Object.entries(messageData).forEach(([key, value]) => {
   lastSentMessages.set(key, {
    value: value,
    timestamp: Date.now()
   });
```

```
    });

    client.publish(topic, message, (err) => {
      if (err) {
        logMessage('error', `Помилка публікації до ${topic}: ${err}`);
      } else {
        logMessage('publish', `Опубліковано до ${topic}: ${message}`);
      }
    });
  } catch (error) {
    logMessage('error', `Невірний формат повідомлення: ${error}`);
  }
};

export const disconnectMQTT = () => {
  if (client && client.connected) {
    client.end();
    logMessage("disconnect', 'Від'єднано від MQTT брокера");
    lastSentMessages.clear();
    client = null;
  } else {
    logMessage("error', 'MQTT клієнт не підключений. Неможливо від'єднатись.");
  }
};
```