ISSN 2306-5478

УДК 004.415.2:004.5:681.5 https://doi.org/10.31713/vt220268

Заставний О.М. ([1:ORCID: 0000-0001-8630-8791])
к.т.н., старший викладач

[1]Західноукраїнський національний університет, м. Тернопіль

ТРАНСПОРТНО-НЕЗАЛЕЖНА АРХІТЕКТУРА КЕРУВАННЯ ПРОМИСЛОВИМИ ВБУДОВАНИМИ ПРИСТРОЯМИ НА ОСНОВІ МОДЕЛІ ЦИФРОВИХ ДВІЙНИКІВ

Сучасні промислові вбудовані системи еволюціонують від ізольованих контролерів, що взаємодіють із локальними системами керування через окремі промислові протоколи, до розподілених кіберфізичних систем із підтримкою кількох каналів зв'язку, віддаленого керування, дистанційного оновлення програмного забезпечення (OTA), інтеграції з хмарними сервісами та тривалої експлуатації різних поколінь обладнання. За таких умов традиційний регістрово-орієнтований підхід поступово ускладнює розвиток, масштабування та супровід програмних систем.

У роботі запропоновано транспортно-незалежну архітектуру керування промисловими вбудованими пристроями на основі моделі цифрових двійників. Запропонований підхід передбачає введення проміжного програмного рівня між фізичними пристроями та системами верхнього рівня, який забезпечує єдину модель взаємодії з обладнанням незалежно від транспортного протоколу, покоління контролера чи особливостей його програмного забезпечення. Прикладні компоненти взаємодіють із цифровими двійниками (Digital Twin), синхронізацію яких із фізичними пристроями забезпечує середовище виконання Industrial Device Runtime (IDR).

Архітектура поєднує модель цифрових двійників, драйверну транспортну абстракцію, керовану синхронізацію стану, механізм логічних знімків даних (Snapshot) і профільно-орієнтований опис обладнання. Це забезпечує ізоляцію прикладної логіки від деталей транспортних протоколів, підтримку різних поколінь контролерів, спрощує інтеграцію нового обладнання та забезпечує розвиток системи без модифікації її ядра.

На відміну від традиційних архітектур, орієнтованих переважно на передавання даних між контролерами та системами диспетчеризації, запропонована модель охоплює повний життєвий цикл промислових вбудованих пристроїв, включаючи ініціалізацію, конфігурацію, синхронізацію стану, виконання керуючих операцій і підтримку різних транспортних середовищ.

Запропонований підхід реалізовано та апробовано під час розробки й експлуатації розподіленої системи керування промисловими контролерами у складі автоматизованих комплексів самообслуговування. Практична апробація підтвердила можливість відокремлення прикладної логіки від транспортного рівня, забезпечення сумісності різних поколінь обладнання та поступового розвитку архітектури без її повного перепроєктування.

Запропонована архітектура формує основу концепції Industrial Device Runtime (IDR) та створює підґрунтя для подальшого розвитку функцій моніторингу технічного стану обладнання, підтримки багатоканального зв'язку, кластеризації Runtime-**вузлів, програмованої автоматизації та інших сервісів сучасних розподілених промислових систем.**

Ключові слова: промислові вбудовані системи; цифровий двійник; транспортно-незалежна архітектура; Industrial Device Runtime; промислова автоматизація; керування пристроями; вбудовані контролери.

Актуальність теми. Стрімкий розвиток промислової автоматизації, Інтернету речей (Industrial IoT), хмарних сервісів та засобів віддаленого керування призводить до суттєвого ускладнення сучасних промислових вбудованих систем. Якщо раніше промисловий контролер розглядався як автономний пристрій, що взаємодіє з локальною системою керування через один промисловий протокол, то сучасні виробничі комплекси характеризуються використанням кількох транспортних середовищ, необхідністю дистанційного моніторингу, підтримкою віддаленого оновлення програмного забезпечення (OTA), інтеграцією із системами верхнього рівня та тривалою експлуатацією обладнання різних поколінь [3-5,10].

Незважаючи на розвиток сучасних технологій, значна частина промислових систем і сьогодні будується за регістрово-орієнтованим принципом, коли прикладне програмне забезпечення безпосередньо

 ISSN 2306-5478

взаємодіє з картою регістрів конкретного промислового протоколу. Такий підхід є ефективним для невеликих локальних систем, однак зі збільшенням кількості пристроїв, транспортних технологій та функціональних вимог він призводить до зростання складності програмного забезпечення, ускладнює підтримку різних поколінь обладнання та обмежує можливості подальшого розвитку системи [1-3].

Особливістю промислових вбудованих систем є їх тривалий життєвий цикл [1,4,10]. На практиці програмне забезпечення повинно одночасно підтримувати контролери різних поколінь, різні транспортні протоколи, поступове оновлення обладнання та безперервну роботу діючих виробничих комплексів. За таких умов архітектура програмного забезпечення повинна забезпечувати масштабованість, розширюваність та незалежність прикладної логіки від особливостей конкретних протоколів обміну й апаратної реалізації пристроїв.

Таким чином, актуальною науково-практичною задачею є розробка архітектурних підходів, які дозволяють відокремити прикладне програмне забезпечення від транспортного рівня, забезпечити підтримку різнорідного промислового обладнання та створити єдину модель взаємодії з промисловими вбудованими пристроями незалежно від використовуваних технологій зв'язку й поколінь контролерів.

Метою роботи є розробка транспортно-незалежної архітектури керування промисловими вбудованими пристроями на основі моделі цифрових двійників, яка забезпечує ізоляцію прикладної логіки від транспортного рівня та підтримує довготривалу еволюцію промислових систем.

Основна частина.

1. **Еволюція архітектур керування промисловими вбудованими пристроями.**

Протягом тривалого часу архітектура програмного забезпечення промислових систем історично будувалася навколо транспортних протоколів обміну даними та карт регістрів контролерів. Такий підхід був цілком виправданим за умов використання окремих програмованих логічних контролерів (PLC), локальних мереж та централізованих SCADA-систем, де основним завданням було періодичне опитування пристроїв і відображення їхнього поточного стану[1-3].

У класичній моделі прикладне програмне забезпечення безпосередньо взаємодіє з картою регістрів конкретного промислового протоколу. Будь-які зміни структури регістрів, версії мікропрограмного забезпечення або моделі контролера потребують внесення змін до прикладної логіки. Зі збільшенням кількості підтримуваних пристроїв така залежність призводить до накопичення технічного боргу та ускладнює подальший розвиток системи [12,13]. На рис. 1 наведено еволюцію програмних архітектур керування промисловими вбудованими пристроями.

Рис.1. Еволюція програмних архітектур керування промисловими вбудованими пристроями

Сучасні промислові комплекси суттєво відрізняються від систем, для яких формувалися класичні підходи. Один об'єкт може одночасно використовувати кілька транспортних технологій (Ethernet, RS-485, Wi-Fi, LoRa, CAN) [4,10], підтримувати дистанційне оновлення програмного забезпечення, взаємодіяти з підлеглими пристроями, працювати у складі розподіленої системи та залишатися в експлуатації протягом багатьох років. У таких умовах транспортний протокол перестає бути основною програмною абстракцією.

У роботі пропонується перенести основний рівень абстракції від транспортних протоколів і регістрів до цифрового представлення промислового пристрою (Digital Twin) [6-11]. Кожний фізичний контролер розглядається як окрема програмна сутність, що має власний стан, набір властивостей, підтримувані функції та визначений життєвий цикл. Прикладне програмне забезпечення взаємодіє саме з цією моделлю, а всі особливості конкретного обладнання, транспортного протоколу та способу синхронізації приховуються всередині середовища виконання IDR.

 ISSN 2306-5478

Запропонована архітектура вводить проміжний програмний рівень між промисловими вбудованими пристроями та системами верхнього рівня. Цей рівень, який у подальшому позначається як Industrial Device Runtime (IDR), забезпечує підтримку цифрових двійників пристроїв, керує взаємодією з обладнанням через спеціалізовані драйвери, виконує синхронізацію стану, обробляє події та надає прикладним системам єдиний програмний інтерфейс незалежно від транспортного середовища.

Industrial Device Runtime (IDR) — транспортно-незалежне середовище виконання для керування промисловими вбудованими пристроями, побудоване на основі моделі цифрових двійників.

На відміну від традиційних шлюзів або програмних модулів доступу до промислових протоколів, IDR не обмежується передаванням даних між контролерами та системами диспетчеризації. Він забезпечує повний програмний супровід життєвого циклу промислового пристрою: виявлення обладнання, синхронізацію стану, виконання керуючих операцій, підтримку конфігурації, взаємодію з кількома транспортними каналами та інтеграцію з системами верхнього рівня.

Таким чином, запропонована архітектура переносить основний рівень взаємодії із транспортних протоколів до моделі цифрових двійників, що дозволяє ізолювати прикладне програмне забезпечення від особливостей конкретного обладнання, забезпечити масштабованість системи та створити основу для її подальшої еволюції без необхідності модифікації ядра програмного забезпечення.

2. Модель цифрового двійника промислового пристрою

Основною концепцією запропонованої архітектури є використання моделі цифрового двійника (Digital Twin)[6-11] як базової програмної сутності для взаємодії із промисловими вбудованими пристроями.

У даній роботі під цифровим двійником розуміється актуальне програмне представлення фізичного промислового пристрою, яке містить його поточний стан, конфігурацію, набір підтримуваних функцій, службову інформацію та засоби взаємодії із прикладним програмним забезпеченням. У більшості сучасних досліджень цифровий двійник розглядається як цифрове представлення фізичного об'єкта, виробничої системи або технологічного процесу [6–11].

У запропонованій роботі ця концепція адаптується до рівня окремого промислового вбудованого пристрою, де цифровий двійник використовується як основна програмна абстракція середовища виконання IDR [6-11].

Кожний фізичний пристрій має власний цифровий двійник, який підтримується середовищем виконання IDR у синхронізованому стані. Прикладні компоненти системи взаємодіють виключно із цифровим двійником, не звертаючись безпосередньо до транспортних протоколів або карт регістрів конкретного обладнання.

Запропонована модель дозволяє представити різнорідні промислові пристрої у вигляді єдиного програмного інтерфейсу незалежно від їх фізичної реалізації, виробника чи способу підключення. У результаті прикладне програмне забезпечення працює з властивостями та функціями пристрою, тоді як особливості транспортного обміну інкапсулюються всередині середовища виконання IDR.

Цифровий двійник містить не лише значення технологічних параметрів, а й службову інформацію, необхідну для підтримки повного життєвого циклу пристрою. До його складу можуть входити ідентифікаційні дані, параметри конфігурації, інформація про версію програмного забезпечення, характеристики підтримуваних функцій, поточний стан зв'язку, результати діагностики та інші атрибути, необхідні для керування пристроєм.

Таким чином, цифровий двійник стає основною програмною сутністю середовища виконання IDR, тоді як фізичний контролер розглядається лише як джерело актуального стану та виконавець команд керування. Такий підхід забезпечує незалежність прикладної логіки від особливостей конкретного обладнання та створює основу для побудови транспортно-незалежної архітектури керування промисловими вбудованими пристроями.

Тобто таким чином можна сформулювати наступне визначення: Industrial Device Digital Twin — це синхронізоване програмне представлення промислового вбудованого пристрою, яке містить його поточний стан, конфігурацію, функціональні можливості та програмний інтерфейс взаємодії, незалежний від транспортного протоколу та апаратної реалізації.

3. Архітектура Industrial Device Runtime

Запропонована транспортно-незалежна архітектура реалізує принцип багаторівневої взаємодії між прикладними системами,

 ISSN 2306-5478

середовищем виконання та фізичними промисловими пристроями. Основною ідеєю є відокремлення прикладної логіки від транспортного рівня шляхом введення спеціалізованого програмного середовища Industrial Device Runtime (IDR).

На відміну від традиційних архітектур, у яких прикладні модулі безпосередньо працюють із транспортними протоколами та картами регістрів, у запропонованій моделі всі операції виконуються через цифрові двійники пристроїв. Середовище виконання IDR підтримує їх актуальний стан, забезпечує взаємодію з фізичним обладнанням та приховує особливості конкретних транспортних технологій.

На відміну від існуючих архітектур, запропонований підхід розглядає цифровий двійник не лише як засіб моделювання фізичного об'єкта, а як базову програмну абстракцію транспортно-незалежного середовища керування промисловими вбудованими пристроями.

Концептуальну структуру запропонованої архітектури наведено на рис. 2.

Рис.2. Загальна архітектура IDR

Архітектура складається з п'яти основних рівнів. Верхній рівень представлено промисловою платформою керування (SCADA, HMI, Web, Mobile, Cloud), яка є системою верхнього рівня та взаємодіє із середовищем виконання Industrial Device Runtime (IDR) через програмний інтерфейс (API). Середовище виконання включає цифрові двійники, програмний інтерфейс драйверів, систему синхронізації цифрових двійників, систему подій, планувальник задач і сервіси пристроїв. Наступний рівень утворюють драйвери промислових пристроїв, які реалізують єдиний програмний інтерфейс та інкапсулюють особливості конкретного обладнання. Нижче розташовано транспортний рівень, який забезпечує обмін

даними між драйверами та фізичними пристроями за допомогою підтримуваних протоколів і інтерфейсів, зокрема Modbus RTU, Modbus TCP, MQTT, CAN, BLE, Ethernet та інших. Нижній рівень архітектури представлено промисловими вбудованими пристроями, які є джерелом даних і безпосередніми виконавцями команд керування.

Порівняння класичних архітектур та запропонованої наведено на рис.3.

У запропонованій архітектурі системи верхнього рівня взаємодіють виключно із середовищем виконання IDR, не використовуючи безпосередній доступ до транспортних протоколів або карт регістрів промислових пристроїв.

а) б)

Рис.3. Порівняння класичної архітектури (а) та запропонованої архітектури (б)

Такий підхід забезпечує незалежність прикладного програмного забезпечення від конкретних способів фізичного підключення обладнання та дозволяє використовувати єдиний програмний інтерфейс для різнорідних промислових систем.

На рис.4 наведено життєвий цикл цифрового двійника.

Рис.4. Життєвий цикл синхронізації цифрового двійника

 ISSN 2306-5478

Кожний фізичний пристрій представлений у Runtime окремим цифровим двійником. Синхронізація його стану здійснюється спеціалізованими драйверами, які реалізують взаємодію з відповідними транспортними протоколами. При цьому Runtime не містить логіки, пов'язаної з конкретними транспортами, а працює виключно з уніфікованими інтерфейсами драйверів.

Важливою особливістю запропонованої архітектури є чітке розділення відповідальності між окремими рівнями системи. Транспортний рівень забезпечує фізичний обмін даними, драйвери реалізують особливості взаємодії з конкретними типами обладнання, Runtime підтримує цифрові двійники та координує їх життєвий цикл, тоді як прикладні системи використовують лише високорівневі програмні інтерфейси без необхідності врахування особливостей фізичних пристроїв.

Така організація дозволяє незалежно розвивати окремі компоненти архітектури. Додавання нового транспортного протоколу, підтримка нового покоління контролерів або інтеграція додаткового обладнання не потребують модифікації прикладної логіки, оскільки всі зміни локалізуються на рівні відповідних драйверів та моделей цифрових двійників.

Порівняння запропонованого підходу та традиційного наведено в табл.1.

Табл.1.

Порівняння традиційної та запропонованої архітектур

Традиційний підхід	Запропонований підхід
Регістр	Властивість цифрового двійника
Карта регістрів	Модель пристрою
Транспортний протокол	Програмний інтерфейс драйвера
Опитування (Polling)	Керована синхронізація
Дані	Події
Контролер	Цифровий двійник

Таким чином, Industrial Device Runtime виконує роль проміжного архітектурного шару між промисловими вбудованими пристроями та системами верхнього рівня, забезпечуючи єдину модель взаємодії з обладнанням незалежно від його фізичної реалізації, транспортного протоколу чи покоління апаратного забезпечення.

3.1. Драйверна модель взаємодії з обладнанням

Однією з основних вимог до запропонованої архітектури є незалежність прикладного програмного забезпечення від транспортних протоколів[13,14] та особливостей конкретних моделей промислового обладнання. Для досягнення цієї мети в Industrial Device Runtime використовується драйверна модель взаємодії з пристроями, відповідно до архітектури, наведеної на рис. 2. Дана модель наведена на рис. 5.

Рис. 5. Модель взаємодії драйвера пристрою

У запропонованій архітектурі кожен тип промислового обладнання описується спеціалізованим драйвером, який інкапсулює всі особливості взаємодії з фізичним пристроєм. Драйвер реалізує єдиний програмний інтерфейс незалежно від використовуваного транспортного протоколу або структури внутрішніх регістрів контролера.

Таким чином, середовище виконання IDR не взаємодіє безпосередньо із протоколами Modbus, MQTT, CAN або іншими транспортними технологіями. Воно використовує лише уніфікований інтерфейс драйвера, який забезпечує отримання та зміну стану цифрового двійника, виконання керуючих операцій та доступ до службових функцій пристрою.

Такий підхід дозволяє локалізувати всі зміни, пов'язані з особливостями конкретного обладнання, всередині відповідного драйвера. У разі появи нового покоління контролерів або зміни транспортного протоколу модифікації потребує лише драйвер відповідного пристрою, тоді як Runtime та прикладне програмне забезпечення залишаються незмінними.

Запропонована драйверна модель також забезпечує можливість одночасної підтримки різних поколінь обладнання. Якщо нова версія контролера має іншу карту регістрів або розширений

 ISSN 2306-5478

набір функцій, відповідні зміни реалізуються лише у драйвері нового покоління без необхідності модифікації інших компонентів системи.

Таким чином, драйвер виконує роль адаптера між фізичним обладнанням та середовище виконання IDR, забезпечуючи транспортну незалежність, ізоляцію прикладної логіки від особливостей реалізації пристроїв та можливість поступової еволюції промислової системи.

3.2. Синхронізація цифрових двійників

Однією з ключових задач середовища виконання IDR є підтримка актуального стану цифрових двійників промислових пристроїв. На відміну від традиційного підходу, який базується на періодичному опитуванні всіх регістрів обладнання, у запропонованій архітектурі використовується керована модель синхронізації, орієнтована на властивості та життєвий цикл цифрового двійника.

Основною одиницею синхронізації є не окремий регістр промислового протоколу, а логічний знімок стану (Snapshot), яка описує певний аспект стану пристрою. Наприклад, окремі Snapshot можуть містити ідентифікаційні дані пристрою, параметри конфігурації, технологічний стан, статистичну інформацію або службові характеристики обладнання.

Такий підхід дозволяє виконувати синхронізацію лише тих даних, які є необхідними у конкретний момент часу, замість постійного опитування всього доступного адресного простору контролера.

У запропонованій архітектурі використовуються кілька режимів синхронізації цифрових двійників.

Початкова синхронізація виконується після підключення нового пристрою або відновлення зв'язку. Її метою є отримання основної інформації про контролер, формування цифрового двійника та визначення підтримуваних функціональних можливостей.

Подієва синхронізація використовується після виконання керуючих операцій або виникнення подій, які потенційно змінюють стан пристрою. У цьому випадку Runtime оновлює лише ті групи даних, які можуть бути змінені внаслідок виконаної операції.

Фонова синхронізація забезпечує періодичне оновлення службової інформації та контроль працездатності обладнання. Вона виконується зі значно меншою частотою та використовується переважно для виявлення змін, які виникли поза межами Runtime,

наприклад у результаті локального керування пристроєм або зміни його конфігурації.

Ручна синхронізація ініціюється оператором або системою верхнього рівня у випадках, коли необхідно примусово оновити певну частину цифрового двійника незалежно від поточного режиму роботи Runtime.

Розділення процесу синхронізації на окремі логічні режими дозволяє істотно зменшити обсяг службового мережевого трафіку, скоротити кількість звернень до промислових контролерів та забезпечити більш ефективне використання транспортних каналів. Особливо важливим це є для систем із великою кількістю пристроїв або транспортних середовищ із обмеженою пропускною здатністю.

Таким чином, синхронізація цифрових двійників розглядається не як безперервне опитування обладнання, а як керований процес підтримки актуального стану Runtime-моделі, що враховує поточний режим роботи системи, характер виконуваних операцій та особливості конкретного промислового обладнання.

3.3. Подійна модель функціонування середовища виконання IDR

У традиційних промислових системах більшість прикладної логіки безпосередньо пов'язана з процесом періодичного опитування обладнання. Після кожного циклу опитування виконується аналіз отриманих значень, виявлення змін та прийняття відповідних керуючих рішень. Такий підхід призводить до тісного поєднання механізмів обміну даними із прикладною логікою системи.

У запропонованій архітектурі ці процеси розділено. Середовище виконання IDR самостійно підтримує актуальний стан цифрових двійників, тоді як прикладні компоненти реагують виключно на зміни їхнього стану.

Будь-яка зміна властивостей цифрового двійника розглядається як подія, яка може бути використана іншими компонентами системи. Джерелами подій можуть бути результати синхронізації із фізичним обладнанням, виконання керуючих операцій, зміни конфігурації пристрою або зовнішні команди систем верхнього рівня.

Після оновлення цифрового двійника IDR автоматично формує відповідні повідомлення про зміну стану та передає їх компонентам, які використовують ці дані. Таким чином прикладне програмне

 ISSN 2306-5478

забезпечення не виконує самостійного опитування обладнання, а працює із вже синхронізованими даними середовища виконання IDR.

Подійна модель дозволяє відокремити процес отримання інформації від процесу її використання. Це забезпечує незалежний розвиток окремих компонентів системи, спрощує масштабування програмного забезпечення та дозволяє інтегрувати нові сервіси без модифікації існуючої логіки взаємодії з обладнанням.

Важливою особливістю запропонованого підходу є те, що джерелом подій виступає не транспортний протокол, а цифровий двійник пристрою. Незалежно від способу фізичного підключення обладнання всі компоненти Runtime працюють із єдиною моделлю стану, що забезпечує транспортну незалежність прикладного програмного забезпечення.

Таким чином, подійна модель є природним продовженням концепції цифрових двійників і дозволяє розглядати Runtime не як систему опитування обладнання, а як середовище підтримки актуального стану промислових пристроїв та координації взаємодії між компонентами програмної системи.

Висновки

У роботі проаналізовано сучасні тенденції розвитку промислових вбудованих систем та показано, що традиційний регістрово-орієнтований підхід до взаємодії з промисловими контролерами поступово втрачає ефективність у розподілених системах із тривалим життєвим циклом. Зростання кількості транспортних протоколів, необхідність підтримки різних поколінь обладнання, інтеграція з хмарними сервісами та вимоги до безперервної модернізації програмного забезпечення потребують переходу до нового рівня програмної абстракції.

У статті запропоновано транспортно-незалежну архітектуру керування промисловими вбудованими пристроями на основі моделі цифрових двійників. На відміну від традиційних підходів, запропонована архітектура вводить проміжний програмний рівень — Industrial Device Runtime (IDR), який забезпечує підтримку цифрових двійників пристроїв, ізоляцію прикладної логіки від транспортних протоколів, синхронізацію стану обладнання та уніфікований програмний інтерфейс взаємодії із різнорідними промисловими контролерами.

Основними результатами роботи є формалізація архітектурної моделі Industrial Device Runtime, запропонування драйверної моделі

інтеграції обладнання, моделі керованої синхронізації цифрових двійників та подійної моделі взаємодії компонентів середовища виконання IDR. Запропонований підхід дозволяє відокремити прикладне програмне забезпечення від деталей транспортного рівня, спростити підтримку різних поколінь обладнання та створити основу для поступового розвитку промислових систем без необхідності перепроєктування їх програмної архітектури.

Практична реалізація запропонованої архітектури виконана під час розробки системи керування промисловими вбудованими контролерами, що підтвердила можливість її використання в умовах реальної експлуатації та продемонструвала доцільність застосування моделі цифрових двійників як базової програмної абстракції для сучасних промислових систем.

Таким чином, запропонована архітектура може розглядатися як основа для побудови нового покоління транспортно-незалежних програмних платформ керування промисловими вбудованими пристроями. Перспективними напрямами подальших досліджень є інтеграція моделей оцінювання технічного стану обладнання (Health Model), підтримка багатоканального зв'язку, побудова кластерних середовищ виконання із резервуванням, розробка засобів автоматизації технологічних процесів та дослідження методів прогнозування деградації промислових систем на основі накопичених експлуатаційних даних.

1. IEC 62264-1:2013. Enterprise-control system integration – Part 1: Models and terminology. Geneva : International Electrotechnical Commission, 2013. 116 p. 2. IEC 61131-3:2013. Programmable Controllers – Part 3: Programming Languages. Geneva : International Electrotechnical Commission, 2013. 3. OPC Foundation. OPC Unified Architecture Specification. Version 1.05. Scottsdale : OPC Foundation, 2023. 4. Kagermann H., Wahlster W., Helbig J. Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0. Frankfurt am Main : acatech – National Academy of Science and Engineering, 2013. 5. Hermann M., Pentek T., Otto B. Design Principles for Industrie 4.0 Scenarios // Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS). 2016. P. 3928–3937. 6. Grieves M., Vickers J. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems // Transdisciplinary Perspectives on Complex Systems. Cham : Springer, 2017. P. 85–113. 7. Tao F., Zhang M., Liu Y., Nee A. Y. C. Digital Twin Driven Smart Manufacturing // Journal of Manufacturing Systems. 2019. Vol. 48. P. 157–169. 8. Fuller A., Fan Z., Day C., Barlow C. Digital Twin: Enabling

 ISSN 2306-5478

Technologies, Challenges and Open Research // IEEE Access. 2020. Vol. 8. P. 108952–108971. 9. Negri E., Fumagalli L., Macchi M. A Review of the Roles of Digital Twin in CPS-based Production Systems // Procedia Manufacturing. 2017. Vol. 11. P. 939–948. 10. Industrial Digital Twin Association (IDTA). Asset Administration Shell: Reading Guide. Version 3.0. Frankfurt am Main : IDTA, 2023. 11. Digital Twin Consortium. Digital Twin Capabilities Periodic Table. Needham : Digital Twin Consortium, 2024. 12. Industrial Internet Consortium. Industrial Internet Reference Architecture (IIRA). Version 1.10. Needham : Industrial Internet Consortium, 2023.13. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Boston : Addison-Wesley, 2021. 14. Buschmann F., Henney K., Schmidt D. C. Pattern-Oriented Software Architecture. Volume 1: A System of Patterns. Chichester : John Wiley & Sons, 1996.

REFERENCES:

1. IEC 62264-1:2013. Enterprise-control system integration – Part 1: Models and terminology. Geneva : International Electrotechnical Commission, 2013. 116 p. 2. IEC 61131-3:2013. Programmable Controllers – Part 3: Programming Languages. Geneva : International Electrotechnical Commission, 2013. 3. OPC Foundation. OPC Unified Architecture Specification. Version 1.05. Scottsdale : OPC Foundation, 2023. 4. Kagermann H., Wahlster W., Helbig J. Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0. Frankfurt am Main : acatech – National Academy of Science and Engineering, 2013. 5. Hermann M., Pentek T., Otto B. Design Principles for Industrie 4.0 Scenarios // Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS). 2016. P. 3928–3937. 6. Grieves M., Vickers J. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems // Transdisciplinary Perspectives on Complex Systems. Cham : Springer, 2017. P. 85–113. 7. Tao F., Zhang M., Liu Y., Nee A. Y. C. Digital Twin Driven Smart Manufacturing // Journal of Manufacturing Systems. 2019. Vol. 48. P. 157–169. 8. Fuller A., Fan Z., Day C., Barlow C. Digital Twin: Enabling Technologies, Challenges and Open Research // IEEE Access. 2020. Vol. 8. P. 108952–108971. 9. Negri E., Fumagalli L., Macchi M. A Review of the Roles of Digital Twin in CPS-based Production Systems // Procedia Manufacturing. 2017. Vol. 11. P. 939–948. 10. Industrial Digital Twin Association (IDTA). Asset Administration Shell: Reading Guide. Version 3.0. Frankfurt am Main : IDTA, 2023. 11. Digital Twin Consortium. Digital Twin Capabilities Periodic Table. Needham : Digital Twin Consortium, 2024. 12. Industrial Internet Consortium. Industrial Internet Reference Architecture (IIRA). Version 1.10. Needham : Industrial Internet Consortium, 2023.13. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Boston : Addison-Wesley, 2021. 14. Buschmann F., Henney K., Schmidt D. C. Pattern-Oriented Software

Architecture. Volume 1: A System of Patterns. Chichester : John Wiley & Sons, 1996.

Zastavnyy **O.**M. ([1:ORCID: 0000-0001-8630-8791])
PhD, Senior Lecturer

[1] West Ukrainian National University

TRANSPORT-INDEPENDENT ARCHITECTURE FOR INDUSTRIAL EMBEDDED DEVICE CONTROL BASED ON THE DIGITAL TWIN MODEL

Modern industrial embedded systems are evolving from isolated controllers communicating with local control systems through dedicated industrial protocols into distributed cyber-physical systems that support multiple communication channels, remote control, over-the-air (OTA) software updates, cloud integration, and long-term operation of multiple generations of industrial equipment. Under these conditions, the traditional register-oriented approach, in which application software directly interacts with protocol register maps, gradually complicates software development, scalability, and long-term system maintenance.

This paper proposes a transport-independent architecture for controlling industrial embedded devices based on the Digital Twin model. The proposed approach introduces an intermediate software layer between physical industrial devices and upper-level management systems, providing a unified interaction model regardless of the underlying transport protocol, controller generation, or firmware implementation. Instead of communicating directly with physical devices or protocol registers, application components interact with Digital Twins synchronized with physical equipment by the Industrial Device Runtime (IDR) execution environment.

The proposed architecture combines the Digital Twin model, a driver-based transport abstraction, controlled state synchronization, a logical data snapshot mechanism (Snapshot), and profile-oriented device descriptions. This approach isolates application logic from transport protocol implementation details, enables simultaneous support for multiple generations of industrial controllers, simplifies

ISSN 2306-5478

the integration of new equipment, and facilitates system evolution without modifications to the core architecture.

Unlike conventional architectures primarily focused on data exchange between controllers and supervisory systems, the proposed model covers the complete life cycle of industrial embedded devices, including initialization, configuration, state synchronization, execution of control operations, and support for multiple transport environments.

The proposed approach has been implemented and validated during the development and operation of a distributed industrial controller management system deployed in self-service automation complexes. Practical evaluation confirmed the feasibility of separating application logic from the transport layer, ensuring compatibility between different generations of industrial equipment, and supporting the gradual evolution of the system architecture without complete redesign.

The proposed architecture establishes the foundation of the Industrial Device Runtime (IDR) concept and provides a basis for further development of equipment health monitoring, multi-channel communication support, Runtime node clustering, programmable automation, and other services required by modern distributed industrial systems.

Keywords: industrial embedded systems; digital twin; transport-independent architecture; Industrial Device Runtime; industrial automation; device management; embedded controllers.

Отримано: 30 беерзня 2026 року
Прорецензовано: 27 квітня 2026 року
Прийнято до друку: 29 травня 2026 року